BUKU AJAR

ANALISIS & PERANCANGAN SISTEM INFORMASI

Ikhwan Arief
Asmuliardi Muluk

BUKU AJAR
ANALISIS & PERANCANGAN SISTEM INFORMASI

Penulis

Ikhwan Arief
Asmuliardi Muluk

Anggota APPTI Nomor : 002.018.1.09.2017
Anggota IKAPI Nomor : 218/Anggota Luar Biasa/JTI/2019

Diterbitkan oleh
UMSIDA PRESS
Jl. Mojopahit 666 B Sidoarjo
ISBN: 978-623-464-137-0

Buku Ajar Analisis & Perancangan Sistem Informasi
Penulis: Ikhwan Arief & Asmuliardi Muluk
ISBN: 978-623-464-137-0
Editor: M. Tanzil Multazam & Mahardika Darmawan K.W.
Copy Editor: Wiwit Wahyu Wijayanti
Design Sampul dan Tata Letak: Wiwit Wahyu Wijayanti
Penerbit: UMSIDA Press
Redaksi: Universitas Muhammadiyah Sidoarjo Jl. Mojopahit No.666B Sidoarjo, Jawa Timur

Cetakan Pertama, Januari 2026

Kata Pengantar

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga buku ajar **Analisis dan Perancangan Sistem Informasi** ini dapat terselesaikan dengan baik. Buku ini disusun sebagai panduan bagi para mahasiswa, khususnya di lingkungan Departemen Teknik Industri, yang sedang menempuh perjalanan akademis untuk memahami bagaimana sistem informasi menjadi tulang punggung operasi industri modern.

Di tengah era transformasi digital dan Revolusi Industri 4.0, kemampuan untuk menganalisis, merancang, dan mengimplementasi-kan sistem informasi yang efektif dan efisien bukan lagi sekedar keahlian tambahan, melainkan sebuah kompetensi inti. Seorang insinyur industri dituntut untuk mampu merancang sistem terintegrasi yang melibatkan manusia, material, mesin, metode, dan tentu saja, informasi. Buku ini hadir untuk menjembatani antara konsep teoretis dan aplikasi praktis dalam dunia perancangan sistem informasi. Buku ini dirancang untuk menjadi sahabat dalam memahami setiap tahapan dalam siklus hidup pengembangan sistem.

Perjalanan akan dimulai dengan **Bab 1: Pendahuluan**, yang akan meletakkan pondasi pemahaman tentang apa itu sistem informasi dan mengapa analisis serta perancangannya begitu krusial. Kemudian dilanjutkan ke **Bab 2: Konsep Dasar Sistem** untuk menyamakan persepsi mengenai elemen-elemen dan karakteristik yang membentuk sebuah sistem.

Selanjutnya, pembaca akan mendalami berbagai pendekatan dalam pengembangan sistem pada **Bab 3: Metodologi Pengembangan Sistem**, mulai dari metode klasik seperti

Waterfall hingga metode adaptif seperti *Agile*. Setelah memahami metodologinya, pembaca akan masuk ke jantung dari proses analisis, yaitu **Bab 4: Pengumpulan Kebutuhan** dan **Bab 5: Analisis Kebutuhan**. Di sini, pembaca akan belajar teknik-teknik praktis untuk menggali informasi dan memodelkannya ke dalam bentuk diagram seperti DFD dan ERD.

Dari hasil analisis, pembaca akan melangkah ke tahap kreatif pada **Bab 6: Desain Sistem**, di mana pembaca akan belajar merancang antarmuka, basis data, arsitektur, hingga proses bisnis. Puncak dari proses ini adalah **Bab 7: Implementasi Sistem** dan **Bab 8: Pengujian Sistem**, yang akan membekali pembaca dengan pengetahuan untuk mewujudkan desain menjadi sistem yang fungsional dan andal.

Sebagai pelengkap, buku ini ditutup dengan **Bab 9: Dokumentasi Sistem** yang seringkali krusial namun terabaikan, serta **Bab 10: Etika dan Profesionalisme** untuk memastikan bahwa setiap sistem yang dibangun tidak hanya canggih secara teknis, tetapi juga bertanggung jawab secara etis. Untuk mengasah kemampuan pemecahan masalah, sebuah **Suplemen** berisi lima studi kasus komprehensif beserta solusinya telah disiapkan untuk menantang pembaca menerapkan seluruh pengetahuan yang telah dipelajari.

Setiap bab dirancang dengan struktur yang konsisten, meliputi Capaian Pembelajaran, isi bab yang mendalam, contoh soal, studi kasus relevan, serta kunci jawaban dan solusi untuk memfasilitasi proses belajar mandiri.

Ucapan terima kasih penulis sampaikan kepada seluruh pihak yang telah memberikan dukungan, baik moril maupun materil. Akhir kata, tiada gading yang tak retak. Buku ini tentu masih jauh dari sempurna. Kritik dan saran yang membangun sangat

diharapkan untuk perbaikan di masa mendatang. Semoga buku ini dapat memberikan manfaat yang sebesar-besarnya.

Selamat belajar, selamat berkarya, dan selamat menjadi agen perubahan di dunia industri.

Salam hormat,
Penulis
2026

Daftar Isi

Bab 1. Pendahuluan

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Mendefinisikan dan membedakan konsep data, informasi, sistem, dan sistem informasi secara komprehensif, dengan mengacu pada pandangan para ahli.
- Mengidentifikasi dan menjelaskan lima komponen fundamental dari sebuah sistem informasi (Perangkat Keras, Perangkat Lunak, Data, Manusia, Proses) beserta peran jaringan.
- Menarasikan evolusi sistem informasi melalui empat era teknologi utama, dari mainframe hingga era kecerdasan buatan (AI) saat ini.
- Menganalisis peran strategis sistem informasi dalam mendukung efisiensi operasional, pengambilan keputusan, dan keunggulan kompetitif organisasi.
- Menjelaskan urgensi dan pentingnya disiplin analisis dan perancangan sistem dengan menggunakan data statistik mengenai tingkat kegagalan proyek Teknologi Informasi (TI).
- Memahami peran, tanggung jawab, dan keahlian yang dibutuhkan oleh seorang analis sistem profesional.

Selamat datang di dunia Analisis dan Perancangan Sistem Informasi (APSI). Anda sedang memulai sebuah perjalanan intelektual yang akan membekali Anda dengan seperangkat keahlian yang paling dicari di era digital. Dalam dunia bisnis yang bergerak cepat dan semakin kompleks, kemampuan untuk memahami masalah, merancang solusi berbasis teknologi, dan mengelola implementasinya secara efektif bukanlah sekedar keahlian teknis, melainkan sebuah kompetensi strategis yang fundamental.

Mata kuliah ini akan membawa Anda dari konsep dasar hingga teknik-teknik canggih yang digunakan oleh para profesional untuk membangun sistem informasi yang andal, efisien, dan mampu memberikan nilai nyata bagi organisasi. Kita tidak hanya akan belajar tentang diagram dan metodologi, tetapi juga tentang bagaimana berpikir secara sistematis, berkomunikasi secara efektif dengan berbagai pemangku kepentingan, dan pada akhirnya, menjadi arsitek solusi digital yang mampu menjembatani kesenjangan antara kebutuhan bisnis dan kapabilitas teknologi. Bab pendahuluan ini akan meletakkan pondasi bagi seluruh perjalanan kita. Kita akan mulai dengan mendefinisikan blok bangunan paling dasar, menelusuri jejak evolusi sistem informasi, memahami peran vitalnya dalam organisasi modern, dan menghadapi realitas mengapa begitu banyak proyek teknologi yang gagal, sebuah kenyataan yang menggarisbawahi betapa pentingnya disiplin ilmu yang akan kita pelajari ini.

1.1 Dasar Konseptual: Dari Data ke Sistem Informasi

Untuk memahami dunia sistem informasi, kita harus terlebih dahulu menguasai kosakata dasarnya. Empat konsep; data,

informasi, sistem, dan sistem informasi, merupakan pilar-pilar yang menopang seluruh disiplin ilmu ini. Memahami perbedaan dan hubungan di antara keempatnya adalah langkah pertama yang krusial.

Definisi Data, Informasi, dan Sistem

Data adalah representasi mentah dari fakta, peristiwa, atau transaksi yang belum diolah. Data dapat berupa angka, teks, gambar, atau sinyal yang, dalam bentuk aslinya, mungkin tidak memiliki makna atau konteks yang jelas bagi penerimanya. Pakar mendefinisikannya sebagai "fakta mentah yang belum diolah". Contoh data adalah angka penjualan harian sebuah produk (misalnya, 357 unit), daftar nama karyawan, atau rekaman suhu dari sebuah sensor. Data adalah bahan baku fundamental yang akan diolah oleh sistem informasi.

Informasi, di sisi lain, adalah hasil dari pengolahan data. Informasi adalah data yang telah diorganisir, diberi konteks, dan disajikan dalam bentuk yang bermakna sehingga menjadi berguna bagi penerimanya, terutama untuk tujuan pengambilan keputusan. Proses transformasi dari data menjadi informasi adalah proses penciptaan nilai. Misalnya, data penjualan harian (357 unit) menjadi informasi ketika diolah menjadi laporan "Penjualan produk X minggu ini 15% di atas rata-rata," yang memungkinkan seorang manajer untuk mengambil tindakan. Informasi adalah data yang diorganisir sehingga dapat mendukung akurasi pengambilan keputusan. Kualitas informasi sangat menentukan kualitas keputusan yang dihasilkan. Oleh karena itu, informasi yang baik harus memenuhi beberapa kriteria, seperti relevan, akurat, tepat waktu, dan dapat dipahami.

Sistem adalah sebuah konsep yang lebih umum. Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan kegiatan atau mencapai sasaran tertentu. Definisi lain dari sistem adalah kumpulan dari sub-sistem atau komponen, baik fisik maupun non-fisik, yang saling berhubungan dan bekerja sama secara harmonis untuk mencapai satu tujuan tertentu. Dua kata kunci yang selalu muncul dalam definisi sistem adalah **keterhubungan (*interrelatedness*)** antar komponen dan **tujuan (*goal*)** yang ingin dicapai. Sebuah mobil adalah sistem yang terdiri dari komponen-komponen seperti mesin, roda, dan kemudi yang bekerja sama untuk tujuan transportasi.

Definisi Sistem Informasi

Dengan memahami ketiga konsep di atas, kita kini dapat mendefinisikan **Sistem Informasi (SI)** secara komprehensif. Secara esensial, sistem informasi adalah sebuah sistem yang tujuan utamanya adalah melakukan transformasi nilai, yaitu mengolah data mentah menjadi informasi yang berguna.

Para ahli telah memberikan berbagai definisi yang, meskipun berbeda dalam penekanannya, memiliki benang merah yang sama. Laudon dan Laudon, dua penulis terkemuka di bidang ini, mendefinisikan SI sebagai "sekumpulan komponen yang saling berhubungan, yang mengumpulkan (atau mendapatkan), memproses, menyimpan, dan mendistribusikan informasi untuk menunjang pengambilan keputusan dan pengawasan dalam suatu organisasi". Definisi ini menyoroti

fungsi-fungsi inti dari sebuah SI: input (mengumpulkan), proses, penyimpanan, dan output (distribusi).

Definisi lain yang lebih luas menyatakan bahwa SI adalah "suatu kombinasi teratur dari orang-orang, *hardware*, *software*, jaringan komunikasi dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi". Definisi ini secara eksplisit menyebutkan komponen-komponen yang membentuk sebuah SI. Tabel 1 merangkum beberapa definisi dari para ahli untuk memberikan perspektif yang lebih kaya.

Dari berbagai definisi ini, sebuah pola yang jelas muncul. Sistem informasi bukanlah sekedar tentang komputer atau perangkat lunak. Ia adalah sebuah **sistem sosio-teknis**, sebuah ekosistem di mana teknologi (perangkat keras, perangkat lunak) dan elemen sosial (manusia, proses) berinteraksi secara erat untuk mencapai tujuan organisasi. Nilai sebuah SI tidak diukur dari kecanggihan teknologinya, melainkan dari kemampuannya untuk secara konsisten dan andal mengubah data berbiaya rendah menjadi informasi bernilai tinggi yang dapat ditindaklanjuti untuk meningkatkan kinerja bisnis.

1.2 Anatomi Sistem Informasi: Lima Komponen Utama

Setiap sistem informasi, baik itu sistem pemesanan tiket pesawat yang kompleks maupun sistem inventaris sederhana di toko kelontong, dibangun di atas pondasi komponen-komponen yang sama. Memahami anatomi ini sangat penting bagi seorang analis sistem karena kegagalan pada salah satu komponen dapat meruntuhkan keseluruhan struktur. Secara

umum, terdapat lima komponen utama yang sering disebut, ditambah satu komponen penghubung yang kini menjadi sangat vital.

Tabel 1. Ringkasan Definisi Sistem Informasi oleh Para Pakar

Nama Pakar	Definisi	Kata Kunci Utama
Kenneth C. Laudon & Jane P. Laudon	Sekumpulan komponen yang saling berhubungan, mengumpulkan, memproses, menyimpan, dan mendistribusikan informasi untuk menunjang pengambilan keputusan dan pengawasan dalam organisasi.	Komponen, Proses (kumpul, olah, simpan, distribusi), Tujuan (keputusan, pengawasan).
John F. Nash	Kombinasi dari manusia, fasilitas/teknologi, media, prosedur, dan pengendalian yang ditujukan untuk menata jaringan komunikasi, memproses transaksi, dan membantu manajemen.	Kombinasi, Manusia, Prosedur, Pengendalian, Komunikasi.
Alter	Kombinasi antara prosedur kerja, informasi, orang, dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah perusahaan.	Kombinasi, Prosedur, Orang, Teknologi, Tujuan.
Bodnar & HopWood	Kumpulan perangkat keras dan lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.	Hardware, Software, Transformasi Data ke Informasi.

1. **Perangkat Keras (*Hardware*)**: Ini adalah komponen fisik dari teknologi informasi. Perangkat keras mencakup segala sesuatu yang dapat Anda sentuh, mulai dari komputer server yang kuat di pusat data, komputer desktop di meja kerja, laptop, tablet, hingga perangkat input (seperti keyboard, mouse, pemindai) dan perangkat output (seperti monitor, printer). Komponen ini juga mencakup perangkat penyimpanan data (seperti hard disk drive dan solid-state drive) serta infrastruktur jaringan fisik. Dahulu, perangkat

keras didominasi oleh server terpusat yang besar dan mahal. Namun, kini arsitekturnya telah berevolusi menjadi sistem terdistribusi dan komputasi awan (*cloud computing*), di mana sumber daya perangkat keras dapat diakses sesuai permintaan melalui internet.

2. **Perangkat Lunak (*Software*)**: Jika perangkat keras adalah "tubuh" dari sistem, maka perangkat lunak adalah "pikiran"-nya. Perangkat lunak adalah serangkaian instruksi atau program yang memberitahu perangkat keras apa yang harus dilakukan. Perangkat lunak secara umum dibagi menjadi dua kategori utama:
    - **Perangkat Lunak Sistem (*System Software*)**: Ini adalah pondasi yang mengelola dan mengontrol sumber daya komputer. Contoh paling umum adalah sistem operasi (seperti Windows, macOS, Linux, Android) dan sistem manajemen basis data (DBMS).
    - **Perangkat Lunak Aplikasi (*Application Software*)**: Ini adalah program yang dirancang untuk melakukan tugas-tugas spesifik bagi pengguna. Contohnya sangat beragam, mulai dari aplikasi perkantoran (seperti Microsoft Office), sistem perencanaan sumber daya perusahaan (ERP), sistem manajemen hubungan pelanggan (CRM), hingga aplikasi khusus industri untuk penjadwalan produksi atau pelacakan logistik.
3. **Data**: Data adalah "darah" yang mengalir dalam sistem informasi. Seperti yang telah dibahas, data adalah fakta mentah yang dikumpulkan, disimpan, dan diproses oleh sistem. Dalam konteks SI modern, data biasanya diorganisir secara sistematis dalam sebuah **basis data (database)**. Basis

data adalah kumpulan data yang saling terkait dan terstruktur yang memungkinkan penyimpanan, pengambilan, dan pengelolaan data secara efisien. Kualitas sistem informasi sangat bergantung pada kualitas datanya. Jika data yang dimasukkan tidak akurat, tidak lengkap, atau tidak konsisten (prinsip "*Garbage In, Garbage Out*"), maka informasi yang dihasilkan, secanggih apa pun perangkat keras dan perangkat lunaknya, tidak akan berharga.

4. **Manusia (*People*/*Brainware*)**: Seringkali diabaikan, manusia adalah komponen yang paling krusial dalam sebuah sistem informasi. Komponen ini mencakup semua orang yang terlibat dalam siklus hidup sistem, mulai dari **pengguna akhir** (seperti kasir, manajer, pelanggan) yang berinteraksi dengan sistem setiap hari, hingga para **profesional TI** yang merancang, membangun, dan memeliharanya (seperti analis sistem, pemrogram, administrator basis data). Keahlian, motivasi, penerimaan terhadap perubahan, dan pelatihan yang memadai bagi komponen manusia ini seringkali menjadi faktor penentu antara keberhasilan dan kegagalan sebuah proyek SI.
5. **Proses (*Procedures*)**: Proses, atau prosedur, adalah aturan, kebijakan, dan strategi formal yang mengatur bagaimana keempat komponen lainnya harus dioperasikan, digunakan, dan dipelihara. Ini adalah "buku panduan" atau "standar operasional prosedur (SOP)" dari sistem informasi. Proses mendefinisikan alur kerja, seperti "bagaimana cara memasukkan pesanan pelanggan baru," "siapa yang berwenang menyetujui diskon," atau "langkah-langkah apa yang harus diambil jika sistem mengalami gangguan." Tanpa proses yang jelas dan terdokumentasi, penggunaan

sistem akan menjadi tidak konsisten, rentan terhadap kesalahan, dan sulit untuk dikelola.

6. **Jaringan (*Networks*/*Telecommunications*)**: Dalam dunia yang terhubung saat ini, jaringan komunikasi sering dianggap sebagai komponen keenam yang fundamental. Jaringan adalah "sistem saraf" yang menghubungkan semua komponen lainnya, memungkinkan mereka untuk berkomunikasi dan berbagi data. Ini mencakup Local Area Network (LAN) di dalam sebuah gedung, Wide Area Network (WAN) yang menghubungkan lokasi-lokasi yang berjauhan, dan tentu saja, internet. Telekomunikasi memungkinkan arsitektur sistem terdistribusi, komputasi awan, dan akses seluler, yang merupakan pondasi dari hampir semua sistem informasi modern.

Semua komponen ini tidak berdiri sendiri; mereka membentuk sebuah ekosistem yang saling bergantung. Pendekatan yang hanya berfokus pada teknologi (perangkat keras dan perangkat lunak) dan mengabaikan interaksi kompleks dengan manusia, proses, dan data, adalah pendekatan yang pasti akan menemui masalah. Analis sistem yang efektif harus mengadopsi pandangan **sosio-teknis**, memahami bahwa merancang sistem informasi berarti merancang sebuah ekosistem yang seimbang. Ketika sebuah perangkat lunak baru diperkenalkan, analis harus bertanya: Perangkat keras apa yang dibutuhkan? Bagaimana struktur data akan berubah? Siapa yang akan menggunakannya dan pelatihan apa yang mereka perlukan? Proses bisnis apa yang harus disesuaikan? Memahami anatomi ini secara holistik adalah kunci untuk merancang sistem yang tidak hanya berfungsi secara teknis, tetapi juga diadopsi, digunakan, dan memberikan nilai bagi organisasi.

Gambar 1. Komponen Sistem Informasi

1.3 Evolusi Sistem Informasi dari Mainframe hingga AI

Memahami kondisi sistem informasi saat ini tidak akan lengkap tanpa menengok kembali perjalanannya. Evolusi SI bukanlah serangkaian lompatan teknologi yang acak, melainkan sebuah narasi yang menarik tentang bagaimana kebutuhan bisnis dan inovasi teknologi saling membentuk satu sama lain. Perjalanan ini dapat dibagi menjadi empat era utama, di mana setiap era ditandai oleh pergeseran fundamental dalam arsitektur komputasi dan cara organisasi memanfaatkan informasi.

Era Mainframe (1960-an – 1970-an): Komputasi Terpusat dan Lahirnya Kompatibilitas

Pada masa-masa awal komputasi, dunia teknologi informasi sangat terfragmentasi. Sebelum tahun 1964, setiap komputer yang diproduksi adalah sebuah "pulau" yang terisolasi. Setiap mesin memiliki arsitektur, set instruksi, dan sistem operasi yang unik. Akibatnya, perusahaan yang ingin meningkatkan kapasitas komputasinya harus membuang sistem lama mereka dan menulis ulang semua perangkat lunak dari awal, sebuah proses yang sangat mahal dan memakan waktu.[7] IBM sendiri, sebagai pemimpin pasar, memiliki lima lini produk yang sama sekali tidak kompatibel, memisahkan secara kaku antara komputer untuk aplikasi "ilmiah" dan "komersial".

Titik balik terjadi pada tanggal 7 April 1964, ketika IBM mengumumkan **System/360 (S/360)**. Ini adalah sebuah pertaruhan besar senilai $5 miliar pada saat itu, yang bertujuan untuk menggantikan semua lini produk IBM yang ada dengan satu keluarga komputer yang arsitekturnya kompatibel. S/360, yang namanya menyiratkan cakupan 360 derajat untuk semua jenis penggunaan, membawa beberapa inovasi revolusioner:

- **Kompatibilitas:** Untuk pertama kalinya, sebuah program yang ditulis untuk model S/360 yang lebih kecil dapat berjalan tanpa modifikasi pada model yang lebih besar. Ini memungkinkan perusahaan untuk memulai dengan sistem kecil dan melakukan upgrade seiring pertumbuhan bisnis mereka tanpa harus menulis ulang perangkat lunak.
- **Pemisahan Perangkat Keras dan Perangkat Lunak:** Dengan adanya arsitektur yang seragam, konsep industri perangkat lunak komersial menjadi mungkin. Perusahaan lain dapat membuat perangkat lunak atau periferal yang kompatibel dengan S/360.

- **Standarisasi:** S/360 mempopulerkan penggunaan byte 8-bit dan arsitektur 32-bit, yang kemudian menjadi standar industri selama beberapa dekade.

Pada era ini, SI dicirikan oleh **sentralisasi ekstrim**. Komputer mainframe yang besar dan mahal ditempatkan di ruang data khusus yang dikelola oleh departemen TI. Pengguna akhir tidak memiliki akses langsung; mereka menyerahkan permintaan pemrosesan data (biasanya dalam bentuk *punch cards*) dan menunggu hasilnya. Aplikasi utama adalah **Transaction Processing Systems (TPS)** untuk mengotomatiskan tugas-tugas administratif bervolume tinggi seperti penggajian, akuntansi, dan manajemen inventaris.

Era Komputer Personal (1980-an): Demokratisasi Komputasi

Revolusi berikutnya dipicu oleh munculnya mikroprosesor, yang melahirkan **Komputer Personal (PC)**. Era ini menandai pergeseran kekuatan komputasi yang dramatis dari pusat data terpusat ke meja kerja individu. Ini adalah era **desentralisasi radikal**. Manajer dan analis bisnis, yang sebelumnya bergantung pada laporan cetak dari departemen TI, kini memiliki alat komputasi di ujung jari mereka.

Fenomena ini melahirkan konsep *end-user computing* dan jenis sistem informasi yang sama sekali baru. Jika era mainframe berfokus pada efisiensi pemrosesan transaksi, era PC berfokus pada efektivitas pengambilan keputusan manajerial. Muncullah **Decision Support Systems (DSS)**, yaitu sistem interaktif yang membantu manajer menganalisis data dan membuat keputusan semi-terstruktur (keputusan yang sebagian dapat diotomatisasi

tetapi masih memerlukan penilaian manusia). "Aplikasi pembunuh" (*killer app*) pertama untuk PC, **VisiCalc**, adalah sebuah program spreadsheet yang pada dasarnya merupakan DSS pribadi. Selain DSS, muncul pula **Executive Information Systems (EIS)**, yang menyediakan ringkasan data tingkat tinggi dan grafis bagi para eksekutif puncak untuk memantau kinerja organisasi.

Era Internet dan E-Commerce (1990-an – 2000-an): Dunia yang Terhubung

Jika PC mendobrak dinding pusat data, maka internet meruntuhkan dinding organisasi itu sendiri. Adopsi massal internet dan World Wide Web pada pertengahan 1990-an secara fundamental mengubah cara sistem informasi dirancang dan digunakan. Sistem tidak lagi beroperasi dalam isolasi; mereka kini terhubung dalam jaringan global.

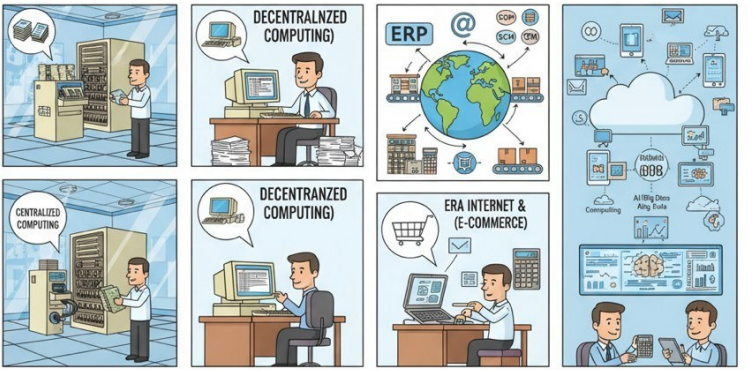
Era ini melahirkan **E-business** dan **E-commerce**. E-commerce mengacu pada proses pembelian dan penjualan produk, layanan, dan informasi melalui jaringan komputer, sedangkan E-business memiliki cakupan yang lebih luas, mencakup kolaborasi dengan mitra bisnis, layanan pelanggan, dan operasi internal berbasis web. Sistem informasi seperti *Enterprise Resource Planning* (ERP), *Supply Chain Management* (SCM), dan *Customer Relationship Management* (CRM) menjadi terintegrasi tidak hanya di dalam perusahaan tetapi juga dengan sistem pemasok dan pelanggan. Arsitektur *client-server* menjadi dominan, di mana PC pengguna (klien) mengakses data dan logika bisnis yang tersimpan di server pusat. Era ini menciptakan model bisnis yang sepenuhnya baru dan

memungkinkan perusahaan, besar maupun kecil, untuk bersaing di pasar global.

Era Cloud, Mobile, dan AI (2010-an – Sekarang): Kecerdasan di Mana Saja

Era terkini didorong oleh konvergensi tiga teknologi transformatif: komputasi awan (*cloud*), komputasi seluler (*mobile*), dan kecerdasan buatan (*Artificial Intelligence* - AI).

- **Cloud Computing** menawarkan sumber daya komputasi (server, penyimpanan, basis data, perangkat lunak) sebagai layanan sesuai permintaan melalui internet. Ini menghilangkan kebutuhan bagi perusahaan untuk berinvestasi dan memelihara infrastruktur TI fisik yang mahal, serta memberikan elastisitas dan skalabilitas yang belum pernah terjadi sebelumnya.
- **Mobile Computing**, yang didorong oleh smartphone dan tablet, telah membuat akses ke sistem informasi menjadi ada di mana-mana (*ubiquitous*). Aplikasi kini dapat diakses kapan saja, di mana saja, mengubah cara kita bekerja, berbelanja, dan berkomunikasi.
- **Artificial Intelligence (AI) dan Big Data** memberikan kemampuan untuk menganalisis kumpulan data yang sangat besar (*Big Data*) untuk menemukan pola tersembunyi, membuat prediksi, dan mengotomatiskan tugas-tugas kognitif yang kompleks. AI kini tertanam dalam berbagai sistem, mulai dari sistem rekomendasi di platform e-commerce hingga analisis prediktif dalam manajemen rantai pasok dan deteksi penipuan di sektor keuangan.



Gambar 2. Evolusi Sistem Informasi

Jika kita lihat kembali perjalanan ini, sebuah pola menarik muncul. Sejarah SI dapat dilihat sebagai pendulum yang terus berayun antara **sentralisasi** dan **desentralisasi**. Era mainframe adalah puncak sentralisasi. Era PC adalah pemberontakan desentralisasi. Era internet dan cloud menciptakan model hibrid yang canggih: sumber daya infrastruktur dapat disentralisasi secara masif di cloud untuk efisiensi, namun akses dan penggunaannya dapat didesentralisasi sepenuhnya melalui perangkat seluler dan IoT di seluruh dunia. AI pun mengikuti pola serupa, dengan pelatihan model raksasa yang dilakukan secara terpusat, tetapi penerapannya (inferensi) semakin didistribusikan ke perangkat di tepi jaringan (*edge devices*). Memahami dinamika ini membantu analis sistem berpikir secara strategis tentang arsitektur sistem yang paling sesuai untuk tantangan bisnis di masa depan.

Tabel 2. Evolusi Sistem Informasi Lintas Era

Era	Rentang Waktu	Teknologi Kunci	Karakteristik Utama SI	Contoh Aplikasi

Mainframe	1960-an – 1970-an	Mainframe, Kompatibilitas (S/360)	Terpusat, Pemrosesan Batch, Dikelola oleh Ahli TI	Sistem Penggajian, Akuntansi, Inventaris (TPS)
Personal Computer (PC)	1980-an	PC, Spreadsheet, LAN	Terdesentralisasi, *End-User Computing*, Interaktif	Decision Support Systems (DSS), Executive Information Systems (EIS)
Internet & E-Commerce	1990-an – 2000-an	Internet, WWW, Client-Server	Terhubung, Global, Lintas Organisasi	E-commerce, ERP, SCM, CRM
Cloud, Mobile, & AI	2010-an – Sekarang	Cloud Computing, Smartphone, AI, Big Data	*Ubiquitous*, Sesuai Permintaan, Cerdas, Berbasis Data	Aplikasi Seluler, SaaS, Analitik Prediktif, Sistem Rekomendasi

1.4 Peran Strategis Sistem Informasi

Setelah memahami apa itu sistem informasi dan bagaimana ia berevolusi, pertanyaan mendasar berikutnya adalah: "Mengapa sistem informasi begitu penting?" Di masa lalu, departemen Teknologi Informasi (TI) seringkali dipandang sebagai *cost center*, sebuah fungsi pendukung yang tugasnya adalah menjaga agar komputer tetap berjalan dan email terkirim. Namun, pandangan ini sudah usang. Di era digital, sistem informasi telah bertransformasi menjadi *value center* dan merupakan jantung dari strategi, operasi, dan inovasi organisasi modern. Peran strategisnya dapat dilihat dari tiga kontribusi utama.

Meningkatkan Efisiensi Operasional

Ini adalah peran paling fundamental dan tradisional dari SI. Dengan mengotomatiskan proses bisnis yang sebelumnya manual dan memakan waktu, SI dapat secara dramatis meningkatkan efisiensi dan produktivitas. Sistem dapat memproses ribuan transaksi dalam hitungan detik, mengurangi kesalahan manusia, dan membebaskan karyawan dari tugas-tugas rutin untuk fokus pada pekerjaan yang lebih bernilai tambah. Contoh klasik adalah sistem **Enterprise Resource Planning (ERP)**, yang mengintegrasikan berbagai fungsi bisnis seperti keuangan, sumber daya manusia, produksi, dan logistik ke dalam satu basis data terpusat. Integrasi ini memungkinkan aliran informasi yang lancar antar departemen, menghilangkan redundansi data, dan memberikan pandangan menyeluruh tentang operasi perusahaan. Demikian pula, dalam industri manufaktur, SI digunakan untuk mengoptimalkan jadwal produksi, mengelola inventaris secara *just-in-time*, dan melacak produk di sepanjang rantai pasok.

Mendukung Pengambilan Keputusan Berbasis Data

Di luar efisiensi, peran SI yang semakin penting adalah sebagai pondasi untuk pengambilan keputusan yang lebih baik. Organisasi modern menghasilkan volume data yang luar biasa dari setiap transaksi, interaksi pelanggan, dan proses operasional. SI tidak hanya menyimpan data ini, tetapi juga menyediakan alat untuk mengubahnya menjadi informasi yang dapat ditindaklanjuti. Sistem seperti **Business Intelligence (BI)** dan platform analitik data memungkinkan manajer untuk:

- **Memantau Kinerja:** Melalui dasbor interaktif, manajer dapat melihat metrik kinerja utama (*Key Performance Indicators* - KPIs) secara *real-time*.

- **Menganalisis Tren:** Manajer dapat mengidentifikasi pola penjualan, perilaku pelanggan, atau inefisiensi operasional yang mungkin tidak terlihat dari laporan manual.
- **Membuat Prediksi:** Dengan menggunakan teknik statistik dan AI, sistem dapat meramalkan permintaan di masa depan, memprediksi pelanggan mana yang berisiko beralih ke pesaing, atau mengidentifikasi potensi kegagalan peralatan.

Dengan menyediakan informasi yang akurat, tepat waktu, dan relevan, SI memungkinkan para pemimpin untuk beralih dari pengambilan keputusan yang didasarkan pada intuisi atau "firasat" menjadi keputusan yang didukung oleh bukti dan data (*data-driven decision making*).

Memungkinkan Inovasi dan Keunggulan Kompetitif

Ini adalah peran SI yang paling strategis. Di era digital, SI bukan lagi hanya tentang melakukan hal-hal lama dengan cara yang lebih baik; tetapi tentang melakukan hal-hal yang sama sekali baru yang sebelumnya tidak mungkin. SI adalah mesin inovasi yang dapat menciptakan keunggulan kompetitif yang berkelanjutan.

- **Model Bisnis Baru:** Perusahaan seperti Netflix (streaming video), Uber (ride-sharing), dan Airbnb (akomodasi) tidak akan ada tanpa platform sistem informasi yang canggih sebagai inti dari model bisnis mereka.
- **Produk dan Layanan Baru:** SI memungkinkan penciptaan produk dan layanan digital. Bank mengembangkan aplikasi *mobile banking*; produsen mobil menanamkan sistem cerdas pada kendaraan mereka.

- **Peningkatan Hubungan Pelanggan:** Sistem **Customer Relationship Management (CRM)** memungkinkan perusahaan untuk mengumpulkan dan menganalisis setiap interaksi dengan pelanggan. Pengetahuan ini digunakan untuk mempersonalisasi pemasaran, memberikan layanan yang lebih proaktif, dan membangun loyalitas pelanggan jangka panjang.
- **Kolaborasi Lintas Disiplin:** SI menjadi jembatan vital antara berbagai bidang keilmuan. Dalam Teknik Industri, misalnya, para insinyur bekerja sama dengan profesional TI untuk merancang dan mengimplementasikan sistem informasi yang mengoptimalkan seluruh rantai nilai, dari desain produk hingga pengiriman ke pelanggan akhir, yang pada akhirnya meningkatkan daya saing perusahaan.

Pergeseran peran SI dari pendukung operasional menjadi penggerak strategis memiliki implikasi mendalam bagi Anda sebagai calon analis sistem. Belajar Analisis dan Perancangan Sistem Informasi bukan lagi hanya tentang menjadi seorang teknisi yang dapat menerjemahkan kebutuhan bisnis menjadi kode. Ini adalah tentang menjadi mitra strategis bagi bisnis, seseorang yang dapat memahami tantangan dan peluang organisasi, dan kemudian secara proaktif merancang solusi berbasis teknologi yang menciptakan nilai, mendorong inovasi, dan mengamankan masa depan perusahaan di lanskap persaingan yang terus berubah.

1.5 Urgensi Analisis dan Perancangan Sistem

Setelah memahami betapa vitalnya peran sistem informasi bagi organisasi modern, kita harus menghadapi sebuah kenyataan

yang mengejutkan dan seringkali pahit: membangun sistem informasi yang sukses adalah pekerjaan yang sangat sulit. Tingkat kegagalan proyek Teknologi Informasi (TI) secara historis sangat tinggi. Kenyataan inilah yang menjadi justifikasi utama dan memberikan urgensi pada disiplin ilmu Analisis dan Perancangan Sistem Informasi. Mata kuliah ini ada bukan karena membangun sistem itu mudah, tetapi justru karena sangat sulit dan taruhannya sangat tinggi.

Realitas Pahit Proyek Teknologi Informasi

Selama beberapa dekade, **The Standish Group**, sebuah firma riset internasional, telah melacak hasil dari puluhan ribu proyek pengembangan perangkat lunak di seluruh dunia. Laporan mereka, yang dikenal sebagai **CHAOS Report**, secara konsisten melukiskan gambaran yang serius. Proyek-proyek diklasifikasikan ke dalam tiga kategori:

- **Sukses (Successful):** Proyek selesai tepat waktu, sesuai anggaran, dan memberikan semua fitur dan fungsionalitas yang direncanakan.
- **Ditentang (Challenged):** Proyek selesai dan dapat dioperasikan, tetapi melebihi anggaran, terlambat dari jadwal, dan/atau dengan fitur yang lebih sedikit dari yang direncanakan.
- **Gagal (Failed):** Proyek dibatalkan sebelum selesai atau selesai tetapi tidak pernah digunakan oleh organisasi.

Data dari laporan CHAOS tahun 2020 menunjukkan gambaran yang menantang, meskipun ada beberapa perbaikan selama bertahun-tahun:

- Hanya **31%** dari proyek yang dianggap sepenuhnya **sukses**.
- Sebanyak **50%** dari proyek masuk kategori **ditantang**.

- **19%** dari proyek **gagal** total.

Tabel 3. Statistik Hasil Proyek TI (Berdasarkan Standish Group CHAOS Report 2020)

Hasil Proyek	Persentase	Deskripsi
Sukses	31%	Selesai tepat waktu, sesuai anggaran, dengan lingkup penuh.
Ditentang	50%	Selesai, tetapi melebihi anggaran/waktu dan/atau lingkup berkurang.
Gagal	19%	Dibatalkan sebelum selesai atau tidak pernah diimplementasikan.

Angka-angka ini menjadi lebih buruk untuk proyek-proyek besar. Laporan tersebut secara konsisten menemukan bahwa semakin besar dan kompleks sebuah proyek, semakin besar kemungkinannya untuk gagal. Proyek-proyek besar (dengan biaya jutaan dolar) memiliki tingkat keberhasilan kurang dari 10%. Biaya dari kegagalan ini sangat besar, mencapai ratusan miliar dolar per tahun secara global dalam bentuk investasi yang terbuang, belum lagi biaya peluang yang hilang.

Menggali Akar Masalah, Mengapa Proyek Gagal?

Apa yang menyebabkan krisis ini? Apakah karena teknologi yang terlalu sulit atau pemrogram yang tidak kompeten? Penelitian secara konsisten menunjukkan bahwa akar penyebab kegagalan jarang bersifat murni teknis. Sebaliknya, kegagalan paling sering berasal dari masalah dalam proses, manajemen, dan komunikasi, masalah-masalah yang berada di ranah manusia dan organisasi.

Laporan CHAOS dan penelitian lain dari lembaga seperti Project Management Institute (PMI) telah mengidentifikasi beberapa faktor utama yang berulang kali muncul sebagai penyebab kegagalan proyek:

1. **Kurangnya Keterlibatan Pengguna (Lack of User Involvement):** Ketika pengguna akhir, orang-orang yang sebenarnya akan menggunakan sistem, tidak dilibatkan secara aktif dalam mendefinisikan kebutuhan dan memberikan umpan balik, sistem yang dihasilkan seringkali tidak sesuai dengan alur kerja mereka dan akhirnya ditolak.
2. **Persyaratan yang Tidak Lengkap atau Tidak Jelas (Incomplete/Unclear Requirements):** Ini adalah penyebab kegagalan yang paling sering dikutip. Jika di awal proyek tidak ada pemahaman yang jelas dan terperinci tentang *apa* yang harus dilakukan oleh sistem, maka tim pengembang pada dasarnya bekerja dalam kegelapan.
3. **Perubahan Persyaratan yang Konstan (Changing Requirements/ Scope Creep):** Ketika lingkup proyek terus-menerus berubah atau bertambah tanpa kontrol yang tepat, proyek akan mengalami "scope creep." Hal ini menyebabkan penundaan jadwal, pembengkakan anggaran, dan seringkali penurunan kualitas karena tim terburu-buru mengakomodasi fitur-fitur baru.
4. **Kurangnya Dukungan Eksekutif (Lack of Executive Support):** Proyek SI yang signifikan memerlukan dukungan yang kuat dari manajemen puncak untuk menyediakan sumber daya, mengatasi hambatan politis, dan memperjuangkan perubahan. Tanpa dukungan ini, proyek seringkali kehilangan momentum dan gagal.

Analisis dan Perancangan Sistem sebagai Solusi

Disiplin **Analisis dan Perancangan Sistem Informasi (APSI)** muncul sebagai respons langsung dan profesional terhadap krisis kegagalan proyek ini. APSI bukanlah sekedar kumpulan alat dan teknik menggambar diagram; ini adalah sebuah pendekatan terstruktur untuk **manajemen risiko** yang dirancang untuk secara sistematis mengatasi akar penyebab kegagalan. Ada hubungan kausal yang sangat jelas antara penyebab kegagalan dan aktivitas inti dalam APSI:

- **Analisis Sistem** secara langsung menanggulangi masalah **persyaratan yang tidak lengkap** dan **kurangnya keterlibatan pengguna**. Analisis adalah proses investigasi yang mendalam untuk mengidentifikasi masalah, memahami konteks bisnis, dan bekerja sama dengan pengguna dan pemangku kepentingan lainnya untuk mendefinisikan dan mendokumentasikan kebutuhan mereka secara tepat dan lengkap.
- **Perancangan Sistem** secara langsung menanggulangi masalah **scope creep** dan memberikan panduan teknis yang jelas. Perancangan adalah proses mengubah kebutuhan yang telah dianalisis menjadi sebuah cetak biru (*blueprint*) yang detail dan stabil untuk sistem. Cetak biru ini menetapkan batasan yang jelas untuk lingkup proyek dan berfungsi sebagai panduan konkret bagi para pemrogram untuk membangun sistem, memastikan bahwa apa yang dibangun sesuai dengan apa yang dibutuhkan.

Setiap topik yang akan dipelajari dalam buku ini, mulai dari teknik wawancara, pemodelan proses, perancangan basis data, hingga metodologi pengembangan, memiliki tujuan akhir yang sama: untuk meningkatkan peluang keberhasilan proyek dengan menerapkan disiplin dan struktur pada proses

pengembangan yang secara inheren kompleks dan kacau. Menguasai APSI berarti tidak hanya belajar cara membangun sistem, tetapi juga belajar cara membangun sistem yang *berhasil*.

1.6 Arsitek Solusi Digital: Peran dan Keahlian Analis Sistem

Di pusat proses analisis dan perancangan sistem berdiri seorang profesional kunci: **Analis Sistem**. Siapakah analis sistem, dan apa yang mereka lakukan? Memahami peran ini akan memberi Anda gambaran tentang jalur karier yang menarik dan berdampak yang terbuka bagi mereka yang menguasai disiplin ilmu ini.

Secara fundamental, seorang analis sistem adalah **pemecah masalah**. Mereka adalah individu yang menggunakan pendekatan analitis dan sistematis untuk mempelajari masalah atau peluang bisnis, dan kemudian merancang perbaikan pada proses bisnis dan sistem informasi untuk mengatasi masalah tersebut. Mereka adalah agen perubahan yang membantu organisasi memanfaatkan teknologi untuk beroperasi lebih efisien dan efektif.

Salah satu metafora terbaik untuk peran analis sistem adalah sebagai **jembatan**. Mereka berdiri di antara dua dunia yang seringkali berbicara dalam bahasa yang berbeda: dunia **bisnis** dan dunia **teknologi**.

- Di satu sisi adalah para pemangku kepentingan bisnis (manajer, pengguna akhir, eksekutif) yang memahami proses operasional, tantangan pasar, dan tujuan strategis. Mereka tahu *apa* yang mereka butuhkan dari sebuah sistem,

tetapi mungkin tidak tahu *bagaimana* hal itu dapat diwujudkan secara teknis.

- Di sisi lain adalah tim teknis (pemrogram, insinyur jaringan, administrator basis data) yang memiliki keahlian mendalam dalam membangun dan mengelola teknologi. Mereka tahu *bagaimana* membangun sesuatu, tetapi memerlukan spesifikasi yang jelas tentang *apa* yang harus dibangun.

Analis sistem adalah penerjemah dan fasilitator yang menjembatani kesenjangan ini. Mereka harus fasih dalam kedua "bahasa" tersebut, mampu mendengarkan masalah bisnis dan menerjemahkannya menjadi persyaratan teknis yang dapat ditindaklanjuti, dan sebaliknya, mampu menjelaskan implikasi dan batasan teknis kepada pemangku kepentingan bisnis dalam istilah yang dapat mereka pahami.

Untuk berhasil dalam peran ini, seorang analis sistem harus mengembangkan serangkaian keahlian yang beragam, yang dapat dikelompokkan menjadi empat bidang utama:

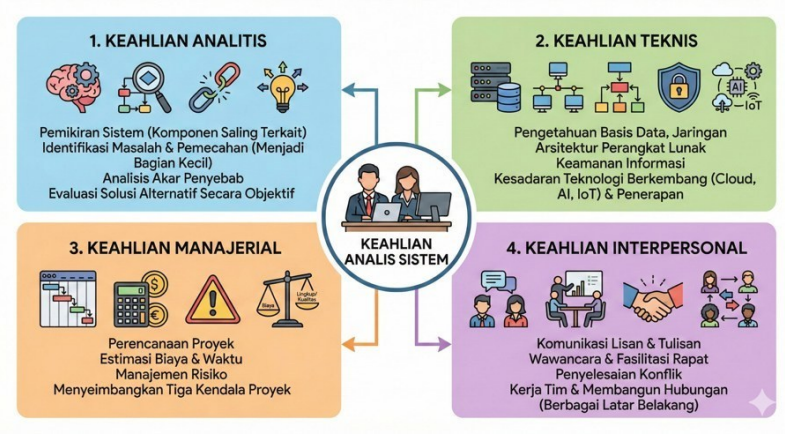
1. **Keahlian Analitis:** Ini adalah inti dari peran mereka. Keahlian ini mencakup pemikiran sistem, yaitu kemampuan untuk melihat organisasi sebagai sebuah sistem dari komponen-komponen yang saling terkait. Ini juga mencakup kemampuan untuk mengidentifikasi masalah, memecahnya menjadi bagian-bagian yang lebih kecil dan dapat dikelola, menganalisis akar penyebab, dan mengevaluasi solusi alternatif secara objektif.
2. **Keahlian Teknis:** Meskipun seorang analis sistem tidak harus menjadi pemrogram ahli, mereka harus memiliki pemahaman yang kuat tentang teknologi. Ini termasuk

pengetahuan tentang basis data, jaringan, arsitektur perangkat lunak, keamanan informasi, dan yang terpenting, kesadaran akan teknologi yang sedang berkembang (seperti cloud, AI, IoT) dan bagaimana teknologi tersebut dapat diterapkan untuk memecahkan masalah bisnis.

3. **Keahlian Manajerial:** Analis sistem seringkali bertindak sebagai manajer proyek atau pemimpin tim. Mereka perlu memahami cara merencanakan proyek, memperkirakan biaya dan waktu, mengelola risiko, dan menyeimbangkan tiga kendala proyek: biaya, waktu, dan lingkup (kualitas).
4. **Keahlian Interpersonal:** Mungkin ini adalah keahlian yang paling penting dari semuanya. Karena peran mereka sebagai jembatan, analis sistem harus menjadi komunikator yang luar biasa, baik secara lisan maupun tulisan. Mereka harus mampu melakukan wawancara yang efektif, memfasilitasi rapat, menyelesaikan konflik, bekerja dalam tim, dan membangun hubungan baik dengan orang-orang dari berbagai latar belakang dan tingkat jabatan dalam organisasi.

Peran analis sistem modern telah berevolusi jauh dari sekedar menjadi seorang juru tulis teknis. Di masa lalu, mereka mungkin hanya menerima daftar persyaratan dari bisnis dan meneruskannya ke tim pengembang. Kini, analis sistem yang efektif adalah seorang **konsultan internal yang proaktif**. Mereka tidak menunggu masalah datang kepada mereka. Sebaliknya, mereka secara aktif bekerja sama dengan para pemimpin bisnis untuk mengidentifikasi peluang strategis, membantu para pengguna untuk menemukan dan mengartikulasikan kebutuhan mereka (bahkan kebutuhan yang belum mereka sadari), dan memandu organisasi melalui proses

perubahan yang seringkali sulit yang menyertai pengenalan sistem baru. Mereka adalah arsitek solusi digital, inovator, dan pemimpin, sebuah peran yang menantang namun sangat memuaskan.



Gambar 3. Keahlian Analis Sistem

Contoh Soal & Studi Kasus

Bagian ini dirancang untuk membantu Anda menerapkan konsep-konsep yang telah dipelajari dalam bab ini. Cobalah untuk menjawab pertanyaan dan menganalisis studi kasus sebelum membaca solusinya.

Soal Konseptual (Esai Singkat)

1. Jelaskan dengan kata-kata Anda sendiri perbedaan mendasar antara data, informasi, dan sistem. Mengapa pembedaan ini penting dalam konteks analisis dan perancangan sistem informasi?
2. Sebuah perusahaan ritel memutuskan untuk mengimplementasikan perangkat lunak CRM (Customer Relationship Management) baru (komponen Perangkat Lunak). Jelaskan bagaimana keputusan ini kemungkinan besar akan berdampak pada empat komponen sistem informasi lainnya (Perangkat Keras, Data, Manusia, dan Proses).
3. Berdasarkan Laporan CHAOS, "persyaratan yang tidak lengkap atau tidak jelas" adalah salah satu penyebab utama kegagalan proyek. Jelaskan mengapa hal ini menjadi masalah besar dan berikan dua contoh tindakan spesifik yang dapat dilakukan seorang analis sistem selama fase analisis untuk memitigasi risiko ini.

Solusi dan Pembahasan Soal Konseptual

1. **Perbedaan Data, Informasi, dan Sistem:**

- **Perbedaan: Data** adalah fakta mentah, terisolasi, dan tanpa konteks (misalnya, angka 150). **Informasi** adalah data yang telah diproses, diorganisir, dan diberi makna sehingga berguna untuk pengambilan keputusan (misalnya, "Penjualan produk A hari ini adalah 150 unit, naik 20% dari kemarin"). **Sistem** adalah kumpulan komponen yang saling terkait yang bekerja sama untuk mencapai tujuan tertentu. Sistem informasi adalah jenis sistem spesifik yang tujuannya adalah mengubah data menjadi informasi.
- **Pentingnya:** Pembedaan ini krusial karena tujuan utama dari analisis dan perancangan sistem bukanlah untuk membangun teknologi yang hanya mengumpulkan **data**, melainkan untuk menciptakan sistem yang menghasilkan **informasi** berkualitas yang mendukung tujuan bisnis. Seorang analis harus fokus pada output (informasi yang dibutuhkan) untuk dapat menentukan input (data yang harus dikumpulkan) dan proses (transformasi yang harus dilakukan) secara benar.

2. **Dampak Implementasi CRM pada Komponen Lain:**
   - **Perangkat Keras:** Perusahaan mungkin perlu membeli atau meningkatkan server untuk menjalankan perangkat lunak CRM dan menyimpan datanya. Jika CRM berbasis cloud, mereka memerlukan koneksi internet yang andal dan mungkin perlu menyediakan

perangkat seluler (tablet/smartphone) bagi tim penjualan untuk mengakses sistem di lapangan.

- **Data:** Perusahaan harus memutuskan data pelanggan apa yang akan dikumpulkan (misalnya, riwayat pembelian, preferensi, interaksi layanan pelanggan). Data pelanggan yang ada dari sistem lama (misalnya, spreadsheet) perlu dibersihkan, diformat ulang, dan dimigrasikan ke basis data CRM yang baru.
- **Manusia:** Tim penjualan, pemasaran, dan layanan pelanggan (pengguna akhir) harus dilatih tentang cara menggunakan sistem CRM yang baru. Mungkin ada penolakan terhadap perubahan karena mereka sudah terbiasa dengan cara kerja lama. Profesional TI juga mungkin memerlukan pelatihan untuk mengelola dan memelihara sistem baru.
- **Proses:** Alur kerja akan berubah secara signifikan. Proses pencatatan interaksi pelanggan, pelacakan prospek penjualan, dan penanganan keluhan harus didefinisikan ulang dan distandarisasi sesuai dengan fungsionalitas CRM. SOP baru harus dibuat dan disosialisasikan.

3. **Mengatasi Persyaratan yang Tidak Lengkap:**
    - **Mengapa Masalah Besar:** Persyaratan yang tidak lengkap atau tidak jelas adalah resep untuk kegagalan karena tim pengembang tidak memiliki panduan yang solid tentang apa yang harus dibangun. Hal ini menyebabkan

pengerjaan ulang yang mahal, fitur yang salah, perkiraan waktu dan biaya yang tidak akurat, dan pada akhirnya menghasilkan sistem yang tidak memenuhi kebutuhan bisnis atau pengguna. Ini seperti mencoba membangun rumah tanpa cetak biru yang detail.

- **Tindakan Mitigasi oleh Analis Sistem:**
    1. **Melakukan Wawancara Mendalam dan Observasi:** Analis tidak bisa hanya bertanya, "Apa yang Anda inginkan?" Mereka harus melakukan wawancara terstruktur dengan berbagai pemangku kepentingan (pengguna akhir, manajer, eksekutif) untuk memahami tujuan, masalah, dan proses kerja mereka saat ini. Selain itu, melakukan observasi langsung terhadap pengguna saat mereka bekerja dapat mengungkapkan kebutuhan dan masalah yang tidak terucapkan.
    2. **Menggunakan Teknik Pemodelan Visual:** Analis dapat membuat model visual seperti Diagram Alir Data (DFD) atau diagram alur kerja untuk merepresentasikan pemahaman mereka tentang proses bisnis saat ini dan yang diusulkan. Model-model ini berfungsi sebagai alat komunikasi yang kuat. Dengan menunjukkannya kepada pengguna, analis dapat memvalidasi pemahamannya ("Apakah ini cara kerja

Anda?") dan memfasilitasi diskusi untuk mengidentifikasi kesenjangan atau kesalahan, sehingga persyaratan menjadi lebih jelas dan lengkap sebelum pengembangan dimulai.

Studi Kasus 1: Optimalisasi Logistik di PT Unilever Indonesia Tbk.

- **Konteks:** PT Unilever Indonesia Tbk. adalah salah satu perusahaan manufaktur barang konsumsi terbesar di Indonesia dengan jaringan distribusi yang sangat luas. Mengelola operasi logistik yang efisien adalah kunci keunggulan kompetitif perusahaan.
- **Masalah Sebelum Implementasi:** Sebelum implementasi Sistem Informasi Manajemen (SIM) Logistik yang terintegrasi, perusahaan menghadapi tantangan signifikan. Dengan skala operasi yang masif, pengelolaan inventaris di banyak gudang, koordinasi pengiriman ke ribuan distributor, dan pemrosesan pesanan yang masih melibatkan proses manual menciptakan potensi inefisiensi. Masalah yang mungkin timbul termasuk kelebihan atau kekurangan stok, keterlambatan pengiriman karena perencanaan rute yang kurang optimal, dan biaya operasional yang tinggi akibat proses yang memakan waktu dan rentan kesalahan.
- **Solusi (SIM Logistik):** Perusahaan mengimplementasikan sebuah SIM Logistik terpusat. Sistem ini dirancang untuk mengintegrasikan seluruh aspek rantai pasok, mulai dari manajemen inventaris,

perencanaan distribusi, hingga pemenuhan pesanan. Fitur utamanya mencakup pemantauan stok di seluruh gudang secara *real-time*, otomatisasi pemrosesan data pesanan, dan alat perencanaan distribusi yang canggih.

- **Hasil Kuantitatif yang Dicapai:** Implementasi sistem ini memberikan dampak positif yang terukur secara signifikan:
    - **Peningkatan Produktivitas:** Produktivitas karyawan yang terlibat dalam proses logistik meningkat sekitar **25%**, karena waktu yang dihemat dari otomatisasi pemrosesan data dan pengambilan keputusan yang lebih cepat.
    - **Penurunan Keterlambatan:** Tingkat keterlambatan pengiriman berhasil ditekan hingga **30%**.
    - **Percepatan Waktu Pengiriman:** Waktu pengiriman rata-rata berhasil dipangkas secara dramatis, dari rata-rata 48 jam menjadi hanya **24 jam** setelah implementasi SIM.
    - **Peningkatan Akurasi:** Tingkat kesalahan pengiriman berkurang menjadi kurang dari 5%.

Pertanyaan dan Analisis Studi Kasus 1

Pertanyaan: Analisislah bagaimana SIM Logistik di Unilever secara langsung berkontribusi pada (a) efisiensi operasional dan (b) pengambilan keputusan yang lebih baik, dengan mengacu pada peran strategis SI yang dibahas di bagian 1.4.

Analisis:

Kasus PT Unilever ini adalah contoh sempurna bagaimana sebuah sistem informasi yang dirancang dengan baik dapat

menjadi penggerak strategis, bukan hanya alat bantu administratif.

(a) Kontribusi pada Efisiensi Operasional:

SIM Logistik secara langsung meningkatkan efisiensi operasional dalam beberapa cara. Pertama, otomatisasi proses, seperti pemrosesan data pesanan, mengurangi pekerjaan manual, yang secara langsung mengarah pada peningkatan produktivitas karyawan sebesar 25%. Karyawan dapat dialihkan dari tugas entri data yang berulang ke aktivitas yang lebih strategis, seperti analisis atau penanganan pengecualian. Kedua, optimalisasi sumber daya. Dengan data stok yang real-time dan perencanaan distribusi yang lebih baik, Unilever dapat mengoptimalkan penggunaan armada transportasinya, mengurangi jarak tempuh yang tidak perlu, dan memastikan tingkat muatan truk yang lebih tinggi. Ini secara langsung mengurangi biaya bahan bakar dan pemeliharaan. Penurunan tingkat kesalahan pengiriman juga mengurangi biaya yang terkait dengan pengiriman ulang dan penanganan keluhan pelanggan.

(b) Kontribusi pada Pengambilan Keputusan yang Lebih Baik:

Sistem ini secara fundamental mengubah cara manajer logistik membuat keputusan. Sebelumnya, keputusan mungkin didasarkan pada data historis yang sudah usang atau perkiraan. Setelah implementasi, manajer memiliki akses ke informasi yang akurat dan tepat waktu. Kemampuan untuk memantau stok secara real-time memungkinkan mereka membuat keputusan pengisian ulang (replenishment) yang lebih tepat,

menghindari penumpukan stok (yang meningkatkan biaya penyimpanan) atau kehabisan stok (yang menyebabkan kehilangan penjualan). Data dari sistem juga memungkinkan analisis kinerja yang lebih mendalam. Manajer dapat mengidentifikasi rute pengiriman mana yang paling sering mengalami keterlambatan atau gudang mana yang memiliki perputaran stok paling lambat, memungkinkan mereka untuk mengambil tindakan korektif yang didukung oleh bukti. Pengurangan waktu pengiriman rata-rata dari 48 menjadi 24 jam adalah hasil langsung dari keputusan yang lebih baik dalam perencanaan dan penjadwalan, yang dimungkinkan oleh informasi berkualitas tinggi dari sistem.

Studi Kasus 2: Mengatasi Keterlambatan Pengiriman di Perusahaan Manufaktur BBI

- **Konteks:** PT BBI adalah perusahaan yang bergerak di bidang jasa rekayasa, manufaktur, dan konstruksi. Salah satu masalah kronis yang dihadapi perusahaan adalah keterlambatan pengiriman produk kepada pelanggan.
- **Masalah:** Sebuah proyek yang direncanakan selesai dalam 12 bulan seringkali mengalami keterlambatan signifikan. Investigasi awal mengidentifikasi dua akar penyebab utama: (1) **Keterlambatan bahan baku** dari pemasok, dan (2) **Penjadwalan produksi yang tidak akurat**. Masalah-masalah ini timbul karena sistem informasi yang terfragmentasi; tidak ada aliran informasi yang lancar antara departemen pengadaan (yang memesan bahan baku), departemen perencanaan produksi, dan lantai pabrik.

- **Dampak Bisnis:** Keterlambatan yang terus-menerus ini merusak reputasi perusahaan secara serius. Tingkat pemesanan ulang dari pelanggan yang sudah ada sangat rendah, hanya sekitar 50%. Selain itu, perusahaan sangat kesulitan mendapatkan pelanggan baru, dengan rata-rata hanya 1-2 pelanggan baru per tahun selama lima tahun terakhir. Reputasi yang buruk ini secara langsung mengancam profitabilitas dan keberlanjutan bisnis perusahaan.

Pertanyaan dan Analisis Studi Kasus 2

Pertanyaan: Anda adalah seorang analis sistem yang baru disewa oleh manajemen PT BBI untuk mengatasi masalah ini. Berdasarkan deskripsi masalah, langkah-langkah analisis apa yang akan Anda lakukan pertama kali? Siapa saja pemangku kepentingan utama yang perlu Anda wawancarai untuk mendapatkan pemahaman yang lengkap tentang masalah ini?

Analisis:

Kasus ini menempatkan Anda pada posisi seorang analis sistem di awal sebuah proyek. Tujuannya adalah untuk menerapkan pendekatan terstruktur dari fase analisis untuk memahami masalah secara mendalam sebelum melompat ke solusi.

Langkah-langkah Analisis Pertama:

Tujuan utama pada tahap awal adalah **memahami dan mendokumentasikan sistem saat ini (*as-is system*)** dan mengidentifikasi titik-titik masalah secara detail. Berdasarkan prinsip-prinsip analisis sistem, langkah-langkah pertama yang akan saya lakukan adalah:

1. **Melakukan Investigasi Awal (Preliminary Investigation):** Meskipun manajemen telah memberikan gambaran umum, saya perlu memvalidasi dan memperdalam pemahaman ini. Saya akan meninjau dokumen-dokumen yang ada, seperti laporan keterlambatan proyek, keluhan pelanggan, dan prosedur operasional standar (jika ada) untuk pengadaan dan penjadwalan. Tujuannya adalah untuk mendapatkan gambaran fakta awal tentang skala dan frekuensi masalah.
2. **Mengidentifikasi Pemangku Kepentingan (Stakeholder Identification):** Saya akan membuat daftar semua individu atau departemen yang terlibat dalam atau terpengaruh oleh proses dari pemesanan bahan baku hingga pengiriman produk. Ini penting untuk memastikan tidak ada perspektif yang terlewat.
3. **Merencanakan dan Melakukan Wawancara:** Saya akan menjadwalkan sesi wawancara dengan pemangku kepentingan kunci yang telah diidentifikasi. Tujuannya bukan hanya untuk mendengar keluhan, tetapi untuk memahami alur kerja, aliran informasi (atau ketiadaannya), aturan keputusan, dan masalah spesifik yang mereka hadapi dari sudut pandang mereka.
4. **Menganalisis Proses Bisnis Saat Ini:** Selama dan setelah wawancara, saya akan mulai memetakan proses bisnis saat ini. Saya akan menggunakan alat pemodelan seperti diagram alur kerja atau Diagram Alir Data (DFD) level konteks untuk secara visual merepresentasikan bagaimana informasi dan material mengalir antar departemen. Ini akan membantu mengidentifikasi

bottlenecks, redundansi, dan titik di mana komunikasi putus.

Pemangku Kepentingan Utama untuk Diwawancarai:

Untuk mendapatkan gambaran 360 derajat, saya perlu berbicara dengan orang-orang dari berbagai level dan fungsi:

1. **Manajer Departemen Pengadaan (Purchasing):** Untuk memahami proses pemesanan bahan baku, bagaimana mereka berinteraksi dengan pemasok, bagaimana mereka mendapatkan informasi kebutuhan dari produksi, dan apa penyebab utama keterlambatan dari sisi pemasok.
2. **Manajer Perencanaan Produksi (Production Planning):** Untuk memahami bagaimana mereka membuat jadwal produksi, informasi apa yang mereka gunakan sebagai dasar, seberapa akurat perkiraan waktu mereka, dan bagaimana mereka menangani situasi ketika bahan baku terlambat.
3. **Supervisor Lantai Pabrik (Shop Floor Supervisor):** Untuk mendapatkan perspektif operasional. Apakah mereka menerima jadwal yang realistis? Apakah mereka sering harus berhenti bekerja karena menunggu bahan? Informasi apa yang mereka butuhkan tetapi tidak mereka dapatkan?
4. **Manajer Penjualan/Akun (Sales/Account Manager):** Untuk memahami dampak masalah ini dari sisi pelanggan. Komunikasi seperti apa yang mereka lakukan dengan pelanggan ketika terjadi keterlambatan? Apa umpan balik spesifik dari pelanggan?

5. **Staf Gudang (Warehouse Staff):** Untuk memahami proses penerimaan dan penyimpanan bahan baku, serta bagaimana informasi stok dikelola dan dibagikan.
6. **Manajemen Puncak (CEO/Direktur Operasi):** Untuk memahami tujuan strategis, batasan anggaran, dan tingkat urgensi proyek perbaikan ini dari perspektif mereka.

Dengan melakukan langkah-langkah analisis ini secara sistematis dan berbicara dengan semua pemangku kepentingan kunci, seorang analis sistem dapat membangun pemahaman yang solid dan berbasis fakta tentang akar masalah, yang merupakan pondasi penting untuk merancang solusi sistem informasi yang efektif nantinya.

Daftar Bahan Bacaan

Untuk memperdalam pemahaman Anda tentang konsep-konsep yang dibahas dalam bab ini dan untuk mempersiapkan diri menghadapi topik-topik selanjutnya, berikut adalah daftar bahan bacaan yang sangat dianjurkan.

Buku Teks Utama (Wajib)

Buku-buku ini merupakan referensi standar internasional dalam bidang Analisis dan Perancangan Sistem Informasi dan akan menjadi acuan utama sepanjang mata kuliah ini.

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.

- Laudon, K. C., & Laudon, J. P. (Edisi terbaru). *Management Information Systems: Managing the Digital Firm*. Pearson.

Bacaan Tambahan (Dianjurkan)

Materi berikut memberikan wawasan tambahan yang berharga, baik berupa data industri maupun konteks lokal.

- The Standish Group. (2020). *CHAOS Report: Beyond Infinity*. Laporan ini adalah sumber data primer yang paling sering dikutip mengenai statistik keberhasilan dan kegagalan proyek TI. Membaca ringkasannya akan memberikan konteks dunia nyata yang kuat tentang pentingnya APSI.
- Artikel jurnal dan publikasi industri terkini yang membahas tren dalam *Cloud Computing*, *Artificial Intelligence*, dan Transformasi Digital. Sumber-sumber seperti publikasi dari Gartner, Forrester, atau jurnal akademik di bidang sistem informasi akan membantu Anda tetap mengikuti perkembangan terbaru yang membentuk masa depan profesi ini.

Bab 2: Konsep Dasar Sistem

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan definisi sistem dari berbagai sudut pandang (prosedur dan komponen) dengan mengacu pada pandangan para ahli, termasuk penggagas Teori Sistem Umum, Ludwig von Bertalanffy.
- Mengidentifikasi dan menguraikan delapan karakteristik fundamental yang mendefinisikan sebuah sistem, mulai dari komponen hingga sasaran.
- Mengklasifikasikan berbagai jenis sistem berdasarkan sifatnya (abstrak vs. fisik, alamiah vs. buatan, deterministik vs. probabilistik).
- Membedakan secara mendalam antara sistem terbuka dan sistem tertutup, serta menganalisis implikasi konsep entropi pada kelangsungan hidup sistem.
- Menjelaskan tahapan-tahapan dalam siklus hidup sistem secara umum (kelahiran, pertumbuhan, kematangan, penurunan) dan membedakannya dari siklus hidup pengembangan sistem (SDLC).

Pada Bab 1, kita telah membangun pemahaman mengenai sistem informasi sebagai sebuah entitas spesifik yang mengolah data menjadi informasi. Sekarang, kita akan mengambil satu langkah mundur untuk memahami konsep yang lebih fundamental dan universal yang menjadi dasar dari "sistem

informasi", yaitu konsep **sistem** itu sendiri. Memahami konsep dasar sistem adalah seperti seorang mahasiswa kedokteran yang mempelajari anatomi dan fisiologi dasar sebelum mendalami organ-organ spesifik. Tanpa pemahaman yang kokoh tentang apa itu sistem, bagaimana komponennya berinteraksi, dan bagaimana ia berperilaku, upaya kita untuk menganalisis dan merancang sistem yang lebih kompleks akan menjadi rapuh.

Dalam bab ini, kita akan membedah anatomi sistem, mengidentifikasi karakteristik universal yang dimiliki oleh semua sistem, mulai dari sistem tata surya hingga sistem ekonomi pasar. Kita akan belajar mengklasifikasikan sistem ke dalam berbagai jenis, yang akan membantu kita memahami perilakunya dengan lebih baik. Kita akan mengeksplorasi perbedaan krusial antara sistem yang berinteraksi dengan lingkungannya (sistem terbuka) dan yang terisolasi (sistem tertutup), sebuah perbedaan yang memiliki implikasi mendalam bagi kelangsungan hidup organisasi.

Terakhir, kita akan melihat bahwa sistem, terutama sistem sosial dan organisasi, memiliki siklus hidupnya sendiri, layaknya organisme hidup. Penguasaan konsep-konsep ini akan memberikan Anda "kacamata sistem" yang memungkinkan Anda melihat dunia, masalah, dan solusi dalam kerangka kerja yang terstruktur dan holistik.

2.1 Definisi dan Karakteristik Sistem

Sebelum kita dapat menganalisis atau merancang sebuah sistem, kita harus memiliki definisi yang jelas tentang apa itu "sistem". Konsep ini begitu fundamental sehingga sering

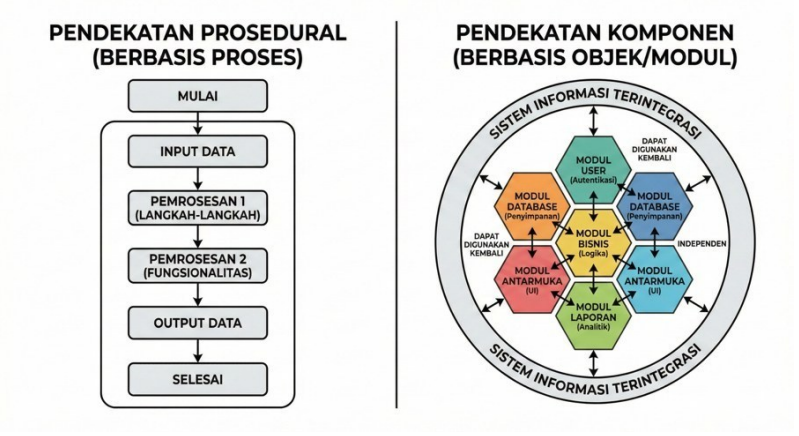
digunakan dalam percakapan sehari-hari, namun dalam konteks teknis, kita memerlukan pemahaman yang lebih presisi.

Definisi Sistem: Dua Pendekatan Utama

Para ahli telah mendefinisikan sistem dari berbagai perspektif, yang secara umum dapat dikelompokkan menjadi dua pendekatan utama: pendekatan prosedur dan pendekatan komponen.

1. **Pendekatan Prosedur:** Pendekatan ini memandang sistem sebagai jaringan kerja dari prosedur-prosedur yang saling berhubungan. Fokusnya adalah pada urutan operasi atau langkah-langkah yang dilakukan untuk mencapai tujuan. Sistem adalah "suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, lalu berkumpul bersama-sama untuk melakukan kegiatan atau untuk melakukan sasaran tertentu". Ahli lain menyatakan sistem adalah "suatu jaringan prosedur yang dibuat menurut pola yang terpadu untuk melaksanakan kegiatan pokok perusahaan". Pendekatan ini sangat relevan dalam analisis proses bisnis, di mana kita memetakan alur kerja langkah demi langkah.
2. **Pendekatan Komponen:** Pendekatan ini mendefinisikan sistem sebagai kumpulan dari komponen atau elemen yang saling berinteraksi dan bekerja sama untuk mencapai tujuan. Fokusnya adalah pada bagian-bagian pembentuk sistem dan hubungan di antara mereka. Sistem didefinisikan sebagai "kumpulan/group dari sub sistem/ bagian/ komponen apapun baik fisik atau pun non fisik yang saling berhubungan satu sama lain dan bekerja sama secara harmonis untuk mencapai satu tujuan tertentu". Romney dan Steinbart menyatakan bahwa sistem adalah "rangkaian dari dua atau lebih komponen-komponen yang

saling berkaitan dan saling berinteraksi untuk mencapai tujuan".



Gambar 4. Pendekatan Prosedural dan Pendekatan Komponen

Kedua pendekatan ini tidak bertentangan, melainkan saling melengkapi. Sebuah sistem memiliki **struktur** (komponen-komponennya) dan **proses** (prosedur kerjanya). Pemikir besar di balik teori ini, Ludwig von Bertalanffy, penggagas **Teori Sistem Umum (General System Theory)**, mendefinisikan sistem sebagai "sekumpulan unsur yang saling terikat dalam suatu antar relasi di antara unsur-unsur tersebut dengan lingkungan". Esensi dari semua definisi ini terangkum dalam tiga kata kunci: **komponen**, **interaksi**, dan **tujuan**.

Karakteristik Fundamental Sebuah Sistem

Untuk dapat disebut sebagai sebuah sistem, suatu entitas harus memiliki karakteristik atau sifat-sifat tertentu. Memahami karakteristik ini membantu kita dalam mengidentifikasi batas-

batas sistem dan menganalisis perilakunya. Setidaknya ada delapan karakteristik utama yang dimiliki oleh setiap sistem.

Tabel 4. Delapan Karakteristik Sistem

Karakteristik	Deskripsi	Contoh dalam Sistem Universitas
1. Komponen (Component)	Bagian-bagian atau elemen-elemen yang membentuk sistem. Komponen ini dapat berupa subsistem yang lebih kecil.	Fakultas, departemen, mahasiswa, dosen, staf administrasi, perpustakaan.
2. Batasan (Boundary)	Daerah yang memisahkan sistem dari lingkungannya. Batasan ini mendefinisikan ruang lingkup sistem dan membedakannya dari sistem lain.	Pagar fisik kampus, peraturan penerimaan mahasiswa baru, kriteria kelulusan.
3. Lingkungan Luar (Environment)	Segala sesuatu di luar batasan sistem yang dapat memengaruhi operasi sistem, baik secara positif maupun negatif.	Pemerintah (regulasi pendidikan), industri (kebutuhan tenaga kerja), masyarakat, sekolah menengah (calon mahasiswa).
4. Penghubung (Interface)	Media yang memungkinkan interaksi dan aliran sumber daya (data, energi, material) antara satu komponen dengan komponen lainnya.	Sistem informasi akademik yang menghubungkan data mahasiswa dengan departemen keuangan, rapat koordinasi antar fakultas.
5. Masukan (Input)	Energi, material, atau data yang dimasukkan ke dalam sistem untuk diolah. Terdiri dari *maintenance input* (untuk menjaga sistem tetap berjalan) dan *signal input* (untuk diproses).	Calon mahasiswa baru (signal input), dana operasional dari pemerintah (maintenance input), data pendaftaran.

Karakteristik	Deskripsi	Contoh dalam Sistem Universitas
6. Proses (Process)	Bagian yang mengubah masukan menjadi keluaran. Ini adalah serangkaian aktivitas atau transformasi yang terjadi di dalam sistem.	Kegiatan belajar mengajar, penelitian, ujian, proses administrasi, sidang skripsi.
7. Keluaran (Output)	Hasil dari energi yang telah diolah oleh sistem. Keluaran ini bisa menjadi masukan bagi sistem lain atau produk akhir.	Lulusan (sarjana, magister), publikasi ilmiah, pengabdian kepada masyarakat, laporan keuangan.
8. Sasaran/Tujuan (Objective/Goal)	Alasan keberadaan sistem. Sasaran inilah yang mengarahkan semua komponen dan proses untuk bekerja sama secara harmonis.	Menghasilkan lulusan yang kompeten, mengembangkan ilmu pengetahuan, dan memberikan kontribusi kepada masyarakat (Tri Dharma Perguruan Tinggi).

Seorang analis sistem harus mampu mengidentifikasi kedelapan karakteristik ini untuk setiap sistem yang sedang dipelajari. Kegagalan dalam mendefinisikan batasan, memahami lingkungan, atau mengidentifikasi tujuan akan menyebabkan analisis yang tidak lengkap dan solusi yang tidak efektif.

2.2 Jenis-jenis Sistem

Sistem hadir dalam berbagai bentuk dan sifat. Dengan mengklasifikasikan sistem, kita dapat menggunakan kerangka kerja yang tepat untuk menganalisisnya. Sistem dapat diklasifikasikan dari beberapa sudut pandang yang berbeda.

1. Sistem Abstrak vs. Sistem Fisik

Klasifikasi ini didasarkan pada wujud dari sistem tersebut.

- **Sistem Abstrak (Abstract System):** Adalah sistem yang tidak tampak secara fisik, melainkan berupa pemikiran, gagasan, atau konsep. Contohnya adalah **sistem teologi** (kerangka pemikiran tentang hubungan manusia dengan Tuhan) atau **teori ekonomi** (seperangkat ide dan model yang menjelaskan perilaku pasar). Sistem ini tidak dapat disentuh, tetapi elemen dan hubungannya dapat diuraikan.
- **Sistem Fisik (Physical System):** Adalah sistem yang ada secara fisik dan dapat diamati. Komponen-komponennya adalah objek material. Contohnya sangat banyak di sekitar kita, seperti **sistem komputer** (terdiri dari hardware yang nyata), **sistem transportasi** (terdiri dari kendaraan, jalan, dan terminal), atau **sistem produksi** di pabrik.

2. Sistem Alamiah vs. Sistem Buatan Manusia

Klasifikasi ini didasarkan pada asal-usul terbentuknya sistem.

- **Sistem Alamiah (Natural System):** Adalah sistem yang terjadi melalui proses alam, tanpa campur tangan manusia. Sistem ini ada dengan sendirinya di alam semesta. Contohnya termasuk **sistem tata surya**, **sistem peredaran darah** dalam tubuh makhluk hidup, dan **ekosistem hutan hujan**.
- **Sistem Buatan Manusia (Human-Made System):** Adalah sistem yang dirancang dan dibuat oleh manusia untuk mencapai tujuan tertentu. Sebagian besar sistem yang kita analisis dalam bidang teknik industri dan sistem informasi masuk dalam kategori ini. Contohnya adalah **sistem informasi akuntansi**, **sistem manufaktur mobil**, dan **sistem**

pemerintahan sebuah negara. Ketika sistem ini melibatkan interaksi antara manusia dan mesin, ia sering disebut *human-machine system*.

3. Sistem Deterministik vs. Sistem Probabilistik

Klasifikasi ini didasarkan pada tingkat kepastian atau prediktabilitas perilaku sistem.

- **Sistem Deterministik (Deterministic System):** Adalah sistem yang operasinya dapat diprediksi dengan pasti. Jika kita mengetahui kondisi awal dan inputnya, kita dapat menentukan outputnya secara akurat. Interaksi antar komponennya mengikuti aturan yang jelas. Contoh utamanya adalah **sistem komputer** yang menjalankan program; dengan input yang sama, ia akan selalu menghasilkan output yang sama.
- **Sistem Probabilistik (Probabilistic System):** Adalah sistem yang kondisi masa depannya tidak dapat diprediksi dengan pasti karena mengandung unsur ketidakpastian atau probabilitas. Meskipun kita mengetahui kondisi awalnya, outputnya bisa bervariasi. Contohnya adalah **sistem cuaca**, **sistem persediaan barang** (yang dipengaruhi oleh permintaan pelanggan yang acak), dan **perilaku pasar saham**.

Dalam praktiknya, banyak sistem merupakan kombinasi dari klasifikasi-klasifikasi ini. Sebuah sistem informasi manufaktur, misalnya, adalah sistem fisik, buatan manusia, dan sebagian besar bersifat deterministik (dalam hal pemrosesan data) tetapi

juga dipengaruhi oleh elemen probabilistik (seperti kerusakan mesin atau fluktuasi permintaan).

2.3 Sistem Terbuka vs. Sistem Tertutup: Interaksi dengan Lingkungan

Salah satu klasifikasi sistem yang paling fundamental dan memiliki implikasi mendalam adalah pembagian antara sistem terbuka dan sistem tertutup. Perbedaan utama terletak pada bagaimana sistem berinteraksi dengan lingkungannya.

- **Sistem Tertutup (Closed System):** Secara teoretis, sistem tertutup adalah sistem yang terisolasi sepenuhnya dari lingkungannya. Ia tidak menerima masukan (energi, materi, atau informasi) dari luar dan tidak memberikan keluaran ke lingkungannya. Sistem ini bekerja secara otomatis tanpa campur tangan dari pihak luar. Contoh teoretis yang sering digunakan adalah
  reaksi kimia dalam tabung reaksi yang terisolasi sempurna. Dalam praktiknya, sangat sedikit sistem yang benar-benar tertutup. Kebanyakan adalah sistem yang *relatif* tertutup (*relatively closed system*).
- **Sistem Terbuka (Open System):** Sebaliknya, sistem terbuka adalah sistem yang secara aktif berinteraksi dengan lingkungannya. Ia menerima masukan dari lingkungan, memprosesnya, dan menghasilkan keluaran yang dikirim kembali ke lingkungan. Sistem terbuka memiliki kemampuan untuk beradaptasi dengan perubahan di lingkungannya untuk mempertahankan eksistensinya. Hampir semua sistem yang menarik untuk kita pelajari, terutama sistem biologis, sosial, dan organisasi, adalah sistem terbuka. Sebuah **perusahaan bisnis** adalah contoh

klasik sistem terbuka: ia menerima input (bahan baku, modal, tenaga kerja) dari lingkungan, memprosesnya, dan menghasilkan output (produk, jasa) untuk pasar.

Konsep Entropi dan Kelangsungan Hidup Sistem

Perbedaan antara sistem terbuka dan tertutup menjadi sangat penting ketika kita memperkenalkan konsep **entropi**. Dalam termodinamika, entropi adalah ukuran tingkat ketidakteraturan, keacakan, atau kekacauan dalam sebuah sistem. Hukum Kedua Termodinamika menyatakan bahwa dalam sistem yang terisolasi (tertutup), entropi akan selalu cenderung meningkat seiring waktu. Artinya, sistem tertutup secara alami bergerak dari keteraturan menuju kekacauan, kerusakan, dan akhirnya "kematian" atau kesetimbangan statis.

Sistem terbuka, di sisi lain, memiliki kemampuan unik untuk melawan tren entropi ini. Dengan cara mengimpor energi dan informasi yang lebih teratur dari lingkungannya, sistem terbuka dapat mempertahankan (bahkan meningkatkan) keteraturan internalnya. Proses ini disebut **negentropi** (entropi negatif). Implikasinya bagi organisasi sangat besar. Sebuah perusahaan yang bertindak sebagai sistem tertutup, mengabaikan perubahan selera pelanggan, inovasi teknologi dari pesaing, atau regulasi baru dari pemerintah, akan mengalami peningkatan entropi. Proses bisnisnya menjadi usang, produknya tidak lagi relevan, dan akhirnya akan gagal. Sebaliknya, perusahaan yang bertindak sebagai sistem terbuka, secara aktif memindai lingkungannya, beradaptasi dengan tren pasar, dan terus berinovasi, dapat mempertahankan keteraturannya, bertumbuh, dan bertahan dalam jangka

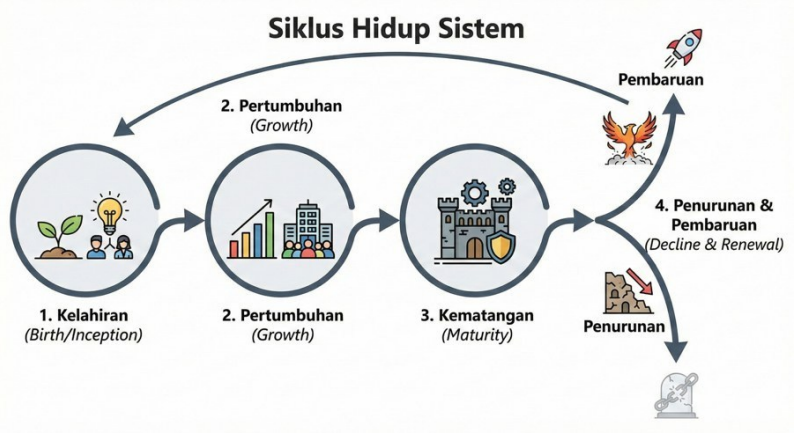
panjang. Oleh karena itu, dari sudut pandang analisis sistem, setiap organisasi harus dipandang dan dikelola sebagai sistem terbuka yang dinamis.

Tabel 5. Perbandingan Sistem Tertutup dan Sistem Terbuka

Fitur	Sistem Tertutup	Sistem Terbuka
Interaksi Lingkungan	Tidak ada atau minimal.	Aktif dan saling memengaruhi.
Batasan (Boundary)	Kaku dan tidak dapat ditembus.	Fleksibel dan permeabel.
Entropi	Cenderung meningkat menuju kekacauan.	Dapat dilawan dengan mengimpor energi (negentropi).
Adaptasi	Tidak mampu beradaptasi.	Mampu beradaptasi terhadap perubahan.
Fokus Analisis	Pada mekanisme internal yang efisien.	Pada hubungan antara sistem dan lingkungannya.
Contoh	Reaksi kimia terisolasi (teoretis).	Organisasi bisnis, makhluk hidup, ekosistem.

2.4 Siklus Hidup Sistem

Sama seperti organisme hidup, sistem, terutama sistem organisasi dan sosial, cenderung mengikuti pola perkembangan yang dapat diprediksi yang dikenal sebagai **siklus hidup sistem (system life cycle)**.



Gambar 5. Siklus hidup sistem

Penting untuk membedakan konsep ini dari *Siklus Hidup Pengembangan Sistem (System Development Life Cycle - SDLC)*, yang merupakan metodologi spesifik untuk membangun sistem informasi dan akan kita bahas secara mendalam di bab berikutnya. Siklus hidup sistem yang dibahas di sini adalah kerangka konseptual yang lebih luas untuk memahami evolusi sebuah sistem (misalnya, sebuah perusahaan) dari waktu ke waktu. Model siklus hidup organisasi umumnya mengidentifikasi empat hingga lima tahapan utama, meskipun dengan nama yang sedikit berbeda-beda.

1. Tahap Kelahiran (Birth/Inception):

   Ini adalah tahap awal di mana sistem (organisasi) diciptakan. Biasanya dimulai dari sebuah ide atau gagasan oleh seorang pendiri (*founder*). Fokus utamanya adalah pada penciptaan produk/layanan dan bertahan hidup. Organisasi pada tahap ini seringkali kecil, strukturnya informal dan fleksibel, serta pengambilan keputusannya sangat terpusat pada pendiri. Tantangan terbesarnya adalah mendapatkan

sumber daya yang cukup (modal, pelanggan pertama) dan menghindari "kematian bayi" (*infant mortality*), di mana banyak organisasi baru gagal.

2. Tahap Pertumbuhan (Growth):

   Jika berhasil melewati tahap kelahiran, sistem akan memasuki fase pertumbuhan. Organisasi mulai memperluas operasinya, meningkatkan penjualan, dan menambah jumlah karyawan. Pada tahap ini, mulai muncul kebutuhan akan struktur yang lebih formal, prosedur standar, dan pendelegasian wewenang. Krisis yang sering muncul adalah krisis kepemimpinan atau krisis otonomi, di mana pendiri harus belajar untuk melepaskan sebagian kendali dan mempercayai manajer lain.

3. Tahap Kematangan (Maturity):

   Pada tahap ini, sistem telah mencapai stabilitas. Organisasi memiliki struktur yang lebih birokratis, prosedur yang mapan, dan pangsa pasar yang kuat. Fokus bergeser dari pertumbuhan cepat ke efisiensi, profitabilitas, dan mempertahankan posisi pasar. Bahaya utama pada tahap ini adalah rasa puas diri dan keengganan untuk berubah. Organisasi bisa menjadi terlalu kaku dan lambat dalam merespons perubahan lingkungan.

4. Tahap Penurunan (Decline) dan Pembaruan (Renewal):

   Jika sistem gagal beradaptasi pada tahap kematangan, ia akan memasuki tahap penurunan. Ini ditandai dengan menurunnya penjualan, profitabilitas, dan pangsa pasar.

Entropi mulai meningkat. Pada titik ini, sistem menghadapi persimpangan jalan:

- **Penurunan Berkelanjutan:** Jika tidak ada tindakan signifikan yang diambil, organisasi akan terus merosot, menghadapi krisis, dan akhirnya bisa menuju "kematian" (kebangkrutan atau likuidasi).
- **Pembaruan (Renewal):** Organisasi yang proaktif dapat menghindari kematian dengan melakukan pembaruan. Ini melibatkan inovasi radikal, restrukturisasi, pengembangan produk baru, atau memasuki pasar baru. Proses ini pada dasarnya adalah upaya untuk memulai kembali siklus hidup, seringkali dengan melahirkan "sistem" baru di dalam sistem yang lama.

Memahami siklus hidup ini penting bagi seorang analis sistem karena jenis masalah dan kebutuhan sistem informasi sangat bervariasi di setiap tahap. Sistem yang dibutuhkan oleh sebuah *startup* (tahap kelahiran) sangat berbeda dengan sistem yang dibutuhkan oleh perusahaan multinasional yang sudah matang. Analis yang baik harus mampu mendiagnosis di tahap mana sebuah organisasi berada untuk dapat merancang solusi yang paling sesuai.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Pilih sebuah sistem yang Anda kenal baik (misalnya, klub sepak bola, restoran favorit Anda, atau sistem transportasi umum di kota Anda). Identifikasi dan jelaskan secara singkat kedelapan karakteristik sistem (komponen, batasan, lingkungan, dll.) yang berlaku pada sistem pilihan Anda.
2. Klasifikasikan sistem-sistem berikut berdasarkan tiga pasangan kategori (Abstrak/Fisik, Alamiah/Buatan, Deterministik/Probabilistik): (a) Sistem pernapasan manusia, (b) Teori relativitas Einstein, (c) Mesin ATM. Berikan alasan singkat untuk setiap klasifikasi Anda.
3. Mengapa sebuah perusahaan yang beroperasi sebagai "sistem tertutup" hampir pasti akan gagal dalam jangka panjang? Hubungkan jawaban Anda dengan konsep entropi.

Solusi dan Pembahasan Soal Konseptual

1. **Analisis Karakteristik Sistem pada Restoran Cepat Saji:**
    - **Komponen:** Koki, kasir, manajer, peralatan masak, bahan makanan, meja, kursi, sistem kasir (POS).
    - **Batasan:** Dinding fisik restoran, jam operasional, menu yang ditawarkan.

- **Lingkungan Luar:** Pemasok bahan baku, pelanggan, pesaing, dinas kesehatan (regulasi), tren kuliner.
- **Penghubung:** Sistem pesanan yang menghubungkan kasir dengan dapur, komunikasi verbal antara staf.
- **Masukan:** Bahan makanan mentah, pesanan pelanggan, uang pembayaran, tenaga kerja.
- **Proses:** Memasak makanan, merakit pesanan, menerima pembayaran, membersihkan area makan.
- **Keluaran:** Makanan siap saji, struk pembayaran, kepuasan (atau ketidakpuasan) pelanggan, limbah.
- **Sasaran/Tujuan:** Menyediakan makanan yang cepat dan lezat kepada pelanggan untuk menghasilkan keuntungan.

2. **Klasifikasi Sistem:**
    - **(a) Sistem Pernapasan Manusia:**
        - **Fisik:** Terdiri dari organ nyata (paru-paru, trakea).
        - **Alamiah:** Terjadi secara alami, bukan buatan manusia.
        - **Probabilistik:** Meskipun sebagian besar otomatis, kinerjanya dapat dipengaruhi oleh faktor tak terduga seperti penyakit atau polusi.
    - **(b) Teori Relativitas Einstein:**
        - **Abstrak:** Berupa seperangkat ide, prinsip, dan persamaan matematika, tidak memiliki wujud fisik.

        - **Buatan Manusia:** Merupakan hasil pemikiran dan perumusan oleh manusia (Einstein).
        - **Deterministik:** Persamaan-persamaannya memberikan hasil yang pasti jika variabel inputnya diketahui.
    - **(c) Mesin ATM:**
        - **Fisik:** Terdiri dari perangkat keras yang nyata (layar, tombol, dispenser uang).
        - **Buatan Manusia:** Dirancang dan diproduksi oleh manusia.
        - **Deterministik:** Untuk perintah dan input yang sama (misalnya, tarik tunai Rp 500.000), mesin akan selalu menjalankan urutan proses yang sama dan memberikan hasil yang sama.
3. Kegagalan Sistem Tertutup dan Entropi:

    Sebuah perusahaan yang beroperasi sebagai sistem tertutup akan gagal karena ia mengabaikan interaksinya dengan lingkungan. Menurut Hukum Kedua Termodinamika, sistem tertutup cenderung bergerak menuju entropi yang lebih tinggi, yaitu keadaan kekacauan, disorganisasi, dan akhirnya berhenti berfungsi. Dalam konteks bisnis, ini berarti:

    - Perusahaan tidak mendapatkan "energi" baru dari lingkungan dalam bentuk informasi tentang tren pasar, teknologi baru, atau perubahan selera pelanggan.

- Akibatnya, produk dan proses internalnya menjadi usang dan tidak efisien (peningkatan entropi).
- Tanpa kemampuan beradaptasi (sebuah ciri sistem terbuka), perusahaan kehilangan daya saingnya, ditinggalkan oleh pelanggan, dan akhirnya mengalami kerugian dan kebangkrutan, yang merupakan bentuk "kematian" organisasi.

Studi Kasus: Evolusi "Kopi Kita"

- **Konteks:** "Kopi Kita" didirikan oleh dua orang sahabat, Budi dan Ani, sebagai sebuah kedai kopi kecil di dekat kampus. Pada awalnya (Tahap Kelahiran), Budi menangani semua urusan resep dan operasional bar, sementara Ani mengurus pemasaran media sosial dan keuangan. Mereka hanya memiliki dua barista paruh waktu. Sistem mereka sangat sederhana: pesanan dicatat manual di buku, dan laporan keuangan dibuat di spreadsheet setiap akhir pekan.
- **Perkembangan:** Dalam dua tahun, "Kopi Kita" menjadi sangat populer (Tahap Pertumbuhan). Mereka membuka cabang kedua dan mempekerjakan manajer untuk setiap lokasi. Jumlah karyawan membengkak menjadi 20 orang. Sistem manual menjadi kacau: pesanan sering salah, stok bahan baku tidak terkontrol, dan laporan keuangan sering terlambat. Budi dan Ani merasa kehilangan kendali.
- **Tantangan Saat Ini:** Lima tahun berjalan, "Kopi Kita" adalah merek yang mapan dengan lima cabang (Tahap Kematangan). Mereka sudah memiliki sistem kasir

(POS) yang terintegrasi dan manajer operasional pusat. Namun, beberapa pesaing baru muncul dengan konsep yang lebih segar, menawarkan biji kopi eksotis dan metode seduh yang inovatif. Pertumbuhan penjualan "Kopi Kita" melambat, dan beberapa pelanggan setia mulai beralih ke pesaing. Budi ingin tetap pada menu klasik yang terbukti laku, sementara Ani merasa mereka harus berinovasi untuk bertahan.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan:

1. Gunakan konsep siklus hidup organisasi untuk menjelaskan evolusi "Kopi Kita" dari awal berdiri hingga saat ini. Krisis apa yang mereka hadapi saat transisi dari tahap kelahiran ke pertumbuhan?
2. Analisislah situasi "Kopi Kita" saat ini menggunakan perspektif sistem terbuka. Ancaman apa yang datang dari lingkungan luar? Mengapa argumen Ani untuk berinovasi sejalan dengan prinsip melawan entropi?

Analisis:

1. Evolusi dan Krisis Transisi:

   "Kopi Kita" dengan jelas mengikuti siklus hidup organisasi.

   - **Kelahiran:** Dimulai sebagai usaha kecil dengan struktur informal dan kendali terpusat pada pendiri (Budi dan Ani).

- **Pertumbuhan:** Ditandai dengan ekspansi (cabang baru) dan peningkatan jumlah karyawan. Di sinilah mereka menghadapi **krisis otonomi/kendali**. Sistem manual yang cukup untuk skala kecil menjadi sumber kekacauan pada skala yang lebih besar. Kebutuhan akan pendelegasian (manajer cabang) dan sistem yang lebih formal (seperti sistem POS) muncul sebagai solusi untuk mengatasi krisis ini dan memungkinkan pertumbuhan lebih lanjut.
- **Kematangan:** Saat ini, mereka berada di tahap kematangan dengan operasi yang stabil dan sistem yang lebih mapan.

2. Analisis Sistem Terbuka dan Entropi:

   Sebagai sistem terbuka, "Kopi Kita" tidak bisa lepas dari lingkungannya.

   - **Ancaman Lingkungan:** Ancaman utama datang dari **pesaing baru** yang membawa inovasi produk (biji kopi eksotis) dan proses (metode seduh baru). Ini adalah perubahan dalam lingkungan kompetitif yang mengancam posisi pasar "Kopi Kita". Perubahan **selera pelanggan**, yang mungkin mencari pengalaman baru, juga merupakan ancaman dari lingkungan.
   - **Melawan Entropi:** Sikap Budi yang ingin mempertahankan status quo adalah sikap yang mendekati sistem tertutup. Jika dibiarkan, ini akan menyebabkan "Kopi Kita" mengalami peningkatan **entropi**: menu menjadi

membosankan, merek dianggap kuno, dan bisnis perlahan-lahan menurun. Argumen Ani untuk **berinovasi** adalah manifestasi dari perilaku sistem terbuka yang sehat. Dengan berinovasi (mengimpor "energi" baru dalam bentuk ide, produk, dan proses dari lingkungan), "Kopi Kita" dapat beradaptasi, mempertahankan relevansinya, dan melawan tren alami menuju penurunan. Ini adalah upaya untuk melakukan **pembaruan (renewal)** agar tetap berada di puncak atau memulai siklus pertumbuhan yang baru.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Laudon, K. C., & Laudon, J. P. (Edisi terbaru). *Management Information Systems: Managing the Digital Firm*. Pearson.

Bacaan Tambahan (Dianjurkan)

- Bertalanffy, L. von. (1968). *General System Theory: Foundations, Development, Applications*. George Braziller. (Untuk pemahaman mendalam tentang asal-usul teori sistem dari sumber primernya).

- Jogiyanto, H.M. (2020). *Sistem Informasi: Konsep dan Aplikasi*. Andi Offset. (Menyediakan pembahasan konsep-konsep dasar sistem dengan banyak contoh aplikasi).

Bab 3: Metodologi Pengembangan Sistem

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan definisi dan pentingnya metodologi pengembangan sistem sebagai kerangka kerja terstruktur untuk mengelola proyek sistem informasi.
- Mengidentifikasi, menjelaskan, dan membandingkan tahapan, kelebihan, serta kekurangan dari metodologi pengembangan sistem tradisional (prediktif) seperti Model Waterfall.
- Mengidentifikasi, menjelaskan, dan membandingkan tahapan, kelebihan, serta kekurangan dari metodologi pengembangan sistem evolusioner seperti Model Spiral, dengan penekanan pada manajemen risiko.
- Memahami filosofi, nilai-nilai inti, dan prinsip-prinsip di balik pendekatan Agile sebagai respons terhadap keterbatasan model tradisional.
- Menjelaskan kerangka kerja Agile yang populer, seperti Scrum (termasuk peran, acara, dan artefak) dan Extreme Programming (XP) (termasuk nilai dan praktik utamanya).
- Menganalisis berbagai faktor proyek (seperti kejelasan kebutuhan, kompleksitas, dan risiko) untuk memilih

dan merekomendasikan metodologi pengembangan sistem yang paling tepat untuk sebuah studi kasus.

Selamat datang di Bab 3. Setelah kita memahami "apa" itu sistem (Bab 2), kini saatnya kita mempelajari "bagaimana" cara membangunnya. Membangun sebuah sistem informasi yang kompleks tanpa panduan adalah seperti mencoba membangun sebuah gedung pencakar langit tanpa cetak biru dan rencana konstruksi. Hasilnya kemungkinan besar akan kacau, melebihi anggaran, terlambat dari jadwal, dan yang terburuk, tidak memenuhi kebutuhan penghuninya. Inilah mengapa kita memerlukan sebuah **metodologi**.

Metodologi pengembangan sistem adalah peta jalan kita. Ia menyediakan serangkaian langkah, aturan, dan praktik terbaik yang terstruktur untuk memandu tim dari ide awal hingga sistem yang berfungsi penuh dan dapat dipelihara. Namun, tidak ada satu peta jalan yang cocok untuk semua perjalanan. Proyek yang sederhana dengan tujuan yang sangat jelas mungkin memerlukan pendekatan yang berbeda dari proyek inovatif yang penuh dengan ketidakpastian. Dalam bab ini, kita akan menjelajahi beberapa metodologi paling berpengaruh dalam sejarah pengembangan perangkat lunak. Kita akan mulai dengan pendekatan klasik yang sangat terstruktur, yaitu Model Waterfall. Kemudian, kita akan melihat Model Spiral yang menempatkan manajemen risiko sebagai pusatnya. Setelah itu, kita akan mendalami revolusi pemikiran yang dibawa oleh pendekatan Agile, yang mengutamakan fleksibilitas, kolaborasi, dan kecepatan. Pada akhirnya, Anda akan belajar bahwa memilih metodologi yang tepat bukanlah sekedar

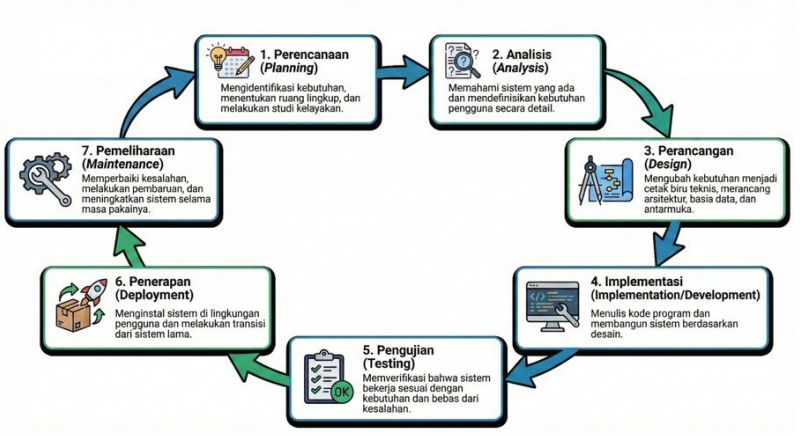
keputusan teknis, melainkan keputusan strategis yang sangat memengaruhi peluang keberhasilan sebuah proyek.

3.1 Pengenalan Metodologi

Dalam dunia rekayasa perangkat lunak dan sistem informasi, proses mengubah ide menjadi sistem yang fungsional adalah sebuah perjalanan yang kompleks dan penuh tantangan. Untuk menavigasi kompleksitas ini, para profesional mengandalkan apa yang disebut **metodologi pengembangan sistem**.

Apa Itu Metodologi?

Secara sederhana, metodologi pengembangan sistem adalah sebuah kerangka kerja (*framework*) yang digunakan untuk menstrukturkan, merencanakan, dan mengendalikan proses pengembangan suatu sistem informasi. Ia merupakan kumpulan metode, prosedur, konsep kerja, dan aturan yang mendefinisikan serangkaian aktivitas yang harus dilakukan oleh tim pengembang dan manajer proyek. Anggaplah metodologi sebagai resep dalam memasak: ia memberitahu Anda bahan apa yang dibutuhkan (sumber daya), langkah-langkah apa yang harus diikuti (fase pengembangan), dalam urutan apa (alur kerja), dan bagaimana cara memastikan hasilnya berkualitas (pengujian dan kontrol).



Gambar 6. Siklus Hidup Pengembangan Sistem

Setiap metodologi pada dasarnya adalah sebuah pendekatan untuk mengelola tahapan-tahapan dalam **Siklus Hidup Pengembangan Sistem (System Development Life Cycle - SDLC)**. Meskipun setiap metodologi memiliki nama dan penekanan yang berbeda, hampir semuanya mencakup serangkaian fase inti yang serupa, yaitu:

1. **Perencanaan (Planning):** Mengidentifikasi kebutuhan, menentukan ruang lingkup, dan melakukan studi kelayakan.
2. **Analisis (Analysis):** Memahami sistem yang ada dan mendefinisikan kebutuhan pengguna secara detail.
3. **Perancangan (Design):** Mengubah kebutuhan menjadi cetak biru teknis, merancang arsitektur, basis data, dan antarmuka.
4. **Implementasi (Implementation/Development):** Menulis kode program dan membangun sistem berdasarkan desain.
5. **Pengujian (Testing):** Memverifikasi bahwa sistem bekerja sesuai dengan kebutuhan dan bebas dari kesalahan.

6. **Penerapan (Deployment):** Menginstal sistem di lingkungan pengguna dan melakukan transisi dari sistem lama.
7. **Pemeliharaan (Maintenance):** Memperbaiki kesalahan, melakukan pembaruan, dan meningkatkan sistem selama masa pakainya.

Perbedaan utama antara satu metodologi dengan metodologi lainnya terletak pada **bagaimana** mereka mengatur dan menjalankan fase-fase ini. Apakah dilakukan secara berurutan dan kaku? Atau secara berulang dalam siklus-siklus pendek?

Mengapa Metodologi Penting?

Seperti yang telah kita bahas di Bab 1, tingkat kegagalan proyek TI sangat tinggi. Penyebab utamanya seringkali bukan masalah teknis, melainkan masalah manajemen dan proses, seperti kebutuhan yang tidak jelas, perubahan yang tidak terkendali, dan kurangnya keterlibatan pengguna. Metodologi pengembangan sistem adalah respons profesional terhadap tantangan-tantangan ini. Dengan menyediakan sebuah proses yang formal dan terstruktur, metodologi membantu untuk:

- **Mengurangi Kompleksitas:** Memecah proyek besar menjadi fase-fase yang lebih kecil dan lebih mudah dikelola.
- **Meningkatkan Prediktabilitas:** Membantu dalam merencanakan, membuat estimasi biaya dan waktu, serta menjadwalkan proyek dengan lebih baik.
- **Memastikan Kualitas:** Menyediakan titik-titik pemeriksaan dan pengujian yang jelas di sepanjang proses untuk memastikan kualitas produk akhir.
- **Meningkatkan Komunikasi:** Memberikan kerangka kerja dan dokumentasi yang sama bagi semua pemangku

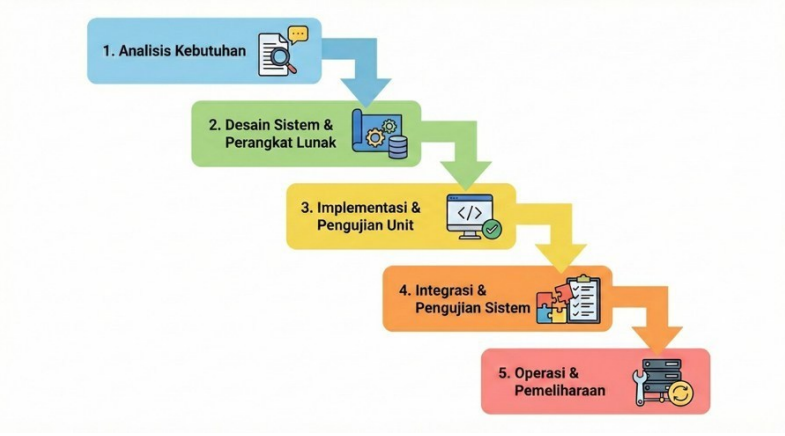
kepentingan (pengguna, manajer, pengembang) untuk berkomunikasi secara efektif.

- **Mengelola Risiko:** Menyediakan cara yang terstruktur untuk mengidentifikasi dan menanggapi masalah atau perubahan yang muncul selama proyek.

Tanpa metodologi, tim pengembang cenderung bekerja secara *ad-hoc*, yang dapat menyebabkan kelalaian, pengerjaan ulang yang mahal, dan pada akhirnya, sistem yang tidak memenuhi tujuan bisnis.

3.2 Waterfall: Pendekatan Klasik yang Terstruktur

Model Waterfall, atau sering disebut model air terjun, adalah salah satu metodologi SDLC paling tua, paling dikenal, dan paling klasik. Diperkenalkan pertama kali pada tahun 1956 oleh Herbert D. Benington, model ini menganalogikan proses pengembangan perangkat lunak seperti aliran air terjun, di mana setiap fase mengalir ke bawah secara berurutan dan setiap fase harus diselesaikan sepenuhnya sebelum fase berikutnya dapat dimulai.



Gambar 7. Ilustrasi Model Waterfall

Tahapan Model Waterfall

Model Waterfall memecah proyek menjadi serangkaian fase yang berbeda dan sekuensial. Meskipun ada sedikit variasi, fase-fase intinya adalah sebagai berikut:

1. **Analisis Kebutuhan (Requirement Analysis):** Ini adalah fase paling krusial. Tim analis sistem bekerja sama dengan klien dan pengguna untuk mengumpulkan semua informasi terkait kebutuhan sistem. Semua kebutuhan fungsional dan non-fungsional didokumentasikan secara lengkap dalam sebuah dokumen spesifikasi kebutuhan (*Software Requirement Specification* / SRS). Fase ini dianggap selesai dan "dibekukan" sebelum melangkah lebih jauh.
2. **Desain Sistem dan Perangkat Lunak (System and Software Design):** Berdasarkan dokumen SRS, tim perancang membuat cetak biru sistem. Fase ini menentukan arsitektur perangkat lunak secara keseluruhan, struktur basis data, desain antarmuka, dan detail teknis lainnya.

Hasil dari fase ini adalah dokumen desain yang akan menjadi panduan bagi para pemrogram.

3. **Implementasi dan Pengujian Unit (Implementation and Unit Testing):** Pada fase ini, desain diterjemahkan ke dalam kode program. Sistem dibangun dalam unit-unit atau modul-modul kecil. Setiap modul yang selesai dibuat kemudian diuji secara individual (pengujian unit) untuk memastikan ia berfungsi sesuai dengan spesifikasinya.
4. **Integrasi dan Pengujian Sistem (Integration and System Testing):** Semua unit atau modul yang telah dikembangkan diintegrasikan menjadi satu sistem yang lengkap. Setelah integrasi, seluruh sistem diuji untuk menemukan kesalahan dalam interaksi antar modul dan untuk memverifikasi bahwa sistem secara keseluruhan memenuhi kebutuhan yang telah didefinisikan di awal.
5. **Operasi dan Pemeliharaan (Operation and Maintenance):** Setelah sistem berhasil diuji dan dinyatakan lulus, sistem tersebut diserahkan kepada klien untuk dioperasikan di lingkungan produksi. Fase pemeliharaan adalah fase terpanjang, di mana tim akan melakukan perbaikan jika ada kesalahan yang ditemukan selama penggunaan, serta melakukan peningkatan atau penambahan fitur jika diperlukan.

Kelebihan dan Kekurangan

Model Waterfall memiliki kelebihan yang membuatnya tetap relevan untuk jenis proyek tertentu, namun juga memiliki kekurangan yang signifikan yang mendorong munculnya metodologi lain.

Tabel 6. Kelebihan dan Kekurangan Model Waterfall

Kelebihan	Kekurangan
Struktur yang Jelas: Alur kerja yang sekuensial dan terdefinisi dengan baik membuat proyek mudah dipahami, direncanakan, dan dikelola.	**Kurang Fleksibel (Kaku):** Sangat sulit untuk kembali ke fase sebelumnya jika ada perubahan kebutuhan. Perubahan di tengah jalan sangat mahal dan mengganggu seluruh jadwal proyek.
Dokumentasi yang Baik: Setiap fase menghasilkan dokumentasi yang lengkap dan detail, yang sangat berguna untuk pemeliharaan dan transfer pengetahuan di masa depan.	**Umpan Balik yang Lambat:** Klien atau pengguna baru dapat melihat dan mencoba sistem yang berfungsi pada akhir siklus pengembangan. Jika ada kesalahpahaman dalam kebutuhan, hal itu baru akan terdeteksi sangat terlambat.
Mudah Diestimasi: Karena ruang lingkup didefinisikan secara penuh di awal, estimasi biaya dan waktu proyek menjadi lebih mudah dan dapat diprediksi.	**Tidak Cocok untuk Proyek Kompleks & Dinamis:** Model ini tidak cocok untuk proyek di mana kebutuhannya tidak dipahami sepenuhnya di awal atau cenderung berubah seiring waktu.
Kualitas Terjaga: Pendekatan yang metodis dan bertahap memastikan setiap fase diperiksa dengan teliti, yang berpotensi menghasilkan kualitas sistem yang lebih baik jika semua berjalan sesuai rencana.	**Risiko "Semua atau Tidak Sama Sekali":** Karena tidak ada produk yang berfungsi hingga akhir proyek, jika proyek dibatalkan di tengah jalan, seringkali tidak ada hasil nyata yang dapat diselamatkan selain dokumentasi.

Model Waterfall paling cocok digunakan untuk proyek-proyek di mana **kebutuhan pengguna sangat jelas, stabil, dan dipahami dengan baik sejak awal**, serta risiko teknologinya rendah.

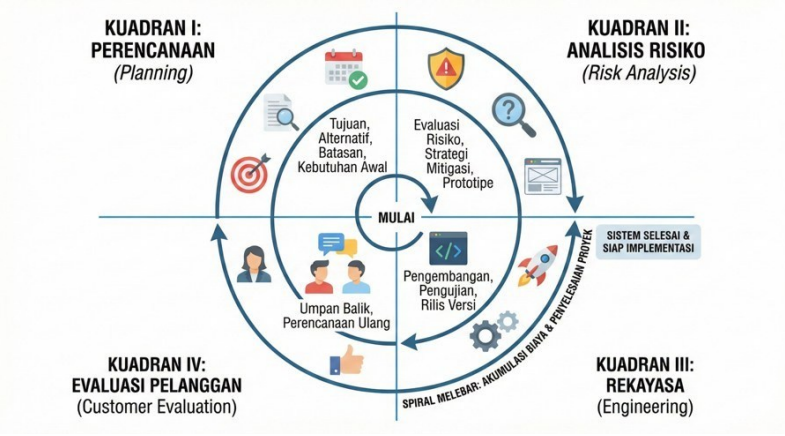
3.3 Spiral: Pendekatan Berbasis Risiko

Menyadari keterbatasan model Waterfall, terutama dalam menangani proyek yang besar dan penuh ketidakpastian, Barry Boehm pada tahun 1986 memperkenalkan **Model Spiral**. Model ini adalah model proses perangkat lunak evolusioner yang menggabungkan sifat iteratif (berulang) dari *prototyping* dengan

aspek kontrol dan sistematis dari model Waterfall. Ciri khas utamanya adalah pendekatan yang digerakkan oleh **risiko (risk-driven)**.

Tahapan dalam Setiap Putaran Spiral

Model Spiral digambarkan sebagai sebuah spiral yang terus berkembang. Setiap putaran atau iterasi dalam spiral merepresentasikan satu fase dalam proses pengembangan. Proyek dimulai dari skala kecil di pusat spiral dan secara bertahap melewati beberapa putaran hingga menghasilkan versi sistem yang lengkap.



Gambar 8. Ilustrasi Model Spiral

Setiap putaran terdiri dari empat aktivitas utama yang dilakukan dalam empat kuadran:

1. **Kuadran I: Perencanaan (Planning):** Pada kuadran ini, tujuan, alternatif, dan batasan untuk iterasi saat ini ditentukan. Ini melibatkan pengumpulan kebutuhan awal dan pemahaman tujuan untuk siklus tersebut.

2. **Kuadran II: Analisis Risiko (Risk Analysis):** Ini adalah jantung dari model Spiral. Untuk setiap alternatif yang diidentifikasi, dilakukan evaluasi dan analisis risiko. Risiko-risiko utama (baik teknis maupun manajerial) diidentifikasi, dan strategi untuk mengatasinya dirumuskan. Aktivitas seperti pembuatan prototipe sering dilakukan di sini untuk memperjelas kebutuhan dan mengurangi risiko ketidakpastian.
3. **Kuadran III: Rekayasa (Engineering):** Setelah risiko dievaluasi, sistem dikembangkan dan diuji. Bergantung pada tingkat risiko, aktivitas di kuadran ini bisa berupa pengembangan model yang lebih detail, penulisan kode, pengujian, hingga perilisan versi operasional dari perangkat lunak.
4. **Kuadran IV: Evaluasi Pelanggan (Customer Evaluation):** Hasil dari fase rekayasa dievaluasi oleh pelanggan atau pengguna. Berdasarkan umpan balik ini, perencanaan untuk putaran spiral berikutnya dimulai.

Proses ini berlanjut dengan spiral yang semakin melebar, menandakan akumulasi biaya dan tingkat penyelesaian proyek, hingga sistem dianggap selesai dan siap untuk diimplementasikan secara penuh.

Kelebihan dan Kekurangan

Model Spiral menawarkan pendekatan yang kuat untuk proyek-proyek tertentu, namun juga membawa kompleksitasnya sendiri.

Tabel 7. Kelebihan dan Kekurangan Model Spiral

Kelebihan	Kekurangan
Fokus pada Manajemen Risiko: Analisis risiko yang eksplisit dan berulang membantu mengurangi kemungkinan kegagalan proyek besar.	**Kompleks dan Mahal:** Model ini lebih rumit untuk dikelola dibandingkan Waterfall. Proses analisis risiko yang konstan membutuhkan keahlian khusus dan dapat membuat proyek menjadi mahal.
Cocok untuk Proyek Besar dan Kompleks: Sangat efektif untuk proyek berskala besar, berisiko tinggi, atau inovatif di mana kebutuhan tidak jelas di awal.	**Memerlukan Keahlian Penilaian Risiko:** Keberhasilan model ini sangat bergantung pada kemampuan tim untuk mengidentifikasi dan mengelola risiko. Jika risiko mayor tidak ditemukan, model ini bisa menjadi masalah serius.
Fleksibilitas dan Adaptasi: Perubahan dapat diakomodasi pada setiap iterasi. Sistem dapat disesuaikan seiring berjalannya waktu.	**Waktu yang Lama:** Proses yang berulang-ulang dan fokus pada analisis dapat membutuhkan waktu yang lama untuk mencapai produk akhir yang pasti.
Penggunaan Prototipe: Prototipe digunakan secara ekstensif untuk mendapatkan umpan balik pengguna lebih awal dan memperjelas kebutuhan, sehingga mengurangi risiko kesalahpahaman.	**Sulit Meyakinkan Pelanggan:** Mungkin sulit untuk meyakinkan pelanggan yang terbiasa dengan model Waterfall bahwa pendekatan evolusioner ini dapat dikontrol dengan baik dari segi biaya dan jadwal.

Model Spiral adalah pilihan yang sangat baik ketika berhadapan dengan proyek yang besar, mahal, dan rumit, di mana manajemen risiko adalah prioritas utama.

3.4 Agile: Revolusi Fleksibilitas dan Kolaborasi

Pada akhir 1990-an dan awal 2000-an, muncul rasa frustrasi yang semakin besar terhadap metodologi tradisional seperti Waterfall. Pendekatan yang kaku dan didorong oleh dokumentasi yang berat dianggap tidak lagi sesuai untuk lingkungan bisnis yang bergerak cepat dan dinamis. Sebagai

respons, pada tahun 2001, sekelompok 17 praktisi pengembangan perangkat lunak berkumpul dan merumuskan apa yang kemudian dikenal sebagai **Manifesto for Agile Software Development**.

Agile bukanlah sebuah metodologi tunggal, melainkan sebuah **filosofi** atau **kerangka berpikir** yang mengutamakan fleksibilitas, kolaborasi, dan pengiriman produk secara inkremental (bertahap) dan iteratif (berulang).

Nilai dan Prinsip Agile

Manifesto Agile didasarkan pada empat nilai inti yang kontras dengan pendekatan tradisional:

1. **Individu dan interaksi** lebih dari proses dan alat.
2. **Perangkat lunak yang berfungsi** lebih dari dokumentasi yang komprehensif.
3. **Kolaborasi dengan pelanggan** lebih dari negosiasi kontrak.
4. **Merespons perubahan** lebih dari mengikuti rencana.

Ini tidak berarti hal-hal di sebelah kanan tidak penting, tetapi Agile lebih menghargai hal-hal di sebelah kiri. Nilai-nilai ini kemudian dijabarkan lebih lanjut ke dalam **12 Prinsip Agile**, yang menekankan kepuasan pelanggan melalui pengiriman perangkat lunak yang cepat dan berkelanjutan, menyambut perubahan, kolaborasi harian, dan refleksi tim secara berkala untuk menjadi lebih efektif.

Kerangka Kerja Agile Populer: Scrum dan XP

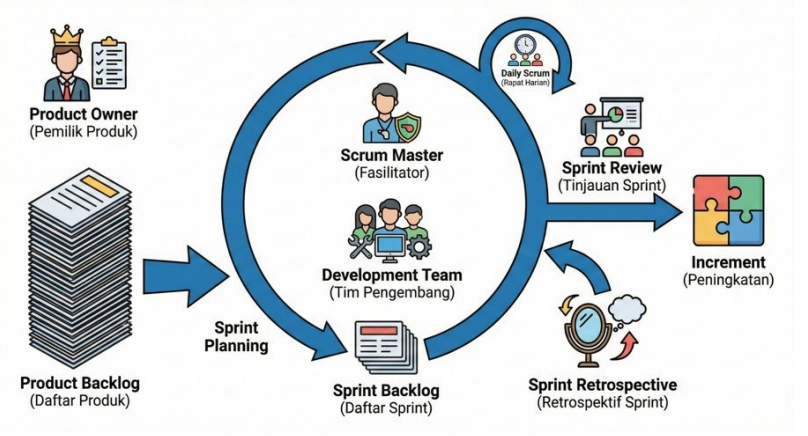
Filosofi Agile diimplementasikan melalui berbagai kerangka kerja (*frameworks*). Dua yang paling populer adalah Scrum dan Extreme Programming (XP).

1. Scrum

Scrum adalah kerangka kerja manajemen proyek yang ringan dan paling banyak digunakan untuk menerapkan Agile. Scrum mengorganisir pekerjaan dalam siklus berulang yang disebut **Sprint**, yang biasanya berdurasi 2-4 minggu. Tujuannya adalah untuk menghasilkan "Increment" atau potongan perangkat lunak yang berpotensi dapat dirilis di akhir setiap Sprint. Kerangka kerja Scrum terdiri dari tiga elemen utama:

- **Peran (Roles):**
    - **Product Owner:** Bertanggung jawab untuk memaksimalkan nilai produk. Ia mengelola daftar kebutuhan produk (*Product Backlog*) dan menjadi suara pelanggan.
    - **Scrum Master:** Bertindak sebagai fasilitator dan pelatih bagi tim, memastikan tim mengikuti praktik Scrum, dan menghilangkan hambatan yang dihadapi tim.
    - **Development Team:** Tim lintas fungsi (analis, desainer, pemrogram, penguji) yang memiliki semua keahlian untuk mengubah item *Product Backlog* menjadi *Increment* yang "Selesai".
- **Acara (Events):**
    - **Sprint Planning:** Di awal setiap Sprint, tim merencanakan pekerjaan yang akan dilakukan.

    - **Daily Scrum:** Rapat harian singkat (15 menit) bagi tim untuk menyinkronkan aktivitas dan merencanakan pekerjaan untuk 24 jam ke depan.
    - **Sprint Review:** Di akhir Sprint, tim mendemonstrasikan *Increment* yang telah dibuat kepada pemangku kepentingan untuk mendapatkan umpan balik.
    - **Sprint Retrospective:** Setelah Sprint Review, tim melakukan refleksi internal untuk mengidentifikasi apa yang berjalan baik dan apa yang bisa diperbaiki untuk Sprint berikutnya.
- **Artefak (Artifacts):**
    - **Product Backlog:** Daftar tunggal dari semua fitur, fungsi, dan perbaikan yang diinginkan untuk produk.
    - **Sprint Backlog:** Kumpulan item dari *Product Backlog* yang dipilih untuk dikerjakan dalam satu Sprint, ditambah rencana untuk menyelesaikannya.
    - **Increment:** Jumlah dari semua item *Product Backlog* yang diselesaikan selama Sprint dan Sprint-sprint sebelumnya.



Gambar 9. Diagram Scrum

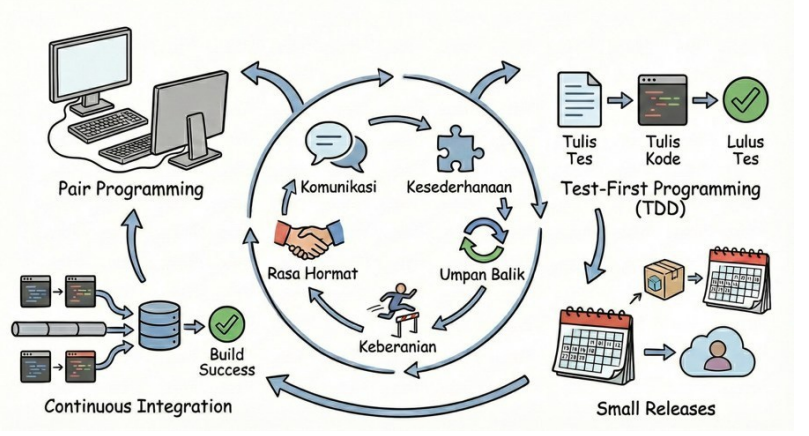
2. Extreme Programming (XP)

XP adalah kerangka kerja Agile lainnya yang sangat menekankan pada keunggulan teknis dan praktik-praktik rekayasa perangkat lunak yang solid. XP berbagi nilai-nilai Agile tetapi memiliki lima nilai spesifiknya sendiri:

- **Komunikasi:** Mendorong interaksi tatap muka yang konstan.
- **Kesederhanaan (Simplicity):** Selalu mencari solusi paling sederhana yang berfungsi.
- **Umpan Balik (Feedback):** Mendapatkan umpan balik secepat dan sesering mungkin dari pelanggan, tim, dan sistem itu sendiri.
- **Keberanian (Courage):** Berani mengatakan kebenaran tentang kemajuan, merevisi pekerjaan, dan mengatasi masalah sulit.
- **Rasa Hormat (Respect):** Anggota tim saling menghormati kontribusi dan keahlian satu sama lain.

Nilai-nilai ini diwujudkan melalui serangkaian praktik teknis yang saling memperkuat, seperti:

- **Pair Programming:** Dua pemrogram bekerja bersama di satu komputer untuk menulis kode.
- **Test-First Programming (TDD):** Menulis tes otomatis sebelum menulis kode fungsional.
- **Continuous Integration:** Mengintegrasikan pekerjaan sesering mungkin (beberapa kali sehari) untuk mendeteksi masalah lebih awal.
- **Small Releases:** Merilis versi perangkat lunak yang berfungsi dalam siklus yang sangat pendek.



Gambar 10. Ilustrasi Extreme Programming

Pendekatan Agile, dengan kerangka kerja seperti Scrum dan XP, sangat cocok untuk proyek di mana **kebutuhan cenderung berubah**, lingkungan bisnis dinamis, dan ada kebutuhan untuk memberikan nilai kepada pelanggan secara cepat dan berkelanjutan.

3.5 Pemilihan Metodologi yang Tepat

Setelah mempelajari berbagai metodologi, pertanyaan terpenting bagi seorang analis sistem adalah: "Metodologi mana yang harus saya pilih untuk proyek ini?" Jawabannya adalah: **tidak ada satu metodologi terbaik untuk semua situasi**. Memilih metodologi yang tepat adalah tentang mencocokkan pendekatan dengan karakteristik unik dari proyek, tim, dan organisasi.

Beberapa faktor kunci yang perlu dipertimbangkan dalam pemilihan metodologi antara lain:

1. **Kejelasan Kebutuhan Pengguna:** Seberapa baik kebutuhan dipahami di awal?
    - **Jelas dan Stabil:** Waterfall adalah pilihan yang kuat karena memungkinkan perencanaan dan desain yang detail di muka.
    - **Tidak Jelas atau Cenderung Berubah:** Agile atau Spiral lebih cocok karena memungkinkan penemuan dan adaptasi kebutuhan seiring berjalannya waktu.
2. **Tingkat Kompleksitas Sistem:** Seberapa rumit sistem yang akan dibangun?
    - **Rendah:** Waterfall bisa mencukupi.
    - **Tinggi:** Spiral (jika risiko tinggi) atau Agile (untuk memecah kompleksitas menjadi bagian-bagian kecil) lebih disarankan.
3. **Penguasaan Teknologi:** Apakah tim akrab dengan teknologi yang akan digunakan?
    - **Akrab/Dikenal Baik:** Waterfall dapat bekerja dengan baik.

- **Baru/Tidak Familiar:** Agile atau Spiral memungkinkan tim untuk belajar dan beradaptasi dengan teknologi baru melalui iterasi dan prototipe.

4. **Risiko Proyek:** Seberapa besar risiko teknis, pasar, atau jadwal yang dihadapi proyek?
    - **Risiko Rendah:** Waterfall bisa menjadi pilihan yang efisien.
    - **Risiko Tinggi:** Spiral dirancang khusus untuk mengelola risiko secara eksplisit dan merupakan pilihan terbaik.
5. **Jadwal dan Waktu:** Seberapa mendesak proyek ini?
    - **Jadwal Panjang dan Fleksibel:** Waterfall atau Spiral bisa dipertimbangkan.
    - **Jadwal Singkat/Butuh Cepat:** Agile dirancang untuk pengiriman cepat dan inkremental.

6. **Ukuran Tim:** Seberapa besar tim pengembang?
    - **Tim Besar:** Waterfall atau Spiral seringkali lebih mudah diskalakan untuk tim yang sangat besar karena strukturnya yang lebih formal.
    - **Tim Kecil hingga Menengah:** Agile (terutama Scrum) bekerja sangat baik dengan tim kecil yang dapat berkolaborasi secara erat.

Tabel 8 merangkum perbandingan ini untuk membantu Anda dalam mengambil keputusan. Sebagai seorang analis sistem, kemampuan Anda untuk mengevaluasi faktor-faktor ini dan merekomendasikan metodologi yang paling sesuai akan menjadi salah satu kontribusi paling strategis yang dapat Anda berikan untuk kesuksesan sebuah proyek.

Tabel 8. Perbandingan Metodologi Pengembangan Sistem

Faktor	Waterfall (Prediktif)	Spiral (Iteratif & Berbasis Risiko)	Agile (Iteratif & Inkremental)
Kebutuhan	Stabil, jelas di awal	Tidak jelas, berevolusi	Dinamis, sering berubah
Fleksibilitas	Rendah, sulit berubah	Sedang, perubahan per iterasi	Tinggi, perubahan disambut
Fokus Utama	Perencanaan & Dokumentasi	Manajemen Risiko	Kolaborasi & Perangkat Lunak Berfungsi
Keterlibatan Klien	Rendah (di awal & akhir)	Sedang (di akhir setiap iterasi)	Tinggi (berkelanjutan setiap hari)
Pengiriman Produk	Sekali di akhir proyek	Prototipe/versi awal per iterasi	Potongan fungsional setiap 2-4 minggu
Dokumentasi	Sangat Komprehensif	Cukup Komprehensif	Minimal, secukupnya
Cocok Untuk	Proyek sederhana, kebutuhan pasti, risiko rendah.	Proyek besar, kompleks, risiko tinggi, inovatif.	Proyek dinamis, kebutuhan tidak pasti, butuh cepat ke pasar.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Jelaskan perbedaan paling mendasar dalam cara metodologi Waterfall dan Agile menangani "perubahan kebutuhan" dari klien di tengah-tengah proyek.
2. Sebuah proyek pengembangan sistem untuk reaktor nuklir memiliki prioritas tertinggi pada keamanan dan keandalan. Kebutuhan sistem sangat kompleks dan ada banyak risiko teknis yang belum pernah dihadapi sebelumnya. Mengapa Model Spiral mungkin menjadi pilihan yang lebih baik daripada Model Waterfall untuk proyek ini?
3. Dalam kerangka kerja Scrum, apa perbedaan peran antara Product Owner dan Scrum Master? Mengapa kedua peran ini penting untuk keberhasilan tim?

Solusi dan Pembahasan Soal Konseptual

1. **Perbedaan Penanganan Perubahan Kebutuhan:**
    - **Waterfall:** Memandang perubahan sebagai sesuatu yang harus dihindari. Karena sifatnya yang sekuensial, setiap fase harus "dibekukan" sebelum melanjutkan ke fase berikutnya. Mengakomodasi perubahan di tengah jalan sangat sulit, mahal, dan mengganggu seluruh rencana proyek yang sudah dibuat di awal. Perubahan biasanya harus melalui proses kontrol perubahan formal yang kaku.

- **Agile:** Memandang perubahan sebagai hal yang tak terhindarkan dan bahkan menyambutnya sebagai peluang untuk memberikan produk yang lebih baik. Melalui siklus pengembangan yang pendek (iterasi/sprint), Agile secara inheren dirancang untuk mengakomodasi perubahan. Kebutuhan dapat ditambahkan, diubah, atau dihapus dari *Product Backlog* di antara setiap siklus, memungkinkan tim untuk beradaptasi dengan cepat terhadap umpan balik pelanggan dan kondisi pasar.

2. Spiral untuk Proyek Berisiko Tinggi (Reaktor Nuklir):

   Model Spiral lebih unggul untuk proyek ini karena fokus utamanya adalah manajemen risiko. Proyek reaktor nuklir memiliki risiko keselamatan dan teknis yang sangat tinggi.

   - **Analisis Risiko Eksplisit:** Setiap putaran dalam spiral dimulai dengan identifikasi dan analisis risiko, memungkinkan tim untuk mengatasi masalah paling kritis terlebih dahulu.
   - **Penggunaan Prototipe:** Tim dapat membangun prototipe untuk mensimulasikan skenario berbahaya atau menguji teknologi baru yang berisiko sebelum mengimplementasikannya secara penuh.
   - **Pendekatan Evolusioner:** Sistem dibangun secara bertahap, memungkinkan validasi dan verifikasi keamanan pada setiap tahap sebelum melanjutkan ke tahap yang lebih kompleks.

Sebaliknya, Model Waterfall akan menuntut semua kebutuhan dan desain didefinisikan secara lengkap di awal, yang hampir tidak mungkin untuk proyek sekompleks dan seberisiko ini. Kesalahan desain yang ditemukan pada tahap pengujian akhir bisa berakibat fatal dan sangat mahal untuk diperbaiki.

3. **Peran Product Owner vs. Scrum Master:**
    - **Product Owner (PO):** Adalah "suara pelanggan" dan bertanggung jawab atas "**APA**" yang akan dibangun. Tugas utamanya adalah mengelola *Product Backlog*, memprioritaskan fitur untuk memaksimalkan nilai bisnis produk, dan memastikan tim pengembang memahami kebutuhan. PO fokus pada produk.
    - **Scrum Master (SM):** Adalah "penjaga proses" dan bertanggung jawab atas "**BAGAIMANA**" tim bekerja. Tugas utamanya adalah memastikan tim memahami dan menerapkan Scrum dengan benar, memfasilitasi acara-acara Scrum, dan menghilangkan segala hambatan (*impediments*) yang menghalangi kemajuan tim. SM fokus pada proses dan tim.
    - **Pentingnya Kedua Peran:** Keduanya sangat penting karena mereka menciptakan keseimbangan. Tanpa PO yang baik, tim mungkin akan membangun produk yang secara teknis bagus tetapi tidak menjawab kebutuhan pasar. Tanpa SM yang baik, tim mungkin akan terhambat oleh masalah, proses yang tidak

efisien, atau gangguan dari luar, sehingga tidak dapat bekerja secara produktif.

Studi Kasus: Pengembangan Aplikasi E-Commerce "CepatLaku"

- **Konteks:** Sebuah perusahaan rintisan (*startup*) "CepatLaku" ingin membangun platform e-commerce baru untuk produk fashion lokal. Pasar e-commerce sangat kompetitif dan tren fashion berubah dengan sangat cepat. Tim inti terdiri dari 10 orang (termasuk pendiri, pemasar, dan pengembang).
- **Visi Produk:** Visi awalnya adalah membuat platform yang memungkinkan pengguna membeli dan menjual produk fashion. Namun, fitur-fitur spesifik seperti "coba virtual" dengan Augmented Reality (AR), sistem rekomendasi berbasis AI, dan integrasi dengan influencer media sosial masih berupa ide yang belum terdefinisi dengan jelas. Manajemen ingin meluncurkan versi dasar dari aplikasi secepat mungkin (dalam 3 bulan) untuk mendapatkan pengguna awal dan umpan balik, kemudian menambahkan fitur-fitur canggih secara bertahap.
- **Tantangan:** Kebutuhan pengguna dan fitur detail tidak diketahui secara pasti. Teknologi untuk fitur AR dan AI adalah hal baru bagi sebagian anggota tim. Anggaran terbatas, sehingga penting untuk fokus pada fitur yang memberikan nilai tertinggi terlebih dahulu.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan:

1. Berdasarkan karakteristik proyek "CepatLaku", metodologi pengembangan mana (Waterfall, Spiral, atau Agile) yang paling sesuai? Berikan justifikasi yang kuat dengan membandingkan setidaknya dua metodologi.
2. Jika Anda memilih Agile dengan kerangka kerja Scrum, jelaskan bagaimana acara **Sprint Review** akan sangat bermanfaat bagi "CepatLaku" dalam menghadapi ketidakpastian pasar.

Analisis:

1. Pemilihan Metodologi yang Tepat:

   Metodologi yang paling sesuai untuk proyek "CepatLaku" adalah Agile.

   Justifikasi (Perbandingan Agile vs. Waterfall):

   - **Kebutuhan Pengguna:** Kebutuhan proyek ini **tidak jelas dan dinamis**. Fitur-fitur canggih seperti AR dan AI masih berupa ide. **Waterfall** akan gagal total di sini karena menuntut semua kebutuhan didefinisikan secara lengkap dan dibekukan di awal. Sebaliknya, **Agile** sangat cocok karena memungkinkan tim untuk memulai dengan visi umum dan menemukan kebutuhan detail melalui iterasi dan umpan balik.
   - **Lingkungan Pasar:** Pasar fashion dan e-commerce **sangat kompetitif dan cepat berubah**. **Waterfall** dengan siklus pengembangannya yang panjang (bisa

berbulan-bulan atau tahunan) akan membuat "CepatLaku" tertinggal dari pesaing. **Agile**, dengan kemampuannya merilis fitur baru setiap beberapa minggu, memungkinkan "CepatLaku" untuk beradaptasi dengan tren pasar dan merespons gerakan pesaing dengan cepat.

- **Tujuan Proyek:** Tujuannya adalah **meluncurkan produk dasar secepat mungkin** untuk validasi pasar. **Waterfall** hanya menghasilkan produk di akhir siklus. **Agile** secara inheren mendukung tujuan ini dengan menghasilkan *Increment* produk yang berfungsi di akhir setiap Sprint, memungkinkan peluncuran *Minimum Viable Product* (MVP) dalam waktu singkat.

Perbandingan Agile vs. Spiral:

Meskipun Spiral juga iteratif dan menangani ketidakpastian, fokus utamanya adalah pada manajemen risiko formal, yang mungkin terlalu berat dan birokratis untuk sebuah startup kecil dengan 10 orang. Agile (Scrum) lebih ringan dan lebih fokus pada kecepatan pengiriman nilai dan kolaborasi tim, yang lebih sesuai dengan budaya startup "CepatLaku".

2. Manfaat Sprint Review untuk "CepatLaku":

   Acara Sprint Review akan menjadi mekanisme vital bagi "CepatLaku" untuk menavigasi ketidakpastian. Sprint Review adalah sesi di akhir setiap Sprint di mana Tim

Scrum mendemonstrasikan produk yang telah mereka bangun kepada para pemangku kepentingan (pendiri, investor, calon pengguna). Manfaatnya adalah:

- **Umpan Balik Cepat dan Nyata:** Alih-alih membahas dokumen spesifikasi yang abstrak, pemangku kepentingan dapat melihat, menyentuh, dan mencoba langsung potongan perangkat lunak yang berfungsi. Mereka bisa memberikan umpan balik konkret, misalnya, "Tombol 'Beli' ini kurang terlihat," atau "Alur checkout terlalu rumit."
- **Validasi Ide dan Arah Produk:** Tim dapat memvalidasi asumsi mereka. Misalnya, setelah satu Sprint, mereka bisa menunjukkan prototipe fitur "coba virtual" AR. Jika umpan balik dari pengguna awal sangat positif, Product Owner dapat memprioritaskan pengembangan fitur ini lebih lanjut. Jika umpan baliknya negatif, mereka dapat dengan cepat mengubah arah tanpa membuang banyak waktu dan sumber daya.
- **Adaptasi terhadap Pasar:** Selama Sprint Review, tim pemasaran bisa berbagi wawasan terbaru tentang tren fashion atau strategi pesaing. Diskusi ini dapat secara langsung memengaruhi prioritas fitur untuk Sprint berikutnya, memastikan produk yang dikembangkan tetap relevan dan kompetitif.

Singkatnya, Sprint Review mengubah pengembangan dari proses linier yang buta menjadi siklus belajar yang

cepat, memungkinkan "CepatLaku" untuk secara bertahap "menemukan" produk yang tepat bersama dengan penggunanya, bukan hanya "membangun" produk berdasarkan asumsi awal.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Laudon, K. C., & Laudon, J. P. (Edisi terbaru). *Management Information Systems: Managing the Digital Firm*. Pearson.

Bacaan Tambahan (Dianjurkan)

- **The Agile Manifesto.** (2001). Tersedia di: https://agilemanifesto.org/ dan versi terjemahan Bahasa Indonesia di https://agilemanifesto.org/iso/id/principles.html. Ini adalah dokumen sumber yang wajib dibaca untuk memahami pondasi filosofis dari pendekatan Agile.
- Schwaber, K., & Sutherland, J. (Edisi terbaru). *The Scrum Guide*. Tersedia di: https://scrumguides.org/. Dokumen definitif yang menjelaskan aturan main Scrum, ditulis oleh para penciptanya.
- Boehm, B. W. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes, 11*(4). Artikel seminal yang memperkenalkan Model Spiral kepada dunia.

Bab 4: Pengumpulan Kebutuhan Sistem

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan pentingnya tahap pengumpulan kebutuhan sebagai fase paling kritis dalam siklus hidup pengembangan sistem.
- Mengidentifikasi dan menjelaskan lima teknik utama pengumpulan kebutuhan: Wawancara, Observasi, Kuisioner, Workshop, dan Analisis Dokumen.
- Membandingkan kelebihan dan kekurangan dari setiap teknik pengumpulan kebutuhan untuk menentukan kesesuaiannya dengan berbagai skenario proyek.
- Merancang dan menyusun pertanyaan wawancara yang efektif, mencakup jenis pertanyaan terbuka, tertutup, dan penyelidikan (*probing*).
- Merancang instrumen kuisioner yang valid dan andal untuk mengumpulkan data dari populasi pengguna yang besar.
- Menerapkan prinsip-prinsip dasar dalam melakukan observasi dan analisis dokumen untuk menggali kebutuhan yang tidak terucapkan atau tersirat.
- Memahami konsep dan manfaat dari sesi workshop kolaboratif seperti *Joint Application Development* (JAD)

dalam mempercepat proses pengumpulan kebutuhan dan membangun konsensus.

Selamat datang di Bab 4. Jika metodologi pengembangan sistem yang kita bahas di Bab 3 adalah peta jalan, maka tahap pengumpulan kebutuhan adalah proses menentukan tujuan akhir dari perjalanan kita. Fase ini, yang juga dikenal sebagai *requirements elicitation* atau *requirements gathering*, adalah pondasi di atas mana seluruh proyek sistem informasi akan dibangun. Seperti yang telah kita singgung di bab-bab sebelumnya, statistik kegagalan proyek secara konsisten menunjuk pada satu biang keladi utama: "kebutuhan yang tidak lengkap atau tidak jelas". Kesalahan yang dibuat pada tahap ini akan merambat dan membesar di fase-fase selanjutnya, dan menjadi semakin mahal untuk diperbaiki. Membangun sistem tanpa pemahaman yang mendalam tentang kebutuhan pengguna adalah seperti seorang arsitek yang merancang rumah tanpa pernah bertanya kepada calon penghuninya tentang berapa kamar yang mereka butuhkan atau gaya hidup seperti apa yang mereka jalani.

Dalam bab ini, kita akan membekali diri dengan seperangkat "alat investigasi" yang digunakan oleh analis sistem profesional untuk menggali, memahami, dan memvalidasi kebutuhan dari berbagai pemangku kepentingan. Kita akan belajar seni bertanya melalui **wawancara**, kekuatan mengamati melalui **observasi**, jangkauan luas dari **kuisioner**, efisiensi kolaborasi melalui **workshop**, dan wawasan tersembunyi dari **analisis dokumen**. Setiap teknik memiliki kekuatan dan kelemahannya sendiri. Analis sistem yang andal tidak hanya menguasai cara menggunakan setiap alat, tetapi juga memiliki kearifan untuk

mengetahui alat mana yang paling tepat untuk situasi yang dihadapi. Mari kita mulai proses menjadi detektif sistem yang ulung.

4.1 Teknik Wawancara

Wawancara adalah salah satu teknik pengumpulan kebutuhan yang paling umum dan seringkali paling efektif. Pada intinya, wawancara adalah percakapan terarah antara analis sistem dengan satu atau lebih pemangku kepentingan (seperti pengguna, manajer, atau staf operasional) dengan tujuan untuk mengumpulkan informasi tentang sistem yang ada, proses bisnis, masalah yang dihadapi, dan kebutuhan untuk sistem yang baru.

Kunci dari wawancara yang sukses adalah persiapan. Seorang analis tidak bisa hanya datang dan bertanya, "Apa yang Anda inginkan?" Analis harus merencanakan wawancara, menyusun pertanyaan yang tepat, mendengarkan secara aktif, dan mendokumentasikan hasilnya secara sistematis.

Jenis Pertanyaan dalam Wawancara

Efektivitas wawancara sangat bergantung pada jenis pertanyaan yang diajukan. Ada tiga jenis utama pertanyaan yang harus dikuasai oleh seorang analis:

1. **Pertanyaan Terbuka (Open-ended Questions):** Pertanyaan ini dirancang untuk mendorong responden memberikan jawaban yang panjang, deskriptif, dan kaya akan detail. Pertanyaan ini tidak bisa dijawab hanya dengan "ya" atau "tidak". Mereka sangat berguna di awal wawancara untuk mendapatkan gambaran umum dan memahami konteks.

- Contoh: *"Bisakah Anda jelaskan langkah-langkah yang Anda lakukan saat menerima pesanan dari pelanggan?"* atau *"Apa saja tantangan terbesar yang Anda hadapi dengan sistem pelaporan saat ini?"*

2. **Pertanyaan Tertutup (Closed-ended Questions):** Pertanyaan ini dirancang untuk mendapatkan jawaban yang spesifik, singkat, dan faktual. Biasanya digunakan untuk mengonfirmasi pemahaman, mendapatkan data kuantitatif, atau memilih dari serangkaian opsi yang terbatas.
    - Contoh: *"Berapa rata-rata jumlah pesanan yang Anda proses setiap hari?"* atau *"Apakah laporan ini harus dicetak setiap pagi?"*
3. **Pertanyaan Penyelidikan (Probing Questions):** Pertanyaan ini adalah pertanyaan lanjutan yang digunakan untuk menggali lebih dalam jawaban yang diberikan oleh responden. Tujuannya adalah untuk mendapatkan klarifikasi, detail tambahan, atau contoh spesifik.
    - Contoh: (Setelah responden berkata, "Prosesnya terkadang lambat.") Analis bertanya, *"Bisakah Anda memberikan contoh spesifik kapan proses tersebut terasa lambat?"* atau *"Apa yang Anda maksud dengan 'lambat'?"*

Proses Wawancara

Wawancara yang efektif mengikuti struktur yang jelas:

1. **Perencanaan:** Tentukan tujuan wawancara, pilih siapa yang akan diwawancarai, buat janji temu, dan yang terpenting, siapkan daftar pertanyaan (kombinasi dari ketiga jenis di atas).

2. **Pelaksanaan:** Mulailah dengan perkenalan dan jelaskan tujuan wawancara. Ajukan pertanyaan dari daftar Anda, tetapi tetap fleksibel untuk mengeksplorasi topik-topik baru yang muncul. Dengarkan secara aktif, buat catatan, dan perhatikan isyarat non-verbal.
3. **Tindak Lanjut:** Segera setelah wawancara, rapikan catatan Anda dan buat ringkasan. Kirimkan ringkasan tersebut kepada orang yang diwawancarai untuk meminta konfirmasi dan koreksi. Ini memastikan pemahaman yang akurat dan menunjukkan bahwa Anda menghargai waktu mereka.

Tabel 9. Kelebihan dan Kekurangan Teknik Wawancara

Kelebihan	Kekurangan
Kaya Informasi: Memungkinkan analis untuk mendapatkan informasi yang mendalam, kualitatif, dan penuh konteks.	**Memakan Waktu:** Membutuhkan waktu yang signifikan untuk perencanaan, pelaksanaan, dan dokumentasi, terutama jika banyak orang yang harus diwawancarai.
Fleksibel: Analis dapat menyesuaikan pertanyaan secara langsung untuk menggali informasi yang tidak terduga.	**Potensi Bias:** Jawaban responden dapat dipengaruhi oleh pendapat pribadi atau politik kantor. Pewawancara juga bisa secara tidak sadar mengarahkan jawaban.
Membangun Hubungan: Interaksi tatap muka membantu membangun kepercayaan dan hubungan baik dengan pengguna, yang penting untuk keberhasilan proyek.	**Ketergantungan pada Keahlian Pewawancara:** Kualitas informasi yang didapat sangat bergantung pada kemampuan analis dalam bertanya dan mendengarkan.
Mengamati Isyarat Non-Verbal: Analis dapat menangkap keraguan atau antusiasme melalui bahasa tubuh, yang memberikan konteks tambahan.	**Sulit untuk Menganalisis:** Informasi kualitatif dari banyak wawancara bisa sulit untuk dibandingkan dan diringkas secara sistematis.

4.2 Observasi

Observasi adalah teknik mengamati dan mendokumentasikan bagaimana pengguna berinteraksi dengan sistem dan menjalankan proses bisnis dalam lingkungan kerja mereka yang sebenarnya. Teknik ini sangat kuat karena seringkali apa yang orang *katakan* mereka lakukan (dalam wawancara) berbeda dengan apa yang mereka *sebenarnya* lakukan. Observasi membantu menjembatani kesenjangan ini.

Seorang analis dapat berperan sebagai **pengamat pasif**, di mana ia hanya melihat dan mencatat tanpa mengganggu alur kerja. Atau, ia bisa menjadi **pengamat aktif**, di mana ia sesekali bertanya kepada pengguna untuk mengklarifikasi tindakan yang sedang mereka lakukan.

Gambar 11. Proses Observasi - Aktif vs Pasif

Kapan Menggunakan Observasi?

Observasi sangat berguna untuk:

- **Memahami Proses yang Kompleks:** Ketika sebuah alur kerja memiliki banyak langkah atau variasi yang sulit dijelaskan dengan kata-kata.
- **Memvalidasi Informasi:** Untuk memeriksa apakah informasi yang didapat dari wawancara atau dokumen sesuai dengan praktik di lapangan.
- **Mengidentifikasi Kebutuhan Tersirat:** Pengguna mungkin telah terbiasa dengan masalah atau cara kerja yang tidak efisien sehingga mereka tidak lagi menyadarinya. Melalui observasi, analis dapat mengidentifikasi "rasa sakit" (*pain points*) yang tidak diungkapkan oleh pengguna.

Tabel 10. Kelebihan dan Kekurangan Teknik Observasi

Kelebihan	Kekurangan
Data yang Sangat Akurat: Memberikan gambaran langsung tentang bagaimana sistem dan proses benar-benar bekerja, bukan bagaimana seharusnya bekerja.	**Efek Hawthorne:** Orang yang merasa diamati mungkin mengubah perilaku mereka, sehingga apa yang diobservasi tidak mencerminkan kenyataan sehari-hari.
Mengungkap Kebutuhan Tersirat: Dapat mengidentifikasi masalah, *workaround*, dan inefisiensi yang tidak disadari atau tidak dilaporkan oleh pengguna.	**Memakan Waktu:** Membutuhkan waktu yang lama untuk mengamati proses secara lengkap, terutama jika proses tersebut jarang terjadi.
Memvalidasi Teknik Lain: Hasil observasi dapat digunakan untuk mengonfirmasi atau menantang temuan dari wawancara dan analisis dokumen.	**Interpretasi Subjektif:** Apa yang dilihat oleh seorang analis dapat diinterpretasikan secara berbeda oleh analis lain. Ada potensi bias dari pengamat.
Baik untuk Proses Fisik: Sangat efektif untuk memahami alur kerja yang melibatkan banyak aktivitas fisik, seperti di lantai pabrik atau gudang.	**Tidak Mengungkap "Mengapa":** Observasi menunjukkan *apa* yang dilakukan seseorang, tetapi tidak selalu menjelaskan *mengapa* mereka

	melakukannya. Ini perlu digali lebih lanjut dengan wawancara.

4.3 Kuisioner

Kuisioner (atau survei) adalah serangkaian pertanyaan tertulis yang didistribusikan kepada sejumlah besar responden untuk mengumpulkan data tentang sikap, opini, atau fakta. Teknik ini sangat efisien ketika informasi perlu dikumpulkan dari kelompok orang yang besar dan tersebar secara geografis.

Merancang Kuisioner yang Baik

Kualitas data dari kuisioner sangat bergantung pada kualitas desainnya. Beberapa panduan penting:

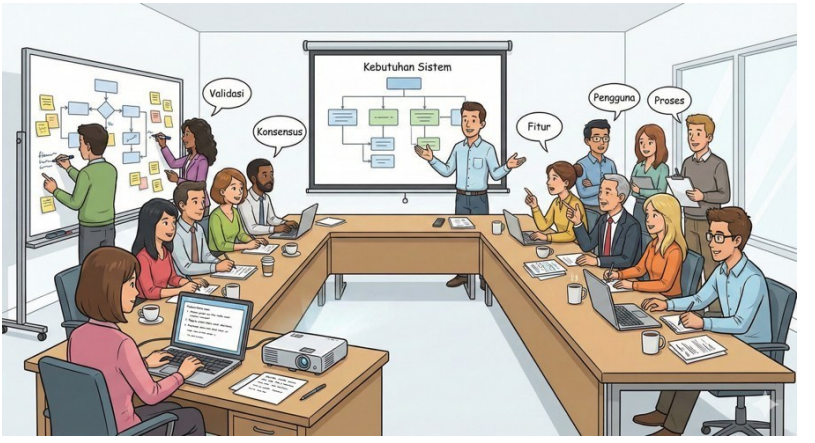
- **Tujuan yang Jelas:** Setiap pertanyaan harus terkait langsung dengan tujuan pengumpulan data.
- **Pertanyaan yang Tidak Ambigu:** Gunakan bahasa yang sederhana dan jelas. Hindari jargon teknis dan pertanyaan dengan makna ganda.
- **Jenis Pertanyaan yang Tepat:** Gunakan kombinasi jenis pertanyaan seperti pilihan ganda, skala peringkat (misalnya, skala Likert dari 1-5), dan pertanyaan terbuka (digunakan secara hemat).
- **Uji Coba:** Sebelum menyebarkan kuisioner secara luas, ujilah terlebih dahulu pada sekelompok kecil responden untuk mengidentifikasi pertanyaan yang membingungkan atau masalah lainnya.

Tabel 11. Kelebihan dan Kekurangan Teknik Kuisioner

Kelebihan	Kekurangan
Jangkauan Luas: Dapat menjangkau sejumlah besar orang dalam waktu singkat dengan biaya yang relatif rendah.	**Tingkat Respons Rendah:** Seringkali, hanya sebagian kecil dari orang yang menerima kuisioner yang akan mengisinya.
Data Kuantitatif: Menghasilkan data yang mudah dianalisis secara statistik, memungkinkan identifikasi tren dan pola.	**Tidak Fleksibel:** Tidak ada kesempatan untuk mengajukan pertanyaan lanjutan atau mengklarifikasi jawaban yang tidak jelas.
Anonimitas: Responden mungkin lebih jujur dalam memberikan jawaban, terutama untuk topik yang sensitif, karena identitas mereka bisa dirahasiakan.	**Potensi Kesalahpahaman:** Responden dapat salah menafsirkan pertanyaan, yang mengarah pada data yang tidak akurat.
Mengurangi Bias Pewawancara: Karena tidak ada interaksi langsung, bias dari analis sistem dapat diminimalkan.	**Tidak Mendalam:** Sulit untuk mendapatkan pemahaman yang kaya dan kontekstual tentang suatu masalah hanya dari jawaban tertulis.

4.4 Workshop

Workshop, seringkali dalam bentuk sesi **Joint Application Development (JAD)**, adalah teknik yang sangat kolaboratif di mana sekelompok pemangku kepentingan utama (pengguna, manajer, analis, dan terkadang pengembang) dikumpulkan dalam satu ruangan untuk sesi terstruktur yang intensif. Tujuannya adalah untuk mendefinisikan, memvalidasi, dan mencapai konsensus tentang kebutuhan sistem dalam waktu yang singkat.



Gambar 12. Sesi Workshop

Sesi ini dipimpin oleh seorang **fasilitator** yang netral, yang tugasnya adalah memandu diskusi, menjaga agar sesi tetap fokus, dan memastikan semua orang memiliki kesempatan untuk berbicara. Ada juga seorang **juru tulis (scribe)** yang bertanggung jawab untuk mendokumentasikan semua keputusan dan poin diskusi secara *real-time*, seringkali menggunakan proyektor agar semua peserta dapat melihatnya.

Manfaat Workshop

Workshop sangat efektif untuk:

- **Mempercepat Proses:** Menggantikan puluhan wawancara individual dengan beberapa sesi kelompok yang terfokus.
- **Membangun Konsensus:** Ketika ada perbedaan pendapat di antara pemangku kepentingan, workshop menyediakan forum untuk mendiskusikan dan menyelesaikan perbedaan tersebut secara langsung.
- **Meningkatkan Kualitas Kebutuhan:** Interaksi langsung antara pengguna dan tim TI membantu mengurangi

kesalahpahaman dan menghasilkan kebutuhan yang lebih jelas dan lengkap.

Tabel 12. Kelebihan dan Kekurangan Teknik Workshop

Kelebihan	Kekurangan
Efisiensi Waktu: Secara dramatis mengurangi waktu yang dibutuhkan untuk mengumpulkan dan memvalidasi kebutuhan dibandingkan dengan wawancara individual.	**Sulit Dijadwalkan:** Mengumpulkan semua pemangku kepentingan utama di satu tempat pada waktu yang sama bisa menjadi tantangan logistik yang besar.
Membangun Konsensus: Mendorong resolusi konflik dan menghasilkan rasa kepemilikan bersama (*shared ownership*) terhadap kebutuhan sistem.	**Dinamika Kelompok:** Sesi dapat didominasi oleh individu yang vokal, sementara peserta yang lebih pendiam mungkin ragu untuk menyuarakan pendapat mereka.
Umpan Balik Langsung: Ide dan model dapat disajikan dan langsung divalidasi atau dikoreksi oleh seluruh kelompok.	**Membutuhkan Fasilitator Ahli:** Keberhasilan sesi sangat bergantung pada keterampilan fasilitator dalam mengelola diskusi dan konflik.
Kualitas Kebutuhan Tinggi: Sinergi dari berbagai perspektif dalam satu ruangan seringkali menghasilkan pemahaman kebutuhan yang lebih kaya dan akurat.	**Biaya Tinggi:** Membutuhkan biaya untuk fasilitas, dan biaya waktu dari banyak orang penting yang meninggalkan pekerjaan mereka selama sesi berlangsung.

4.5 Analisis Dokumen

Teknik ini melibatkan pengumpulan dan peninjauan sistematis terhadap dokumen, formulir, laporan, dan file yang ada dalam organisasi. Tujuannya adalah untuk memahami sistem yang ada (*as-is system*), aturan bisnis yang berlaku, dan jenis data yang penting bagi organisasi.

Seorang analis dapat mempelajari berbagai jenis dokumen, seperti:

- **Manual Prosedur dan SOP:** Untuk memahami alur kerja formal.
- **Formulir Bisnis:** Untuk melihat data apa yang dikumpulkan (misalnya, formulir pemesanan, faktur).
- **Laporan Manajemen:** Untuk memahami informasi apa yang dianggap penting untuk pengambilan keputusan.
- **Dokumentasi Sistem Lama:** Untuk memahami logika dan struktur sistem yang ada.
- **Bagan Organisasi:** Untuk memahami struktur pelaporan dan tanggung jawab.

Tabel 13. Kelebihan dan Kekurangan Teknik Analisis Dokumen

Kelebihan	Kekurangan
Memberikan Konteks: Membantu analis memahami kosakata, aturan, dan proses bisnis organisasi sebelum berbicara dengan orang-orang.	**Dokumen Mungkin Usang:** Prosedur dan formulir yang terdokumentasi mungkin tidak lagi mencerminkan cara kerja yang sebenarnya.
Sumber Aturan Bisnis: Dokumen seringkali merupakan sumber yang baik untuk mengidentifikasi aturan bisnis formal (misalnya, "Diskon 10% berlaku untuk pelanggan A").	**Tidak Lengkap:** Dokumentasi seringkali tidak lengkap atau hanya menjelaskan "kasus ideal" dan mengabaikan pengecualian atau masalah.
Biaya Rendah: Merupakan cara yang tidak mengganggu dan relatif murah untuk mendapatkan pemahaman awal tentang sistem.	**Bisa Membingungkan:** Dokumen dari sistem yang berbeda mungkin berisi informasi yang bertentangan atau menggunakan terminologi yang tidak konsisten.
Dasar untuk Pertanyaan: Analisis dokumen dapat membantu analis menyiapkan pertanyaan yang lebih relevan dan cerdas untuk wawancara.	**Tidak Menjelaskan Implementasi Aktual:** Dokumen menjelaskan bagaimana proses *seharusnya* bekerja, bukan bagaimana proses itu *sebenarnya* diimplementasikan oleh staf.

Tidak ada satu pun teknik pengumpulan kebutuhan yang sempurna. Analis sistem yang efektif menggunakan kombinasi dari beberapa teknik ini. Misalnya, memulai dengan analisis dokumen untuk mendapatkan konteks, dilanjutkan dengan wawancara untuk pemahaman mendalam, kemudian melakukan observasi untuk memvalidasi temuan, dan diakhiri dengan workshop untuk menyelesaikan detail dan membangun konsensus.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Anda ditugaskan untuk mengembangkan sistem informasi baru untuk 300 agen penjualan yang tersebar di seluruh Indonesia. Tujuan utama sistem adalah untuk melacak prospek penjualan dan melaporkan aktivitas harian. Mengingat jumlah dan lokasi pengguna, teknik pengumpulan kebutuhan mana yang akan menjadi pilihan utama Anda, dan mengapa? Teknik apa yang akan Anda gunakan sebagai pelengkap?
2. Jelaskan "Efek Hawthorne" dalam konteks teknik observasi. Bagaimana seorang analis sistem dapat mencoba meminimalkan dampak dari efek ini?
3. Mengapa sesi workshop seperti JAD seringkali lebih efektif dalam menyelesaikan perbedaan pendapat antar departemen dibandingkan dengan serangkaian wawancara individual?

Solusi dan Pembahasan Soal Konseptual

1. **Sistem untuk 300 Agen Penjualan:**
    - **Pilihan Utama: Kuisioner** akan menjadi pilihan utama. Alasannya adalah karena audiensnya besar (300 orang) dan tersebar secara geografis di seluruh Indonesia. Kuisioner memungkinkan pengumpulan data kuantitatif (misalnya, "Berapa banyak waktu yang Anda habiskan untuk pelaporan setiap hari?") dan data

kualitatif terstruktur dari semua agen secara efisien dan dengan biaya rendah.

- **Teknik Pelengkap:** Sebagai pelengkap, **wawancara mendalam** akan dilakukan dengan sampel kecil agen penjualan (misalnya, 5-10 orang) yang mewakili berbagai tingkat kinerja (berkinerja tinggi, rata-rata, rendah) dan wilayah. Wawancara ini akan memberikan konteks dan pemahaman yang lebih kaya di balik data kuantitatif yang diperoleh dari kuisioner. Selain itu, **analisis dokumen** terhadap formulir laporan aktivitas yang ada saat ini juga penting untuk memahami data apa yang dikumpulkan sekarang.

2. **Efek Hawthorne dan Mitigasinya:**
   - **Definisi:** Efek Hawthorne adalah fenomena di mana individu mengubah atau meningkatkan aspek perilaku mereka sebagai respons karena kesadaran bahwa mereka sedang diamati, bukan karena perubahan pada kondisi eksperimental. Dalam konteks observasi, seorang staf mungkin bekerja lebih cepat, lebih teliti, atau mengikuti prosedur "sesuai buku" karena analis sistem sedang mengawasinya, yang mungkin tidak mencerminkan cara kerja normalnya.
   - **Mitigasi:** Untuk meminimalkannya, analis dapat: (1) **Melakukan observasi dalam periode yang lebih lama**, sehingga subjek terbiasa dengan kehadiran analis dan kembali ke perilaku normal mereka. (2) **Melakukan**

observasi secara tidak mencolok jika memungkinkan dan etis. (3) **Menginformasikan kepada subjek bahwa tujuannya adalah untuk memahami proses, bukan untuk mengevaluasi kinerja individu**, untuk mengurangi kecemasan. (4) **Menggabungkan observasi dengan teknik lain** (seperti wawancara dan analisis dokumen) untuk memvalidasi temuan.

3. Efektivitas Workshop dalam Menyelesaikan Konflik:

   Workshop seperti JAD lebih efektif karena beberapa alasan:

   - **Komunikasi Langsung:** Semua pihak yang berselisih berada di ruangan yang sama. Mereka dapat mendengar perspektif satu sama lain secara langsung, bukan melalui interpretasi analis dari wawancara terpisah. Ini mengurangi potensi kesalahpahaman.
   - **Transparansi:** Semua diskusi dan keputusan didokumentasikan secara *real-time* dan terlihat oleh semua orang. Ini menciptakan pemahaman bersama dan mencegah "pertemuan setelah pertemuan" di mana keputusan dipertanyakan kembali.
   - **Fasilitasi Netral:** Seorang fasilitator yang terlatih memandu diskusi, memastikan bahwa percakapan tetap konstruktif, semua pihak didengar, dan tujuannya adalah untuk menemukan solusi terbaik bagi organisasi,

bukan untuk "memenangkan" argumen departemen.

- **Momentum dan Tekanan Positif:** Adanya batas waktu dan kehadiran para pembuat keputusan dalam sesi menciptakan tekanan positif untuk mencapai kompromi dan kesepakatan, daripada membiarkan masalah berlarut-larut.

Studi Kasus: Sistem Pendaftaran Ulang Mahasiswa Universitas Cendekia

- **Konteks:** Universitas Cendekia, sebuah universitas swasta dengan 15.000 mahasiswa, menghadapi masalah setiap awal semester. Proses pendaftaran ulang (herregistrasi) dan pemilihan mata kuliah (KRS) masih semi-manual dan menyebabkan antrean panjang, kebingungan mahasiswa, dan kesalahan data.
- **Proses Saat Ini:** Mahasiswa harus datang ke bagian administrasi akademik (BAA) untuk mengambil formulir, kemudian menemui dosen wali untuk persetujuan KRS, lalu ke bagian keuangan untuk membayar, dan kembali lagi ke BAA untuk menyerahkan berkas. Proses ini memakan waktu berhari-hari.
- **Tujuan Proyek:** Manajemen universitas ingin membangun sebuah sistem informasi online terintegrasi yang memungkinkan mahasiswa melakukan seluruh proses pendaftaran ulang dan pengisian KRS dari mana saja.

- **Pemangku Kepentingan:** Mahasiswa, Dosen Wali, Staf BAA, Staf Keuangan, dan Pimpinan Fakultas/Universitas.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan: Anda adalah analis sistem utama untuk proyek ini. Rancanglah sebuah rencana pengumpulan kebutuhan yang komprehensif. Jelaskan teknik apa saja yang akan Anda gunakan, siapa target dari setiap teknik tersebut, dan informasi spesifik apa yang ingin Anda dapatkan dari masing-masing teknik.

Analisis Rencana Pengumpulan Kebutuhan:

Sebagai analis sistem, saya akan menggunakan pendekatan multi-teknik untuk memastikan semua perspektif tercakup dan kebutuhan dipahami secara mendalam.

1. **Teknik: Analisis Dokumen**
    - **Target:** Dokumen-dokumen yang ada di BAA dan Bagian Keuangan.
    - **Dokumen yang Dianalisis:** Formulir pendaftaran ulang, formulir KRS, buku panduan akademik (terkait aturan SKS dan mata kuliah prasyarat), laporan pembayaran mahasiswa, dan SOP yang ada (jika ada).
    - **Informasi yang Dicari:**
        - Data apa saja yang saat ini dikumpulkan dari mahasiswa?
        - Apa saja aturan bisnis formal terkait pengambilan SKS (misalnya, SKS

maksimum berdasarkan IPK, mata kuliah prasyarat)?

- Bagaimana alur persetujuan dan pembayaran yang seharusnya berjalan?

2. **Teknik: Observasi**
    - **Target:** Staf BAA dan Staf Keuangan selama periode puncak pendaftaran ulang.
    - **Informasi yang Dicari:**
        - Di mana titik-titik kemacetan (*bottlenecks*) dalam proses saat ini? (misalnya, antrean terpanjang, proses yang paling memakan waktu).
        - Masalah apa yang paling sering ditanyakan atau dikeluhkan oleh mahasiswa saat berinteraksi dengan staf?
        - Adakah *workaround* atau langkah tidak resmi yang dilakukan staf untuk mempercepat proses?
        - Bagaimana interaksi antara staf BAA dan staf Keuangan?
3. **Teknik: Kuisioner Online**
    - **Target:** Populasi mahasiswa secara luas (dikirim melalui email atau portal mahasiswa yang ada).
    - **Informasi yang Dicari:**
        - Mengukur tingkat kepuasan mahasiswa terhadap proses saat ini (menggunakan skala peringkat).

- Bagian mana dari proses yang dianggap paling menyulitkan oleh mayoritas mahasiswa?
- Fitur apa yang paling diinginkan dalam sistem online yang baru (misalnya, notifikasi jadwal, rekomendasi mata kuliah)?
- Perangkat apa yang paling sering digunakan mahasiswa untuk mengakses internet (penting untuk desain antarmuka: desktop vs. mobile)?

4. **Teknik: Wawancara**
    - **Target:** Sampel representatif dari setiap kelompok pemangku kepentingan.
        - **Mahasiswa:** Beberapa mahasiswa dari berbagai fakultas dan angkatan.
        - **Dosen Wali:** Beberapa dosen dari fakultas yang berbeda.
        - **Staf BAA & Keuangan:** Staf kunci yang terlibat langsung dalam proses.
        - **Pimpinan:** Kepala BAA, Kepala Keuangan, dan Dekan.
    - **Informasi yang Dicari:**
        - Dari **Mahasiswa & Dosen Wali:** Pengalaman, frustrasi, dan harapan mereka terhadap sistem baru.
        - Dari **Staf BAA & Keuangan:** Detail operasional, pengecualian-pengecualian yang sering terjadi, dan kebutuhan pelaporan mereka.

        - Dari **Pimpinan:** Tujuan strategis dari sistem baru (misalnya, meningkatkan citra universitas, efisiensi data untuk akreditasi), batasan anggaran, dan metrik keberhasilan proyek.

5. **Teknik: Workshop (Sesi JAD)**
    - **Target:** Mengundang perwakilan kunci dari semua kelompok pemangku kepentingan yang telah diwawancarai (mahasiswa, dosen wali, staf BAA, staf keuangan, pimpinan) dalam satu sesi.
    - **Informasi yang Dicari:**
        - **Memvalidasi Alur Kerja:** Memetakan alur kerja sistem baru secara visual (misalnya, dengan *storyboard* atau diagram alur) dan mendapatkan persetujuan dari semua pihak.
        - **Menyelesaikan Konflik:** Jika ada kebutuhan yang bertentangan (misalnya, Bagian Keuangan ingin proses pembayaran yang ketat, sementara BAA ingin fleksibilitas bagi mahasiswa), sesi ini digunakan untuk mencari kompromi.
        - **Memprioritaskan Fitur:** Bersama-sama menentukan fitur mana yang paling penting untuk diluncurkan pada versi pertama sistem.

Dengan menggabungkan kelima teknik ini, tim proyek akan memiliki pemahaman yang holistik, tervalidasi, dan disepakati bersama mengenai kebutuhan Sistem Pendaftaran Ulang Mahasiswa yang baru, sehingga meminimalkan risiko kegagalan di tahap selanjutnya.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Valacich, J. S., & George, J. F. (Edisi terbaru). *Modern Systems Analysis and Design*. Pearson.

Bacaan Tambahan (Dianjurkan)

- Gottesdiener, E. (2002). *Requirements by Collaboration: Workshops for Defining Needs*. Addison-Wesley Professional. (Buku klasik yang membahas secara mendalam tentang pelaksanaan workshop kebutuhan yang efektif).
- Wiegers, K., & Beatty, J. (2013). *Software Requirements, 3rd Edition*. Microsoft Press. (Referensi komprehensif tentang berbagai aspek rekayasa kebutuhan, melampaui sekedar pengumpulan data).
- Artikel-artikel dari Project Management Institute (PMI) atau International Institute of Business Analysis (IIBA)

yang membahas praktik terbaik dalam elisitasi dan manajemen kebutuhan.

Bab 5: Analisis Kebutuhan

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Membedakan secara tegas antara kebutuhan fungsional (*functional requirements*) dan kebutuhan non-fungsional (*non-functional requirements*) beserta contoh-contohnya.
- Memahami tujuan, komponen, dan notasi standar dari Diagram Alir Data (DFD) untuk memodelkan proses bisnis dan aliran data.
- Menerapkan teknik dekomposisi untuk membuat DFD berjenjang, mulai dari Diagram Konteks (Level 0) hingga diagram level yang lebih rinci, dengan memastikan prinsip keseimbangan (*balancing*).
- Memahami tujuan, komponen, dan notasi standar (termasuk *Crow's Foot*) dari *Entity Relationship Diagram* (ERD) untuk memodelkan struktur data.
- Menganalisis dan menentukan entitas, atribut, kunci primer (*primary key*), serta hubungan (kardinalitas dan modalitas) antar entitas dalam sebuah sistem.
- Menjelaskan struktur, tujuan, dan pentingnya dokumen Spesifikasi Kebutuhan Sistem (*System Requirements Specification* - SRS) sebagai kontrak formal antara pemangku kepentingan dan tim pengembang.

Selamat datang di Bab 5, tahap di mana seorang analis sistem bertransformasi dari seorang detektif menjadi seorang arsitek.

Pada Bab 4, kita telah menguasai berbagai teknik untuk mengumpulkan "bahan mentah", fakta, opini, masalah, dan keinginan dari para pemangku kepentingan. Kini, tugas kita adalah mengambil bahan mentah yang seringkali tidak terstruktur tersebut dan mulai membentuknya menjadi sesuatu yang logis, terorganisir, dan dapat dipahami secara teknis. Fase analisis kebutuhan adalah jembatan krusial antara pengumpulan informasi dan perancangan sistem. Di sinilah kita berhenti bertanya "apa masalahnya?" dan mulai mendefinisikan secara presisi "apa yang harus dilakukan oleh sistem?"

Dalam bab ini, kita akan mempelajari cara memilah dan mengklasifikasikan kebutuhan menjadi dua kategori fundamental: fungsional dan non-fungsional. Selanjutnya, kita akan terjun ke dalam dua alat pemodelan visual yang paling kuat dan fundamental dalam analisis sistem. Pertama, **Diagram Alir Data (DFD)**, yang akan membantu kita memetakan bagaimana data bergerak dan diubah oleh proses-proses di dalam sistem. Kedua, **Entity Relationship Diagram (ERD)**, yang akan menjadi cetak biru kita untuk merancang struktur data dan basis data yang kokoh. Terakhir, kita akan belajar bagaimana menuangkan semua hasil analisis ini ke dalam sebuah dokumen formal yang disebut **Spesifikasi Kebutuhan Sistem (SRS)**, yang akan berfungsi sebagai landasan dan kontrak untuk seluruh sisa proyek. Menguasai bab ini berarti Anda mampu mengubah percakapan dan observasi menjadi model dan spesifikasi yang jelas, sebuah keahlian inti dari seorang analis sistem profesional.

5.1 Identifikasi Kebutuhan Fungsional dan Non-fungsional

Setelah informasi terkumpul, langkah pertama dalam analisis adalah memilah dan mengkategorikan kebutuhan. Ini adalah proses distilasi, di mana kita menyaring esensi dari apa yang harus dilakukan sistem. Secara umum, kebutuhan sistem dibagi menjadi dua kategori utama: kebutuhan fungsional dan kebutuhan non-fungsional. Memahami perbedaan ini sangat penting karena keduanya memiliki implikasi yang berbeda pada desain, pengembangan, dan pengujian sistem.

Kebutuhan Fungsional (Functional Requirements)

Kebutuhan fungsional mendefinisikan **apa yang harus dilakukan oleh sistem**. Kebutuhan ini menggambarkan proses, layanan, atau fungsi spesifik yang harus disediakan oleh sistem kepada pengguna. Ini adalah tentang perilaku sistem, aktivitas yang dapat dilakukan, informasi yang dapat dikelola, dan interaksi yang didukung. Kebutuhan fungsional seringkali dapat diungkapkan dalam bentuk "Sistem harus dapat [melakukan aksi]".

Contoh kebutuhan fungsional untuk sistem informasi perpustakaan:

- Sistem harus dapat **mencari buku** berdasarkan judul, penulis, atau ISBN.
- Sistem harus memungkinkan anggota untuk **meminjam** hingga tiga buku sekaligus.
- Sistem harus secara otomatis **menghitung denda** untuk buku yang terlambat dikembalikan.
- Sistem harus dapat **mengelola data anggota**, termasuk pendaftaran anggota baru dan pembaruan data.
- Sistem harus **menghasilkan laporan bulanan** tentang buku yang paling sering dipinjam.

Kebutuhan fungsional adalah dasar untuk merancang alur kerja, antarmuka pengguna, dan logika bisnis dalam sistem. Saat pengujian, setiap kebutuhan fungsional harus dapat diverifikasi secara langsung: apakah sistem benar-benar dapat melakukan fungsi X? Ya atau tidak.

Kebutuhan Non-fungsional (Non-functional Requirements)

Jika kebutuhan fungsional adalah tentang "apa", maka kebutuhan non-fungsional adalah tentang "**bagaimana**". Kebutuhan ini mendefinisikan kualitas, karakteristik, dan batasan dari sistem. Mereka tidak menggambarkan fungsi bisnis secara langsung, tetapi menetapkan kriteria tentang seberapa baik sistem menjalankan fungsinya tersebut. Kebutuhan non-fungsional sering disebut sebagai "kualitas" sistem dan sangat penting untuk kepuasan pengguna dan keberhasilan teknis.
Kebutuhan non-fungsional dapat diklasifikasikan lebih lanjut ke dalam beberapa kategori:

Tabel 14. Kategori dan Contoh Kebutuhan Non-fungsional

Kategori	Deskripsi	Contoh untuk Sistem E-commerce
Kinerja (Performance)	Seberapa cepat dan responsif sistem harus bekerja.	• Halaman produk harus dimuat dalam waktu kurang dari 2 detik. • Sistem harus mampu menangani 1.000 transaksi per menit selama periode promosi.

Keamanan (Security)	Perlindungan sistem dan data dari akses yang tidak sah.	• Semua data pembayaran pelanggan harus dienkripsi menggunakan standar AES-256. • Pengguna harus mengganti kata sandi setiap 90 hari.
Keandalan (Reliability)	Seberapa sering sistem boleh mengalami kegagalan.	• Sistem harus memiliki *uptime* (waktu aktif) sebesar 99.9%. • Waktu henti untuk pemeliharaan tidak boleh lebih dari 2 jam per bulan.
Kegunaan (Usability)	Seberapa mudah sistem dapat dipelajari dan digunakan oleh pengguna.	• Pengguna baru harus dapat menyelesaikan proses pembelian pertama dalam waktu kurang dari 5 menit. • Sistem harus kompatibel dengan pembaca layar untuk pengguna tunanetra.
Skalabilitas (Scalability)	Kemampuan sistem untuk menangani pertumbuhan jumlah pengguna atau data di masa depan.	Arsitektur sistem harus dirancang untuk dapat menangani peningkatan lalu lintas sebesar 50% per tahun tanpa penurunan kinerja yang signifikan.
Kompatibilitas (Compatibility)	Kemampuan sistem untuk berinteraksi dengan sistem lain.	• Sistem harus dapat terintegrasi dengan sistem pembayaran pihak ketiga melalui API yang disediakan. • Laporan penjualan harus dapat diekspor dalam format.csv dan.pdf.

Mengabaikan kebutuhan non-fungsional adalah kesalahan fatal. Sebuah sistem yang secara fungsional lengkap tetapi sangat lambat, tidak aman, atau sulit digunakan akan dianggap gagal oleh penggunanya. Oleh karena itu, analis sistem harus

secara proaktif menggali kebutuhan ini selama proses pengumpulan data.

5.2 Diagram Alir Data (DFD)

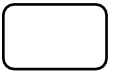
Setelah kebutuhan diidentifikasi, langkah selanjutnya adalah memodelkannya secara visual. Pemodelan membantu kita untuk memahami sistem yang kompleks, mengidentifikasi inkonsistensi, dan berkomunikasi secara efektif dengan pemangku kepentingan. Salah satu alat pemodelan klasik dan paling fundamental untuk analisis proses adalah **Diagram Alir Data (Data Flow Diagram - DFD)**.

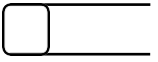
DFD adalah representasi grafis yang menggambarkan aliran data melalui sebuah sistem. DFD berfokus pada **proses** (apa yang dilakukan sistem untuk mengubah data) dan **data** (informasi apa yang bergerak di antara proses). Penting untuk diingat: DFD menunjukkan *aliran data*, bukan *alur kontrol* (seperti *flowchart*). DFD tidak menunjukkan urutan atau waktu eksekusi proses.

Komponen dan Notasi DFD

DFD menggunakan empat simbol dasar. Notasi yang paling umum digunakan adalah notasi Gane & Sarson.

Tabel 15. Simbol-simbol DFD (Notasi Gane & Sarson)

Simbol	Nama	Deskripsi dan Aturan
	**Proses (Process)**	Merepresentasikan sebuah aktivitas atau fungsi yang mengubah data. **Aturan:** - Diberi label dengan kata kerja + kata benda (misal: "1.0 Proses Pesanan"). - Harus memiliki setidaknya satu input

Simbol	Nama	Deskripsi dan Aturan
		dan satu output. - Diberi nomor unik.
	**Aliran Data (Data Flow)**	Merepresentasikan pergerakan data antara proses, data store, dan entitas eksternal. **Aturan:** - Diberi label dengan kata benda (misal: "Data Pesanan"). - Panah menunjukkan arah aliran. - Data tidak dapat mengalir langsung antara dua entitas eksternal atau dua data store.
	**Penyimpanan Data (Data Store)**	Merepresentasikan tempat di mana data disimpan (misal: file, tabel basis data). **Aturan:** - Diberi label dengan kata benda jamak (misal: "D1: Pelanggan"). - Harus memiliki aliran data masuk (untuk menulis) dan keluar (untuk membaca).
	**Entitas Eksternal (External Entity)**	Merepresentasikan sumber atau tujuan data di luar batas sistem (misal: orang, departemen, sistem lain). **Aturan:** - Diberi label dengan kata benda (misal: "Pelanggan"). - Juga dikenal sebagai *source* atau *sink*.

DFD Berjenjang (Leveled DFD)

Sistem yang kompleks tidak dapat digambarkan dalam satu diagram tunggal tanpa menjadi terlalu rumit. Oleh karena itu, DFD menggunakan pendekatan dekomposisi atau *leveling*, di mana sistem dipecah menjadi lapisan-lapisan yang semakin detail.

- **Diagram Konteks (Context Diagram / Level 0):** Ini adalah level tertinggi dan paling abstrak. Diagram ini menampilkan seluruh sistem sebagai **satu proses tunggal** (biasanya diberi nomor 0) dan menunjukkan semua entitas eksternal yang berinteraksi dengannya serta aliran data utama yang masuk dan keluar dari sistem. Tujuannya adalah untuk mendefinisikan ruang lingkup dan batasan sistem.
- **Diagram Level 1:** Diagram ini adalah "ledakan" (*explosion*) dari Diagram Konteks. Proses tunggal pada Level 0 dipecah menjadi beberapa proses utama yang membentuk sistem. Diagram Level 1 menunjukkan proses-proses utama ini, aliran data di antara mereka, dan penyimpanan data (*data stores*) yang mereka gunakan.
- **Diagram Level 2 (dan seterusnya):** Jika salah satu proses di Level 1 masih terlalu kompleks, proses tersebut dapat dipecah lagi menjadi sub-proses yang lebih detail dalam Diagram Level 2. Proses ini dapat berlanjut ke Level 3, 4, dan seterusnya hingga setiap proses cukup sederhana untuk dipahami.

Prinsip Kunci: **Keseimbangan (Balancing)**. Saat melakukan dekomposisi, aliran data yang masuk dan keluar dari sebuah proses "induk" di level yang lebih tinggi harus sama persis dengan total aliran data yang masuk dan keluar dari diagram "anak" di level yang lebih rendah.

Langkah-langkah Teknis Membuat DFD

1. **Identifikasi Entitas Eksternal dan Aliran Data Utama:** Buat daftar semua sumber dan tujuan data di luar sistem (pelanggan, pemasok, departemen lain) dan data utama yang mereka kirim atau terima dari sistem.
2. **Buat Diagram Konteks (Level 0):** Gambar satu lingkaran di tengah untuk mewakili seluruh sistem. Letakkan entitas eksternal di sekelilingnya. Gambar panah aliran data antara entitas dan sistem.
3. **Identifikasi Proses Utama:** Pikirkan tentang fungsi-fungsi atau aktivitas utama yang dilakukan sistem untuk mengubah input menjadi output. Ini akan menjadi proses-proses di Diagram Level 1 Anda.
4. **Buat Diagram Level 1:** "Ledakkan" proses tunggal dari Diagram Konteks. Gambar proses-proses utama yang telah Anda identifikasi. Tambahkan penyimpanan data (*data stores*) yang diperlukan. Hubungkan proses, data store, dan entitas eksternal dengan aliran data yang sesuai.
5. **Periksa Keseimbangan:** Pastikan semua aliran data yang masuk dan keluar dari Diagram Konteks (Level 0) juga muncul di Diagram Level 1.
6. **Dekomposisi ke Level Lebih Rendah:** Untuk setiap proses di Level 1 yang masih kompleks, ulangi langkah 3-5 untuk membuat Diagram Level 2. Lanjutkan proses ini hingga setiap proses di level terendah cukup sederhana dan jelas.
7. **Validasi:** Tinjau semua DFD dengan pemangku kepentingan untuk memastikan diagram tersebut secara akurat merepresentasikan proses bisnis.

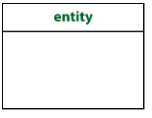
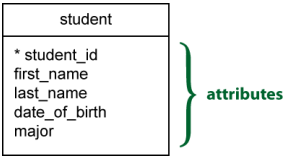
5.3 Entity Relationship Diagram (ERD)

Jika DFD menjawab pertanyaan "bagaimana data bergerak?", maka **Entity Relationship Diagram (ERD)** menjawab pertanyaan "data apa yang perlu disimpan dan bagaimana hubungannya?". ERD adalah model konseptual yang menggambarkan struktur data dalam sebuah sistem.[3] Ini adalah alat fundamental untuk merancang basis data. ERD berfokus pada tiga konsep inti: entitas, atribut, dan hubungan.

Komponen dan Notasi ERD

Ada beberapa notasi untuk ERD, tetapi salah satu yang paling populer dan deskriptif adalah notasi **Crow's Foot**.

Tabel 16. Simbol-simbol ERD (Notasi Crow's Foot)

Simbol	Nama	Deskripsi
	**Entitas (Entity)**	Merepresentasikan objek, orang, tempat, atau konsep yang datanya perlu disimpan oleh sistem. **Aturan:** Diberi label dengan kata benda tunggal (misal: "Mahasiswa", "Buku").
	**Atribut (Attribute)**	Merepresentasikan properti atau karakteristik dari sebuah entitas. **Aturan:** Ditulis di dalam kotak entitas. Salah satu atribut (atau kombinasi) harus menjadi **Kunci Primer (Primary Key - PK)** yang unik mengidentifikasi setiap instans entitas.
	Hubungan (Relationship)	Merepresentasikan asosiasi antara dua entitas. **Aturan:** Digambarkan sebagai garis yang menghubungkan

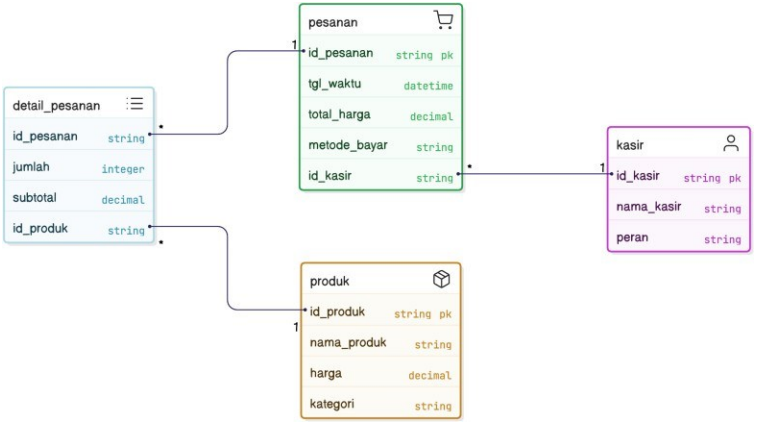
Simbol	Nama	Deskripsi
		entitas, dengan label kata kerja yang menjelaskan hubungan tersebut (misal: "meminjam").

Kardinalitas dan Modalitas

Notasi di ujung garis hubungan (bentuk "kaki gagak" atau *crow's foot*) adalah bagian terpenting dari ERD karena mendefinisikan aturan bisnis yang presisi. Notasi ini menunjukkan dua hal:

1. **Kardinalitas (Cardinality):** Menjawab pertanyaan "berapa banyak?". Ini menunjukkan jumlah maksimum instans satu entitas yang dapat berhubungan dengan satu instans entitas lain.
    - **Satu (One):** Digambarkan dengan satu garis tegak lurus (|).
    - **Banyak (Many):** Digambarkan dengan tiga garis cabang seperti kaki gagak (<).
2. **Modalitas (Modality) / Opsionalitas (Optionality):** Menjawab pertanyaan "haruskah?". Ini menunjukkan apakah hubungan tersebut wajib (*mandatory*) atau opsional (*optional*).
    - **Wajib (Mandatory):** Digambarkan dengan satu garis tegak lurus (|). Artinya, sebuah instans harus berpartisipasi dalam hubungan tersebut.
    - **Opsional (Optional):** Digambarkan dengan lingkaran (O). Artinya, sebuah instans tidak harus berpartisipasi dalam hubungan tersebut.

Kombinasi dari kedua notasi ini menghasilkan hubungan yang sangat deskriptif, seperti "Satu dan Hanya Satu", "Satu atau Banyak", "Nol atau Satu", dan "Nol atau Banyak".



Gambar 13. Diagram ERD Point of Sales

Langkah-langkah Teknis Membuat ERD

1. **Identifikasi Entitas:** Tinjau kembali hasil pengumpulan kebutuhan dan cari kata-kata benda yang penting yang mewakili orang, tempat, benda, atau peristiwa yang datanya perlu dilacak oleh sistem (misal: Pelanggan, Produk, Pesanan).
2. **Identifikasi Atribut:** Untuk setiap entitas, tentukan properti atau data yang perlu disimpan (misal: untuk entitas Pelanggan, atributnya adalah ID Pelanggan, Nama, Alamat, Email).
3. **Tentukan Kunci Primer (Primary Key):** Pilih satu atau lebih atribut yang secara unik mengidentifikasi setiap baris data dalam sebuah entitas (misal: ID Pelanggan).

4. **Identifikasi Hubungan:** Cari kata kerja yang menghubungkan entitas-entitas yang telah Anda identifikasi (misal: Pelanggan *membuat* Pesanan; Pesanan *terdiri dari* Produk).
5. **Tentukan Kardinalitas dan Modalitas:** Untuk setiap hubungan, ajukan pertanyaan bisnis yang spesifik:
    - *Kardinalitas:* "Satu Pelanggan dapat membuat berapa banyak Pesanan?" (Banyak). "Satu Pesanan dapat dibuat oleh berapa banyak Pelanggan?" (Satu). Jadi, hubungannya adalah Satu-ke-Banyak.
    - *Modalitas:* "Apakah sebuah Pesanan *harus* memiliki seorang Pelanggan?" (Ya, wajib). "Apakah seorang Pelanggan *harus* memiliki sebuah Pesanan?" (Tidak, pelanggan baru mungkin belum memesan, jadi opsional).
6. **Gambar ERD:** Gunakan perangkat lunak pemodelan atau gambar secara manual. Kotak untuk entitas, garis untuk hubungan, dan notasi *Crow's Foot* untuk kardinalitas/modalitas.
7. **Atasi Hubungan Banyak-ke-Banyak (Many-to-Many):** Hubungan M:N (misal: Mahasiswa *mengambil* Mata Kuliah) tidak dapat diimplementasikan langsung dalam basis data relasional. Hubungan ini harus dipecah menjadi dua hubungan Satu-ke-Banyak dengan membuat **entitas asosiatif** atau **entitas penghubung** di tengah (misal: entitas "KRS" yang menghubungkan Mahasiswa dan Mata Kuliah).

5.4 Spesifikasi Kebutuhan Sistem

Setelah semua kebutuhan diidentifikasi, diklasifikasikan, dan dimodelkan, langkah terakhir dalam fase analisis adalah

mendokumentasikannya secara formal. Dokumen ini disebut **Spesifikasi Kebutuhan Sistem (System Requirements Specification - SRS)** atau terkadang *Software Requirements Specification*.

SRS adalah deskripsi lengkap tentang perilaku sistem yang akan dikembangkan. Ini adalah dokumen yang paling penting karena berfungsi sebagai:

- **Kontrak:** Menjadi kesepakatan formal antara klien/pengguna dan tim pengembang tentang apa yang akan dibangun.
- **Sumber Kebenaran Tunggal:** Menjadi referensi utama bagi semua pemangku kepentingan (analis, desainer, pemrogram, penguji, manajer proyek).
- **Dasar untuk Desain dan Pengujian:** Tim desain menggunakan SRS sebagai input untuk membuat arsitektur sistem, dan tim penguji menggunakannya untuk membuat kasus uji (*test cases*).

Struktur Umum Dokumen SRS

Meskipun formatnya dapat bervariasi, SRS yang baik biasanya mengikuti struktur standar (seperti yang diadaptasi dari IEEE Standard 830) untuk memastikan kelengkapan.

Tabel 17. Contoh Kerangka Dokumen SRS

Bagian	Sub-Bagian	Deskripsi
1. Pendahuluan	1.1 Tujuan 1.2 Ruang Lingkup 1.3 Definisi & Akronim 1.4 Referensi 1.5 Tinjauan Dokumen	Menjelaskan tujuan dokumen SRS, ruang lingkup produk perangkat lunak yang akan dibuat, definisi istilah, daftar dokumen

Bagian	Sub-Bagian	Deskripsi
		referensi, dan struktur sisa dokumen.
2. Deskripsi Umum	2.1 Perspektif Produk 2.2 Fungsi Produk 2.3 Karakteristik Pengguna 2.4 Batasan Umum 2.5 Asumsi & Ketergantungan	Memberikan gambaran umum tentang produk, hubungannya dengan sistem lain, ringkasan fungsi utama, deskripsi pengguna yang dituju, serta batasan dan asumsi yang memengaruhi pengembangan.
3. Kebutuhan Spesifik	3.1 Kebutuhan Fungsional 3.2 Kebutuhan Non-fungsional 3.3 Kebutuhan Antarmuka Eksternal	Ini adalah inti dari SRS. Bagian ini merinci semua kebutuhan yang harus dipenuhi oleh sistem. - **Fungsional:** Deskripsi detail setiap fungsi, input, proses, dan output. Dapat diorganisir berdasarkan fitur. - **Non-fungsional:** Kebutuhan kinerja, keamanan, keandalan, dll. yang spesifik dan terukur. - **Antarmuka:** Deskripsi antarmuka pengguna (GUI), antarmuka dengan perangkat keras, dan antarmuka dengan sistem perangkat lunak lain.
Lampiran		Informasi tambahan seperti DFD, ERD, prototipe antarmuka, dll.

Kunci dari SRS yang baik adalah **presisi dan kejelasan**. Setiap pernyataan kebutuhan harus tidak ambigu, dapat diuji (*testable*), dan lengkap. Menulis SRS yang baik adalah sebuah seni yang membutuhkan latihan, tetapi merupakan investasi yang sangat berharga untuk mencegah kesalahpahaman dan kegagalan proyek di kemudian hari.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Berikan satu contoh kebutuhan fungsional dan satu contoh kebutuhan non-fungsional (kategori kinerja) untuk sistem ATM perbankan. Jelaskan mengapa pembedaan ini penting.
2. Apa perbedaan fundamental antara Diagram Alir Data (DFD) dan *Entity Relationship Diagram* (ERD)? Kapan Anda akan menggunakan masing-masing model tersebut dalam sebuah proyek?
3. Dalam sebuah ERD untuk sistem perpustakaan, terdapat hubungan antara entitas BUKU dan PEMINJAM. Jelaskan apa arti dari hubungan dengan kardinalitas/modalitas "Satu atau Banyak" dari sisi PEMINJAM ke BUKU, dan "Nol atau Banyak" dari sisi BUKU ke PEMINJAM.

Solusi dan Pembahasan Soal Konseptual

1. **Kebutuhan Sistem ATM:**
    - **Fungsional:** "Sistem harus memungkinkan nasabah untuk menarik tunai dari rekening tabungan mereka."
    - **Non-fungsional (Kinerja):** "Sistem harus menyelesaikan transaksi penarikan tunai dalam waktu rata-rata 15 detik setelah PIN dimasukkan dengan benar."
    - **Pentingnya Pembedaan:** Pembedaan ini penting karena keduanya diukur dengan cara

yang berbeda. Kebutuhan fungsional diuji dengan basis "bisa/tidak bisa". Kebutuhan non-fungsional diuji dengan pengukuran dan metrik. Seorang pemrogram bisa saja membuat fungsi tarik tunai yang benar (fungsional terpenuhi), tetapi jika prosesnya memakan waktu 2 menit, sistem tersebut gagal memenuhi kebutuhan non-fungsional dan akan membuat nasabah frustrasi.

2. **Perbedaan DFD dan ERD:**
    - **Perbedaan Fundamental:** DFD memodelkan **proses dan aliran data** (dinamis), menjawab pertanyaan "apa yang dilakukan sistem dan bagaimana data bergerak?". ERD memodelkan **struktur data dan hubungannya** (statis), menjawab pertanyaan "data apa yang disimpan sistem?".
    - **Penggunaan:** Anda akan menggunakan **DFD** pada tahap awal analisis untuk memahami dan memvisualisasikan proses bisnis yang ada (*as-is*) dan yang diusulkan (*to-be*). Ini membantu dalam berkomunikasi dengan pengguna tentang alur kerja. Anda akan menggunakan **ERD** setelah proses dipahami, untuk merancang struktur basis data yang akan mendukung proses-proses tersebut. ERD adalah cetak biru untuk basis data.
3. **Interpretasi Kardinalitas/Modalitas ERD Perpustakaan:**
    - **PEMINJAM ke BUKU ("Satu atau Banyak"):** Ini berarti satu PEMINJAM dapat meminjam

minimal satu buku dan **maksimal banyak** buku. Aturan bisnisnya adalah seseorang tidak bisa disebut "Peminjam" jika tidak sedang meminjam setidaknya satu buku.

- **BUKU ke PEMINJAM ("Nol atau Banyak"):** Ini berarti satu BUKU dapat dipinjam oleh **minimal nol** peminjam (artinya buku tersebut sedang ada di rak/tersedia) atau **maksimal banyak** peminjam (secara berurutan dari waktu ke waktu, bukan pada saat yang sama).

Studi Kasus: Analisis Sistem Pemesanan "Kopi Kita"

- **Konteks:** Melanjutkan studi kasus dari Bab 2, "Kopi Kita" telah berkembang dan sistem manualnya tidak lagi memadai. Manajemen memutuskan untuk membangun sistem *Point of Sale* (POS) berbasis tablet untuk kasir di setiap cabang.
- **Informasi yang Terkumpul:**
  - Kasir harus dapat melihat daftar menu (kopi, non-kopi, makanan) beserta harganya.
  - Kasir harus dapat memasukkan pesanan pelanggan, memilih item, dan jumlahnya.
  - Sistem harus dapat menghitung total tagihan, termasuk pajak 10%.
  - Sistem harus dapat menerima pembayaran tunai atau kartu debit/kredit.
  - Setelah pembayaran, sistem harus mencetak struk untuk pelanggan dan mengirimkan pesanan ke dapur/barista secara otomatis.

- Manajer cabang harus dapat melihat laporan penjualan harian.
- Sistem harus cepat, kasir tidak boleh menunggu lebih dari 2 detik untuk setiap transaksi.
- Hanya manajer yang boleh mengakses fungsi laporan.

Solusi dan Analisis Studi Kasus

1. Identifikasi Kebutuhan Fungsional dan Non-fungsional

- **Kebutuhan Fungsional:**
  - Sistem harus menampilkan daftar menu dan harga.
  - Sistem harus dapat membuat pesanan baru.
  - Sistem harus dapat menambahkan item ke dalam pesanan.
  - Sistem harus menghitung total tagihan, termasuk pajak 10%.
  - Sistem harus memproses pembayaran tunai.
  - Sistem harus memproses pembayaran kartu.
  - Sistem harus mencetak struk.
  - Sistem harus mengirimkan detail pesanan ke dapur/barista.
  - Sistem harus menghasilkan laporan penjualan harian.
- **Kebutuhan Non-fungsional:**

- **Kinerja:** Waktu respons untuk setiap transaksi tidak boleh lebih dari 2 detik.
- **Keamanan:** Hanya pengguna dengan peran "Manajer" yang dapat mengakses fungsi laporan penjualan.
- **Kegunaan:** (Tersirat) Antarmuka harus cukup intuitif untuk digunakan oleh kasir dengan pelatihan minimal.

2. Diagram Alir Data (DFD)

- **Diagram Konteks (Level 0):**
  - **Entitas Eksternal:** Pelanggan, Kasir, Dapur/Barista, Manajer, Sistem Bank.
  - **Proses Tunggal:** 0 - Sistem POS Kopi Kita.
  - **Aliran Data Utama:**
    - Pelanggan -> Sistem: Pesanan Lisan
    - Kasir -> Sistem: Input Pesanan, Input Pembayaran
    - Sistem -> Pelanggan: Struk, Tagihan
    - Sistem -> Dapur/Barista: Detail Pesanan
    - Sistem -> Sistem Bank: Data Transaksi Kartu
    - Sistem Bank -> Sistem: Konfirmasi Pembayaran
    - Manajer -> Sistem: Permintaan Laporan
    - Sistem -> Manajer: Laporan Penjualan
- **Diagram Level 1 (Contoh Sederhana):**
  - **Proses:** 1.0 Catat Pesanan, 2.0 Proses Pembayaran, 3.0 Buat Laporan.

- **Data Store:** D1: Menu, D2: Transaksi Penjualan.
- **Aliran Data:** "Input Pesanan" dari Kasir masuk ke proses 1.0. Proses 1.0 membaca "Data Menu" dari D1 dan menghasilkan "Detail Tagihan" ke proses 2.0 dan "Detail Pesanan" ke Dapur/Barista. Proses 2.0 menerima "Input Pembayaran" dari Kasir dan "Detail Tagihan" dari 1.0, kemudian menulis "Data Transaksi" ke D2 dan menghasilkan "Struk" untuk Pelanggan. Proses 3.0 membaca "Data Transaksi" dari D2 berdasarkan "Permintaan Laporan" dari Manajer, dan menghasilkan "Laporan Penjualan".

3. Entity Relationship Diagram (ERD)

- **Entitas:** PRODUK, PESANAN, DETAIL_PESANAN, KASIR.
- **Atribut (Contoh):**
  - PRODUK: **ID_Produk (PK)**, Nama_Produk, Harga, Kategori.
  - PESANAN: **ID_Pesanan (PK)**, Tgl_Waktu, Total_Harga, Metode_Bayar, *ID_Kasir (FK)*.
  - DETAIL_PESANAN: ***ID_Pesanan (FK)***, ***ID_Produk (FK)***, Jumlah, Subtotal. (Ini adalah entitas asosiatif untuk mengatasi hubungan M:N antara Pesanan dan Produk).
  - KASIR: **ID_Kasir (PK)**, Nama_Kasir, Peran.
- **Hubungan:**
  - KASIR (1) ---< PESANAN (0..*): Satu Kasir dapat memproses Nol atau Banyak Pesanan.
  - PESANAN (1..*) >---< *PRODUK (1..)*: Hubungan M:N yang dipecah oleh DETAIL_PESANAN.

- PESANAN (1) ---< DETAIL_PESANAN (1..*): Satu Pesanan harus terdiri dari minimal Satu Detail Pesanan.
- PRODUK (1) >--- DETAIL_PESANAN (0..*): Satu Produk bisa ada di Nol atau Banyak Detail Pesanan.

4. Cuplikan Spesifikasi Kebutuhan Sistem (SRS)

- **ID Kebutuhan:** F-004
- **Nama Kebutuhan:** Perhitungan Total Tagihan
- **Deskripsi:** Sistem harus mampu menghitung total tagihan untuk sebuah pesanan. Perhitungan harus mencakup subtotal dari semua item yang dipesan ditambah dengan Pajak Pertambahan Nilai (PPN) sebesar 10% dari subtotal.
- **Input:** Daftar item pesanan beserta jumlah dan harga satuan.
- **Proses:**
    1. Untuk setiap item dalam pesanan, hitung subtotal item (Jumlah x Harga Satuan).
    2. Jumlahkan semua subtotal item untuk mendapatkan Subtotal Pesanan.
    3. Hitung nilai PPN (Subtotal Pesanan x 0.10).
    4. Hitung Total Tagihan (Subtotal Pesanan + PPN).
- **Output:** Total Tagihan yang ditampilkan kepada kasir dan dicetak di struk.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Valacich, J. S., & George, J. F. (Edisi terbaru). *Modern Systems Analysis and Design*. Pearson.

Bacaan Tambahan (Dianjurkan)

- Wiegers, K., & Beatty, J. (Edisi terbaru). *Software Requirements*. Microsoft Press. (Buku ini adalah referensi standar industri untuk rekayasa kebutuhan perangkat lunak dan memberikan detail yang sangat mendalam tentang penulisan SRS yang berkualitas).
- *IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998*. (Meskipun sudah berumur, standar ini masih menjadi dasar bagi banyak templat dan praktik SRS modern).
- Artikel dan tutorial online tentang pemodelan DFD dan ERD menggunakan notasi spesifik (misalnya, Gane & Sarson, Yourdon, Crow's Foot, Chen).

Bab 6: Desain Sistem

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan transisi dari fase analisis ("apa") ke fase desain ("bagaimana") dalam siklus hidup pengembangan sistem.
- Menerapkan prinsip-prinsip fundamental desain antarmuka pengguna (UI) dan pengalaman pengguna (UX) untuk menciptakan sistem yang efektif dan mudah digunakan.
- Menguasai langkah-langkah teknis dalam desain UI, mulai dari pembuatan *wireframe*, *mockup*, hingga prototipe interaktif.
- Mengonversi model data konseptual (ERD) menjadi desain basis data logis dan fisik, termasuk penerapan kunci primer dan kunci asing (*foreign key*).
- Memahami dan menerapkan konsep normalisasi basis data (hingga Bentuk Normal Ketiga/3NF) untuk mengurangi redundansi dan meningkatkan integritas data.
- Mengidentifikasi dan membandingkan berbagai pola arsitektur sistem, seperti arsitektur *client-server*, *three-tier*, dan berbasis layanan (*service-oriented*).
- Merancang logika proses bisnis yang detail menggunakan alat seperti *pseudocode*, tabel keputusan (*decision table*), dan pohon keputusan (*decision tree*).

Selamat datang di Bab 6. Pada bab-bab sebelumnya, kita telah berfokus pada fase analisis, sebuah proses investigasi untuk memahami masalah bisnis dan mendefinisikan *apa* yang harus dilakukan oleh sistem. Kita telah mengumpulkan kebutuhan, memodelkan aliran data dengan DFD, dan memetakan struktur data dengan ERD. Sekarang, kita memasuki fase yang sangat menentukan: **desain sistem**. Fase ini adalah tentang transisi dari "apa" ke "bagaimana". Jika analisis adalah tentang membuat daftar keinginan dan persyaratan untuk sebuah rumah, maka desain adalah proses di mana arsitek dan insinyur sipil membuat cetak biru (*blueprint*) yang detail, lengkap dengan denah lantai, spesifikasi material, dan rencana struktur.

Dalam bab ini, kita akan mengubah hasil analisis kita menjadi spesifikasi teknis yang siap untuk diimplementasikan oleh para pemrogram. Kita akan memulai dengan merancang bagian yang paling terlihat oleh pengguna: **antarmuka pengguna**, memastikan sistem tidak hanya fungsional tetapi juga nyaman dan intuitif untuk digunakan. Selanjutnya, kita akan menerjemahkan ERD kita menjadi **desain basis data** yang solid dan efisien, pondasi di mana semua data sistem akan disimpan. Kita juga akan membuat keputusan strategis tentang **arsitektur sistem**, menentukan bagaimana komponen-komponen utama sistem akan diorganisir dan berkomunikasi. Terakhir, kita akan merinci logika di balik setiap **proses bisnis**, memastikan setiap aturan dan keputusan diimplementasikan dengan benar. Fase desain adalah tempat di mana kreativitas bertemu dengan ketelitian teknis. Mari kita mulai membangun cetak biru untuk solusi kita.

6.1 Desain Antarmuka Pengguna (UI Design)

Desain Antarmuka Pengguna atau *User Interface* (UI) *Design* adalah proses merancang tampilan visual dan interaktif dari sebuah sistem. UI adalah jembatan antara pengguna dan fungsionalitas sistem. Secanggih apa pun logika di balik layar, jika antarmukanya membingungkan, tidak efisien, atau tidak menyenangkan untuk digunakan, sistem tersebut berisiko tinggi untuk ditolak oleh pengguna. Tujuan utama dari desain UI adalah untuk menciptakan pengalaman pengguna (*User Experience* - UX) yang positif, di mana pengguna dapat mencapai tujuan mereka dengan efektif, efisien, dan memuaskan.

Prinsip-prinsip Desain UI yang Baik

Desain UI yang efektif tidak hanya soal estetika, tetapi juga didasarkan pada prinsip-prinsip psikologi dan interaksi manusia-komputer. Beberapa prinsip fundamental yang harus dipegang oleh seorang desainer sistem adalah:

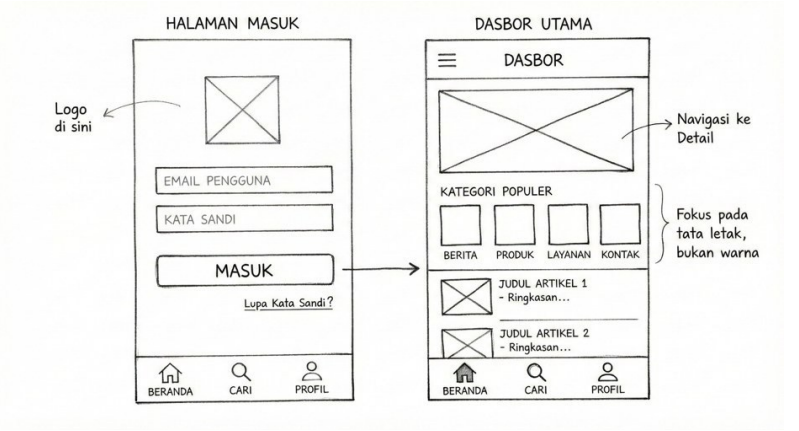
1. **Konsistensi (Consistency):** Elemen-elemen yang serupa harus memiliki tampilan dan perilaku yang sama di seluruh sistem. Misalnya, tombol "Simpan" harus selalu berada di lokasi yang sama dan memiliki ikon yang sama di setiap layar. Konsistensi membuat sistem dapat diprediksi dan lebih cepat untuk dipelajari.
2. **Kejelasan (Clarity):** Antarmuka harus jelas dan mudah dipahami. Hindari ambiguitas. Gunakan label yang jelas, ikon yang universal, dan tata letak yang logis. Pengguna tidak seharusnya perlu menebak-nebak apa fungsi sebuah tombol.

3. **Umpan Balik (Feedback):** Sistem harus selalu memberikan umpan balik kepada pengguna atas setiap tindakan yang mereka lakukan. Misalnya, setelah mengklik tombol "Simpan", sistem harus menampilkan pesan "Data berhasil disimpan" atau ikon pemuatan (*loading spinner*) jika prosesnya memakan waktu.
4. **Efisiensi (Efficiency):** Antarmuka harus dirancang untuk memungkinkan pengguna menyelesaikan tugas mereka dengan jumlah langkah atau klik sesedikit mungkin. Pikirkan tentang alur kerja pengguna dan optimalkan untuk tugas-tugas yang paling sering dilakukan.
5. **Fleksibilitas (Flexibility):** Sediakan cara bagi pengguna untuk menyesuaikan antarmuka atau menggunakan *shortcut* (jalan pintas). Pengguna ahli mungkin lebih suka menggunakan pintasan keyboard daripada mouse untuk mempercepat pekerjaan mereka.

Langkah-langkah Teknis dalam Desain UI

Proses desain UI bergerak dari abstraksi tingkat tinggi ke detail yang konkret. Proses ini biasanya melibatkan tiga artefak utama: *wireframe*, *mockup*, dan prototipe.

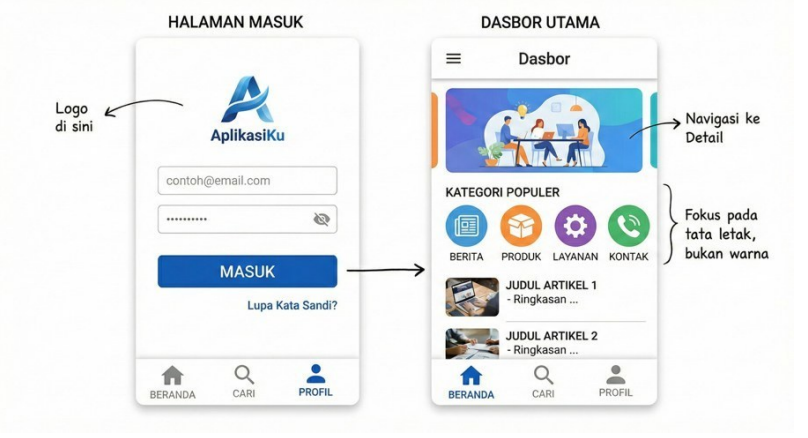
1. **Membuat *Wireframe***



- **Apa itu?** *Wireframe* adalah kerangka dasar atau "kerangka tulang" dari sebuah halaman atau layar. Ini adalah representasi visual dengan fidelitas rendah (*low-fidelity*) yang berfokus pada struktur, tata letak, dan penempatan elemen-elemen utama (seperti tombol, kolom input, dan area konten). *Wireframe* sengaja dibuat tanpa warna, font, atau gambar untuk menjaga fokus pada fungsionalitas dan alur, bukan pada estetika.
- **Tujuan:** Untuk menyepakati tata letak dasar dan alur navigasi dengan pemangku kepentingan sebelum menghabiskan waktu pada detail visual.
- **Cara Membuat:** Dapat dibuat dengan cepat menggunakan pena dan kertas, papan tulis, atau perangkat lunak khusus seperti Balsamiq atau Figma.

2. **Mengembangkan *Mockup***
   - **Apa itu?** *Mockup* adalah evolusi dari *wireframe*. Ini adalah representasi visual dengan fidelitas tinggi (*high-fidelity*) yang statis. *Mockup* menambahkan detail visual seperti skema warna, tipografi (jenis

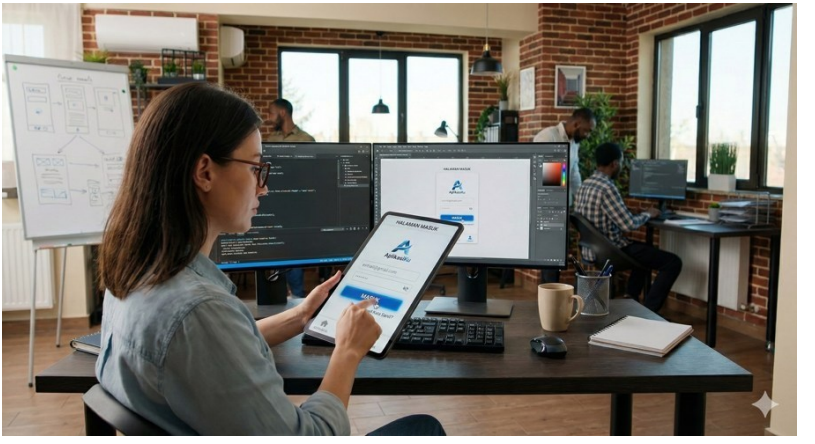
dan ukuran font), ikon, gambar, dan elemen branding. *Mockup* menunjukkan bagaimana tampilan produk akhir akan *terlihat*.



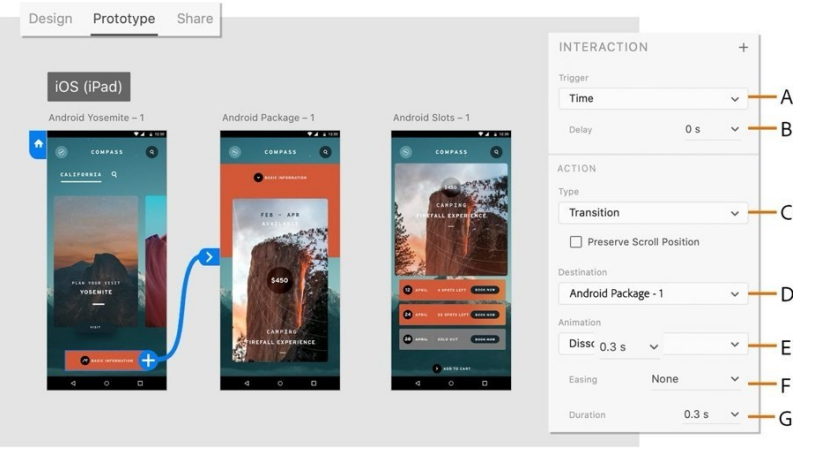
- **Tujuan:** Untuk mendapatkan umpan balik tentang aspek visual dan estetika dari antarmuka. Ini adalah representasi yang paling mendekati produk jadi tanpa harus menulis satu baris kode pun.
- **Cara Membuat:** Dibuat menggunakan perangkat lunak desain grafis seperti Figma, Sketch, atau Adobe XD.

3. **Membangun Prototipe (Prototype)**
   - **Apa itu?** Prototipe adalah *mockup* yang interaktif. Ia mensimulasikan pengalaman pengguna yang sebenarnya dengan memungkinkan pengguna untuk mengklik tombol, menavigasi antar layar, dan berinteraksi dengan elemen-elemen UI. Prototipe bisa berupa fidelitas rendah (menghubungkan *wireframe*) atau fidelitas tinggi (menghubungkan *mockup*).

Tujuan: Untuk melakukan **uji kegunaan (*usability testing*)**. Dengan memberikan prototipe kepada calon pengguna, desainer dapat mengamati bagaimana mereka berinteraksi dengan sistem, di mana mereka mengalami kesulitan, dan mendapatkan umpan balik yang sangat berharga sebelum pengembangan dimulai.



- **Cara Membuat:** Dibuat menggunakan perangkat lunak yang sama dengan *mockup* (seperti Figma atau Adobe XD) yang memiliki fitur *prototyping* untuk menghubungkan layar dan menambahkan interaksi.



Gambar 14. Prototype dengan Adobe XD.
Sumber: https://helpx.adobe.com/xd/help/create-prototypes.html

6.2 Desain Basis Data

Desain basis data adalah proses mengubah model data konseptual (ERD yang kita buat di Bab 5) menjadi sebuah struktur basis data yang logis dan fisik yang dapat diimplementasikan dalam sebuah *Database Management System* (DBMS). Jika UI adalah wajah sistem, maka basis data adalah otaknya, tempat semua informasi penting disimpan, diorganisir, dan dijaga integritasnya.

Dari ERD ke Skema Relasional

Langkah pertama dalam desain basis data adalah menerjemahkan ERD menjadi sekumpulan tabel dalam model relasional. Proses ini mengikuti serangkaian aturan yang jelas.

Langkah-langkah Teknis Konversi ERD ke Tabel:

1. **Setiap Entitas Menjadi Sebuah Tabel:** Untuk setiap entitas dalam ERD, buatlah sebuah tabel. Nama entitas menjadi nama tabel.
2. **Setiap Atribut Menjadi Kolom:** Setiap atribut dari entitas menjadi sebuah kolom (atau *field*) dalam tabel tersebut.
3. **Kunci Primer (Primary Key) Ditetapkan:** Atribut yang diidentifikasi sebagai kunci primer (PK) dalam ERD menjadi kolom kunci primer di dalam tabel. Kunci primer harus unik dan tidak boleh kosong (*not null*).
4. **Menerapkan Hubungan:**
    - **Hubungan Satu-ke-Banyak (1:M):** Untuk mengimplementasikan hubungan 1:M, ambil kunci primer dari tabel di sisi "satu" dan tambahkan sebagai kolom baru di tabel sisi "banyak". Kolom baru ini disebut **kunci asing (*foreign key* - FK)**. Kunci asing inilah yang menciptakan tautan logis antara kedua tabel.
        - *Contoh:* Dalam hubungan DOSEN (satu) mengajar MATAKULIAH (banyak), tabel MATAKULIAH akan memiliki kolom ID_Dosen sebagai kunci asing yang merujuk ke kunci primer di tabel DOSEN.
    - **Hubungan Banyak-ke-Banyak (M:N):** Hubungan M:N tidak dapat diimplementasikan secara langsung. Hubungan ini harus dipecah dengan membuat sebuah **tabel perantara (associative/junction table)**. Tabel baru ini akan memiliki setidaknya dua kolom, yang keduanya adalah kunci asing yang merujuk ke kunci primer dari dua tabel yang dihubungkannya. Kombinasi

kedua kunci asing ini biasanya menjadi kunci primer komposit untuk tabel perantara tersebut.

- *Contoh:* Untuk hubungan MAHASISWA (banyak) mengambil MATAKULIAH (banyak), kita membuat tabel baru bernama KRS. Tabel KRS akan memiliki kolom NIM (FK ke MAHASISWA) dan Kode_MK (FK ke MATAKULIAH).

Normalisasi Basis Data

Setelah struktur tabel awal terbentuk, langkah selanjutnya adalah **normalisasi**. Normalisasi adalah proses formal untuk meninjau skema tabel untuk meminimalkan redundansi data (pengulangan data yang tidak perlu) dan meningkatkan integritas data (konsistensi dan akurasi data). Proses ini melibatkan serangkaian aturan yang disebut **Bentuk Normal (Normal Forms)**. Untuk sebagian besar aplikasi bisnis, mencapai Bentuk Normal Ketiga (3NF) sudah dianggap cukup.

Langkah-langkah Teknis Normalisasi (disederhanakan):

1. **Bentuk Normal Pertama (1NF):** Pastikan tabel Anda memenuhi kriteria berikut:
    - Memiliki kunci primer.
    - Tidak ada grup berulang (misalnya, kolom Telepon1, Telepon2). Setiap data telepon harus berada di barisnya sendiri, mungkin di tabel terpisah.
    - Setiap sel di persimpangan baris dan kolom hanya berisi satu nilai (bersifat atomik).
2. **Bentuk Normal Kedua (2NF):**
    - Tabel harus sudah dalam 1NF.

- Semua atribut non-kunci harus bergantung secara fungsional pada **keseluruhan** kunci primer. Aturan ini terutama relevan untuk tabel yang memiliki kunci primer komposit (terdiri dari beberapa kolom). Jika sebuah atribut non-kunci hanya bergantung pada sebagian dari kunci komposit, maka atribut tersebut harus dipindahkan ke tabel baru.

3. **Bentuk Normal Ketiga (3NF):**
   - Tabel harus sudah dalam 2NF.
   - Tidak boleh ada **dependensi transitif**. Artinya, atribut non-kunci tidak boleh bergantung pada atribut non-kunci lainnya. Jika ada, atribut yang bergantung tersebut harus dipindahkan ke tabel baru.
     - *Contoh:* Jika tabel MAHASISWA memiliki kolom Kode_Prodi dan Nama_Prodi, ini melanggar 3NF karena Nama_Prodi bergantung pada Kode_Prodi (atribut non-kunci), bukan pada NIM (kunci primer). Solusinya adalah membuat tabel PRODI terpisah dengan Kode_Prodi sebagai PK dan Nama_Prodi sebagai atributnya.

6.3 Desain Arsitektur Sistem

Desain arsitektur sistem adalah tentang membuat keputusan tingkat tinggi mengenai struktur keseluruhan sistem. Ini mendefinisikan bagaimana komponen-komponen utama perangkat lunak (seperti antarmuka pengguna, logika bisnis, dan penyimpanan data) diorganisir dan bagaimana mereka

saling berkomunikasi. Pilihan arsitektur sangat dipengaruhi oleh kebutuhan non-fungsional seperti kinerja, skalabilitas, dan keamanan.

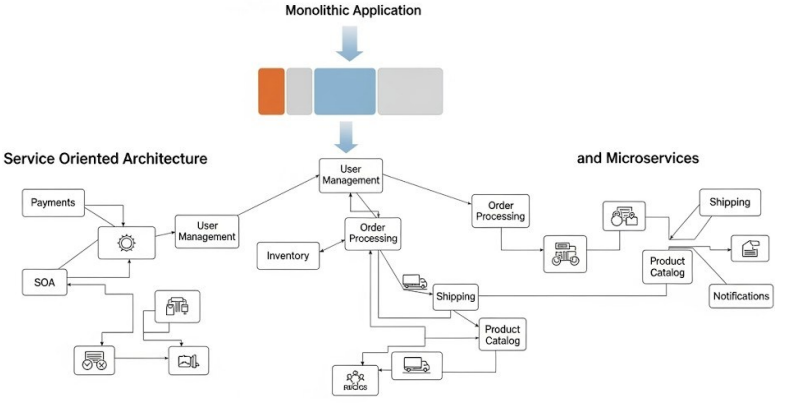
Pola Arsitektur Umum

Ada beberapa pola arsitektur yang telah terbukti efektif untuk berbagai jenis sistem.

1. **Arsitektur Dua Tingkat (Two-Tier / Client-Server):**
    - **Struktur:** Terdiri dari dua komponen utama: **klien** (yang bertanggung jawab atas presentasi/UI) dan **server** (yang bertanggung jawab atas logika bisnis dan akses data).
    - **Cara Kerja:** Klien mengirimkan permintaan ke server, dan server memproses permintaan tersebut (seringkali dengan mengakses basis data) lalu mengirimkan kembali responsnya.
    - **Contoh:** Aplikasi desktop sederhana yang terhubung langsung ke server basis data.
2. **Arsitektur Tiga Tingkat (Three-Tier):**
    - **Struktur:** Memisahkan sistem menjadi tiga lapisan logis yang berbeda:
        - **Tingkat Presentasi (Presentation Tier):** Antarmuka pengguna (UI) yang dilihat dan diinteraksikan oleh pengguna.
        - **Tingkat Aplikasi/Logika (Application/Logic Tier):** Otak dari sistem, tempat semua logika bisnis, aturan, dan pemrosesan data terjadi.

        - **Tingkat Data (Data Tier):** Bertanggung jawab untuk menyimpan dan mengambil data, biasanya terdiri dari server basis data.
    - **Keuntungan:** Pemisahan ini membuat sistem lebih fleksibel, skalabel, dan mudah dipelihara. Setiap tingkatan dapat dikembangkan dan dimodifikasi secara independen.
    - **Contoh:** Sebagian besar aplikasi web modern menggunakan arsitektur ini (Browser sebagai Presentation Tier, Web Server sebagai Application Tier, Database Server sebagai Data Tier).

3. **Arsitektur Berbasis Layanan (Service-Oriented Architecture - SOA) & Microservices:**
    - **Struktur:** Pendekatan modern di mana aplikasi dibangun sebagai kumpulan layanan (*services*) yang independen dan dapat digunakan kembali. Setiap layanan menyediakan fungsionalitas bisnis tertentu (misalnya, layanan "Pembayaran", layanan "Manajemen Pengguna").
    - ***Microservices*** adalah bentuk SOA yang lebih granular, di mana setiap layanan sangat kecil, independen, dan fokus pada satu tugas bisnis saja.
    - **Keuntungan:** Sangat skalabel, memungkinkan tim yang berbeda untuk bekerja pada layanan yang berbeda secara paralel, dan memungkinkan penggunaan teknologi yang berbeda untuk setiap layanan.
    - **Contoh:** Platform besar seperti Netflix atau Amazon dibangun menggunakan arsitektur microservices.



Gambar 15. Service Oriented Architecture & Microservices

6.4 Desain Proses Bisnis

Setelah arsitektur dan struktur data ditetapkan, langkah terakhir adalah merancang logika detail di dalam setiap proses. Jika DFD pada fase analisis menunjukkan *apa* yang dilakukan sebuah proses, desain proses merinci *bagaimana* proses tersebut melakukannya, langkah demi langkah, termasuk semua aturan, perhitungan, dan keputusan.

Alat untuk Desain Logika Proses

Untuk mendokumentasikan logika proses yang kompleks, analis menggunakan beberapa alat yang lebih formal daripada deskripsi naratif.

1. **Structured English / Pseudocode:**
    - **Apa itu?** Sebuah cara untuk mendeskripsikan logika proses menggunakan subset dari bahasa Inggris yang diperkuat dengan kata kunci dan struktur dari bahasa pemrograman (seperti IF...THEN...ELSE, DO

WHILE, CASE). *Pseudocode* tidak terikat pada sintaks bahasa pemrograman tertentu, sehingga mudah dipahami oleh analis dan cukup detail untuk dipahami oleh pemrogram.

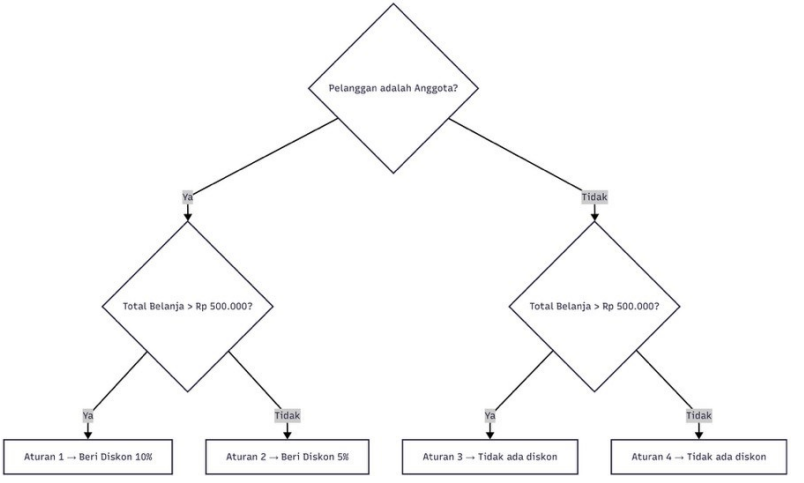
- **Contoh (untuk proses "Hitung Denda"):**

```
GET Tanggal_Kembali, Tanggal_Jatuh_Tempo
IF Tanggal_Kembali > Tanggal_Jatuh_Tempo THEN
    Hitung Jumlah_Hari_Terlambat = Tanggal_Kembali -
Tanggal_Jatuh_Tempo
    Hitung Denda = Jumlah_Hari_Terlambat
Tarif_Denda_Per_Hari
ELSE
    Set Denda = 0
ENDIF
RETURN Denda
```

2. **Tabel Keputusan (Decision Table):**
   - **Apa itu?** Sebuah matriks dalam bentuk tabel yang merepresentasikan logika kondisional yang kompleks. Sangat berguna ketika sebuah proses memiliki banyak kondisi yang kombinasinya menghasilkan tindakan yang berbeda.
   - **Struktur:** Terdiri dari empat bagian: Kondisi (*Conditions*), Aturan (*Rules*), dan Tindakan (*Actions*).
   - **Contoh (untuk menentukan diskon):**

Kondisi	Aturan 1	Aturan 2	Aturan 3	Aturan 4
Pelanggan adalah Anggota?	Y	Y	N	N
Total Belanja > Rp 500.000?	Y	N	Y	N
Tindakan				
Beri Diskon 10%	X			
Beri Diskon 5%		X		
Tidak ada diskon			X	X

3. **Pohon Keputusan (Decision Tree):**
    - **Apa itu?** Representasi grafis dari logika keputusan yang sama dengan tabel keputusan, tetapi dalam bentuk struktur pohon.
    - **Struktur:** Dimulai dari satu simpul (*node*) akar, yang kemudian bercabang berdasarkan kondisi. Setiap cabang merepresentasikan sebuah aturan, dan daun (*leaf*) di ujung cabang menunjukkan tindakan yang harus diambil. Banyak orang merasa pohon keputusan lebih intuitif dan mudah dibaca daripada tabel keputusan.



Gambar 16. Pohon keputusan penetapan diskon

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Jelaskan hubungan antara ERD, Desain Basis Data, dan Normalisasi. Mengapa seorang desainer tidak bisa hanya mengubah ERD langsung menjadi tabel tanpa memikirkan normalisasi?
2. Sebuah *startup* sedang membangun aplikasi seluler baru. Mereka memprediksi pertumbuhan pengguna yang sangat cepat. Arsitektur sistem mana (Two-Tier atau Three-Tier) yang lebih Anda rekomendasikan, dan mengapa?
3. Kapan Anda akan memilih untuk menggunakan Tabel Keputusan daripada *Pseudocode* untuk mendesain logika sebuah proses? Berikan contoh proses bisnis yang cocok untuk Tabel Keputusan.

Solusi dan Pembahasan Soal Konseptual

1. **Hubungan ERD, Desain Basis Data, dan Normalisasi:**
    - **Hubungan:** ERD adalah model konseptual tingkat tinggi yang menjadi **input** untuk Desain Basis Data. Proses Desain Basis Data **menerjemahkan** ERD menjadi skema tabel logis. Normalisasi adalah proses **penyempurnaan** dalam Desain Basis Data yang memastikan skema tabel tersebut efisien dan bebas dari anomali data.
    - **Mengapa Normalisasi Penting:** Mengubah ERD langsung menjadi tabel tanpa normalisasi dapat menyebabkan masalah serius. ERD

mungkin secara tidak sengaja mengandung redundansi (misalnya, menyimpan nama dan alamat pemasok di setiap baris transaksi pembelian, bukan di tabel Pemasok terpisah). Tanpa normalisasi, data yang berulang ini akan membuang ruang penyimpanan dan, yang lebih penting, menciptakan **anomali pembaruan, penyisipan, dan penghapusan**. Misalnya, jika alamat pemasok berubah, Anda harus memperbaruinya di ratusan baris transaksi, dan jika Anda melewatkan satu saja, integritas data akan rusak. Normalisasi memastikan setiap bagian data disimpan hanya di satu tempat.

2. Rekomendasi Arsitektur untuk Startup:

   Saya akan merekomendasikan Arsitektur Tiga Tingkat (Three-Tier).

   - **Alasan:** Alasan utamanya adalah **skalabilitas**. *Startup* tersebut memprediksi pertumbuhan cepat. Dalam arsitektur tiga tingkat, setiap lapisan (presentasi, aplikasi, data) bersifat independen. Jika server aplikasi menjadi *bottleneck* karena lonjakan pengguna, mereka dapat menambahkan lebih banyak server aplikasi (*scaling out*) tanpa harus menyentuh server basis data atau antarmuka pengguna. Sebaliknya, pada arsitektur dua tingkat, logika bisnis seringkali menyatu dengan basis data atau klien, membuatnya jauh lebih sulit untuk diskalakan secara efisien. Fleksibilitas dan

kemudahan pemeliharaan dari arsitektur tiga tingkat juga merupakan keuntungan besar bagi *startup* yang perlu beradaptasi dengan cepat.

3. **Tabel Keputusan vs. Pseudocode:**
    - **Kapan Menggunakan Tabel Keputusan:** Anda akan memilih Tabel Keputusan ketika logika sebuah proses sangat bergantung pada **kombinasi dari beberapa kondisi independen**. Jika ada banyak cabang IF...ELSE IF...ELSE yang bersarang, *pseudocode* bisa menjadi sangat rumit dan sulit dibaca. Tabel Keputusan menyajikan semua kemungkinan kombinasi dan tindakan yang dihasilkan secara ringkas dan terstruktur, sehingga lebih mudah untuk memverifikasi bahwa semua kasus telah ditangani.
    - **Contoh Proses:** Proses yang cocok adalah **penentuan premi asuransi mobil**. Kondisinya bisa meliputi: Usia Pengemudi (<25, 25-60, >60), Jenis Mobil (Sport, Sedan, SUV), Riwayat Kecelakaan (Ada, Tidak Ada), dan Lokasi (Kota Besar, Kota Kecil). Kombinasi dari kondisi-kondisi ini akan menentukan tarif premi yang berbeda (tindakan), yang sangat ideal untuk direpresentasikan dalam sebuah Tabel Keputusan.

Studi Kasus: Desain Sistem Peminjaman Buku Perpustakaan Online

- **Konteks:** Sebuah perpustakaan universitas ingin membuat sistem online di mana mahasiswa dapat mencari buku dan melihat status peminjaman mereka.

- **Hasil Analisis (dari Bab 5):**
    - **ERD:** Terdapat entitas MAHASISWA (NIM, Nama, Prodi), BUKU (ISBN, Judul, Penulis), dan PEMINJAMAN (ID_Peminjaman, Tgl_Pinjam, Tgl_Jatuh_Tempo). Hubungannya adalah MAHASISWA (1) ---< PEMINJAMAN (0..*) dan BUKU (1) ---< PEMINJAMAN (0..*).
    - **Proses Kunci:** Salah satu proses kunci adalah "Validasi Peminjaman", yang harus memeriksa apakah mahasiswa tersebut masih memiliki kuota peminjaman yang tersedia (maksimal 3 buku) dan tidak memiliki denda yang belum dibayar.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan:

1. Berdasarkan ERD yang diberikan, buatlah **desain basis data logis** (skema tabel) yang sesuai. Tentukan kunci primer (PK) dan kunci asing (FK) untuk setiap tabel.
2. Buatlah **desain proses bisnis** untuk proses "Validasi Peminjaman" menggunakan **Pohon Keputusan (Decision Tree)**.

Analisis:

1. Desain Basis Data Logis

Berdasarkan ERD, kita akan membuat tiga tabel. Kunci asing akan ditambahkan ke tabel PEMINJAMAN untuk mengimplementasikan hubungan 1:M.

- **Tabel: MAHASISWA**

Nama Kolom	Tipe Data	Keterangan
NIM	VARCHAR(10)	**Primary Key**
Nama_Mahasiswa	VARCHAR(100)	

Program_Studi	VARCHAR(50)	

- **Tabel: BUKU**

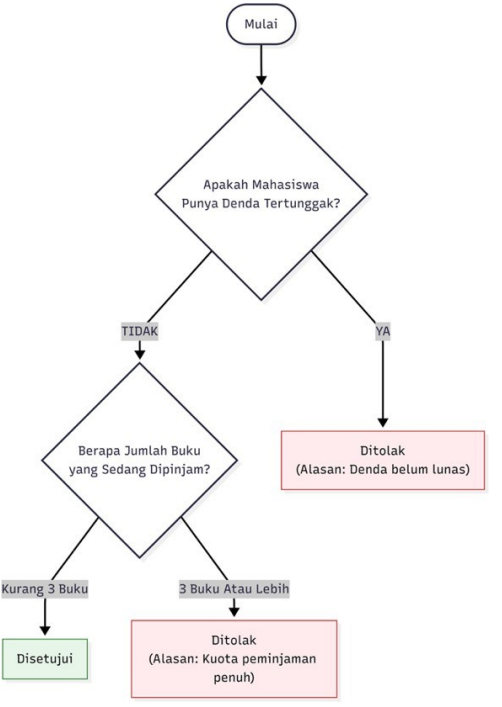
Nama Kolom	Tipe Data	Keterangan
ISBN	VARCHAR(13)	**Primary Key**
Judul_Buku	VARCHAR(255)	
Penulis	VARCHAR(100)	

- **Tabel: PEMINJAMAN**

Nama Kolom	Tipe Data	Keterangan
ID_Peminjaman	INT	**Primary Key**
Tgl_Pinjam	DATE	
Tgl_Jatuh_Tempo	DATE	
Tgl_Kembali	DATE	(Bisa NULL jika belum dikembalikan)
NIM	VARCHAR(10)	**Foreign Key** ke MAHASISWA(NIM)
ISBN	VARCHAR(13)	**Foreign Key** ke BUKU(ISBN)

2. Desain Proses "Validasi Peminjaman" dengan Pohon Keputusan

Pohon keputusan ini akan memvisualisasikan logika untuk menyetujui atau menolak permintaan peminjaman buku.



Gambar 17. Pohon Keputusan Validasi Peminjaman

Pohon keputusan ini secara jelas dan visual menunjukkan dua kondisi yang harus diperiksa secara berurutan (status denda dan jumlah buku yang dipinjam) dan tiga kemungkinan hasil (Tolak karena denda, Tolak karena kuota penuh, atau Setujui). Ini jauh lebih mudah dipahami daripada blok kode IF-ELSE yang bersarang.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.

- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Valacich, J. S., & George, J. F. (Edisi terbaru). *Modern Systems Analysis and Design*. Pearson.

Bacaan Tambahan (Dianjurkan)

- Krug, S. (Edisi terbaru). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. New Riders. (Bacaan wajib untuk siapa pun yang tertarik dengan desain antarmuka pengguna yang efektif).
- Connolly, T., & Begg, C. (Edisi terbaru). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson. (Referensi komprehensif untuk desain dan implementasi basis data).
- Bass, L., Clements, P., & Kazman, R. (Edisi terbaru). *Software Architecture in Practice*. Addison-Wesley Professional. (Memberikan wawasan mendalam tentang berbagai pola dan praktik arsitektur perangkat lunak).

Bab 7: Implementasi Sistem

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menganalisis dan memilih bahasa pemrograman, kerangka kerja (*framework*), dan teknologi pendukung yang paling sesuai untuk sebuah proyek sistem informasi berdasarkan kebutuhan fungsional dan non-fungsional.
- Membedakan antara prototipe *throwaway* dan prototipe evolusioner, serta menjelaskan peran strategis prototipe dalam fase implementasi.
- Menjelaskan dan membandingkan berbagai strategi integrasi modul, seperti *Big Bang*, *Top-Down*, dan *Bottom-Up*, serta memahami konsep *Continuous Integration* (CI).
- Menerapkan langkah-langkah teknis untuk mengimplementasikan basis data, termasuk memilih DBMS yang tepat dan menerjemahkan desain basis data logis ke dalam perintah *Data Definition Language* (DDL) SQL.
- Menulis perintah SQL DDL dasar (CREATE TABLE) untuk membuat struktur tabel, lengkap dengan pendefinisian kolom, tipe data, kunci primer, dan kunci asing.

Selamat datang di Bab 7. Sejauh ini, perjalanan kita telah membawa kita dari konsep abstrak, analisis mendalam, hingga perancangan yang detail. Kita telah memiliki cetak biru yang solid dari Bab 6. Sekarang, tiba saatnya untuk memasuki fase yang paling dinanti oleh banyak orang: **implementasi**. Fase ini adalah momen di mana semua diagram, model, dan spesifikasi diubah menjadi sesuatu yang nyata dan dapat dieksekusi, yaitu, perangkat lunak yang berfungsi. Ini adalah tahap konstruksi, di

mana para pemrogram (*programmer*) dan insinyur perangkat lunak (*software engineer*) mengambil alih peran utama, menerjemahkan desain logis menjadi baris-baris kode.

Namun, implementasi lebih dari sekedar pengkodean. Ini adalah fase yang penuh dengan keputusan teknis krusial yang akan menentukan kinerja, keandalan, dan kemudahan pemeliharaan sistem di masa depan. Dalam bab ini, kita akan membahas keputusan strategis pertama: **pemilihan bahasa pemrograman dan teknologi** yang tepat. Kita akan melihat bagaimana **pengembangan prototipe** dapat digunakan untuk memvalidasi ide dan mengurangi risiko. Kita juga akan mempelajari bagaimana modul-modul perangkat lunak yang dikembangkan secara terpisah dapat **diintegrasikan** menjadi satu kesatuan yang utuh. Terakhir, kita akan masuk ke ranah teknis **implementasi basis data**, mengubah skema relasional kita menjadi basis data fisik yang siap menampung informasi. Fase implementasi adalah tempat di mana teori bertemu dengan praktik, dan di mana visi mulai terbentuk menjadi kenyataan.

7.1 Pemilihan Bahasa Pemrograman dan Teknologi

Sebelum satu baris kode pun ditulis, tim pengembang harus membuat salah satu keputusan teknis yang paling fundamental: teknologi apa yang akan digunakan untuk membangun sistem? Keputusan ini, yang sering disebut sebagai pemilihan "tumpukan teknologi" (*technology stack*), akan berdampak pada seluruh siklus hidup proyek, mulai dari kecepatan pengembangan, kinerja sistem, hingga ketersediaan tenaga ahli untuk pemeliharaan di masa depan. Pemilihan ini bukanlah sekedar preferensi pribadi para pemrogram, melainkan sebuah keputusan rekayasa yang harus didasarkan pada analisis cermat terhadap berbagai faktor.

Faktor-faktor dalam Pemilihan Teknologi

1. **Kebutuhan Proyek:** Ini adalah faktor yang paling penting. Kebutuhan fungsional dan non-fungsional yang telah kita definisikan di Bab 5 menjadi panduan utama.
    - *Contoh:* Sebuah sistem analitik *big data* mungkin lebih cocok dibangun dengan Python karena ekosistemnya yang kaya akan pustaka (*library*) analisis data. Sementara itu, sebuah aplikasi web dengan interaktivitas tinggi mungkin lebih diuntungkan dengan menggunakan JavaScript dan kerangka kerja (*framework*) seperti React atau Vue.js.
2. **Platform Target:** Di mana sistem akan dijalankan?
    - **Aplikasi Web:** Teknologi umum meliputi HTML, CSS, JavaScript di sisi klien (*front-end*), dan bahasa seperti Node.js, Python (Django/Flask), PHP, atau Java di sisi server (*back-end*).
    - **Aplikasi Seluler (Mobile):** Pilihan utamanya adalah pengembangan *native* (Swift untuk iOS, Kotlin/Java untuk Android) atau pengembangan lintas platform (*cross-platform*) menggunakan kerangka kerja seperti React Native atau Flutter.
    - **Aplikasi Desktop:** Bahasa seperti C#, Java, atau Python dengan pustaka GUI (Graphical User Interface) dapat digunakan.
3. **Skalabilitas dan Kinerja:** Seberapa besar pertumbuhan pengguna dan data yang diantisipasi? Kebutuhan non-fungsional terkait kinerja (misalnya, waktu respons di bawah 1 detik) sangat memengaruhi pilihan. Teknologi seperti Node.js dikenal karena kemampuannya menangani banyak koneksi secara bersamaan, membuatnya populer untuk aplikasi *real-time*.
4. **Ekosistem dan Dukungan Komunitas:** Sebuah bahasa pemrograman lebih dari sekedar sintaksnya. Ekosistemnya, termasuk ketersediaan pustaka, kerangka kerja, dan alat bantu, sangat penting. Bahasa dengan komunitas yang besar dan aktif (seperti Python, JavaScript, Java) cenderung

memiliki lebih banyak sumber daya, tutorial, dan solusi untuk masalah umum, yang dapat mempercepat pengembangan secara signifikan.

5. **Keahlian Tim Pengembang:** Secara praktis, keahlian yang sudah dimiliki oleh tim adalah faktor besar. Memilih teknologi yang benar-benar baru bagi seluruh tim akan memperkenalkan kurva belajar yang curam dan dapat memperlambat proyek. Namun, ini harus diimbangi dengan kesesuaian teknologi untuk proyek tersebut; memaksakan teknologi yang familiar tetapi tidak cocok dapat menyebabkan masalah di kemudian hari.
6. **Biaya dan Lisensi:** Apakah teknologi tersebut bersifat *open-source* (gratis) atau komersial (berbayar)? Sebagian besar pengembangan modern mengandalkan teknologi *open-source* untuk menekan biaya, tetapi beberapa solusi khusus mungkin memerlukan perangkat lunak berlisensi.

7.2 Pengembangan Prototipe

Meskipun prototipe sering dibahas dalam fase desain untuk memvalidasi konsep, ia juga memainkan peran penting selama fase implementasi, terutama dalam metodologi iteratif seperti Agile atau Spiral. Prototipe dalam tahap ini bukan lagi sekedar gambar statis, melainkan versi awal dari perangkat lunak yang berfungsi, meskipun dengan fungsionalitas yang terbatas.

Tujuan utama pengembangan prototipe pada tahap ini adalah untuk mendapatkan umpan balik nyata dari pengguna secepat mungkin dan untuk menguji asumsi teknis. Ada dua pendekatan utama dalam pengembangan prototipe:

1. **Prototipe *Throwaway* (Cepat dan Kotor):**
    - **Konsep:** Prototipe ini dibangun dengan cepat untuk mendemonstrasikan atau menguji satu ide spesifik (misalnya, alur kerja baru atau konsep antarmuka). Penekanannya adalah pada kecepatan, bukan pada kualitas kode.

- **Tujuan:** Untuk belajar. Setelah umpan balik didapatkan dan konsep divalidasi, prototipe ini **dibuang**. Kode dari prototipe ini tidak digunakan dalam produk akhir.
- **Kapan Digunakan:** Sangat berguna untuk mengeksplorasi kebutuhan yang sangat tidak jelas atau untuk menguji kelayakan teknis dari sebuah pendekatan yang berisiko tinggi.

2. **Prototipe Evolusioner:**
   - **Konsep:** Pendekatan ini dimulai dengan membangun fondasi sistem yang solid tetapi dengan fitur yang sangat terbatas. Prototipe awal ini kemudian secara bertahap **berevolusi**, fitur-fitur baru ditambahkan, dan fungsionalitas yang ada disempurnakan dalam setiap iterasi berdasarkan umpan balik pengguna.
   - **Tujuan:** Untuk membangun produk secara inkremental. Prototipe awal menjadi inti dari produk akhir.
   - **Kapan Digunakan:** Ini adalah pendekatan inti dari banyak metodologi Agile. Setiap *sprint* dalam Scrum, misalnya, menghasilkan prototipe evolusioner yang berfungsi (disebut *Increment*) yang selangkah lebih dekat ke produk akhir.

Pengembangan prototipe membantu menjembatani kesenjangan komunikasi antara pengguna dan pengembang. Dengan memberikan sesuatu yang dapat dicoba secara langsung, pengguna dapat memberikan umpan balik yang jauh lebih konkret daripada hanya meninjau dokumen spesifikasi.

7.3 Integrasi Modul

Sistem perangkat lunak modern jarang dibangun sebagai satu program monolitik yang besar. Sebaliknya, mereka dibangun menggunakan pendekatan modular, di mana sistem dipecah

menjadi komponen-komponen atau **modul-modul** yang lebih kecil dan dapat dikelola. Setiap modul bertanggung jawab atas satu set fungsionalitas yang terkait erat (misalnya, modul "Manajemen Pengguna", modul "Proses Pembayaran").

Setelah modul-modul ini dikembangkan (seringkali secara paralel oleh pengembang yang berbeda), mereka harus digabungkan menjadi satu sistem yang berfungsi. Proses ini disebut **integrasi modul**. Pengujian integrasi adalah aktivitas kunci untuk memverifikasi bahwa modul-modul tersebut dapat berkomunikasi dan bekerja sama dengan benar.

Strategi Integrasi Modul

Ada beberapa strategi untuk melakukan integrasi, masing-masing dengan kelebihan dan kekurangannya.

1. **Integrasi *Big Bang*:**
    - **Konsep:** Semua modul dikembangkan dan diuji secara terpisah. Setelah semua modul selesai, mereka diintegrasikan sekaligus untuk membentuk sistem yang lengkap.
    - **Kelebihan:** Sederhana untuk direncanakan.
    - **Kekurangan:** Sangat berisiko tinggi. Jika terjadi kesalahan setelah integrasi, sangat sulit untuk melacak sumber masalahnya karena ada begitu banyak titik interaksi yang harus diperiksa.
2. **Integrasi *Top-Down*:**
    - **Konsep:** Integrasi dimulai dari modul tingkat atas (misalnya, antarmuka pengguna utama). Modul-modul tingkat bawah yang dipanggil olehnya pada awalnya digantikan oleh **stubs**, program tiruan sederhana yang mensimulasikan fungsionalitas dasar. Secara bertahap, *stubs* diganti dengan modul yang sebenarnya.

- **Kelebihan:** Memungkinkan pengujian alur kerja dan logika kontrol utama sejak dini.
- **Kekurangan:** Fungsionalitas tingkat rendah yang krusial baru diuji di akhir proses.

3. **Integrasi *Bottom-Up*:**
   - **Konsep:** Integrasi dimulai dari modul tingkat terendah (misalnya, modul yang berinteraksi langsung dengan basis data). Modul-modul ini diuji menggunakan **drivers**, program kecil yang memanggil modul tersebut dan memberikan data uji. Modul-modul yang telah teruji kemudian digabungkan ke tingkat yang lebih tinggi.
   - **Kelebihan:** Modul-modul fundamental diuji secara menyeluruh sejak awal.
   - **Kekurangan:** Prototipe sistem secara keseluruhan tidak terlihat sampai akhir proses.
4. **Integrasi Berkelanjutan (*Continuous Integration* - CI):**
   - **Konsep:** Ini adalah praktik modern yang dominan, terutama dalam pengembangan Agile. Setiap kali seorang pengembang menyelesaikan perubahan kecil pada kode, perubahan tersebut secara otomatis diintegrasikan ke dalam basis kode utama. Setelah integrasi, sistem secara otomatis dibangun dan diuji.
   - **Kelebihan:** Masalah integrasi ditemukan dengan sangat cepat (dalam hitungan menit atau jam, bukan minggu atau bulan), membuat perbaikannya jauh lebih mudah dan lebih murah. Ini mendorong kolaborasi tim yang lebih baik dan kualitas kode yang lebih tinggi.

7.4 Implementasi Basis Data

Implementasi basis data adalah proses teknis untuk menciptakan basis data fisik berdasarkan desain yang telah dibuat di Bab 6.[1] Ini melibatkan pemilihan

software basis data dan penulisan kode untuk mendefinisikan strukturnya.

Langkah-langkah Teknis Implementasi Basis Data

1. **Memilih Sistem Manajemen Basis Data (DBMS):**
    - Berdasarkan desain dan kebutuhan non-fungsional, tim harus memilih DBMS yang tepat.
    - **Basis Data Relasional (SQL):** Pilihan yang paling umum untuk data terstruktur. Contoh: MySQL (populer untuk aplikasi web, *open-source*), PostgreSQL (*open-source*, dikenal karena kepatuhan standar dan fitur canggih), Microsoft SQL Server (komersial, terintegrasi baik dengan ekosistem Microsoft), Oracle Database (komersial, kuat untuk aplikasi perusahaan skala besar).
    - **Basis Data NoSQL:** Pilihan yang baik untuk data tidak terstruktur atau semi-terstruktur, atau ketika skalabilitas horizontal yang masif menjadi prioritas. Contoh: MongoDB (berbasis dokumen), Redis (penyimpanan *key-value*).
2. **Menerjemahkan Desain ke *Data Definition Language* (DDL):**
    - Struktur basis data (tabel, kolom, tipe data, relasi) diimplementasikan menggunakan sub-bahasa SQL yang disebut DDL. Perintah utamanya adalah CREATE.
    - **Langkah Teknis:** Untuk setiap tabel dalam desain basis data logis Anda, tulis pernyataan CREATE TABLE.
    - **Contoh:** Mari kita implementasikan tabel PEMINJAMAN dari studi kasus Bab 6.

```
SQL
CREATE TABLE PEMINJAMAN (
    ID_Peminjaman   INT             PRIMARY KEY,
    Tgl_Pinjam      DATE            NOT NULL,
    Tgl_Jatuh_Tempo DATE            NOT NULL,
    Tgl_Kembali     DATE,
```

```
    NIM             VARCHAR(10)     NOT NULL,
    ISBN            VARCHAR(13)     NOT NULL,
    FOREIGN KEY (NIM) REFERENCES MAHASISWA(NIM),
    FOREIGN KEY (ISBN) REFERENCES BUKU(ISBN)
);
```

- **Penjelasan Kode:**
  - CREATE TABLE PEMINJAMAN (...): Membuat tabel baru bernama PEMINJAMAN.
  - ID_Peminjaman INT PRIMARY KEY: Mendefinisikan kolom ID_Peminjaman dengan tipe data integer dan menjadikannya sebagai kunci primer.
  - Tgl_Pinjam DATE NOT NULL: Mendefinisikan kolom tanggal yang tidak boleh kosong.
  - FOREIGN KEY (NIM) REFERENCES MAHASISWA(NIM): Mendefinisikan NIM sebagai kunci asing yang terhubung ke kolom NIM di tabel MAHASISWA. Ini memastikan integritas referensial, Anda tidak dapat memasukkan data peminjaman untuk mahasiswa yang tidak ada di tabel MAHASISWA.

3. **Memuat Data Awal (Data Loading):**
   - Setelah struktur basis data dibuat, seringkali perlu diisi dengan data awal. Ini bisa berupa data master (seperti daftar produk atau daftar program studi) atau data yang dimigrasi dari sistem lama.
   - Proses ini dapat dilakukan dengan menulis skrip SQL menggunakan perintah INSERT atau menggunakan alat bantu *Extract, Transform, Load* (ETL) untuk migrasi data yang lebih kompleks.

Setelah basis data diimplementasikan, ia siap untuk dihubungkan dengan kode aplikasi yang akan melakukan operasi *Create, Read, Update, dan Delete* (CRUD) menggunakan perintah *Data Manipulation Language* (DML) dari SQL.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Sebuah tim sedang membangun sistem perbankan inti yang membutuhkan tingkat keamanan dan konsistensi transaksi yang sangat tinggi. Pilihan teknologi mereka adalah antara Python dengan *framework* Django atau Java dengan *framework* Spring. Teknologi mana yang cenderung lebih umum dipilih untuk kasus seperti ini, dan mengapa?
2. Jelaskan perbedaan utama dalam risiko antara strategi integrasi *Big Bang* dan *Continuous Integration*.
3. Apa tujuan dari klausa FOREIGN KEY dalam pernyataan CREATE TABLE di SQL? Apa yang akan terjadi jika Anda mencoba memasukkan baris data ke dalam tabel PEMINJAMAN dengan nilai NIM yang tidak ada di tabel MAHASISWA?

Solusi dan Pembahasan Soal Konseptual

1. Pemilihan Teknologi untuk Sistem Perbankan:

   Java dengan framework Spring cenderung menjadi pilihan yang lebih umum untuk sistem perbankan inti.

   - **Alasan:** Meskipun Python sangat mumpuni, ekosistem Java secara historis sangat kuat di dunia perusahaan (*enterprise*), terutama di sektor keuangan. Java dikenal karena kinerjanya yang tinggi, *strong typing* (yang membantu mengurangi *bug*), dan sifatnya yang *multi-threaded*, yang sangat penting untuk memproses ribuan transaksi secara bersamaan. *Framework* seperti Spring menyediakan fitur keamanan, manajemen transaksi, dan skalabilitas yang

sudah teruji dan matang untuk aplikasi skala besar dan kritis seperti perbankan.

2. **Perbedaan Risiko Integrasi:**
    - **Integrasi *Big Bang*:** Risikonya sangat tinggi dan terkonsentrasi di akhir fase pengembangan. Ketika semua modul digabungkan sekaligus dan terjadi kesalahan, sangat sulit dan memakan waktu untuk menemukan modul mana atau interaksi mana yang menjadi penyebabnya. Ini sering disebut "neraka integrasi" (*integration hell*).
    - ***Continuous Integration* (CI):** Risikonya didistribusikan dan dikelola dalam siklus yang sangat kecil dan sering. Karena kode diintegrasikan dan diuji secara otomatis setiap kali ada perubahan kecil, kesalahan integrasi dapat dideteksi hampir seketika. Ini membuat masalah lebih mudah diisolasi dan diperbaiki, sehingga risiko kegagalan besar di akhir proyek diminimalkan secara drastis.
3. **Tujuan Kunci Asing (Foreign Key):**
    - **Tujuan:** Tujuan utama dari FOREIGN KEY adalah untuk menegakkan **integritas referensial** antara dua tabel. Ini memastikan bahwa sebuah nilai di kolom kunci asing harus cocok dengan nilai yang ada di kolom kunci primer tabel yang dirujuknya. Secara bisnis, ini berarti Anda tidak bisa membuat catatan "anak" tanpa adanya catatan "induk" yang valid.
    - **Apa yang Terjadi:** Jika Anda mencoba INSERT sebuah baris ke dalam tabel PEMINJAMAN dengan NIM '12345' sementara NIM '12345' tidak ada di tabel MAHASISWA, DBMS akan **menolak** operasi tersebut dan mengembalikan pesan kesalahan.

Ini mencegah terciptanya data "yatim" atau data yang tidak konsisten di dalam basis data.

Studi Kasus: Implementasi Sistem POS "Kopi Kita"

- **Konteks:** Tim pengembang "Kopi Kita" siap untuk memulai implementasi sistem POS berbasis tablet yang telah dirancang. Tim terdiri dari pengembang dengan berbagai latar belakang, beberapa kuat di pengembangan web dan beberapa di pengembangan seluler. Mereka menggunakan metodologi Agile (Scrum).
- **Keputusan Desain:**
    - Aplikasi kasir akan berjalan di tablet Android.
    - Basis data akan terpusat di *cloud* agar manajer dapat mengakses laporan dari mana saja.
    - Desain basis data logis telah dibuat (seperti pada studi kasus Bab 6), dengan tabel PRODUK, PESANAN, DETAIL_PESANAN, dan KASIR.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan:

1. Sebagai pemimpin teknis, Anda harus memilih teknologi untuk aplikasi kasir di tablet Android. Pilihan Anda adalah antara (a) Pengembangan *Native* dengan Kotlin atau (b) Pengembangan *Cross-Platform* dengan React Native. Pilihan mana yang akan Anda rekomendasikan dan mengapa, mengingat konteks tim dan proyek?
2. Tuliskan pernyataan SQL DDL (CREATE TABLE) lengkap untuk tabel PRODUK dan KASIR berdasarkan desain dari Bab 6. Asumsikan tipe data yang sesuai.

Analisis:

1. Rekomendasi Teknologi Aplikasi Seluler:

Mengingat konteksnya, React Native adalah pilihan yang lebih direkomendasikan.

- **Justifikasi:**
  - **Keahlian Tim:** Tim memiliki anggota yang kuat dalam pengembangan web. React Native menggunakan JavaScript dan konsep dari React, yang sangat familiar bagi pengembang web. Ini akan secara signifikan mengurangi kurva belajar dibandingkan jika semua orang harus belajar Kotlin dari awal.
  - **Kecepatan Pengembangan:** Dengan keahlian yang sudah ada, tim dapat membangun dan melakukan iterasi pada prototipe fungsional lebih cepat menggunakan React Native, yang sangat sejalan dengan metodologi Agile/Scrum yang mereka gunakan.
  - **Fleksibilitas Masa Depan:** Meskipun target saat ini adalah tablet Android, jika di masa depan "Kopi Kita" ingin membuat aplikasi untuk pelanggan di iOS, sebagian besar basis kode React Native dapat digunakan kembali, menghemat waktu dan biaya.
  - **Kesesuaian:** Untuk aplikasi POS, yang sebagian besar adalah antarmuka berbasis formulir dan interaksi dengan API, kinerja React Native lebih dari cukup dan tidak memerlukan optimasi tingkat rendah yang mungkin ditawarkan oleh pengembangan *native*.

2. **Pernyataan SQL DDL (**CREATE TABLE**)**

- **Tabel: PRODUK**

```
SQL
CREATE TABLE PRODUK (
    ID_Produk       VARCHAR(10)     PRIMARY KEY,
    Nama_Produk     VARCHAR(100)    NOT NULL,
    Harga           DECIMAL(10, 2)  NOT NULL,
    Kategori        VARCHAR(50)
);
```

Catatan: DECIMAL(10, 2) dipilih untuk Harga karena merupakan tipe data yang tepat untuk menyimpan nilai moneter dengan presisi.

- **Tabel: KASIR**

```
SQL
CREATE TABLE KASIR (
    ID_Kasir        VARCHAR(10)     PRIMARY KEY,
    Nama_Kasir      VARCHAR(100)    NOT NULL,
    Peran           VARCHAR(20)     DEFAULT 'Kasir'
);
```

Catatan: Kolom Peran ditambahkan untuk mengantisipasi kebutuhan keamanan di masa depan (misalnya, 'Kasir' vs. 'Manajer'), dengan nilai default 'Kasir'.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Connolly, T., & Begg, C. (Edisi terbaru). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.

Bacaan Tambahan (Dianjurkan)

- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

(Memberikan wawasan mendalam tentang berbagai pola arsitektur sistem untuk aplikasi skala besar).

- Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall. (Meskipun berfokus pada pengkodean, buku ini memberikan prinsip-prinsip yang sangat baik tentang cara menulis perangkat lunak yang mudah dipelihara dan diintegrasikan).
- Dokumentasi resmi dari bahasa pemrograman atau kerangka kerja yang populer (misalnya, dokumentasi Java, Python, React, atau Android Developer). Sumber-sumber ini adalah referensi teknis paling akurat untuk implementasi.

Bab 8: Pengujian Sistem

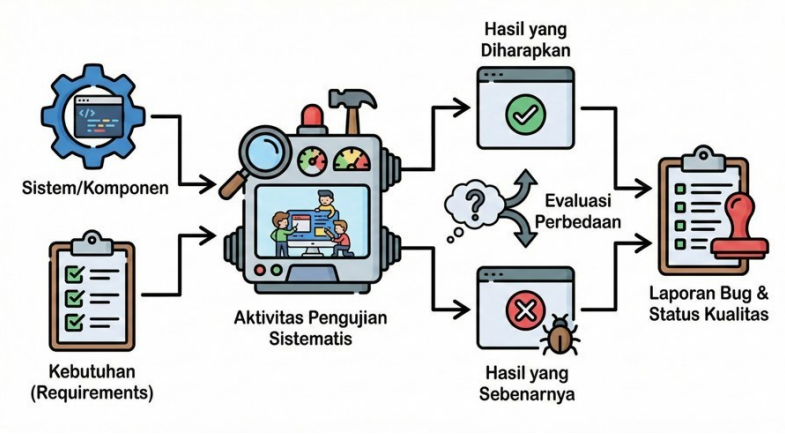
Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan tujuan fundamental dari pengujian perangkat lunak sebagai proses untuk menemukan kesalahan, bukan untuk membuktikan ketiadaan kesalahan.
- Membedakan konsep Verifikasi ("Apakah kita membangun sistem dengan benar?") dan Validasi ("Apakah kita membangun sistem yang benar?").
- Mengidentifikasi dan menjelaskan empat level utama pengujian perangkat lunak: Pengujian Unit, Pengujian Integrasi, Pengujian Sistem, dan Uji Penerimaan Pengguna (UAT).
- Menganalisis tujuan, lingkup, dan pelaku dari setiap level pengujian, dari level mikro (kode) hingga level makro (bisnis).
- Membandingkan berbagai strategi dalam pengujian integrasi, termasuk pendekatan *Big Bang*, *Top-Down*, dan *Bottom-Up*.
- Menjelaskan peran krusial Uji Penerimaan Pengguna (UAT) sebagai tahap validasi akhir oleh pengguna untuk memastikan sistem memenuhi kebutuhan bisnis mereka.

Selamat datang di Bab 8. Kita telah melewati perjalanan panjang dari analisis, desain, hingga implementasi. Sistem yang kita rancang kini telah berwujud dalam barisan kode dan basis data yang fungsional. Namun, pekerjaan kita belum selesai. Sebuah jembatan yang baru selesai dibangun tidak akan langsung dibuka untuk umum sebelum diuji kekuatannya. Begitu pula dengan sistem informasi. Fase **pengujian** adalah tahap kritis di mana kita secara sistematis memeriksa, menginterogasi, dan bahkan mencoba "merusak" sistem yang telah kita bangun untuk satu tujuan mulia: menemukan kesalahan (*error*, *bug*, *defect*) sebelum ditemukan oleh pengguna akhir.

Banyak yang keliru menganggap pengujian sebagai aktivitas di akhir proyek untuk membuktikan bahwa sistem bekerja. Pandangan ini salah. Pengujian adalah aktivitas konstruktif yang destruktif; tujuannya adalah untuk menemukan di mana dan bagaimana sistem bisa gagal. Semakin banyak kesalahan yang kita temukan dan perbaiki pada tahap ini, semakin andal dan berkualitas produk yang akan kita serahkan. Dalam bab ini, kita akan mempelajari pendekatan berlapis dalam pengujian, yang sering dianalogikan dengan "V-Model". Kita akan mulai dari level terkecil dengan **Pengujian Unit**, memastikan setiap "batu bata" kode berfungsi dengan benar. Kemudian, kita akan naik ke **Pengujian Integrasi** untuk memastikan "batu bata" tersebut dapat direkatkan dengan baik. Selanjutnya, kita akan melakukan **Pengujian Sistem** untuk memeriksa apakah "bangunan" secara keseluruhan kokoh dan sesuai dengan cetak biru. Terakhir, dan yang paling penting, kita akan menyerahkan kunci kepada calon "penghuni" melalui **Uji Penerimaan Pengguna (UAT)** untuk memastikan bangunan tersebut benar-benar rumah yang mereka impikan.



Gambar 18. Ilustrasi Pengujian Sistem

8.1 Pengenalan Pengujian

Pengujian sistem adalah serangkaian aktivitas yang direncanakan dan sistematis untuk mengevaluasi sebuah sistem atau komponennya dengan tujuan menemukan apakah sistem tersebut memenuhi kebutuhan yang telah ditentukan dan untuk mengidentifikasi perbedaan antara hasil yang diharapkan dan hasil yang sebenarnya.

Filosofi Pengujian: Menemukan Kesalahan

Penting untuk memahami filosofi dasar di balik pengujian. Tujuan utama pengujian **bukanlah untuk menunjukkan bahwa perangkat lunak tidak memiliki kesalahan**. Membuktikan ketiadaan absolut dari kesalahan adalah hal yang mustahil untuk sistem yang cukup kompleks. Sebaliknya, tujuan pengujian adalah **untuk menemukan kesalahan sebanyak mungkin**. Seorang penguji yang baik memiliki pola pikir seorang detektif yang skeptis; mereka secara aktif mencari

skenario, input, dan kondisi yang dapat membuat sistem gagal. Proyek dianggap lebih berhasil jika tim pengujian menemukan dan memperbaiki 500 kesalahan sebelum rilis, daripada jika mereka hanya menemukan 50 kesalahan dan sisanya ditemukan oleh pelanggan.

Verifikasi vs. Validasi

Dalam konteks pengujian, dua istilah yang sering digunakan adalah verifikasi dan validasi. Keduanya penting untuk memastikan kualitas, tetapi memiliki fokus yang berbeda.

- **Verifikasi:** Menjawab pertanyaan, "**Apakah kita membangun sistem dengan benar?**" Ini adalah proses memeriksa apakah perangkat lunak memenuhi spesifikasi desain dan kebutuhan yang telah didokumentasikan. Verifikasi berfokus pada kepatuhan terhadap cetak biru. Sebagian besar aktivitas pengujian teknis (Unit, Integrasi, Sistem) adalah bentuk verifikasi.
- **Validasi:** Menjawab pertanyaan, "**Apakah kita membangun sistem yang benar?**" Ini adalah proses memeriksa apakah perangkat lunak yang telah dibangun benar-benar memenuhi kebutuhan bisnis dan harapan pengguna. Validasi berfokus pada kesesuaian dengan tujuan. Uji Penerimaan Pengguna (UAT) adalah bentuk utama dari validasi.

Anda bisa saja membangun sistem yang 100% terverifikasi (sesuai spesifikasi) tetapi gagal validasi (tidak berguna bagi pengguna). Inilah mengapa kedua proses tersebut sangat krusial.

8.2 Pengujian Unit (Unit Testing)

Pengujian unit adalah level pengujian paling dasar dan fundamental. Fokusnya adalah pada unit atau komponen terkecil dari perangkat lunak yang dapat diuji secara terisolasi. "Unit" ini bisa berupa sebuah fungsi, *method*, prosedur, atau *class*.

- **Tujuan:** Untuk memverifikasi bahwa setiap potongan kode individual berfungsi persis seperti yang dirancang. Pengujian ini bertujuan untuk menemukan kesalahan logika, perhitungan yang salah, atau perilaku yang tidak diharapkan di dalam unit itu sendiri, sebelum digabungkan dengan bagian lain.
- **Siapa yang Melakukan:** Biasanya dilakukan oleh **pemrogram** yang menulis kode tersebut. Dalam praktik modern seperti *Test-Driven Development* (TDD), pemrogram bahkan menulis tes unit *sebelum* menulis kode fungsionalnya.
- **Pendekatan:** Pengujian unit adalah bentuk **pengujian kotak putih (*white-box testing*)**, di mana penguji (pemrogram) mengetahui dan peduli tentang struktur internal dan logika kode. Mereka akan membuat kasus uji untuk memastikan setiap jalur logika (misalnya, setiap cabang IF-ELSE) dieksekusi dan memberikan hasil yang benar.

Contoh: Untuk sebuah fungsi hitungPajak(totalBelanja), tes unit akan mencakup skenario seperti:

- Memasukkan nilai positif (misalnya, 100.000) dan memeriksa apakah hasilnya adalah 10.000 (jika pajak 10%).
- Memasukkan nilai nol dan memeriksa apakah hasilnya nol.

- Memasukkan nilai negatif dan memeriksa apakah sistem menangani kesalahan dengan benar.

8.3 Pengujian Integrasi (Integration Testing)

Setelah unit-unit individual terbukti berfungsi dengan benar, langkah selanjutnya adalah menggabungkannya dan menguji interaksi di antara mereka. Inilah yang disebut dengan pengujian integrasi.

- **Tujuan:** Untuk menemukan kesalahan pada antarmuka (*interface*) dan interaksi antara komponen-komponen yang terintegrasi. Masalah yang sering ditemukan pada tahap ini termasuk data yang tidak cocok, pemanggilan fungsi yang salah, atau asumsi yang keliru tentang bagaimana modul lain bekerja.
- **Siapa yang Melakukan:** Biasanya dilakukan oleh pemrogram atau tim pengujian khusus (*Quality Assurance* - QA).
- **Pendekatan:** Pengujian integrasi dapat menggunakan pendekatan kotak putih, kotak hitam (*black-box*), atau kotak abu-abu (*grey-box*). Strategi untuk melakukan integrasi telah kita bahas di Bab 7, dan strategi ini secara langsung memengaruhi cara pengujian integrasi dilakukan.
    - **Pendekatan *Top-Down*:** Menguji dari modul tingkat atas ke bawah, menggunakan *stubs* untuk menggantikan modul tingkat rendah yang belum ada.
    - **Pendekatan *Bottom-Up*:** Menguji dari modul tingkat bawah ke atas, menggunakan *drivers* untuk memanggil dan menguji modul-modul tersebut.
    - **Pendekatan *Sandwich*:** Kombinasi dari *Top-Down* dan *Bottom-Up*.

Contoh: Dalam sistem e-commerce, pengujian integrasi akan memeriksa:

- Apakah modul "Keranjang Belanja" dapat dengan benar mengirimkan total harga ke modul "Pembayaran"?
- Setelah pembayaran berhasil, apakah modul "Pembayaran" dapat dengan benar memicu modul "Manajemen Pesanan" untuk mengubah status pesanan menjadi "Dibayar"?

8.4 Pengujian Sistem (System Testing)

Pada tahap ini, semua modul telah diintegrasikan dan sistem dipandang sebagai satu kesatuan yang utuh. Pengujian sistem adalah pengujian terhadap sistem yang telah terintegrasi penuh untuk memverifikasi bahwa sistem tersebut memenuhi semua kebutuhan yang telah ditentukan dalam dokumen SRS (Spesifikasi Kebutuhan Sistem).

- **Tujuan:** Untuk mengevaluasi kepatuhan sistem terhadap kebutuhan fungsional dan non-fungsional. Ini adalah level pertama di mana sistem diuji dari ujung ke ujung (*end-to-end*) dari perspektif pengguna.
- **Siapa yang Melakukan:** Dilakukan oleh **tim pengujian independen** (tim QA), bukan oleh tim pengembang. Ini penting untuk memastikan objektivitas.
- **Pendekatan:** Pengujian sistem hampir selalu merupakan **pengujian kotak hitam (*black-box testing*)**. Penguji tidak perlu tahu tentang struktur kode internal. Mereka hanya peduli pada input dan output, dan apakah perilaku sistem sesuai dengan spesifikasi.

Pengujian sistem mencakup berbagai jenis pengujian, antara lain:

- **Pengujian Fungsional:** Memeriksa apakah setiap fungsi yang dijelaskan dalam SRS bekerja dengan benar.
- **Pengujian Kinerja (*Performance Testing*):** Mengukur waktu respons, *throughput*, dan penggunaan sumber daya sistem di bawah beban kerja yang berbeda.
- **Pengujian Keamanan (*Security Testing*):** Mencoba menemukan kerentanan dalam sistem, seperti celah untuk akses tidak sah atau kebocoran data.
- **Pengujian Kegunaan (*Usability Testing*):** Mengevaluasi seberapa mudah dan intuitif antarmuka pengguna sistem.

Contoh: Untuk sistem pendaftaran ulang mahasiswa, pengujian sistem akan melibatkan skenario lengkap seperti:

- Seorang penguji (berperan sebagai mahasiswa) mencoba untuk login, memilih mata kuliah, mendapatkan persetujuan, melakukan pembayaran, dan mencetak KRS, lalu memverifikasi bahwa semua data tercatat dengan benar di basis data.

8.5 Uji Penerimaan (User Acceptance Testing - UAT)

Uji Penerimaan Pengguna adalah tahap pengujian formal terakhir sebelum sistem diluncurkan secara resmi. Pada tahap ini, sistem diuji oleh pengguna akhir atau klien yang sebenarnya untuk memastikan bahwa sistem dapat menangani tugas-tugas bisnis di dunia nyata dan memenuhi kebutuhan mereka.

- **Tujuan:** Untuk **validasi**, bukan verifikasi. Tujuannya adalah untuk mendapatkan konfirmasi dan persetujuan dari pengguna bahwa sistem yang dibangun dapat diterima dan siap untuk digunakan dalam operasi sehari-hari.
- **Siapa yang Melakukan:** Dilakukan oleh **pengguna akhir (end-users)**, klien, atau perwakilan mereka.

- **Pendekatan:** UAT adalah bentuk pengujian kotak hitam yang berfokus pada skenario bisnis. Pengguna akan menjalankan serangkaian kasus uji yang merepresentasikan alur kerja mereka yang sebenarnya.

Ada dua jenis utama UAT:

1. **Alpha Testing:** Dilakukan di lingkungan pengembang (atau lingkungan pengujian yang terkontrol) oleh sekelompok pengguna internal (misalnya, karyawan dari departemen lain). Tim pengembang biasanya hadir untuk mencatat setiap kesalahan dan masalah.
2. **Beta Testing:** Dilakukan di lingkungan pengguna yang sebenarnya oleh sekelompok pengguna eksternal yang terbatas. Perangkat lunak dirilis ke grup ini ("beta testers") sebelum dirilis ke publik. Tujuannya adalah untuk mendapatkan umpan balik dari penggunaan di dunia nyata dalam berbagai kondisi yang tidak dapat disimulasikan di laboratorium.

Keberhasilan melewati UAT biasanya menjadi syarat bagi tim pengembang untuk mendapatkan "tanda tangan" dari klien, yang menandakan bahwa proyek telah selesai dan dapat diluncurkan.

Tabel 18. Ringkasan Level-level Pengujian

Level Pengujian	Fokus Utama	Siapa yang Melakukan	Tipe Pengujian	Tujuan
Unit	Komponen/Fungsi individual	Pengembang	Kotak Putih	Verifikasi logika internal kode

Integrasi	Interaksi antar komponen	Pengembang / Tim QA	Kotak Putih/Hitam	Verifikasi antarmuka antar modul
Sistem	Sistem terintegrasi penuh	Tim QA Independen	Kotak Hitam	Verifikasi terhadap SRS (fungsional & non-fungsional)
Penerimaan (UAT)	Kesesuaian dengan kebutuhan bisnis	Pengguna Akhir / Klien	Kotak Hitam	Validasi bahwa sistem siap digunakan

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Jelaskan dengan kata-kata Anda sendiri perbedaan antara Pengujian Sistem dan Uji Penerimaan Pengguna (UAT). Mengapa UAT tetap diperlukan meskipun sebuah sistem telah lulus semua skenario Pengujian Sistem dengan sempurna?
2. Sebuah modul perangkat lunak untuk menghitung gaji karyawan bergantung pada modul lain untuk mendapatkan data jam kerja dan modul lainnya lagi untuk mendapatkan data tarif pajak. Strategi pengujian integrasi mana (*Top-Down* atau *Bottom-Up*) yang lebih masuk akal untuk menguji modul penghitungan gaji ini terlebih dahulu? Jelaskan alasannya.
3. Apa yang dimaksud dengan pengujian "kotak hitam" (*black-box testing*)? Level pengujian mana yang paling sering menggunakan pendekatan ini, dan mengapa?

Solusi dan Pembahasan Soal Konseptual

1. **Perbedaan Pengujian Sistem dan UAT:**
    - **Perbedaan: Pengujian Sistem** berfokus pada **verifikasi**; tujuannya adalah untuk memeriksa apakah sistem yang dibangun sesuai dengan spesifikasi teknis dan fungsional yang tertulis di dokumen SRS. Ini dilakukan oleh tim QA. **UAT** berfokus pada **validasi**; tujuannya adalah untuk memeriksa apakah sistem tersebut benar-benar berguna dan dapat diterima untuk menyelesaikan pekerjaan di dunia nyata dari

sudut pandang pengguna bisnis. Ini dilakukan oleh pengguna akhir.

- **Mengapa UAT Tetap Perlu:** UAT tetap diperlukan karena sebuah sistem bisa saja secara teknis sempurna (lulus Pengujian Sistem) tetapi tidak memenuhi kebutuhan bisnis yang sebenarnya. Mungkin ada kesalahpahaman dalam dokumen SRS, atau alur kerja yang dirancang secara teknis benar tetapi secara praktis tidak efisien bagi pengguna. UAT adalah jaring pengaman terakhir untuk memastikan sistem yang dibangun adalah sistem yang *benar*, bukan hanya sistem yang dibangun dengan *benar*.

2. Strategi Integrasi untuk Modul Gaji:

   Strategi yang lebih masuk akal adalah Pengujian Integrasi Bottom-Up.

   - **Alasan:** Modul penghitungan gaji (modul tingkat menengah/atas) sangat bergantung pada data yang akurat dari modul tingkat bawah (modul jam kerja dan modul tarif pajak). Dengan pendekatan *Bottom-Up*, kita akan menguji modul jam kerja dan modul tarif pajak terlebih dahulu secara terpisah (menggunakan *drivers*). Setelah kita yakin kedua modul fundamental ini bekerja dengan benar, barulah kita mengintegrasikannya dengan modul penghitungan gaji. Ini memastikan bahwa jika ada kesalahan pada saat pengujian integrasi,

kemungkinan besar masalahnya ada di modul gaji atau antarmukanya, bukan pada data input yang diterimanya.

3. **Pengujian Kotak Hitam (*Black-Box Testing*):**
    - **Definisi:** Pengujian kotak hitam adalah metode pengujian di mana penguji tidak memiliki pengetahuan tentang struktur internal, desain, atau kode dari sistem yang diuji. Penguji hanya berfokus pada fungsionalitas sistem dari luar, yaitu dengan memberikan input dan memvalidasi output yang dihasilkan, seolah-olah sistem adalah "kotak hitam" yang misterius.
    - **Level Pengujian:** Pendekatan ini paling sering digunakan pada **Pengujian Sistem** dan **Uji Penerimaan Pengguna (UAT)**.
    - **Alasan:** Pada level-level ini, fokusnya adalah pada perilaku sistem secara keseluruhan dari sudut pandang pengguna, bukan pada detail implementasi teknis. Tim QA (pada Pengujian Sistem) dan pengguna akhir (pada UAT) tidak perlu menjadi pemrogram. Tugas mereka adalah untuk memastikan sistem berperilaku sesuai dengan kebutuhan yang diharapkan, terlepas dari bagaimana kode di dalamnya ditulis.

Studi Kasus: Pengujian Sistem Reservasi Hotel "StayEasy"

- **Konteks:** Sebuah sistem reservasi hotel online baru bernama "StayEasy" telah selesai dikembangkan. Sistem ini memiliki modul-modul utama: Manajemen Kamar,

Pencarian & Reservasi, Manajemen Pengguna, dan Proses Pembayaran.

- **Tujuan:** Tim QA dan perwakilan dari pihak hotel (klien) akan memulai fase pengujian.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan:

Rancanglah sebuah rencana pengujian singkat untuk sistem "StayEasy". Untuk setiap level pengujian (Unit, Integrasi, Sistem, UAT), berikan satu contoh skenario pengujian yang spesifik.

Analisis Rencana Pengujian:
Berikut adalah contoh rencana pengujian singkat untuk sistem "StayEasy" di setiap level:

1. **Pengujian Unit:**
    - **Skenario Spesifik:** Menguji fungsi `hitungTotalHarga(jumlahMalam, hargaPerMalam, kodeDiskon)`.
        - **Kasus Uji 1:** `hitungTotalHarga(3, 500000, null)` harus mengembalikan `1500000`.
        - **Kasus Uji 2:** `hitungTotalHarga(2, 700000, "DISKON10")` harus mengembalikan `1260000` (dengan asumsi diskon 10%).
        - **Kasus Uji 3:** `hitungTotalHarga(0, 500000, null)` harus mengembalikan `0`.
2. **Pengujian Integrasi:**

- **Skenario Spesifik:** Menguji integrasi antara modul **Manajemen Kamar** dan modul **Pencarian & Reservasi**.
    - **Langkah-langkah:**
        1. Di modul Manajemen Kamar, ubah status sebuah kamar (misal, Kamar 101) dari "Tersedia" menjadi "Dalam Perbaikan".
        2. Di modul Pencarian & Reservasi, lakukan pencarian kamar untuk tanggal yang relevan.
        3. **Verifikasi:** Pastikan Kamar 101 **tidak muncul** dalam hasil pencarian.
        4. Ubah kembali status Kamar 101 menjadi "Tersedia".
        5. Lakukan pencarian ulang.
        1. **Verifikasi:** Pastikan Kamar 101 **muncul** dalam hasil pencarian.

3. **Pengujian Sistem:**
    - **Skenario Spesifik:** Pengujian fungsional *end-to-end* untuk alur reservasi oleh pengguna tamu.
        - **Langkah-langkah:**
            1. Buka halaman utama sebagai pengguna yang belum login.
            2. Masukkan tanggal check-in dan check-out, serta jumlah tamu. Klik "Cari".

3. Dari daftar kamar yang tersedia, pilih satu kamar dan klik "Pesan Sekarang".
4. Isi formulir data diri (nama, email, telepon).
5. Pilih metode pembayaran (misal, Kartu Kredit) dan masukkan data kartu kredit dummy yang valid.
6. Klik "Konfirmasi & Bayar".
7. **Verifikasi:**
    - Sistem menampilkan halaman konfirmasi dengan nomor reservasi.
    - Email konfirmasi terkirim ke alamat email yang dimasukkan.
    - Di *backend* (dilihat oleh penguji), status kamar yang dipesan berubah menjadi "Dipesan" untuk tanggal tersebut.
    - Transaksi pembayaran tercatat di log pembayaran.

4. **Uji Penerimaan Pengguna (UAT):**
    - **Skenario Spesifik:** Skenario bisnis yang dilakukan oleh staf resepsionis hotel.

- **Tugas:** "Seorang tamu, Bapak Budi, menelepon untuk mengubah tanggal reservasi dengan nomor booking #XYZ123 dari tanggal 15-17 Agustus menjadi 20-22 Agustus. Lakukan perubahan tersebut di sistem."
- **Langkah-langkah (dilakukan oleh staf resepsionis):**
    1. Login ke sistem dengan akun resepsionis.
    2. Cari reservasi berdasarkan nomor booking #XYZ123.
    3. Buka detail reservasi dan klik tombol "Ubah Reservasi".
    4. Ubah tanggal check-in dan check-out.
    5. Sistem memeriksa ketersediaan kamar untuk tanggal baru.
    6. Simpan perubahan.
- **Kriteria Penerimaan (Validasi oleh staf):**
    1. Apakah prosesnya mudah dan intuitif?
    2. Apakah sistem memberikan informasi yang jelas tentang ketersediaan kamar di tanggal baru?

3. Apakah perubahan tersebut tercatat dengan benar di kalender reservasi hotel?
4. Apakah prosesnya lebih cepat daripada menggunakan sistem lama?

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.
- Laudon, K. C., & Laudon, J. P. (Edisi terbaru). *Management Information Systems: Managing the Digital Firm*. Pearson.

Bacaan Tambahan (Dianjurkan)

- Pressman, R. S., & Maxim, B. R. (Edisi terbaru). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education. (Memberikan pembahasan yang sangat mendalam dan teknis tentang berbagai strategi dan teknik pengujian perangkat lunak).
- Myers, G. J., Sandler, C., & Badgett, T. (Edisi terbaru). *The Art of Software Testing*. Wiley. (Buku klasik yang membahas filosofi dan pendekatan praktis dalam menjadi seorang penguji perangkat lunak yang efektif).

- Publikasi dari International Software Testing Qualifications Board (ISTQB). (ISTQB menyediakan silabus dan glosarium standar industri yang menjadi acuan bagi para profesional pengujian di seluruh dunia).

Bab 9: Dokumentasi Sistem

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan peran krusial dokumentasi sebagai bagian integral dari siklus hidup pengembangan sistem, bukan sebagai aktivitas tambahan.
- Membedakan secara jelas antara dokumentasi teknis dan dokumentasi pengguna, termasuk tujuan, audiens, dan konten utama dari masing-masing jenis.
- Mengidentifikasi dan menguraikan komponen-komponen esensial dari dokumentasi teknis, seperti spesifikasi desain, dokumentasi API, dan komentar kode.
- Mengidentifikasi dan menguraikan komponen-komponen esensial dari dokumentasi pengguna, seperti panduan pengguna, tutorial, dan FAQ.
- Menjelaskan pentingnya penerapan standar penulisan, templat, dan kontrol versi untuk menjaga kualitas dan konsistensi dokumentasi.
- Menganalisis tantangan dalam pemeliharaan dokumentasi dan merumuskan strategi efektif untuk menjaga agar dokumentasi tetap relevan dan mutakhir seiring dengan evolusi sistem.

Selamat datang di Bab 9. Kita telah melalui perjalanan panjang dari analisis, desain, hingga implementasi sistem. Namun, sebuah sistem, secanggih apa pun, tidak akan lengkap tanpa adanya **dokumentasi**. Seringkali dianggap sebagai tugas akhir yang membosankan, dokumentasi sebenarnya adalah salah satu pilar terpenting yang menopang keberlanjutan, pemeliharaan, dan kegunaan sebuah sistem. Tanpa dokumentasi yang baik, sistem informasi ibarat sebuah mesin canggih tanpa buku manual atau cetak biru; pengguna tidak tahu cara mengoperasikannya, dan mekanik tidak tahu cara memperbaikinya.

Dokumentasi adalah jembatan komunikasi lintas waktu. Ia berkomunikasi dengan pengguna baru tentang cara memanfaatkan sistem secara maksimal. Ia berkomunikasi dengan tim pengembang di masa depan tentang mengapa sebuah keputusan desain dibuat, memungkinkan mereka untuk memperbaiki dan mengembangkan sistem tanpa harus menebak-nebak. Dalam bab ini, kita akan membedah dua dunia utama dalam dokumentasi: **dokumentasi teknis** yang ditujukan bagi para pembangun dan pemelihara sistem, dan **dokumentasi pengguna** yang ditujukan bagi mereka yang akan mengoperasikan sistem sehari-hari. Kita juga akan membahas pentingnya **standar penulisan** untuk memastikan kejelasan dan konsistensi, serta tantangan terbesar dari semuanya: bagaimana **memelihara dokumentasi** agar tetap hidup dan relevan seiring dengan perkembangan sistem. Menguasai seni dokumentasi adalah tanda seorang profesional sejati dalam analisis dan perancangan sistem.

9.1 Dokumentasi Teknis

Dokumentasi teknis adalah kumpulan materi yang menjelaskan arsitektur, desain, kode, dan cara kerja internal sebuah sistem. Audiens utamanya adalah para profesional teknis, seperti pengembang perangkat lunak, analis sistem, administrator basis data, dan tim pemeliharaan yang akan bekerja dengan sistem tersebut di masa depan. Tujuannya adalah untuk menyediakan informasi yang cukup agar mereka dapat memahami, memelihara, memodifikasi, dan bahkan mengintegrasikan sistem dengan sistem lain.

Komponen Utama Dokumentasi Teknis

Dokumentasi teknis bukanlah satu dokumen tunggal, melainkan sebuah koleksi artefak yang dihasilkan sepanjang siklus hidup pengembangan. Komponen-komponen utamanya meliputi:

1. **Dokumen Analisis dan Desain:** Ini adalah pondasi dari dokumentasi teknis. Semua artefak yang kita hasilkan di Bab 5 dan 6 masuk dalam kategori ini.
    - **Spesifikasi Kebutuhan Sistem (SRS):** Menjelaskan secara detail apa yang harus dilakukan sistem.
    - **Diagram Alir Data (DFD):** Memvisualisasikan aliran data dan proses.
    - ***Entity Relationship Diagram* (ERD):** Menggambarkan struktur data dan basis data.
    - **Dokumen Desain Arsitektur:** Menjelaskan arsitektur tingkat tinggi yang dipilih (misalnya, *three-tier*) dan justifikasi di baliknya.
    - **Desain Antarmuka Pengguna (*Wireframe* & *Mockup*):** Menunjukkan tata letak dan tampilan visual sistem.

2. **Komentar Kode (*Code Comments*):**
    - Ini adalah dokumentasi yang ditulis langsung di dalam kode sumber (*source code*). Komentar yang baik tidak menjelaskan *apa* yang dilakukan kode (karena kode yang baik seharusnya sudah jelas), melainkan *mengapa* kode tersebut ditulis dengan cara tertentu.
    - *Contoh:* Menjelaskan algoritma yang kompleks, alasan di balik penggunaan nilai konstanta tertentu, atau referensi ke tiket perbaikan *bug* yang terkait.
3. **Dokumentasi API (*Application Programming Interface*):**
    - Sangat krusial untuk sistem yang dirancang untuk berkomunikasi dengan sistem lain. Dokumentasi API adalah "buku manual" bagi pengembang lain yang ingin menggunakan layanan yang disediakan oleh sistem kita.
    - **Isi:** Penjelasan tentang setiap *endpoint* (URL), metode HTTP yang didukung (GET, POST, dll.), parameter yang dibutuhkan, format permintaan dan respons (biasanya dalam format JSON), serta contoh kode. Standar seperti **OpenAPI (Swagger)** sering digunakan untuk menghasilkan dokumentasi API yang interaktif.
4. **Panduan Instalasi dan Konfigurasi:**
    - Ditujukan untuk administrator sistem atau tim DevOps. Dokumen ini berisi instruksi langkah demi langkah tentang cara memasang (*install*) sistem di lingkungan server, termasuk prasyarat perangkat keras dan perangkat lunak, konfigurasi basis data, dan pengaturan variabel lingkungan.

5. **Catatan Rilis (*Release Notes*):**
    - Dokumen yang menyertai setiap versi baru dari perangkat lunak yang dirilis.
    - **Isi:** Ringkasan tentang fitur-fitur baru yang ditambahkan, perbaikan *bug* yang telah dilakukan, masalah yang diketahui (*known issues*), dan perubahan signifikan lainnya sejak versi sebelumnya.

9.2 Dokumentasi Pengguna

Jika dokumentasi teknis adalah tentang "cara kerja sistem", maka dokumentasi pengguna adalah tentang "cara menggunakan sistem". Audiensnya adalah pengguna akhir, orang-orang yang akan berinteraksi dengan sistem setiap hari untuk melakukan pekerjaan mereka. Bahasa yang digunakan harus sederhana, jelas, dan bebas dari jargon teknis. Tujuannya adalah untuk memberdayakan pengguna agar dapat memanfaatkan sistem secara mandiri dan percaya diri.

Komponen Utama Dokumentasi Pengguna

Dokumentasi pengguna dapat hadir dalam berbagai format untuk melayani kebutuhan yang berbeda, mulai dari pengenalan cepat hingga referensi mendalam.

1. **Panduan Memulai Cepat (*Quick Start Guide*):**
    - Sebuah dokumen singkat (seringkali hanya satu atau dua halaman) yang dirancang untuk membantu pengguna baru memulai tugas-tugas paling dasar secepat mungkin. Tujuannya adalah memberikan "kemenangan cepat" dan membangun kepercayaan diri pengguna.

2. **Panduan Pengguna Lengkap (*User Manual*):**
    - Ini adalah referensi komprehensif yang mencakup semua fitur dan fungsionalitas sistem. Biasanya diorganisir berdasarkan modul atau tugas (misalnya, "Cara Mengelola Data Pelanggan", "Cara Membuat Laporan Penjualan"). Panduan ini harus mudah dinavigasi, seringkali dengan daftar isi, indeks, dan fungsi pencarian (jika online).
3. **Tutorial:**
    - Berbeda dari panduan yang bersifat referensi, tutorial bersifat instruksional. Tutorial memandu pengguna langkah demi langkah melalui sebuah alur kerja atau skenario tertentu untuk mencapai tujuan spesifik. Seringkali disajikan dalam format video atau panduan interaktif.
4. **FAQ (*Frequently Asked Questions*):**
    - Kumpulan pertanyaan yang paling sering diajukan oleh pengguna beserta jawaban yang jelas dan ringkas. FAQ sangat efektif untuk mengatasi masalah umum dan mengurangi beban tim dukungan teknis (*help desk*).
5. **Bantuan Kontekstual (*Context-Sensitive Help*):**
    - Bantuan yang terintegrasi langsung ke dalam antarmuka pengguna.
    - *Contoh:* Ikon tanda tanya (?) di sebelah kolom formulir yang, ketika diklik, akan menampilkan penjelasan tentang data apa yang harus dimasukkan. Atau *tooltips* yang muncul saat pengguna mengarahkan kursor ke sebuah tombol.

6. **Materi Pelatihan:**
    - Ini mencakup semua materi yang digunakan dalam sesi pelatihan formal, seperti presentasi slide, buku kerja, dan contoh data untuk latihan.

9.3 Standar Penulisan Dokumentasi

Membuat dokumentasi yang baik bukan hanya tentang apa yang Anda tulis, tetapi juga bagaimana Anda menuliskannya. Tanpa standar, dokumentasi yang dibuat oleh anggota tim yang berbeda akan terlihat dan terasa berbeda, membuatnya sulit dibaca dan dipahami. Standar penulisan memastikan **konsistensi, kejelasan, dan profesionalisme**.

Elemen Kunci dalam Standar Dokumentasi

1. **Templat (*Templates*):**
    - Menggunakan templat yang telah ditentukan untuk berbagai jenis dokumen (misalnya, templat untuk laporan analisis, templat untuk panduan pengguna) adalah cara termudah untuk memastikan konsistensi. Templat mendefinisikan struktur, bagian-bagian yang harus ada, dan format dasar.
2. **Panduan Gaya (*Style Guide*):**
    - Ini adalah seperangkat aturan tentang cara menulis. Panduan gaya yang baik mencakup:
        - **Terminologi:** Menetapkan istilah resmi yang harus digunakan di seluruh dokumentasi (misalnya, selalu gunakan

"Pelanggan", bukan "Klien" atau "Kustomer").

- **Nada Suara (*Tone of Voice*):** Menentukan apakah bahasa yang digunakan harus formal, semi-formal, atau santai. Untuk dokumentasi pengguna, nada yang jelas, langsung, dan membantu biasanya paling efektif.
- **Aturan Format:** Aturan tentang penggunaan huruf tebal, miring, judul, daftar bernomor, dan poin-poin.
- **Konvensi Tangkapan Layar (*Screenshot*):** Aturan tentang bagaimana mengambil dan memberi anotasi pada gambar tangkapan layar agar terlihat seragam.

3. **Sistem Kontrol Versi (*Version Control*):**
   - Dokumentasi adalah artefak proyek yang hidup, sama seperti kode. Oleh karena itu, ia harus dikelola dalam sistem kontrol versi seperti Git. Ini memungkinkan tim untuk melacak setiap perubahan, melihat siapa yang mengubah apa dan kapan, serta dengan mudah kembali ke versi sebelumnya jika terjadi kesalahan.

4. **Proses Tinjauan (*Review Process*):**
   - Tidak ada penulis yang sempurna. Setiap draf dokumentasi harus ditinjau oleh orang lain. Proses tinjauan yang baik melibatkan:
     - **Tinjauan Sejawat (*Peer Review*):** Ditinjau oleh anggota tim lain untuk kejelasan dan kelengkapan.

- **Tinjauan Teknis:** Ditinjau oleh ahli materi pelajaran (*subject matter expert*) atau pengembang untuk memastikan akurasi teknis.
- **Tinjauan Pengguna:** Jika memungkinkan, mintalah beberapa pengguna target untuk membaca draf dokumentasi pengguna dan memberikan umpan balik tentang apakah dokumen tersebut mudah dipahami dan membantu.

9.4 Pemeliharaan Dokumentasi

Tantangan terbesar dalam dokumentasi bukanlah membuatnya, tetapi **menjaganya agar tetap relevan**. Dokumentasi yang tidak lagi sesuai dengan sistem yang sebenarnya disebut mengalami **pembusukan dokumentasi (*documentation rot*)**. Dokumentasi yang usang tidak hanya tidak berguna, tetapi juga berbahaya karena dapat menyesatkan pengguna dan pengembang, menyebabkan kesalahan dan frustrasi.

Strategi Efektif untuk Pemeliharaan Dokumentasi

1. **"Dokumentasi sebagai Kode" (*Documentation as Code*):**
    - Ini adalah pendekatan modern di mana dokumentasi diperlakukan persis seperti kode. Ia ditulis dalam format teks sederhana (seperti Markdown), disimpan di repositori Git yang sama

dengan kode sumber, dan diperbarui sebagai bagian dari alur kerja pengembangan normal. Setiap kali seorang pengembang membuat perubahan pada fitur, mereka juga bertanggung jawab untuk memperbarui dokumentasi terkait dalam *commit* yang sama.

2. **Integrasikan ke dalam "Definisi Selesai" (*Definition of Done*):**
    - Dalam metodologi Agile seperti Scrum, tim memiliki kriteria yang jelas tentang kapan sebuah pekerjaan dianggap "Selesai". Salah satu kriteria terpenting adalah "dokumentasi telah diperbarui". Sebuah fitur baru tidak dianggap selesai sampai dokumentasi pengguna dan teknis yang relevan telah ditulis, ditinjau, dan disetujui.

3. **Tetapkan Kepemilikan yang Jelas:**
    - Setiap dokumen atau bagian dari dokumentasi harus memiliki pemilik yang jelas. Pemilik ini bertanggung jawab untuk memastikan kontennya tetap akurat. Tanpa kepemilikan, semua orang akan berasumsi bahwa orang lain yang akan memperbaruinya.

4. **Lakukan Audit Secara Berkala:**
    - Jadwalkan waktu secara teratur (misalnya, setiap kuartal atau setiap dua rilis) untuk meninjau seluruh set dokumentasi. Audit ini bertujuan untuk menemukan informasi yang usang, tautan yang rusak, atau bagian yang tidak lagi relevan.

5. **Manfaatkan Otomatisasi:**
    - Kurangi beban pemeliharaan manual dengan mengotomatiskan pembuatan dokumentasi jika memungkinkan. Misalnya, alat seperti Swagger/OpenAPI dapat secara otomatis menghasilkan dokumentasi API yang interaktif langsung dari komentar di dalam kode.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Anda sedang menulis panduan pengguna untuk aplikasi kasir (*Point of Sale*) baru yang akan digunakan oleh staf di sebuah kafe. Siapakah audiens utama Anda? Sebutkan dan jelaskan tiga bagian terpenting yang akan Anda masukkan ke dalam panduan tersebut.
2. Seorang pengembang di tim Anda mengatakan, "Kode saya sangat bersih dan mudah dibaca, jadi tidak perlu diberi komentar." Bagaimana Anda akan menanggapi pernyataan ini sebagai seorang analis sistem?
3. Mengapa menjaga dokumentasi agar tetap mutakhir sama pentingnya dengan membuatnya pertama kali? Jelaskan satu strategi spesifik untuk mencegah "pembusukan dokumentasi".

Solusi dan Pembahasan Soal Konseptual

1. **Panduan Pengguna Aplikasi Kasir:**
    - **Audiens Utama:** Staf kasir kafe, yang mungkin memiliki tingkat keahlian teknis yang bervariasi, dari pemula hingga berpengalaman. Bahasa yang digunakan harus sederhana dan fokus pada tugas.
    - **Tiga Bagian Terpenting:**
        1. **Panduan Memulai Cepat (*Quick Start Guide*):** Bagian ini akan mencakup cara *login*, membuka sesi kasir, dan melakukan transaksi penjualan tunai

sederhana. Tujuannya agar kasir baru bisa langsung bekerja setelah membaca satu halaman.

2. **Menangani Transaksi Umum:** Bagian ini akan berisi tutorial langkah demi langkah untuk skenario paling umum, seperti: memproses pembayaran kartu kredit/debit, memberikan diskon, membatalkan item, dan mencetak ulang struk.
3. **Penyelesaian Masalah (FAQ):** Bagian ini akan menjawab pertanyaan umum seperti, "Apa yang harus dilakukan jika printer struk macet?" atau "Bagaimana cara melakukan *refund*?". Ini membantu kasir mengatasi masalah kecil secara mandiri.

2. Tanggapan untuk Pengembang:

   Saya akan setuju bahwa kode yang bersih itu penting, namun itu tidak menghilangkan kebutuhan akan komentar. Saya akan menjelaskan bahwa: "Kode yang bersih menjelaskan apa yang dilakukannya, tetapi komentar yang baik menjelaskan mengapa hal itu dilakukan. Enam bulan dari sekarang, ketika kita atau pengembang baru melihat kode ini, kita mungkin tidak ingat alasan di balik pilihan algoritma yang kompleks itu atau mengapa kita harus menangani kasus pengecualian yang aneh itu. Komentar adalah catatan untuk diri kita di masa depan dan untuk rekan tim kita,

yang akan menghemat banyak waktu saat melakukan debugging atau pemeliharaan."

3. **Pentingnya Pemeliharaan Dokumentasi:**
    - **Pentingnya:** Menjaga dokumentasi tetap mutakhir sama pentingnya karena dokumentasi yang usang lebih buruk daripada tidak ada dokumentasi sama sekali. Dokumentasi yang salah dapat menyesatkan pengguna (menyebabkan frustrasi dan kesalahan) dan pengembang (menyebabkan mereka membuat perbaikan yang salah atau membuang waktu untuk memahami kode yang tidak lagi cocok dengan deskripsinya). Ini merusak kepercayaan pada semua dokumentasi.
    - **Strategi Pencegahan:** Salah satu strategi paling efektif adalah **mengintegrasikan pembaruan dokumentasi ke dalam "Definisi Selesai" (*Definition of Done*)** dalam proses kerja tim. Ini berarti sebuah tugas atau fitur tidak dianggap selesai 100% sampai kode yang berfungsi, pengujian yang lulus, *dan* dokumentasi yang relevan (baik teknis maupun pengguna) telah diperbarui dan ditinjau. Ini mengubah dokumentasi dari sebuah renungan menjadi bagian yang tidak terpisahkan dari pekerjaan pengembangan itu sendiri.

Studi Kasus: Warisan Sistem Informasi Kepegawaian

- **Konteks:** PT Maju Jaya menggunakan Sistem Informasi Kepegawaian (SIK) yang dibangun lima tahun lalu. Sistem ini sangat penting untuk mengelola data karyawan, penggajian, dan cuti. Namun, tim pengembang asli yang membangun sistem tersebut telah lama pindah ke perusahaan lain.

- **Masalah:**
    1. Departemen SDM ingin menambahkan fitur baru untuk manajemen kinerja, tetapi tim pengembang saat ini tidak tahu harus mulai dari mana. Mereka kesulitan memahami struktur basis data dan bagaimana modul-modul yang ada saling berinteraksi.
    2. Staf SDM yang baru sering membuat kesalahan saat memasukkan data cuti karena mereka tidak yakin tentang aturan-aturan yang berlaku (misalnya, sisa cuti tahunan, cuti khusus). Mereka terus-menerus bertanya kepada rekan senior, yang memperlambat pekerjaan semua orang.
- **Investigasi:** Setelah dicari, satu-satunya dokumentasi yang ditemukan adalah beberapa diagram ERD versi awal dan presentasi PowerPoint yang digunakan saat peluncuran sistem lima tahun lalu.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan:

1. Identifikasi dua jenis dokumentasi utama (teknis dan pengguna) yang jelas-jelas tidak ada atau tidak memadai dalam skenario ini.
2. Untuk setiap jenis dokumentasi yang Anda identifikasi, sebutkan dua dokumen spesifik yang, jika dibuat dan dipelihara dengan baik, dapat mencegah masalah yang dihadapi PT Maju Jaya. Jelaskan secara singkat isi dari setiap dokumen tersebut.

Analisis:

1. **Jenis Dokumentasi yang Hilang:**
    - **Dokumentasi Teknis:** Sangat tidak memadai. Ketiadaan dokumentasi ini menyebabkan tim pengembang baru tidak dapat memelihara dan mengembangkan sistem secara efektif.
    - **Dokumentasi Pengguna:** Juga sangat tidak memadai. Ketiadaan panduan yang jelas menyebabkan pengguna baru kesulitan dan sering membuat kesalahan.
2. **Dokumen Spesifik yang Dibutuhkan:**
    - **Untuk Mengatasi Masalah Tim Pengembang (Dokumentasi Teknis):**
        1. **Dokumen Desain Arsitektur:** Dokumen ini akan menjelaskan struktur tingkat tinggi dari SIK, bagaimana sistem dipecah menjadi modul-modul (misalnya, modul Penggajian, modul Cuti, modul Data Karyawan), dan bagaimana modul-modul tersebut

berkomunikasi. Ini akan memberi tim baru "peta" sistem secara keseluruhan.

2. **Kamus Data (*Data Dictionary*) dan ERD Final:** ERD versi awal tidak cukup. Tim membutuhkan ERD final yang sesuai dengan basis data saat ini, beserta kamus data yang menjelaskan setiap tabel dan kolom (misalnya, apa arti kolom `status_karyawan` dengan nilai 'A1' vs 'A2'). Ini akan menghilangkan tebakan saat bekerja dengan basis data.

- **Untuk Mengatasi Masalah Staf SDM (Dokumentasi Pengguna):**
    1. **Panduan Pengguna (User Manual):** Dokumen ini akan berisi instruksi langkah demi langkah yang terorganisir berdasarkan tugas. Misalnya, akan ada bab khusus "Mengelola Cuti Karyawan" yang menjelaskan cara mengajukan cuti, cara menyetujui cuti, dan cara melihat sisa cuti. Panduan ini akan menjadi referensi utama bagi staf baru.
    2. **FAQ atau Basis Pengetahuan (*Knowledge Base*):** Sebuah dokumen atau halaman web yang berisi jawaban atas pertanyaan-pertanyaan umum terkait aturan bisnis. Contoh isinya: "T: Berapa sisa cuti tahunan yang hangus jika tidak diambil? J: Sisa cuti maksimal 6 hari dapat diakumulasikan ke tahun berikutnya." Ini akan memberdayakan

staf untuk menemukan jawaban secara mandiri.

Daftar Bahan Bacaan

Buku Teks Utama (Wajib)

- Kendall, K. E., & Kendall, J. E. (Edisi terbaru). *Systems Analysis and Design*. Pearson.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (Edisi terbaru). *Systems Analysis and Design in a Changing World*. Cengage Learning.

Bacaan Tambahan (Dianjurkan)

- **The Good Docs Project.** Tersedia di: https://thegooddocsproject.dev/. Sebuah proyek komunitas yang menyediakan templat dan praktik terbaik untuk menulis dokumentasi teknis yang berkualitas tinggi.
- **Microsoft Manual of Style.** (Edisi terbaru). Microsoft Press. Meskipun berfokus pada produk Microsoft, buku ini adalah referensi klasik yang memberikan panduan gaya yang sangat baik untuk penulisan teknis secara umum.
- "**What is Docs as Code?**" - Artikel oleh Anne Gentle di situs *Write the Docs*. Memberikan pengenalan yang sangat baik tentang filosofi dan praktik memperlakukan dokumentasi seperti kode.

Bab 10: Etika dan Profesionalisme

Capaian Pembelajaran

Setelah menyelesaikan bab ini, mahasiswa diharapkan mampu:

- Menjelaskan pentingnya etika dan profesionalisme sebagai pilar fundamental dalam praktik analisis dan perancangan sistem informasi.
- Mengidentifikasi dan menganalisis dilema etis yang umum dihadapi oleh analis sistem, terutama yang berkaitan dengan privasi data, akurasi, kekayaan intelektual, dan akses.
- Membedakan berbagai jenis lisensi perangkat lunak, termasuk lisensi hak milik (*proprietary*), sumber terbuka (*open source*), dan variasi utamanya (misalnya, GPL, MIT), serta memahami implikasinya.
- Memahami dan menerapkan prinsip-prinsip utama dari kode etik profesional, seperti yang dikeluarkan oleh ACM (Association for Computing Machinery), dalam pengambilan keputusan.
- Menganalisis studi kasus nyata yang melibatkan kegagalan etis dalam pengembangan sistem dan merumuskan tindakan yang seharusnya diambil oleh seorang profesional yang bertanggung jawab.

Selamat datang di bab terakhir dari perjalanan kita dalam dunia Analisis dan Perancangan Sistem Informasi. Sejauh ini, kita telah membekali diri dengan berbagai keahlian teknis, mulai dari metodologi, pemodelan, perancangan, hingga pengujian. Kita telah belajar *bagaimana* membangun sistem. Namun, seorang profesional sejati tidak hanya bertanya "Bisakah kita membangun ini?", tetapi juga mengajukan pertanyaan yang lebih mendalam: "**Haruskah** kita membangun ini?" dan "Bagaimana kita membangunnya secara **bertanggung jawab**?"

Bab ini akan membawa kita ke ranah etika dan profesionalisme, sebuah dimensi yang seringkali menjadi pembeda antara seorang teknisi yang kompeten dan seorang arsitek solusi yang bijaksana. Sistem informasi yang kita bangun tidak beroperasi di ruang hampa; ia memiliki dampak nyata dan mendalam terhadap individu, organisasi, dan masyarakat luas. Sistem yang kita rancang dapat melindungi privasi atau melanggarnya, dapat mempromosikan keadilan atau memperkuat bias, dapat memberdayakan pengguna atau justru merugikan mereka.

Oleh karena itu, pemahaman yang kuat tentang tanggung jawab etis dan profesional bukanlah sebuah tambahan, melainkan inti dari praktik rekayasa sistem yang baik. Kita akan menjelajahi dilema yang akan Anda hadapi, memahami aturan main seputar hak cipta dan lisensi, serta membedah kasus-kasus nyata di mana keputusan etis menjadi titik penentu antara kesuksesan yang bertanggung jawab dan kegagalan yang merusak.

10.1 Etika dalam Analisis dan Perancangan

Sebagai seorang analis atau perancang sistem, Anda akan berada dalam posisi yang memiliki akses dan pengaruh yang signifikan. Anda akan berinteraksi dengan data sensitif, memengaruhi cara orang bekerja, dan membuat keputusan desain yang berdampak pada banyak pihak. Posisi ini menuntut kesadaran etis yang tinggi. Etika dalam konteks ini adalah tentang membuat pilihan yang benar dan adil ketika dihadapkan pada berbagai dilema. Beberapa area utama yang menjadi perhatian etis adalah privasi, akurasi, kekayaan intelektual, dan akses.

Privasi (Privacy)

Analis sistem seringkali memiliki akses ke informasi rahasia, mulai dari data pribadi karyawan, rekam medis pasien, hingga strategi bisnis perusahaan. Tanggung jawab etis yang utama adalah melindungi privasi dan menjaga kerahasiaan informasi ini.

- **Kewajiban:** Anda berkewajiban untuk tidak mengungkapkan informasi rahasia kepada pihak yang tidak berwenang, baik di dalam maupun di luar organisasi. Anda juga harus merancang sistem dengan prinsip *privacy by design*, yaitu memastikan bahwa mekanisme perlindungan privasi sudah tertanam dalam arsitektur sistem sejak awal, bukan sebagai tambahan di akhir. Ini termasuk enkripsi data, kontrol akses yang ketat, dan kebijakan anonimisasi data.
- **Dilema:** Anda mungkin diminta oleh manajemen untuk membangun fitur yang mengumpulkan lebih banyak data pribadi pengguna daripada yang sebenarnya dibutuhkan untuk fungsionalitas sistem. Secara etis, Anda harus

mempertanyakan justifikasi dari pengumpulan data tersebut dan mengadvokasi pendekatan yang paling tidak invasif terhadap privasi pengguna.

Akurasi (Accuracy)

Sistem informasi adalah pondasi pengambilan keputusan. Jika data yang disimpan atau informasi yang dihasilkan oleh sistem tidak akurat, keputusan yang salah dapat dibuat, yang berpotensi menyebabkan kerugian finansial atau bahkan membahayakan keselamatan manusia.

- **Kewajiban:** Anda memiliki tanggung jawab profesional untuk memastikan integritas dan akurasi data. Ini berarti merancang proses validasi input yang kuat, mekanisme pemeriksaan kesalahan, dan prosedur audit untuk menjaga agar data tetap bersih dan andal.
- **Dilema:** Anda menemukan bahwa data dari sistem lama yang akan dimigrasikan ke sistem baru memiliki tingkat kesalahan yang tinggi. Manajer proyek menekan Anda untuk melanjutkan migrasi agar sesuai jadwal. Secara etis, Anda harus menolak dan bersikeras untuk melakukan proses pembersihan data terlebih dahulu, meskipun itu berarti menunda proyek, karena meluncurkan sistem dengan data yang tidak akurat adalah tindakan yang tidak bertanggung jawab.

Kekayaan Intelektual (Property)

Kekayaan intelektual dalam konteks ini mencakup kode program, desain sistem, dokumentasi, dan data. Ini adalah aset berharga milik perusahaan atau klien.

- **Kewajiban:** Anda harus menghormati kepemilikan kekayaan intelektual. Kode atau desain yang Anda buat sebagai bagian dari pekerjaan Anda adalah milik perusahaan, bukan milik pribadi Anda. Menggunakan kembali kode dari proyek sebelumnya untuk klien baru tanpa izin adalah pelanggaran etika (dan mungkin hukum).
- **Dilema:** Anda pindah kerja ke perusahaan baru dan menyadari bahwa masalah yang sedang Anda hadapi dapat dengan mudah diselesaikan dengan menggunakan sebagian kecil kode yang pernah Anda tulis di perusahaan lama. Secara etis, Anda tidak boleh melakukannya. Anda harus mengembangkan solusi baru dari awal atau mencari alternatif lain yang sah.

Akses (Access)

Sistem yang Anda bangun dapat memengaruhi siapa yang memiliki akses terhadap informasi dan teknologi. Keputusan desain Anda dapat menciptakan atau justru menghilangkan hambatan.

- **Kewajiban:** Anda memiliki tanggung jawab untuk merancang sistem yang dapat diakses secara adil. Ini termasuk mempertimbangkan pengguna dengan disabilitas (misalnya, dengan memastikan kompatibilitas dengan pembaca layar) dan pengguna dengan literasi digital yang beragam. Selain itu, Anda harus memastikan bahwa mekanisme keamanan dan kontrol akses dirancang untuk mencegah penyalahgunaan, seperti pencurian identitas atau penipuan.
- **Dilema:** Untuk menghemat biaya, tim memutuskan untuk tidak mengimplementasikan fitur aksesibilitas untuk

pengguna tunanetra karena jumlah mereka dianggap kecil. Secara etis, Anda harus memperjuangkan inklusivitas dan mengingatkan tim bahwa mengabaikan kebutuhan sekelompok pengguna adalah bentuk diskriminasi.

10.2 Hak Cipta dan Lisensi Perangkat Lunak

Setiap baris kode yang ditulis adalah sebuah karya kreatif yang dilindungi oleh undang-undang hak cipta, sama seperti sebuah buku atau lagu. Memahami dasar-dasar hak cipta dan bagaimana perangkat lunak dilisensikan adalah kewajiban bagi setiap profesional TI. Lisensi adalah dokumen hukum yang memberikan izin kepada pengguna untuk menggunakan, memodifikasi, atau mendistribusikan perangkat lunak di bawah syarat dan ketentuan tertentu.

Secara umum, lisensi perangkat lunak dapat dibagi menjadi dua kategori besar: hak milik (*proprietary*) dan sumber terbuka (*open source*).

Lisensi Hak Milik (Proprietary License)

Ini adalah model lisensi yang paling tradisional.

- **Konsep:** Penerbit perangkat lunak memberikan pengguna hak untuk **menggunakan** satu atau lebih salinan perangkat lunak, tetapi kepemilikan perangkat lunak itu sendiri tetap pada penerbit. Pengguna tidak memiliki akses ke kode sumber (*source code*) dan dilarang keras untuk memodifikasi, merekayasa balik (*reverse engineering*), atau mendistribusikan ulang perangkat lunak tersebut.
- **Contoh:** Microsoft Windows, Adobe Photoshop, Microsoft Office.

Lisensi Sumber Terbuka (Open Source License)

Lisensi sumber terbuka didasarkan pada filosofi bahwa kode sumber harus tersedia untuk umum agar dapat dipelajari, dimodifikasi, dan ditingkatkan oleh siapa saja. Namun, "sumber terbuka" tidak berarti "tanpa aturan". Ada banyak jenis lisensi sumber terbuka, yang dapat dikelompokkan menjadi dua jenis utama:

1. **Lisensi Permisif (Permissive Licenses):**
    - **Konsep:** Lisensi ini memberikan kebebasan maksimal kepada pengguna. Anda dapat dengan bebas menggunakan, memodifikasi, dan mendistribusikan ulang kode, bahkan sebagai bagian dari produk komersial yang bersifat hak milik. Satu-satunya syarat utama biasanya adalah Anda harus tetap menyertakan pemberitahuan hak cipta asli.
    - **Contoh:** Lisensi MIT, Lisensi Apache 2.0, Lisensi BSD.
2. **Lisensi *Copyleft*:**
    - **Konsep:** Lisensi ini juga memungkinkan Anda untuk menggunakan, memodifikasi, dan mendistribusikan ulang kode secara bebas. Namun, ia memiliki satu syarat penting yang disebut "viral": setiap perangkat lunak turunan yang Anda buat dan distribusikan yang menggunakan kode berlisensi *copyleft* juga **harus** dilisensikan di bawah lisensi

copyleft yang sama. Ini memastikan bahwa kebebasan perangkat lunak tetap terjaga di semua versi turunannya.
    - **Contoh:** GNU General Public License (GPL).

Implikasi bagi Pengembang: Sebagai seorang analis atau pengembang, Anda harus sangat berhati-hati saat menggunakan pustaka atau komponen dari pihak ketiga dalam proyek Anda. Menggunakan komponen berlisensi GPL dalam produk komersial yang bersifat hak milik dapat secara hukum memaksa perusahaan Anda untuk membuka seluruh kode sumber produk tersebut. Oleh karena itu, audit lisensi adalah bagian penting dari profesionalisme dalam pengembangan sistem.

10.3 Tanggung Jawab Profesional

Profesionalisme melampaui sekedar keahlian teknis; ia mencakup komitmen terhadap standar perilaku, kompetensi, dan integritas yang tinggi. Banyak organisasi profesi di bidang komputasi, seperti **Association for Computing Machinery (ACM)** dan **IEEE Computer Society**, telah merumuskan kode etik untuk memandu para anggotanya.

Kode Etik dan Praktik Profesional Rekayasa Perangkat Lunak dari ACM/IEEE adalah salah satu yang paling dihormati. Kode ini didasarkan pada delapan prinsip utama yang merangkum tanggung jawab seorang profesional:

1. **PUBLIK:** Bertindak secara konsisten demi kepentingan publik.

2. **KLIEN DAN PEMBERI KERJA:** Bertindak dengan cara yang paling menguntungkan klien dan pemberi kerja, yang konsisten dengan kepentingan publik.
3. **PRODUK:** Memastikan bahwa produk dan modifikasinya memenuhi standar profesional tertinggi yang mungkin.
4. **PENILAIAN (JUDGMENT):** Menjaga integritas dan independensi dalam penilaian profesional.
5. **MANAJEMEN:** Manajer dan pemimpin rekayasa perangkat lunak harus berlangganan dan mempromosikan pendekatan etis dalam pengelolaan pengembangan dan pemeliharaan perangkat lunak.
6. **PROFESI:** Meningkatkan integritas dan reputasi profesi yang konsisten dengan kepentingan publik.
7. **KOLEGA:** Bersikap adil dan suportif terhadap kolega.
8. **DIRI SENDIRI (SELF):** Berpartisipasi dalam pembelajaran seumur hidup mengenai praktik profesi dan mempromosikan pendekatan etis dalam praktik tersebut.

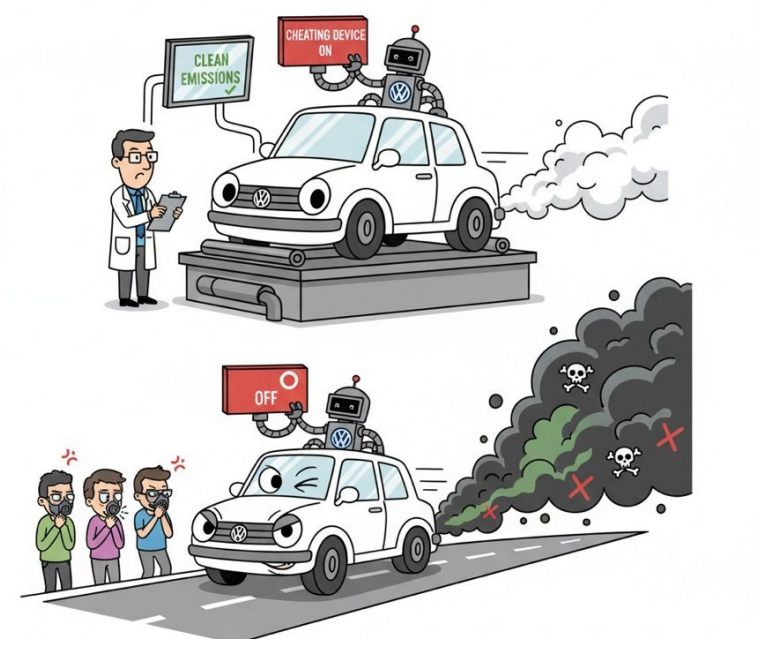
Prinsip-prinsip ini berfungsi sebagai kompas moral. Ketika menghadapi dilema, seorang profesional harus merujuk pada prinsip-prinsip ini untuk memandu keputusannya. Tanggung jawab profesional berarti memiliki keberanian untuk menolak permintaan yang tidak etis, mengakui kesalahan, terus belajar untuk menjaga kompetensi, dan selalu menempatkan dampak sosial dari pekerjaan di atas keuntungan pribadi atau tekanan jangka pendek.

10.4 Kasus-kasus dalam Etika Pengembangan Sistem

Mempelajari kasus-kasus nyata adalah cara terbaik untuk memahami bagaimana prinsip-prinsip etika diterapkan (atau dilanggar) dalam praktik.

Kasus 1: Skandal Emisi Volkswagen ("Dieselgate")

- **Situasi:** Pada tahun 2015, terungkap bahwa Volkswagen telah secara sengaja memprogram mesin diesel mereka dengan perangkat lunak yang dapat mendeteksi kapan mobil sedang diuji emisi. Selama pengujian, perangkat lunak akan mengaktifkan kontrol emisi penuh. Namun, dalam kondisi mengemudi normal, kontrol ini dimatikan, menyebabkan mobil mengeluarkan polutan nitrogen oksida hingga 40 kali di atas batas legal.
- **Pelanggaran Etis:** Ini adalah kasus penipuan yang disengaja dan masif. Para insinyur dan pemrogram yang terlibat melanggar hampir setiap prinsip etika profesional. Mereka tidak bertindak demi kepentingan **publik** (dengan menyebabkan polusi udara yang signifikan), tidak jujur kepada **klien** (pembeli mobil) dan regulator, dan menghasilkan **produk** yang secara fundamental cacat dan menipu. Kasus ini adalah contoh ekstrim dari konflik antara perintah atasan dan tanggung jawab profesional kepada masyarakat.

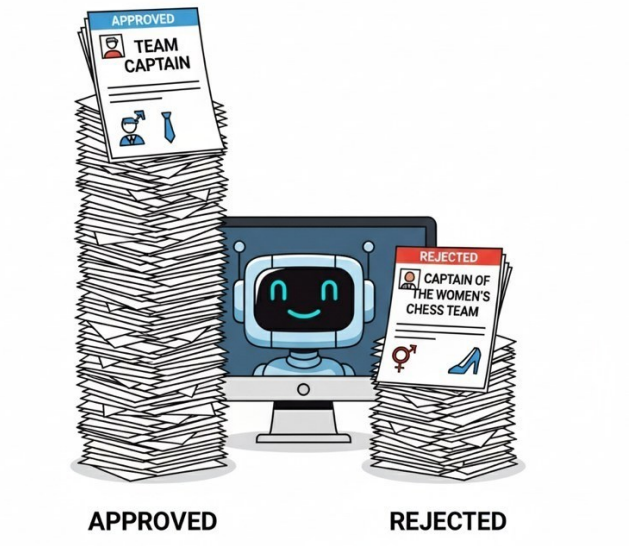


Gambar 19. Ilustrasi skandal emisi Volkswagen

Kasus 2: Alat Perekrutan Berbasis AI yang Bias Gender

- **Situasi:** Amazon mengembangkan sebuah sistem AI untuk menyaring resume kandidat. Tim melatih model tersebut menggunakan data resume yang telah diterima perusahaan selama 10 tahun terakhir. Karena sebagian besar resume di bidang teknis berasal dari laki-laki, sistem tersebut "belajar" bahwa kandidat laki-laki lebih disukai. AI tersebut mulai menghukum resume yang mengandung kata "wanita" (misalnya, "kapten tim catur wanita") dan memberikan skor lebih rendah pada lulusan dari dua universitas khusus wanita.
- **Pelanggaran Etis:** Meskipun tidak disengaja, ini adalah kegagalan etis yang serius. Tim gagal mempertimbangkan potensi **bias** dalam data historis mereka. Mereka melanggar

prinsip untuk **menghindari kerugian** dan memastikan sistem yang mereka bangun adil. Kasus ini menyoroti tanggung jawab baru yang muncul di era AI: memastikan bahwa sistem yang kita bangun tidak mengotomatiskan atau bahkan memperkuat ketidakadilan sosial yang ada. Amazon akhirnya membatalkan proyek ini.



Gambar 20. Ilustrasi Alat Perekrutan Berbasis AI yang Bias Gender

Kasus 3: Bencana Mesin Radioterapi Therac-25

- **Situasi:** Pada pertengahan 1980-an, sebuah mesin radioterapi yang dikendalikan oleh perangkat lunak, Therac-25, menyebabkan setidaknya enam kecelakaan di mana pasien menerima overdosis radiasi masif. Beberapa pasien meninggal akibat luka bakar radiasi. Investigasi menemukan bahwa kecelakaan tersebut disebabkan oleh *bug* perangkat lunak yang kompleks, terutama kondisi

balapan (*race condition*), yang terjadi ketika operator memasukkan data terlalu cepat.

- **Pelanggaran Etis:** Kasus ini adalah studi klasik tentang kegagalan dalam rekayasa perangkat lunak untuk sistem yang kritis terhadap keselamatan. Tim pengembang gagal melakukan **pengujian** yang memadai, mengabaikan laporan masalah awal, dan memiliki **desain** perangkat lunak yang tidak aman. Ini adalah pelanggaran berat terhadap tanggung jawab untuk menghasilkan **produk** berkualitas tinggi dan, yang terpenting, untuk tidak membahayakan **publik**.

Gambar 21. Ilustrasi Bencana Mesin Radioterapi Therac-25

Kasus-kasus ini memberikan pelajaran yang kuat: konsekuensi dari keputusan teknis dan etis bisa sangat besar. Sebagai calon profesional, adalah tugas Anda untuk belajar dari kesalahan ini dan berkomitmen pada standar integritas tertinggi dalam pekerjaan Anda.

Contoh Soal & Studi Kasus

Soal Konseptual (Esai Singkat)

1. Seorang analis sistem menemukan celah keamanan yang signifikan dalam sistem e-commerce yang akan segera diluncurkan. Manajernya memerintahkan untuk tetap meluncurkan sistem sesuai jadwal dan akan memperbaiki celah tersebut pada "pembaruan berikutnya" untuk menghindari penundaan. Menggunakan prinsip-prinsip dari kode etik ACM/IEEE, jelaskan mengapa perintah ini tidak etis dan apa tindakan yang seharusnya diambil oleh analis tersebut.
2. Perusahaan Anda sedang mengembangkan perangkat lunak pengolah kata baru. Tim pengembang ingin menggunakan pustaka penyunting teks sumber terbuka yang sangat bagus, tetapi pustaka tersebut dilisensikan di bawah GNU GPL. Perusahaan berencana untuk menjual perangkat lunak baru ini sebagai produk komersial dengan lisensi hak milik. Apa implikasi dari penggunaan pustaka berlisensi GPL tersebut? Apa saran Anda kepada tim?
3. Jelaskan perbedaan antara privasi data dan keamanan data. Mengapa keduanya penting dari sudut pandang etika?

Solusi dan Pembahasan Soal Konseptual

1. **Dilema Celah Keamanan:**

- **Analisis Etis:** Perintah manajer tersebut sangat tidak etis karena melanggar beberapa prinsip utama kode etik. Prinsip **1 (PUBLIK)** dan **2 (KLIEN)** dilanggar karena meluncurkan sistem dengan celah keamanan yang diketahui akan membahayakan data pelanggan dan merusak kepercayaan publik serta klien. Prinsip **3 (PRODUK)** dilanggar karena secara sadar merilis produk di bawah standar profesional. Prinsip **4 (PENILAIAN)** dilanggar karena analis diminta untuk mengabaikan penilaian profesionalnya demi jadwal.
- **Tindakan yang Seharusnya Diambil:** Analis tersebut memiliki kewajiban profesional untuk menolak perintah tersebut. Langkah pertama adalah menjelaskan kembali risiko yang ada kepada manajer secara jelas dan terdokumentasi (misalnya, melalui email), menyoroti potensi kerugian finansial dan reputasi bagi perusahaan. Jika manajer tetap bersikeras, analis harus melaporkan masalah ini ke tingkat manajemen yang lebih tinggi atau ke komite etika perusahaan, jika ada. Mengundurkan diri bisa menjadi pilihan terakhir jika perusahaan secara sadar memutuskan untuk melanjutkan tindakan yang tidak etis dan membahayakan tersebut.

2. **Implikasi Lisensi GPL:**
    - **Implikasi:** Lisensi GNU GPL bersifat *copyleft*. Ini berarti jika perusahaan Anda menggunakan pustaka berlisensi GPL dalam perangkat lunak

pengolah kata mereka dan kemudian mendistribusikan (menjual) perangkat lunak tersebut, mereka secara hukum **diwajibkan** untuk merilis seluruh kode sumber dari perangkat lunak pengolah kata mereka di bawah lisensi GPL juga. Hal ini akan menghancurkan model bisnis mereka yang ingin menjualnya sebagai produk hak milik.

- **Saran:** Saran saya kepada tim adalah untuk **tidak menggunakan pustaka berlisensi GPL tersebut**. Mereka harus mencari alternatif lain yang memiliki fungsionalitas serupa tetapi dilisensikan di bawah lisensi permisif (seperti MIT atau Apache), yang memungkinkan penggunaan dalam perangkat lunak komersial tanpa kewajiban untuk membuka kode sumber.

3. **Privasi vs. Keamanan Data:**
   - **Perbedaan: Keamanan data** adalah tentang **melindungi data dari akses yang tidak sah**. Ini adalah mekanisme teknis seperti enkripsi, firewall, dan kontrol akses. **Privasi data** adalah tentang **bagaimana data pribadi dikumpulkan, digunakan, dan dibagikan secara sah dan etis**. Ini adalah masalah kebijakan dan hak. Sebuah sistem bisa sangat aman (sulit diretas) tetapi sangat melanggar privasi (mengumpulkan dan menjual data pengguna tanpa izin).
   - **Pentingnya:** Keduanya penting. Keamanan yang buruk dapat menyebabkan pelanggaran privasi (misalnya, data pribadi dicuri). Namun, kebijakan privasi yang buruk adalah

pelanggaran etika itu sendiri, bahkan jika datanya aman. Profesional TI memiliki tanggung jawab etis untuk memastikan keduanya: membangun sistem yang **aman** untuk melindungi data dan merancang kebijakan yang **menghormati privasi** pengguna.

Studi Kasus: Algoritma Persetujuan Pinjaman "KreditCepat"

- **Konteks:** Sebuah perusahaan teknologi finansial (*fintech*), "KreditCepat", mengembangkan sistem baru untuk menyetujui atau menolak aplikasi pinjaman mikro secara otomatis. Untuk membangun model AI-nya, tim menggunakan data historis pinjaman dari bank konvensional selama 20 tahun terakhir.
- **Masalah:** Setelah sistem diluncurkan, seorang analis data junior di perusahaan tersebut menyadari sebuah pola yang mengkhawatirkan. Tingkat penolakan aplikasi dari lingkungan dengan kode pos mayoritas minoritas secara signifikan lebih tinggi daripada lingkungan lain, bahkan ketika faktor-faktor seperti pendapatan dan riwayat kredit pemohon serupa. Analis tersebut menduga bahwa data historis dari bank mungkin mengandung bias sistemik yang tidak disadari (misalnya, praktik *redlining* di masa lalu), dan AI telah "mempelajari" dan mengotomatiskan bias tersebut.
- **Dilema:** Analis tersebut melaporkan temuannya kepada manajer produk. Manajer tersebut khawatir. Mengakui masalah ini akan memerlukan penarikan produk, investigasi yang mahal, dan dapat menimbulkan publisitas negatif yang merusak reputasi perusahaan.

Manajer menyarankan untuk "memantau situasi" untuk saat ini dan tidak mengambil tindakan drastis.

Pertanyaan dan Analisis Studi Kasus

Pertanyaan: Anda adalah analis data junior tersebut. Apa tanggung jawab profesional dan etis Anda dalam situasi ini? Uraikan langkah-langkah yang akan Anda ambil, dengan mengacu pada prinsip-prinsip etika yang telah dibahas.

Analisis:

Ini adalah dilema etis yang kompleks yang menguji integritas profesional seorang analis. Analis tersebut memiliki tanggung jawab yang jelas berdasarkan prinsip-prinsip etika.

Tanggung Jawab Profesional dan Etis:

Tanggung jawab utama analis adalah pada **Prinsip 1: PUBLIK**. Sistem yang secara sistematis mendiskriminasi sekelompok orang berdasarkan lokasi tempat tinggal mereka (yang berkorelasi dengan ras atau etnis) menyebabkan kerugian sosial yang nyata. Membiarkan sistem ini beroperasi adalah tindakan yang tidak etis. Selain itu, **Prinsip 3: PRODUK** menuntut agar produk memenuhi standar profesional tertinggi, dan produk yang bias secara inheren adalah produk yang cacat. **Prinsip 2: KLIEN DAN PEMBERI KERJA** juga relevan; meskipun dalam jangka pendek manajer khawatir tentang publisitas negatif, dalam jangka panjang, skandal diskriminasi akan jauh lebih merusak reputasi dan bisnis perusahaan.

Langkah-langkah yang Harus Diambil:

1. **Dokumentasikan Semuanya Secara Objektif:** Langkah pertama adalah mendokumentasikan analisis secara

menyeluruh dan objektif. Buat laporan formal yang menunjukkan bukti statistik dari bias tersebut, metodologi yang digunakan untuk menemukannya, dan potensi dampaknya. Hindari bahasa yang emosional atau menuduh; fokus pada data.

2. **Komunikasi Formal dengan Manajer:** Kirimkan laporan tersebut kepada manajer produk melalui email, menciptakan jejak komunikasi tertulis. Dalam email tersebut, nyatakan kembali keprihatinan Anda secara profesional dan jelaskan mengapa "memantau situasi" tidak cukup karena sistem tersebut secara aktif menyebabkan potensi kerugian setiap hari. Sarankan untuk segera menonaktifkan fitur persetujuan otomatis dan membentuk tim untuk menginvestigasi dan memperbaiki model tersebut.
3. **Eskalasi jika Diperlukan:** Jika manajer produk tetap tidak mau mengambil tindakan, analis memiliki kewajiban etis untuk melakukan eskalasi. Rantai komando yang tepat bisa jadi adalah atasan dari manajer produk, departemen hukum/kepatuhan perusahaan, atau kepala bagian data/analitik. Menyajikan laporan berbasis data yang objektif akan memberikan bobot pada klaim Anda.
4. **Advokasi untuk Solusi:** Selain melaporkan masalah, seorang profesional yang proaktif juga akan menyarankan solusi. Ini bisa termasuk:
    - Melakukan audit bias pada set data pelatihan.
    - Menjelajahi teknik-teknik *fairness-aware machine learning* untuk melatih ulang model.

- Menyarankan agar keputusan penolakan dari AI selalu ditinjau oleh manusia sebelum final.

5. **Perlindungan Diri (Whistleblowing sebagai Opsi Terakhir):** Jika semua saluran internal gagal dan perusahaan secara sadar memilih untuk melanjutkan praktik yang diskriminatif, analis tersebut menghadapi pilihan yang sulit. Melaporkan masalah ini ke regulator atau media (*whistleblowing*) adalah langkah ekstrim yang memiliki risiko pribadi yang signifikan. Namun, dalam kasus di mana kerugian publik yang ditimbulkan sangat besar, kode etik menyiratkan bahwa tanggung jawab kepada masyarakat pada akhirnya mengalahkan loyalitas kepada pemberi kerja.

Dalam kasus ini, tindakan yang paling profesional dan etis adalah tidak diam. Analis harus bertindak sebagai suara hati nurani teknis perusahaan, menggunakan keahliannya untuk mengidentifikasi dan mengadvokasi perbaikan masalah yang memiliki dampak sosial yang serius.

Daftar Bahan Bacaan

Buku Teks dan Referensi Utama

- Reynolds, G. (Edisi terbaru). *Ethics in Information Technology*. Cengage Learning. (Buku teks komprehensif yang didedikasikan untuk topik ini).
- Baase, S. (Edisi terbaru). *A Gift of Fire: Social, Legal, and Ethical Issues for Computing and the Internet*. Pearson. (Referensi klasik yang membahas dampak sosial dan etis dari teknologi komputasi).

Kode Etik Profesional (Wajib Dibaca)

- ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices. (1999). *Software Engineering Code of Ethics and Professional Practice*. Tersedia di situs web ACM dan IEEE.
- Association for Computing Machinery (ACM). (2018). *ACM Code of Ethics and Professional Conduct*. Tersedia di https://www.acm.org/code-of-ethics.

Bacaan Tambahan (Dianjurkan)

- O'Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown. (Buku yang sangat mudah diakses yang membahas kasus-kasus nyata tentang bagaimana algoritma dapat memperkuat bias dan menyebabkan kerugian).
- Stallman, R. & Free Software Foundation. *The GNU General Public License (GPL)*. Tersedia di https://www.gnu.org/licenses/gpl-3.0.en.html. (Membaca langsung teks lisensi seperti GPL akan memberikan pemahaman yang lebih baik tentang konsep *copyleft*).

Penutup

Selamat, Anda telah sampai di akhir perjalanan kita dalam mempelajari Analisis dan Perancangan Sistem Informasi. Selama sepuluh bab terakhir, kita telah menempuh sebuah lintasan pengetahuan yang dirancang untuk mengubah Anda dari seorang mahasiswa menjadi seorang pemikir sistem, seorang arsitek solusi digital yang siap menghadapi tantangan di dunia nyata. Buku ajar ini telah membekali Anda dengan pondasi konseptual, metodologi terstruktur, dan perangkat teknis yang menjadi inti dari profesi analis sistem.

Mari kita sejenak menengok kembali jejak perjalanan yang telah kita lalui bersama, merajut kembali benang-benang pengetahuan dari setiap bab menjadi sebuah pemahaman yang utuh dan kohesif.

Pada **Bab 1**, kita memulai dengan membangun pondasi, mendefinisikan pilar-pilar utama seperti data, informasi, dan sistem informasi. Kita memahami bahwa sistem informasi adalah sebuah ekosistem sosio-teknis yang terdiri dari lima komponen vital: perangkat keras, perangkat lunak, data, manusia, dan proses. Perjalanan historis dari era *mainframe* hingga era kecerdasan buatan saat ini menunjukkan betapa sistem informasi telah berevolusi dari sekedar alat bantu hitung menjadi jantung strategis organisasi modern.

Selanjutnya, di **Bab 2**, kita memperdalam pemahaman tentang konsep sistem itu sendiri. Kita membedah anatomi universal sebuah sistem dengan mengidentifikasi delapan

karakteristiknya, mulai dari komponen hingga tujuan. Kita belajar mengklasifikasikan sistem, abstrak atau fisik, alamiah atau buatan, deterministik atau probabilistik, dan yang terpenting, kita memahami perbedaan krusial antara sistem terbuka yang dinamis dan sistem tertutup yang rentan terhadap entropi. Konsep ini mengajarkan kita untuk melihat organisasi sebagai entitas hidup yang harus terus beradaptasi dengan lingkungannya.

Dengan pemahaman konseptual yang kokoh, kita beralih ke aspek praktis dalam **Bab 3** dengan mempelajari berbagai metodologi pengembangan sistem. Kita melihat pendekatan klasik yang sekuensial dan terstruktur dalam model **Waterfall**, pendekatan evolusioner yang berfokus pada risiko dalam model **Spiral**, hingga revolusi fleksibilitas dan kolaborasi yang dibawa oleh filosofi **Agile** beserta kerangka kerja populernya seperti Scrum dan XP. Inti dari bab ini adalah kearifan untuk memilih peta jalan yang tepat untuk setiap perjalanan proyek yang unik.

Bab 4 membekali kita dengan perangkat investigasi seorang analis sistem. Kita belajar seni dan ilmu dalam **pengumpulan kebutuhan**, sebuah fase yang sangat menentukan keberhasilan proyek. Kita menguasai teknik-teknik kunci seperti **wawancara** untuk menggali informasi mendalam, **observasi** untuk memahami realitas di lapangan, **kuisioner** untuk menjangkau audiens yang luas, **workshop** untuk membangun konsensus secara efisien, dan **analisis dokumen** untuk mendapatkan konteks awal.

Setelah bahan mentah terkumpul, **Bab 5** mengajarkan kita cara menganalisis dan menerjemahkannya menjadi model yang terstruktur. Di sinilah kita belajar memilah kebutuhan menjadi **fungsional** ("apa" yang dilakukan sistem) dan **non-fungsional** ("bagaimana" sistem melakukannya). Kita kemudian menguasai dua alat pemodelan fundamental: **Diagram Alir Data (DFD)** untuk memetakan proses dan aliran data, serta **Entity Relationship Diagram (ERD)** untuk merancang struktur data. Semua temuan ini kita tuangkan dalam **Spesifikasi Kebutuhan Sistem (SRS)**, dokumen yang menjadi kontrak dan sumber kebenaran proyek.

Dari analisis "apa", kita beralih ke desain "bagaimana" di **Bab 6**. Bab ini adalah fase arsitektural di mana kita merancang cetak biru sistem. Kita membahas empat pilar desain: **desain antarmuka pengguna (UI/UX)** untuk memastikan sistem mudah digunakan, **desain basis data** dengan mengubah ERD menjadi skema tabel yang ternormalisasi, **desain arsitektur sistem** untuk menentukan struktur keseluruhan, dan **desain proses bisnis** yang merinci setiap logika dan aturan keputusan.

Bab 7 membawa kita ke fase konstruksi, yaitu **implementasi sistem**. Di sini, desain diwujudkan menjadi kode yang berfungsi. Kita mempelajari faktor-faktor dalam memilih **bahasa pemrograman dan teknologi** yang tepat, peran strategis **prototipe** dalam memvalidasi ide, berbagai strategi untuk **mengintegrasikan modul-modul** perangkat lunak, dan langkah-langkah teknis untuk **mengimplementasikan basis data** menggunakan SQL.

Sebuah sistem yang telah dibangun harus dipastikan kualitasnya. **Bab 8** memperkenalkan kita pada disiplin **pengujian sistem**. Kita memahami pentingnya pengujian sebagai jaring pengaman kualitas dan menjelajahi berbagai tingkatannya, mulai dari **pengujian unit** pada level kode terkecil, **pengujian integrasi** untuk memastikan modul bekerja sama, **pengujian sistem** secara keseluruhan, hingga tahap akhir yang krusial: **Uji Penerimaan Pengguna (UAT)**, di mana pengguna memberikan persetujuan akhir.

Pada **Bab 9**, kita membahas aktivitas yang sering diabaikan namun sangat vital: **dokumentasi sistem**. Kita belajar membedakan antara **dokumentasi teknis** untuk pengembang dan **dokumentasi pengguna** untuk operator sistem. Kita juga menekankan pentingnya standar penulisan dan, yang terpenting, strategi untuk **memelihara dokumentasi** agar tetap hidup dan relevan, mencegahnya dari "pembusukan" yang dapat membuatnya tidak berguna.

Terakhir, di **Bab 10**, kita menutup perjalanan teknis kita dengan refleksi tentang aspek manusia dan profesionalisme. Kita membahas **etika dan tanggung jawab** seorang analis sistem, menavigasi isu-isu seperti hak cipta dan lisensi perangkat lunak, serta memahami bahwa menjadi seorang profesional TI bukan hanya tentang keahlian teknis, tetapi juga tentang integritas, tanggung jawab, dan kesadaran akan dampak sosial dari teknologi yang kita bangun.

Kini, Anda telah memiliki gambaran utuh dari siklus hidup pengembangan sistem informasi. Anda telah belajar untuk berpikir secara sistematis, menganalisis masalah secara

mendalam, merancang solusi yang elegan, dan memahami proses untuk mewujudkannya. Pengetahuan yang Anda peroleh dari buku ini adalah pondasi yang kuat. Namun, ingatlah bahwa dunia teknologi terus bergerak dan berevolusi. Anggaplah buku ini bukan sebagai akhir dari pembelajaran Anda, melainkan sebagai titik awal dari sebuah perjalanan seumur hidup untuk terus belajar, beradaptasi, dan berinovasi.

Selamat berkarya sebagai arsitek masa depan digital!

Suplemen

Lima Studi Kasus Komprehensif

Pendahuluan

Tujuan dan Ruang Lingkup Suplemen

Analisis dan perancangan sistem informasi adalah disiplin ilmu yang fundamental dalam pengembangan solusi teknologi untuk menjawab tantangan bisnis modern. Namun, mahasiswa sering kali menghadapi kesulitan dalam menghubungkan konsep-konsep teoretis yang dipelajari di kelas dengan kompleksitas masalah yang terjadi di dunia nyata. Suplemen buku ajar ini dirancang secara spesifik untuk menjembatani kesenjangan tersebut. Tujuannya adalah untuk menyediakan materi pembelajaran yang kaya akan konteks praktis melalui penyajian lima studi kasus komprehensif.

Setiap studi kasus dalam suplemen ini disusun untuk mensimulasikan siklus hidup pengembangan sistem (System Development Life Cycle - SDLC) secara utuh, mulai dari identifikasi masalah dan inisiasi proyek hingga perencanaan implementasi dan strategi pemeliharaan jangka panjang. Kelima kasus ini secara sengaja dirancang untuk mencakup berbagai jenis industri, skala organisasi, dan tantangan bisnis yang berbeda. Pendekatan ini memungkinkan pembaca untuk melihat bagaimana prinsip-prinsip analisis dan perancangan yang sama dapat diaplikasikan secara fleksibel dalam konteks yang beragam.

Untuk memastikan relevansi akademis, setiap tahapan dalam studi kasus akan secara eksplisit merujuk pada konsep-konsep inti yang diajarkan dalam kurikulum standar analisis dan perancangan sistem informasi, sebagaimana yang ditemukan dalam buku ajar terkemuka seperti *Systems Analysis and Design in a Changing World* oleh Satzinger, Jackson, dan Burd, serta *Systems Analysis and Design* oleh Kendall dan Kendall. Dengan demikian, suplemen ini berfungsi sebagai pelengkap yang ideal, memberikan "laboratorium" virtual di mana teori dapat diuji dan dipahami melalui aplikasi.

Kerangka Kerja Konseptual

Di balik penyajian kelima studi kasus ini, terdapat sebuah kerangka kerja konseptual yang menjadi benang merah pemersatu. Kerangka ini dibangun di atas beberapa pemahaman mendalam yang melampaui sekedar penerapan teknis, dan bertujuan untuk menanamkan wawasan strategis kepada pembaca.

Pertama, suplemen ini akan menunjukkan bahwa **pemilihan metodologi pengembangan bukanlah sekedar pilihan teknis, melainkan sebuah respons strategis terhadap tingkat ketidakpastian lingkungan proyek**. Berbagai laporan industri secara konsisten menunjukkan bahwa kegagalan proyek sering kali bukan disebabkan oleh kegagalan teknis, melainkan oleh faktor-faktor seperti perubahan kebutuhan yang tidak terkelola dan kurangnya keterlibatan pengguna.[10] Setiap kasus akan mendemonstrasikan bagaimana analisis cermat terhadap stabilitas kebutuhan, kompleksitas sistem, dan tingkat risiko akan menuntun pada pemilihan metodologi yang paling sesuai,

baik itu pendekatan prediktif seperti Waterfall maupun adaptif seperti Agile, sebagai alat mitigasi risiko yang paling efektif.

Kedua, akan dieksplorasi **hubungan simbiosis antara siklus hidup sistem (System Life Cycle) dan siklus hidup organisasi (Organizational Life Cycle)**. Sebuah sistem informasi tidak beroperasi dalam ruang hampa; ia ada untuk melayani organisasi yang dinamis, yang juga bergerak melalui tahapan kelahiran, pertumbuhan, kedewasaan, dan penurunan. Sistem yang dirancang untuk sebuah startup di fase pertumbuhan mungkin menjadi penghambat di fase kedewasaan jika tidak berevolusi. Konsep entropi, kecenderungan alami menuju ketidakteraturan, akan digunakan untuk menjelaskan mengapa sistem yang tidak dipelihara dan disesuaikan dengan kebutuhan organisasi yang berubah akan mengalami degradasi fungsional. Dengan demikian, pemeliharaan sistem akan dibingkai bukan hanya sebagai aktivitas perbaikan bug, tetapi sebagai upaya strategis untuk menyuntikkan keteraturan baru (negentropi) agar sistem tetap relevan.

Ketiga, suplemen ini akan menyajikan **evolusi sistem informasi sebagai cerminan dari evolusi era teknologi**. Kelima studi kasus ini sengaja disusun secara tematis untuk merepresentasikan paradigma teknologi yang berbeda: mulai dari era sistem terstruktur berbasis client-server, revolusi PC yang melahirkan *Decision Support Systems* (DSS), kemunculan internet yang memfasilitasi e-commerce, hingga era modern yang didominasi oleh komputasi awan (cloud), perangkat mobile, dan kecerdasan buatan (AI). Dengan menempatkan setiap kasus dalam konteks teknologinya, pembaca akan memahami bukan hanya *bagaimana* sebuah sistem dirancang,

tetapi *mengapa* ia dirancang dengan cara tertentu, sebagai produk dari batasan dan peluang yang ditawarkan oleh teknologi pada masanya.

Melalui ketiga lensa konseptual ini, suplemen ini diharapkan dapat memberikan pemahaman yang lebih dalam, bernuansa, dan strategis mengenai disiplin analisis dan perancangan sistem informasi.

Studi Kasus

Studi Kasus 1: Sistem Manajemen Rantai Pasok (SCM) untuk PT. Manufaktur Jaya (Paradigma Sistem Terstruktur dan Prediktif)

1.1 Latar Belakang dan Konteks Sistem (Aplikasi Bab 1: Konsep Sistem)

Profil Organisasi

PT. Manufaktur Jaya adalah sebuah perusahaan manufaktur furnitur skala menengah yang telah beroperasi selama lebih dari 20 tahun. Perusahaan ini memproduksi berbagai jenis perabotan kayu, seperti meja, kursi, dan lemari, untuk pasar domestik. Berdasarkan siklus hidup organisasi, PT. Manufaktur Jaya berada pada tahap "Kematangan" (Maturity). Pada tahap ini, fokus utama perusahaan bukanlah lagi pada pertumbuhan eksplosif, melainkan pada optimalisasi proses, efisiensi operasional, dan pengendalian biaya untuk mempertahankan profitabilitas dan pangsa pasar yang sudah ada. Struktur organisasi relatif stabil, dan proses bisnis telah terdefinisi dengan baik meskipun belum didukung oleh teknologi informasi yang memadai.

Analisis Sistem Saat Ini

Sistem manajemen rantai pasok yang berjalan di PT. Manufaktur Jaya saat ini dapat dianalisis menggunakan teori sistem. Sistem ini merupakan sistem terbuka (open system), karena secara konstan berinteraksi dengan lingkungannya,

menerima masukan dari pemasok dan mengirimkan keluaran kepada pelanggan. Namun, proses internalnya cenderung bersifat deterministik (deterministic system), di mana alur kerja dari pemesanan bahan baku hingga produksi barang jadi mengikuti urutan langkah yang dapat diprediksi dan terstandarisasi.

Karakteristik sistem yang ada dapat diuraikan sebagai berikut:

- **Komponen (Component):** Departemen Pembelian, Gudang Bahan Baku, Lantai Produksi, Gudang Barang Jadi, dan Departemen Penjualan. Setiap departemen berfungsi sebagai subsistem.
- **Batasan (Boundary):** Batasan sistem ini adalah operasional internal perusahaan, mulai dari interaksi dengan pemasok hingga pengiriman produk ke distributor.
- **Lingkungan Luar (Environment):** Pemasok kayu, distributor furnitur, pelanggan akhir, dan regulasi pemerintah terkait industri manufaktur.
- **Masukan (Input):** Pesanan pembelian dari pelanggan, daftar harga dari pemasok, faktur bahan baku, dan data penerimaan barang.
- **Proses (Process):** Verifikasi pesanan, pembuatan *purchase order* (PO) ke pemasok, penerimaan dan inspeksi bahan baku, proses produksi (pemotongan, perakitan, finishing), dan pengemasan produk jadi.
- **Keluaran (Output):** Produk furnitur jadi, faktur penjualan, laporan stok bulanan, dan surat jalan pengiriman.
- **Tujuan (Objective):** Memenuhi pesanan pelanggan secara tepat waktu dengan biaya produksi yang efisien.

Identifikasi Masalah

Meskipun proses bisnisnya sudah mapan, ketergantungan pada sistem manual menjadi sumber utama inefisiensi. Departemen pembelian dan gudang sangat bergantung pada spreadsheet yang terpisah dan dokumen fisik seperti formulir permintaan barang dan kartu stok. Kurangnya integrasi data antar subsistem ini menyebabkan serangkaian masalah operasional yang signifikan. Fenomena ini merupakan manifestasi dari entropi sistem, yaitu meningkatnya ketidakteraturan dan menurunnya efisiensi karena ketiadaan mekanisme kontrol dan informasi yang terintegrasi. Masalah-masalah spesifik yang teridentifikasi antara lain:

- **Keterlambatan Pengadaan Bahan Baku:** Informasi level stok di gudang tidak *real-time*. Departemen pembelian sering kali baru mengetahui bahwa stok bahan baku kritis telah menipis setelah adanya permintaan dari bagian produksi, menyebabkan keterlambatan dalam pemesanan dan potensi terhentinya produksi.
- **Kelebihan Stok (Overstocking):** Karena ketidakpastian data stok, departemen pembelian cenderung melakukan pemesanan dalam jumlah besar untuk "berjaga-jaga", yang mengakibatkan biaya penyimpanan (carrying cost) yang tinggi dan risiko kerusakan bahan baku.
- **Kekurangan Stok (Stockouts):** Di sisi lain, untuk produk yang permintaannya fluktuatif, sering terjadi kekurangan stok barang jadi, yang berujung pada kehilangan potensi penjualan dan menurunnya kepuasan pelanggan.

- **Kesulitan Pelacakan Pesanan:** Manajemen tidak memiliki visibilitas yang jelas terhadap status pesanan, baik pesanan bahan baku ke pemasok maupun pesanan produk dari pelanggan, sehingga sulit untuk memberikan estimasi waktu pengiriman yang akurat.

1.2 Inisiasi dan Perancangan Proyek (Aplikasi Bab 2: Peran Analis & Manajemen Proyek)

Pemicu Proyek

Proyek pengembangan sistem informasi ini dipicu oleh kebutuhan mendesak untuk meningkatkan performa sistem operasional secara keseluruhan.15 Manajemen puncak menyadari bahwa inefisiensi dalam rantai pasok telah menggerus margin keuntungan dan menurunkan daya saing perusahaan. Oleh karena itu, seorang analis sistem eksternal direkrut untuk melakukan investigasi awal dan merumuskan proposal solusi berbasis teknologi.

Analisis Kelayakan (PIECES Framework)

Analis sistem menggunakan kerangka kerja PIECES untuk menganalisis masalah dan memvalidasi kebutuhan akan sistem baru. Hasil analisisnya adalah sebagai berikut:

- **Performance:** Waktu siklus pemenuhan pesanan (dari pesanan diterima hingga barang dikirim) terlalu lama, rata-rata 25 hari, sementara kompetitor mampu melakukannya dalam 15 hari. Volume transaksi yang meningkat membuat sistem manual semakin kewalahan.

- **Information:** Data inventaris sering kali tidak akurat. Laporan stok yang dihasilkan bersifat periodik (bulanan) dan tidak mencerminkan kondisi *real-time*, sehingga tidak dapat diandalkan untuk pengambilan keputusan harian.
- **Economy:** Biaya penyimpanan untuk kelebihan stok diperkirakan mencapai 15% dari total biaya operasional. Di sisi lain, estimasi kerugian penjualan akibat kekurangan stok mencapai ratusan juta rupiah per tahun.
- **Control:** Tidak ada kontrol yang terpusat dan terotomatisasi atas alur material. Pelacakan barang dari penerimaan hingga pengiriman sangat sulit dilakukan, meningkatkan risiko kehilangan dan kesalahan pencatatan.
- **Efficiency:** Staf gudang dan pembelian menghabiskan sekitar 30% waktu kerja mereka untuk tugas-tugas administratif manual, seperti mencocokkan dokumen dan memperbarui spreadsheet, yang sebenarnya dapat diotomatisasi.
- **Service:** Ketidakmampuan memberikan informasi status pesanan yang akurat dan seringnya terjadi keterlambatan pengiriman telah menyebabkan peningkatan keluhan dari pelanggan dan distributor.

Penentuan Ruang Lingkup dan Tujuan

Berdasarkan analisis kelayakan, proyek ini disetujui untuk dilanjutkan. Tujuan utama proyek ini adalah merancang dan mengimplementasikan sebuah Sistem Informasi Manajemen Logistik (SIM Logistik) yang terintegrasi.16 Ruang lingkup

sistem akan mencakup tiga modul utama: manajemen pengadaan (procurement), manajemen inventaris (inventory), dan manajemen pemenuhan pesanan (order fulfillment). Sistem ini diharapkan dapat memberikan data yang akurat dan *real-time*, mengotomatisasi proses-proses kunci, dan menyediakan laporan manajerial untuk mendukung pengambilan keputusan yang lebih baik.

1.3 Pemilihan Metodologi Pengembangan (Aplikasi Bab 3: Metodologi Pengembangan)

Analisis Faktor Pemilihan

Pemilihan metodologi pengembangan sistem adalah keputusan krusial yang akan menentukan alur kerja, tingkat fleksibilitas, dan cara pengelolaan proyek. Untuk PT. Manufaktur Jaya, analis sistem menggunakan kerangka kerja pemilihan metodologi yang mempertimbangkan beberapa faktor kunci:

- **Kejelasan Kebutuhan Pengguna (Clarity of User Requirements): Tinggi.** Proses bisnis inti PT. Manufaktur Jaya (pengadaan, produksi, penjualan) sudah sangat mapan, terstruktur, dan tidak sering berubah. Pengguna (staf gudang, staf pembelian) dapat dengan jelas mendefinisikan apa yang mereka butuhkan dari sistem baru, yaitu otomatisasi dari proses yang sudah ada.
- **Penguasaan Teknologi (Familiarity with Technology): Menengah.** Tim IT internal perusahaan memiliki kompetensi dalam pengembangan aplikasi web standar dan manajemen basis data relasional, namun belum

memiliki pengalaman dengan teknologi baru atau metodologi pengembangan yang kompleks.

- **Tingkat Kerumitan Sistem (System Complexity): Menengah.** Sistem yang akan dibangun memerlukan integrasi antar modul (pembelian, inventaris, penjualan), namun tidak melibatkan algoritma yang sangat rumit atau teknologi yang belum teruji.
- **Tingkat Keandalan Sistem (System Reliability): Sangat Tinggi.** Sistem ini akan menjadi tulang punggung operasional perusahaan. Kegagalan sistem dapat menyebabkan terhentinya seluruh aktivitas bisnis, sehingga keandalan dan stabilitas adalah prioritas utama.
- **Jadwal Waktu Pelaksanaan (Short Time Schedules): Fleksibel.** Manajemen lebih memprioritaskan kualitas dan keandalan sistem daripada kecepatan penyelesaian proyek. Jadwal yang terprediksi lebih diutamakan daripada peluncuran cepat.

Justifikasi Pemilihan Metodologi Waterfall

Berdasarkan analisis faktor-faktor di atas, Model Waterfall dipilih sebagai metodologi pengembangan yang paling tepat. Alasan utamanya adalah keselarasan antara karakteristik model Waterfall dengan kondisi proyek di PT. Manufaktur Jaya:

1. **Sifat Sekuensial dan Terstruktur:** Pendekatan Waterfall yang linear dan bertahap (analisis -> desain -> implementasi -> pengujian) sangat cocok untuk proyek di mana kebutuhan sudah dipahami dengan baik di

awal. Hal ini meminimalkan risiko kesalahpahaman dan pengerjaan ulang yang mahal.

2. **Penekanan pada Dokumentasi:** Waterfall menuntut adanya dokumentasi yang lengkap di setiap akhir fase (misalnya, Dokumen Spesifikasi Kebutuhan, Dokumen Desain Sistem). Ini sangat penting untuk memastikan kualitas, menjadi panduan bagi tim pengembang, dan mempermudah proses pemeliharaan sistem di masa depan.
3. **Prediktabilitas dan Kontrol:** Alur kerja yang jelas memungkinkan estimasi waktu dan biaya yang lebih akurat serta memberikan kontrol yang ketat terhadap ruang lingkup proyek, yang sesuai dengan preferensi manajemen PT. Manufaktur Jaya.

Metodologi lain seperti Agile atau Spiral dinilai kurang sesuai untuk konteks ini. Metodologi Agile, yang dirancang untuk mengakomodasi perubahan, tidak terlalu diperlukan karena kebutuhan bisnis relatif stabil. Sementara itu, Model Spiral, yang berfokus pada manajemen risiko intensif, akan menjadi terlalu kompleks dan memakan waktu untuk proyek dengan tingkat risiko teknis yang dapat dikelola seperti ini. Keputusan ini menunjukkan bagaimana pemilihan metodologi merupakan sebuah respons strategis terhadap karakteristik unik dari sebuah proyek dan organisasinya.

1.4 Analisis Kebutuhan Sistem (Aplikasi Bab 4: Analisis Kebutuhan)

Tahap analisis kebutuhan adalah pondasi dari seluruh proyek dalam model Waterfall. Tujuannya adalah untuk memahami dan mendokumentasikan secara detail apa yang harus

dilakukan oleh sistem. Kesalahan atau kelalaian pada tahap ini akan berdampak besar pada fase-fase berikutnya dan merupakan salah satu penyebab utama kegagalan proyek.[10]
Teknik Pengumpulan Kebutuhan

Analis sistem menggunakan kombinasi beberapa teknik untuk mengumpulkan informasi secara komprehensif:

1. **Wawancara:** Sesi wawancara terstruktur dilakukan dengan para pemangku kepentingan utama, termasuk Kepala Departemen Pembelian, Manajer Gudang, dan Manajer Penjualan. Tujuannya adalah untuk memahami tujuan, masalah, dan ekspektasi mereka terhadap sistem baru.
2. **Observasi:** Analis sistem menghabiskan beberapa hari di gudang dan departemen pembelian untuk mengamati alur kerja secara langsung. Observasi ini membantu mengidentifikasi inefisiensi dan langkah-langkah proses yang mungkin tidak terungkap selama wawancara.
3. **Analisis Dokumen:** Analis mengumpulkan dan mempelajari dokumen-dokumen yang digunakan saat ini, seperti formulir permintaan barang, kartu stok manual, *purchase order*, dan laporan stok bulanan. Analisis ini memberikan pemahaman mendalam tentang data apa yang penting dan bagaimana data tersebut mengalir dalam organisasi.

Dokumentasi Kebutuhan

Hasil dari pengumpulan data tersebut kemudian diorganisir dan didokumentasikan ke dalam daftar kebutuhan fungsional dan non-fungsional. Dokumen ini, yang sering disebut Software Requirement Specification (SRS), menjadi acuan utama bagi seluruh tim proyek.

Tabel 1.1: Kebutuhan Fungsional dan Non-Fungsional SCM PT. Manufaktur Jaya

ID	Kategori	Jenis	Deskripsi Kebutuhan	Prioritas
KF-01	Manajemen Pemasok	Fungsional	Sistem harus dapat menyimpan dan mengelola data pemasok, termasuk nama, alamat, kontak, dan daftar produk yang mereka suplai.	Tinggi
KF-02	Manajemen Pembelian	Fungsional	Sistem harus dapat membuat, mencetak, dan melacak status *Purchase Order* (PO) dari dibuat, disetujui, dikirim ke pemasok, hingga barang diterima.	Tinggi
KF-03	Manajemen Penerimaan	Fungsional	Sistem harus dapat mencatat penerimaan barang dari pemasok, memvalidasi kuantitas terhadap PO, dan secara otomatis memperbarui jumlah stok.	Tinggi
KF-04	Manajemen Inventaris	Fungsional	Sistem harus dapat melacak kuantitas stok untuk setiap item bahan baku dan produk jadi secara *real-time*.	Tinggi
KF-05	Manajemen Inventaris	Fungsional	Sistem harus dapat menetapkan level stok minimum untuk setiap item dan memberikan notifikasi otomatis kepada departemen	Tinggi

ID	Kategori	Jenis	Deskripsi Kebutuhan	Prioritas
			pembelian ketika stok mencapai level tersebut.	
KF-06	Manajemen Penjualan	Fungsional	Sistem harus dapat mencatat pesanan penjualan dari pelanggan dan secara otomatis mengurangi stok barang jadi yang tersedia.	Sedang
KF-07	Pelaporan	Fungsional	Sistem harus dapat menghasilkan laporan stok terkini, laporan riwayat pembelian per pemasok, dan laporan penjualan per produk dalam periode tertentu.	Tinggi
KNF-01	Keamanan	Non-Fungsional	Akses ke sistem harus dibatasi berdasarkan peran pengguna (misalnya, staf gudang hanya bisa melihat dan mengupdate stok, manajer pembelian bisa membuat dan menyetujui PO).	Tinggi
KNF-02	Performa	Non-Fungsional	Waktu respons untuk semua transaksi dan permintaan data di dalam sistem tidak boleh lebih dari 3 detik.	Tinggi
KNF-03	Keandalan	Non-Fungsional	Sistem harus memiliki *uptime* (waktu operasional) minimal 99.5% selama jam kerja.	Tinggi
KNF-04	Skalabilitas	Non-Fungsional	Arsitektur sistem harus mampu menangani peningkatan volume transaksi hingga 20% per tahun tanpa degradasi performa yang signifikan.	Sedang

Tabel ini menjadi artefak krusial yang mengubah diskusi kualitatif menjadi spesifikasi yang konkret dan dapat diverifikasi. Ini berfungsi sebagai "kontrak" antara pemangku kepentingan dan tim pengembang, memastikan pemahaman yang sama dan menjadi dasar untuk semua aktivitas desain, implementasi, dan pengujian selanjutnya.

1.5 Pemodelan Proses Bisnis (Aplikasi Bab 5: Pemodelan Proses)

Setelah kebutuhan sistem didefinisikan, langkah selanjutnya adalah memodelkan bagaimana sistem akan bekerja dan bagaimana data akan mengalir di dalamnya. Untuk ini, digunakan alat pemodelan visual seperti Diagram Alir Data (DFD) dan Diagram Aktivitas UML.

Diagram Alir Data (Data Flow Diagram - DFD)

DFD digunakan untuk menggambarkan alur data melalui sistem dan proses-proses yang mentransformasikan data tersebut.3

- **DFD Konteks (Level 0):** Diagram ini memberikan pandangan tingkat tertinggi dari sistem. Sistem "Manajemen Rantai Pasok" digambarkan sebagai satu proses tunggal. Ia berinteraksi dengan empat entitas eksternal: **Pemasok** (menerima PO, mengirim barang & faktur), **Pelanggan** (mengirim pesanan), **Manajemen** (menerima laporan), dan **Departemen Produksi** (meminta bahan baku, melaporkan produk jadi).
- **DFD Level 0:** Diagram ini memecah "Sistem Manajemen Rantai Pasok" menjadi tiga proses utama:

1. **1.0 Kelola Pembelian:** Menangani semua interaksi dengan pemasok, mulai dari pembuatan PO hingga penerimaan barang.
2. **2.0 Kelola Inventaris:** Mengelola semua data stok, baik bahan baku maupun barang jadi, termasuk pembaruan kuantitas dan pemantauan level stok.
3. 3.0 Kelola Penjualan: Memproses pesanan dari pelanggan dan memperbarui stok barang jadi.

   Aliran data antar proses ini dan ke penyimpanan data (misalnya, Data Store "Stok Barang" dan "Data Pesanan") juga digambarkan dengan jelas.

- **DFD Level 1 (Dekomposisi Proses 2.0 Kelola Inventaris):** Untuk memberikan detail lebih lanjut, proses "Kelola Inventaris" dipecah lagi menjadi sub-proses:
    1. **2.1 Catat Barang Masuk:** Menerima data dari proses 1.0 dan memperbarui *Data Store* "Stok Barang".
    2. **2.2 Catat Barang Keluar:** Menerima data permintaan dari Departemen Produksi atau data penjualan dari proses 3.0 dan memperbarui *Data Store* "Stok Barang".
    3. **2.3 Monitor Level Stok:** Secara periodik memeriksa *Data Store* "Stok Barang" dan mengirimkan notifikasi "Stok Rendah" ke proses 1.0 jika diperlukan.

Diagram Aktivitas (UML Activity Diagram)

Untuk memodelkan logika alur kerja (workflow) dari sebuah proses bisnis yang spesifik, Diagram Aktivitas digunakan. Sebagai contoh, proses "Pemesanan Bahan Baku" dimodelkan sebagai berikut:

1. Aktivitas dimulai saat **Sistem** mengirimkan notifikasi "Stok Rendah".
2. **Staf Pembelian** menerima notifikasi dan membuat draf *Purchase Order* (PO) di sistem.
3. Sebuah *decision node* muncul: "Apakah total nilai PO > Rp 10.000.000?".
4. Jika ya, PO diteruskan ke **Manajer Pembelian** untuk persetujuan. Jika tidak, PO langsung disetujui secara otomatis.
5. Setelah disetujui, **Sistem** mengirimkan PO ke **Pemasok**.
6. Aktivitas berakhir.

Pemodelan ini memastikan bahwa semua logika bisnis dan alur kerja telah dipahami dengan benar sebelum perancangan detail dan pengkodean dimulai.

1.6 Pemodelan Data (Aplikasi Bab 6: Pemodelan Data)

Setelah memodelkan proses, fokus beralih ke pemodelan data. Tujuannya adalah untuk mendefinisikan data apa yang perlu disimpan oleh sistem dan bagaimana data-data tersebut saling berhubungan. Alat utama yang digunakan adalah *Entity-Relationship Diagram* (ERD).
Entity-Relationship Diagram (ERD)

ERD secara visual merepresentasikan entitas-entitas data utama dalam sistem dan hubungan (relationship) di antara mereka. Untuk sistem SCM PT. Manufaktur Jaya, entitas-entitas berikut diidentifikasi:

- **Entitas:**
    - **PEMASOK:** Menyimpan informasi tentang setiap pemasok (ID_Pemasok, Nama, Alamat, Kontak).
    - **PRODUK:** Menyimpan detail setiap item, baik bahan baku maupun barang jadi (ID_Produk, Nama, Deskripsi, Satuan, Level_Stok_Min).
    - **PESANAN_PEMBELIAN:** Menyimpan data setiap PO (No_PO, Tanggal, ID_Pemasok, Status).
    - **PELANGGAN:** Menyimpan informasi tentang pelanggan (ID_Pelanggan, Nama, Alamat).
    - **PESANAN_PENJUALAN:** Menyimpan data setiap pesanan dari pelanggan (No_Pesanan, Tanggal, ID_Pelanggan).
- **Atribut:** Setiap entitas memiliki atribut yang relevan, dengan *primary key* yang digarisbawahi (misalnya, ID_Pemasok untuk PEMASOK).
- **Hubungan:**
    - Seorang **PEMASOK** dapat memiliki banyak **PESANAN_PEMBELIAN** (Hubungan satu-ke-banyak).
    - Sebuah **PESANAN_PEMBELIAN** harus dimiliki oleh tepat satu **PEMASOK**.
    - Sebuah **PESANAN_PEMBELIAN** dapat terdiri dari banyak **PRODUK**, dan sebuah **PRODUK**

dapat ada di banyak **PESANAN_PEMBELIAN**. Ini adalah hubungan banyak-ke-banyak, yang diselesaikan dengan membuat entitas asosiatif bernama **DETAIL_PEMBELIAN** (berisi No_PO, ID_Produk, Kuantitas, Harga).

- Hubungan serupa (banyak-ke-banyak) ada antara **PESANAN_PENJUALAN** dan **PRODUK**, yang diselesaikan dengan entitas asosiatif **DETAIL_PENJUALAN**.

ERD ini menjadi cetak biru untuk desain basis data, memastikan bahwa struktur data logis dan mampu mendukung semua proses bisnis yang telah dimodelkan sebelumnya.

1.7 Perancangan Arsitektur dan Antarmuka (Aplikasi Bab 7: Perancangan Sistem)

Pada tahap ini, model logis yang telah dibuat (DFD dan ERD) diterjemahkan menjadi spesifikasi desain fisik. Ini mencakup keputusan tentang arsitektur teknologi dan bagaimana pengguna akan berinteraksi dengan sistem.

Desain Arsitektur

Untuk memastikan skalabilitas, kemudahan pemeliharaan, dan aksesibilitas, dipilih arsitektur client-server berbasis web tiga lapis (three-tier architecture):

1. **Lapis Presentasi (Presentation Tier):** Ini adalah antarmuka pengguna (UI) yang diakses melalui browser web oleh pengguna. Lapis ini bertanggung jawab untuk menampilkan data dan menangkap input dari pengguna.

2. **Lapis Logika Bisnis (Business Logic Tier):** Ini adalah server aplikasi yang menampung semua aturan dan proses bisnis. Misalnya, logika untuk memeriksa level stok, menghitung total pesanan, dan memvalidasi data input berada di lapis ini.
3. **Lapis Data (Data Tier):** Ini adalah server basis data yang bertanggung jawab untuk menyimpan dan mengelola semua data aplikasi. Lapis ini hanya berinteraksi dengan lapis logika bisnis, tidak langsung dengan pengguna.

Arsitektur ini memisahkan antara tampilan, proses, dan data, sehingga perubahan pada satu lapis (misalnya, pembaruan desain UI) tidak akan mempengaruhi lapis lainnya, yang sangat mendukung kebutuhan pemeliharaan jangka panjang.

Desain Antarmuka Pengguna (User Interface - UI)

Desain UI berfokus pada kemudahan penggunaan (usability) dan efisiensi, terutama karena pengguna sistem ini adalah staf operasional yang membutuhkan kecepatan dalam input data. Wireframe dan mockup dibuat untuk beberapa layar kunci:

- **Dashboard Utama:** Menampilkan ringkasan informasi penting secara visual, seperti jumlah PO yang tertunda, daftar item dengan stok di bawah minimum, dan grafik penjualan bulanan. Ini memberikan visibilitas cepat bagi manajer.
- **Formulir Input Purchase Order:** Dirancang untuk meminimalkan pengetikan manual. Nama pemasok dan produk dapat dipilih dari daftar *drop-down*. Sistem secara otomatis mengisi harga satuan produk

berdasarkan data yang tersimpan, dan pengguna hanya perlu memasukkan kuantitas.

- **Tampilan Daftar Inventaris:** Menampilkan daftar semua produk dengan informasi stok saat ini, level stok minimum, dan status. Terdapat fitur pencarian dan filter untuk memudahkan navigasi. Baris untuk item yang stoknya di bawah minimum diberi warna merah untuk menarik perhatian.
- **Halaman Laporan:** Antarmuka yang sederhana untuk memilih jenis laporan (misalnya, Laporan Stok), menentukan rentang tanggal, dan menghasilkan laporan dalam format tabel yang dapat diekspor ke format PDF atau Excel.

1.8 Perancangan Basis Data dan Program (Aplikasi Bab 8: Perancangan Detail)

Tahap perancangan detail menerjemahkan desain tingkat tinggi menjadi spesifikasi teknis yang siap untuk diimplementasikan oleh para programmer.

Desain Basis Data

ERD yang telah dibuat pada Bab 6 dikonversi menjadi skema basis data relasional. Proses ini melibatkan:

1. **Normalisasi:** Setiap tabel data dianalisis dan distrukturkan untuk memenuhi setidaknya Bentuk Normal Ketiga (3NF). Ini dilakukan untuk menghilangkan redundansi data dan mencegah anomali pembaruan, penyisipan, dan penghapusan

data, yang sangat penting untuk menjaga integritas data.

2. **Pembuatan Kamus Data:** Sebuah kamus data yang komprehensif dibuat untuk mendokumentasikan setiap detail dari skema basis data.

Tabel 1.2: Contoh Kamus Data untuk Tabel tbl_produk

Nama Kolom	Tipe Data	Panjang	Keterangan	Batasan
id_produk	VARCHAR	10	Kunci utama, kode unik untuk setiap produk.	Primary Key, NOT NULL
nama_produk	VARCHAR	100	Nama lengkap produk.	NOT NULL
deskripsi	TEXT	-	Deskripsi detail produk.	-
satuan	VARCHAR	20	Satuan unit produk (misalnya, 'meter', 'unit', 'kg').	NOT NULL
stok_saat_ini	INT	11	Jumlah stok yang tersedia saat ini di gudang.	NOT NULL, Default 0
level_stok_min	INT	11	Batas minimum stok sebelum notifikasi pemesanan ulang.	NOT NULL, Default 0

Kamus data ini berfungsi sebagai referensi tunggal yang presisi bagi pengembang basis data, memastikan konsistensi dalam implementasi dan menjadi dokumentasi vital untuk pemeliharaan di masa depan.

Spesifikasi Program

Untuk logika bisnis yang kompleks dan krusial, spesifikasi program dibuat menggunakan pseudocode. Ini memberikan

panduan langkah-demi-langkah bagi programmer tanpa terikat pada sintaks bahasa pemrograman tertentu.

Contoh Pseudocode: `prosesPemicuPemesananOtomatis`

```
FUNCTION prosesPemicuPemesananOtomatis()
  // Fungsi ini dijalankan setiap kali ada perubahan stok

  DAFTAR_PRODUK_KRITIS = SELECT id_produk FROM tbl_produk WHERE
stok_saat_ini <= level_stok_min

  FOR EACH produk IN DAFTAR_PRODUK_KRITIS
    // Periksa apakah sudah ada PO aktif untuk produk ini
    PO_AKTIF_ADA = CHECK_PO_AKTIF(produk.id_produk)

    IF NOT PO_AKTIF_ADA THEN
      // Jika tidak ada PO aktif, buat notifikasi
      BUAT_NOTIFIKASI("Stok Kritis untuk " + produk.id_produk,
"Departemen Pembelian")
    END IF
  END FOR
END FUNCTION
```

Spesifikasi seperti ini memastikan bahwa logika inti diimplementasikan secara konsisten dan sesuai dengan aturan bisnis yang telah ditetapkan.

1.9 Implementasi dan Pengujian (Aplikasi Bab 9: Konstruksi dan Pengujian Sistem)

Rencana Konstruksi

Tahap konstruksi adalah saat di mana desain fisik diterjemahkan menjadi kode program yang berfungsi. Berdasarkan arsitektur yang telah dipilih, tim pengembang akan menggunakan tumpukan teknologi (technology stack) yang telah disepakati:

- **Backend (Lapis Logika Bisnis):** Framework PHP Laravel, karena kematangan, keamanan, dan dukungan komunitasnya yang kuat.
- **Frontend (Lapis Presentasi):** Framework JavaScript Vue.js, karena reaktivitasnya yang memungkinkan pembuatan antarmuka pengguna yang dinamis dan responsif.
- **Database (Lapis Data):** MySQL, karena merupakan sistem manajemen basis data relasional yang andal, open-source, dan banyak digunakan.
- **Kontrol Versi:** Git akan digunakan untuk mengelola perubahan kode dan memfasilitasi kolaborasi antar pengembang.

Strategi Pengujian

Strategi pengujian dirancang agar selaras dengan sifat sekuensial dari model Waterfall. Pengujian dilakukan secara bertahap dan sistematis setelah fase konstruksi selesai.

1. **Pengujian Unit (*Unit Testing*):** Setelah seorang programmer menyelesaikan sebuah modul atau fungsi (misalnya, fungsi untuk menyimpan PO baru), mereka akan menulis dan menjalankan tes untuk memverifikasi bahwa modul tersebut bekerja sesuai spesifikasi secara terisolasi.
2. **Pengujian Integrasi (*Integration Testing*):** Setelah beberapa modul selesai, modul-modul tersebut digabungkan. Pengujian integrasi dilakukan untuk memastikan bahwa modul-modul tersebut dapat berinteraksi satu sama lain tanpa masalah. Contohnya,

menguji apakah pembaruan stok setelah penerimaan barang (modul inventaris) berjalan dengan benar setelah sebuah PO (modul pembelian) ditandai sebagai 'selesai'.

3. **Pengujian Sistem (*System Testing*):** Setelah seluruh sistem selesai dibangun dan diintegrasikan, tim *Quality Assurance* (QA) akan melakukan pengujian menyeluruh. Pengujian ini memvalidasi sistem secara keseluruhan terhadap dokumen SRS (Tabel 1.1). Semua kebutuhan fungsional (apakah sistem bisa melakukan X?) dan non-fungsional (apakah sistem cukup cepat dan aman?) akan diuji.
4. **Pengujian Penerimaan Pengguna (*User Acceptance Testing* - UAT):** Ini adalah tahap pengujian terakhir sebelum sistem diluncurkan. Sekelompok pengguna akhir yang representatif (misalnya, dua staf gudang dan satu staf pembelian) akan diberikan skenario tugas nyata (misalnya, "Buat PO untuk 100 meter kayu jati dari Pemasok A"). Mereka akan menjalankan tugas tersebut di sistem dan memberikan umpan balik akhir. Sistem dianggap siap untuk implementasi hanya setelah berhasil melewati UAT.

1.10 Penerapan dan Pemeliharaan (Aplikasi Bab 10: Implementasi dan Dukungan)

Strategi Penerapan (Cutover)

Memilih strategi penerapan yang tepat sangat penting untuk meminimalkan gangguan terhadap operasional bisnis yang sedang berjalan. Mengingat kompleksitas sistem SCM, strategi big bang (mengganti seluruh sistem lama dengan sistem baru

secara serentak) dianggap terlalu berisiko. Sebaliknya, strategi implementasi bertahap (phased implementation) dipilih:

- **Fase 1:** Modul Manajemen Inventaris akan diimplementasikan terlebih dahulu. Selama fase ini, staf gudang akan mulai menggunakan sistem baru untuk mencatat stok, sementara departemen lain masih menggunakan metode lama. Ini memungkinkan perusahaan untuk fokus pada stabilisasi bagian paling inti dari sistem.
- **Fase 2:** Setelah modul inventaris berjalan stabil selama satu bulan, Modul Manajemen Pembelian akan diaktifkan.
- **Fase 3:** Terakhir, Modul Manajemen Penjualan akan diintegrasikan.

Pendekatan ini memungkinkan masalah untuk diisolasi dan diselesaikan per fase, sehingga mengurangi risiko kegagalan total.

Rencana Pelatihan dan Dukungan

Pelatihan pengguna adalah kunci untuk memastikan adopsi sistem yang sukses. Rencana pelatihan yang komprehensif akan dilaksanakan:

- Sesi pelatihan berbasis peran akan diadakan untuk setiap kelompok pengguna (staf gudang, staf pembelian, manajer).
- Manual pengguna yang detail dan panduan referensi cepat akan disediakan.

- Sebuah tim dukungan (*help desk*) internal akan dibentuk untuk menangani pertanyaan dan masalah pengguna selama dan setelah periode transisi.

Rencana Pemeliharaan

Rencana pemeliharaan dirancang untuk memastikan sistem tetap andal, aman, dan relevan dalam jangka panjang. Rencana ini mencakup:

- **Pemeliharaan Korektif:** Prosedur untuk melaporkan dan memperbaiki bug atau kesalahan yang ditemukan setelah sistem beroperasi.
- **Pemeliharaan Adaptif:** Prosedur untuk memodifikasi sistem sebagai respons terhadap perubahan di lingkungan bisnis atau teknologi (misalnya, perubahan peraturan pajak, pembaruan sistem operasi server).
- **Pemeliharaan Perfektif:** Peningkatan fungsionalitas atau performa sistem berdasarkan umpan balik dari pengguna.

Secara strategis, rencana pemeliharaan ini terhubung dengan siklus hidup organisasi. Manajemen PT. Manufaktur Jaya berkomitmen untuk melakukan tinjauan sistem tahunan. Tinjauan ini tidak hanya membahas masalah teknis, tetapi juga menilai kembali keselarasan sistem dengan tujuan bisnis perusahaan yang mungkin telah berevolusi. Proses ini merupakan upaya sadar untuk menyuntikkan keteraturan baru dan melawan entropi, memastikan bahwa investasi teknologi

terus memberikan nilai seiring berjalannya waktu dan tidak menjadi sistem warisan yang kaku.

Studi Kasus 2: Aplikasi Mobile E-Health "Klinik Sehat Selalu" (Paradigma Adaptif dan Berpusat pada Pengguna)

2.1 Latar Belakang dan Konteks Sistem (Aplikasi Bab 1: Konsep Sistem)

Profil Organisasi

"Klinik Sehat Selalu" adalah sebuah klinik kesehatan swasta yang telah beroperasi selama lima tahun. Klinik ini menawarkan layanan dokter umum dan beberapa spesialis. Sebagai organisasi yang relatif muda dan beroperasi di lingkungan perkotaan yang kompetitif, Klinik Sehat Selalu berada pada tahap "Pertumbuhan" (Growth) dalam siklus hidupnya. Fokus utamanya adalah memperluas jangkauan pasar, meningkatkan pengalaman pasien untuk membangun loyalitas, dan mengelola peningkatan volume pasien secara efisien. Inovasi layanan dianggap sebagai kunci untuk memenangkan persaingan.

Analisis Sistem Saat Ini

Sistem yang ada di klinik saat ini merupakan kombinasi dari sistem fisik dan sistem abstrak yang belum terintegrasi dengan baik. Proses pendaftaran dan penjadwalan janji temu masih dilakukan secara konvensional melalui telepon atau datang langsung. Sistem ini dapat dianalisis sebagai berikut:

- **Komponen:** Resepsionis, staf administrasi, dokter, pasien, catatan medis kertas, dan buku janji temu.
- **Batasan:** Interaksi antara klinik dan pasien yang terbatas pada kontak fisik atau telepon.

- **Lingkungan Luar:** Pasien, perusahaan asuransi, laboratorium rujukan, dan klinik kompetitor.
- **Masukan:** Panggilan telepon dari pasien, data pasien baru yang diisi di formulir kertas, hasil lab dari laboratorium eksternal.
- **Proses:** Pencatatan janji temu di buku besar, pencarian manual rekam medis pasien, penulisan resep secara manual.
- **Keluaran:** Jadwal janji temu yang dikonfirmasi secara lisan, resep kertas, bukti pembayaran fisik.

Identifikasi Masalah

Seiring dengan pertumbuhan jumlah pasien, sistem manual yang ada mulai menunjukkan kelemahan signifikan yang menghambat skalabilitas dan kualitas layanan:

- **Antrian Panjang:** Proses pendaftaran dan pencarian rekam medis yang manual menyebabkan waktu tunggu yang lama bagi pasien, baik di telepon maupun di klinik.
- **Keterbatasan Akses Informasi:** Pasien tidak memiliki cara mudah untuk melihat jadwal dokter yang tersedia, riwayat kunjungan mereka, atau hasil tes laboratorium tanpa harus menghubungi klinik secara langsung.
- **Beban Administratif Tinggi:** Staf resepsionis menghabiskan sebagian besar waktunya untuk tugas-tugas repetitif seperti penjadwalan, konfirmasi janji temu, dan menjawab pertanyaan umum, yang mengurangi waktu untuk melayani pasien di tempat.

- **Keterlibatan Pasien Rendah:** Interaksi dengan pasien bersifat transaksional dan hanya terjadi saat kunjungan. Tidak ada platform untuk memberikan pengingat janji temu, edukasi kesehatan, atau mengelola hubungan pasien secara proaktif.

2.2 Inisiasi dan Perancangan Proyek (Aplikasi Bab 2: Peran Analis & Manajemen Proyek)

Pemicu Proyek

Manajemen klinik mengidentifikasi kebutuhan baru untuk meningkatkan daya saing melalui inovasi layanan digital.1 Mereka melihat tren penggunaan aplikasi mobile di sektor kesehatan dan ingin memanfaatkan teknologi ini untuk memecahkan masalah operasional sekaligus meningkatkan nilai bagi pasien. Proyek ini diinisiasi dengan visi untuk menciptakan "klinik digital" yang lebih mudah diakses dan berpusat pada pasien.

Analisis Kelayakan (PIECES Framework)

- **Performance:** Waktu layanan di bagian administrasi lambat, menyebabkan throughput pasien rendah.
- **Information:** Informasi jadwal dokter tidak transparan bagi pasien. Pasien juga tidak memiliki akses mudah ke data kesehatan pribadi mereka.
- **Economy:** Potensi kehilangan pasien yang beralih ke kompetitor yang menawarkan kemudahan digital. Beban kerja administratif yang tinggi dapat diartikan sebagai biaya operasional yang tidak efisien.
- **Control:** Risiko kesalahan penjadwalan (misalnya, *double booking*) tinggi karena proses manual.

- **Efficiency:** Proses penjadwalan dan konfirmasi janji temu sangat tidak efisien dan memakan banyak waktu staf.
- **Service:** Kualitas layanan secara keseluruhan terpengaruh oleh waktu tunggu yang lama dan akses informasi yang terbatas, yang berpotensi menurunkan kepuasan dan loyalitas pasien.

Penentuan Ruang Lingkup dan Tujuan

Tujuan proyek adalah mengembangkan aplikasi mobile "Klinik Sehat Selalu" untuk platform Android dan iOS. Visi produknya adalah menjadi asisten kesehatan digital pribadi bagi pasien klinik. Namun, disadari bahwa membangun semua fitur sekaligus tidak realistis. Oleh karena itu, ruang lingkup awal (untuk Minimum Viable Product - MVP) difokuskan pada fungsionalitas inti yang memberikan nilai terbesar: pendaftaran pasien, melihat jadwal dokter, dan membuat janji temu secara online.

2.3 Pemilihan Metodologi Pengembangan (Aplikasi Bab 3: Metodologi Pengembangan)

Analisis Faktor Pemilihan

Konteks proyek ini sangat berbeda dari PT. Manufaktur Jaya, yang mengarah pada pilihan metodologi yang berbeda.

- **Kejelasan Kebutuhan Pengguna: Rendah hingga Sedang.** Meskipun tujuan utamanya jelas (membuat janji temu online), detail spesifik mengenai fitur dan alur interaksi terbaik bagi pasien belum sepenuhnya

dipahami. Kebutuhan ini kemungkinan besar akan berkembang dan berubah seiring dengan diperolehnya umpan balik dari pengguna nyata.

- **Penguasaan Teknologi: Tinggi.** Proyek ini akan dikerjakan oleh agensi pengembangan aplikasi mobile eksternal yang memiliki keahlian tinggi dalam teknologi mobile dan metodologi modern.
- **Tingkat Kerumitan Sistem: Sedang.** Melibatkan integrasi dengan sistem internal klinik (jika ada) dan memerlukan desain UI/UX yang baik, namun fungsionalitas intinya tidak terlalu kompleks.
- **Tingkat Keandalan Sistem: Tinggi.** Sistem harus andal, tetapi sifatnya yang iteratif memungkinkan perbaikan berkelanjutan.
- **Jadwal Waktu Pelaksanaan: Cepat.** Ada tekanan pasar untuk meluncurkan produk secepat mungkin untuk mendapatkan keunggulan kompetitif.

Justifikasi Pemilihan Metodologi Agile (Scrum)

Berdasarkan analisis ini, metodologi Agile dengan kerangka kerja Scrum dipilih sebagai pendekatan yang paling sesuai. Justifikasinya adalah:

1. **Fleksibilitas terhadap Perubahan:** Sifat iteratif Agile sangat ideal untuk proyek di mana kebutuhan tidak sepenuhnya beku di awal. Scrum memungkinkan tim untuk beradaptasi dengan umpan balik dan perubahan prioritas di setiap siklus pengembangan (*sprint*).
2. **Fokus pada Pengiriman Nilai Cepat:** Agile memprioritaskan pengiriman perangkat lunak yang

berfungsi dalam siklus pendek (biasanya 2-4 minggu). Ini memungkinkan klinik untuk meluncurkan MVP lebih cepat dan mulai mendapatkan manfaat serta umpan balik dari pasar.

3. **Keterlibatan Pelanggan yang Tinggi:** Scrum menekankan kolaborasi erat antara tim pengembang dan pemangku kepentingan (dalam hal ini, perwakilan dari klinik yang bertindak sebagai *Product Owner*). Keterlibatan ini secara langsung mengatasi dua penyebab utama kegagalan proyek menurut laporan CHAOS: "Kurangnya Masukan Pengguna" (12.8%) dan "Perubahan Kebutuhan" (11.8%).[10]
4. **Peningkatan Berkelanjutan:** Mekanisme seperti *Sprint Review* dan *Sprint Retrospective* memastikan bahwa baik produk maupun proses kerja tim terus menerus dievaluasi dan ditingkatkan.

Menggunakan model Waterfall dalam proyek ini akan sangat berisiko. Menetapkan semua persyaratan di awal tanpa validasi dari pengguna nyata kemungkinan besar akan menghasilkan produk yang tidak sesuai dengan harapan dan memerlukan perombakan besar yang mahal.

2.4 Analisis Kebutuhan Sistem (Aplikasi Bab 4: Analisis Kebutuhan)

Dalam pendekatan Agile, analisis kebutuhan adalah proses yang berkelanjutan, bukan fase satu kali. Kebutuhan dikelola dalam sebuah **Product Backlog**, yang merupakan daftar terprioritaskan dari semua fitur, fungsionalitas, dan perbaikan yang diinginkan untuk produk. Kebutuhan ini sering kali

diekspresikan dalam format **User Stories** untuk menjaga fokus pada nilai bagi pengguna.
Teknik Pengumpulan Kebutuhan

Product Owner (seorang manajer dari Klinik Sehat Selalu) bekerja sama dengan analis sistem untuk mengumpulkan kebutuhan awal melalui:

- **Workshop dengan Staf Klinik:** Diskusi kelompok terfokus dengan resepsionis dan dokter untuk memahami alur kerja dan poin-poin masalah mereka.
- **Survei dan Wawancara Pasien:** Mengumpulkan masukan dari pasien tentang apa yang akan membuat pengalaman mereka lebih baik.
- **Analisis Kompetitor:** Mempelajari aplikasi e-health lain yang ada di pasar untuk mengidentifikasi fitur standar dan peluang diferensiasi.

Product Backlog Awal

Hasil dari kegiatan di atas disusun menjadi Product Backlog awal. Setiap item adalah sebuah User Story yang mengikuti format: "Sebagai , saya ingin agar ".

Contoh Item dalam Product Backlog (Diberi Peringkat Prioritas):

1. **Pendaftaran & Login:** "Sebagai *pasien baru,* saya ingin *bisa mendaftar akun menggunakan email dan nomor telepon* agar *saya dapat menggunakan aplikasi*."

2. **Melihat Jadwal Dokter:** "Sebagai *pasien*, saya ingin *bisa melihat jadwal praktik semua dokter untuk satu minggu ke depan* agar *saya bisa merencanakan kunjungan saya*."
3. **Membuat Janji Temu:** "Sebagai *pasien*, saya ingin *bisa memilih dokter, tanggal, dan slot waktu yang tersedia untuk membuat janji temu* agar *saya tidak perlu menelepon klinik*."
4. **Melihat Riwayat Janji Temu:** "Sebagai *pasien*, saya ingin *bisa melihat daftar janji temu saya yang akan datang dan yang sudah selesai* agar *saya bisa mengingat jadwal saya*."
5. **Pembatalan Janji Temu:** "Sebagai *pasien*, saya ingin *bisa membatalkan janji temu melalui aplikasi setidaknya 24 jam sebelumnya* agar *saya tidak perlu menelepon dan slot waktu bisa digunakan orang lain*."
6. **Notifikasi Pengingat:** "Sebagai *pasien*, saya ingin *menerima notifikasi pengingat H-1 sebelum janji temu saya* agar *saya tidak lupa*."
7. **Profil Dokter:** "Sebagai *pasien*, saya ingin *bisa melihat profil singkat dan spesialisasi dokter* agar *saya bisa memilih dokter yang tepat untuk keluhan saya*."

Product Backlog ini adalah dokumen "hidup" yang akan terus disempurnakan (*refined*) dan diprioritaskan ulang oleh *Product Owner* di setiap *sprint*.

2.5 Pemodelan Proses Bisnis (Aplikasi Bab 5: Pemodelan Proses)

Dalam Scrum, pemodelan formal yang ekstensif seperti DFD sering kali digantikan dengan alat yang lebih ringan dan kolaboratif. Namun, untuk tujuan pemahaman, Diagram

Aktivitas UML masih sangat berguna untuk memvisualisasikan alur kerja yang diinginkan.

Diagram Aktivitas UML: Alur "Membuat Janji Temu"

Diagram ini akan memodelkan alur interaksi pengguna dengan aplikasi:

1. Alur dimulai saat **Pasien** memilih menu "Buat Janji Temu" di aplikasi.
2. **Sistem** menampilkan daftar spesialisasi dokter.
3. **Pasien** memilih spesialisasi (misalnya, "Dokter Umum").
4. **Sistem** menampilkan daftar dokter yang tersedia untuk spesialisasi tersebut beserta jadwal terdekat mereka.
5. **Pasien** memilih seorang dokter.
6. **Sistem** menampilkan kalender dengan tanggal-tanggal yang tersedia untuk dokter tersebut.
7. **Pasien** memilih tanggal.
8. **Sistem** menampilkan slot-slot waktu yang masih tersedia pada tanggal tersebut.
9. **Pasien** memilih slot waktu.
10. **Sistem** menampilkan halaman konfirmasi yang merangkum pilihan pasien (dokter, tanggal, waktu) dan meminta konfirmasi akhir.
11. **Pasien** menekan tombol "Konfirmasi Janji Temu".
12. **Sistem** menyimpan janji temu, memperbarui ketersediaan jadwal dokter, dan menampilkan pesan sukses kepada pasien. Alur berakhir.

Model seperti ini membantu seluruh tim (termasuk *Product Owner* dan pengembang) untuk memiliki pemahaman yang sama tentang fungsionalitas yang akan dibangun sebelum *sprint* dimulai.

2.6 Pemodelan Data (Aplikasi Bab 6: Pemodelan Data)

Meskipun Agile tidak menuntut desain data yang lengkap di awal, pemodelan data dasar tetap penting untuk membangun pondasi yang kuat. ERD dibuat secara iteratif, dimulai dengan entitas inti yang diperlukan untuk MVP.

ERD Awal untuk MVP:

- **Entitas:**
    - **PASIEN:** (ID_Pasien, Nama, Tgl_Lahir, Email, No_Telepon, Kata_Sandi_Hash)
    - **DOKTER:** (ID_Dokter, Nama, Spesialisasi, Foto_Profil)
    - **JADWAL_PRAKTIK:** (ID_Jadwal, ID_Dokter, Hari, Jam_Mulai, Jam_Selesai)
    - **JANJI_TEMU:** (ID_Janji, ID_Pasien, ID_Dokter, Tgl_Janji, Waktu_Janji, Status)
- **Hubungan:**
    - Seorang **DOKTER** dapat memiliki banyak **JADWAL_PRAKTIK**.
    - Seorang **PASIEN** dapat membuat banyak **JANJI_TEMU**.
    - Sebuah **JANJI_TEMU** dibuat oleh satu **PASIEN** dan untuk satu **DOKTER**.

Seiring berjalannya pengembangan, ERD ini akan diperluas untuk mencakup entitas lain seperti **REKAM_MEDIS**, **RESEP**,

dan **HASIL_LAB** saat fitur-fitur tersebut ditambahkan ke dalam *Product Backlog*.

2.7 Perancangan Arsitektur dan Antarmuka (Aplikasi Bab 7: Perancangan Sistem)

Desain Arsitektur

Arsitektur yang dipilih adalah arsitektur modern yang umum digunakan untuk aplikasi mobile:

- **Aplikasi Mobile (Frontend):** Aplikasi *native* untuk iOS (dikembangkan dengan Swift) dan Android (dikembangkan dengan Kotlin) untuk memberikan performa dan pengalaman pengguna terbaik.
- **Backend (API):** Sebuah *backend* berbasis layanan mikro (*microservices*) yang diekspos melalui REST API. Arsitektur ini memungkinkan setiap layanan (misalnya, layanan pengguna, layanan jadwal, layanan janji temu) untuk dikembangkan dan diskalakan secara independen.
- **Basis Data:** Menggunakan kombinasi basis data SQL (misalnya, PostgreSQL) untuk data transaksional yang terstruktur dan NoSQL (misalnya, MongoDB) jika diperlukan untuk data yang kurang terstruktur di masa depan.
- **Platform:** Seluruh *backend* akan di-deploy di platform *cloud* (misalnya, Google Cloud atau AWS) untuk memastikan skalabilitas, ketersediaan, dan kemudahan manajemen.

Desain Antarmuka Pengguna (UI/UX)

Desain UI/UX adalah bagian sentral dari proyek ini. Prosesnya sangat iteratif:

1. **Low-Fidelity Wireframes:** Sketsa kasar dibuat untuk setiap layar aplikasi (login, beranda, daftar dokter, kalender, konfirmasi) untuk merencanakan tata letak dan alur navigasi.
2. **High-Fidelity Mockups:** *Wireframe* diubah menjadi desain visual yang detail menggunakan alat seperti Figma atau Sketch, lengkap dengan skema warna, tipografi, dan ikonografi yang sesuai dengan branding klinik.
3. **Prototipe Interaktif:** *Mockup* dihubungkan menjadi prototipe yang dapat diklik. Prototipe ini kemudian diuji dengan beberapa pasien untuk mendapatkan umpan balik awal mengenai kemudahan penggunaan sebelum satu baris kode pun ditulis. Umpan balik ini digunakan untuk menyempurnakan desain.

2.8 Perancangan Basis Data dan Program (Aplikasi Bab 8: Perancangan Detail)

Dalam Scrum, perancangan detail terjadi "just-in-time" di dalam setiap *sprint*. Sebelum *sprint* dimulai, selama sesi *Sprint Planning*, tim akan mengambil beberapa *User Story* teratas dari *Product Backlog*. Kemudian, tim akan memecah *user story* tersebut menjadi tugas-tugas teknis yang lebih kecil, termasuk perancangan basis data dan logika program yang diperlukan.

Contoh untuk User Story "Membuat Janji Temu":

- **Tugas Desain Basis Data:** Memastikan tabel JANJI_TEMU memiliki semua kolom yang diperlukan (ID_Pasien, ID_Dokter, Tgl_Janji, dll.) dan memiliki indeks yang tepat untuk pencarian cepat.
- **Tugas Desain API:** Merancang *endpoint* API, misalnya POST /api/appointments, yang akan menerima data janji temu baru dari aplikasi mobile. Mendefinisikan format JSON untuk *request* dan *response*.
- **Tugas Logika Backend:** Menulis logika untuk memvalidasi permintaan janji temu (misalnya, memeriksa apakah slot waktu masih tersedia), menyimpan data ke basis data, dan mengirim email konfirmasi ke pasien.
- **Tugas UI Frontend:** Mengembangkan layar kalender dan pemilihan waktu di aplikasi mobile yang akan berkomunikasi dengan API *backend*.

Pendekatan ini memastikan bahwa upaya perancangan detail hanya difokuskan pada fitur yang akan segera dikerjakan, menghindari pemborosan waktu untuk merancang fitur yang mungkin berubah atau dibatalkan nanti.

2.9 Implementasi dan Pengujian (Aplikasi Bab 9: Konstruksi dan Pengujian Sistem)

Proses implementasi dan pengujian dalam Scrum terjadi dalam siklus berulang yang disebut **Sprint**.

- **Sprint:** Sebuah periode waktu yang dibatasi (*time-boxed*), biasanya 2 minggu, di mana tim bekerja untuk menyelesaikan sejumlah pekerjaan dari *Sprint Backlog* dan menghasilkan *increment* produk yang berpotensi dapat dirilis.

- **Sprint 1 - Fokus pada Pondasi:** Tim mungkin mengambil *user story* untuk "Pendaftaran & Login" dan "Melihat Jadwal Dokter". Di akhir *sprint* 1, tim harus menghasilkan versi aplikasi yang berfungsi di mana pengguna dapat membuat akun dan melihat jadwal, meskipun belum bisa membuat janji temu.
- **Sprint 2 - Fokus pada Fungsionalitas Inti:** Tim mengambil *user story* "Membuat Janji Temu". Di akhir *sprint* 2, *increment* produk yang dihasilkan kini memiliki fungsionalitas inti yang lengkap.
- **Daily Scrum:** Setiap hari, tim mengadakan rapat singkat (15 menit) untuk menyinkronkan pekerjaan dan mengidentifikasi hambatan.
- **Pengujian Berkelanjutan:** Pengujian bukanlah fase terpisah di akhir. Pengujian unit dan integrasi dilakukan secara terus-menerus oleh pengembang selama *sprint*. Tim QA juga mulai menguji fungsionalitas segera setelah fungsionalitas tersebut selesai dikembangkan, tidak menunggu hingga akhir *sprint*.

2.10 Penerapan dan Pemeliharaan (Aplikasi Bab 10: Implementasi dan Dukungan)

Penerapan (Deployment)

Salah satu prinsip Agile adalah merilis perangkat lunak yang berfungsi secara rutin. Setelah beberapa sprint, produk MVP (yang mencakup fitur-fitur paling prioritas) siap untuk diluncurkan.

- **Peluncuran Awal (Soft Launch):** Aplikasi dirilis ke sekelompok kecil pasien terpilih (*beta testers*) untuk mengumpulkan umpan balik di dunia nyata.
- **Peluncuran Publik:** Setelah umpan balik dari *soft launch* ditangani, aplikasi dirilis secara publik di Google Play Store dan Apple App Store.

Dukungan dan Pemeliharaan (Operasi Berkelanjutan)

Dalam Agile, pengembangan tidak berhenti setelah rilis pertama. Prosesnya terus berlanjut.

- **Umpan Balik sebagai Input:** Umpan balik dari pengguna, ulasan di app store, dan data analitik penggunaan aplikasi menjadi masukan baru untuk *Product Backlog*.
- **Iterasi Berkelanjutan:** Tim terus bekerja dalam *sprint-sprint* berikutnya untuk memperbaiki bug, meningkatkan fitur yang ada, dan membangun fitur-fitur baru dari *Product Backlog* (misalnya, integrasi pembayaran, konsultasi video, dll.).

Pendekatan ini secara langsung menerapkan wawasan tentang hubungan antara sistem dan organisasi yang sedang tumbuh. Aplikasi ini tidak dirancang sebagai produk final, melainkan sebagai platform yang dapat terus berevolusi seiring dengan pertumbuhan Klinik Sehat Selalu dan perubahan ekspektasi pasiennya. Setiap *sprint* adalah siklus penyuntikan energi baru yang menjaga agar sistem tetap relevan dan kompetitif, secara efektif melawan entropi di pasar yang dinamis.

Studi Kasus 3: Sistem Pendukung Keputusan (DSS) Penilaian Risiko Kredit "Dana Cepat" (Paradigma Berbasis Risiko dan Evolusioner)

3.1 Latar Belakang dan Konteks Sistem (Aplikasi Bab 1: Konsep Sistem)

Profil Organisasi

"Dana Cepat" adalah sebuah lembaga keuangan non-bank yang berspesialisasi dalam penyediaan pinjaman mikro dan menengah untuk usaha kecil. Sebagai pemain di industri keuangan, operasional inti mereka bergantung pada kemampuan untuk secara akurat menilai risiko kredit dari calon peminjam. Organisasi ini berada pada tahap "Pertumbuhan" menuju "Kematangan", di mana mereka perlu menstandarisasi proses pengambilan keputusan untuk mengurangi subjektivitas dan mengelola risiko secara lebih sistematis seiring dengan meningkatnya volume aplikasi pinjaman.

Analisis Sistem Saat Ini

Proses penilaian kredit saat ini adalah sistem buatan manusia (human-made system) yang sangat bergantung pada keahlian dan penilaian analis kredit individual.

- **Komponen:** Analis Kredit, Komite Kredit, aplikasi pinjaman (dokumen fisik), laporan keuangan peminjam, data historis pembayaran (dalam spreadsheet).
- **Masukan:** Formulir aplikasi pinjaman, laporan laba rugi, neraca, dan riwayat kredit dari peminjam.
- **Proses:** Analis kredit secara manual meninjau dokumen, melakukan analisis rasio keuangan dasar, dan

memberikan skor subjektif berdasarkan pengalaman. Aplikasi dengan nilai pinjaman besar kemudian dibahas dalam rapat Komite Kredit.
- **Keluaran:** Keputusan "Disetujui" atau "Ditolak" beserta syarat pinjaman (suku bunga, tenor).

Identifikasi Masalah

Sistem manual ini memiliki risiko inheren yang signifikan:

- **Inkonsistensi Keputusan:** Penilaian yang sangat bergantung pada analis individu menyebabkan inkonsistensi. Dua analis yang berbeda mungkin memberikan keputusan yang berbeda untuk peminjam yang sama.
- **Risiko Tinggi Kesalahan Manusia:** Proses manual rentan terhadap kesalahan perhitungan dan pengabaian faktor-faktor risiko penting.
- **Skalabilitas Rendah:** Proses ini lambat dan tidak dapat menangani lonjakan jumlah aplikasi pinjaman tanpa menambah jumlah analis secara signifikan, yang meningkatkan biaya operasional.
- **Kurangnya Pemanfaatan Data:** Data historis pembayaran pinjaman yang berharga tidak digunakan secara sistematis untuk membangun model prediksi risiko yang lebih akurat.

3.2 Inisiasi dan Perancangan Proyek (Aplikasi Bab 2: Peran Analis & Manajemen Proyek)

Pemicu Proyek

Manajemen puncak ingin mengimplementasikan teknologi baru untuk menciptakan keunggulan kompetitif.1 Mereka menargetkan pengembangan sebuah

Decision Support System (DSS) yang dapat membantu analis kredit dalam membuat keputusan yang lebih cepat, konsisten, dan berbasis data. Proyek ini dianggap berisiko tinggi tetapi juga berpotensi memberikan imbal hasil yang tinggi.

Analisis Kelayakan (PIECES Framework)

- **Performance:** Waktu proses dari aplikasi masuk hingga keputusan keluar rata-rata 7 hari kerja, terlalu lama untuk pasar pinjaman mikro.
- **Information:** Tidak ada informasi terpusat mengenai profil risiko peminjam. Analisis bersifat ad-hoc dan tidak terstandarisasi.
- **Economy:** Tingkat kredit macet (*Non-Performing Loan* - NPL) lebih tinggi dari rata-rata industri, menyebabkan kerugian finansial langsung.
- **Control:** Kurangnya kontrol atas konsistensi dan objektivitas proses pengambilan keputusan.
- **Efficiency:** Analis kredit menghabiskan terlalu banyak waktu untuk pengumpulan dan perhitungan data manual, bukan pada analisis strategis.
- **Service:** Peminjam sering mengeluhkan lambatnya proses persetujuan.

Penentuan Ruang Lingkup dan Tujuan

Tujuan proyek adalah membangun DSS Penilaian Risiko Kredit. Sistem ini tidak bertujuan untuk menggantikan analis kredit sepenuhnya, tetapi untuk memberdayakan mereka. Sistem harus mampu: (1) Mengambil data dari berbagai sumber (aplikasi, data historis), (2) Secara otomatis menghitung rasio keuangan dan variabel risiko lainnya, (3) Menjalankan model scoring untuk menghasilkan skor risiko kredit, dan (4) Menyajikan semua informasi ini dalam sebuah dasbor yang komprehensif untuk membantu analis membuat keputusan akhir.

3.3 Pemilihan Metodologi Pengembangan (Aplikasi Bab 3: Metodologi Pengembangan)

Analisis Faktor Pemilihan

Proyek DSS ini memiliki karakteristik yang menuntut pendekatan yang sangat hati-hati terhadap risiko.

- **Kejelasan Kebutuhan Pengguna: Rendah.** Meskipun tujuan umumnya jelas, model *scoring* yang paling efektif dan variabel-variabel prediktif yang paling signifikan tidak diketahui di awal. Hal ini perlu ditemukan melalui eksplorasi data dan pembuatan prototipe.
- **Penguasaan Teknologi: Rendah hingga Sedang.** Tim internal belum memiliki pengalaman dalam membangun model statistik atau *machine learning* untuk penilaian kredit. Ada risiko teknis yang signifikan.
- **Tingkat Kerumitan Sistem: Tinggi.** Sistem ini melibatkan logika bisnis yang kompleks (algoritma *scoring*) dan integrasi data dari berbagai sumber.

- **Tingkat Keandalan Sistem: Sangat Tinggi.** Keputusan yang salah dari sistem ini memiliki dampak finansial langsung dan signifikan.
- **Jadwal Waktu Pelaksanaan: Fleksibel.** Akurasi dan keandalan model lebih penting daripada kecepatan pengiriman.

Justifikasi Pemilihan Metodologi Spiral

Mengingat tingginya ketidakpastian teknis dan bisnis, Model Spiral yang diperkenalkan oleh Barry Boehm dipilih sebagai metodologi yang paling tepat. Model Spiral secara inheren adalah pendekatan berbasis risiko, yang menggabungkan elemen-elemen dari model prototipe dan model Waterfall dalam siklus berulang. Setiap putaran spiral mencakup empat kuadran aktivitas: (1) Perencanaan, (2) Analisis Risiko, (3) Rekayasa (pembangunan), dan (4) Evaluasi Pelanggan.

1. **Fokus pada Manajemen Risiko:** Ini adalah kekuatan utama Model Spiral. Proyek ini akan dimulai dengan menganalisis risiko terbesar (misalnya, "Apakah kita bisa membangun model *scoring* yang akurat?") dan membangun prototipe untuk menjawab pertanyaan tersebut sebelum melanjutkan ke pengembangan skala penuh.
2. **Sifat Evolusioner:** Sistem akan dibangun secara bertahap, dari prototipe konsep, prototipe yang lebih fungsional, hingga sistem operasional penuh. Setiap iterasi memungkinkan validasi dan penyempurnaan berdasarkan pembelajaran dari iterasi sebelumnya.

3. **Fleksibilitas Terkontrol:** Tidak seperti Waterfall yang kaku, Spiral memungkinkan perubahan dan penemuan. Namun, tidak seperti Agile yang sangat cair, setiap siklus Spiral diakhiri dengan evaluasi formal dan perencanaan yang terstruktur, memberikan tingkat kontrol yang dibutuhkan untuk proyek berisiko tinggi.

3.4 - 3.8 Analisis, Pemodelan, dan Perancangan dalam Konteks Spiral

Berbeda dengan pendekatan linear, dalam Model Spiral, fase analisis, pemodelan, dan perancangan terjadi berulang kali di setiap putaran spiral, dengan tingkat detail yang semakin meningkat.

Putaran Spiral 1: Validasi Konsep

- **Tujuan:** Memvalidasi kelayakan teknis dari model *scoring* kredit.
- **Analisis Risiko:** Risiko utama yang diidentifikasi adalah ketidakmampuan untuk menemukan variabel prediktif yang kuat dari data yang ada.
- **Rekayasa (Prototyping):** Tim (termasuk seorang ilmuwan data) membangun sebuah **prototipe analitis** menggunakan perangkat lunak statistik (seperti R atau Python). Mereka mengambil sampel data historis pinjaman dan mencoba membangun model regresi logistik sederhana untuk memprediksi kemungkinan gagal bayar berdasarkan variabel seperti pendapatan, lama usaha, dan rasio utang terhadap pendapatan.
- **Evaluasi:** Hasil prototipe (bukan antarmuka, tetapi laporan statistik tentang akurasi model) dievaluasi bersama analis kredit senior. Kesimpulannya adalah

model tersebut menunjukkan potensi tetapi membutuhkan lebih banyak data dan variabel yang lebih canggih.

Putaran Spiral 2: Pengembangan Prototipe Fungsional

- **Tujuan:** Mengembangkan prototipe dengan antarmuka pengguna yang dapat diuji dan menyempurnakan model *scoring*.
- **Analisis Risiko:** Risiko baru yang diidentifikasi adalah penolakan dari analis kredit jika sistem dianggap terlalu rumit atau tidak transparan (*black box*).
- **Rekayasa (Analisis & Desain):**
    - **Analisis Kebutuhan:** Wawancara mendalam dengan analis kredit untuk merancang dasbor yang informatif dan mudah dipahami.
    - **Pemodelan Data:** ERD awal dibuat untuk entitas PEMINJAM, APLIKASI_PINJAMAN, dan PEMBAYARAN_HISTORIS.
    - **Perancangan UI:** *Mockup* untuk dasbor utama dirancang, menampilkan skor risiko, variabel-variabel utama yang mempengaruhinya, dan data keuangan peminjam.
    - **Implementasi:** Sebuah prototipe berbasis web yang fungsional dibangun. Model *scoring* disempurnakan dengan menambahkan lebih banyak variabel.
- **Evaluasi:** Analis kredit berinteraksi langsung dengan prototipe, memberikan umpan balik berharga tentang tata letak dasbor dan informasi apa yang paling mereka butuhkan.

Putaran Spiral 3: Pembangunan Versi Operasional Awal

- **Tujuan:** Membangun versi pertama dari sistem yang siap untuk diuji coba secara paralel dengan proses manual.
- **Analisis Risiko:** Risiko integrasi dengan sistem lain dan risiko performa saat menangani data *real-time*.
- **Rekayasa (Desain Detail & Konstruksi):**
    - **Desain Detail:** Skema basis data dinormalisasi, spesifikasi program untuk semua modul ditulis.
    - **Konstruksi:** Tim pengembang membangun sistem lengkap berdasarkan desain yang telah divalidasi pada putaran sebelumnya.
- **Evaluasi:** Sistem diuji secara menyeluruh (unit, integrasi, sistem) dan kemudian dievaluasi oleh sekelompok analis dalam uji coba terbatas.

3.9 Implementasi dan Pengujian

Strategi pengujian dalam Model Spiral bersifat inkremental. Setiap prototipe dan setiap *build* diuji. Pengujian tidak hanya berfokus pada pencarian bug, tetapi juga pada validasi asumsi dan pengurangan risiko.

- **Pengujian Prototipe:** Fokus pada pengujian fungsionalitas dan pengumpulan umpan balik dari pengguna untuk menyempurnakan desain.
- **Pengujian Model:** Model *scoring* diuji secara statistik untuk akurasi, presisi, dan recall menggunakan data historis yang tidak digunakan dalam pelatihan model (*hold-out validation*).

- **Pengujian Sistem Akhir:** Melakukan pengujian fungsional, performa, dan keamanan secara komprehensif sebelum penerapan.

3.10 Penerapan dan Pemeliharaan

Strategi Penerapan

Strategi penerapan paralel (parallel adoption) dipilih. Selama 3 bulan, sistem DSS baru akan digunakan bersamaan dengan proses penilaian manual yang lama. Keputusan dari kedua sistem akan dibandingkan. Pendekatan ini, meskipun mahal, sangat penting untuk proyek berisiko tinggi karena memungkinkan validasi akhir terhadap keandalan sistem sebelum proses manual dihentikan sepenuhnya.

Rencana Dukungan dan Pemeliharaan

Pemeliharaan DSS ini bersifat sangat dinamis:

- **Pemantauan Model:** Akurasi model *scoring* harus dipantau secara terus-menerus. Seiring waktu, perilaku peminjam dan kondisi ekonomi dapat berubah, yang dapat menyebabkan degradasi performa model (*model drift*).
- **Pelatihan Ulang Model (*Model Retraining*):** Rencana pemeliharaan mencakup jadwal untuk melatih ulang model *scoring* secara periodik (misalnya, setiap 6 bulan) menggunakan data baru untuk memastikan akurasinya tetap tinggi.
- **Peningkatan Berkelanjutan:** Umpan balik dari analis kredit akan terus dikumpulkan untuk

menyempurnakan antarmuka dan menambahkan fitur-fitur analitis baru.

Kasus ini secara jelas mengilustrasikan bagaimana pendekatan yang berpusat pada risiko dapat mengelola ketidakpastian yang tinggi. Setiap putaran spiral berfungsi sebagai mekanisme pembelajaran, yang memungkinkan tim untuk membuat keputusan yang lebih terinformasi dan secara bertahap "menemukan" solusi yang tepat, bukan hanya "membangun" solusi berdasarkan asumsi awal yang mungkin salah.

Studi Kasus 4: Sistem Informasi Akademik Berbasis Cloud "Universitas Cendekia Bangsa" (Paradigma Modernisasi dan Integrasi)

4.1 Latar Belakang dan Konteks Sistem (Aplikasi Bab 1: Konsep Sistem)

Profil Organisasi

Universitas Cendekia Bangsa (UCB) adalah sebuah universitas besar dan ternama dengan puluhan ribu mahasiswa dan ribuan staf pengajar. Telah berdiri selama lebih dari 40 tahun, UCB berada pada tahap "Kematangan" yang cenderung mengarah ke "Penurunan" (Decline) dalam hal infrastruktur teknologinya. Birokrasi yang mapan dan keengganan untuk berubah telah menyebabkan stagnasi teknologi.

Analisis Sistem Saat Ini

Sistem Informasi Akademik (SIA) UCB saat ini adalah sebuah sistem warisan (legacy system) yang berjalan di atas infrastruktur mainframe IBM S/360 era lama.18 Sistem ini dapat dikarakterisasi sebagai *sistem tertutup* (*closed system*) dalam praktiknya; meskipun secara teoretis dapat berinteraksi, arsitekturnya yang monolitik dan teknologinya yang usang membuatnya sangat sulit untuk diintegrasikan dengan sistem modern lainnya.

- **Komponen:** Aplikasi COBOL untuk registrasi mahasiswa, penjadwalan kelas, dan manajemen nilai; basis data hierarkis; terminal *dumb* untuk akses staf administrasi.

- **Masukan:** Formulir registrasi kertas, formulir perubahan rencana studi, daftar nilai dari dosen.
- **Proses:** Proses *batch* yang berjalan pada malam hari untuk memproses pendaftaran dan memperbarui catatan mahasiswa.
- **Keluaran:** Kartu Hasil Studi (KHS) cetak, jadwal kelas cetak, laporan pendaftaran untuk manajemen.

Identifikasi Masalah

Sistem warisan ini telah menjadi penghambat utama bagi inovasi dan efisiensi universitas. Ini adalah contoh klasik dari entropi organisasi, di mana sistem yang pernah efisien kini menjadi sumber kekacauan dan ketidakteraturan.

- **Biaya Tinggi:** Biaya pemeliharaan perangkat keras *mainframe* dan gaji untuk programmer COBOL yang semakin langka sangatlah mahal.
- **Kurangnya Fleksibilitas:** Mengubah atau menambahkan fitur baru pada sistem membutuhkan waktu berbulan-bulan dan biaya yang sangat besar, menghambat kemampuan universitas untuk beradaptasi dengan kebutuhan pendidikan modern.
- **Tidak Adanya Akses Mobile/Web:** Mahasiswa dan dosen tidak dapat mengakses informasi akademik secara mandiri melalui web atau perangkat mobile. Semua interaksi harus melalui staf administrasi.
- **Fragmentasi Data:** Banyak departemen telah membuat "sistem bayangan" (*shadow systems*) menggunakan spreadsheet atau aplikasi desktop kecil untuk mengatasi keterbatasan SIA utama, yang menyebabkan data

menjadi terfragmentasi, tidak konsisten, dan tidak aman.

4.2 Inisiasi dan Perancangan Proyek (Aplikasi Bab 2: Peran Analis & Manajemen Proyek)

Pemicu Proyek

Tekanan dari dewan universitas dan keluhan yang terus-menerus dari mahasiswa dan fakultas menjadi pemicu utama proyek ini. Visi proyek adalah melakukan transformasi digital pada layanan akademik dengan memodernisasi SIA melalui migrasi ke platform berbasis cloud.

Analisis Kelayakan (PIECES Framework)

- **Performance:** Proses *batch* menyebabkan keterlambatan informasi. Mahasiswa baru bisa melihat hasil registrasi keesokan harinya.
- **Information:** Data terisolasi di *mainframe*. Manajemen tidak bisa mendapatkan laporan analitik secara *real-time* untuk mendukung pengambilan keputusan strategis.
- **Economy:** Biaya operasional tahunan untuk sistem warisan sangat tinggi.
- **Control:** Kurangnya kontrol atas integritas data karena adanya banyak sistem bayangan.
- **Efficiency:** Proses manual dan birokratis yang mengelilingi sistem ini sangat tidak efisien bagi semua pihak.
- **Service:** Pengalaman mahasiswa dan dosen sangat buruk dibandingkan dengan universitas lain yang sudah modern.

Penentuan Ruang Lingkup dan Tujuan

Proyek ini, yang diberi nama "SIA Generasi Baru", memiliki dua tujuan utama: (1) Mempensiunkan sistem mainframe yang lama. (2) Mengimplementasikan SIA berbasis cloud yang terintegrasi, dapat diakses melalui web dan mobile, dan mencakup modul untuk Pendaftaran Mahasiswa, Perwalian Online, Manajemen Nilai, dan Portal Dosen & Mahasiswa.

4.3 Pemilihan Metodologi Pengembangan (Aplikasi Bab 3: Metodologi Pengembangan)

Analisis Faktor Pemilihan

Proyek ini memiliki sifat ganda yang unik.

- **Aspek Migrasi Data:** Proses memindahkan data puluhan tahun dari basis data hierarkis ke basis data relasional di *cloud* adalah tugas yang sangat terstruktur, berisiko tinggi, dan memiliki kebutuhan yang jelas dan tidak akan berubah.
- **Aspek Pengembangan Fitur Baru:** Pengembangan portal web dan aplikasi mobile untuk mahasiswa dan dosen adalah tugas di mana kebutuhan antarmuka dan pengalaman pengguna akan mendapat manfaat dari umpan balik iteratif.

Justifikasi Pemilihan Metodologi Hibrid

Karena sifat ganda ini, tidak ada satu metodologi tunggal yang cocok. Oleh karena itu, pendekatan hibrid dipilih:

1. **Model Waterfall untuk Migrasi Data Inti:** Bagian proyek yang berkaitan dengan analisis data warisan, pemetaan data, transformasi, dan pemuatan (*Extract, Transform, Load* - ETL) akan mengikuti pendekatan Waterfall. Sifatnya yang sekuensial dan penekanannya pada perencanaan dan dokumentasi yang detail sangat penting untuk memastikan tidak ada data historis yang hilang atau rusak selama migrasi.
2. **Model Agile (Scrum) untuk Pengembangan Antarmuka:** Bagian proyek yang berhadapan langsung dengan pengguna, seperti pengembangan Portal Mahasiswa dan Aplikasi Mobile, akan menggunakan Scrum. Ini memungkinkan tim untuk membangun fitur secara inkremental, mendapatkan umpan balik dari mahasiswa dan dosen di setiap *sprint*, dan menyesuaikan desain UI/UX untuk menciptakan pengalaman pengguna terbaik.

Pendekatan hibrid ini memungkinkan tim untuk menerapkan disiplin dan struktur di area yang membutuhkannya (migrasi data) sambil mempertahankan fleksibilitas dan responsivitas di area lain (pengembangan fitur pengguna).

4.4 - 4.8 Analisis, Pemodelan, dan Perancangan

Proses analisis dan perancangan dilakukan secara paralel sesuai dengan metodologi yang dipilih untuk masing-masing bagian.

Jalur Waterfall (Migrasi Data):

- **Analisis:** Tim analis data melakukan *reverse engineering* terhadap struktur data COBOL dan basis data hierarkis untuk memahami setiap elemen data dan aturan bisnis yang tersembunyi di dalam kode program lama.
- **Pemodelan Data:** Sebuah ERD yang komprehensif untuk SIA baru dirancang, mencakup entitas seperti MAHASISWA, MATA_KULIAH, KELAS, JADWAL, dan NILAI.
- **Desain:** Desain detail untuk proses ETL dibuat. Ini termasuk skrip untuk mengekstrak data dari *mainframe*, aturan untuk mentransformasi data (misalnya, membersihkan data yang tidak konsisten, mengubah format), dan prosedur untuk memuatnya ke dalam basis data relasional baru di *cloud*.

Jalur Agile (Pengembangan Antarmuka):

- **Analisis Kebutuhan:** *Product Owner* (perwakilan dari bagian akademik) membuat *Product Backlog* yang berisi *user stories* untuk Portal Mahasiswa dan Dosen. Contoh: "Sebagai seorang mahasiswa, saya ingin bisa melihat transkrip nilai saya kapan saja melalui portal agar saya bisa memantau kemajuan akademik saya."
- **Desain:** Di setiap *sprint*, tim desain UI/UX membuat *wireframe* dan prototipe untuk *user stories* yang akan dikerjakan. Prototipe ini diuji dengan kelompok mahasiswa dan dosen untuk mendapatkan umpan balik cepat.
- **Arsitektur:** Arsitektur berbasis layanan mikro di platform *cloud* (misalnya, Google Cloud Platform) dipilih untuk memastikan skalabilitas dan fleksibilitas. Portal web dan aplikasi mobile akan berkomunikasi dengan layanan *backend* melalui API.

4.9 Implementasi dan Pengujian

Jalur Waterfall:

- **Implementasi:** Tim data menulis dan mengeksekusi skrip ETL. Proses ini dilakukan berulang kali di lingkungan pengembangan untuk memastikan keakuratannya.
- **Pengujian:** Pengujian data sangat krusial. Ini melibatkan validasi data secara ekstensif, seperti membandingkan jumlah catatan antara sistem lama dan baru, memeriksa integritas referensial, dan memvalidasi data sampel secara manual untuk memastikan transformasi berjalan dengan benar.

Jalur Agile:

- **Implementasi:** Tim pengembang membangun fitur-fitur portal dalam *sprint* dua mingguan. Mereka menggunakan praktik *Continuous Integration* (CI) untuk mengintegrasikan kode mereka secara berkala.
- **Pengujian:** Pengujian otomatis (unit dan integrasi) dijalankan setiap kali ada perubahan kode. Pengujian manual fungsional dilakukan oleh tim QA di dalam setiap *sprint*.

4.10 Penerapan dan Pemeliharaan

Strategi Penerapan

Penerapan sistem sebesar ini memerlukan perencanaan yang sangat matang. Strategi penerapan paralel dengan pilot project dipilih:

1. **Migrasi Data Final:** Selama liburan semester, migrasi data final dari sistem *mainframe* ke basis data *cloud* akan dilakukan (*downtime* terencana).
2. **Pilot Project:** Pada semester berikutnya, SIA baru akan diimplementasikan hanya untuk satu fakultas terlebih dahulu. Selama periode ini, sistem *mainframe* lama masih berjalan dalam mode *read-only* sebagai cadangan (*fallback*).
3. **Peluncuran Penuh:** Setelah pilot project berhasil dan semua masalah besar teratasi, sistem baru akan diluncurkan untuk seluruh universitas pada semester berikutnya, dan sistem *mainframe* akan dipensiunkan secara resmi.

Rencana Dukungan dan Pemeliharaan

- **Dukungan:** Tim dukungan IT universitas akan dilatih secara intensif untuk menangani masalah terkait sistem baru.
- **Pemeliharaan:** Rencana pemeliharaan akan mengikuti model Agile. *Product Backlog* akan terus hidup, diisi dengan permintaan fitur baru dari departemen, perbaikan bug, dan penyesuaian yang diperlukan untuk mengikuti perubahan kurikulum atau peraturan akademik. Ini menunjukkan bagaimana organisasi yang matang dapat mengadopsi praktik adaptif untuk merevitalisasi dirinya dan melawan stagnasi teknologi, sebuah contoh nyata dari penyuntikan negentropi untuk memperpanjang siklus hidup sistem dan organisasi.

Studi Kasus 5: Platform E-Commerce untuk UMKM "Pasar Kreatif Lokal" (Paradigma Pengembangan Cepat)

5.1 Latar Belakang dan Konteks Sistem (Aplikasi Bab 1: Konsep Sistem)

Profil Organisasi

"Pasar Kreatif Lokal" adalah sebuah startup teknologi yang baru didirikan dengan misi untuk menyediakan platform e-commerce yang dikhususkan bagi Usaha Mikro, Kecil, dan Menengah (UMKM) yang memproduksi kerajinan tangan dan produk kreatif lokal. Sebagai sebuah startup, organisasi ini berada pada tahap "Kelahiran" (Birth). Lingkungannya sangat tidak pasti, model bisnisnya belum terbukti, dan kecepatan untuk masuk ke pasar (time-to-market) adalah faktor penentu keberhasilan yang paling krusial. Timnya kecil, terdiri dari beberapa pendiri dengan keahlian di bidang bisnis, teknologi, dan desain.

Analisis Sistem Saat Ini

Karena ini adalah startup, belum ada sistem yang berjalan. Proyek ini adalah tentang menciptakan sistem dari nol. Tujuan utamanya adalah membangun sebuah platform digital yang berfungsi sebagai pasar dua sisi, menghubungkan penjual (UMKM) dan pembeli.

Identifikasi Masalah (atau Peluang)

Peluang yang ingin ditangkap adalah kurangnya platform e-commerce yang secara spesifik fokus pada kurasi dan promosi produk kreatif lokal. Masalah yang harus dipecahkan adalah

bagaimana membangun platform yang fungsional, andal, dan menarik bagi kedua sisi pasar (penjual dan pembeli) dengan sumber daya yang sangat terbatas dan dalam waktu yang sangat cepat. Ketidakpastian sangat tinggi: fitur apa yang paling penting bagi penjual? Apa yang akan menarik pembeli untuk mencoba platform baru?

5.2 Inisiasi dan Perancangan Proyek (Aplikasi Bab 2: Peran Analis & Manajemen Proyek)

Pemicu Proyek

Proyek dipicu oleh ide bisnis para pendiri dan didukung oleh pendanaan awal (seed funding). Tujuannya adalah meluncurkan Minimum Viable Product (MVP) secepat mungkin untuk memvalidasi ide bisnis dan mulai membangun basis pengguna.

Analisis Kelayakan (PIECES Framework - Diadaptasi untuk Peluang)

- **Performance:** Kebutuhan akan platform yang cepat dan responsif untuk memberikan pengalaman belanja yang baik.
- **Information:** Kebutuhan untuk menyediakan informasi produk yang kaya (gambar, deskripsi, cerita di balik produk) kepada pembeli dan data penjualan yang berguna bagi penjual.
- **Economy:** Model bisnis bergantung pada kemampuan untuk memfasilitasi transaksi dan mengambil komisi kecil. Kecepatan peluncuran sangat mempengaruhi potensi pendapatan.

- **Control:** Kebutuhan akan sistem manajemen pesanan dan pembayaran yang aman dan andal.
- **Efficiency:** Kebutuhan untuk mengotomatisasi proses pendaftaran penjual, unggah produk, dan pemrosesan pesanan.
- **Service:** Kebutuhan untuk memberikan layanan pelanggan yang baik bagi penjual dan pembeli.

Penentuan Ruang Lingkup dan Tujuan

Tujuan utama adalah meluncurkan MVP dalam waktu 3 bulan. Ruang lingkup MVP didefinisikan secara ketat untuk hanya mencakup fungsionalitas paling inti:

- **Untuk Penjual:** Pendaftaran toko, mengunggah dan mengelola produk, melihat dan memproses pesanan.
- **Untuk Pembeli:** Menjelajahi produk, menambahkan produk ke keranjang belanja, melakukan *checkout* dan pembayaran.

5.3 Pemilihan Metodologi Pengembangan (Aplikasi Bab 3: Metodologi Pengembangan)

Analisis Faktor Pemilihan

Kondisi startup ini sangat unik dan menuntut metodologi yang sangat spesifik.

- **Kejelasan Kebutuhan Pengguna: Sangat Rendah.** Tim memiliki hipotesis tentang apa yang diinginkan pengguna, tetapi ini perlu divalidasi dengan cepat.

Kebutuhan diharapkan berubah secara drastis berdasarkan umpan balik pasar.

- **Penguasaan Teknologi: Tinggi.** Tim teknis inti terdiri dari pengembang senior yang sangat kompeten.
- **Tingkat Kerumitan Sistem: Tinggi.** Membangun platform e-commerce dari nol itu kompleks.
- **Tingkat Keandalan Sistem: Tinggi.** Kepercayaan pengguna sangat penting.
- **Jadwal Waktu Pelaksanaan: Sangat Cepat.** *Time-to-market* adalah segalanya.

Justifikasi Pemilihan Metodologi Extreme Programming (XP)

Mengingat kebutuhan akan kecepatan, fleksibilitas, dan kualitas teknis yang tinggi, Extreme Programming (XP), sebuah varian dari Agile, dipilih. XP sangat cocok karena penekanannya pada praktik-praktik rekayasa perangkat lunak yang memungkinkan pengembangan cepat tanpa mengorbankan kualitas. Lima nilai inti XP sangat selaras dengan kebutuhan startup:

1. **Komunikasi:** XP mendorong komunikasi tatap muka yang konstan antara pengembang dan perwakilan bisnis (pelanggan di tempat), yang sangat penting ketika kebutuhan tidak jelas.
2. **Kesederhanaan (*Simplicity*):** Nilai ini mendorong tim untuk selalu membangun "hal paling sederhana yang bisa berfungsi", menghindari *over-engineering* dan fokus pada pengiriman nilai saat ini.
3. **Umpan Balik (*Feedback*):** XP mencari umpan balik dalam siklus yang sangat cepat, baik dari pelanggan

(melalui rilis kecil yang sering) maupun dari kode itu sendiri (melalui pengujian otomatis).

4. **Keberanian (*Courage*):** Keberanian untuk merancang ulang (*refactor*) kode yang buruk, membuang prototipe, dan mengubah arah berdasarkan umpan balik.
5. **Rasa Hormat (*Respect*):** Kolaborasi yang erat dalam tim, terutama melalui praktik seperti *pair programming*.

5.4 - 5.8 Analisis, Pemodelan, dan Perancangan dalam Konteks XP

Dalam XP, analisis, desain, dan implementasi sangat terjalin erat dan terjadi dalam siklus yang sangat pendek (biasanya mingguan).

Praktik-Praktik XP yang Diterapkan:

- **The Planning Game:** Di awal setiap siklus mingguan, perwakilan bisnis (salah satu pendiri) akan mempresentasikan beberapa *User Story* yang paling prioritas. Tim pengembang akan memberikan estimasi upaya. Bersama-sama, mereka akan memutuskan *story* mana yang akan dikerjakan dalam minggu itu.
- **Small Releases:** Tujuannya adalah untuk memiliki versi platform yang dapat di-deploy di akhir setiap minggu. Rilis awal mungkin hanya internal, tetapi secepat mungkin akan dirilis ke pengguna beta.
- **Metaphor:** Tim menyepakati metafora sederhana untuk sistem, misalnya "Pasar Digital", untuk memandu desain dan komunikasi.
- **Simple Design:** Tim tidak akan membangun arsitektur yang rumit untuk menangani jutaan pengguna di hari pertama. Mereka akan merancang arsitektur paling

sederhana yang dapat mendukung MVP dan siap untuk dikembangkan lebih lanjut nanti.

- **Test-First Development:** Sebelum menulis kode untuk sebuah fitur (misalnya, menambahkan item ke keranjang), pengembang akan menulis tes otomatis yang gagal. Mereka kemudian menulis kode fungsional secukupnya agar tes tersebut berhasil. Ini memastikan *code coverage* yang tinggi dan berfungsi sebagai jaring pengaman untuk perubahan di masa depan.
- **Refactoring:** Tim secara terus-menerus membersihkan dan menyederhanakan kode untuk menjaga agar desain tetap bersih dan mudah diubah.
- **Pair Programming:** Dua programmer bekerja bersama di satu komputer. Satu orang ("driver") menulis kode, sementara yang lain ("navigator") meninjau kode secara *real-time* dan memikirkan gambaran besarnya. Praktik ini meningkatkan kualitas kode dan penyebaran pengetahuan di dalam tim.
- **Collective Code Ownership:** Siapa pun di tim dapat mengubah bagian mana pun dari kode. Ini mendorong tanggung jawab bersama atas kualitas seluruh sistem.
- **Continuous Integration:** Setiap kali sepotong kode selesai, kode tersebut segera diintegrasikan ke dalam basis kode utama dan semua tes otomatis dijalankan untuk memastikan tidak ada yang rusak.

5.9 Implementasi dan Pengujian

Implementasi dan pengujian dalam XP tidak dapat dipisahkan. Berkat praktik *Test-First Development* dan *Continuous Integration*, sistem berada dalam keadaan yang selalu teruji.

- **Siklus Mingguan (Weekly Cycle):** Setiap minggu dimulai dengan *Planning Game* dan diakhiri dengan rilis kecil dari fungsionalitas yang telah selesai dan teruji sepenuhnya.
- **Pengujian Penerimaan:** Di akhir setiap siklus, perwakilan bisnis akan melakukan tes penerimaan terhadap *user story* yang telah selesai untuk memverifikasi bahwa fungsionalitas tersebut memenuhi kebutuhan bisnis.

5.10 Penerapan dan Pemeliharaan

Penerapan

Penerapan dalam XP bersifat berkelanjutan (continuous delivery). Setelah beberapa siklus mingguan, MVP siap untuk diluncurkan ke publik. Setelah itu, peningkatan dan fitur baru akan terus di-deploy ke lingkungan produksi, sering kali beberapa kali dalam seminggu atau bahkan sehari.

Pemeliharaan

Dalam XP, tidak ada fase "pemeliharaan" yang terpisah. Karena tim terus-menerus melakukan refactoring dan menjaga kualitas kode tetap tinggi, dan karena mereka terus mengerjakan fitur baru dan perbaikan bug dalam siklus yang sama, pemeliharaan menjadi bagian integral dari proses pengembangan yang berkelanjutan. Tim tidak pernah "selesai" membangun produk;

mereka terus-menerus meningkatkannya berdasarkan umpan balik pasar.

Kasus ini menunjukkan bagaimana metodologi yang sangat adaptif dan berdisiplin secara teknis seperti XP dapat menjadi alat yang ampuh untuk menavigasi ketidakpastian tinggi di lingkungan *startup*. Dengan fokus pada siklus umpan balik yang cepat dan keunggulan teknis, XP memungkinkan tim kecil untuk membangun produk yang kompleks secara iteratif dan merespons perubahan pasar dengan kecepatan yang tidak mungkin dicapai dengan metodologi tradisional.

Penutup

Sintesis Pembelajaran Lintas Kasus

Analisis mendalam terhadap kelima studi kasus ini memberikan serangkaian pelajaran penting yang melampaui detail teknis masing-masing proyek. Sintesis dari pengalaman-pengalaman ini menegaskan bahwa dalam analisis dan perancangan sistem, konteks adalah segalanya. Tidak ada satu metodologi, arsitektur, atau pendekatan yang secara universal "terbaik". Keberhasilan sebuah proyek pengembangan sistem informasi sangat bergantung pada keselarasan strategis antara pendekatan yang dipilih dengan karakteristik unik dari masalah yang dihadapi, lingkungan operasional, tingkat risiko yang dapat diterima, dan tahap kematangan organisasi yang dilayani.

- **PT. Manufaktur Jaya** menunjukkan bahwa dalam lingkungan yang stabil dengan kebutuhan yang terdefinisi dengan baik, pendekatan prediktif seperti **Waterfall** memberikan struktur, kontrol, dan keandalan yang diperlukan. Upaya untuk menerapkan metodologi yang lebih fleksibel di sini mungkin hanya akan menambah overhead tanpa memberikan manfaat yang sepadan.
- **Klinik Sehat Selalu** mengilustrasikan skenario sebaliknya. Dalam pasar yang dinamis dan dengan kebutuhan pengguna yang masih perlu dieksplorasi, pendekatan adaptif seperti **Agile (Scrum)** menjadi krusial. Kemampuannya untuk mengakomodasi perubahan dan melibatkan pengguna secara terus-

menerus secara langsung memitigasi risiko utama yang sering menyebabkan kegagalan proyek.

- "**Dana Cepat**" menyoroti pentingnya manajemen risiko proaktif. Untuk proyek yang kompleks dan berisiko tinggi seperti DSS, **Model Spiral** menyediakan kerangka kerja yang sistematis untuk menavigasi ketidakpastian, memungkinkan tim untuk belajar dan beradaptasi melalui pembuatan prototipe dan evaluasi di setiap siklus.
- **Universitas Cendekia Bangsa** memperlihatkan bahwa proyek-proyek modern sering kali tidak cocok dengan satu metodologi tunggal. Pendekatan **Hibrid** memungkinkan organisasi untuk menerapkan disiplin dan perencanaan ketat di area yang membutuhkannya (migrasi data), sambil memanfaatkan fleksibilitas dan kecepatan di area lain (pengembangan fitur baru).
- "**Pasar Kreatif Lokal**" membawa kita ke ekstrim kecepatan dan adaptabilitas. Dalam lingkungan startup, metodologi seperti **Extreme Programming (XP)** menunjukkan bagaimana disiplin rekayasa yang ketat (seperti pengembangan berbasis tes dan integrasi berkelanjutan) justru menjadi pondasi yang memungkinkan kecepatan dan fleksibilitas, bukan menghambatnya.

Secara kolektif, kelima kasus ini menggarisbawahi sebuah kebenaran fundamental dalam pengembangan sistem: keberhasilan lebih sering ditentukan oleh faktor-faktor manusia dan proses, seperti komunikasi yang efektif, keterlibatan pemangku kepentingan yang bermakna, dan manajemen

ekspektasi yang realistis, daripada oleh kecanggihan teknologi yang digunakan semata.

Masa Depan Analisis dan Perancangan Sistem

Disiplin analisis dan perancangan sistem informasi berada dalam keadaan evolusi yang konstan, didorong oleh gelombang inovasi teknologi yang tak henti-hentinya. Jika studi kasus dalam suplemen ini mencerminkan paradigma masa lalu dan masa kini, maka masa depan akan dibentuk oleh tren yang lebih transformatif lagi.

Kemunculan **Kecerdasan Buatan (AI) dan Pembelajaran Mesin (Machine Learning)** mengubah peran sistem dari sekedar alat pelaporan menjadi mitra prediktif dan preskriptif. Analis sistem di masa depan tidak hanya akan merancang alur kerja untuk manusia, tetapi juga untuk agen-agen AI, dan harus mampu mendefinisikan kebutuhan untuk sistem yang dapat belajar dan beradaptasi secara otonom.

Ledakan **Big Data** dan **Internet of Things (IoT)** secara fundamental mengubah skala dan sifat data yang harus dikelola. Analisis dan perancangan tidak lagi hanya berfokus pada data transaksional yang terstruktur, tetapi juga pada aliran data *real-time* yang masif dan tidak terstruktur dari sensor dan perangkat yang saling terhubung. Ini menuntut pemahaman tentang arsitektur data baru, seperti
data lake dan *streaming analytics*.

Pergeseran yang tak terelakkan menuju **komputasi awan (*cloud computing*)** terus berlanjut, membebaskan organisasi dari beban pengelolaan infrastruktur fisik dan memungkinkan skalabilitas

yang belum pernah terjadi sebelumnya. Analis sistem harus mahir dalam merancang solusi *cloud-native*, memahami model layanan (IaaS, PaaS, SaaS), dan menavigasi kompleksitas keamanan dan privasi data di lingkungan terdistribusi.

Pada akhirnya, peran analis sistem akan terus menjadi semakin strategis. Mereka tidak hanya akan menjadi penerjemah kebutuhan bisnis menjadi spesifikasi teknis, tetapi juga menjadi arsitek solusi digital yang inovatif, pemandu transformasi organisasi, dan penjaga etika dalam dunia yang semakin didorong oleh data. Kemampuan untuk belajar secara terus-menerus, beradaptasi dengan teknologi baru, dan tetap fokus pada penciptaan nilai bisnis akan menjadi kunci keberhasilan bagi para profesional di bidang yang dinamis ini.

ISBN 978-623-464-137-0 (PDF)

UMSIDA PRESS
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No. 666 B
Sidoarjo , Jawa Timur