- Marniyati H. Botutihe
- Sitti Najmia Rifai
- Taufik Rachman
- Heliawati Hamrul

Get Press Indonesia



ANALISIS DAN PERANCANGAN SISTEM INFORMASI: TEORI DAN APLIKASI



ISBN 978-634-255-789-1

ANALISIS DAN PERANCANGAN SISTEM INFORMASI: TEORI DAN APLIKASI

Marniyati H. Botutihe
Sitti Najmia Rifai
Taufik Rachman
Heliawati Hamrul

GET PRESS INDONESIA

ANALISIS DAN PERANCANGAN SISTEM INFORMASI: TEORI DAN APLIKASI

Penulis :
Marniyati H. Botutihe
Sitti Najmia Rifai
Taufik Rachman
Heliawati Hamrul

ISBN: 978-634-255-789-1

Editor : Mila Sari, M. Si.
Desain Sampul dan Tata Letak : Tri Putri Wahyuni, S. Pd.

PENERBIT : GET PRESS INDONESIA
Anggota IKAPI No. 033/SBA/2022
Jl. Palarik RT 01 RW 06 Kelurahan Air Pacah
Kecamatan Koto Tangah Padang Sumatera Barat
website: www.getpress.co.id
email: adm.getpress@gmail.com

Cetakan pertama, Oktober 2026

Hak cipta dilindungi undang-undang
Dilarang memperbanyak karya tulis ini dalam bentuk
dan dengan cara apapun tanpa izin tertulis dari penerbit.

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku berjudul *Analisis dan Perancangan Sistem Informasi: Teori dan Aplikasi* ini dapat diselesaikan dengan baik. Buku ini disusun sebagai pedoman komprehensif bagi mahasiswa, akademisi, maupun praktisi yang ingin memahami fondasi sistem informasi secara holistik. Dimulai dari pengenalan konsep dasar dan klasifikasi sistem informasi, analisis kebutuhan pengguna yang krusial, hingga teknik perancangan antarmuka (*UI/UX*) dan basis data yang terintegrasi, buku ini menyajikan alur pengembangan sistem secara terstruktur dan teruji.

Tidak hanya menyajikan landasan teoretis, buku ini melengkapinya dengan studi kasus nyata—seperti perancangan Sistem Informasi Perpustakaan—serta pembahasan mendalam mengenai pengujian (*system testing*) dan strategi implementasi di lapangan. Melalui pendekatan yang aplikatif ini, pembaca diharapkan mampu menguasai siklus hidup pengembangan sistem informasi (*System Development Life Cycle*) secara mendalam dan siap mengimplementasikannya dalam dunia kerja modern. Penulis menyadari bahwa karya ini tak lepas dari kekurangan, oleh karena itu kritik dan saran yang membangun sangat diharapkan demi penyempurnaan di masa mendatang.

Padang, September 2026

Penulis

DAFTAR ISI

KATA PENGANTAR ..... **i**
DAFTAR ISI ..... **ii**
DAFTAR GAMBAR ..... **ix**
DAFTAR TABEL ..... **xi**
BAB 1 PENGANTAR SISTEM INFORMASI **1**
1.1 Pengertian Sistem Informasi 1
1.2 Konsep Dasar Sistem Informasi 3
1.3 Sistem dan Subsitem 4
1.4 Jenis – Jenis Sistem Informasi 6
1.4.1 Sistem Pemrosesan Transaksi (TPS) 6
1.4.2 Sistem Pendukung Keputusan (SPK) 6
1.4.3 Sistem Informasi Eksekutif (EIS) 7
1.4.4 *Management Information Systems* (MIS) 7
1.4.5 Sistem Alur Kerja 7
1.4.6 *Enterprise Resource Planning* (ERP) 8
1.4.7 Sistem Pakar 8
1.5 Manfaat Sistem Informasi 8
1.6 Tujuan dan Fungsi Sistem Informasi 9
1.7 Peran Sistem Informasi 10
1.8 Tantangan dan Peluang Sistem Informasi 11
1.9 Klasifikasi dan Jenis-Jenis Sistem Informasi 11
DAFTAR PUSTAKA **14**
BAB 2 ANALISIS KEBUTUHAN SISTEM **17**
2.1 Konsep Dasar Analisis Kebutuhan Sistem 19
2.1.1 Kebutuhan Sistem 20
2.1.2 Tujuan Analisis Kebutuhan Sistem 26
2.1.3 Manfaat Analisis Kebutuhan Sistem 27

2.1.4 Ruang Lingkup Analisis Kebutuhan Sistem 30
2.2 Pengumpulan Data dan Informasi 31
2.2.1 Wawancara 33
2.2.2 Observasi 34
2.2.3 Analisis Dokumen 35
2.2.4 Integrasi Teknik Pengumpulan Data 36
2.3 Dokumentasi Spesifikasi Sistem 37
2.3.1 *Software Requirements Specification* (SRS) 39
2.3.2 Validasi Spesifikasi Sistem 41
2.4 Contoh Kasus Pengaplikasian Analisis Kebutuhan Sistem 42
DAFTAR PUSTAKA 54
BAB 3 PERANCANGAN ANTAR MUKA DAN BASIS DATA 63
3.1 Pentingnya Desain Antarmuka dalam Keberhasilan Sistem Informasi 63
3.2 Peran Basis Data sebagai Pusat Penyimpanan Informasi 64
3.3 Hubungan antara UI, Proses Bisnis, dan Database 65
3.4 Tantangan dalam Merancang Sistem Modern 66
3.6 Konsep Dasar Perancangan Antarmuka 67
3.6.1 Pengertian Antarmuka Pengguna 67
3.6.2 *User Interface* (UI) vs *User Experience* (UX) 69
3.6.3 Jenis-Jenis Antarmuka 70
3.7 Prinsip-Prinsip Perancangan Antarmuka 74
3.7.1 Prinsip Konsistensi (*Consistency*) 74
3.7.2 Kemudahan Penggunaan (*Usability*) 75
3.7.3 *Learnability* 76
3.7.4 *Visibility* 77
3.7.5 *Feedback* 77
3.7.6 *Error Prevention* 77
3.7.7 *Flexibility* 78

3.7.8 *Accessibility* 78
3.7.9 *Responsive Design* 79
3.8 Komponen Antarmuka Sistem Informasi 79
3.8.1 Layout 80
3.8.2 *Typography* 81
3.8.3 Warna 82
3.8.4 Ikon 83
3.8.5 Button 84
3.8.6 Form Input 85
3.8.7 *Navigation* 85
3.8.8 Dashboard 86
3.8.9 *Notification* 87
3.9 Proses Perancangan Antarmuka 88
3.9.1 Analisis Kebutuhan Pengguna 89
3.9.2 User Persona 90
3.9.3 *User Journey* 92
3.9.4 *Wireframe* 93
3.9.5 *Mockup* 94
3.9.6 *Prototype* 95
3.9.7 Evaluasi Prototype 96
3.10 Tool Perancangan Antarmuka 98
3.11 Pengujian Antarmuka 106
3.11.1 *Usability Testing* 107
3.11.2 *Heuristic Evaluation* 110
3.11.3 *User Acceptance Test* (UAT) 112
3.11.4 *Think Aloud Method* 114
3.11.5 Perbandingan Metode Pengujian Antarmuka 116
3.11.6 Integrasi Keempat Metode 117
3.11.7 Contoh Rancangan Pengujian Antarmuka 117

3.12 Konsep Dasar Basis Data 119
3.12.1 Pengertian Basis Data 119
3.12.2 Tujuan Basis Data 121
3.12.3 Komponen Basis Data 124
3.12.4 Karakteristik Basis Data 129
3.12.5 Jenis Basis Data 130
3.12.7 Hubungan Basis Data dengan Sistem Informasi 137
3.13 *Database Management System* (DBMS) 139
3.13.1 Pengertian DBMS 140
3.13.2 Fungsi DBMS 141
3.13.3 Arsitektur DBMS 145
3.12.6 Contoh DBMS 147
3.12.7 Perbandingan Contoh DBMS 148
3.13 Perancangan Basis Data 149
3.13.1 Analisis Kebutuhan Data 150
3.13.2 Identifikasi Entitas 152
3.13.3 Identifikasi Atribut 154
3.13.4 Primary Key 155
3.13.5 Foreign Key 155
3.13.6 *Candidate Key* 156
3.13.7 *Composite Key* 156
3.14 *Entity Relationship Diagram* (ERD) 157
3.14.1 Konsep EDR 157
3.14.2 Komponen ERD 158
3.14.3 Kardinalitas (*Cardinality*) 159
3.14.4 Contoh Perancangan ERD 160
3.15 Normalisasi Basis Data 164
3.15.1 *First Normal Form* (1NF) 166
3.15.2 *Second Normal Form* (2NF) 166

10.15.3 *Third Normal Form* (3NF) 167
3.16 Transformasi ERD ke Model Relasional 168
3.16.1 Aturan Transformasi ERD ke Model Relasional 169
3.16.2. Contoh Implementasi 170
3.17 Perancangan Kamus Data (*Data Dictionary*) 172
3.17.1 Pengertian Kamus Data 172
3.17.2 Fungsi Kamus Data 173
3.17.3 Struktur Kamus Data 175
3.17.4 Manfaat Kamus Data dalam Pengembangan SI 176
3.18 Perancangan Struktur Tabel 177
3.18.1 Penentuan Tipe Data 178
3.18.2 Constraint 181
3.18.3 *Default Value* 182
3.18.4 *Auto Increment* 183
3.18.5 Index 184
3.18.6 Unique 185
10.18.7 *Check Constraint* 186
3.19 Integrasi Antarmuka dengan Basis Data 188
3.19.1 Alur Input Data 190
3.19.2 Validasi Data 191
3.19.3 CRUD 192
3.19.4 Form Login 193
3.19.5 Form Master Data 195
3.19.6 Dashboard 196
3.19.7 Laporan 197
3.19.8 Integrasi Keseluruhan Antarmuka dan Basis Data 199
3.19.9 Integrasi Antarmuka Berdasarkan Hak Akses 200
3.19.10 Prinsip Keamanan Integrasi Antarmuka dan Database 200

3.19.11 Hubungan Antarmuka, Proses, dan Database 201
3.20 Studi Kasus Perancangan S.I. Perpustakaan 207
3.21.1 Gambaran Umum Studi Kasus 207
3.21.2 Analisis Kebutuhan 209
3.21.3 *Desain Wireframe* 213
3.21.4 *Prototype* 214
3.21.5 Perancangan ERD 216
3.21.6 Normalisasi 217
3.21.7 Struktur Tabel 221
3.21.8 Kamus Data 222
3.21.9 Implementasi Database 223
3.21.10 Contoh Data 229
3.21.11 Rancangan Antarmuka Utama 229
3.21.12 Dashboard Utama 230
3.21.13 Halaman Data Buku 230
3.21.14 Halaman Peminjaman 231
3.21.15 Halaman Pengembalian 231
3.21.16 Halaman Laporan 232
3.21.17 Integrasi Seluruh Hasil Perancangan 233
3.21.18 Contoh Alur Operasional Sistem 233
3.21.19 Kesimpulan Studi Kasus 235
DAFTAR PUSTAKA 237
BAB 4 PENGUJIAN SISTEM DAN IMPLEMENTASI 241
4.1 Pengujian Sistem 242
4.1.1 Konsep Dasar Pengujian Sistem 242
4.1.2 Prinsip-Prinsip Pengujian 244
4.1.3 Strategi dan Level Pengujian 245
4.1.4 Metode dan Teknik Pengujian Sistem 248
4.2 Implementasi Sistem 260

4.2.1 Strategi Implementasi Sistem 261
4.2.2 Pelatihan Pengguna, Dokumentasi dan Manual Pengguna 265
DAFTAR PUSTAKA 271
BIODATA PENULIS 273

DAFTAR GAMBAR

Gambar 1. 1 Komponen Utama Sistem Informasi 2
Gambar 2. 1 Ruang Lingkup Analisis Kebutuhan Sistem 31
Gambar 2. 2 Pengumpulan Data dan Informasi 33
Gambar 3. 1 Contoh Perancangan Antarmuka 106
Gambar 3. 2 Komponen Basis Data 125
Gambar 3. 3 *Database Management System* 140
Gambar 3. 4 *Entity Relationship Diagram* 162
Gambar 3. 5 Proses Normalisasi Data 168
Gambar 3. 6 Implementasi ERD 171
Gambar 3. 7 Arsitektur Integrasi 189
Gambar 3. 8 Alur Input Data 191
Gambar 3. 9 Validasi Data 192
Gambar 3. 10 Operasi CRUD 193
Gambar 3. 11 Proses Login 194
Gambar 3. 12 Alur Login 195
Gambar 3. 13 Form Master Data 196
Gambar 3. 14 Dashboard Sistem Informasi 197
Gambar 3. 15 Sistem Pelaporan 199
Gambar 3. 16 Integrasi Keseluruhan 199
Gambar 3. 17 Interaksi Tiga Lapisan Utama 202
Gambar 3. 18 Hubungan Komponen Sistem Informasi Akademik 207
Gambar 3. 19 Konsep Sistem Informasi Perpustakaan 209
Gambar 3. 20 Aktor Sistem 210
Gambar 3. 21 Contoh *Wireframe* Sistem Perpustakaan 213
Gambar 3. 22 Contoh Rancangan 214
Gambar 3. 23 Perbedaan Wireframe dan Prototype 215
Gambar 3. 24 Perbedaan Wireframe dan Prototype 215
Gambar 3. 25 ERD Sistem Informasi Perpustakaan 216
Gambar 3. 26 Relalsi Antar entitas Sistem Informasi Perpustakaan 217
Gambar 3. 27 Proses Normalisasi 1NF 219
Gambar 3. 28 Rancangan Login 230
Gambar 3. 29 Dashboard Perpustakaan 230
Gambar 3. 30 Antarmuka Data Buku 231
Gambar 3. 31 Antarmuka Transaksi Peminjaman 231
Gambar 3. 32 Rancangan Laporan 232

Gambar 3. 33 Alur Lengkap Perancangan Sistem Informasi Perpustakaan 233
Gambar 4. 1 *Workflow* Pengujian Sistem 246

DAFTAR TABEL

Tabel 2. 1 Informasi Dokumen 42
Tabel 2. 2 Definis dan Istilah 43
Tabel 2. 3 Karakteristik Pengguna 45
Tabel 2. 4 Kebutuhan Fungsional 46
Tabel 2. 5 Spesifikasi Atribut dan Luaran Sistem 48
Tabel 2. 6 Kebutuhan Nonfungsional 48
Tabel 2. 7 Ketentuan Akses Sistem 50
Tabel 2. 8 Komponen Antarmuka Sistem 50
Tabel 2. 9 Atribut Data 51
Tabel 3. 1 Perbedaan Mendasar Antara UI dan UX 69
Tabel 3. 2 Kamus Data Anggota 222
Tabel 3. 3 Kamus Data Buku 223
Tabel 4. 1 Level Pengujian 247
Tabel 4. 2 Contoh Pengujian BVA 251
Tabel 4. 3 Contoh Pengujian *Decision Table* 254
Tabel 4. 4 Contoh *Use Case "Place an Order"* 256

BAB 1
PENGANTAR SISTEM INFORMASI

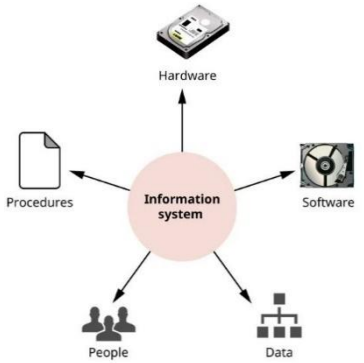
Oleh Marniyati H. Botutihe

1.1 Pengertian Sistem Informasi

Sistem informasi ialah seperangkat komponen yang saling terhubung yang mengintegrasikan pengumpulan, pengolahan, penyimpanan dan distribusi data, informasi dan produk digital untuk mendukung pengambilan keputusan, koordinasi, pengendalian, analisis dan visualisasi dalam sebuah organisasi.

Sistem informasi adalah kombinasi perangkat keras, perangkat lunak, dan jaringan telekomunikasi yang dibuat untuk mengumpulkan, menghasilkan, dan mendistribusikan data yang bermanfaat, pada suatu organisasi. Sistem ini menentukan alur informasi di dalamnya. Tujuan sistem informasi ialah menyediakan informasi relevan bagi pengguna, serta mengumpulkan dan mengolah data lalu mengkomunikasikan informasi tersebut kepada pengguna sistem.

Sistem informasi umunya terdiri dari lima komponen utama yaitu perangkat keras, perangkat lunak, data, manusia dan prosedur.



Gambar 1. 1 Komponen Utama Sistem Informasi

(Sumber : Raisinghani, M. S. (2025). Foundations of information systems.)

Perangkat fisik seperti komputer, server, jaringan, dan perangkat penyimpanan yang digunakan untuk mengumpulkan, memproses, dan menyimpan data disebut perangkat keras atau hardware. Program dan aplikasi yang berjalan pada perangkat keras, yang memungkinkan pengguna melakukan tugas-tugas tertentu, disebut perangkat lunak atau software. Perangkat lunak dapat mencakup berbagai hal, mulai dari sistem operasi dan sistem manajemen basis data hingga aplikasi bisnis khusus. akta dan angka mentah yang diproses dan diubah menjadi informasi yang bermakna disebut data. Bukti nyata yang kita gunakan untuk mempelajari dan memahami orang, tempat, dan berbagai hal membentuk informasi. Informasi ialah data mentah yang sudah diproses dan dimanipulasi untuk memberikan konteks dan makna. Setelah data diproses menjadi informasi, kita bisa memakai informasi tersebut untuk keperluan pribadi maupun profesional. Sekumpulan instruksi dan aturan yang mengatur penggunaan komponen perangkat keras, perangkat lunak, dan data dikenal sebagai prosedur. Prosedur operasi standar menjamin konsistensi dan keandalan dalam penggunaan sistem informasi. Individu

yang menggunakan sistem informasi termasuk pengguna akhir yang memasukkan dan mengambil data dalam sistem, serta profesional teknologi informasi yang merancang, mengembangkan, dan memelihara sistem merupakan pihak-pihak yang membentuk sistem informasi tersebut. (Raisinghani, M. S.,).

1.2 Konsep Dasar Sistem Informasi

Isi pikiran manusia dapat diklasifikasikan ke dalam tiga kategori. (Russell Ackoff)

1. **Data** merepresentasikan fakta atau pernyataan mengenai suatu peristiwa yang tidak terkait dengan hal-hal lain. Data umumnya berkaitan dengan fakta-fakta konkret. Data dapat berupa simbol matematika atau teks yang digunakan untuk mengidentifikasi, mendeskripsikan, atau merepresentasikan sesuatu, seperti suhu atau identitas seseorang. Data itu sendiri sekadar ada dan tidak memiliki makna di luar keberadaannya itu. Data dapat hadir dalam berbagai bentuk, baik yang dapat digunakan maupun tidak, serta tersedia dalam beragam format seperti teks, gambar, suara, atau bahkan video.
2. **Informasi** adalah data yang dipadukan dengan makna. Informasi mencakup pemahaman mengenai suatu hubungan, seperti hubungan sebab-akibat (Watson R T.). Contoh: Suhu turun 15 derajat, lalu hujan mulai turun. Angka suhu 100 dapat memiliki makna yang berbeda jika dikaitkan dengan satuan Fahrenheit atau Celsius. Makna lebih lanjut dapat diperoleh jika tersedia konteks tambahan mengenai pembacaan suhu tersebut, misalnya informasi bahwa suhu itu berkaitan dengan cairan atau gas, ataupun perbandingannya dengan suhu normal musiman sebesar 20°. Dengan kata lain, informasi adalah data yang memiliki makna melalui keterkaitan

relasional. Menurut Ackoff, informasi merupakan data yang bermanfaat; informasi memberikan jawaban atas pertanyaan: "siapa," "apa," "di mana," dan "kapan."

3. **Pengetahuan** dapat dipandang sebagai informasi yang dipadukan dengan pengalaman, konteks, dan interpretasi. Pengetahuan merupakan tingkat semantik tambahan yang diperoleh dari informasi melalui suatu proses. Terkadang, proses ini bersifat observasional. Ackoff mendefinisikannya sebagai penerapan data dan informasi; pengetahuan memberikan jawaban atas pertanyaan "bagaimana". Sebagai contoh, apa yang terjadi pada cuaca dingin bagi pengelola pesawat terbang? Para ahli pengetahuan observasional menafsirkan kondisi dingin berdasarkan dampaknya, yaitu terbentuknya es pada pesawat yang dapat mengurangi gaya dorong aerodinamis serta berpotensi menghambat kinerja permukaan kendali pesawat. (Ackoff R L.)

1.3 Sistem dan Subsitem

Sebuah sistem dibentuk oleh serangkaian komponen terintegrasi yang saling terhubung demi mencapai target tertentu. Istilah elemen sistem mengacu pada komponen-komponen penyusun yang membentuk struktur utama sistem. Secara umum, struktur elemen sistem mencakup masukan (*input*), pemrosesan (*process*), penyimpanan (*storage*), keluaran (*output*), pengendalian (*control*), tujuan (*goal*), serta umpan balik (*feedback*). Sebuah sistem dibentuk oleh kumpulan subsistem yang saling terhubung dan bekerja sama untuk mencapai tujuan akhir. Meskipun setiap subsistem (seperti CPU, masukan, keluaran, dan penyimpanan pada sistem komputer) memiliki tugas serta batasan masing-masing, semuanya tetap berinteraksi secara terpadu. (E. Widarti *et al*., 2024.).

1. Masukan (*Input*)

   Sistem menerima data dan informasi dari berbagai sumber, termasuk sensor, papan ketik, pemindai, dan basis data. Komponen ini memuat seluruh data atau sumber daya yang dimasukkan ke sistem untuk diolah. Input berfungsi sebagai bahan mentah utama bagi sistem sebelum menghasilkan keluaran (*output*).
2. Pemrosesan (*Proses*)

   Bagian sistem yang bertugas mengolah atau mentransformasi masukan menjadi bentuk keluaran. Tahapan ini mencakup serangkaian kegiatan pengolahan, analisis, serta manipulasi data. Sistem mengubah data mentah menjadi informasi yang bermanfaat menggunakan berbagai teknik seperti pengurutan, pengelompokan, penghitungan, analisis, dan sintesis.
3. Penyimpanan (*Storage*)

   Sistem menyimpan data yang telah diproses secara terstruktur dan aman, misalnya dalam basis data, sistem berkas, atau penyimpanan awan.
4. Keluaran (*Output*)

   Sistem menampilkan informasi kepada pengguna dalam bentuk seperti laporan, grafik, bagan, atau dasbor. Komponen berupa hasil akhir atau informasi yang didapat setelah proses pengolahan input selesai. Hasil keluaran ini dimanfaatkan sebagai acuan pengambilan keputusan atau disalurkan sebagai input bagi sistem lain.
5. Pengendalian (*Control*)

   Elemen yang mengawasi serta mengatur jalannya seluruh operasional sistem. Pengendali memastikan agar kinerja sistem tetap berada di jalur yang benar sesuai target yang ditetapkan.
6. Tujuan (*Goal*)

   Target atau hasil akhir yang ingin dicapai oleh keseluruhan jalannya sistem.

7. Umpan Balik (*Feedback*)
   Sistem meminta umpan balik dari pengguna dan pemangku kepentingan lainnya untuk menilai efektivitasnya serta menyempurnakan desain dan kinerjanya.

1.4 Jenis – Jenis Sistem Informasi

1.4.1 Sistem Pemrosesan Transaksi (TPS)

Sistem ini terutama digunakan untuk pencatatan data yang diperlukan oleh organisasi dalam menjalankan kegiatan usahanya. Contoh TPS meliputi pencatatan pesanan penjualan, penggajian, dan catatan pengiriman. TPS digunakan untuk menghasilkan laporan secara berkala sesuai jadwal, serta untuk membuat laporan berdasarkan permintaan maupun laporan pengecualian.

1.4.2 Sistem Pendukung Keputusan (SPK)

Sistem ini melayani manajemen organisasi. Sistem pendukung keputusan (DSS) memiliki perangkat analisis data canggih yang mendukung serta membantu seluruh aspek pengambilan keputusan terkait masalah tertentu. DSS dapat memanfaatkan data dari sumber eksternal, seperti harga saham terkini, untuk meningkatkan kualitas pengambilan keputusan. DSS digunakan ketika menghadapi masalah yang kompleks, di mana informasi yang diperlukan untuk mengambil keputusan terbaik sulit diperoleh dan dimanfaatkan. DSS dikembangkan dengan melibatkan para pengambil keputusan di dalam organisasi. Sistem ini membantu proses pengambilan keputusan yang tepat, namun tidak mengambil keputusan itu sendiri.

1.4.3 Sistem Informasi Eksekutif (EIS)

Sistem ini juga disebut sebagai *Executive Support System* (Sistem Pendukung Eksekutif). EIS digunakan oleh para manajer senior dalam suatu organisasi. Oleh karena itu, sistem ini harus mudah digunakan agar para eksekutif dapat mengoperasikannya tanpa bantuan pihak lain. EIS mampu melakukan analisis tren dan pelaporan pengecualian, serta memiliki kemampuan *drill-down* (penelusuran data secara mendalam). Hasilnya biasanya disajikan dalam bentuk grafis yang disesuaikan dengan kebutuhan informasi eksekutif. EIS dilengkapi dengan perangkat analisis daring serta mengakses beragam data internal maupun eksternal.

1.4.4 *Management Information Systems* (MIS)

Sistem ini menyajikan ringkasan rutin mengenai operasional dasar organisasi. Berbagai layanan penting dicatat oleh sistem pemrosesan transaksi (TPS) organisasi, sementara sistem informasi manajemen (MIS) mengonsolidasikan data terkait penjualan, produksi, dan sebagainya. MIS menyediakan informasi rutin bagi para manajer dan pengambil keputusan. Tujuan utama penerapan MIS dalam organisasi adalah untuk meningkatkan efisiensi operasional. MIS dapat mendukung berbagai bidang seperti pemasaran, produksi, keuangan, dan lain-lain..

1.4.5 Sistem Alur Kerja

Sistem alur kerja adalah sistem manajemen berbasis aturan yang mengarahkan, mengoordinasikan, dan memantau pelaksanaan serangkaian tugas yang saling terkait, yang disusun untuk membentuk suatu proses bisnis. Terdapat tiga jenis perangkat lunak alur kerja.

1.4.6 *Enterprise Resource Planning* (ERP)

Sistem *Enterprise Resource Planning* (ERP) adalah perangkat lunak manajemen proses bisnis yang memungkinkan organisasi menggunakan sistem program terintegrasi yang mampu mengelola operasi bisnis vital perusahaan di seluruh organisasi global yang memiliki banyak lokasi.

1.4.7 Sistem Pakar

Sistem pakar dapat memberikan saran dan bertindak layaknya seorang ahli dalam bidang tertentu di suatu organisasi. Sistem pakar memiliki basis pengetahuan yang luas.

1.5 Manfaat Sistem Informasi

Berikut ini merupakan manfaat sistem informasi :

1. Meningkatkan Kinerja dan Produktivitas
   Teknologi modern dan sistem informasi dapat meningkatkan kinerja serta produktivitas perusahaan secara signifikan.
2. Inovasi dan Efisiensi Operasional
   Perusahaan memanfaatkan sistem informasi untuk meneliti dan mengembangkan cara baru dalam menghasilkan pendapatan, menjangkau pelanggan, serta menyederhanakan tugas-tugas yang memakan waktu.
3. Menghemat Waktu dan Biaya
   Sistem informasi membantu bisnis menghemat waktu dan uang sekaligus mendukung pengambilan keputusan yang lebih cerdas.
4. Mempermudah Komunikasi Antardepartemen
   Departemen internal perusahaan (seperti pemasaran dan penjualan) dapat saling berkomunikasi dan berbagi informasi dengan lebih mudah.

5. Mengurangi Kesalahan Manusia (*Human Error*)
   Karena otomatisasi dan algoritma kompleks, sistem informasi mengurangi risiko kesalahan manual. Karyawan dapat lebih fokus pada aspek inti bisnis ketimbang menghabiskan waktu mengumpulkan data, mengisi dokumen, atau melakukan analisis manual.
6. Akses Data Terpusat
   Anggota tim dapat mengakses data dalam jumlah besar melalui satu platform terpadu.
7. Pengolahan Data Multi-Sumber
   Mengumpulkan dan memproses data dari berbagai sumber (seperti vendor, pelanggan, gudang, dan agen penjualan) dapat dilakukan hanya dengan beberapa klik.

1.6 Tujuan dan Fungsi Sistem Informasi

Dikembangkannya sistem informasi dengan tujuan memperoleh data dan sekumpulan informasi. Sebuah sistem wajib melibatkan berbagai tipe atau jenis data yang bisa diolah untuk ditampilkan secara mudah pada para pengguna. untuk memperoleh informasi yang valid, wajib memperhatikan tiga faktor utama berikut ini:

1. *Accurate*, informasi tersebut harus tepat sasaran dan akurat.
2. *Relevance*, informasi tersebut harus relevan dengan target.
3. *Timeliness*, informasi tersebut harus efisien dan tepat waktu.

Fungsi sistem informasi yaitu :

1. Meningkatkan pengalaman dalam mengolah suatu sistem informasi secara logis dan kritis serta ketersediaan kualitas.
2. Bisa memberi aksesibilitas yang layak bagi para pengguna.

3. Dapat membuat analisa dan mengurangi potensi kerugian dalam segi ekonomi.
4. Bisa mencapai goals perusahaan dengan cepat berdasarkan dengan dukungan informasi yang akurat dan bisa dipertanggung jawabkan.

1.7 Peran Sistem Informasi

Peran sistem informasi sangat dibutuhkan dalam dunia kerja moderan guna membantu perkembangan bisnis dan organisasi. Dibawah ini peran utama sistem informasi

1. Mendukung digitalisasi bisnis

   Sekarang zaman sudah berkembang seluruh perusahaan melakukan transformasi digital. Kehadiran sistem informasi menjamin data bisnis bisa dikelola dengan baik mulai dari transaksi, pemasaran, hingga manajemen pelanggan.
2. Meningkatkan produktivitas

   Dengan otomatisasi proses melalui sistem informasi, perusahaan dapat meminimalisir kesalahan manusia dan meningkatkan kecepatan kerja. contohnya, sistem HRIS (*Human Resource Information System*) yang memudahkan perusahaan dalam mengelola data karyawan secara otomatis.
3. Menunjang Transformasi Organisasi

   Dengan kehadiran sistem informasi, suatu organisasi dapat menyesuaikan diri dengan mudah terhadap dinamika pasar. Contohnya, perusahaan logistik yang memakai fitur pelacakan secara real-time demi meningkatkan pengalaman dan kepuasan pelanggan.
4. Mendukung Analisis Data

   Zaman Big Data, sistem informasi dipakai untuk mengolah data besar menjadi informasi yang

bermanfaat. Hal ini penting bagi perusahaan dalam menyusun strategi bisnis yang lebih tepat sasaran.

5. Menghubungkan Teknologi dan Bisnis

   Alasan mengapa jurusan sistem informasi adalah pilihan menarik. Mahasiswa diajarkan bagaimana menghubungkan kebutuhan bisnis dengan solusi teknologi yang efektif. Hal ini membuat lulusan sistem informasi memiliki peran penting di dunia kerja modern.

1.8 Tantangan dan Peluang Sistem Informasi

Tantangan dalam sistem informasi yaitu ketergantungan terhadap Teknologi, Penggunaan sistem informasi menuntut kestabilan dan keamanan infrastruktur teknologi yang tinggi agar kinerja operasional tidak terganggu.

Peluang dalam sistem informasi yaitu Efisiensi dan Efektivitas Operasional, Penerapan sistem informasi mempercepat alur kerja, mengotomatisasi tugas rutin, serta menekan biaya operasional. Peningkatan Kualitas Keputusan, Akses data secara konsisten dan real-time memudahkan manajemen dalam menganalisis informasi secara akurat untuk menentukan langkah strategis. Perluasan Jangkauan Pasar, Teknologi berbasis sistem informasi memungkinkan bisnis menjangkau pelanggan secara lebih luas melalui platform digital tanpa terbatas geografis.

1.9 Klasifikasi dan Jenis-Jenis Sistem Informasi

Dalam memahami sistem, penting untuk disadari bahwa bentuk sistem tidak tunggal. Setiap sistem memiliki karakteristik serta struktur tersendiri yang dipengaruhi oleh ruang lingkup, tujuan, dan proses kerja di dalamnya. Oleh karena itu, pengelompokan sistem ke dalam beberapa kategori sangat diperlukan untuk mempermudah analisis, perancangan, dan pengelolaannya. Klasifikasi ini juga menjadi fondasi dalam pengembangan sistem informasi agar

sistem yang dibangun benar-benar sesuai dengan kebutuhan lingkungan serta dapat menyelesaikan permasalahan yang dihadapi.

1. Klasifikasi sistem fisik dan sistem abstrak Secara garis besar, sistem dibedakan menjadi dua bentuk utama, yaitu sistem fisik dan sistem abstrak. Sistem fisik merupakan sistem yang berwujud nyata, dapat dilihat, dan diamati secara langsung.
2. Sistem juga dapat dikelompokkan berdasarkan asal mula terbentuknya, yaitu sistem alami dan sistem buatan manusia. Sistem alami terbentuk secara otomatis oleh proses alam tanpa campur tangan manusia, seperti sistem cuaca maupun ekosistem. Sementara itu, sistem buatan manusia sengaja dirancang dan dibangun untuk memenuhi kebutuhan serta tujuan tertentu, contohnya sistem pendidikan, sistem informasi, hingga sistem transportasi di kawasan perkotaan.
3. Berdasarkan tingkat kepastian cara kerjanya, sistem dibedakan menjadi sistem deterministik dan sistem probabilistik. Sistem deterministik beroperasi dengan pola teratur sehingga hasil akhirnya dapat diperkirakan secara akurat, seperti kalkulasi otomatis pada mesin kasir. Sebaliknya, sistem probabilistik mengandung ketidakpastian akibat pengaruh faktor eksternal atau variabel yang acak, sebagaimana terlihat pada sistem perkiraan cuaca atau strategi pemasaran.
4. Klasifikasi sistem deterministik dan probabilistik, Klasifikasi ini didasarkan pada tingkat kepastian perilaku dan hasil kerja sistem. Sistem deterministik beroperasi menurut aturan yang pasti dan konsisten, sehingga keluarannya dapat diprediksi secara akurat, seperti pada perhitungan matematis otomatis mesin kasir. Di sisi lain, sistem probabilistik memiliki tingkat ketidakpastian tinggi karena dipengaruhi oleh berbagai kondisi ekstrinsik atau variabel acak, contohnya pada

sistem ramalan cuaca maupun perencanaan strategi pemasaran.

DAFTAR PUSTAKA

Abdul Kadir. (2003). *Pengenalan Sistem Informasi*. Yogyakarta: ANDI. Data bibliografis buku ini tercatat sebagai karya Abdul Kadir.

A. O'Brien and G. M. Marakas, *Management Information Systems*, 10th ed. New York, NY, USA: McGraw-Hill/Irwin, 2011.

Chapter 1 *Introduction to Information System NOTES 0*.pdf," Scribd. Available: https://www.scribd.com/document/437176446/Chapter-1-Introduction-to-Information-System-NOTES-0-pdf

E. Widarti *et al*., *Buku Ajar Pengantar Sistem Informasi*, Efitra, Ed. Jambi, Indonesia: PT. Sonpedia Publishing Indonesia, 2024.

K. C. Laudon dan J. P. Laudon, *Management Information Systems: Managing the Digital Firm*, ed. ke-16. Upper Saddle River, NJ, USA: Pearson, 2020.

L. Zemmouchi-Ghomari*, "The basic concepts of information systems,"* in Contemporary Issues in Information Systems - a Global Perspective, IntechOpen, 2021. [Online]. Available: https://www.researchgate.net/publication/353381296_the_basic_concepts_of_information_systems. doi:10.5772/intechopen.97644

M. Y. Fathoni, I. Admirani, F. N. Afiana, T. Istanto, A. Bardadi, D. A. Prabowo, dan Tarwoto, *Pengantar Sistem Informasi*. Wawasan Ilmu.

Raisinghani, M. S. (2025). *Foundations of information systems*

R. K. Rainer dan B. Prince, *Introduction to Information Systems: Supporting and Transforming Business*, ed. ke-8. Hoboken, NJ, USA: Wiley, 2020.

BAB 2
ANALISIS KEBUTUHAN SISTEM

Oleh Sitti Najmia Rifai

Perkembangan teknologi informasi telah mendorong organisasi untuk mengintegrasikan sistem digital ke dalam berbagai aktivitas operasional, pelayanan, pengambilan keputusan, dan pengelolaan data. Sistem informasi tidak lagi hanya berfungsi sebagai alat bantu administratif, tetapi telah menjadi bagian penting dalam mendukung pencapaian tujuan organisasi. Akan tetapi, keberhasilan penerapan suatu sistem tidak hanya ditentukan oleh kecanggihan teknologi yang digunakan. Sistem yang dibangun harus mampu menyelesaikan permasalahan nyata, mendukung proses bisnis, dan memenuhi kebutuhan pengguna. Oleh karena itu, analisis kebutuhan sistem menjadi tahapan penting yang harus dilakukan sebelum sistem dirancang dan dikembangkan.

Analisis kebutuhan merupakan tahapan krusial dalam rekayasa sistem untuk menjamin keselarasan antara fungsi perangkat lunak dengan kebutuhan riil pengguna serta tujuan organisasi (Allan, 2021; Sandfreni, dkk. 2021). Kegagalan dalam merumuskan spesifikasi kebutuhan yang akurat sering menjadi akar penyebab ketidakefektifan proyek, sehingga pendekatan sistematis sangat diperlukan untuk meminimalisir risiko kegagalan pengembangan (Baso, dkk. 2024). Permasalahan utama yang sering dihadapi organisasi atau perusahaan saat ini adalah ketidakselarasan antara sistem informasi yang berjalan dengan strategi bisnis, yang secara signifikan berdampak pada

penurunan efektivitas pengambilan keputusan dan daya saing (Ariati, dkk. 2026). Kebutuhan yang tidak terdefinisi dengan baik cenderung memicu ketidakefisienan operasional serta kesenjangan fungsional yang menghambat pencapaian tujuan organisasi (Fani dkk, 2026).

Kebutuhan sistem umumnya dibedakan menjadi kebutuhan fungsional dan nonfungsional. Kebutuhan fungsional menjelaskan layanan atau aktivitas yang harus dilakukan oleh sistem, sedangkan kebutuhan nonfungsional berkaitan dengan kualitas sistem, seperti keamanan, kinerja, keandalan, dan kemudahan penggunaan. Penggalian kebutuhan dapat dilakukan melalui wawancara, observasi, diskusi, dan analisis dokumen. (Akram *et al*. 2024) menjelaskan bahwa proses elisitasi kebutuhan meliputi identifikasi pemangku kepentingan, pengumpulan kebutuhan, analisis, dan penentuan prioritas kebutuhan. Kebutuhan yang tidak lengkap, ambigu, atau saling bertentangan dapat menyebabkan kesalahan pengembangan, perubahan berulang, peningkatan biaya, dan keterlambatan proyek. Karena itu, kebutuhan harus dirumuskan secara jelas, konsisten, relevan, dapat ditelusuri, dan dapat diuji. (Umar dan Lano, 2024) menegaskan bahwa rekayasa kebutuhan yang sistematis membantu mengurangi kesalahan, memperkuat komunikasi antarpemangku kepentingan, dan meningkatkan peluang keberhasilan pengembangan sistem.

Dengan demikian, analisis kebutuhan sistem menjadi landasan utama untuk memastikan bahwa sistem yang dikembangkan mampu menyelesaikan permasalahan dan memenuhi kebutuhan penggunanya. Bab ini membahas konsep kebutuhan sistem, kebutuhan fungsional dan nonfungsional, teknik pengumpulan data, analisis, dokumentasi, prioritas, dan validasi kebutuhan sebagai dasar perancangan sistem yang tepat guna.

2.1 Konsep Dasar Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan tahap awal dalam pengembangan sistem yang bertujuan untuk memahami permasalahan, tujuan organisasi, kebutuhan pengguna, serta batasan yang harus diperhatikan. Tahap ini diperlukan agar sistem yang dibangun tidak sekadar berfungsi secara teknis, tetapi juga mampu mendukung proses bisnis dan memberikan solusi terhadap persoalan yang dihadapi pengguna (*User*). Analisis kebutuhan sistem merupakan proses sistematis untuk mengidentifikasi, mendokumentasikan, dan memvalidasi layanan atau fitur yang harus disediakan oleh perangkat lunak. Proses ini bertujuan memastikan bahwa sistem yang dikembangkan mampu menyelesaikan permasalahan pengguna sekaligus mendukung pencapaian tujuan organisasi (Faridiana, dkk 2024, p. 3; Liliana, dkk. 2021). Dengan demikian, analisis kebutuhan tidak hanya menghasilkan daftar fitur, tetapi juga memberikan gambaran tentang proses kerja dan hasil yang diharapkan dari sistem.

Secara konseptual, analisis kebutuhan berfungsi sebagai penghubung antara kebutuhan pengguna dan rancangan teknis sistem. Keinginan pengguna yang masih bersifat umum perlu diterjemahkan menjadi spesifikasi yang jelas, terstruktur, dan dapat diuji oleh tim pengembang (Kaur & Kaur, 2024; Noftianto & Kusuma, 2021). Spesifikasi tersebut mencakup kebutuhan fungsional, seperti layanan yang harus disediakan sistem, serta kebutuhan nonfungsional, seperti keamanan, kinerja, keandalan, dan kemudahan penggunaan.

Keberhasilan analisis kebutuhan sangat dipengaruhi oleh keterlibatan pemangku kepentingan sejak tahap awal pengembangan. Informasi dapat dikumpulkan melalui teknik elisitasi, seperti wawancara, observasi, diskusi kelompok, kuesioner, dan analisis dokumen (Kusuma, dkk. 2021; Noftianto & Kusuma, 2021). Keterlibatan pengguna

membantu mengurangi kesalahan penafsiran dan memastikan bahwa kebutuhan yang dirumuskan sesuai dengan kondisi operasional. Hasil analisis tersebut kemudian digunakan sebagai dasar dalam perancangan, pengembangan, dan pengujian sistem.

2.1.1 Kebutuhan Sistem

Secara konseptual, kebutuhan sistem didefinisikan sebagai sekumpulan spesifikasi teknis dan fungsional yang mendeskripsikan apa yang harus dilakukan oleh sistem untuk memenuhi kebutuhan informasi pengguna serta menyelesaikan masalah operasional organisasi (Aman & Suroso, 2021; Bangsawan, 2021). Kebutuhan ini tidak sekadar mencerminkan keinginan subjektif pengguna, melainkan harus diterjemahkan ke dalam batasan teknis yang selaras dengan tujuan strategis organisasi (Amalia, dkk. 2023; Faridiana, dkk. 2024). Secara umum, kebutuhan sistem dikategorikan menjadi dua jenis utama, yakni kebutuhan fungsional dan kebutuhan non-fungsional (Manis, dkk. 2021; Permadi & Saurina, 2022; Syukron, dkk. 2022).

1. Kebutuhan Fungsional

Kebutuhan fungsional (*functional requirement*) menggambarkan fungsi utama yang harus dijalankan oleh sistem untuk memenuhi kebutuhan pengguna dan mendukung pencapaian tujuan organisasi. Perumusannya tidak hanya memuat daftar fitur, tetapi juga menjelaskan cara sistem merespons tindakan pengguna, mengelola data, dan menerapkan ketentuan organisasi. Kebutuhan fungsional yang dirumuskan secara jelas membantu pengembang memahami proses kerja sistem secara menyeluruh sekaligus memudahkan pengguna memastikan kesesuaian sistem dengan kebutuhan nyata. Secara umum, kebutuhan fungsional meliputi

proses sistem, alur data, dan aturan bisnis (Rafildo, dkk. 2022; Kaur & Kaur, 2024).

- **Proses Sistem :** Proses sistem menjelaskan berbagai fitur dan layanan yang harus disediakan agar sistem dapat membantu pengguna menyelesaikan pekerjaannya. Setiap proses dirumuskan berdasarkan aktivitas nyata pengguna, seperti memasukkan data, mencari informasi, memperbarui data, mencetak laporan, dan menghapus data yang tidak lagi diperlukan. Proses sistem juga perlu menjelaskan respons yang diberikan sistem setelah pengguna melakukan suatu tindakan. Sebagai contoh, ketika pengguna mengisi formulir pendaftaran, sistem harus memeriksa kelengkapan data, menyimpan data yang valid, dan menampilkan pemberitahuan bahwa pendaftaran berhasil dilakukan. Dengan demikian, proses sistem tidak hanya menentukan daftar fitur, tetapi juga menggambarkan cara setiap fitur bekerja sesuai dengan kebutuhan pengguna (Pamungkas & Anggoro, 2023; Tanjung & Widodo, 2024).

- **Alur Data :** Alur data menjelaskan perjalanan data sejak dimasukkan, diperiksa, diproses, dan disimpan hingga menghasilkan keluaran yang dibutuhkan pengguna. Data dapat berasal dari pengguna, perangkat lain, maupun sistem yang saling terhubung. Sebelum disimpan, data perlu diperiksa untuk memastikan kelengkapan, kebenaran format, dan kesesuaiannya dengan ketentuan yang berlaku. Selanjutnya, sistem mengolah data tersebut

menjadi informasi yang berguna, seperti laporan, ringkasan, grafik, atau pemberitahuan. Misalnya, data transaksi yang dimasukkan pengguna akan diverifikasi, disimpan dalam basis data, dihitung, dan ditampilkan kembali dalam bentuk laporan transaksi. Pengaturan alur data yang jelas diperlukan untuk menjaga ketepatan, konsistensi, dan keamanan informasi yang dihasilkan oleh sistem (Wijaya & Susanto, 2021; Kaur & Kaur, 2024).

- **Aturan Bisnis :** Aturan bisnis merupakan ketentuan yang mengarahkan sistem agar bekerja sesuai dengan kebijakan, prosedur, dan alur kerja organisasi. Aturan ini menentukan tindakan yang dapat dilakukan pengguna, persyaratan yang harus dipenuhi, serta kondisi tertentu yang perlu ditangani oleh sistem. Sebagai contoh, administrator memiliki kewenangan untuk menambah, mengubah, dan menghapus data, sedangkan pengguna biasa hanya dapat melihat atau memasukkan data sesuai dengan tugasnya. Contoh lainnya, sistem hanya dapat memproses pengajuan apabila seluruh dokumen persyaratan telah dinyatakan lengkap. Penerapan aturan bisnis memastikan fungsi sistem tidak hanya berjalan secara teknis, tetapi juga sesuai dengan tata kelola dan kebutuhan organisasi. Oleh karena itu, aturan bisnis harus dirumuskan secara jelas, konsisten, dan dapat diuji agar sistem mampu mendukung kegiatan organisasi secara tertib, aman, dan efektif (Mutiari & Wahyuningtyas, 2021; Sukmawati, dkk. 2024).

2. Kebutuhan Non Fungsional

Kebutuhan nonfungsional (*non-functional requirement*) menjelaskan standar kualitas dan batasan yang harus dipenuhi agar sistem dapat beroperasi secara optimal. Jika kebutuhan fungsional menjelaskan fungsi yang dikerjakan sistem, kebutuhan nonfungsional menekankan kualitas sistem ketika menjalankan fungsi tersebut. Perumusan kebutuhan ini membantu memastikan bahwa sistem tidak hanya berfungsi dengan benar, tetapi juga memiliki kinerja yang baik, aman, andal, dan dapat dijalankan pada lingkungan teknologi yang sesuai. Secara umum, kebutuhan nonfungsional mencakup kinerja, keamanan, keandalan, dan kompatibilitas sistem (Almufareh & Humayun, 2023; Melén, dkk. 2022).

- **Kinerja (*Performance*) :** Kinerja menunjukkan tingkat kecepatan dan efisiensi sistem dalam memproses permintaan pengguna dengan memanfaatkan sumber daya yang tersedia. Kebutuhan kinerja dapat ditentukan berdasarkan waktu respons, jumlah transaksi yang mampu diproses, kapasitas penyimpanan, penggunaan memori, dan jumlah pengguna yang dapat mengakses sistem secara bersamaan. Sebagai contoh, sistem dapat dipersyaratkan untuk menampilkan halaman dalam waktu maksimal tiga detik pada kondisi penggunaan normal atau melayani sejumlah pengguna secara bersamaan tanpa mengalami penurunan kecepatan yang berarti. Rumusan yang spesifik dan terukur membantu pengembang memilih teknologi, merancang basis data, dan menentukan kapasitas server

yang sesuai dengan beban kerja sistem (Melén, dkk. 2022).

- **Keamanan (*Security*) :** Keamanan berkaitan dengan kemampuan sistem dalam melindungi data, informasi, dan layanan dari akses, penggunaan, perubahan, pengungkapan, atau penghapusan oleh pihak yang tidak berwenang. Kebutuhan keamanan mengatur mekanisme autentikasi pengguna, pembagian hak akses berdasarkan peran, enkripsi data, pencatatan aktivitas, pencadangan data, serta perlindungan terhadap berbagai ancaman keamanan. Misalnya, administrator dapat diberikan akses untuk mengelola seluruh data, sedangkan pengguna biasa hanya dapat mengakses informasi yang berkaitan dengan tugasnya. Sistem juga dapat dirancang untuk mengunci akun setelah lima kali percobaan masuk yang gagal. Penerapan kebutuhan keamanan sangat penting, terutama pada sistem yang mengelola data pribadi, keuangan, dan informasi organisasi yang bersifat sensitif (Wouters, dkk. 2022).
- **Keandalan (*Realibility*) :** Keandalan menjelaskan kemampuan sistem untuk tetap beroperasi secara stabil, konsisten, dan benar dalam jangka waktu tertentu, termasuk ketika terjadi kesalahan atau gangguan. Sistem yang andal seharusnya mampu meminimalkan kegagalan, menjaga keutuhan data, memulihkan layanan setelah gangguan, dan memberikan hasil yang konsisten pada kondisi penggunaan yang sama. Sebagai contoh, apabila jaringan terputus saat pengguna memasukkan data, sistem perlu mencegah terjadinya data ganda atau kehilangan data dan

memberikan pemberitahuan yang mudah dipahami. Keandalan dapat dirumuskan melalui tingkat ketersediaan sistem, frekuensi kegagalan, waktu pemulihan, serta mekanisme pencadangan dan pemulihan data. Dengan demikian, pengguna tetap dapat mengandalkan sistem dalam mendukung pekerjaan penting organisasi (Almufareh & Humayun, 2023; Melén, dkk. 2022).

- **Kompatibilitas (*Compatibility*) :** Kompatibilitas merupakan kemampuan sistem untuk beroperasi pada perangkat keras, perangkat lunak, sistem operasi, peramban, jaringan, atau platform tertentu. Kebutuhan ini perlu dirumuskan agar sistem dapat digunakan pada lingkungan teknologi yang tersedia di organisasi dan mampu berinteraksi dengan sistem lain apabila diperlukan. Sebagai contoh, aplikasi berbasis web harus dapat diakses dengan baik melalui komputer dan telepon pintar serta berjalan pada beberapa jenis peramban tanpa kehilangan fungsi utama. Kompatibilitas juga mencakup kesesuaian sistem dengan kapasitas prosesor, memori, ruang penyimpanan, versi sistem operasi, format data, dan perangkat pendukung lainnya. Penentuan persyaratan kompatibilitas sejak awal dapat mengurangi kendala teknis ketika sistem diterapkan dan memastikan bahwa layanan dapat digunakan secara konsisten pada perangkat yang berbeda (Almufareh & Humayun, 2023).

2.1.2 Tujuan Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan menghasilkan pemahaman yang sama antara pengguna, pemilik sistem, analis, dan pengembang mengenai sistem yang akan dibangun. Pemahaman tersebut diperlukan agar proses pengembangan tidak didasarkan pada perkiraan atau asumsi yang belum dikonfirmasi.

Analisis kebutuhan sistem merupakan proses terstruktur yang dilakukan untuk memperoleh pemahaman menyeluruh mengenai permasalahan, kebutuhan pengguna, serta tujuan pengembangan sistem. Proses ini dimulai dengan **mengidentifikasi masalah**, yaitu kegiatan untuk menemukan permasalahan yang terjadi pada sistem berjalan, memahami penyebabnya, serta menganalisis dampaknya terhadap organisasi. Tahap ini menjadi dasar agar sistem yang dikembangkan benar-benar mampu memberikan solusi terhadap kebutuhan nyata.

Tahap berikutnya adalah **memahami kebutuhan pengguna**. Pada tahap ini, analis melakukan penggalian informasi mengenai tugas, kepentingan, harapan, dan kendala yang dimiliki oleh pengguna maupun pemangku kepentingan. Pemahaman terhadap kebutuhan pengguna sangat penting agar sistem yang dibangun sesuai dengan proses bisnis dan mampu memberikan manfaat yang optimal bagi pihak yang menggunakannya. Setelah kebutuhan pengguna dipahami, proses dilanjutkan dengan **menentukan fungsi dan kualitas sistem**. Tahap ini bertujuan untuk menetapkan layanan atau fitur yang harus tersedia dalam sistem serta menentukan standar kualitas yang perlu dipenuhi, seperti aspek keamanan, kinerja, keandalan, kemudahan penggunaan, dan kemampuan pengembangan sistem di masa mendatang. Selanjutnya, dilakukan penentuan batas sistem untuk memperjelas ruang lingkup

pengembangan. Tahap ini menjelaskan bagian-bagian yang termasuk dalam sistem yang akan dibangun dan membedakannya dengan proses atau komponen yang berada di luar sistem. Penetapan batas sistem membantu menghindari perluasan ruang lingkup yang tidak terkendali selama proses pengembangan.

Tahap kelima adalah **menyelesaikan konflik kebutuhan**, yaitu proses menyelaraskan berbagai kebutuhan dari pengguna atau pemangku kepentingan yang terkadang memiliki perbedaan kepentingan maupun prioritas. Melalui proses komunikasi dan negosiasi, kebutuhan yang saling bertentangan dapat diselesaikan sehingga menghasilkan kesepakatan yang dapat diterapkan dalam sistem.

Tahap terakhir adalah **menyusun prioritas kebutuhan**. Pada tahap ini, setiap kebutuhan dievaluasi berdasarkan tingkat kepentingan, manfaat, risiko, biaya, serta sumber daya yang tersedia. Hasil dari seluruh tahapan analisis kebutuhan sistem adalah terciptanya **kesepahaman bersama antara pengguna, pengembang, dan pemilik sistem** yang menjadi dasar dalam proses perancangan dan pengembangan sistem agar berjalan efektif dan sesuai tujuan organisasi.

2.1.3 Manfaat Analisis Kebutuhan Sistem

Analisis kebutuhan sistem yang dilakukan secara sistematis memberikan manfaat besar bagi keberhasilan pengembangan perangkat lunak. Tahapan ini tidak hanya berfungsi untuk mengetahui fitur atau layanan yang harus disediakan oleh sistem, tetapi juga menjadi proses untuk membangun kesamaan pemahaman antara pengguna, organisasi, analis, dan pengembang mengenai tujuan serta harapan terhadap sistem yang akan dibuat. Dengan adanya analisis kebutuhan yang baik, proses pengembangan dapat berjalan lebih terarah karena

setiap keputusan yang diambil didasarkan pada kebutuhan nyata, bukan hanya asumsi dari pihak pengembang.

Salah satu manfaat utama analisis kebutuhan sistem adalah **mengurangi kemungkinan terjadinya kesalahan dalam proses pengembangan**. Kebutuhan yang telah diidentifikasi dan didokumentasikan dengan jelas membantu pengembang memahami fungsi, batasan, serta karakteristik sistem yang harus dibangun. Hal ini dapat mencegah terjadinya pembuatan fitur yang tidak diperlukan atau tidak sesuai dengan kebutuhan pengguna. Selain itu, analisis kebutuhan juga dapat mengurangi perubahan besar yang sering terjadi ketika sistem telah memasuki tahap pemrograman. Dengan memahami kebutuhan sejak awal, berbagai permasalahan dapat ditemukan dan diselesaikan lebih cepat sebelum membutuhkan biaya dan waktu tambahan.

Selain mendukung proses teknis pengembangan, analisis kebutuhan sistem juga berperan penting dalam **meningkatkan komunikasi antar pihak yang terlibat**. Dalam sebuah proyek perangkat lunak, setiap pihak memiliki sudut pandang dan kepentingan yang berbeda. Pengguna berfokus pada kebutuhan pekerjaan, sedangkan pengembang berfokus pada aspek teknis pembangunan sistem. Dokumen kebutuhan dan model sistem menjadi media komunikasi bersama yang membantu menyatukan pemahaman sehingga seluruh pihak memiliki gambaran yang sama mengenai sistem yang akan dikembangkan.

Manfaat lainnya adalah **membantu proses perencanaan dan pengelolaan proyek**. Kebutuhan sistem yang jelas memberikan dasar bagi tim pengembang untuk memperkirakan sumber daya yang diperlukan, seperti waktu pengerjaan, jumlah tenaga

kerja, biaya, teknologi yang digunakan, serta risiko yang mungkin muncul selama proses pengembangan. Dengan demikian, perencanaan proyek dapat dilakukan secara lebih realistis dan membantu organisasi dalam mengendalikan proses pengembangan agar tetap sesuai dengan tujuan yang telah ditetapkan.

Analisis kebutuhan juga menjadi **dasar penting dalam tahap perancangan dan pengujian sistem**. Hasil analisis digunakan oleh perancang untuk menentukan struktur, arsitektur, serta komponen sistem yang sesuai dengan kebutuhan pengguna. Sementara itu, tim penguji menggunakan dokumen kebutuhan sebagai acuan dalam membuat skenario pengujian untuk memastikan bahwa setiap fungsi sistem telah berjalan sesuai dengan spesifikasi yang telah ditentukan. Dengan kata lain, kebutuhan sistem menjadi pedoman yang menghubungkan seluruh tahapan dalam siklus pengembangan perangkat lunak.

Selain itu, analisis kebutuhan yang baik dapat meningkatkan tingkat penerimaan pengguna terhadap sistem yang dikembangkan. Sistem yang dibangun berdasarkan pemahaman terhadap kebutuhan dan permasalahan pengguna akan lebih mudah diterapkan karena mampu mendukung aktivitas kerja secara efektif. Keterlibatan pengguna dalam proses identifikasi kebutuhan juga membuat mereka merasa memiliki peran dalam pengembangan sistem sehingga meningkatkan kepercayaan dan penerimaan terhadap hasil akhir sistem.

Secara keseluruhan, analisis kebutuhan sistem merupakan fondasi penting dalam keberhasilan pengembangan perangkat lunak. Rekayasa kebutuhan yang dilakukan secara terstruktur mampu mengurangi risiko kegagalan proyek, meningkatkan kualitas komunikasi antar pemangku kepentingan, serta

memastikan sistem yang dihasilkan benar-benar memberikan manfaat bagi pengguna dan organisasi (Umar & Lano, 2024). Sebaliknya, kebutuhan yang tidak dianalisis secara tepat dapat menyebabkan berbagai permasalahan, seperti ketidaksesuaian sistem dengan harapan pengguna, meningkatnya biaya pengembangan, keterlambatan penyelesaian proyek, hingga kegagalan sistem dalam mencapai tujuan yang telah ditetapkan (Bourque & Fairley, 2014). Oleh karena itu, analisis kebutuhan menjadi tahapan awal yang sangat menentukan kualitas dan keberhasilan seluruh proses pengembangan sistem.

2.1.4 Ruang Lingkup Analisis Kebutuhan Sistem

Ruang lingkup analisis kebutuhan sistem mencakup seluruh kegiatan yang diperlukan untuk memahami kebutuhan pengguna dan menerjemahkannya menjadi spesifikasi yang jelas. Kegiatan tersebut tidak hanya berkaitan dengan aspek teknis, tetapi juga mencakup aspek organisasi, manusia, data, proses bisnis, dan lingkungan operasional.

Ruang lingkupnya seperti gambar dibawah ini:



Gambar 2. 1 Ruang Lingkup Analisis Kebutuhan Sistem

2.2 Pengumpulan Data dan Informasi

Pengumpulan data dan informasi merupakan salah satu tahapan penting dalam proses analisis kebutuhan sistem. Tahap ini bertujuan untuk memperoleh pemahaman yang mendalam mengenai kondisi sistem yang sedang berjalan, permasalahan yang dihadapi pengguna, proses bisnis organisasi, serta kebutuhan yang harus dipenuhi oleh sistem yang akan dikembangkan. Kualitas hasil analisis kebutuhan sangat dipengaruhi oleh ketepatan metode pengumpulan data yang digunakan, karena kebutuhan sistem tidak hanya

berasal dari perspektif pengembang, tetapi juga harus menggambarkan kebutuhan nyata dari pengguna dan pemangku kepentingan.

Dalam pengembangan sistem informasi, analis sistem perlu mengumpulkan informasi dari berbagai sumber agar memperoleh gambaran yang menyeluruh. Informasi tersebut dapat berupa data mengenai aktivitas pengguna, alur kerja organisasi, dokumen operasional, aturan bisnis, kendala sistem saat ini, serta harapan terhadap sistem baru. (Sugiyono, 2020), teknik pengumpulan data dapat dilakukan melalui observasi, wawancara, dokumentasi, maupun kombinasi berbagai teknik tersebut melalui proses triangulasi untuk memperoleh data yang lebih akurat. Dalam analisis kebutuhan sistem, terdapat beberapa teknik utama yang sering digunakan, yaitu wawancara, observasi, dan analisis dokumen. Ketiga teknik tersebut memiliki karakteristik yang berbeda, tetapi saling melengkapi. Wawancara digunakan untuk menggali informasi secara langsung dari pengguna, observasi digunakan untuk memahami aktivitas dan proses kerja secara nyata, sedangkan analisis dokumen digunakan untuk mempelajari informasi yang telah tersedia dalam bentuk dokumen organisasi.



Gambar 2. 2 Pengumpulan Data dan Informasi

2.2.1 Wawancara

Wawancara merupakan teknik pengumpulan data yang dilakukan melalui interaksi langsung antara analis sistem dengan pihak-pihak yang memiliki informasi terkait sistem. Melalui wawancara, analis dapat menggali kebutuhan, permasalahan, harapan, serta pengalaman pengguna dalam menjalankan proses bisnis tertentu. Teknik ini menjadi salah satu metode yang paling banyak digunakan karena mampu memperoleh informasi yang bersifat mendalam dan kontekstual.

Dalam analisis kebutuhan sistem, wawancara biasanya dilakukan kepada berbagai pemangku kepentingan, seperti pemilik sistem, manajemen, pengguna akhir, staf operasional, maupun pihak lain yang terdampak oleh keberadaan sistem. Setiap pihak dapat memberikan perspektif yang berbeda mengenai kebutuhan sistem. Sebagai contoh, manajemen mungkin lebih berfokus pada kebutuhan laporan dan pengambilan keputusan, sedangkan pengguna akhir

lebih memahami kebutuhan terkait kemudahan penggunaan dan fitur operasional.

Wawancara dapat dilakukan dalam beberapa bentuk, yaitu wawancara terstruktur, semi-terstruktur, dan tidak terstruktur. Wawancara terstruktur menggunakan daftar pertanyaan yang telah disiapkan sebelumnya sehingga proses pengumpulan informasi lebih terarah. Sementara itu, wawancara semi-terstruktur memberikan ruang bagi analis untuk mengembangkan pertanyaan berdasarkan jawaban narasumber. Bentuk ini sering digunakan dalam analisis kebutuhan karena memungkinkan analis memperoleh informasi baru yang sebelumnya belum diperkirakan.

Agar wawancara menghasilkan data yang berkualitas, analis sistem perlu mempersiapkan pertanyaan yang jelas, memahami konteks organisasi, menciptakan komunikasi yang baik dengan narasumber, serta melakukan pencatatan atau dokumentasi hasil wawancara. Hasil wawancara kemudian dianalisis untuk menemukan kebutuhan fungsional, kebutuhan nonfungsional, aturan bisnis, serta permasalahan yang perlu diselesaikan melalui sistem baru.

2.2.2 Observasi

Observasi merupakan teknik pengumpulan data dengan cara melakukan pengamatan langsung terhadap aktivitas, proses kerja, lingkungan operasional, serta interaksi pengguna dengan sistem yang sedang berjalan. Berbeda dengan wawancara yang bergantung pada penjelasan pengguna, observasi memungkinkan analis melihat kondisi sebenarnya sehingga dapat menemukan perbedaan antara prosedur formal dengan praktik yang terjadi di lapangan.

Dalam analisis kebutuhan sistem, observasi memiliki peran penting karena tidak semua kebutuhan dapat dijelaskan secara lengkap oleh pengguna. Terkadang pengguna telah terbiasa dengan suatu proses sehingga beberapa aktivitas dianggap tidak perlu dijelaskan, padahal aktivitas tersebut memiliki pengaruh terhadap rancangan sistem. Melalui observasi, analis dapat mengidentifikasi aliran pekerjaan, penggunaan dokumen, perpindahan informasi, hambatan operasional, serta aktivitas yang membutuhkan dukungan teknologi.

Observasi dapat dilakukan secara langsung maupun tidak langsung. Observasi langsung dilakukan dengan hadir di lokasi kerja dan mengamati proses yang berlangsung, sedangkan observasi tidak langsung dapat dilakukan melalui rekaman aktivitas, laporan penggunaan sistem, atau data historis. Dalam praktik pengembangan sistem, observasi sering dikombinasikan dengan wawancara untuk memastikan bahwa informasi yang diperoleh melalui pengamatan sesuai dengan penjelasan pengguna.

Keunggulan observasi adalah kemampuannya memberikan gambaran nyata mengenai sistem yang berjalan. Namun, analis juga perlu memperhatikan kemungkinan perubahan perilaku pengguna ketika mereka mengetahui sedang diamati. Oleh karena itu, observasi harus dilakukan secara sistematis, objektif, dan dalam kondisi yang menggambarkan aktivitas normal organisasi.

2.2.3 Analisis Dokumen

Analisis dokumen merupakan teknik pengumpulan data dengan mempelajari berbagai dokumen yang berkaitan dengan proses bisnis dan sistem yang sedang berjalan. Dokumen tersebut dapat berupa formulir,

laporan, prosedur operasional standar (SOP), kebijakan organisasi, arsip transaksi, struktur organisasi, maupun dokumen digital yang digunakan dalam aktivitas sehari-hari.

Dalam analisis kebutuhan sistem, dokumen berfungsi sebagai sumber informasi yang membantu analis memahami aturan, prosedur, data yang digunakan, serta hubungan antarproses dalam organisasi. Sebagai contoh, formulir transaksi dapat memberikan informasi mengenai data apa saja yang harus disimpan oleh sistem, sedangkan laporan operasional dapat menunjukkan informasi apa yang dibutuhkan oleh pengguna dan manajemen.

Analisis dokumen juga membantu memastikan bahwa kebutuhan sistem sesuai dengan kebijakan dan aturan yang berlaku dalam organisasi. Melalui dokumen, analis dapat menemukan informasi yang mungkin tidak muncul dalam wawancara maupun observasi. Oleh karena itu, dokumen menjadi sumber pendukung yang penting dalam proses validasi kebutuhan.

Meskipun demikian, analis perlu melakukan evaluasi terhadap dokumen yang diperoleh karena tidak semua dokumen selalu mencerminkan kondisi terbaru. Beberapa dokumen mungkin sudah tidak sesuai dengan proses aktual yang berjalan. Oleh sebab itu, analisis dokumen sebaiknya dikombinasikan dengan wawancara dan observasi agar menghasilkan pemahaman kebutuhan yang lebih lengkap.

2.2.4 Integrasi Teknik Pengumpulan Data

Wawancara, observasi, dan analisis dokumen merupakan teknik yang saling melengkapi dalam proses analisis kebutuhan sistem. Penggunaan satu metode saja sering kali belum mampu menggambarkan kebutuhan sistem secara menyeluruh. Oleh karena itu, analis sistem

perlu menggunakan pendekatan kombinasi atau triangulasi untuk membandingkan informasi dari berbagai sumber.

Sebagai contoh, hasil wawancara dengan pengguna dapat menunjukkan bahwa suatu proses telah berjalan dengan baik, tetapi hasil observasi mungkin menemukan adanya aktivitas tambahan yang belum terdokumentasi. Begitu pula dokumen organisasi dapat menunjukkan prosedur resmi, tetapi observasi dapat memperlihatkan bagaimana prosedur tersebut diterapkan dalam praktik. Dengan menggabungkan ketiga teknik tersebut, analis dapat menghasilkan kebutuhan sistem yang lebih akurat, realistis, dan sesuai dengan tujuan organisasi.

Dengan demikian, pengumpulan data dan informasi menjadi fondasi utama dalam analisis kebutuhan sistem. Proses ini tidak hanya bertujuan mengumpulkan fakta, tetapi juga membangun pemahaman bersama antara pengguna, organisasi, dan pengembang mengenai sistem yang akan dibangun. Kebutuhan yang diperoleh melalui proses pengumpulan data yang baik akan menjadi dasar dalam tahap perancangan, pembangunan, pengujian, hingga evaluasi sistem

2.3 Dokumentasi Spesifikasi Sistem

Dokumentasi spesifikasi sistem informasi merupakan dokumen yang berisi gambaran menyeluruh mengenai kebutuhan sistem yang telah melalui proses identifikasi, analisis, serta kesepakatan bersama antara pengembang dan para pemangku kepentingan pada tahap awal rekayasa kebutuhan (Franch, dkk. 2023). Dokumen ini menjadi pedoman utama dalam proses pengembangan karena menjelaskan tujuan sistem, fungsi yang harus disediakan, batasan operasional, serta karakteristik teknis yang perlu diperhatikan sebelum tahap implementasi dilakukan. Dengan

adanya dokumentasi spesifikasi yang jelas, seluruh pihak yang terlibat memiliki pemahaman yang sama mengenai sistem yang akan dibangun sehingga dapat mengurangi perbedaan persepsi dan meminimalkan risiko kesalahan selama proses pengembangan (Qamar, dkk. 2025).

Selain berperan sebagai panduan teknis, dokumentasi spesifikasi sistem juga berfungsi sebagai media komunikasi antara pengguna, analis sistem, perancang, dan pengembang perangkat lunak. Dokumen ini membantu memastikan bahwa kebutuhan pengguna telah diterjemahkan secara tepat ke dalam rancangan sistem yang dapat diimplementasikan. Kualitas dokumentasi sangat menentukan keberhasilan pengembangan sistem karena harus memenuhi beberapa karakteristik penting, seperti lengkap, akurat, konsisten, tidak ambigu, dan mudah dipahami oleh seluruh pihak yang berkepentingan (Qamar, dkk. 2023). Dokumentasi yang tersusun dengan baik dapat menjadi dasar dalam proses validasi kebutuhan, pengujian sistem, pemeliharaan, serta pengembangan sistem pada tahap berikutnya.

Dalam praktik pengembangan perangkat lunak, penyusunan dokumentasi spesifikasi sistem sering mengacu pada standar internasional seperti ISO/IEC/IEEE 29148 maupun IEEE 830-1998. Standar tersebut memberikan pedoman mengenai bagaimana kebutuhan sistem harus didokumentasikan secara sistematis, mulai dari deskripsi kebutuhan pengguna, kebutuhan fungsional dan nonfungsional, rancangan antarmuka, hingga pemodelan sistem menggunakan diagram seperti use case diagram dan model proses lainnya (Allan, 2021; Farid & Nuryasin, 2024; Tanatti & Wicaksono, 2024). Penerapan standar ini membantu memastikan bahwa setiap kebutuhan dapat ditelusuri, diverifikasi, dan divalidasi sebelum sistem dikembangkan lebih lanjut.

Dengan demikian, dokumentasi spesifikasi sistem informasi tidak hanya dipandang sebagai dokumen administratif, tetapi merupakan instrumen strategis dalam menjamin keberhasilan proyek pengembangan sistem. Dokumentasi yang baik mampu menjembatani kebutuhan organisasi dengan solusi teknologi yang dibangun, meningkatkan efektivitas komunikasi antaraktor, serta mengurangi kemungkinan terjadinya kegagalan proyek akibat perubahan kebutuhan, pembengkakan biaya, maupun keterlambatan implementasi (Qamar, dkk. 2025).

2.3.1 *Software Requirements Specification* (SRS)

Penyusunan *Software Requirements Specification* (SRS) merupakan proses menerjemahkan kebutuhan sistem informasi yang diperoleh dari para pemangku kepentingan menjadi spesifikasi teknis yang lebih rinci dan terstruktur sebagai bagian dari *requirement baseline* sistem (Montalvo, dkk. 2021). Tahapan ini bertujuan untuk memastikan bahwa setiap kebutuhan pengguna dapat dipahami, didokumentasikan, dan dikembangkan menjadi acuan yang jelas dalam proses pembangunan perangkat lunak. Melalui penyusunan SRS, kebutuhan yang sebelumnya masih bersifat konseptual diubah menjadi deskripsi sistem yang lebih spesifik, sehingga dapat menjadi dasar bagi pengembang dalam merancang, membangun, dan menguji sistem sesuai dengan tujuan yang telah ditetapkan.

Proses pengembangan SRS melibatkan beberapa aktivitas penting, seperti identifikasi, pengelompokan, analisis, serta evaluasi kebutuhan fungsional dan nonfungsional. Setiap kebutuhan harus dikaji secara menyeluruh agar hubungan antara kebutuhan pengguna, fungsi sistem, serta batasan teknis dapat ditelusuri dengan baik. Pendekatan ini memungkinkan organisasi untuk menentukan prioritas kebutuhan secara konsisten

dan memastikan bahwa setiap komponen sistem memiliki keterkaitan yang jelas dengan tujuan bisnis maupun kebutuhan pengguna (Kaur, dkk. 2024).

Kebutuhan yang telah dikumpulkan melalui proses elisitasi perlu divalidasi dan dipetakan ke dalam spesifikasi sistem yang lebih terukur. Tahap validasi berperan penting untuk memastikan bahwa rancangan teknis yang dihasilkan benar-benar sesuai dengan harapan pengguna serta memiliki kelayakan untuk diterapkan secara operasional. Dengan demikian, SRS tidak hanya menjadi dokumen teknis, tetapi juga menjadi media komunikasi yang menyatukan pemahaman antara pengguna, analis sistem, dan tim pengembang mengenai ruang lingkup serta karakteristik sistem yang akan dibangun.

Dalam penyusunan SRS, penggunaan teknik elisitasi seperti *user persona*, pemodelan proses bisnis, dan berbagai pendekatan analisis kebutuhan menjadi penting untuk mengurangi kesenjangan informasi antara kebutuhan pengguna dan rancangan sistem. Teknik tersebut membantu menggambarkan karakteristik pengguna, pola interaksi, serta permasalahan yang harus diselesaikan oleh sistem secara lebih mendalam (Andhika, dkk. 2021). Oleh karena itu, dokumentasi SRS yang disusun secara sistematis dapat menjadi dasar yang kuat dalam memperkirakan kebutuhan sumber daya, biaya, waktu, serta potensi risiko selama siklus pengembangan perangkat lunak (Rączkowska-Gzowska & Walkowiak-Gall, 2023).

Struktur standar dokumen *Software Requirements Specification* (SRS) disusun untuk menggambarkan kebutuhan sistem secara terarah dan sistematis. Secara umum, SRS terdiri atas bagian pendahuluan, deskripsi umum sistem, dan kebutuhan spesifik yang menjelaskan

fungsi sistem, kinerja, serta batasan operasional (Almeida & Dávila, 2023).

Bagian kebutuhan spesifik memuat kebutuhan fungsional yang menjelaskan layanan atau aktivitas yang harus dilakukan sistem, serta kebutuhan nonfungsional yang menjelaskan kualitas sistem seperti keamanan, keandalan, dan kemudahan penggunaan. Selain itu, SRS juga mencakup dokumentasi antarmuka eksternal dan batasan sistem untuk memastikan sistem dapat berjalan sesuai lingkungan operasional dan standar yang telah ditetapkan (Siahaan, dkk.2024).

Dengan struktur yang jelas, dokumen SRS menjadi pedoman bersama antara pengguna, analis, dan pengembang dalam memahami sistem yang akan dibangun. Dokumentasi ini membantu mengurangi kesalahan pemahaman, memastikan kebutuhan dapat diterapkan dengan tepat, serta menjadi dasar dalam proses pengembangan dan pengujian perangkat lunak.

2.3.2 Validasi Spesifikasi Sistem

Validasi spesifikasi sistem dilakukan untuk memastikan bahwa seluruh kebutuhan yang terdokumentasi dalam SRS telah sesuai dengan harapan pemangku kepentingan dan dapat diterapkan secara jelas tanpa menimbulkan perbedaan pemahaman. Proses ini bertujuan untuk menilai kualitas dokumen spesifikasi berdasarkan beberapa aspek penting, seperti kelengkapan, ketepatan, konsistensi, keterujian, dan kemudahan pemahaman (Krishna, dkk. 2024).

Dalam proses validasi, setiap kebutuhan yang tercantum dalam dokumen dibandingkan dengan hasil pengumpulan kebutuhan dari pengguna untuk menemukan kemungkinan perbedaan antara kebutuhan sebenarnya dan interpretasi pengembang (Rizkifar, dkk. 2023). Keterlibatan aktif pemangku kepentingan sangat

diperlukan agar spesifikasi sistem yang dihasilkan benar-benar mencerminkan tujuan organisasi serta mampu mengurangi risiko kesalahan pengembangan, cacat produk, dan kegagalan proyek (Rizqi & Priyadi, 2023).

Dengan mengintegrasikan proses validasi ke dalam tahapan pengembangan sistem, pengujian dapat dilakukan secara lebih terarah dan berkelanjutan pada setiap tingkat sistem. Hal ini membantu memastikan bahwa sistem yang dibangun tidak hanya memenuhi kebutuhan teknis, tetapi juga memberikan solusi yang sesuai dengan kebutuhan pengguna (Siakas, dkk. 2024; Fahl *et al*., 2021).

2.4 Contoh Kasus Pengaplikasian Analisis Kebutuhan Sistem

***Software Requirements Specification* (SRS)**

Tabel 2. 1 Informasi Dokumen

Elemen	Keterangan
Judul	Software Requirements Specification - Sistem Informasi Perpustakaan
Status	Dokumen awal / baseline kebutuhan
Tujuan	Menjadi acuan pengembangan, pengujian, dan penerimaan sistem

1. Tujuan Dokumen

Dokumen Software Requirements Specification (SRS) ini menjelaskan kebutuhan perangkat lunak untuk Sistem Informasi Perpustakaan Sederhana. Sistem dirancang untuk membantu petugas mengelola data buku, data anggota, transaksi peminjaman,

pengembalian, denda, dan laporan secara terkomputerisasi. Dokumen ini menjadi acuan bersama bagi pengembang, pengguna, penguji, serta pengelola perpustakaan selama proses perancangan, pembangunan, pengujian, penerapan, dan pemeliharaan sistem.

2. Ruang Lingkup Sistem

Sistem berbasis web dan digunakan dalam lingkungan perpustakaan sekolah atau organisasi berskala kecil. Fungsi utama sistem meliputi:

- Mengelola akun pengguna dan hak akses.
- Mengelola data anggota perpustakaan.
- Mengelola data buku dan ketersediaan stok.
- Mencari informasi buku.
- Mencatat transaksi peminjaman dan pengembalian.
- Menghitung keterlambatan serta denda secara otomatis.
- Menampilkan dan mencetak laporan perpustakaan.

3. Definisi dan Istilah

Tabel 2. 2 Definis dan Istilah

Istilah	Definisi
SRS	Dokumen yang menjelaskan kebutuhan perangkat lunak secara terstruktur.
Admin	Pengguna dengan hak akses penuh terhadap sistem.
Petugas	Pengguna yang mengelola data dan transaksi perpustakaan.
Anggota	Siswa atau pengguna yang terdaftar sebagai anggota perpustakaan.
Peminjaman	Proses penyerahan buku kepada anggota untuk jangka waktu tertentu.

Istilah	Definisi
Pengembalian	Proses pencatatan buku yang dikembalikan oleh anggota.
Denda	Biaya akibat keterlambatan pengembalian buku.
Stok tersedia	Jumlah eksemplar buku yang dapat dipinjam.

4. Deskripsi Umum Sistem

Perspektif Produk

Sistem Informasi Perpustakaan Sederhana merupakan aplikasi mandiri yang menggantikan pencatatan manual menggunakan buku besar atau lembar kerja. Seluruh data disimpan pada basis data terpusat sehingga pencarian, pemutakhiran stok, pelacakan transaksi, dan penyusunan laporan dapat dilakukan lebih cepat serta konsisten.

Permasalahan Sistem Berjalan

- Pencarian data buku dan anggota membutuhkan waktu relatif lama.
- Data peminjaman rentan hilang, rusak, atau tercatat ganda.
- Petugas kesulitan mengetahui ketersediaan buku secara langsung.
- Kesalahan pencatatan batas pengembalian dan perhitungan denda masih sering terjadi.
- Penyusunan laporan membutuhkan pekerjaan manual yang berulang.

Tujuan Pengembangan

- Memudahkan pengelolaan data buku dan anggota.

- Mempercepat transaksi peminjaman dan pengembalian.
- Menyediakan informasi ketersediaan buku secara aktual.
- Mengurangi kesalahan pencatatan dan perhitungan.
- Menghasilkan laporan yang cepat, akurat, dan mudah ditelusuri.

Tabel 2. 3 Karakteristik Pengguna

Pengguna	Karakteristik	Hak Akses
Admin	Memahami pengoperasian dasar komputer.	Seluruh data, akun, transaksi, laporan, dan pengaturan.
Petugas	Memahami pelayanan perpustakaan.	Buku, anggota, peminjaman, pengembalian, dan laporan.
Anggota	Mampu menggunakan aplikasi sederhana.	Mencari dan melihat informasi buku.

Inafrasturuktur Operasional Sistem

- Sistem operasi Windows, Linux, atau macOS.
- Peramban Google Chrome, Microsoft Edge, atau Mozilla Firefox.
- Basis data MySQL atau PostgreSQL.
- Server lokal atau layanan web hosting.
- Jaringan lokal atau internet.

Batasan Sistem

- Setiap admin dan petugas harus mempunyai akun terdaftar.
- Transaksi hanya dapat diproses oleh admin atau petugas.

- Satu anggota maksimal meminjam tiga buku selama tujuh hari.
- Denda keterlambatan ditetapkan Rp1.000 per buku per hari.
- Buku hanya dapat dipinjam apabila stok tersedia.
- Pembayaran denda daring belum termasuk dalam ruang lingkup versi ini.

5. Kebutuhan Fungsional

Tabel 2. 4 Kebutuhan Fungsional

Kode	Fungsi	Spesifikasi Kebutuhan
FR-01	Login	Sistem memvalidasi nama pengguna dan kata sandi, lalu menampilkan halaman sesuai hak akses. Data yang tidak sesuai menghasilkan pesan kesalahan.
FR-02	Pengelolaan Pengguna	Admin dapat menambah, melihat, mengubah, mengaktifkan, menonaktifkan, menghapus akun, dan mengubah kata sandi.
FR-03	Pengelolaan Anggota	Admin dan petugas dapat menambah, melihat, mencari, mengubah, menonaktifkan, dan menghapus data anggota.
FR-04	Pengelolaan Buku	Admin dan petugas dapat menambah, melihat, mengubah, menghapus, mengelompokkan, dan memperbarui stok buku.
FR-05	Pencarian Buku	Sistem menyediakan pencarian berdasarkan kode, judul, penulis, penerbit, dan kategori

Kode	Fungsi	Spesifikasi Kebutuhan
		serta menampilkan status ketersediaan.
FR-06	Peminjaman	Sistem memeriksa keaktifan anggota, stok, jumlah pinjaman, dan keterlambatan sebelum menyimpan transaksi serta mengurangi stok.
FR-07	Pengembalian	Sistem mencatat tanggal dan kondisi buku, menghitung keterlambatan dan denda, menyelesaikan transaksi, serta menambah stok.
FR-08	Perhitungan Denda	Sistem menghitung denda berdasarkan jumlah hari keterlambatan dikalikan jumlah buku dan tarif Rp1.000.
FR-09	Dashboard	Sistem menampilkan jumlah buku, anggota aktif, pinjaman aktif, buku terlambat, stok tersedia, dan transaksi terbaru.
FR-10	Laporan	Sistem menghasilkan laporan buku, anggota, peminjaman, pengembalian, keterlambatan, denda, dan buku populer berdasarkan periode.
FR-11	Logout	Sistem mengakhiri sesi pengguna dan mengarahkan kembali ke halaman login.

Tabel 2. 5 Spesifikasi Atribut dan Luaran Sistem

Fungsi	Data Masukan Utama	Keluaran Utama
Login	Nama pengguna, kata sandi	Status autentikasi dan halaman utama
Anggota	Nomor, nama, kelas/unit, alamat, telepon, status	Data anggota tersimpan
Buku	Kode, ISBN, judul, penulis, penerbit, kategori, rak, stok	Data dan ketersediaan buku
Peminjaman	Anggota, buku, tanggal pinjam	Nomor transaksi dan batas pengembalian
Pengembalian	Nomor transaksi, tanggal kembali, kondisi	Status selesai, keterlambatan, dan denda

6. Kebutuhan Nonfungsional

Tabel 2. 6 Kebutuhan Nonfungsional

Kode	Aspek	Spesifikasi
NFR-01	Kinerja	Halaman tampil maksimal tiga detik; pencarian maksimal dua detik; minimal 20 pengguna bersamaan.
NFR-02	Keamanan	Autentikasi wajib, kata sandi di-hash, akses berbasis peran, sesi

Kode	Aspek	Spesifikasi
		berakhir setelah 30 menit tidak aktif.
NFR-03	Keandalan	Transaksi disimpan konsisten; transaksi gagal tidak mengubah stok; pencadangan dilakukan berkala.
NFR-04	Kemudahan penggunaan	Antarmuka berbahasa Indonesia, menu konsisten, validasi jelas, dan konfirmasi sebelum penghapusan.
NFR-05	Kompatibilitas	Berfungsi pada Chrome, Edge, dan Firefox serta menyesuaikan layar komputer, laptop, dan tablet.
NFR-06	Pemeliharaan	Kode modular, struktur basis data terdokumentasi, dan tarif denda dapat diubah melalui pengaturan.
NFR-07	Ketersediaan	Sistem tersedia selama jam operasional dengan target minimal 98 persen.

7. Ketentuan Akses Sistem

Tabel 2. 7 Ketentuan Akses Sistem

No	Akses Sistem
1	Hanya anggota aktif yang dapat meminjam buku.
2	Setiap anggota maksimal meminjam tiga buku.
3	Masa peminjaman maksimal tujuh hari.
4	Buku hanya dapat dipinjam apabila stok tersedia.
5	Anggota yang memiliki keterlambatan tidak dapat melakukan peminjaman baru.
6	Denda sebesar Rp1.000 per buku per hari keterlambatan.
7	Pengembalian harus merujuk pada transaksi peminjaman yang sah.
8	Penghapusan transaksi hanya dapat dilakukan oleh admin.
9	Kode buku, nomor anggota, dan nomor transaksi harus unik.

8. Kebutuhan Antarmuka

Tabel 2. 8 Komponen Antarmuka Sistem

Antarmuka	Komponen Minimal
Halaman Login	Logo/nama perpustakaan, nama pengguna, kata sandi, tombol masuk, dan pesan kesalahan.
Dashboard	Menu navigasi, statistik utama, transaksi terbaru, dan informasi keterlambatan.

Antarmuka	Komponen Minimal
Form Buku	Kode, ISBN, judul, penulis, penerbit, tahun, kategori, rak, dan stok.
Form Anggota	Nomor, nama, jenis kelamin, kelas/unit, alamat, telepon, dan status.
Form Peminjaman	Nomor anggota, identitas anggota, kode/judul buku, tanggal pinjam, batas kembali, simpan, dan batal.
Form Pengembalian	Nomor transaksi, identitas, buku, tanggal pinjam/kembali, keterlambatan, kondisi, dan denda.
Halaman Laporan	Filter periode, jenis laporan, pratinjau, cetak, dan ekspor PDF.

9. Atribut Data Utama

Tabel 2. 9 Atribut Data

Entitas	Atribut Utama
Pengguna	ID pengguna; nama; nama pengguna; kata sandi; peran; status akun.
Anggota	ID; nomor anggota; nama; jenis kelamin; kelas/unit; alamat; telepon; status.
Buku	ID; kode; ISBN; judul; penulis; penerbit; tahun; kategori; rak; total stok; stok tersedia.

Entitas	Atribut Utama
Peminjaman	ID; nomor transaksi; ID anggota; ID petugas; tanggal pinjam; batas kembali; status.
Detail Peminjaman	ID detail; ID peminjaman; ID buku; tanggal kembali; keterlambatan; denda; kondisi.

Dokumen SRS ini menjadi baseline awal pengembangan Sistem Informasi Perpustakaan Sederhana. Seluruh kebutuhan perlu diverifikasi bersama calon pengguna sebelum memasuki tahap perancangan dan implementasi. Setiap perubahan kebutuhan selama pengembangan harus dicatat melalui mekanisme pengendalian perubahan agar kesesuaian antara kebutuhan pengguna, rancangan, implementasi, dan pengujian tetap terjaga.

Analisis kebutuhan sistem merupakan landasan utama dalam pengembangan sistem informasi karena menentukan arah perancangan, pembangunan, pengujian, hingga penerapan sistem. Melalui proses ini, permasalahan pada sistem yang sedang berjalan, kebutuhan pengguna, tujuan organisasi, ruang lingkup, dan batasan pengembangan dapat dipahami secara menyeluruh. Kebutuhan yang telah diidentifikasi kemudian dikelompokkan menjadi kebutuhan fungsional yang menjelaskan layanan dan proses yang harus dijalankan sistem serta kebutuhan nonfungsional yang menentukan kualitas operasionalnya, seperti kinerja, keamanan, keandalan, kemudahan penggunaan, dan kompatibilitas. Perumusan kedua jenis kebutuhan tersebut secara jelas dan terukur dapat mengurangi kesalahan penafsiran, perubahan berulang, pembengkakan biaya, dan risiko kegagalan pengembangan.

Kualitas hasil analisis sangat dipengaruhi oleh ketepatan proses pengumpulan data dan keterlibatan aktif pemangku kepentingan. Wawancara membantu menggali pengalaman, harapan, dan kendala pengguna; observasi memberikan gambaran mengenai praktik kerja yang berlangsung; sedangkan analisis dokumen memperkuat pemahaman terhadap aturan, prosedur, serta data organisasi. Integrasi ketiga teknik tersebut menghasilkan informasi yang lebih lengkap dan dapat dipercaya. Selanjutnya, kebutuhan yang diperoleh perlu didokumentasikan dalam *Software Requirements Specification* (SRS) dan divalidasi berdasarkan aspek kelengkapan, ketepatan, konsistensi, keterujian, serta kemudahan pemahaman. Dengan demikian, SRS tidak hanya berfungsi sebagai dokumen teknis, tetapi juga menjadi media komunikasi dan kesepakatan bersama antara pengguna, analis, pengembang, dan penguji.

Penerapan analisis kebutuhan pada contoh Sistem Informasi Perpustakaan Sederhana menunjukkan bahwa kebutuhan pengguna dapat diterjemahkan secara sistematis ke dalam fungsi pengelolaan akun, anggota, buku, peminjaman, pengembalian, denda, dan laporan. Kebutuhan tersebut dilengkapi dengan ketentuan akses, atribut data, rancangan antarmuka, serta standar kualitas sistem agar implementasinya sesuai dengan kondisi operasional perpustakaan. Oleh karena itu, keberhasilan pengembangan sistem pada akhirnya tidak hanya ditentukan oleh kecanggihan teknologi, tetapi juga oleh kemampuan pengembang memahami manusia, proses kerja, dan tujuan organisasi yang hendak didukung. Analisis kebutuhan yang dilakukan secara sistematis, kolaboratif, dan berorientasi pada pengguna akan menghasilkan sistem yang lebih tepat guna, mudah diterima, andal, serta memberikan manfaat nyata bagi organisasi.

DAFTAR PUSTAKA

Akram, F., Ahmad, T., & Sadiq, M. (2024). Recommendation systems-based software requirements elicitation process—A systematic literature review. Journal of Engineering and Applied Science, 71, 29. https://doi.org/10.1186/s44147-024-00363-4

Allan, G. (2021). ANALISA KEBUTUHAN KEBUTUHAN SISTEM INFORMASI MANAJEMEN PERUSAHAAN DAGANG. Kurawal - Jurnal Teknologi Informasi dan Industri, 4(1), 17–30. https://doi.org/10.33479/kurawal.v4i1.400

Almeida, S., & Dávila, A. (2023). A Systematic Mapping Study on Process Improvement in Software Requirements Engineering. Proceedings of the Institute for System Programming of RAS, 35(1), 141–162. https://doi.org/10.15514/ispras-2023-35(1)-10

Almufareh, M. F., & Humayun, M. (2023). Improving the Safety and Security of Software Systems by Mediating SAP Verification. Applied Sciences, 13(1), 647–647. https://doi.org/10.3390/app13010647

Amalia, S., Yasnita, Z., & Rahmayani, M. T. I. (2023). Perancangan Sistem Informasi Pencatatan dan Pemberitahuan Pembayaran Sumbangan Pembinaan Pendidikan (SPP) pada SMP Swasta Muhammadiyah Plus Bengkalis. Jurnal Teknik Industri Terintegrasi, 6(3), 847–861. https://doi.org/10.31004/jutin.v6i3.16948

Aman, M., & Suroso, S. (2021). Pengembangan Sistem Informasi Wedding Organizer Menggunakan Pendekatan Sistem Berorientasi Objek Pada CV Pesta. Jurnal Janitra Informatika dan Sistem Informasi, 1(1), 47–60. https://doi.org/10.25008/janitra.v1i1.119

Andhika, W., Iqbal, M., Nur, M. R., & Nugroho, R. D. (2021). Teknik Representasi Kebutuhan Pengguna Menggunakan User Persona (Studi Kasus: Relasi antara Presensi dengan Kemampuan Kompetensi). Jurnal Teknologi Informasi dan Ilmu Komputer, 8(2), 245–245. https://doi.org/10.25126/jtiik.0813444

Ariati, N., Kamandanu, A., Maharani, P. A., Pratama, A. W., & Febriansyah, D. P. (2026). Analisis Perencanaan Strategis Sistem Informasi Menggunakan Metode Ward and Peppard. RIGGS Journal of Artificial Intelligence and Digital Business, 4(4), 10433–10440. https://doi.org/10.31004/riggs.v4i4.5141

Bangsawan, I. (2021). ANALISIS SITEM_INDRA BANGSAWAN. https://doi.org/10.31219/osf.io/tjxpc

Baso, F., S., A. N., Alkhalil, I., & Imansyah, D. A. (2024). SRS and Design Analysis: Kunci Kesuksesan Proyek Sistem Informasi yang Efektif. Indonesian Technology and Education Journal, 92–101. https://doi.org/10.61255/itej.v2i1.449

Bourque, P., & Fairley, R. E. (Eds.). (2014). Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 3.0. IEEE Computer Society.

Fahl, J., Hirschter, T., Wöhrle, G., & Albers, A. (2021). PROPOSING A SPECIFICATION STRUCTURE FOR COMPLEX PRODUCTS IN MODEL-BASED SYSTEMS ENGINEERING (MBSE). Proceedings of the Design Society, 1, 2481–2490. https://doi.org/10.1017/pds.2021.509

Fani, R., Simanungkalit, C. J., Shandy, M., Naufal, N., Alfryan, M., Ikhsan, I., Arimbi, A., Carlos, C., Bangun, O., & Larosa, T. (2026). Pengaruh TO-BE pada Perancangan Sistem Informasi. Neptunus, 4(1), 106–114. https://doi.org/10.61132/neptunus.v4i1.1478

Farid, A. M. F., & Nuryasin, I. (2024). Spesifikasi Kebutuhan Perangkat Lunak Sistem Informasi E-Survey Badan Litbang dan Diklat Kementerian Agama dengan standar ISO/IEC/IEEE 29148:2018. Infotek Jurnal Informatika Dan Teknologi, 7(2), 318–328. https://doi.org/10.29408/jit.v7i2.26082

Faridiana, P. D. N. R., Salsabilla, K. A. Z., Syahputra, R. P., Hardiartama, R., Kristana, B. P., & Fitri, A. S. (2024). Perancangan Sistem Informasi Pendataan Tahanan dan Narapidana Terintegrasi Antar Kepolisian, Kejaksaan, Lapas dan Bapas Kota Malang Menggunakan Iconix Proses. Jurnal Ilmiah Edutic Pendidikan Dan Informatika, 11(1), 1–13. https://doi.org/10.21107/edutic.v11i1.18009

Franch, X., Palomares, C., Quer, C., Chatzipetrou, P., & Gorschek, T. (2023). The state-of-practice in requirements specification: an extended interview study at 12 companies. Requirements Engineering, 28(3), 377–409. https://doi.org/10.1007/s00766-023-00399-7

Kaur, K., & Kaur, P. (2024). The application of AI techniques in requirements classification: a systematic mapping. Artificial Intelligence Review, 57(3). https://doi.org/10.1007/s10462-023-10667-1

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024a). Using LLMs in Software Requirements Specifications: An Empirical Evaluation. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2404.17842

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024b). Using LLMs in Software Requirements Specifications: An Empirical Evaluation. 475–483. https://doi.org/10.1109/re59067.2024.00056

Kusuma, W. A., Amir, A., & Dirgantara, A. (2021). ANALISIS DAN EVALUASI KEBUTUHAN PERANGKAT LUNAK DENGAN MENGGUNAKAN METODE USER PERSONA. Prosiding SENTRA (Seminar Teknologi dan Rekayasa), 6, 74–84. https://doi.org/10.22219/sentra.v0i6.3768

Kusuma, W. A., Nurhuda, M. A., & Purwanto, W. (2021). Peningkatan Pemahaman Pengguna Pada Proses Penggalian Kebutuhan Menggunakan Metode Persona. Prosiding SENTRA (Seminar Teknologi dan Rekayasa), 6, 184–193. https://doi.org/10.22219/sentra.v0i6.3819

Kusuma, W. A., Wulansari, N. D., & Fakhriansyah, M. I. (2021). ELISITASI KEBUTUHAN PERANGKAT LUNAK DENGAN METODE USER PERSONA DALAM UPAYA MENINGKATKAN PENGALAMAN PENGGUNA. Prosiding SENTRA (Seminar Teknologi dan Rekayasa), 6, 194–205. https://doi.org/10.22219/sentra.v0i6.3823

Liliana, D. Y., Hikmah, N. N., & Harjono, M. (2021). Pengembangan Sistem Pemantauan Sentimen Berita Berbahasa Indonesia Berdasarkan Konten dengan Long-Short-Term Memory. Jurnal Teknologi Informasi dan Ilmu Komputer, 8(5), 995–1002. https://doi.org/10.25126/jtiik.2021854624

Manis, R., Setiyaningsih, W., & Kuswinardi, W. (2021). RANCANG BANGUN SISTEM INFORMASI PENJUALAN LAPTOP BERBASIS WEB DENGAN METODE WATERFALL. Rainstek Jurnal Terapan Sains dan Teknologi, 3(3), 197–207. https://doi.org/10.21067/jtst.v3i3.6065

Melén, C. N. K. T., Johansson, C., & Bertoni, M. (2022). Issues in Capturing and Utilizing Non-Functional Requirements along the Road to Servitization. Proceedings of the Design Society, 2, 623–632. https://doi.org/10.1017/pds.2022.64

Montalvo, A., Parra, P., Polo, Ó. R., Carrasco, A., Silva, A. da, Martínez, A., & Sánchez, S. (2021). Model-driven system-level validation and verification on the space software domain. Software & Systems Modeling, 21(6), 2367–2394. https://doi.org/10.1007/s10270-021-00940-8

Mutiari, L., & Wahyuningtyas, E. (2021). RANCANG BANGUN SISTEM INFORMASI POINT OF SALE BERBASIS WEBSITE PADA CV.LULUS. Melek IT Information Technology Journal, 7(2), 91–100. https://doi.org/10.30742/melekitjournal.v7i2.179

Noftianto, E. F. N., & Kusuma, W. A. (2021a). Teknik Iterasi User Persona dalam Penggalian Kebutuhan Perangkat Lunak Sistem Pembelajaran Jarak Jauh. Jurnal Sosial Teknologi, 1(7), 633–643. https://doi.org/10.59188/jurnalsostech.v1i7.134

Pamungkas, R. D., & Anggoro, D. A. (2023). Sistem Informasi Pelayanan TOEP di LBIPU UMS Berbasis Web. Emitor Jurnal Teknik Elektro, 22(2), 93–99. https://doi.org/10.23917/emitor.v22i2.22314

Permadi, T. T. A., & Saurina, N. (2022). SISTEM INFORMASI PENGOLAHAN DATA DISTRIBUTOR BERBASIS WEB PADA CV. AYAM ADI BROILER. Melek IT Information Technology Journal, 8(1), 27–36. https://doi.org/10.30742/melekitjournal.v8i1.195

Qamar, N., Sabahat, N., Mashmool, A., & Mosavi, A. (2023). Evaluating the Impact of Pair Documentation on Requirements Quality and Team Productivity. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2304.14255

Qamar, N., Sabahat, N., & Mosavi, A. (2025). Evaluating the Impact of Pair Documentation on Requirements Quality in Agile Software Development. Közszolgálati Tudásportál, 13, 45784–45794. https://doi.org/10.1109/access.2025.3550123

Rączkowska-Gzowska, K., & Walkowiak-Gall, A. (2023). What Should a Good Software Requirements Specification Include? Results of a Survey. Foundations of Computing and Decision Sciences, 48(1), 57–81. https://doi.org/10.2478/fcds-2023-0004

Rafildo, R., Munti, N. Y. S., & Azriadi, E. (2022). Rancang Bangun Sistem Informasi E-Laundry dengan Implementasi Berbasis Web (Analysis). INNOVATIVE Journal Of Social Science Research, 2(1), 524–532. https://doi.org/10.31004/innovative.v2i1.3806

Rizkifar, M. A., Priyadi, Y., & Alibasa, M. J. (2023). Validating Requirement Specification Using Text Mining Analysis. 252–257. https://doi.org/10.1109/ice3is59323.2023.10335293

Rizqi, A. M., & Priyadi, Y. (2023). Text Validity Application Forming Functional and Non-Functional Requirements Based on Documentation Analysis on TESA Applications. 342–347. https://doi.org/10.1109/ieit59852.2023.10335493

Sandfreni, S., Ulum, M. B., & Azizah, A. H. (2021). ANALISIS PERANCANGAN SISTEM INFORMASI PUSAT STUDI PADA FAKULTAS ILMU KOMPUTER UNIVERSITAS ESA UNGGUL. Sebatik, 25(2), 345–356. https://doi.org/10.46984/sebatik.v25i2.1587

Siahaan, D., Fauzan, R., Widyadhana, A., Firmawan, D. B., Putri, R. R., Desnelita, Y., Gustientiedina, G., & Putrian, R. N. (2024). A scoping review of auto-generating transformation between software development artifacts. Frontiers in Computer Science, 5. https://doi.org/10.3389/fcomp.2023.1306064

Siakas, K., Georgiadou, E., Rahanu, H., Siakas, E., Meggoudis, N., & Siakas, D. (2024). Overcoming Obstacles in Global Requirements Elicitation: A Multicultural Perspective. Journal of Software Engineering Research and Development, 12(1). https://doi.org/10.5753/jserd.2024.2552

Sugiyono. (2020). Metode penelitian kuantitatif, kualitatif, dan R&D (Cetakan ke-2). Alfabeta.

Sukmawati, Y., Panduardi, F., Febrita, R. E., Umam, K., & Rini, E. M. (2024). Pengembangan Sistem Informasi Manajemen Jasa Layanan Internet di PT Semesta Multitekno Indonesia. INFOMATEK, 26(1), 75–90. https://doi.org/10.23969/infomatek.v26i1.13705

Syukron, A., Priyono, J., & Sardiarinto, S. (2022). Perancangan Sistem Informasi Permintaan Pembelian Perlengkapan Kantor (SIP3K) Berbasis Web. Indonesian Journal Computer Science, 1(1), 40–45. https://doi.org/10.31294/ijcs.v1i1.1144

Tanatti, A. S. K., & Wicaksono, S. R. (2024). PEMBUATAN DOKUMEN SRS (SOFTWARE REQUIREMENT SYSTEM) SISTEM INFORMASI MANUFAKTUR PERUSAHAAN MESIN KARET (STUDI KASUS PT. ABC). Sainsbertek Jurnal Ilmiah Sains & Teknologi, 4(2), 1–15. https://doi.org/10.33479/sb.v4i2.307

Tanjung, F. R., & Widodo, S. (2024). Metricfy: Aplikasi Pembelajaran Numerasi berbasis Web. EDUMATIC Jurnal Pendidikan Informatika, 8(1), 242–251. https://doi.org/10.29408/edumatic.v8i1.25757

Umar, M. A., & Lano, K. (2024). Advances in automated support for requirements engineering: A systematic literature review. Requirements Engineering, 29(2), 177–207. https://doi.org/10.1007/s00766-023-00411-0

Wijaya, W. W. W., & Susanto, E. (2021). New Normal: Pengembangan Sistem Informasi Penjualan Menggunakan Metode SDLC (System Development Life Cycle). Jurnal Sustainable Jurnal Hasil Penelitian Dan Industri Terapan, 10(1), 1–9. https://doi.org/10.31629/sustainable.v10i1.3190

Wouters, J., Menkveld, A., Brinkkemper, S., & Dalpiaz, F. (2022). Crowd-based requirements elicitation via pull feedback: method and case studies. Requirements Engineering, 27(4), 429–455. https://doi.org/10.1007/s00766-022-00384-6

BAB 3

PERANCANGAN ANTAR MUKA DAN BASIS DATA

Oleh Taufik Rachman

3.1 Pentingnya Desain Antarmuka dalam Keberhasilan Sistem Informasi

Perkembangan teknologi informasi menjadikan sistem informasi sebagai sarana penting dalam mendukung proses bisnis, pengambilan keputusan, dan peningkatan layanan. Keberhasilannya tidak hanya ditentukan oleh teknologi dan fitur, tetapi juga oleh kemampuan sistem memberikan pengalaman penggunaan yang efektif, efisien, dan memuaskan. Karena itu, desain antarmuka (*User Interface/UI*) menjadi faktor penting dalam keberhasilan sistem informasi.

Antarmuka merupakan media komunikasi antara pengguna dan sistem untuk memasukkan data, mengakses informasi, menjalankan fungsi, dan menghasilkan laporan. Antarmuka yang baik meningkatkan efektivitas interaksi dan membantu pengguna mencapai tujuan sistem secara optimal (Pressman & Maxim, 2020). (McGraw Hill Education)

Pengembangan sistem modern semakin menerapkan pendekatan *user-centered design*, yaitu menempatkan kebutuhan dan karakteristik pengguna sebagai dasar perancangan sejak analisis hingga evaluasi. Sistem dengan teknologi canggih tetap berpotensi gagal apabila sulit

digunakan, sedangkan antarmuka yang intuitif dapat meningkatkan penerimaan pengguna (ISO 9241-210:2019). (ISO)

Desain antarmuka yang baik harus memiliki *learnability*, *usability*, konsistensi, efisiensi, dan kemampuan meminimalkan kesalahan. Antarmuka yang efektif juga meningkatkan kepuasan, produktivitas, dan kualitas layanan serta mengurangi waktu pelatihan dan kesalahan pengguna.

Perkembangan aplikasi web, mobile, dan IoT menuntut penerapan *responsive design* dan *accessibility* agar sistem dapat digunakan pada berbagai perangkat dan oleh beragam kelompok pengguna. Perancang juga perlu mengikuti proses sistematis, mulai dari analisis kebutuhan, *persona*, *wireframe*, *mockup*, *prototype*, hingga pengujian *usability*.

Desain antarmuka juga berkaitan erat dengan basis data karena digunakan untuk memasukkan, mengubah, mencari, dan menampilkan data. Oleh sebab itu, UI dan basis data perlu dirancang secara terpadu agar aliran data berlangsung konsisten, akurat, dan efisien. Dengan demikian, desain antarmuka merupakan salah satu faktor kunci dalam menghasilkan sistem informasi yang efektif dan berorientasi pada kebutuhan pengguna.

3.2 Peran Basis Data sebagai Pusat Penyimpanan Informasi

Basis data (*database*) merupakan komponen inti sistem informasi yang berfungsi menyimpan, mengelola, dan menyediakan data untuk mendukung proses bisnis. Data terstruktur memungkinkan organisasi menjaga konsistensi, keakuratan, ketersediaan, serta mendukung pengambilan keputusan berbasis data.

Pengelolaan basis data dilakukan melalui Database Management System (DBMS), yang menyediakan mekanisme penyimpanan, manipulasi, keamanan, dan pengendalian akses data. DBMS juga mendukung akses data secara bersamaan dengan tetap menjaga integritas dan konsistensi data (Elmasri & Navathe, 2016).

Perancangan basis data yang baik mengurangi redundansi dan inkonsistensi, meningkatkan keamanan, serta memudahkan pemeliharaan. Teknik seperti normalisasi, *primary key*, *foreign key*, dan *integrity constraints* digunakan untuk menjaga efisiensi dan keakuratan data. Perkembangan teknologi juga melahirkan berbagai pendekatan seperti *cloud database*, *distributed database*, NoSQL, dan *real-time database* untuk memenuhi kebutuhan skalabilitas dan kecepatan pengolahan data.

3.3 Hubungan antara UI, Proses Bisnis, dan Database

Keberhasilan sistem informasi bergantung pada integrasi antara *User Interface*, proses bisnis, dan basis data. UI menjadi media interaksi pengguna, proses bisnis menerapkan aturan organisasi, sedangkan basis data menyimpan dan menyediakan data yang diperlukan. Ketiganya membentuk alur kerja yang saling berkaitan dalam menghasilkan layanan informasi yang efektif.

Sebagai contoh, pada sistem informasi akademik, registrasi mahasiswa meliputi pengisian formulir, validasi persyaratan, penyimpanan data, hingga penerbitan bukti registrasi. Setiap perubahan pada struktur basis data atau prosedur bisnis perlu diikuti penyesuaian UI dan logika aplikasi agar seluruh komponen tetap konsisten.

Dalam arsitektur modern, hubungan tersebut umumnya diterapkan melalui pola *Model-View-Controller* (MVC) atau three-tier architecture, yang memisahkan lapisan presentasi, logika bisnis, dan penyimpanan data. Pemisahan ini meningkatkan kemudahan pengembangan, pemeliharaan, pengujian, dan skalabilitas sistem.

3.4 Tantangan dalam Merancang Sistem Modern

Sistem informasi modern berkembang dari sekadar pencatatan transaksi menjadi platform yang mendukung kolaborasi, analisis data, layanan daring, komputasi awan, dan kecerdasan buatan. Perkembangan tersebut menghadirkan beberapa tantangan yaitu:

1. Pengalaman pengguna, yang menuntut antarmuka mudah dipelajari, responsif, konsisten, dan dapat digunakan pada berbagai perangkat.
2. Volume dan kompleksitas data, yang membutuhkan basis data dengan skalabilitas, performa, dan kemampuan analisis real-time.
3. Keamanan informasi, yang memerlukan autentikasi, otorisasi, enkripsi, audit trail, dan pengendalian hak akses.
4. Integrasi sistem, yang membutuhkan API, *web services*, dan mekanisme interoperabilitas untuk pertukaran data.
5. Adaptabilitas sistem, yang menuntut kemampuan merespons perubahan regulasi, proses bisnis, dan teknologi melalui pendekatan seperti Agile, DevOps, dan arsitektur berbasis layanan.

Dengan demikian, perancangan sistem informasi modern harus mengintegrasikan kebutuhan pengguna, proses bisnis, basis data, keamanan, interoperabilitas, dan kemampuan adaptasi. Pendekatan tersebut menghasilkan

sistem yang berkualitas, mudah dikembangkan, dan mampu memberikan nilai tambah bagi organisasi.

3.6 Konsep Dasar Perancangan Antarmuka

Perancangan antarmuka pengguna (*User Interface Design/UI*) merupakan tahap penting dalam pengembangan sistem informasi untuk membangun interaksi antara pengguna dan sistem. UI tidak hanya berfungsi sebagai tampilan visual, tetapi juga membantu pengguna menyelesaikan tugas secara efektif, efisien, dan nyaman. Karena itu, kualitas UI berpengaruh terhadap penerimaan, produktivitas, dan keberlanjutan penggunaan sistem.

Perancangan UI modern tidak berfokus pada estetika, tetapi juga mempertimbangkan ergonomi, psikologi kognitif, *Human-Computer Interaction* (HCI), *User Experience* (UX), dan kebutuhan bisnis.

3.6.1 Pengertian Antarmuka Pengguna

1. Definisi User Interface

User Interface (UI) adalah media interaksi manusia dengan sistem komputer yang memungkinkan pengguna memberikan input, mengendalikan proses, dan menerima output dalam bentuk informasi yang mudah dipahami. UI menjadi penghubung antara kebutuhan pengguna dan fungsi sistem (Pressman & Maxim, 2020; Shneiderman *et al*., 2016).

Dalam perspektif HCI, UI mencakup GUI serta berbagai bentuk interaksi seperti suara (*Voice User Interface*), sentuhan (*Touch Interface*), gestur, dan realitas virtual. Komponen utamanya meliputi kontrol input, navigasi, penyajian informasi, *feedback*, dan validasi.

2. Tujuan User Interface

Tujuan utama UI adalah menciptakan interaksi yang efektif sehingga pengguna dapat memahami dan mengoperasikan sistem secara optimal. Tujuan tersebut meliputi:

a. Mempermudah interaksi, melalui menu, ikon, dan navigasi yang jelas serta konsisten.
b. Meningkatkan efisiensi kerja, dengan mengurangi langkah, waktu, dan usaha pengguna.
c. Mengurangi kesalahan, melalui validasi, petunjuk, dan pesan kesalahan yang jelas.
d. Meningkatkan kepuasan, melalui tampilan yang mudah digunakan, menarik, dan responsif.
e. Mendukung tujuan organisasi, dengan menyediakan sistem yang efektif, efisien, dan berkualitas.

3. Fungsi User Interface dalam Sistem Informasi

UI berfungsi sebagai penghubung antara pengguna, proses bisnis, dan basis data. Fungsi utamanya meliputi:

1. Media komunikasi antara pengguna dan sistem.
2. Media input data melalui formulir dan komponen interaktif.
3. Media penyajian informasi melalui tabel, grafik, dashboard, dan laporan.
4. Media navigasi untuk berpindah antarfitur.
5. Media pengendalian proses bisnis melalui validasi dan konfirmasi.

6. Media feedback untuk memberikan informasi mengenai status, keberhasilan, atau kesalahan proses.

UI yang berkualitas meningkatkan efektivitas interaksi, produktivitas, dan mengurangi kesalahan pengguna.

3.6.2 *User Interface* (UI) vs *User Experience* (UX)

UI dan UX saling berkaitan tetapi memiliki fokus berbeda. UI menekankan tampilan visual dan mekanisme interaksi, sedangkan UX mencakup keseluruhan pengalaman pengguna, termasuk persepsi, kenyamanan, kemudahan, efisiensi, dan kepuasan.

Tabel 3. 1 Perbedaan Mendasar Antara UI dan UX

Aspek	*User Interface* (UI)	*User Experience* (UX)
Fokus	Tampilan visual dan interaksi	Pengalaman pengguna secara keseluruhan
Tujuan	Menciptakan antarmuka yang menarik dan mudah digunakan	Menciptakan pengalaman yang efektif, efisien, dan memuaskan
Ruang Lingkup	Warna, ikon, tombol, tipografi, layout	Kebutuhan pengguna, alur kerja, kepuasan, usability
Orientasi	Estetika dan interaksi	Perilaku serta persepsi pengguna
Hasil Akhir	Tampilan aplikasi	Pengalaman menggunakan aplikasi

Dengan demikian, UI berfokus pada bagaimana sistem terlihat dan digunakan, sedangkan UX berfokus pada bagaimana sistem dirasakan dan dialami pengguna.

a) Hubungan UI dan UX

UI dan UX harus dirancang secara terpadu. Tampilan menarik tidak menjamin UX yang baik jika navigasi membingungkan atau proses terlalu kompleks. Sebaliknya, UX yang baik membutuhkan UI yang jelas, konsisten, dan mudah dipahami.

Dalam praktiknya, UX umumnya diawali dengan analisis kebutuhan, *user persona*, *user journey*, dan *wireframe*, kemudian diterjemahkan menjadi desain visual melalui UI Design.

b) Studi Kasus: Aplikasi Perpustakaan Digital

Pada aplikasi perpustakaan dengan tombol pencarian kecil, navigasi tersembunyi, kontras rendah, dan formulir yang terlalu panjang, pengguna akan lebih mudah mengalami kesulitan dan kesalahan. Setelah evaluasi *usability*, perbaikan dapat dilakukan melalui tombol yang lebih jelas, navigasi sederhana, kontras yang lebih baik, pengurangan kolom input, dan fitur *auto-complete*. Hasilnya adalah waktu pencarian lebih singkat, kesalahan berkurang, dan kepuasan meningkat.

3.6.3 Jenis-Jenis Antarmuka

Pemilihan jenis UI harus disesuaikan dengan tujuan sistem, karakteristik pengguna, lingkungan penggunaan, dan perangkat. Beberapa jenis UI yang umum adalah sebagai berikut.

1. Graphical User Interface (GUI)

GUI memungkinkan pengguna berinteraksi melalui elemen visual seperti ikon, tombol, menu, dan jendela menggunakan mouse, touchpad, atau sentuhan. GUI mengurangi beban kognitif dan memudahkan proses belajar (Shneiderman *et al*., 2016).

- Karakteristik: ikon visual, menu dan tombol, navigasi intuitif, WYSIWYG, dan feedback langsung.
- Kelebihan: mudah dipelajari, interaktif, mengurangi kesalahan, mendukung multitasking, dan meningkatkan produktivitas.
- Kekurangan: membutuhkan sumber daya lebih besar, pengembangan relatif kompleks, dan kurang efisien untuk pekerjaan administratif berulang.
- Contoh: Windows, macOS, Ubuntu Desktop, Microsoft Office, Photoshop, Sistem Informasi Akademik, perbankan, dan ERP.

2. Command Line Interface (CLI)

CLI memungkinkan pengguna berinteraksi melalui perintah teks pada terminal atau konsol. CLI banyak digunakan oleh administrator sistem dan pengembang karena memberikan kontrol dan fleksibilitas tinggi.

- Karakteristik: berbasis teks, menggunakan sintaks, tanpa elemen grafis, mendukung script, dan hemat sumber daya.
- Kelebihan: cepat, efisien, fleksibel, ringan, dan mendukung otomatisasi.
- Kekurangan: membutuhkan pembelajaran, pengguna harus memahami perintah, rentan

syntax error, dan kurang ramah bagi pengguna umum.

- Contoh: Command Prompt, PowerShell, Linux Terminal, macOS Terminal, Git, dan Docker CLI.

3. Menu Driven Interface

Menu Driven Interface menggunakan daftar menu sebagai sarana utama interaksi sehingga pengguna cukup memilih fungsi yang tersedia. Antarmuka ini sederhana dan cocok bagi pengguna umum.

- Karakteristik: menu bertingkat, pilihan terstruktur, mudah dipelajari, dan mengurangi kesalahan input.
- Kelebihan: sederhana, mudah digunakan, cocok bagi pengguna awam, dan memperpendek pelatihan.
- Kekurangan: kurang fleksibel, menu yang terlalu banyak dapat membingungkan, dan navigasi dapat menjadi panjang.
- Contoh: ATM, kios layanan mandiri, POS, sistem perpustakaan, dan menu televisi digital.

4. Touch Interface

Touch Interface memungkinkan interaksi langsung melalui layar sentuh dan mendukung gestur seperti *tap*, *swipe*, *drag*, *pinch*, dan *zoom*. Jenis UI ini banyak digunakan pada smartphone, tablet, mesin absensi, dan kios digital.

- Kelebihan: alami, cepat, sederhana, mendukung mobilitas.
- Kekurangan: kurang nyaman untuk pengetikan panjang, rentan sidik jari, dan membutuhkan desain tombol yang ergonomis.

5. *Voice User Interface* (VUI)

VUI memungkinkan interaksi menggunakan suara dengan dukungan *speech recognition*, NLP, dan AI. VUI cocok untuk penggunaan *hands-free* serta dapat meningkatkan aksesibilitas.

- Kelebihan: alami, mendukung multitasking, dan membantu aksesibilitas.
- Kekurangan: dipengaruhi kebisingan, aksen, pelafalan, serta kebutuhan koneksi dan pemrosesan.
- Contoh: Google Assistant, Siri, Alexa, sistem navigasi kendaraan, dan layanan pelanggan berbasis suara.

6. *Web Interface*

Web Interface merupakan antarmuka yang diakses melalui browser menggunakan HTML, CSS, JavaScript, dan framework web. Sistem dapat digunakan melalui internet atau intranet tanpa instalasi khusus.

- Kelebihan: lintas sistem operasi, mudah diperbarui, dapat diakses dari berbagai lokasi, dan mendukung kolaborasi daring.
- Kekurangan: bergantung pada jaringan, browser, dan keamanan aplikasi web.
- Contoh: Sistem Informasi Akademik, LMS, e-commerce, perbankan daring, portal pemerintahan, dan ERP berbasis web.

7. *Mobile Interface*

Mobile Interface dirancang untuk smartphone dan tablet dengan mengutamakan penggunaan pada layar kecil dan kondisi mobilitas. Pendekatan yang digunakan antara lain *mobile-first*, *responsive design*,

serta pemanfaatan GPS, kamera, biometrik, dan sensor.

- Kelebihan: fleksibel, mendukung mobilitas, memanfaatkan fitur perangkat, dan memberikan pengalaman personal.
- Kekurangan: ukuran layar terbatas, bergantung pada baterai dan penyimpanan, serta memerlukan optimasi untuk berbagai ukuran layar.
- Contoh: mobile banking, e-wallet, transportasi daring, marketplace, aplikasi produktivitas, dan Sistem Informasi Akademik Android/iOS.

3.7 Prinsip-Prinsip Perancangan Antarmuka

Perancangan antarmuka tidak hanya berfokus pada estetika, tetapi juga pada efektivitas interaksi pengguna dengan sistem. Antarmuka yang baik membantu pengguna bekerja secara cepat, akurat, dan nyaman, sedangkan desain yang buruk dapat meningkatkan kesalahan dan menurunkan produktivitas. Dalam HCI, prinsip perancangan UI didasarkan pada perilaku pengguna, psikologi kognitif, ergonomi, dan pengalaman pengembangan perangkat lunak. Prinsip utama meliputi konsistensi, *usability*, *learnability*, *visibility*, *feedback*, *error prevention*, *flexibility*, *accessibility*, dan *responsive design*.

3.7.1 Prinsip Konsistensi (*Consistency*)

Konsistensi berarti penggunaan pola desain, tata letak, istilah, ikon, warna, dan mekanisme interaksi yang seragam di seluruh sistem. Konsistensi membantu pengguna mengenali pola dan mengurangi kebutuhan mempelajari kembali cara menggunakan sistem. Prinsip

ini merupakan salah satu *Eight Golden Rules of Interface Design* yang dapat mengurangi beban kognitif dan meningkatkan efisiensi (Shneiderman *et al*., 2016).

Tujuan**:** mempermudah pengenalan pola, mempercepat pembelajaran dan penyelesaian tugas, mengurangi kesalahan, serta meningkatkan kenyamanan.

Implementasi: gunakan warna untuk fungsi yang sama secara konsisten, pertahankan posisi tombol dan menu, gunakan ikon dengan makna yang sama, serta seragamkan format tanggal, waktu, dan tipografi.

Contoh: pada Sistem Informasi Akademik, tombol Simpan menggunakan warna yang sama di seluruh modul, tombol Hapus menggunakan warna berbeda secara konsisten, dan menu utama ditempatkan pada posisi yang tetap.

3.7.2 Kemudahan Penggunaan (*Usability*)

Usability adalah kemampuan sistem untuk dipelajari dan digunakan dalam mencapai tujuan secara efektif, efisien, dan memuaskan. ISO 9241-11 mendefinisikannya berdasarkan efektivitas, efisiensi, dan kepuasan dalam konteks penggunaan tertentu.

Menurut Nielsen, komponen utama *usability* meliputi:

- *Learnability*: mudah dipelajari.
- *Efficiency*: pekerjaan dapat diselesaikan dengan cepat.
- *Memorability*: mudah diingat saat digunakan kembali.

- *Errors*: kesalahan dapat diminimalkan dan ditangani.
- *Satisfaction*: pengguna merasa nyaman dan puas.

Implementasi**:** sederhanakan alur kerja, gunakan bahasa yang mudah dipahami, sediakan navigasi jelas dan formulir ringkas, serta lakukan usability testing secara berkala.

Contoh: aplikasi pembayaran yang menyelesaikan transaksi dalam tiga langkah umumnya lebih mudah digunakan dibandingkan aplikasi dengan banyak tahapan dan navigasi rumit.

3.7.3 *Learnability*

Learnability adalah kemampuan sistem untuk dipelajari dengan cepat oleh pengguna baru. Semakin tinggi *learnability*, semakin singkat waktu yang diperlukan untuk memahami sistem.

Faktor utam**a:** istilah familiar, ikon mudah dipahami, navigasi konsisten, tata letak sederhana, dan petunjuk yang jelas.

Strategi**:** gunakan *onboarding tutorial*, ikon umum, menu logis, desain minimalis, dan *help system*.

Contoh**:** aplikasi perpustakaan dengan menu Cari Buku, Pinjam, dan Riwayat Peminjaman pada halaman utama memungkinkan pengguna memahami fungsi utama tanpa pelatihan khusus.

3.7.4 *Visibility*

Visibility berarti fungsi penting sistem harus mudah ditemukan dan terlihat sehingga pengguna tidak perlu menebak lokasi fitur. Prinsip ini meningkatkan kemungkinan fungsi digunakan dengan benar (Norman, 2013).

Implementasi: sediakan menu utama yang jelas, ikon mudah dikenali, tombol berukuran memadai, kontras yang baik, dan tempatkan informasi penting pada posisi strategis.

Contoh: pada e-commerce, tombol Tambah ke Keranjang ditempatkan dekat informasi produk dan dibuat mudah dikenali pengguna.

3.7.5 *Feedback*

Feedback adalah respons sistem terhadap tindakan pengguna yang menunjukkan status atau hasil suatu proses. Feedback membantu pengguna mengetahui bahwa sistem telah menerima dan memproses perintah.

Bentuk *feedback* dapat berupa pesan sukses, indikator proses, notifikasi kesalahan, suara, perubahan warna tombol, atau animasi sederhana.

Contoh: setelah data mahasiswa disimpan, sistem menampilkan pesan "Data berhasil disimpan." Jika terjadi kesalahan validasi, sistem menjelaskan penyebab dan cara memperbaikinya.

3.7.6 *Error Prevention*

Error Prevention merupakan prinsip untuk mencegah kesalahan sebelum terjadi, bukan sekadar memberikan pesan setelah kesalahan dilakukan.

Strategi: validasi otomatis, *drop-down list*, pembatasan format input, konfirmasi sebelum penghapusan, dan penonaktifan tombol yang belum dapat digunakan.

Contoh: pada formulir registrasi mahasiswa, kolom NIM hanya menerima angka dengan jumlah digit tertentu sehingga kesalahan dapat diketahui sebelum data dikirim ke server.

3.7.7 *Flexibility*

Flexibility adalah kemampuan UI mengakomodasi kebutuhan, preferensi, dan tingkat pengalaman pengguna yang berbeda.

Implementasi: sediakan *keyboard shortcut*, personalisasi tampilan, beberapa metode input, ekspor berbagai format, dan dukungan lintas perangkat.

Contoh: Microsoft Excel menyediakan menu grafis dan *keyboard shortcut* sehingga pengguna dapat memilih cara kerja yang paling sesuai.

3.7.8 *Accessibility*

Accessibility adalah kemampuan sistem untuk digunakan oleh semua kelompok pengguna, termasuk penyandang disabilitas. WCAG menekankan pentingnya akses bagi pengguna dengan keterbatasan penglihatan, pendengaran, motorik, maupun kognitif.

Implementasi: gunakan kontras yang memadai, ukuran huruf yang dapat disesuaikan, *screen reader*, navigasi keyboard, *alternative text*, dan bahasa sederhana.

Contoh: portal layanan publik menyediakan pengaturan ukuran huruf, mode kontras tinggi, dan kompatibilitas *screen reader* agar dapat digunakan oleh pengguna dengan keterbatasan penglihatan.

3.7.9 *Responsive Design*

Responsive Design adalah pendekatan yang memungkinkan antarmuka menyesuaikan ukuran layar, resolusi, dan orientasi perangkat secara otomatis sehingga pengalaman penggunaan tetap konsisten pada desktop, laptop, tablet, maupun smartphone.

Tujuan: meningkatkan kenyamanan lintas perangkat, mengurangi kebutuhan desain terpisah, memperluas akses, dan meningkatkan kepuasan pengguna.

Implementasi: gunakan *flexible grid*, *fluid images*, *responsive typography*, CSS *media queries*, dan pendekatan *mobile-first*.

Contoh: Sistem Informasi Akademik dapat menampilkan menu horizontal pada desktop dan mengubahnya menjadi *hamburger menu* pada smartphone untuk menyesuaikan ruang layar.

3.8 Komponen Antarmuka Sistem Informasi

Antarmuka pengguna (*User Interface/UI*) terdiri atas komponen visual dan interaktif yang mendukung komunikasi pengguna dengan sistem. Setiap komponen berfungsi mengatur tampilan informasi, memudahkan navigasi, menerima input, dan memberikan *feedback*. Perancangannya perlu memperhatikan *usability*, konsistensi, estetika,

aksesibilitas, dan *User Experience (UX)* agar sistem mudah dipelajari dan digunakan.

Komponen UI bukan sekadar elemen visual, tetapi bagian dari mekanisme interaksi manusia dengan komputer. Karena itu, pemilihan dan penempatannya harus disesuaikan dengan kebutuhan pengguna dan proses bisnis.

3.8.1 Layout

1. Pengertian

   Layout adalah pengaturan posisi, ukuran, dan hubungan antar-elemen pada halaman atau layar. Layout yang baik membantu pengguna memahami struktur informasi dan menemukan fungsi dengan cepat. Prinsip utamanya meliputi hierarki visual, *balance*, *proximity*, dan *alignment* (Lidwell, Holden, & Butler, 2010).

2. Tujuan
   - Menyusun informasi secara sistematis.
   - Memudahkan pencarian fitur.
   - Meningkatkan efisiensi interaksi.
   - Menciptakan tampilan profesional.
   - Mengurangi kesalahan navigasi.

3. Prinsip
   - *Alignment*: menyelaraskan posisi elemen.
   - *Proximity*: mengelompokkan elemen yang berkaitan.
   - *White Space*: menyediakan ruang kosong yang cukup.

- *Hierarchy*: menunjukkan prioritas informasi.
- *Balance*: menjaga keseimbangan visual.

4. Contoh

   Pada Sistem Informasi Akademik, menu utama ditempatkan di sisi kiri, profil pengguna di bagian atas, dan konten utama di tengah agar navigasi lebih mudah.

3.8.2 *Typography*

1. Pengertian

   Typography adalah teknik memilih dan mengatur teks agar mudah dibaca sekaligus mendukung estetika. Unsurnya meliputi jenis, ukuran, ketebalan, warna, spasi, dan perataan teks.

2. Unsur Utama

   - Font Family: jenis huruf, seperti Arial, Roboto, atau Open Sans.
   - Font Size: ukuran teks.
   - Font Weight: ketebalan teks.
   - Line Spacing: jarak antarbaris.
   - Letter Spacing: jarak antarhuruf.
   - Text Alignment: posisi atau perataan teks.

3. Prinsip

   - Gunakan maksimal 2–3 jenis huruf.
   - Pilih ukuran yang mudah dibaca.
   - Pastikan kontras teks dan latar memadai.
   - Hindari penggunaan huruf kapital berlebihan.

- Gunakan variasi ukuran dan ketebalan untuk membentuk hierarki.

4. Contoh

   Judul halaman dapat menggunakan huruf *bold* berukuran 24 pt, sedangkan isi menggunakan sans-serif berukuran 12–14 pt untuk meningkatkan keterbacaan.

3.8.3 Warna

1. Pengertian

   Warna berfungsi memperkuat identitas sistem, membedakan informasi, menarik perhatian, dan membangun hierarki visual.

2. Fungsi

   - Memperjelas hierarki informasi.
   - Membedakan status atau kategori.
   - Menekankan aksi tertentu.
   - Memperkuat identitas organisasi.
   - Meningkatkan daya tarik visual.

3. Prinsip

   - Gunakan palet warna yang konsisten.
   - Hindari terlalu banyak warna.
   - Pastikan kontras memenuhi kebutuhan aksesibilitas.
   - Jangan menjadikan warna sebagai satu-satunya penanda informasi.

4. Contoh
    - Hijau: berhasil.
    - Merah: kesalahan atau peringatan.
    - Kuning: perhatian atau menunggu.
    - Biru: menu atau tautan.

3.8.4 Ikon

1. Pengertian

    Ikon adalah simbol visual yang mewakili fungsi, objek, atau tindakan. Ikon membantu pengguna mengenali fungsi tanpa harus membaca teks panjang.

2. Karakteristik

    Ikon yang baik harus sederhana, mudah dikenali, konsisten, memiliki makna yang umum, dan sesuai konteks.

3. Fungsi
    - Mempermudah navigasi.
    - Menghemat ruang.
    - Mempercepat pengenalan fungsi.
    - Meningkatkan daya tarik visual.
4. Contoh
    - *Home*: halaman utama.
    - *Search*: pencarian.
    - *Delete*: menghapus.
    - *Settings*: pengaturan.

3.8.5 Button

1. Pengertian

   Button adalah komponen yang digunakan untuk memicu aksi atau proses, seperti menyimpan, menghapus, mencetak, atau mengirim data.

2. Jenis

   - *Primary Button*: aksi utama.
   - *Secondary Button*: aksi pendukung.
   - *Icon Button*: tombol berbasis ikon.
   - *Floating Action Button* (FAB): tombol aksi utama yang melayang.
   - Toggle Button: mengaktifkan atau menonaktifkan fungsi.

3. Prinsip

   - Ukuran mudah diklik.
   - Label singkat dan jelas.
   - Warna menunjukkan prioritas.
   - Memberikan hover/pressed state.
   - Konsisten di seluruh aplikasi.

4. Contoh

   Formulir registrasi mahasiswa dapat menggunakan tombol Simpan, Reset, dan Batal dengan fungsi dan tampilan yang berbeda secara konsisten.

3.8.6 Form Input

1. Pengertian

   Form Input adalah kumpulan komponen untuk memasukkan data ke sistem. Rancangan formulir yang baik membantu meningkatkan keakuratan data dalam basis data.

2. Komponen

   Text Box, *Password Field*, *Text Area*, *Radio Button*, *Check Box*, *Drop-down List*, *Date Picker*, *File Upload*, dan *Search Box*.

3. Prinsip

   - Gunakan label yang jelas.
   - Kelompokkan data berdasarkan kategori.
   - Terapkan validasi secara langsung.
   - Minimalkan kolom yang tidak diperlukan.
   - Berikan petunjuk pengisian bila diperlukan.

4. Contoh

   Formulir mahasiswa cukup meminta data yang diperlukan, seperti NIM, nama, program studi, email, dan nomor telepon. Format email dapat divalidasi sebelum formulir dikirim.

3.8.7 *Navigation*

1. Pengertian

   Navigasi adalah mekanisme untuk berpindah antarhalaman atau fungsi. Navigasi yang baik membantu pengguna menemukan informasi tanpa kebingungan.

2. Jenis
    - *Navigation Bar*
    - *Sidebar Navigation*
    - *Breadcrumb*
    - *Tab Navigation*
    - *Hamburger Menu*
    - *Footer Navigation*
3. Prinsip

    Navigasi harus mudah ditemukan, konsisten, sederhana, mendukung orientasi pengguna, dan menunjukkan posisi halaman saat ini.
4. Contoh

    Sistem Informasi Akademik dapat menyediakan menu Dashboard, Data Mahasiswa, KRS, Nilai, Keuangan, dan Laporan secara konsisten pada setiap halaman.

3.8.8 Dashboard

1. Pengertian

    Dashboard adalah halaman yang menyajikan informasi penting secara ringkas melalui indikator, grafik, tabel, atau visualisasi sehingga pengguna dapat memahami kondisi sistem dengan cepat.
2. Fungsi
    - Menyajikan informasi utama.
    - Mendukung pengambilan keputusan.
    - Memantau indikator kinerja.

- Mempercepat akses ke fitur penting.

3. Karakteristik

   Dashboard yang baik harus sederhana, informatif, interaktif, mudah dipahami, dan mendukung visualisasi data.

4. Contoh

   Dashboard Sistem Informasi Akademik dapat menampilkan jumlah mahasiswa aktif, jumlah dosen, jadwal kuliah, grafik pembayaran UKT, dan notifikasi akademik.

3.8.9 *Notification*

1. Pengertian

   Notification adalah mekanisme untuk menyampaikan status proses, peringatan, pembaruan, atau aktivitas sistem kepada pengguna.

2. Jenis

   - *Success Notification*
   - *Warning Notification*
   - *Error Notification*
   - *Information Notification*
   - *Push Notification*
   - *Email Notification*
   - *Toast Notification*

3. Prinsip

   Notifikasi harus menggunakan bahasa yang jelas, muncul pada waktu yang tepat, tidak

berlebihan, memiliki pembeda tingkat urgensi, dan memberikan solusi jika terjadi kesalahan.

4. Contoh

   Setelah mahasiswa berhasil mengisi KRS, sistem menampilkan "Pengisian KRS berhasil disimpan." Jika terjadi kesalahan, sistem memberikan informasi spesifik, misalnya "Mata kuliah tidak dapat diambil karena prasyarat belum terpenuhi."

3.9 Proses Perancangan Antarmuka

Perancangan antarmuka (*User Interface Design*) merupakan proses sistematis untuk menghasilkan media interaksi yang efektif, efisien, mudah dipahami, dan mampu memberikan pengalaman pengguna (*User Experience*) yang positif. Proses ini diawali dengan analisis kebutuhan pengguna dan dilakukan secara iteratif hingga menghasilkan desain yang sesuai dengan kebutuhan pengguna serta tujuan organisasi.

Pendekatan *User-Centered Design* (UCD) menempatkan pengguna sebagai fokus utama dalam pengembangan antarmuka. ISO 9241-210:2019 menekankan pemahaman konteks penggunaan, identifikasi kebutuhan pengguna, pengembangan solusi, dan evaluasi berulang untuk menghasilkan desain yang sesuai dengan kebutuhan pengguna serta mendukung proses bisnis. (ISO)

Secara umum, proses perancangan antarmuka meliputi tujuh tahap: analisis kebutuhan pengguna, user persona, user journey, wireframe, mockup, prototype, dan evaluasi prototype. Setiap tahap menghasilkan keluaran yang menjadi dasar bagi tahap berikutnya.

3.9.1 Analisis Kebutuhan Pengguna

1. Pengertian

   Analisis kebutuhan pengguna merupakan tahap awal untuk mengidentifikasi karakteristik pengguna, tujuan penggunaan, lingkungan operasional, serta kebutuhan fungsional dan nonfungsional sistem. Hasilnya menjadi dasar penentuan interaksi, navigasi, dan penyajian informasi.

   Menurut Pressman dan Maxim (2020), keberhasilan desain antarmuka dipengaruhi oleh pemahaman terhadap pengguna, aktivitas, dan permasalahan yang dihadapi. Oleh karena itu, analisis kebutuhan mencakup fungsi sistem sekaligus perilaku dan harapan pengguna.

2. Tujuan Analisis Kebutuhan Pengguna

   Analisis kebutuhan bertujuan untuk:

   - mengidentifikasi karakteristik pengguna;
   - memahami proses bisnis;
   - mengetahui tujuan penggunaan sistem;
   - mengidentifikasi permasalahan pengguna;
   - menentukan kebutuhan informasi dan fitur.

3. Teknik Pengumpulan Data

   Teknik yang umum digunakan meliputi:

   - Wawancara, untuk menggali kebutuhan secara mendalam;
   - Observasi, untuk memahami aktivitas pengguna secara langsung;

- Kuesioner, untuk memperoleh masukan dari banyak responden;
- *Focus Group Discussion* (FGD), untuk memperoleh perspektif berbagai pemangku kepentingan;
- Analisis dokumen, seperti SOP, formulir, laporan, dan kebijakan organisasi.

4. Hasil Analisis

Hasil analisis meliputi:

- profil pengguna;
- user requirements;
- kebutuhan fungsional dan nonfungsional;
- prioritas fitur;
- rekomendasi awal rancangan antarmuka.

Contoh Kasus:

Pada Sistem Informasi Akademik, mahasiswa membutuhkan akses cepat ke jadwal, KRS, nilai, dan tagihan. Dosen membutuhkan fitur pengisian nilai dan presensi, sedangkan administrator memerlukan pengelolaan data akademik. Perbedaan kebutuhan tersebut menjadi dasar perancangan antarmuka sesuai karakteristik setiap pengguna.

3.9.2 User Persona

1. Pengertian

User Persona merupakan representasi fiktif pengguna berdasarkan hasil penelitian untuk menggambarkan karakteristik, tujuan, perilaku, kebutuhan, dan kendala pengguna. Persona

membantu pengembang mengambil keputusan berdasarkan kebutuhan pengguna, bukan asumsi.

2. Tujuan User Persona

   Persona bertujuan untuk:

   - memahami karakteristik pengguna;
   - menghindari desain yang berorientasi pada pengembang;
   - mempermudah pengambilan keputusan desain;
   - meningkatkan pengalaman pengguna.

3. Komponen User Persona

   Komponen persona umumnya meliputi:

   - nama dan usia;
   - pekerjaan atau status;
   - pendidikan;
   - kemampuan teknologi;
   - tujuan dan kebutuhan;
   - pain points;
   - motivasi pengguna.

Contoh User Persona

Nama : Andi Pratama

Usia : 20 tahun

Status : Mahasiswa Semester 4

Tujuan : Mengisi KRS dan melihat nilai dengan cepat.

Hambatan : Sulit menemukan jadwal dan informasi pembayaran karena terlalu banyak menu.

Harapan : Menu sederhana, pencarian cepat, dan akses melalui telepon pintar. Persona tersebut membantu menentukan prioritas fitur, struktur navigasi, dan penyajian informasi yang relevan.

3.9.3 *User Journey*

1. Pengertian

   User Journey merupakan representasi visual tahapan interaksi pengguna untuk mencapai tujuan tertentu. Pemetaan ini mencakup aktivitas, *touchpoints*, emosi, hambatan, dan peluang perbaikan.

2. Tujuan User Journey

   User Journey bertujuan untuk:

   - memahami alur interaksi;
   - mengidentifikasi hambatan proses;
   - menemukan peluang peningkatan pengalaman pengguna;
   - menyelaraskan kebutuhan pengguna dengan fitur sistem.

3. Komponen User Journey

   Komponen umumnya meliputi:

   - tujuan pengguna;
   - tahapan aktivitas;
   - tindakan pengguna;
   - touchpoints;
   - emosi;
   - pain points;

- peluang perbaikan.

4. Contoh *User Journey*

   Alur pengisian KRS pada Sistem Informasi Akademik meliputi:

   - membuka halaman login;
   - memasukkan akun dan kata sandi;
   - melakukan autentikasi;
   - memilih menu KRS;
   - melihat daftar mata kuliah;
   - memilih mata kuliah;
   - memvalidasi prasyarat dan kapasitas kelas;
   - menyimpan data KRS;
   - menerima notifikasi keberhasilan.

Pemetaan tersebut membantu mengidentifikasi tahapan yang berpotensi menimbulkan kesalahan atau keterlambatan sehingga dapat diperbaiki pada desain maupun proses bisnis.

3.9.4 *Wireframe*

1. Pengertian

   Wireframe merupakan rancangan awal yang menggambarkan struktur halaman, posisi komponen, dan alur navigasi tanpa elemen visual seperti warna, ikon, dan tipografi. *Wireframe* berfungsi sebagai *blueprint* antarmuka.

2. Tujuan Wireframe

   Wireframe bertujuan untuk:

   - menentukan struktur halaman;
   - menyusun layout;
   - memvalidasi navigasi;
   - mempermudah diskusi dengan pemangku kepentingan;
   - mengurangi perubahan saat implementasi.

3. Karakteristik Wireframe

   Karakteristik *wireframe* meliputi:

   - bentuk sederhana;
   - tanpa warna;
   - fokus pada struktur informasi;
   - mudah direvisi;
   - cepat dibuat.

4. Contoh *Wireframe* halaman login dapat memuat logo, kolom nama pengguna, kata sandi, tombol Masuk, dan tautan Lupa Kata Sandi. Rancangan ini digunakan untuk mengevaluasi susunan komponen sebelum desain visual.

3.9.5 *Mockup*

1. Pengertian

   Mockup merupakan pengembangan *wireframe* yang dilengkapi elemen visual seperti warna, tipografi, ikon, gambar, dan identitas organisasi. *Mockup* memberikan gambaran tampilan sistem yang lebih realistis, tetapi belum interaktif.

2. Tujuan Mockup

   Mockup digunakan untuk:

   - menampilkan desain visual;
   - mengevaluasi estetika antarmuka;
   - memperoleh masukan pengguna dan klien;
   - memastikan konsistensi identitas visual.

3. Karakteristik Mockup

   Karakteristik *mockup* meliputi:

   - menggunakan warna dan ikon;
   - menerapkan tipografi sebenarnya;
   - menyerupai tampilan aplikasi final;
   - belum memiliki fungsi interaktif.

4. Contoh *Mockup* dashboard Sistem Informasi Akademik dapat menampilkan warna institusi, ikon menu, grafik jumlah mahasiswa, jadwal kuliah, dan informasi akademik dalam tata letak yang menyerupai aplikasi sebenarnya.

3.9.6 *Prototype*

1. Pengertian

   Prototype merupakan model awal aplikasi yang memiliki kemampuan interaktif sehingga pengguna dapat mencoba alur sistem sebelum implementasi. Prototype memungkinkan simulasi fungsi utama tanpa membangun sistem secara penuh.

   ISO 9241-210:2019 menempatkan evaluasi dan umpan balik pengguna sebagai bagian penting dalam proses desain berpusat pada manusia. (ISO)

2. Tujuan Prototype

   Prototype bertujuan untuk:

   - menguji rancangan antarmuka;
   - mengevaluasi alur penggunaan;
   - memperoleh umpan balik pengguna;
   - mengurangi biaya perubahan saat implementasi.

3. Jenis Prototype

   - *Low-Fidelity:* sketsa atau wireframe interaktif sederhana.
   - *Medium-Fidelity:* memiliki struktur dan navigasi lebih lengkap.
   - *High-Fidelity:* menyerupai aplikasi akhir dari sisi visual dan interaksi.

Contoh: Prototype Sistem Informasi Perpustakaan memungkinkan pengguna mencoba pencarian, peminjaman, dan pengembalian buku menggunakan data simulasi tanpa terhubung ke basis data sebenarnya.

3.9.7 Evaluasi Prototype

1. Pengertian

   Evaluasi prototype merupakan proses penilaian untuk memastikan desain memenuhi kebutuhan pengguna dan mendukung proses bisnis. Evaluasi dilakukan secara iteratif agar hasil pengujian dapat digunakan untuk memperbaiki desain sebelum implementasi.

2. Tujuan Evaluasi

   Evaluasi bertujuan untuk:

   - mengidentifikasi masalah usability;
   - menilai efektivitas desain;
   - mengukur kepuasan pengguna;
   - mengurangi risiko kegagalan implementasi.

3. Metode Evaluasi

   Metode yang dapat digunakan meliputi:

   - *Usability Testing,* mengamati pengguna saat menyelesaikan tugas tertentu;
   - *Heuristic Evaluation,* mengevaluasi desain berdasarkan prinsip usability;
   - *Cognitive Walkthrough,* menilai kemudahan penggunaan dari perspektif pengguna baru;
   - *Think Aloud Protocol,* meminta pengguna mengungkapkan pikirannya saat menggunakan prototype;
   - *User Acceptance Testing* (UAT), memastikan rancangan memenuhi kebutuhan pengguna akhir.

Contoh: Evaluasi prototype Sistem Informasi Akademik menunjukkan mahasiswa kesulitan menemukan menu pembayaran UKT. Perbaikan dilakukan dengan memindahkan menu ke dashboard utama dan memperjelas ikonnya. Hasilnya, waktu pencarian berkurang dan kepuasan pengguna meningkat.

3.10 Tool Perancangan Antarmuka

Perkembangan teknologi digital telah melahirkan berbagai perangkat lunak (*tools*) yang dirancang khusus untuk membantu proses perancangan antarmuka (*User Interface Design*). Pada masa lalu, rancangan antarmuka umumnya dibuat menggunakan perangkat lunak pengolah gambar atau bahkan secara manual menggunakan kertas. Namun, meningkatnya *kompleksitas* sistem informasi menuntut adanya alat yang mampu mendukung kolaborasi tim, mempercepat proses desain, menghasilkan *prototype* interaktif, serta memudahkan evaluasi sebelum tahap implementasi.

Saat ini tersedia berbagai aplikasi yang dapat digunakan oleh analis sistem, perancang UI/UX, maupun pengembang perangkat lunak. Masing-masing memiliki karakteristik, *keunggulan*, serta keterbatasan yang berbeda sehingga pemilihannya harus disesuaikan dengan kebutuhan proyek, kompleksitas sistem, kemampuan tim, dan anggaran yang tersedia.

Subbab ini membahas beberapa tools yang paling banyak digunakan dalam industri *maupun* lingkungan akademik, yaitu **Figma**, **Adobe XD**, **Balsamiq**, **Sketch**, **Canva**, dan **Draw.io**.

1. Figma

a) Pengertian **:** Figma merupakan aplikasi desain antarmuka berbasis *cloud* yang memungkinkan beberapa pengguna bekerja secara kolaboratif dalam waktu yang sama (*real-time collaboration*). Figma menjadi salah satu tools paling populer dalam pengembangan UI/UX karena mendukung proses desain, pembuatan *wireframe*, *mockup*, *prototype*, hingga *design handoff* kepada pengembang. Berbeda dengan perangkat lunak konvensional, Figma dapat

dijalankan melalui peramban (*browser*) maupun aplikasi desktop tanpa bergantung pada sistem operasi tertentu.

b) Fitur Utama **:** Figma menyediakan berbagai fitur al:

- *Frame* dan *Auto Layout*
- *Components* dan *Variants*
- *Interactive Prototype*
- *Design System*
- *Team Collaboration*
- *Version History*
- *Plugin* dan *Widget*
- *Developer Mode*

c) Contoh Penggunaan

Figma banyak digunakan untuk mendesain:

- Sistem Informasi Akademik
- Dashboard Manajemen
- Mobile Banking
- Marketplace
- Aplikasi Rumah Sakit
- Website Pemerintahan

2. Adobe XD

a) Pengertia**n :** Adobe XD (*Experience Design*) merupakan perangkat lunak yang dikembangkan oleh Adobe untuk mendesain antarmuka pengguna serta membuat prototype interaktif. Adobe XD banyak digunakan oleh desainer profesional yang

telah terbiasa menggunakan ekosistem Adobe seperti Photoshop dan Illustrator.

b) Fitur Utama **:** Adobe XD memiliki fitur:

- *Artboard*
- *Repeat Grid*
- *Interactive Prototype*
- *Auto Animate*
- *Voice Prototype*
- *Component*
- *Responsive Resize*
- *Share Prototype*

c) Contoh Penggunaan

Adobe XD sesuai untuk:

- Aplikasi Mobile
- Website Perusahaan
- Dashboard Analitik
- Sistem Informasi Berbasis Web

3. Balsamiq

a) Pengertian : Balsamiq merupakan perangkat lunak yang difokuskan untuk pembuatan *wireframe* dengan gaya gambar tangan (low-fidelity wireframe). Tujuan utama Balsamiq adalah membantu tim berkonsentrasi pada struktur, fungsi, dan alur aplikasi tanpa terganggu oleh aspek visual.

b) Fitur Utama

- *Wireframe Components*
- *UI Controls*
- *Drag and Drop*
- *Mockup Library*
- *Collaboration*

c) Contoh Penggunaan

- Wireframe Sistem Informasi Akademik
- Wireframe ATM
- Wireframe Aplikasi Perpustakaan
- Wireframe Sistem Penjualan

4. Sketch

a) Pengertian : Sketch merupakan aplikasi desain antarmuka berbasis vektor yang dikembangkan khusus untuk sistem operasi macOS. Sketch menjadi salah satu tools populer dalam industri UI/UX karena memiliki performa tinggi serta dukungan plugin yang sangat banyak.

b) Fitur Utama

- *Symbol*
- *Shared Library*
- *Component*
- *Artboard*
- *Prototype*
- *Plugin*

c) Contoh Penggunaan

Sketch banyak digunakan pada pengembangan:

- Aplikasi iOS
- Website Perusahaan
- Dashboard Enterprise

5. Canva

a) Pengertian : Canva merupakan aplikasi desain grafis berbasis web yang dikenal karena kemudahan penggunaannya. Walaupun bukan tools khusus UI/UX, Canva sering dimanfaatkan untuk membuat mockup sederhana, presentasi desain, ikon, infografik, maupun aset visual aplikasi.

b) Fitur Utama

- *Template*
- *Icon Library*
- *Image Library*
- *Collaboration*
- *Presentasi*
- *Mockup*

c) Contoh Penggunaan

- Presentasi Prototype
- Mockup Landing Page
- Desain Dashboard Sederhana
- Infografik Sistem Informasi

6. Draw.io (diagrams.net)

a) Pengertian : Draw.io (sekarang dikenal sebagai diagrams.net) merupakan aplikasi berbasis web yang digunakan untuk membuat berbagai jenis diagram, termasuk flowchart, UML, BPMN, ERD, DFD, dan wireframe sederhana. Dalam analisis dan perancangan sistem informasi, Draw.io banyak digunakan untuk memvisualisasikan proses bisnis dan struktur sistem.

b) Fitur Utama

- *Flowchart*
- *UML Diagram*
- *Entity Relationship Diagram*
- *BPMN*
- *Mind Map*
- *Wireframe Library*

c) Contoh Penggunaan

- Diagram Alir Sistem
- Use Case Diagram
- Activity Diagram
- Wireframe Login
- Entity Relationship Diagram

7. Contoh Pembuatan Prototype Sederhana

Sebagai ilustrasi, berikut disajikan tahapan pembuatan prototype sederhana untuk **Sistem Informasi Perpustakaan** menggunakan Figma. Studi kasus ini dapat diterapkan pula pada Adobe XD maupun Sketch karena konsep perancangannya relatif serupa.

a) Studi Kasus

Sistem Informasi Perpustakaan Digital memiliki tiga kebutuhan utama:

- Mahasiswa melakukan login.
- Mahasiswa mencari buku.
- Mahasiswa melakukan peminjaman buku.

b) Tahapan Pembuatan Prototype

Langkah 1. Analisis Kebutuhan

Pengguna utama adalah mahasiswa dan pustakawan.

Fitur utama:

- Login
- Dashboard
- Pencarian Buku
- Detail Buku
- Peminjaman Buku

Langkah 2. Membuat Wireframe

Buat lima halaman utama:

- Login
- Dashboard
- Daftar Buku
- Detail Buku
- Konfirmasi Peminjaman

Pada tahap ini cukup menggunakan bentuk sederhana berupa kotak, garis, dan teks tanpa warna.

Langkah 3. Membuat Mockup

Tambahkan elemen visual:

- Logo perpustakaan
- Warna identitas institusi
- Ikon pencarian
- Gambar sampul buku
- Tombol aksi
- Tipografi yang konsisten

Mockup memberikan gambaran antarmuka yang mendekati aplikasi akhir.

Langkah 4. Membuat Prototype Interaktif

Hubungkan setiap halaman menggunakan fitur Prototype.

Alur interaksi yang dirancang sebagai berikut:

Login -> Dashboard -> Daftar Buku -> Detail Buku -> Konfirmasi Peminjaman -> Notifikasi Berhasil.

Pengguna dapat mengklik tombol Masuk, memilih buku, melihat detail, menekan tombol Pinjam, kemudian menerima notifikasi bahwa proses peminjaman berhasil

Langkah 5. Melakukan Pengujian

Prototype diuji oleh lima hingga sepuluh calon pengguna dengan beberapa skenario, misalnya:

- melakukan login;
- mencari buku berdasarkan judul;
- meminjam buku;
- membatalkan peminjaman.

Selama pengujian diamati waktu penyelesaian tugas, jumlah kesalahan, kemudahan navigasi, dan kepuasan pengguna. Masukan yang diperoleh digunakan untuk menyempurnakan desain sebelum implementasi.



Gambar 3. 1 Contoh Perancangan Antarmuka

3.11 Pengujian Antarmuka

Pengujian antarmuka merupakan tahap penting untuk memastikan sistem tidak hanya berfungsi secara teknis, tetapi juga mudah dipahami, dipelajari, efisien, mampu mencegah kesalahan, dan sesuai kebutuhan pengguna.

Pengujian berkaitan erat dengan *usability*, yaitu sejauh mana sistem dapat digunakan oleh pengguna tertentu untuk mencapai tujuan secara efektif (*effectiveness*), efisien (*efficiency*), dan memuaskan (*satisfaction*) dalam konteks penggunaan tertentu (ISO 9241-11:2018).

Dengan demikian, pengujian tidak hanya memeriksa fungsi tombol atau tampilan halaman, tetapi juga mengevaluasi keberhasilan tugas, waktu penyelesaian, kesalahan, kenyamanan, kepuasan, dan kesesuaian dengan proses bisnis.

Empat metode yang umum digunakan adalah Usability Testing, Heuristic Evaluation, User Acceptance Test (UAT), dan Think Aloud Method. Keempatnya memiliki fokus berbeda dan dapat digunakan secara komplementer.

3.11.1 *Usability Testing*

Usability Testing merupakan pengujian yang melibatkan pengguna atau calon pengguna secara langsung untuk menyelesaikan tugas tertentu. Tujuannya adalah menilai apakah pengguna dapat mencapai tujuan secara efektif, efisien, dan memuaskan.

Contoh pada Sistem Informasi Akademik: pengguna diminta login, mencari mata kuliah Basis Data, dan melakukan pengisian KRS, sementara penguji mengamati proses tanpa memberikan bantuan di luar protokol pengujian.

a. Tujuan *Usability Testing*

Usability Testing bertujuan untuk:

- Mengukur keberhasilan penyelesaian tugas.
- Mengukur efisiensi berdasarkan waktu atau sumber daya.
- Mengidentifikasi kesalahan pengguna.

- Menemukan masalah antarmuka.
- Mengukur kepuasan pengguna.
- Membandingkan alternatif desain.
- Menjadi dasar perbaikan antarmuka.

Indikator yang umum digunakan meliputi task success rate, time on task, jumlah error, bantuan yang diperlukan, dan kepuasan pengguna (ScienceDirect).

b. Komponen *Usability Testing*

Komponen	Penjelasan	Contoh Indikator
Effectiveness	Kemampuan menyelesaikan tugas dengan benar	*Task success rate*
Efficiency	Waktu atau sumber daya yang digunakan	*Time on task*
Satisfaction	Persepsi dan kepuasan pengguna	Kuesioner
Error	Kesalahan selama interaksi	Jumlah kesalahan
Learnability	Kemudahan mempelajari sistem	Waktu belajar, keberhasilan tugas
Assistance	Bantuan yang diperlukan	Jumlah intervensi moderator

c. Tahapan *Usability Testing*

1. Menentukan tujuan pengujian, misalnya navigasi, pencarian, transaksi, atau pengisian formulir.

2. Menentukan pengguna, sesuai karakteristik target sistem.
3. Menyusun skenario dan task yang realistis.
4. Menyiapkan instrumen, seperti lembar observasi, pencatat waktu, rekaman layar, dan kuesioner.
5. Melaksanakan pengujian dengan mengamati interaksi pengguna.
6. Menganalisis hasil berdasarkan keberhasilan, waktu, error, komentar, dan kepuasan.
7. Memperbaiki desain berdasarkan prioritas masalah.

d. Contoh Perhitungan Task Success Rate

Misalnya lima pengguna menjalankan lima tugas dengan hasil:

Pengguna	Tugas Berhasil	Total Tugas
P1	5	5
P2	4	5
P3	5	5
P4	4	5
P5	3	5
Total	21	25

Rumus Task Success Rate **=**

(Jumlah tugas berhasil / Total tugas) × 100%

= (21 / 25) × 100% = 84%

Artinya, tingkat keberhasilan penyelesaian tugas adalah **84%**. Nilai tersebut selanjutnya dibandingkan dengan target yang ditetapkan dalam penelitian atau proyek.

3.11.2 *Heuristic Evaluation*

Heuristic Evaluation merupakan evaluasi antarmuka berdasarkan prinsip atau heuristik *usability*. Berbeda dengan Usability Testing, metode ini umumnya dilakukan oleh evaluator yang memahami usability dan desain interaksi.

Salah satu pendekatan yang banyak digunakan adalah **10 *Usability Heuristics for User Interface Design*** dari Jakob Nielsen.

No	Heuristik	Inti Prinsip
1	Visibility of System Status	Sistem memberikan informasi tentang kondisi/proses
2	Match Between System and the Real World	Bahasa dan konsep sesuai pengguna
3	User Control and Freedom	Pengguna dapat membatalkan atau kembali
4	Consistency and Standards	Elemen dan mekanisme konsisten
5	Error Prevention	Mencegah kesalahan sebelum terjadi
6	Recognition Rather Than Recall	Informasi penting mudah dikenali
7	Flexibility and Efficiency of Use	Mendukung pengguna pemula dan berpengalaman

No	Heuristik	Inti Prinsip
8	Aesthetic and Minimalist Design	Informasi relevan dan tidak berlebihan
9	Recognize, Diagnose, and Recover from Errors	Pesan kesalahan jelas dan solutif
10	Help and Documentation	Menyediakan bantuan yang relevan

a. Contoh *Heuristic Evaluation*

Jika login dengan password salah menghasilkan pesan **"Error 104"**, evaluator dapat menilainya sebagai masalah pada heuristik *Recognize, Diagnose, and Recover from Errors*. Pesan sebaiknya diganti menjadi:

"Username atau password yang Anda masukkan salah. Silakan periksa kembali data login Anda."

b. *Severity Rating*

Nilai	Tingkat	Interpretasi
0	Tidak ada masalah	Tidak ditemukan pelanggaran
1	Cosmetic	Masalah visual kecil
2	Minor	Masalah usability kecil
3	Major	Masalah serius yang menghambat pengguna

Nilai	Tingkat	Interpretasi
4	Catastrophic	Masalah sangat serius yang dapat menggagalkan proses utama

Severity rating digunakan untuk menentukan prioritas perbaikan.

3.11.3 *User Acceptance Test* (UAT)

User Acceptance Test (UAT) adalah pengujian untuk memastikan sistem memenuhi kebutuhan dan persyaratan yang disepakati serta dapat diterima oleh pengguna atau pemilik proses bisnis. UAT umumnya dilakukan menjelang akhir pengembangan sebelum sistem diterapkan.

Berbeda dengan *Usability Testing* yang berfokus pada kemudahan dan pengalaman penggunaan, UAT berfokus pada kesesuaian sistem dengan kebutuhan dan proses bisnis.

a. Tujuan UAT

- Memastikan kebutuhan pengguna telah diimplementasikan.
- Memastikan fungsi sesuai proses bisnis.
- Memastikan sistem siap digunakan secara operasional.
- Menemukan ketidaksesuaian sebelum rilis.
- Memperoleh persetujuan pengguna.
- Menjadi dasar keputusan implementasi.

b. Aktor UAT

Pihak yang dapat terlibat meliputi Business Owner, End User, Project Manager, System Analyst, Developer, dan Tester/QA.

c. Tahapan UAT

- Menentukan *acceptance criteria*.
- Menyusun skenario pengujian.
- Menyiapkan *test case*.
- Melaksanakan pengujian.
- Mencatat hasil (*Accepted, Rejected,* atau *Need Improvement*).
- Melakukan perbaikan.
- Melakukan *retesting*.
- Melakukan *acceptance/sign-off* apabila kriteria terpenuhi.

Contoh *acceptance criteria*: "Sistem harus memungkinkan mahasiswa mengisi KRS hanya untuk mata kuliah yang ditawarkan pada semester aktif."

d. Contoh Format UAT

No.	Skenario	Expected Result	Actual Result	Status
1	Login akun valid	Dashboard tampil	Dashboard tampil	Accepted
2	Password salah	Pesan kesalahan tampil	Pesan tampil	Accepted

No.	Skenario	Expected Result	Actual Result	Status
3	Tambah mahasiswa	Data tersimpan	Data tersimpan	Accepted
4	Edit mahasiswa	Data diperbarui	Data diperbarui	Accepted
5	Hapus mahasiswa	Data terhapus setelah konfirmasi	Data terhapus	Accepted

Jika seluruh skenario kritis ***Accepted***, sistem dapat dipertimbangkan untuk diimplementasikan dengan tetap memperhatikan hasil pengujian lainnya.

3.11.4 *Think Aloud Method*

Think Aloud Method merupakan metode evaluasi usability dengan meminta pengguna mengungkapkan apa yang dipikirkan, dipahami, diharapkan, atau dirasakan saat menjalankan tugas. Metode ini membantu evaluator memahami alasan di balik tindakan pengguna.

Contoh pernyataan pengguna pada Sistem Informasi Akademik: "Saya ingin melihat jadwal kuliah, tetapi tidak tahu apakah menu ini untuk jadwal atau kalender." Pernyataan tersebut dapat menunjukkan masalah pada label menu, navigasi, atau *information architecture*.

a. Tujuan *Think Aloud*

- Memahami cara pengguna menafsirkan antarmuka.
- Mengidentifikasi ekspektasi pengguna.
- Mengetahui alasan suatu tindakan.

- Menemukan istilah atau ikon yang membingungkan.
- Mengidentifikasi titik kesulitan.
- Memahami persepsi terhadap menu, pesan, dan navigasi.
- Menghasilkan data kualitatif untuk perbaikan desain.

b. Prosedur *Think Aloud*

- Menentukan tugas yang realistis.
- Menjelaskan metode kepada peserta.
- Meminta peserta mengungkapkan pikirannya selama tugas.
- Mengamati tanpa mengarahkan.
- Mencatat tindakan, kesalahan, waktu, kebingungan, komentar, dan ekspektasi.
- Mengelompokkan temuan berdasarkan navigasi, terminologi, feedback, layout, form, dan error handling.

c. Contoh

Pengguna diminta mencari mahasiswa dengan NIM 23010025, kemudian mengatakan: "Saya mencari menu mahasiswa. Sepertinya Data Master, tetapi saya kurang yakin." Temuan tersebut menunjukkan bahwa pengguna berhasil menemukan fungsi, tetapi struktur navigasinya belum sepenuhnya intuitif. Perbaikan dapat berupa:

Sebelum : Data Master → Data Mahasiswa

Alternatif : Akademik → Mahasiswa

3.11.5 Perbandingan Metode Pengujian Antarmuka

Aspek	Usability Testing	Heuristic Evaluation	UAT	Think Aloud
Fokus	Kemudahan penggunaan	Prinsip usability	Penerimaan sistem	Proses berpikir pengguna
Pelak sana	Pengguna	Evaluator/ UX expert	Pengguna/ pemilik bisnis	Pengguna
Data	Kuantitatif & kualitatif	Umumnya kualitatif	Pass/ fail & catatan	Kualitatif
Fokus utama	*Task performance*	Prinsip desain	Kebutuhan bisnis	*Mental model*
Hasil	Metrik & masalah usability	Daftar masalah heuristik	Status penerimaan	Insight perilaku
Cocok untuk	Evaluasi penggunaan	Review desain	Validasi sebelum implementasi	Memahami kebingungan pengguna

Keempat metode dapat digunakan secara berurutan dan saling melengkapi:

Heuristic Evaluation → Perbaikan UI → Usability Testing + Think Aloud → Perbaikan → UAT → Implementasi

Heuristic Evaluation menemukan masalah berdasarkan prinsip desain, *Usability Testing* mengukur kinerja pengguna, *Think Aloud* menggali alasan dan persepsi pengguna, sedangkan UAT memastikan kesesuaian dengan kebutuhan bisnis.

3.11.6 Integrasi Keempat Metode

Dalam Sistem Informasi Akademik, integrasi pengujian dapat dilakukan sebagai berikut:

1. Heuristic Evaluation pada login, dashboard, KRS, jadwal, nilai, dan profil untuk menemukan masalah navigasi, istilah, feedback, dan *error handling*.
2. Usability Testing kepada mahasiswa dan dosen dengan skenario penggunaan nyata.
3. Think Aloud selama pengujian untuk memahami penyebab kebingungan pengguna.
4. Perbaikan UI berdasarkan temuan pengujian.
5. UAT bersama pengguna atau pemilik proses akademik berdasarkan *acceptance criteria*.

Pendekatan ini menjadikan pengujian sebagai proses **iteratif** sehingga masalah dapat ditemukan dan diperbaiki sebelum sistem diterapkan secara luas.

3.11.7 Contoh Rancangan Pengujian Antarmuka

Misalnya Sistem Informasi Perpustakaan memiliki fitur login, pencarian buku, peminjaman, pengembalian, dan pencetakan laporan.

Metode	Aktivitas	Responden/ Evaluator	Output
Heuristic Evaluation	Evaluasi berdasarkan 10 heuristik Nielsen	2–5 evaluator	Daftar masalah usability
Usability Testing	Pencarian dan peminjaman buku	Mahasiswa/ petugas	Success rate, waktu, error

Metode	Aktivitas	Responden/ Evaluator	Output
Think Aloud	Mengungkapkan pikiran saat menjalankan tugas	Mahasiswa/ petugas	Insight kualitatif
UAT	Validasi fungsi berdasarkan kebutuhan bisnis	Petugas/ pemilik proses	Accepted/ Rejected

Hasil pengujian menjadi dasar untuk menentukan apakah antarmuka perlu diperbaiki, dilanjutkan ke pengujian berikutnya, atau telah memenuhi kriteria implementasi. Pengujian antarmuka merupakan bagian integral dari pengembangan sistem informasi. Keberhasilan sistem tidak hanya ditentukan oleh fungsi teknis, tetapi juga oleh efektivitas, efisiensi, kepuasan, kemudahan penggunaan, dan kesesuaian dengan kebutuhan pengguna (ISO 9241-11:2018).

Usability Testing mengukur kinerja pengguna dalam menyelesaikan tugas; Heuristic Evaluation menemukan masalah berdasarkan prinsip usability; UAT memvalidasi kesesuaian sistem dengan kebutuhan dan proses bisnis; sedangkan Think Aloud menggali proses berpikir dan kebingungan pengguna. Kombinasi keempat metode menghasilkan evaluasi yang lebih komprehensif dari aspek desain, perilaku pengguna, pengalaman interaksi, dan kebutuhan bisnis.

3.12 Konsep Dasar Basis Data

Basis data merupakan salah satu komponen fundamental dalam sistem informasi karena berfungsi sebagai pusat penyimpanan, pengelolaan, dan penyediaan data yang dibutuhkan oleh organisasi. Hampir seluruh sistem informasi modern, seperti sistem informasi akademik, sistem informasi keuangan, e-commerce, rumah sakit, perpustakaan, hingga aplikasi media sosial, membutuhkan basis data untuk menyimpan dan mengelola informasi secara terstruktur.

Dalam konteks analisis dan perancangan sistem informasi, basis data tidak hanya dipandang sebagai kumpulan tabel yang menyimpan data. Basis data merupakan bagian integral dari arsitektur sistem yang harus dirancang selaras dengan kebutuhan pengguna, proses bisnis, aturan organisasi, serta antarmuka aplikasi. Perancangan basis data yang baik akan mendukung konsistensi data, mengurangi redundansi, meningkatkan keamanan, dan memudahkan pengembangan sistem di masa mendatang.

Secara sederhana, hubungan antara pengguna, aplikasi, DBMS, dan basis data dapat digambarkan sebagai berikut:

Pengguna → Antarmuka Aplikasi → Proses Bisnis → DBMS → Basis Data → DBMS → Aplikasi → Pengguna

Dengan demikian, pemahaman mengenai konsep dasar basis data menjadi landasan penting sebelum membahas perancangan ERD, normalisasi, transformasi model data, struktur tabel, maupun implementasi basis data.

3.12.1 Pengertian Basis Data

> Basis data atau **database** adalah sekumpulan data yang saling berhubungan, disimpan secara terorganisasi, dan dikelola sedemikian rupa sehingga data dapat disimpan, dicari, diperbarui, dan digunakan kembali secara efektif sesuai kebutuhan.

Data dalam basis data tidak berdiri sendiri. Data biasanya memiliki hubungan tertentu dengan data lainnya. Sebagai contoh, dalam sistem informasi akademik terdapat data mahasiswa, dosen, mata kuliah, kelas, KRS, dan nilai. Data mahasiswa berhubungan dengan KRS, KRS berhubungan dengan mata kuliah, sedangkan nilai berhubungan dengan mahasiswa dan mata kuliah.

Sebagai ilustrasi, data mahasiswa dapat memiliki struktur sederhana seperti berikut:

NIM	Nama	Program Studi	Angkatan
230001	Andi Pratama	Teknik Informatika	2023
230002	Siti Rahma	Teknik Informatika	2023
230003	Budi Santoso	Arsitektur	2023

Tabel tersebut merupakan salah satu bentuk representasi data dalam basis data relasional. Dalam sistem yang lebih kompleks, tabel mahasiswa dapat dihubungkan dengan tabel program studi, KRS, mata kuliah, dosen, dan nilai.

a) Basis Data dan Data

Perlu dibedakan antara **data** dan **basis data**. **Data** merupakan fakta atau informasi mentah yang menggambarkan suatu objek atau kejadian. Contohnya: NIM:230001, Nama: Andi Pratama, Program Studi: Teknik Informatika Sementara itu, **basis data** merupakan kumpulan data yang telah diorganisasi sehingga dapat dikelola dan digunakan secara sistematis. Dengan demikian: Data merupakan isi, sedangkan basis data merupakan

lingkungan terstruktur untuk menyimpan dan mengelola data tersebut.

b) Basis Data dalam Sistem Informasi

Dalam sebuah sistem informasi, basis data berfungsi sebagai pusat penyimpanan informasi. Aplikasi dapat mengambil data dari basis data untuk ditampilkan kepada pengguna dan mengirimkan data baru ke basis data ketika pengguna melakukan transaksi.

Contohnya pada sistem informasi akademik:

- ✓ Mahasiswa mengisi KRS.
- ✓ Sistem menerima data KRS.
- ✓ Aplikasi melakukan validasi.
- ✓ DBMS menyimpan data KRS.
- ✓ Data KRS dapat digunakan untuk menghasilkan kartu studi.
- ✓ Data yang sama dapat digunakan untuk proses perkuliahan dan pelaporan akademik.

Oleh karena itu, basis data harus dirancang agar mampu mendukung kebutuhan informasi secara konsisten dan berkelanjutan.

3.12.2 Tujuan Basis Data

Penggunaan basis data bertujuan untuk menyediakan mekanisme pengelolaan data yang terstruktur, efisien, aman, dan dapat digunakan oleh berbagai bagian dalam organisasi.

Beberapa tujuan utama basis data adalah sebagai berikut.

1. Mengurangi Redundansi Data

Redundansi merupakan kondisi ketika data yang sama disimpan berulang kali tanpa kebutuhan yang jelas. Misalnya data mahasiswa disimpan berulang pada tabel KRS, pembayaran, dan nilai. Jika nama mahasiswa berubah, seluruh data yang berulang tersebut harus diperbarui. Dengan desain basis data yang baik, data mahasiswa cukup disimpan pada tabel mahasiswa dan tabel lain cukup menyimpan **NIM** sebagai referensi.

2. Menjaga Konsistensi Data

Basis data harus memastikan bahwa informasi yang disimpan tidak saling bertentangan. Sebagai contoh, apabila seorang mahasiswa bernama "Andi Pratama" telah tersimpan dalam tabel mahasiswa, sistem tidak seharusnya menghasilkan data berbeda seperti "Andi Pratama Putra" pada tabel lain tanpa dasar perubahan yang sah.

Konsistensi dapat dibantu melalui:

- primary key;
- foreign key;
- constraint;
- validasi data;
- aturan bisnis.

3. Memudahkan Akses terhadap Data

Basis data memungkinkan pengguna atau aplikasi memperoleh informasi dengan cepat. Misalnya administrator akademik dapat menjalankan pencarian: "Tampilkan seluruh

mahasiswa Teknik Informatika angkatan 2024." DBMS akan memproses permintaan tersebut menggunakan mekanisme query.

4. Meningkatkan Keamanan Data

Tidak semua pengguna memiliki hak akses yang sama. Basis data memungkinkan administrator mengatur siapa yang dapat:

- melihat data;
- menambahkan data;
- mengubah data;
- menghapus data;
- melakukan administrasi basis data.

Sebagai contoh:

Pengguna	Hak Akses
Administrator	Semua data
Operator Akademik	Data mahasiswa dan akademik
Dosen	Data perkuliahan dan nilai
Mahasiswa	Data akademik pribadi

5. Mendukung Integrasi Data

Basis data memungkinkan berbagai modul aplikasi menggunakan sumber data yang sama. Contohnya: **Data Mahasiswa → Akademik → Keuangan → Perpustakaan → Pelaporan.** Dengan integrasi tersebut, organisasi tidak perlu membuat basis data yang terpisah untuk setiap unit jika kebutuhan informasinya dapat dikelola dalam arsitektur data yang terintegrasi.

6. Mendukung Pengambilan Keputusan

Data yang tersimpan dalam basis data dapat diolah menjadi informasi untuk mendukung pengambilan keputusan.

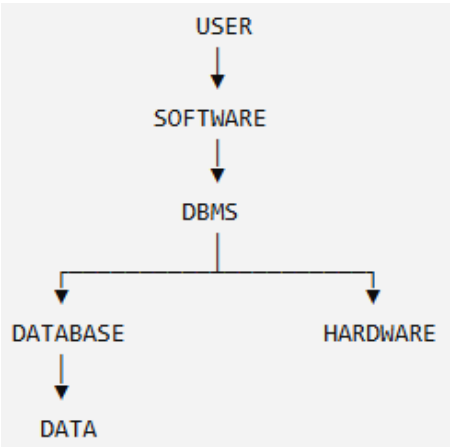
Contohnya:

- jumlah mahasiswa aktif;
- tingkat kelulusan;
- jumlah mahasiswa berdasarkan program studi;
- tingkat pembayaran mahasiswa;
- statistik penggunaan perpustakaan;
- tren pendaftaran mahasiswa baru.

Dengan demikian, basis data tidak hanya berfungsi sebagai tempat penyimpanan data, tetapi juga menjadi fondasi bagi sistem Business Intelligence, analitik data, dan pengambilan keputusan.

3.12.3 Komponen Basis Data

Sebuah sistem basis data terdiri atas beberapa komponen yang saling berinteraksi. Komponen tersebut meliputi **data, database, DBMS, *user*, *hardware*, dan *software***. Secara konseptual dapat digambarkan:



Gambar 3. 2 Komponen Basis Data

A. Data

Data merupakan komponen utama yang disimpan dan dikelola dalam sistem basis data. Data dapat berupa:

- teks;
- angka;
- tanggal;
- gambar;
- audio;
- video;
- dokumen;
- koordinat lokasi;
- data sensor;
- data transaksi.

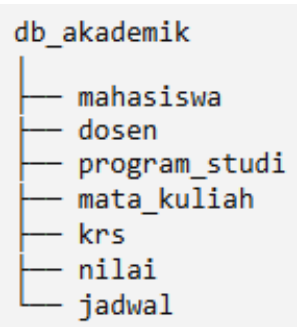
Dalam sistem informasi akademik, data dapat berupa:

- data mahasiswa;
- data dosen;
- data program studi;
- data mata kuliah;
- data KRS;
- data nilai;
- data jadwal;
- data pembayaran

Data merupakan aset penting organisasi sehingga kualitas data harus dijaga.

B. Database

Database merupakan kumpulan data yang terorganisasi dan saling berhubungan serta disimpan dalam suatu sistem penyimpanan. Contohnya sebuah database bernama: **db_akademik,** dapat memiliki tabel:



Setiap tabel dapat memiliki hubungan dengan tabel lainnya.

C. DBMS

Database Management System (DBMS) adalah perangkat lunak yang digunakan untuk membuat, mengelola, mengakses, mengamankan, dan memelihara basis data. Contoh DBMS:

- MySQL;
- PostgreSQL;
- Microsoft SQL Server;
- Oracle Database;
- MariaDB;
- SQLite;
- MongoDB untuk kategori database NoSQL.

DBMS menyediakan berbagai fungsi, seperti:

- membuat database;
- membuat tabel;
- memasukkan data;
- mengubah data;
- menghapus data;
- mencari data;
- mengatur hak akses;
- menjaga integritas data;
- melakukan backup dan recovery;
- mengelola transaksi.

Dalam database relasional, interaksi dengan DBMS umumnya dilakukan menggunakan SQL (*Structured Query Language*).

D. User

User adalah pihak yang menggunakan atau berinteraksi dengan sistem basis data. User dapat dikategorikan menjadi beberapa kelompok.

1. Database Administrator: bertanggung jawab terhadap pengelolaan basis data, termasuk: keamanan; backup; recovery; konfigurasi; monitoring; optimasi kinerja; pengelolaan hak akses.
2. Application Developer: menggunakan DBMS untuk membangun aplikasi yang berinteraksi dengan database.
3. End User: adalah pengguna akhir aplikasi, misalnya: mahasiswa; dosen; staf; administrator; pelanggan.

E. *Hardware*

Hardware merupakan perangkat fisik yang digunakan untuk menjalankan sistem basis data. Contohnya: server; komputer; storage; hard disk; SSD; RAM; perangkat jaringan; router; switch; perangkat backup. Dalam sistem berskala besar, database dapat dijalankan pada server dengan kapasitas tinggi dan menggunakan sistem penyimpanan yang memiliki redundansi untuk meningkatkan keandalan.

F. *Software*

Software merupakan perangkat lunak yang mendukung pengoperasian sistem basis data. Software dapat meliputi: sistem operasi; DBMS; aplikasi server; application framework; middleware; perangkat monitoring; software backup; aplikasi pengguna.

Contohnya:

```
User
  ↓
Web Browser
  ↓
Web/Application Server
  ↓
DBMS
  ↓
Database
```

Arsitektur tersebut banyak digunakan dalam sistem informasi berbasis web.

3.12.4 Karakteristik Basis Data

Basis data yang baik memiliki karakteristik tertentu yang membedakannya dari sekadar kumpulan file biasa.

1. Terorganisasi: Data disimpan dengan struktur tertentu sehingga mudah dikelola dan ditemukan.
2. Saling Berhubungan: Data dalam database memiliki hubungan logis.
3. Memiliki Integritas Data : Integritas memastikan data tetap benar, valid, dan konsisten. Contohnya, nilai mahasiswa tidak boleh diberikan kepada NIM yang tidak terdaftar.
4. Mengurangi Redundansi : Data yang sama tidak perlu disimpan secara berulang apabila dapat direpresentasikan melalui hubungan antarentitas.
5. Mendukung Data Sharing : Data dapat digunakan oleh beberapa pengguna atau aplikasi sesuai dengan hak akses yang diberikan.
6. Memiliki Keamanan : Basis data harus melindungi data dari akses yang tidak sah. Mekanisme keamanan dapat meliputi: autentikasi; otorisasi; role-based access control; enkripsi; audit log; backup.
7. Mendukung Concurrency: Database modern memungkinkan banyak pengguna mengakses data pada waktu yang bersamaan. Sebagai contoh, dalam sistem akademik, banyak mahasiswa dapat mengakses sistem untuk melakukan KRS secara bersamaan. DBMS harus mampu mengelola akses tersebut sehingga tidak menyebabkan konflik data.
8. Mendukung Backup dan Recovery: Basis data harus memiliki mekanisme untuk memulihkan data apabila terjadi: kerusakan hardware; kesalahan

pengguna; kegagalan sistem; serangan keamanan; kehilangan data.

9. *Data Independence*: merupakan kemampuan untuk melakukan perubahan tertentu pada struktur penyimpanan tanpa harus mengubah seluruh aplikasi yang menggunakan data tersebut. Konsep ini menjadi penting dalam sistem informasi yang terus berkembang.
10. *Scalable*. Basis data harus dapat dikembangkan mengikuti pertumbuhan: jumlah pengguna; jumlah transaksi; jumlah data; jumlah aplikasi. Kemampuan tersebut dikenal sebagai scalability.

3.12.5 Jenis Basis Data

Perkembangan teknologi informasi menyebabkan munculnya berbagai model basis data. Pemilihan jenis database harus disesuaikan dengan karakteristik data, kebutuhan aplikasi, skala sistem, pola transaksi, serta kebutuhan performa.

Lima jenis basis data yang umum dibahas adalah relasional, NoSQL, object database, distributed database, dan cloud database.

A. Basis Data Relasional

Relational Database merupakan basis data yang menyimpan data dalam bentuk tabel yang terdiri atas baris dan kolom. Model relasional menggunakan hubungan antar tabel untuk merepresentasikan keterkaitan data. Contohnya:

Tabel Mahasiswa

NIM	Nama	ID_Prodi
230001	Andi	P01
230002	Budi	P01

Tabel Program Studi

ID_Prodi	Nama_Prodi
P01	Teknik Informatika
P02	Arsitektur

Relasi kedua tabel dapat dilakukan melalui atribut **ID_Prodi**. Karakteristik utama database relasional: menggunakan tabel; memiliki primary key; menggunakan foreign key; mendukung SQL; memiliki struktur data yang relatif terdefinisi; mendukung transaksi dengan kuat.

Contoh penggunaan : sistem informasi akademik; sistem keuangan; sistem rumah sakit; sistem perbankan; sistem inventori.

B. Basis Data NoSQL

NoSQL (Not Only SQL) merupakan kelompok basis data yang dirancang untuk menangani data dengan struktur yang lebih fleksibel serta kebutuhan skalabilitas tertentu. NoSQL tidak selalu menggunakan tabel relasional. Beberapa model NoSQL yang umum adalah: Document Database, Key-Value Database, Column-Family Database, Graph Database,

Contoh struktur document database:

```
JSON

{
  "nim": "230001",
  "nama": "Andi Pratama",
  "program_studi": "Teknik Informatika",
  "mata_kuliah": [
    "Basis Data",
    "Pemrograman Web"
  ]
}
```

Data seperti tersebut dapat disimpan sebagai satu dokumen tanpa harus memecahnya menjadi beberapa tabel seperti pada database relasional. Contoh teknologi: MongoDB; Redis; Apache Cassandra; Neo4j.

Kelebihan: struktur data fleksibel; cocok untuk data semi-terstruktur; dapat menangani volume data besar; mendukung kebutuhan tertentu dengan skalabilitas horizontal.

Kekurangan: tidak semua NoSQL memiliki model transaksi dan relasi sekuat database relasional; desain dan pengelolaannya membutuhkan pendekatan yang berbeda; tidak selalu cocok untuk sistem yang membutuhkan relasi kompleks dan integritas transaksional tinggi.

Contoh penggunaan: media sosial; aplikasi real-time; katalog produk; Internet of Things; sistem analitik berskala besar.

C. *Object Database*

Object Database atau *Object-Oriented Database* merupakan basis data yang menyimpan informasi

dalam bentuk objek sebagaimana konsep pemrograman berorientasi objek. Objek dapat memiliki: atribut; method; identitas objek; hubungan dengan objek lainnya. Konsep ini cocok ketika aplikasi dibangun dengan paradigma object-oriented dan data yang dikelola memiliki struktur objek yang kompleks.

Contoh konseptual:

```
Object: Mahasiswa

Atribut:
- NIM
- Nama
- Alamat
- ProgramStudi

Method:
- registrasi()
- ambilKRS()
- lihatNilai()
```

Berbeda dengan pendekatan relasional yang memetakan objek menjadi tabel, object database dapat menyimpan objek secara lebih langsung dalam bentuk objek.

Kelebihan: mendukung struktur objek kompleks; sesuai dengan paradigma object-oriented; dapat mengurangi kebutuhan pemetaan objek ke tabel pada kondisi tertentu; cocok untuk aplikasi dengan objek kompleks.

Kekurangan: penggunaannya tidak sepopuler database relasional; ekosistem dan keterampilan yang tersedia lebih terbatas; integrasi dengan

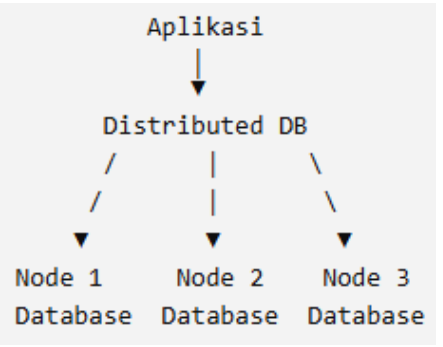
sistem berbasis SQL dapat membutuhkan pendekatan khusus.

Contoh penggunaan**:** *Computer-Aided Design* (CAD); sistem simulasi; sistem multimedia; aplikasi ilmiah; sistem dengan struktur objek kompleks.

D. *Distributed Database*

Distributed Database adalah basis data yang data atau komponennya disimpan pada lebih dari satu lokasi atau node dalam suatu jaringan, tetapi tetap dipandang sebagai satu kesatuan sistem oleh pengguna atau aplikasi.

Secara konseptual:



Sebagai contoh, organisasi yang memiliki kantor di Malang, Surabaya, dan Jakarta dapat memiliki node database pada masing-masing lokasi.

Distributed database dapat menggunakan pendekatan seperti: replication; partitioning; sharding; distributed transaction.

Kelebihan : meningkatkan ketersediaan; dapat mendekatkan data dengan lokasi pengguna; mendukung skalabilitas; mengurangi ketergantungan terhadap satu lokasi database.

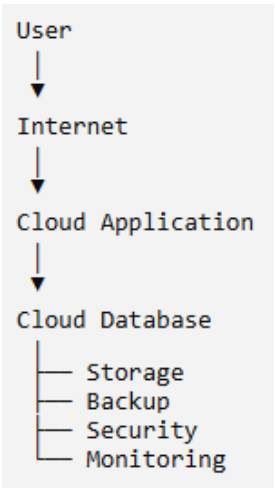
Kekurangan: arsitektur lebih kompleks; sinkronisasi data menjadi tantangan; pengelolaan konsistensi lebih sulit; keamanan jaringan harus diperhatikan secara serius.

Contoh penggunaan : perusahaan multinasional; jaringan perbankan; sistem e-commerce berskala besar; platform digital global; sistem yang memiliki banyak lokasi operasional.

E. *Cloud Database*

Cloud Database adalah basis data yang disediakan dan dijalankan pada lingkungan komputasi awan sehingga sumber daya database dapat diakses melalui jaringan dan dikelola secara elastis sesuai kebutuhan. Dalam model cloud, organisasi tidak selalu harus menyediakan dan mengelola seluruh infrastruktur database secara fisik sendiri.

Arsitektur sederhananya:



Cloud database dapat digunakan dalam berbagai model layanan, baik database yang dikelola sendiri pada infrastruktur cloud maupun layanan database terkelola.

Kelebihan : mudah dikembangkan; fleksibel; dapat diakses dari berbagai lokasi; mendukung elastisitas sumber daya; dapat menyediakan backup dan monitoring terintegrasi; mengurangi kebutuhan investasi hardware awal.

Kekurangan: bergantung pada koneksi jaringan; biaya dapat meningkat sesuai penggunaan; terdapat aspek keamanan dan privasi yang harus dikelola; organisasi perlu memahami ketergantungan terhadap penyedia cloud.

Contoh penggunaan: aplikasi web; aplikasi mobile; startup digital; e-commerce; sistem informasi organisasi; aplikasi SaaS.

F. Perbandingan Jenis Basis Data

Jenis	Struktur Data	Kekuatan Utama	Cocok Untuk
Relasional	Tabel	Integritas dan transaksi	Sistem informasi transaksional
NoSQL	Dokumen, key-value, graph, column	Fleksibilitas dan skalabilitas tertentu	Data besar dan semi-terstruktur
Object Database	Objek	Struktur objek kompleks	CAD, multimedia, simulasi

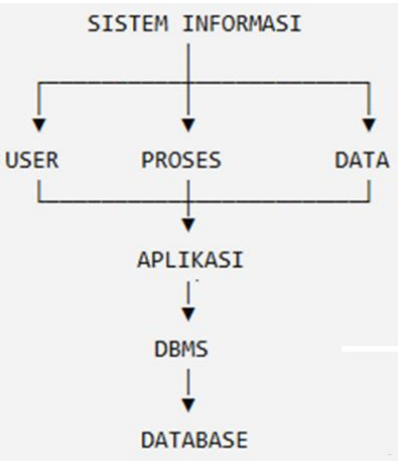
Jenis	Struktur Data	Kekuatan Utama	Cocok Untuk
Distributed Database	Multi-node	Ketersediaan dan distribusi	Sistem multi-lokasi
Cloud Database	Berbasis cloud	Elastisitas dan kemudahan pengelolaan	Aplikasi modern dan scalable

Pemilihan jenis database tidak seharusnya hanya didasarkan pada popularitas teknologi, tetapi harus mempertimbangkan karakteristik sistem yang akan dibangun. Sistem informasi akademik, misalnya, umumnya memiliki banyak transaksi dan hubungan data yang kuat antara mahasiswa, mata kuliah, KRS, dosen, dan nilai. Karena itu, database relasional sering menjadi pilihan yang sesuai. Sebaliknya, aplikasi dengan data yang sangat fleksibel atau kebutuhan distribusi dan skalabilitas tertentu dapat mempertimbangkan pendekatan NoSQL atau distributed database.

3.12.7 Hubungan Basis Data dengan Sistem Informasi

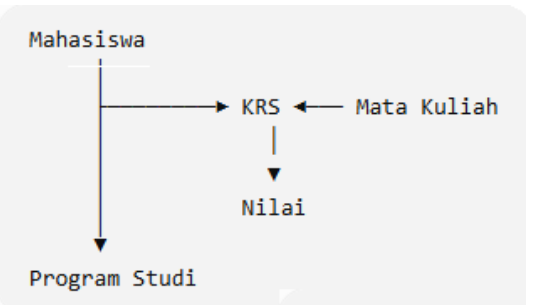
Basis data merupakan bagian dari ekosistem sistem informasi. Sistem informasi dapat dipandang sebagai kombinasi antara manusia, proses, teknologi, dan data.

Hubungan tersebut dapat digambarkan sebagai berikut:



Dalam arsitektur tersebut, database menjadi fondasi penyimpanan data, sedangkan aplikasi bertindak sebagai media untuk menjalankan proses bisnis dan menyediakan layanan kepada pengguna. Oleh sebab itu, kesalahan pada perancangan basis data dapat berdampak langsung terhadap: kualitas informasi; performa aplikasi; keamanan; konsistensi data; kemudahan pemeliharaan; pengembangan sistem di masa mendatang.

Contoh Penerapan pada Sistem Informasi Akademik : Untuk memahami konsep basis data secara lebih konkret, dapat digunakan contoh Sistem Informasi Akademik. Sistem tersebut membutuhkan beberapa data utama:



Contoh entitas:

Entitas	Contoh Data
Mahasiswa	NIM, nama, alamat
Dosen	NIDN, nama, bidang keahlian
Program Studi	kode, nama program studi
Mata Kuliah	kode, nama, SKS
KRS	mahasiswa, mata kuliah, semester
Nilai	mahasiswa, mata kuliah, nilai

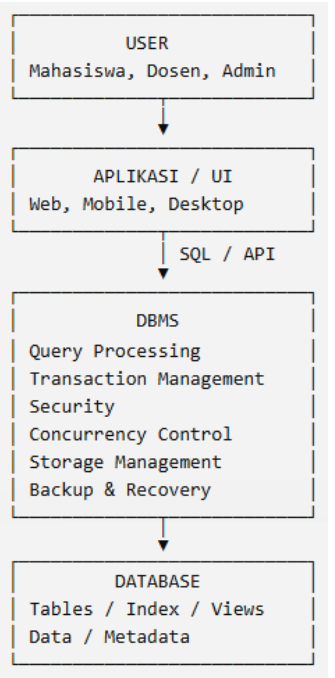
Dari contoh tersebut dapat dilihat bahwa basis data bukan sekadar tempat menyimpan data, tetapi juga menyediakan struktur hubungan antar-data.

Misalnya : Mahasiswa mengambil Mata Kuliah melalui KRS dan memperoleh Nilai. Hubungan tersebut nantinya dapat diterjemahkan ke dalam *Entity Relationship Diagram* (ERD), kemudian dinormalisasi dan diubah menjadi struktur tabel relasional.

3.13 *Database Management System* (DBMS)

DBMS merupakan salah satu komponen utama dalam sistem informasi yang berfungsi sebagai penghubung antara pengguna atau aplikasi dengan basis data. Jika pada sub-BAB sebelumnya basis data dipahami sebagai kumpulan data yang terorganisasi, maka DBMS dapat dipahami sebagai perangkat lunak yang mengatur bagaimana BD dibuat, disimpan, diakses, diperbarui, diamankan, dan dipelihara. Dalam sistem informasi modern, DBMS memiliki posisi strategis karena aplikasi umumnya tidak berinteraksi langsung dengan media penyimpanan fisik.

Aplikasi mengirimkan permintaan kepada DBMS, kemudian DBMS menerjemahkan permintaan tersebut menjadi operasi terhadap data. Dengan pendekatan ini, pengelolaan data dapat dilakukan secara terkontrol, konsisten, aman, dan efisien. Literatur database modern juga menempatkan query processing, storage, concurrency, security, dan recovery sebagai bagian penting dari arsitektur DBMS. Secara sederhana, hubungan tersebut dapat digambarkan sebagai berikut:



Gambar 3. 3 *Database Management System*

3.13.1 Pengertian DBMS

Database Management System (DBMS) adalah perangkat lunak yang menyediakan fasilitas untuk mendefinisikan, membuat, menyimpan, mengelola, mengakses, mengubah, dan mengamankan basis data.

Secara konseptual, DBMS berperan sebagai perantara (*intermediary*) antara aplikasi/pengguna dengan database. Pengguna tidak perlu mengetahui bagaimana data secara fisik ditempatkan pada hard disk atau SSD. Pengguna cukup memberikan perintah melalui bahasa atau antarmuka yang disediakan DBMS, kemudian DBMS menangani proses tersebut. Sebagai contoh, pengguna ingin memperoleh daftar mahasiswa TI. Aplikasi dapat mengirimkan query:

```sql
SELECT *
FROM mahasiswa
WHERE program_studi = 'Teknik Informatika';
```

DBMS kemudian:

- menerima query;
- memeriksa sintaks;
- menentukan cara terbaik menjalankan query;
- mencari data yang diperlukan;
- membaca data dari storage;
- mengembalikan hasil kepada aplikasi.

3.13.2 Fungsi DBMS

DBMS memiliki banyak fungsi yang saling berhubungan. Fungsi-fungsi tersebut tidak hanya berkaitan dengan penyimpanan data, tetapi juga mencakup keamanan, transaksi, integritas, concurrency, dan pemulihan data.

1. **Database *Definition***

DBMS menyediakan fasilitas untuk mendefinisikan struktur database. Misalnya:

```SQL
CREATE TABLE mahasiswa (
    nim VARCHAR(10) PRIMARY KEY,
    nama VARCHAR(100),
    alamat VARCHAR(200)
);
```

Perintah DDL antara lain: CREATE, ALTER, DROP, TRUNCATE

2. ***Data Storage Management***

   DBMS mengatur bagaimana data disimpan dan diambil dari media penyimpanan. Pengguna tidak perlu mengetahui secara langsung: lokasi fisik data; blok penyimpanan; struktur file; mekanisme buffering; indeks; mekanisme I/O. DBMS mengelola kompleksitas tersebut sehingga aplikasi dapat bekerja melalui abstraksi database.

3. ***Data Manipulation***

   DBMS memungkinkan pengguna melakukan operasi terhadap data, yaitu: Insert: menambahkan data; Select: membaca data; Update: mengubah data; Delete: menghapus data.

4. ***Query Processing***

   DBMS bertanggung jawab memproses permintaan query dari pengguna atau aplikasi. Misalnya:

```sql
SELECT nama
FROM mahasiswa
WHERE angkatan = 2025;
```

5. *Transaction Management*

Transaksi adalah sekumpulan operasi database yang diperlakukan sebagai satu kesatuan. Contohnya transaksi pembayaran mahasiswa:

```
1. Catat pembayaran
2. Update status tagihan
3. Simpan bukti transaksi
4. Update saldo administrasi
```

Keempat operasi tersebut idealnya diperlakukan sebagai satu transaksi. Jika salah satu operasi gagal, DBMS dapat membatalkan perubahan melalui mekanisme **rollback**.

6. *Concurrency Control*

Dalam sistem informasi, banyak pengguna dapat mengakses database pada waktu yang sama. Contohnya:

```
08:00:01  Mahasiswa A → Mengakses KRS
08:00:02  Mahasiswa B → Mengakses KRS
08:00:02  Operator    → Memperbarui data mata kuliah
08:00:03  Dosen       → Melihat data mahasiswa
```

7. *Security Management*

DBMS menyediakan mekanisme untuk mengontrol siapa yang dapat mengakses data dan operasi apa yang boleh dilakukan. Contohnya :

```
Administrator
    |
    ├── CREATE
    ├── INSERT
    ├── UPDATE
    ├── DELETE
    └── SELECT

Dosen
    |
    ├── SELECT
    └── UPDATE nilai

Mahasiswa
    |
    └── SELECT data pribadi
```

8. *Integrity Management*

DBMS membantu memastikan data memenuhi aturan yang telah ditentukan. Contohnya:

```sql
CREATE TABLE mahasiswa (
    nim VARCHAR(10) PRIMARY KEY,
    nama VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE
);
```

Dalam contoh tersebut: PRIMARY KEY memastikan identitas mahasiswa unik; NOT NULL memastikan nama harus diisi; UNIQUE mencegah email yang sama digunakan lebih dari sekali.

9. *Backup* dan *Recovery*

DBMS menyediakan atau mendukung mekanisme untuk membuat salinan data dan memulihkan database ketika terjadi masalah. Risiko yang mungkin terjadi antara lain: hardware failure;

kesalahan manusia; kerusakan database; serangan malware; kegagalan sistem; kehilangan storage. Strategi backup dapat mencakup:Full Backup, Incremental Backup, Differential Backup, Transaction Log Backup. Tujuannya adalah memastikan database dapat dipulihkan dengan kehilangan data seminimal mungkin.

10. *Metadata Management*

DBMS tidak hanya menyimpan data pengguna, tetapi juga menyimpan informasi mengenai struktur database yang disebut metadata. Contohnya:

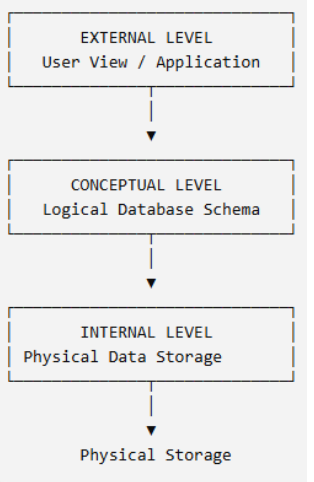
```
Database
│
├── Table: mahasiswa
│   ├── nim      VARCHAR(10)
│   ├── nama     VARCHAR(100)
│   └── alamat   VARCHAR(200)
│
└── Table: dosen
    ├── nidn     VARCHAR(20)
    └── nama     VARCHAR(100)
```

3.13.3 Arsitektur DBMS

Arsitektur DBMS dapat dibahas dari beberapa perspektif. Dalam pembelajaran basis data, salah satu pendekatan fundamental adalah arsitektur tiga level (*three-schema architecture*), sedangkan dalam implementasi sistem informasi modern juga dikenal arsitektur client-server dan multi-tier

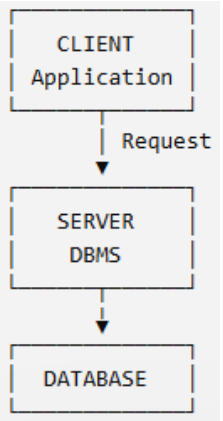
A. Arsitektur Tiga Level

Arsitektur tiga level memisahkan representasi database menjadi: External Level; Conceptual Level; Internal Level.



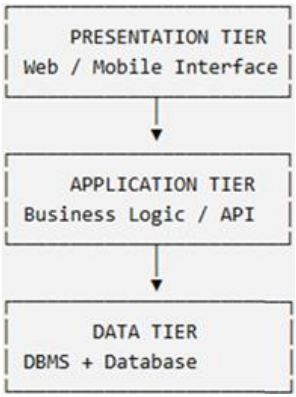
B. *Client-Server Architecture*

Pada arsitektur client-server, fungsi sistem dibagi antara sisi client dan server.



C. *Three-Tier Architecture*

Dalam sistem informasi berbasis web, arsitektur yang sering digunakan adalah ***three-tier architecture***.



3.12.6 Contoh DBMS

Berikut beberapa DBMS yang banyak digunakan dalam pembelajaran maupun pengembangan sistem informasi.

A. MySQL

MySQL merupakan DBMS relasional yang banyak digunakan untuk aplikasi web dan sistem informasi. Dokumentasi resmi MySQL menjelaskan MySQL Server sebagai database server SQL yang mendukung lingkungan multi-threaded dan multi-user serta berbagai kemampuan administrasi, keamanan, backup, optimasi, replication, dan storage engine.

B. PostgreSQL

PostgreSQL adalah *open-source object-relational database management system* (ORDBMS) yang mendukung sebagian besar standar SQL dan menyediakan berbagai fitur modern, termasuk *complex query, foreign key, trigger, view, transactional integrity,* serta *multiversion concurrency control.*

D. *Oracle Database*

Oracle Database merupakan DBMS enterprise yang digunakan untuk mengelola data dalam berbagai skenario organisasi berskala besar. Dokumentasi Oracle menyediakan pembahasan khusus mengenai arsitektur database, *database concepts, database reference, concurrency, consistency,* PL/SQL, serta berbagai komponen administrasi dan pengembangan

E. SQLite

SQLite memiliki karakteristik yang berbeda dari kebanyakan DBMS server. SQLite merupakan embedded, serverless, self-contained, zero-configuration transactional SQL database engine. SQLite tidak membutuhkan proses server terpisah; aplikasi dapat membaca dan menulis langsung ke file database. Satu database SQLite dapat disimpan dalam satu file yang berisi tabel, indeks, trigger, dan view. SQLite juga mendukung transaksi ACID dan memiliki ukuran footprint yang relatif kecil.

3.12.7 Perbandingan Contoh DBMS

DBMS	Model Utama	Karakteristik	Kekuatan	Contoh Penggunaan
MySQL	Relasional	Multi-user, SQL, populer untuk web	Sederhana, ekosistem luas	Web, akademik, e-commerce
Postgre SQL	Relasional / Object-Relational	Open source, extensible	Query kompleks, integritas, extensibility	Enterprise, GIS, analitik
SQL Server	Relasional	Enterprise RDBMS, T-SQL	Integrasi enterprise dan	ERP, BI, sistem perusahaan

DBMS	Model Utama	Karakteristik	Kekuatan	Contoh Penggunaan
			Microsoft ecosystem	
Oracle	Relasional / Enterprise	Enterprise-scale	Transaksi, availability, security	Perbankan, pemerintahan, enterprise
SQLite	Relasional Embedded	Serverless, single-file	Ringan dan mudah digunakan	Mobile, desktop, embedded

3.13 Perancangan Basis Data

Basis data (*database*) merupakan komponen fundamental dalam sebuah sistem informasi yang berfungsi sebagai tempat penyimpanan, pengelolaan, dan penyediaan data secara terstruktur. Seluruh aktivitas sistem, mulai dari proses input, pengolahan, hingga penyajian informasi kepada pengguna, sangat bergantung pada kualitas rancangan basis data. Oleh karena itu, perancangan basis data menjadi salah satu tahapan yang paling penting dalam siklus pengembangan sistem informasi.

Perancangan basis data tidak hanya bertujuan menyimpan data, tetapi juga memastikan bahwa data dapat diakses dengan cepat, memiliki tingkat integritas yang tinggi, menghindari redundansi, serta mendukung kebutuhan operasional maupun pengambilan keputusan organisasi. Menurut Elmasri dan Navathe (2017), desain basis data yang baik harus mampu merepresentasikan kebutuhan bisnis secara akurat, menjaga konsistensi data, serta memberikan fleksibilitas terhadap perubahan kebutuhan di masa mendatang.

Secara umum, proses perancangan basis data dimulai dengan analisis kebutuhan data, dilanjutkan dengan identifikasi entitas, atribut, dan hubungan antardata, kemudian menentukan berbagai jenis kunci (*key*) yang

berfungsi menjaga keunikan serta integritas data. Tahapan tersebut menjadi dasar dalam penyusunan *Entity Relationship Diagram* (ERD), normalisasi, hingga implementasi basis data menggunakan *Database Management System* (DBMS).

Pada subbab ini akan dibahas tahapan-tahapan dasar dalam perancangan basis data yang meliputi analisis kebutuhan data, identifikasi entitas, identifikasi atribut, serta konsep *Primary Key*, *Foreign Key*, *Candidate Key*, dan *Composite Key*.

3.13.1 Analisis Kebutuhan Data

1. Pengertian Analisis Kebutuhan Data

   Analisis kebutuhan data merupakan tahap awal dalam perancangan basis data yang bertujuan mengidentifikasi seluruh data yang dibutuhkan oleh organisasi agar proses bisnis dapat berjalan secara efektif. Pada tahap ini analis sistem mempelajari aktivitas organisasi, aliran informasi, dokumen yang digunakan, serta kebutuhan laporan yang dihasilkan.

   Analisis kebutuhan data menjadi landasan dalam menentukan struktur basis data sehingga seluruh informasi yang dibutuhkan dapat disimpan secara lengkap tanpa adanya data yang berlebihan maupun data yang terlewat.

   Menurut Connolly dan Begg (2015), analisis kebutuhan data merupakan proses memahami kebutuhan informasi organisasi sehingga dapat diterjemahkan menjadi struktur data yang terorganisasi dengan baik.

2. Tujuan Analisis Kebutuhan Data

   Analisis kebutuhan data bertujuan untuk:

   - mengidentifikasi seluruh data yang diperlukan organisasi;
   - memahami hubungan antar data;
   - menentukan data yang akan disimpan dalam basis data;
   - mengurangi redundansi data;
   - mendukung penyusunan model konseptual basis data.

3. Sumber Analisis Data

   Informasi mengenai kebutuhan data dapat diperoleh melalui beberapa metode berikut.

   - Wawancara : Dilakukan kepada pengguna sistem untuk mengetahui data apa saja yang digunakan dalam aktivitas sehari-hari.
   - Observasi: Mengamati proses bisnis secara langsung untuk mengetahui data yang benar-benar digunakan.
   - Studi Dokumen : Menganalisis formulir, laporan, SOP, buku induk, maupun arsip organisasi.
   - Kuesioner : Digunakan apabila jumlah pengguna relatif banyak.

4. Contoh Analisis Kebutuhan Data

   Misalnya akan dibangun **Sistem Informasi Akademik**. Hasil analisis menunjukkan bahwa sistem membutuhkan data berikut.

Data	Keterangan
Mahasiswa	Identitas mahasiswa
Dosen	Data tenaga pengajar
Mata Kuliah	Informasi mata kuliah
KRS	Pengambilan mata kuliah
Jadwal	Jadwal perkuliahan
Nilai	Hasil belajar mahasiswa

Dari hasil tersebut dapat diketahui bahwa keenam kelompok data tersebut nantinya akan menjadi kandidat entitas dalam basis data.

3.13.2 Identifikasi Entitas

Entitas (*Entity*) adalah objek nyata maupun abstrak yang informasinya perlu disimpan dalam basis data. Entitas dapat berupa orang, tempat, benda, kejadian, maupun konsep yang memiliki keberadaan tersendiri.

Dalam *Entity Relationship Diagram* (ERD), entitas digambarkan menggunakan simbol persegi panjang (*rectangle*).

1. Karakteristik Entitas

   Suatu objek dapat dikatakan sebagai entitas apabila:

   - ✓ memiliki identitas yang jelas;
   - ✓ mempunyai sejumlah atribut;
   - ✓ datanya perlu disimpan;
   - ✓ terlibat dalam proses bisnis organisasi.

2. Langkah Identifikasi Entitas

   - ✓ Identifikasi entitas dilakukan melalui tahapan berikut.
   - ✓ Mengidentifikasi objek yang terlibat dalam proses bisnis.
   - ✓ Mengelompokkan objek berdasarkan jenisnya.
   - ✓ Menghilangkan objek yang bersifat sementara.
   - ✓ Menentukan entitas utama dan entitas pendukung.

3. Contoh Entitas

   Pada Sistem Informasi Akademik diperoleh beberapa entitas berikut.

No	Entitas
1	Mahasiswa
2	Dosen
3	Program Studi
4	Mata Kuliah
5	KRS
6	Nilai
7	Ruang
8	Jadwal

Entitas-entitas tersebut nantinya akan saling berhubungan membentuk struktur basis data.

3.13.3 Identifikasi Atribut

Atribut merupakan karakteristik atau informasi yang dimiliki oleh setiap entitas. Atribut menjelaskan detail mengenai suatu objek sehingga dapat dibedakan dengan objek lainnya.

Sebagai contoh, entitas **Mahasiswa** memiliki atribut berupa NIM, nama, alamat, tanggal lahir, nomor telepon, dan program studi.

1. Jenis-Jenis Atribut

   - ✓ Atribut Sederhana (*Simple Attribute*) : Tidak dapat dipecah lagi. Contoh: Nama, Jenis Kelamin, Golongan Darah
   - ✓ Atribut Komposit (*Composite Attribute*) : Masih dapat dipecah menjadi atribut yang lebih kecil. Contoh: Alamat, Contoh: Tanggal Lahir
   - ✓ Atribut Bernilai Bandapat dipecah menjadi: Jalan, Kota, Kecamatan, Kode Pos
   - ✓ Atribut Bernilai Tunggal (*Single-Valued*) : Hanya memiliki satu nilai. yak (*Multi-Valued*) : Memiliki lebih dari satu nilai. Contoh: Nomor Telepon
   - ✓ Atribut Turunan (*Derived Attribute*) : Nilainya diperoleh dari atribut lain. Contoh: Umur dihitung dari Tanggal Lahir.

2. Contoh Atribut Entitas Mahasiswa

Atribut	Keterangan
NIM	Nomor induk mahasiswa
Nama	Nama mahasiswa
Jenis Kelamin	L/P

Atribut	Keterangan
Tanggal Lahir	Tanggal lahir
Alamat	Domisili mahasiswa
Email	Alamat email
No HP	Nomor telepon

3.13.4 Primary Key

Primary Key adalah atribut atau sekumpulan atribut yang digunakan untuk mengidentifikasi setiap record secara unik di dalam sebuah tabel. Nilai *primary key* tidak boleh kosong (*NULL*) dan tidak boleh memiliki nilai yang sama.

Primary key berfungsi sebagai identitas utama suatu data. Primary key harus memenuhi syarat/ciri berikut. unik; tidak bernilai NULL; stabil; tidak berubah selama data digunakan. Primary key digunakan untuk: membedakan setiap record; mencegah data ganda; membangun relasi antar tabel; menjaga integritas data.

Contoh : Tabel Mahasiswa**,** **NIM** merupakan Primary Key

3.13.5 Foreign Key

Foreign Key adalah atribut pada suatu tabel yang mengacu pada *Primary Key* di tabel lain. Foreign key digunakan untuk membangun hubungan (*relationship*) antar tabel sehingga data tetap konsisten.

Foreign key berfungsi untuk: membentuk relasi antar tabel; menjaga integritas referensial (*referential integrity*); menghindari data yang tidak valid. Contoh : pada tabel Mahasiswa**,** atribut **IdProdi** merupakan Foreign key yang berasala dari tabel Program Studi.

3.13.6 *Candidate Key*

Candidate Key adalah satu atau lebih atribut yang memiliki kemampuan untuk mengidentifikasi setiap record secara unik dan berpotensi dijadikan *Primary Key*. Dari beberapa candidate key yang tersedia, hanya satu yang dipilih sebagai primary key, sedangkan yang lain disebut *Alternate Key*. Candidate key harus: unik; tidak bernilai NULL; mampu mengidentifikasi data secara tunggal. Contoh pada tabel Mahasiswa terdapat atribut NIM, NIK, E-mai. **NIM → Primary Key,** Sedangkan NIK dan Email menjadi **Alternate Key**.

3.13.7 *Composite Key*

Composite Key adalah gabungan dua atau lebih atribut yang secara bersama-sama digunakan sebagai identitas unik suatu record. Tidak ada satu atribut pun yang dapat mengidentifikasi record secara mandiri sehingga kombinasi beberapa atribut diperlukan. Kapan Composite Key Digunakan?, apabila: tidak tersedia atribut tunggal yang unik; tabel merupakan hasil relasi many-to-many; diperlukan kombinasi beberapa atribut sebagai identitas. Contoh pada Tabel KRS Primary key dapat berupa kombinasi: NIM, KodeMK, Semester.

Studi Kasus: Identifikasi Entitas, Atribut, dan Key pada SIAKAD

Misalkan akan dibangun basis data untuk Sistem Informasi Akademik. Berdasarkan analisis kebutuhan diperoleh tiga entitas utama, yaitu **Mahasiswa**, **Mata Kuliah**, dan **KRS**.

Entitas Mahasiswa

- Primary Key: NIM
- Candidate Key: NIM, NIK, Email

- Atribut: Nama, Tanggal Lahir, JK, Alamat, Email, No. HP

Entitas Mata Kuliah

- Primary Key: KodeMK
- Atribut: Nama Mata Kuliah, SKS, Semester

Entitas KRS

- Primary Key: ID_KRS (atau Composite Key: NIM + KodeMK + Semester)
- Foreign Key: NIM, KodeMK
- Atribut: Semester, Tahun Akademik, Tanggal Pengambilan

Dengan struktur tersebut, data mahasiswa dapat dihubungkan dengan mata kuliah yang diambil melalui tabel KRS. Penggunaan *Primary Key* menjamin setiap data bersifat unik, sedangkan *Foreign Key* menjaga hubungan antarentitas agar tetap konsisten.

3.14 *Entity Relationship Diagram* (ERD)

3.14.1 Konsep EDR

ERD merupakan model konseptual yang digunakan untuk menggambarkan struktur data beserta hubungan antar data di dalam suatu sistem informasi. ERD diperkenalkan oleh Peter Chen (1976) sebagai metode pemodelan data yang mampu menjembatani kebutuhan pengguna dengan implementasi basis data relasional.

Pada tahap analisis dan perancangan sistem informasi, ERD berfungsi sebagai representasi visual mengenai objek-objek yang akan disimpan di dalam basis data, atribut yang dimiliki setiap objek, serta hubungan antar objek tersebut. Dengan adanya ERD, pengembang dapat memahami

struktur data sebelum membuat tabel pada DBMS seperti MySQL, PostgreSQL, SQL Server, maupun Oracle.

Tujuan utama ERD adalah:

- ✓ Mengidentifikasi seluruh data yang diperlukan sistem.
- ✓ Menghindari redundansi data.
- ✓ Mempermudah proses normalisasi database.
- ✓ Menjadi acuan dalam pembuatan tabel database.
- ✓ Memudahkan komunikasi antara analis sistem, programmer, dan pengguna.

Keuntungan penggunaan ERD antara lain:

- ✓ Struktur database menjadi lebih terorganisasi.
- ✓ Hubungan antar data lebih mudah dipahami.
- ✓ Mengurangi kesalahan dalam implementasi database.
- ✓ Mempermudah pemeliharaan sistem.
- ✓ Menjadi dokumentasi resmi sistem informasi.

3.14.2 Komponen ERD

Tersusun atas tiga komponen utama, yaitu: Entity, Attribute, Relationship.

- ✓ ***Entity*** adalah objek nyata maupun abstrak yang informasinya perlu disimpan di dalam basis data, dapat berupa: Orang, Tempat, Barang, Organisasi, Peristiwa Dalam diagram ERD, entity digambarkan menggunakan **persegi panjang** (*Rectangle*). Contoh Entity pada Sistem Informasi Akademik
- ✓ ***Attribute*** merupakan karakteristik yang dimiliki oleh suatu entity. Misalnya entity Mahasiswa

memiliki atribut: NIM, Nama, Jenis Kelamin, Alamat, Tanggal Lahir, Email. Dalam ERD atribut digambarkan menggunakan **oval** (*ellipse*).

- ✓ ***Relationship*** : Relationship merupakan hubungan antar entity. Relationship digambarkan menggunakan **belah ketupat** (Diamond). Contoh: Mahasiswa Mengambil Mata Kuliah

3.14.3 Kardinalitas (*Cardinality*)

Kardinalitas menunjukkan banyaknya hubungan antar entity. Ada tiga jenis kardinalitas.

- ✓ One to One (**1 : 1**) : Satu data pada entity pertama hanya berhubungan dengan satu data pada entity kedua. Contoh: satu mahasiswa memiliki satu kartu mahasiswa, satu kartu hanya dimiliki satu mahasiswa, contoh lainnya Penduduk ↔ KTP, Pegawai ↔ NPWP
- ✓ One to Many (**1 : N**) : Satu data pada entity pertama dapat memiliki banyak data pada entity kedua. Contoh: 1 Dosen dengan N mahasisa, Artinya: satu dosen wali membimbing banyak mahasiswa. Namun satu mahasiswa hanya memiliki satu dosen wali. Contoh lain 1 Prodi dengan N mahasiswa
- ✓ Many to Many (**M : N**) : Satu data pada entity pertama berhubungan dengan banyak data pada entity kedua. Sebaliknya juga demikian. Contoh: Mahasiswa M -- N Mata Kuliah, artinya: Seorang mahasiswa mengambil banyak mata kuliah. Satu mata kuliah juga diambil banyak mahasiswa. Hubungan M:N harus dipecah menjadi entity baru.
- ✓ Partisipasi (*Participation*): Partisipasi menunjukkan apakah keberadaan suatu entity bergantung pada relationship. Ada dua jenis. a. Total

Participation : Semua entity harus ikut dalam relationship. Contoh: Mahasiswa Mengambil Mata Kuliah, Artinya seluruh mahasiswa wajib mengambil mata kuliah. b. Partial Participation: Tidak semua entity harus mengikuti relationship. Contoh: Dosen Menjadi Kaprodi, Tidak semua dosen menjadi Kaprodi.

3.14.4 Contoh Perancangan ERD

Studi Kasus: Sistem Informasi Akademik

Deskripsi Kasus : Sebuah perguruan tinggi memiliki sistem akademik dengan aturan sebagai berikut:

1. Setiap mahasiswa terdaftar pada satu program studi.
2. Setiap dosen dapat mengajar banyak mata kuliah.
3. Setiap mata kuliah diajar oleh satu dosen.
4. Mahasiswa dapat mengambil banyak mata kuliah.
5. Setiap pengambilan mata kuliah dicatat dalam KRS.
6. Nilai diberikan pada setiap mata kuliah yang diambil mahasiswa.

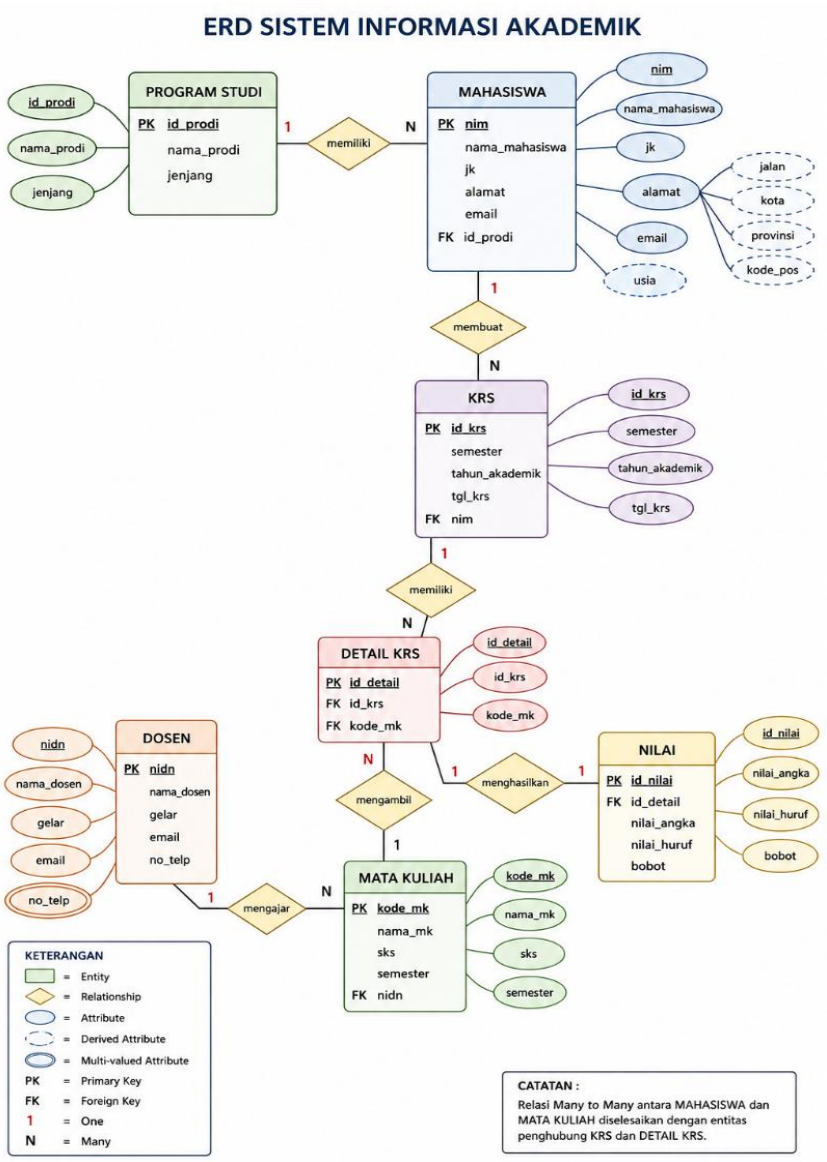
Identifikasi Entity

Tabel/Entity	Primary Key
Program Studi	id_prodi
Mahasiswa	nim
Dosen	nidn
Mata Kuliah	kode_mk
KRS	id_krs
Detail KRS	id_detail

Tabel/Entity	Primary Key
Nilai	id_nilai

Hubungan Antar Entity

Tabel 1	Kardinalitas	Tabel 2
Program Studi	1 : N	Mahasiswa
Dosen	1 : N	Mata Kuliah
Mahasiswa	1 : N	KRS
KRS	1 : N	Detail KRS
Mata Kuliah	1 : N	Detail KRS
Detail KRS	1 : 1	Nilai



Gambar 3. 4 *Entity Relationship Diagram*

Penjelasan ERD

- Program Studi → Mahasiswa (1:N): Satu program studi memiliki banyak mahasiswa, sedangkan setiap mahasiswa hanya berada pada satu program studi.
- Mahasiswa → KRS (1:N): Setiap mahasiswa dapat membuat KRS pada beberapa semester.
- KRS → Detail KRS (1:N): Satu KRS memuat banyak mata kuliah yang diambil.
- Dosen → Mata Kuliah (1:N): Satu dosen dapat mengampu beberapa mata kuliah.
- Mata Kuliah → Detail KRS (1:N): Satu mata kuliah dapat diambil oleh banyak mahasiswa melalui entitas Detail KRS.
- Detail KRS → Nilai (1:1): Setiap mata kuliah yang diambil mahasiswa menghasilkan satu data nilai.

ERD tersebut menunjukkan bagaimana hubungan many-to-many antara Mahasiswa dan Mata Kuliah diselesaikan menggunakan entitas penghubung KRS dan Detail KRS, sehingga struktur basis data menjadi terorganisasi, mudah dinormalisasi, serta siap diimplementasikan ke dalam tabel-tabel relasional pada DBMS. Diagram ini juga mencerminkan praktik perancangan basis data yang umum digunakan dalam pengembangan Sistem Informasi Akademik di perguruan tinggi.

3.15 Normalisasi Basis Data

Normalisasi basis data (*Database Normalization*) adalah proses sistematis dalam merancang struktur basis data dengan cara mengorganisasikan data ke dalam beberapa tabel yang saling berhubungan sehingga mampu mengurangi redundansi (duplikasi data), menghilangkan ketergantungan yang tidak diperlukan, serta menjaga konsistensi dan integritas data.

Konsep normalisasi pertama kali diperkenalkan oleh **Edgar F. Codd (1970)** sebagai bagian dari model basis data relasional. Melalui proses normalisasi, data yang semula tersimpan dalam satu tabel besar akan dipecah menjadi beberapa tabel yang lebih kecil berdasarkan hubungan logis antar data. Setiap tabel dirancang agar hanya menyimpan informasi yang berkaitan dengan satu entitas tertentu.

Normalisasi merupakan salah satu tahapan penting dalam perancangan basis data setelah penyusunan ***Entity Relationship Diagram*** **(ERD)**. Hasil normalisasi akan menjadi dasar dalam pembentukan tabel-tabel relasional pada DBMS seperti MySQL, PostgreSQL, SQL Server, maupun Oracle.

Sebagai ilustrasi, sebuah tabel yang berisi data mahasiswa, dosen, mata kuliah, dan nilai dalam satu tabel besar akan mengakibatkan banyak pengulangan data dosen maupun mata kuliah. Dengan normalisasi, data tersebut dipisahkan ke dalam tabel **Mahasiswa**, **Dosen**, **Mata Kuliah**, **KRS**, dan **Nilai** sehingga penyimpanan menjadi lebih efisien.

Normalisasi dilakukan untuk menghasilkan struktur basis data yang efisien, fleksibel, dan mudah dikelola. Tujuan utama normalisasi meliputi:

1. Mengurangi Redundansi Data : Normalisasi menghindari penyimpanan data yang sama secara berulang sehingga penggunaan media penyimpanan menjadi lebih efisien.

Contoh: Nama dosen tidak perlu ditulis berulang pada setiap data mahasiswa yang mengambil mata kuliah tersebut.

2. Menghindari Anomali Data : Normalisasi mampu mengurangi tiga jenis anomali, yaitu:
   - ✓ *Insert Anomaly* **:** Kesulitan menambahkan data baru karena harus mengisi data lain yang belum tersedia. Contoh: Tidak dapat menambahkan mata kuliah baru apabila belum ada mahasiswa yang mengambil mata kuliah tersebut.
   - ✓ *Update Anomaly*: Perubahan satu data harus dilakukan pada banyak baris. Contoh: Jika nama dosen berubah, seluruh data mahasiswa yang mengambil mata kuliah tersebut harus diperbarui.
   - ✓ *Delete Anomaly*: Penghapusan satu data menyebabkan hilangnya informasi lain. Contoh: Jika seluruh mahasiswa pada suatu mata kuliah dihapus, maka informasi mata kuliah juga ikut hilang.
3. Menjaga Integritas Data : Data yang tersimpan tetap konsisten dan tidak saling bertentangan.
4. Mempermudah Pemeliharaan Database: Struktur tabel menjadi lebih sederhana sehingga proses pengembangan dan pemeliharaan sistem lebih mudah dilakukan.
5. Meningkatkan Efisiensi Penyimpanan: Ukuran database menjadi lebih kecil karena tidak terjadi pengulangan data yang tidak diperlukan.
6. Mempermudah Pengembangan Sistem: Jika suatu saat diperlukan penambahan fitur baru, struktur basis data lebih mudah dikembangkan.

3.15.1 *First Normal Form* (1NF)

Pengertian : Suatu tabel dikatakan memenuhi First Normal Form (1NF) apabila: Setiap atribut memiliki nilai atomik (tidak dapat dibagi lagi), Tidak terdapat atribut yang memiliki banyak nilai (multivalued attribute), Setiap baris bersifat unik.

Contoh Tabel Belum Normal

NIM	Nama	Mata Kuliah
22001	Andi	Pemrograman , Basis Data, Algoritma
22002	Budi	Pemrograman
22003	Citra	Basis Data, Jaringan

Kolom Mata Kuliah masih berisi lebih dari satu nilai.

Setelah 1NF

NIM	Nama	Mata Kuliah
22001	Budi	Pemrograman
22001	Andi	Basis Data
22001	Andi	Algoritma
22002	Budi	Pemrograman
22003	Citra	Basis Data
22003	Citra	Jaringan

3.15.2 *Second Normal Form* (2NF)

Suatu tabel berada pada *Second Normal Form* (2NF) apabila: Telah memenuhi 1NF. Seluruh atribut non-primary key bergantung sepenuhnya pada primary key. Tidak terdapat Partial Dependency.

Contoh: Misalkan tabel KRS: {NIM, KodeMK, NamaMahasiswa, NamaMK, sks, Nilai}. Primary Key: (NIM, KodeMK), Terjadi ketergantungan parsial karena: NamaMahasiswa hanya bergantung pada NIM. NamaMK dan sks hanya bergantung pada KodeMK. Maka hasil Normalisasi 2NF dipecah menjadi **tiga** tabel: Tabel Mahasiswa {NIM, NamaMahasiswa}, Tabel Mata Kuliah{ KodeMK, NamaMK, sks }, dan Tabel KRS {NIM, KodeMK, Nilai}.

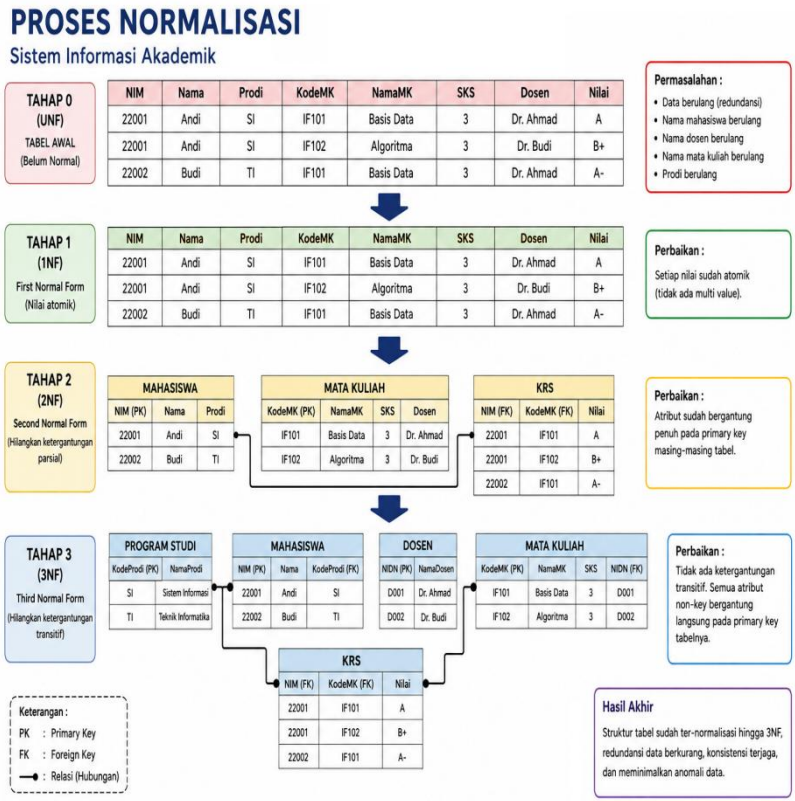
10.15.3 *Third Normal Form* (3NF)

Suatu tabel memenuhi *Third Normal Form* (3NF) apabila: Telah memenuhi 2NF. Tidak memiliki Transitive Dependency. Transitive Dependency adalah kondisi ketika atribut non-key bergantung pada atribut non-key lainnya.

Contoh: Misalkan tabel Mahasiswa: {NIM, NamaMahasiswa, KodeProdi, Namaprodi}. Ketergantungan: NIM → KodeProdi, KodeProdi → NamaProdi, Artinya: NamaProdi tidak bergantung langsung pada NIM. Hasil Normalisasi 3NF dipecah menjadi **dua** tabel**:** Tabel Mahasiswa { NIM, NamaMahasiswa, KodeProdi}, tabel Prodi { KodeProdi, Namaprodi}

Proses normalisasi menunjukkan transformasi bertahap dari sebuah tabel besar yang mengandung banyak redundansi menjadi sekumpulan tabel yang saling berelasi dengan baik. Pada **1NF**, data diubah menjadi bernilai atomik; pada **2NF**, ketergantungan parsial dihilangkan dengan memisahkan entitas yang berbeda; pada **3NF**, ketergantungan transitif dihapus sehingga setiap atribut non-kunci hanya bergantung pada primary key. Apabila masih terdapat determinan yang bukan candidate key, penyempurnaan dilakukan

melalui **BCNF**. Hasil akhir adalah struktur basis data yang lebih efisien, konsisten, mudah dipelihara, dan siap diimplementasikan pada sistem basis data relasional.



Gambar 3. 5 Proses Normalisasi Data

3.16 Transformasi ERD ke Model Relasional

Transformasi ERD ke Model Relasional adalah proses mengubah model konseptual hasil analisis kebutuhan menjadi struktur tabel yang dapat diimplementasikan pada sistem manajemen basis data relasional (*Relational Database Management System*/RDBMS) seperti MySQL, PostgreSQL, Oracle Database, SQL Server, maupun MariaDB.

ERD digunakan untuk menggambarkan objek (*entity*), atribut (*attribute*), dan hubungan (*relationship*) antarobjek secara konseptual. Namun, DBMS tidak menyimpan data dalam bentuk diagram, melainkan dalam bentuk tabel-tabel relasional. Oleh karena itu, setiap komponen pada ERD harus dipetakan (*mapping*) ke dalam tabel sesuai aturan transformasi yang baku.

Transformasi ERD merupakan tahap yang sangat penting dalam proses pengembangan sistem informasi karena menjadi jembatan antara hasil analisis kebutuhan pengguna dengan implementasi basis data. Kesalahan pada tahap ini dapat menyebabkan redundansi data, anomali, serta kesulitan dalam pengembangan sistem di masa mendatang.

Tujuan transformasi ERD ke model relasional adalah:

- ✓ Mengubah model konseptual menjadi model logis yang siap diimplementasikan.
- ✓ Menentukan struktur tabel beserta atributnya.
- ✓ Menentukan Primary Key (PK) dan Foreign Key (FK).
- ✓ Menjaga integritas referensial antar tabel.
- ✓ Menghasilkan basis data yang efisien, konsisten, dan mudah dikembangkan.

3.16.1 Aturan Transformasi ERD ke Model Relasional

Transformasi ERD ke model relasional dilakukan berdasarkan sejumlah aturan (*mapping rules*) yang telah menjadi standar dalam perancangan basis data relasional.

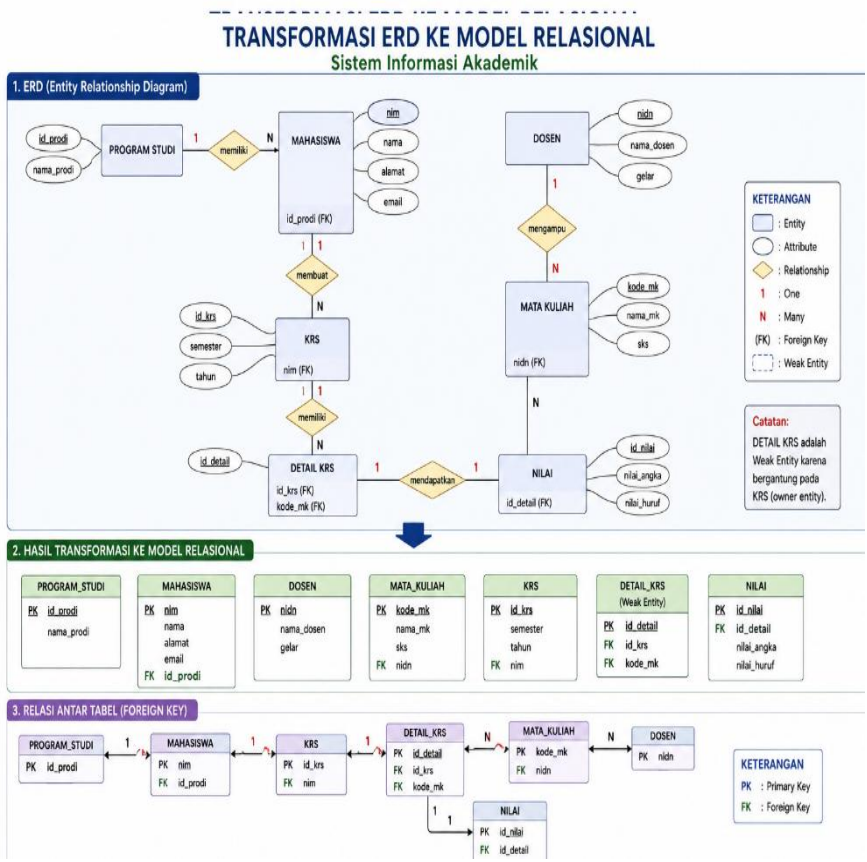
- Setiap Entity pada ERD menjadi Satu Tabel, misal Entity Mahasiswa→ tabel Mahasiswa.

- Attribute menjadi Kolom, Seluruh atribut pada entity akan menjadi kolom dalam tabel. Contoh Entity Dosen {NIDN, Nama, Gelar , Email }
- Primary Key tetap menjadi Primary Key, Atribut yang menjadi identitas unik pada ERD akan menjadi Primary Key pada tabel. Misal: NIM → Primary Key Mahasiswa, KodeMK → Primary Key Mata Kuliah.
- Relationship akan menjadi Foreign Key atau Tabel Baru, Bergantung pada jenis kardinalitas relationship.

Kardinalitas	Hasil Transformasi
1 : 1	Foreign Key
1 : N	Foreign Key pada sisi N
M : N	Dibuat tabel baru

3.16.2. Contoh Implementasi

Studi Kasus : ERD Sistem Informasi Akademik dengan Entity: Program Studi, Mahasiswa, Dosen, Mata Kuliah, KRS, Detail KRS, Nilai.



Gambar 3. 6 Implementasi ERD

Transformasi ERD ke model relasional merupakan tahapan penting dalam desain basis data yang menghubungkan model konseptual dengan implementasi fisik pada DBMS. Proses ini memastikan setiap entity menjadi tabel, setiap atribut menjadi kolom, serta setiap relationship diterjemahkan menjadi foreign key atau tabel penghubung sesuai kardinalitasnya.

Dengan menerapkan aturan transformasi yang benar, struktur basis data akan memiliki integritas

referensial yang baik, meminimalkan redundansi, mendukung proses normalisasi, dan siap digunakan sebagai dasar implementasi sistem informasi.

3.17 Perancangan Kamus Data (*Data Dictionary*)

3.17.1 Pengertian Kamus Data

Kamus Data (*Data Dictionary*) adalah dokumentasi terstruktur yang berisi informasi atau metadata mengenai data yang digunakan dalam suatu sistem informasi. Kamus data menjelaskan secara rinci nama data, arti data, tipe data, panjang data, sumber data, penggunaan data, serta aturan atau batasan yang berlaku terhadap data tersebut.

Kamus data tidak menyimpan data operasional seperti tabel database, tetapi menyimpan **informasi tentang data**. Dengan demikian, kamus data dapat dipahami sebagai "daftar informasi" yang menjelaskan struktur dan karakteristik data dalam suatu sistem.

Dalam proses analisis dan perancangan sistem informasi, kamus data memiliki hubungan yang erat dengan *Data Flow Diagram* (DFD), *Entity Relationship Diagram* (ERD), basis data, input, output, dan proses sistem. Setiap data yang muncul dalam proses analisis sebaiknya memiliki definisi yang jelas sehingga tidak terjadi perbedaan pemahaman antara analis sistem, programmer, database administrator, dan pengguna.

Sebagai contoh, pada Sistem Informasi Akademik terdapat data NIM. Kamus data tidak hanya mencatat bahwa NIM merupakan nama suatu data, tetapi juga menjelaskan bahwa NIM merupakan identitas unik mahasiswa, memiliki tipe data tertentu, memiliki

panjang tertentu, bersifat wajib diisi, dan digunakan sebagai primary key pada tabel mahasiswa.

Dengan demikian, kamus data dapat dirumuskan sebagai: *Kamus data adalah katalog metadata yang mendeskripsikan secara sistematis struktur, arti, karakteristik, sumber, penggunaan, dan aturan suatu data dalam sistem informasi.*

3.17.2 Fungsi Kamus Data

Kamus data memiliki beberapa fungsi penting dalam analisis dan perancangan sistem informasi.

1. Mendokumentasikan Data, kamus data menjadi dokumentasi formal mengenai seluruh data yang digunakan oleh sistem. Dengan dokumentasi tersebut, seluruh pihak mempunyai pemahaman yang sama mengenai data yang digunakan
2. Menjelaskan Arti Data. Nama data terkadang belum cukup untuk menjelaskan maknanya. Sebagai contoh, istilah status dapat memiliki banyak arti. Dalam kamus data dapat dijelaskan: status_mahasiswa = kondisi akademik mahasiswa yang terdiri dari Aktif, Cuti, Lulus, atau Nonaktif. Definisi tersebut menghindari interpretasi yang berbeda.
3. Menentukan Tipe dan Format Data. Kamus data dapat digunakan untuk menentukan apakah suatu data menggunakan: Integer, Varchar, Date, Boolean
4. Menentukan Panjang Data. Kamus data menentukan batas panjang suatu data. Nama Mahasiswa → Varchar(25) Hal ini penting agar struktur database dapat dirancang secara konsisten.
5. Menentukan Primary Key dan Foreign Key. Kamus data dapat memberikan informasi mengenai fungsi

setiap field dalam database. Contoh: nim → Primary Key pada tabel Mahasiswa. id_prodi → Foreign Key pada tabel Mahasiswa yang mengacu ke tabel Program Studi.

6. Mendukung Perancangan Database. Kamus data dapat menjadi dasar dalam pembuatan tabel database.

Dari kamus data:

Nama Field	Tipe	Panjang
nim	Char	12
nama_mahasiswa	Varchar	100
email	Varchar	100

dapat dibentuk tabel:

```
CREATE TABLE mahasiswa
(nim CHAR(12) PRIMARY KEY,
  nama_mahasiswa VARCHAR(100),
  email VARCHAR(100) );
```

7. Mengurangi Kesalahan Pengembangan. Programmer dapat menggunakan kamus data sebagai acuan ketika membuat program sehingga kesalahan penamaan field, tipe data, maupun format data dapat diminimalkan.
8. Mempermudah Pemeliharaan Sistem. Apabila sistem dikembangkan atau dimodifikasi, pengembang dapat melihat dokumentasi data melalui kamus data.
9. Menjadi Media Komunikasi. Kamus data menjadi media komunikasi antara: Analis sistem, Database designer, Programmer, Database administrator, Pengguna, Manajemen

3.17.3 Struktur Kamus Data

Tidak ada satu format tunggal yang wajib digunakan untuk semua proyek. Struktur kamus data dapat disesuaikan dengan kompleksitas sistem. Namun, untuk perancangan sistem informasi akademik, beberapa elemen berikut sangat disarankan.

No	Elemen	Penjelasan
1	Nama Data	Nama field atau elemen data
2	Alias	Nama lain yang digunakan
3	Deskripsi	Penjelasan mengenai arti data
4	Tipe Data	Jenis data yang digunakan
5	Panjang	Ukuran maksimum data
6	Format	Format penulisan data
7	Default	Nilai awal apabila tersedia
8	Nullability	Apakah data boleh kosong
9	Key	Status PK, FK, atau bukan
10	Domain	Nilai yang diperbolehkan
11	Sumber	Asal data
12	Penggunaan	Proses atau tabel yang menggunakan data
13	Aturan Validasi	Ketentuan validasi data
14	Contoh Nilai	Contoh nilai yang valid

Contoh Kamus Data Sistem Informasi Akademik

Kamus Data Tabel Mahasiswa

Nama Field	Tipe	Panjang	Key	Null	Deskripsi	Contoh
nim	char	12	PK	Tidak	Nomor identitas mahasiswa	2026001001
nama_mhs	varchar	100	-	Tidak	Nama lengkap mahasiswa	Andi Pratama
jenis_kelamin	char	1	-	Tidak	Jenis kelamin mahasiswa	L
tempat_lahir	varchar	50	-	Ya	Tempat lahir	Malang
tanggal_lahir	date	-	-	Ya	Tanggal lahir	20/05/2004
alamat	varchar	255	-	Ya	Alamat mahasiswa	Malang
email	varchar	100	UQ	Ya	Email mahasiswa	andi@gmail.com
id_prodi	int	11	FK	Tidak	Program studi mahasiswa	101
status_mhs	varchar	20	-	Tidak	Status akademik mahasiswa	Aktif

3.17.4 Manfaat Kamus Data dalam Pengembangan SI

Penggunaan kamus data memberikan manfaat pada berbagai tahap pengembangan sistem.

Pada tahap analisis:

- membantu mengidentifikasi data;
- memperjelas definisi data;
- mengurangi ambiguitas kebutuhan.

Pada tahap desain:

- membantu menentukan struktur database;
- membantu menentukan tipe dan ukuran data;
- mendukung proses normalisasi;
- membantu menentukan PK dan FK.

Pada tahap implementasi:

- menjadi acuan programmer;
- membantu pembuatan tabel database;
- membantu validasi input;
- membantu pengembangan API.

Pada tahap pemeliharaan:

- memudahkan pemahaman struktur database;
- mempercepat proses troubleshooting;
- menjadi dokumentasi ketika terjadi pergantian pengembang.

3.18 Perancangan Struktur Tabel

Perancangan struktur tabel merupakan tahap dalam perancangan basis data yang bertujuan menentukan bentuk fisik dari tabel-tabel yang telah diperoleh melalui proses analisis, ERD, normalisasi, dan transformasi ke model relasional. Pada tahap ini, setiap tabel ditentukan secara lebih rinci, meliputi:

- Nama tabel.
- Nama kolom.
- Tipe data.
- Panjang atau ukuran data.
- Primary Key.
- Foreign Key.
- Constraint.
- Default Value.
- Auto Increment.
- Index.
- Unique.
- Check Constraint

Dengan demikian, jika pada tahap ERD sistem masih digambarkan secara konseptual, maka pada tahap perancangan struktur tabel rancangan tersebut mulai diterjemahkan menjadi struktur yang dapat langsung diimplementasikan pada DBMS. Secara sederhana, alur perancangannya dapat urutkan sebagai berikut : Kebutuhan Sistem-> ERD-> Normalisasi-> Model Relasional -> Kamus Data -> Struktur Tabel -> Implementasi -> MySQL.

Perancangan struktur tabel harus dilakukan secara hati-hati karena keputusan yang dibuat pada tahap ini akan berpengaruh terhadap integritas, keamanan, kinerja, dan kemudahan pemeliharaan basis data.

3.18.1 Penentuan Tipe Data

a). Pengertian

Tipe data menentukan jenis nilai yang dapat disimpan pada suatu kolom. MySQL menyediakan berbagai kelompok tipe data, antara lain tipe numerik, karakter/string, tanggal dan waktu, serta tipe data khusus seperti JSON. Pemilihan tipe data harus disesuaikan dengan karakteristik data yang akan disimpan.

Kesalahan dalam menentukan tipe data dapat menyebabkan:

- pemborosan ruang penyimpanan;
- kesalahan pengolahan data;
- kesulitan dalam pencarian;
- kesalahan perhitungan;
- penurunan performa database.

b). Tipe Data Numerik

Tipe numerik digunakan untuk menyimpan bilangan.

Tipe	Kegunaan	Contoh
TINYINT	Bilangan kecil	1, 2, 3
SMALLINT	Bilangan relatif kecil	100, 200
INT	Bilangan bulat umum	10000
BIGINT	Bilangan bulat sangat besar	1000000000
DECIMAL	Bilangan presisi tetap	85.50
FLOAT	Bilangan pecahan	3.14
DOUBLE	Bilangan pecahan presisi lebih tinggi	123.456

c). Tipe Data Karakter

Tipe karakter digunakan untuk menyimpan teks.

Tipe	Kegunaan
CHAR(n)	Teks dengan panjang relatif tetap
VARCHAR(n)	Teks dengan panjang bervariasi

Tipe	Kegunaan
TEXT	Teks yang relatif panjang
ENUM	Nilai yang berasal dari pilihan tertentu

d). Tipe Data Tanggal dan Waktu

MySQL menyediakan tipe data seperti:

- DATE
- TIME
- DATETIME
- TIMESTAMP
- YEAR

e). Prinsip Pemilihan Tipe Data

Beberapa prinsip yang perlu diperhatikan:

- Gunakan tipe data sesuai karakteristik data. NIM yang berupa identitas dan dapat mengandung angka yang tidak digunakan untuk perhitungan lebih tepat diperlakukan sebagai karakter: nim CHAR(12) bukan nim INT
- Hindari ukuran yang berlebihan. Jika nama mahasiswa maksimal 100 karakter, tidak perlu menggunakan: VARCHAR(1000) , cukup VARCHAR(50)
- Gunakan DECIMAL untuk data yang membutuhkan presisi. Untuk data seperti nilai, harga, atau IPK, decimal lebih tepat daripada tipe floating-point ketika presisi desimal harus dikontrol.

3.18.2 Constraint

a. Pengertian

Constraint adalah aturan yang diterapkan pada tabel atau kolom untuk menjaga validitas dan integritas data.

Constraint membantu memastikan bahwa data yang dimasukkan ke dalam database memenuhi aturan bisnis yang telah ditentukan.

Beberapa constraint utama dalam MySQL antara lain:

- *Primary Key*
- *Foreign Key*
- *Not Null*
- *Unique*
- *Check*

MySQL menyediakan mekanisme constraint pada definisi tabel melalui CREATE TABLE.

b. *Primary Key*

Primary Key merupakan atribut yang digunakan sebagai identitas unik setiap baris.

Contoh:

CREATE TABLE mahasiswa (nim CHAR(12) **NOT NULL**, nama_mahasiswa VARCHAR(100) **NOT NULL**, **PRIMARY KEY** (nim));

Aturan utama:

- setiap nilai harus unik;
- tidak boleh bernilai NULL;

- satu tabel hanya mempunyai satu primary key, walaupun primary key tersebut dapat terdiri dari beberapa kolom *(composite key)*.

c. *Foreign Key*

Foreign Key digunakan untuk menghubungkan suatu tabel dengan tabel lain dan membantu menjaga integritas referensial. Contoh:

CREATE TABLE mahasiswa (nim CHAR(12) **PRIMARY KEY**, nama_mahasiswa VARCHAR(100) **NOT NULL**, id_prodi INT **NOT NULL**, **FOREIGN KEY** (id_prodi) **REFERENCES** program_studi(id_prodi));

Artinya, nilai id_prodi pada tabel mahasiswa harus mengacu pada data yang tersedia pada tabel program_studi. MySQL juga mendukung tindakan referensial seperti:

ON DELETE CASCADE, ON DELETE RESTRICT, ON DELETE SET NULL, ON UPDATE CASCADE. yang dapat digunakan sesuai aturan bisnis sistem.

d. *Not Null*

NOT NULL digunakan untuk memastikan sebuah kolom harus memiliki nilai. Contoh: nama_mahasiswa VARCHAR(100) **NOT NULL** Artinya nama mahasiswa wajib diisi.

3.18.3 *Default Value*

Default Value merupakan nilai yang otomatis digunakan oleh MySQL ketika data tidak diberikan pada saat proses INSERT, sepanjang kondisi kolom dan definisinya memungkinkan penggunaan nilai default. Default Value sangat berguna untuk data yang memiliki nilai awal yang umum atau standar. Contoh: status_mahasiswa Varchar(20) **Not Null Default** 'Aktif'

Jika pengguna tidak memberikan nilai status_mahasiswa, maka MySQL akan menggunakan: Aktif. MySQL mendukung penggunaan Default untuk menentukan nilai awal sebuah kolom.

3.18.4 *Auto Increment*

A*UTO_INCREMENT* merupakan fitur MySQL yang digunakan untuk menghasilkan nilai numerik secara otomatis ketika sebuah baris baru dimasukkan. Fitur ini sangat berguna untuk membuat ID yang bersifat unik. Contoh: id_mahasiswa INT **NOT NULL** AUTO_INCREMENT,
PRIMARY KEY (id_mahasiswa)

MySQL mendukung AUTO_INCREMENT pada tipe integer dan kolom tersebut harus diindeks; dalam penggunaan yang umum, kolom tersebut dijadikan primary key.

Contoh : CREATE TABLE mahasiswa (id_mahasiswa INT NOT NULL AUTO_INCREMENT, nim CHAR(12) NOT NULL, nama_mahasiswa VARCHAR(100) NOT NULL, PRIMARY KEY (id_mahasiswa), UNIQUE (nim));

Data dapat dimasukkan tanpa menentukan id_mahasiswa:

INSERT INTO mahasiswa (nim, nama_mahasiswa)
VALUES ('2026001001', 'Andi Pratama');

Kemudian:

INSERT INTO mahasiswa (nim, nama_mahasiswa)
VALUES ('2026001002', 'Budi Santoso');

Secara otomatis MySQL dapat menghasilkan:

id_mahasiswa	nim	nama_mahasiswa
1	2026001001	Andi Pratama
2	2026001002	Budi Santoso

3.18.5 Index

Index merupakan struktur yang digunakan untuk membantu MySQL menemukan data secara lebih efisien ketika melakukan pencarian, pengurutan, atau operasi tertentu.

Secara sederhana, index dapat dianalogikan dengan indeks pada sebuah buk**u**. Daripada membaca seluruh halaman untuk menemukan suatu istilah, indeks membantu menemukan lokasi informasi tersebut dengan lebih cepat.

a. Tanpa Index

Misalnya sistem sering melakukan pencarian:

SELECT * FROM mahasiswa WHERE nama_mahasiswa = 'Andi Pratama';

Jika kolom tersebut belum memiliki index, MySQL mungkin harus memeriksa banyak baris.

b. Membuat Index

CREATE INDEX idx_mahasiswa_nama **ON** mahasiswa (nama_mahasiswa);

Setelah index dibuat, pencarian pada kolom tersebut dapat menggunakan struktur index sesuai keputusan optimizer MySQL. Index dapat dibuat ketika tabel dibuat:

CREATE TABLE mahasiswa (id_mahasiswa INT AUTO_INCREMENT **PRIMARY KEY**,
nim CHAR(12) **NOT NULL**,
nama_mahasiswa VARCHAR(100) **NOT NULL**,
INDEX idx_nama_mahasiswa (nama_mahasiswa));

MySQL mendokumentasikan KEY dan INDEX sebagai sinonim dalam definisi index.

c. Kapan Index Digunakan?

Index cocok digunakan pada kolom yang:

- ✓ sering digunakan untuk pencarian;
- ✓ sering digunakan dalam WHERE;
- ✓ sering digunakan dalam JOIN;
- ✓ sering digunakan dalam ORDER BY;
- ✓ menjadi foreign key atau bagian dari pola akses data tertentu.

Namun, terlalu banyak index juga tidak selalu baik karena index membutuhkan ruang penyimpanan dan dapat menambah pekerjaan saat INSERT, UPDATE, atau DELETE.

3.18.6 Unique

UNIQUE digunakan untuk memastikan bahwa nilai pada suatu kolom atau kombinasi kolom tidak boleh sama.

Contoh: NIM harus unik. nim CHAR(12) **NOT NULL UNIQUE.**

Contoh lainnya: email VARCHAR(100) **UNIQUE**

MySQL menggunakan unique index untuk memastikan nilai indeks bersifat berbeda. Untuk kolom

yang mengizinkan NULL, beberapa nilai NULL dapat diperbolehkan.

Contoh:

```
CREATE TABLE mahasiswa (
  id_mahasiswa INT AUTO_INCREMENT PRIMARY KEY, nim CHAR(12) NOT NULL UNIQUE, nama_mahasiswa VARCHAR(100) NOT NULL, email VARCHAR(100) UNIQUE );
```

Jika sudah terdapat: 2026001001, kemudian dimasukkan lagi:

```
INSERT INTO mahasiswa (nim, nama_mahasiswa)
VALUES ('2026001001', 'Citra');
```

maka operasi tersebut akan ditolak karena nilai nim sudah digunakan.

Perbedaan PRIMARY KEY dan UNIQUE

Aspek	PRIMARY KEY	UNIQUE
Nilai harus unik	Ya	Ya
NULL	Tidak diperbolehkan	Dapat diperbolehkan sesuai definisi kolom
Jumlah per tabel	Satu	Dapat lebih dari satu
Fungsi utama	Identitas utama baris	Menjamin keunikan

10.18.7 *Check Constraint*

CHECK CONSTRAINT digunakan untuk membatasi nilai berdasarkan kondisi tertentu. Contoh:

Nilai akademik harus berada antara 0 dan 100.

nilai_angka Decimal(5,2) Check **(nilai_angka** Between **0** And **100)**

Pada MySQL 8.4, CHECK constraint dapat dibuat sebagai column constraint maupun table constraint dan secara default diberlakukan (ENFORCED) apabila tidak dinyatakan lain. Jika kondisi bernilai FALSE, operasi yang melanggar constraint akan gagal.

Contoh

```
CREATE TABLE nilai (
id_nilai BIGINT AUTO_INCREMENT,
nilai_angka DECIMAL(5,2) NOT NULL,
nilai_huruf VARCHAR(2) NOT NULL,
PRIMARY KEY (id_nilai),
CHECK (nilai_angka BETWEEN 0 AND 100),
CHECK (nilai_huruf IN ('A','A-','B+','B',
B-','C+','C','D','E')));
```

Dengan aturan tersebut, data berikut valid:

```
INSERT INTO nilai (nilai_angka, nilai_huruf)
VALUES (85.50, 'A');
```

Sedangkan:

```
INSERT INTO nilai (nilai_angka, nilai_huruf)
VALUES (125.00, 'A');
```

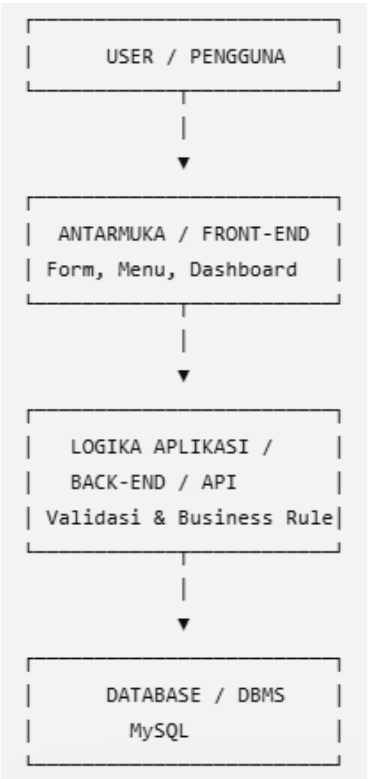
akan ditolak karena 125 berada di luar rentang nilai yang ditentukan.

3.19 Integrasi Antarmuka dengan Basis Data

Integrasi antarmuka dengan basis data merupakan proses menghubungkan antarmuka pengguna (*user interface*/UI) dengan basis data sehingga pengguna dapat memasukkan, melihat, mengubah, dan menghapus data melalui aplikasi tanpa berinteraksi langsung dengan database.

Dalam sistem informasi modern, pengguna berinteraksi dengan form, menu, tabel, tombol, dashboard, dan laporan, sedangkan proses pengelolaan data dilakukan oleh aplikasi melalui lapisan logika pemrograman dan kemudian diteruskan ke DBMS.

Secara sederhana, arsitektur integrasinya dapat digambarkan:



Gambar 3. 7 Arsitektur Integrasi

Dengan arsitektur tersebut, antarmuka tidak seharusnya berkomunikasi langsung secara sembarangan dengan database. Permintaan data idealnya melewati lapisan aplikasi yang bertanggung jawab terhadap autentikasi, otorisasi, validasi, aturan bisnis, dan pengelolaan transaksi. Pada Sistem Informasi Akademik, integrasi ini memungkinkan mahasiswa mengisi KRS, dosen menginput nilai, admin mengelola data mahasiswa dan mata kuliah, serta pimpinan melihat dashboard dan laporan akademik.

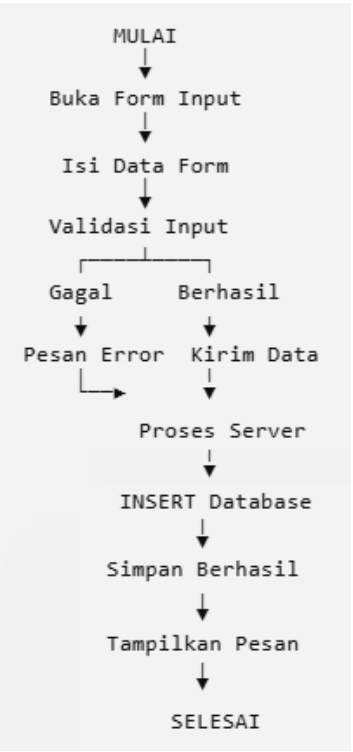
3.19.1 Alur Input Data

Alur input data adalah rangkaian proses yang terjadi sejak pengguna memasukkan data melalui antarmuka sampai data tersebut berhasil disimpan ke dalam basis data.

Contohnya, ketika admin menambahkan mahasiswa baru:

1. Admin membuka form mahasiswa.
2. Admin mengisi NIM.
3. Admin mengisi nama mahasiswa.
4. Admin memilih program studi.
5. Sistem melakukan validasi.
6. Data dikirim ke server.
7. Server melakukan validasi kembali.
8. Sistem menjalankan perintah INSERT.
9. Database menyimpan data.
10. Sistem memberikan pesan bahwa data berhasil disimpan.

Secara konseptual:



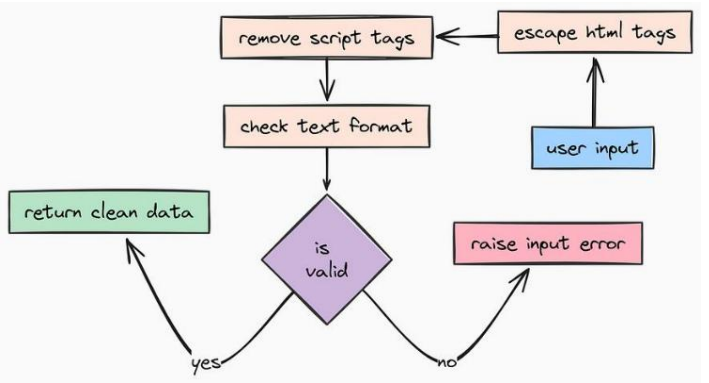
Gambar 3. 8 Alur Input Data

3.19.2 Validasi Data

Validasi data merupakan proses pemeriksaan untuk memastikan bahwa data yang dimasukkan pengguna sesuai dengan aturan yang telah ditentukan. Validasi diperlukan agar database tidak dipenuhi data yang:

- kosong;
- salah format;
- tidak logis;
- duplikat;
- berada di luar batas yang ditentukan;
- tidak sesuai dengan relasi database.

Dalam pengembangan aplikasi, validasi sebaiknya dilakukan **lebih dari satu lapisan**. Validasi pada antarmuka memberikan umpan balik cepat kepada pengguna, sedangkan validasi pada server menjadi lapisan yang lebih dipercaya sebelum data diproses dan disimpan.



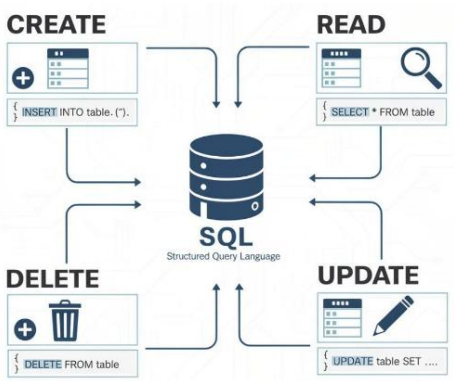
Gambar 3. 9 Validasi Data

3.19.3 CRUD

CRUD merupakan singkatan dari:

- ✓ C = Create
- ✓ R = Read
- ✓ U = Update
- ✓ D = Delete

CRUD merupakan operasi dasar yang digunakan aplikasi untuk mengelola data dalam database.



Gambar 3. 10 Operasi CRUD

Hubungan CRUD dengan Antarmuka

Operasi	Antarmuka	SQL
Create	Tambah/Simpan	INSERT
Read	Lihat/Cari	SELECT
Update	Edit/Ubah	UPDATE
Delete	Hapus	DELETE

3.19.4 Form Login

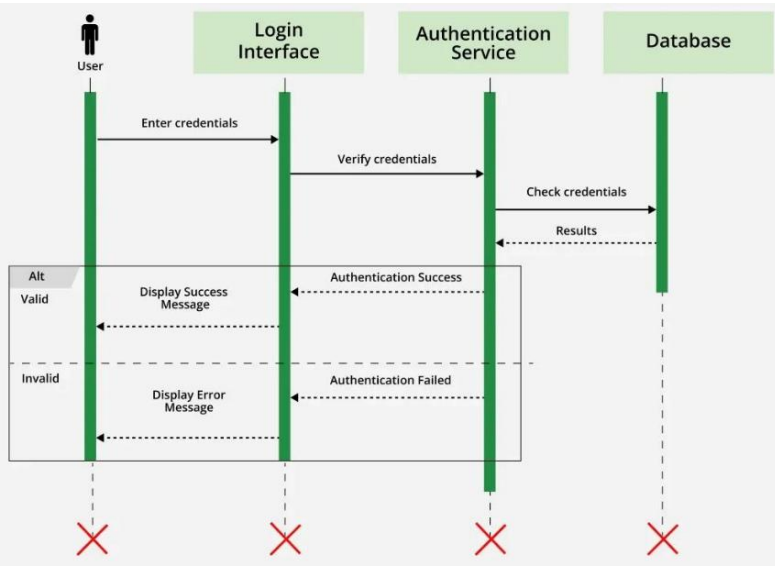
Form login merupakan antarmuka yang digunakan untuk melakukan **autentikasi pengguna** sebelum pengguna memperoleh akses terhadap fitur tertentu.

Dalam Sistem Informasi Akademik, pengguna dapat terdiri atas:

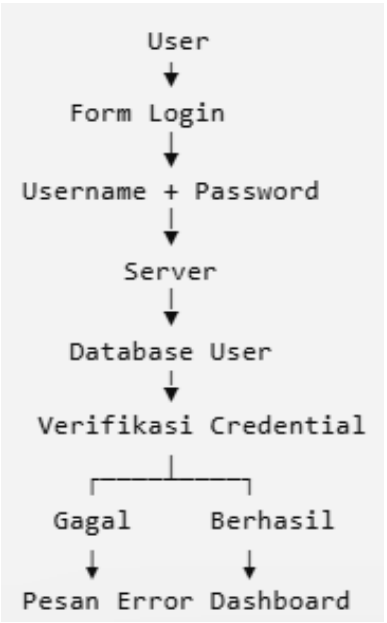
- ✓ Administrator;
- ✓ Mahasiswa;
- ✓ Dosen;

- ✓ Kaprodi;
- ✓ Pimpinan.

Setiap pengguna dapat mempunyai hak akses yang berbeda.



Gambar 3. 11 Proses Login



Gambar 3. 12 Alur Login

3.19.5 Form Master Data

Master data adalah data utama yang menjadi referensi bagi proses transaksi dalam sistem.

Form master umumnya menggabungkan: FORM INPUT +DATA TABLE + SEARCH + EDIT + DELETE. Sehingga satu halaman dapat menyediakan hampir seluruh kebutuhan CRUD. Dalam Sistem Informasi Akademik, contoh master data antara lain:

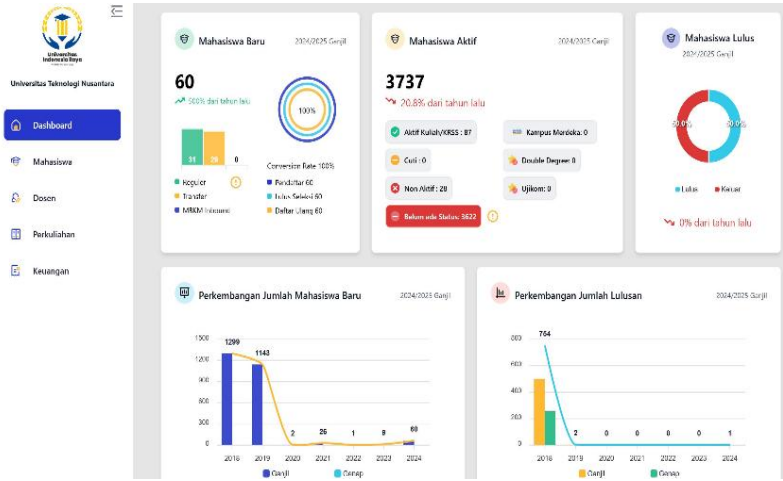
- Program Studi
- Mahasiswa
- Dosen
- Mata Kuliah
- Ruang
- Tahun Akademik
- Kurikulum



Gambar 3. 13 Form Master Data

3.19.6 Dashboard

Dashboard adalah antarmuka yang menyajikan informasi penting secara ringkas melalui angka indikator, grafik, tabel, diagram, dan visualisasi sehingga pengguna dapat memahami kondisi sistem dengan cepat. Dashboard berbeda dengan form transaksi. Form digunakan untuk memasukkan atau mengelola data, sedangkan dashboard lebih berorientasi pada monitoring dan pengambilan keputusan.



Gambar 3. 14 Dashboard Sistem Informasi

Contoh Dashboard Akademik

```
┌──────────────────────────────────────────────────┐
│              DASHBOARD AKADEMIK                  │
├──────────────────────────────────────────────────┤
│                                                  │
│  MAHASISWA      DOSEN       MATA KULIAH     KRS  │
│    1.250         85             320         1.180│
│                                                  │
├──────────────────────┬───────────────────────────┤
│ Jumlah Mahasiswa     │ Distribusi Mahasiswa      │
│                      │                           │
│      ▐               │      TI ▐▐▐▐▐▐▐▐▐▐        │
│      ▐               │      SI ▐▐▐▐▐▐▐▐          │
│   ▐  ▐  ▐            │      MI ▐▐▐▐▐▐            │
│   ▐  ▐  ▐            │                           │
├──────────────────────┼───────────────────────────┤
│ IPS Rata-rata        │ Status Mahasiswa          │
│                      │                           │
│       3.42           │ Aktif        1.100        │
│                      │ Cuti            50        │
│                      │ Lulus          100        │
└──────────────────────┴───────────────────────────┘
```

3.19.7 Laporan

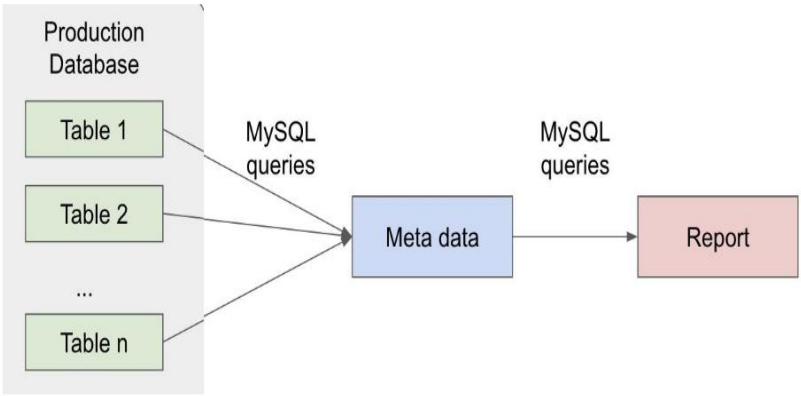
Laporan merupakan keluaran sistem yang menyajikan data yang telah diproses dalam bentuk yang terstruktur dan mudah dibaca.

Laporan dapat digunakan untuk:

- dokumentasi;
- monitoring;
- evaluasi;
- pertanggungjawaban;
- pengambilan keputusan.

Contoh Sistem Informasi Akademik dapat menghasilkan:

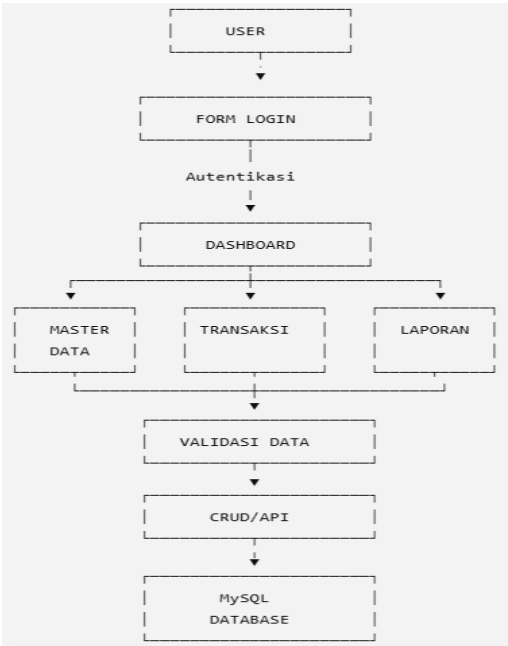
- Laporan data mahasiswa.
- Laporan KRS.
- Kartu Hasil Studi (KHS).
- Transkrip nilai.
- Daftar nilai mahasiswa.
- Daftar mahasiswa aktif.
- Daftar mahasiswa lulus.
- Laporan IPS dan IPK.
- Laporan jumlah mahasiswa berdasarkan program studi.
- Laporan aktivitas perkuliahan.



Gambar 3. 15 Sistem Pelaporan

3.19.8 Integrasi Keseluruhan Antarmuka dan Basis Data

Ketujuh komponen pada subbab ini sebenarnya merupakan bagian dari satu kesatuan sistem.



Gambar 3. 16 Integrasi Keseluruhan

Alur tersebut menunjukkan bahwa: Login → Dashboard → Form/Transaksi → Validasi → CRUD → Database → Dashboard/Laporan merupakan suatu siklus yang saling terintegrasi.

3.19.9 Integrasi Antarmuka Berdasarkan Hak Akses

Sistem Informasi Akademik biasanya memiliki beberapa jenis pengguna. Oleh sebab itu, antarmuka yang ditampilkan sebaiknya disesuaikan dengan hak akses.

Pengguna	Menu Utama	Data
Administrator	Dashboard, Master, Transaksi, Laporan	Semua
Dosen	Dashboard, Jadwal, Nilai	Data perkuliahan
Mahasiswa	Dashboard, KRS, KHS	Data pribadi
Kaprodi	Dashboard, Monitoring, Laporan	Data program studi
Pimpinan	Dashboard, Laporan	Data agregat

3.19.10 Prinsip Keamanan Integrasi Antarmuka dan Database

Integrasi antarmuka dengan database tidak hanya memperhatikan fungsi, tetapi juga keamanan. Beberapa prinsip yang perlu diterapkan antara lain:

- *Authentication* : Memastikan identitas pengguna.
- *Authorization* : Memastikan pengguna hanya dapat mengakses fungsi yang menjadi haknya.

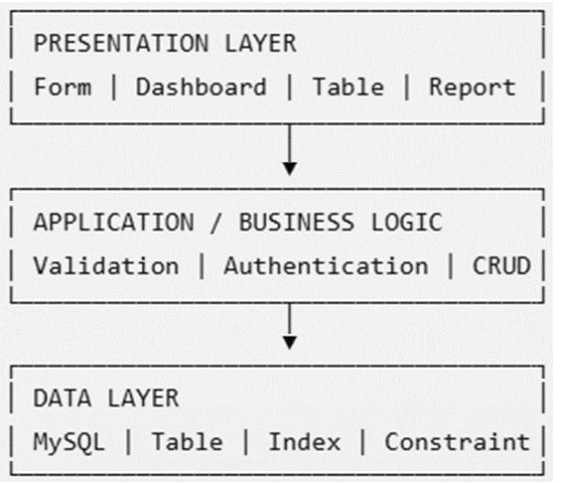
- *Password Hashing*: Password tidak disimpan sebagai teks biasa.
- *Prepared Statement*: Gunakan parameterized query untuk mengurangi risiko SQL injection.
- *Validasi Input:* Input pengguna harus diperiksa sebelum diproses.
- *Session Management:* Session pengguna harus dikelola dengan baik.
- *Least Privilege*: Akun aplikasi sebaiknya hanya mempunyai hak database yang memang diperlukan.
- *Audit Trail*: Untuk data penting, sistem dapat mencatat.

3.19.11 Hubungan Antarmuka, Proses, dan Database

Dalam sistem informasi, antarmuka (*user interface*), proses bisnis/aplikasi, dan database merupakan tiga komponen utama yang saling berhubungan. Antarmuka menjadi media interaksi pengguna dengan sistem, proses mengolah permintaan dan aturan bisnis, sedangkan database menyimpan dan menyediakan data yang diperlukan sistem.

Ketiganya membentuk alur kerja yang terintegrasi dari input → proses → penyimpanan/pengambilan data → output**.**

Integrasi dapat dipahami melalui tiga lapisan utama:



Gambar 3. 17 Interaksi Tiga Lapisan Utama

1. Antarmuka sebagai Media Interaksi

Antarmuka (*User Interface*/UI) merupakan bagian sistem yang berinteraksi langsung dengan pengguna. Melalui antarmuka, pengguna dapat memasukkan data, memilih menu, menjalankan fungsi, serta melihat hasil dari proses sistem. Contohnya pada Sistem Informasi Akademik, mahasiswa menggunakan halaman KRS untuk memilih mata kuliah dan mengirimkan data pengisian KRS. Dengan demikian, antarmuka berfungsi sebagai pintu masuk dan keluaran informasi antara pengguna dan sistem.

2. Proses sebagai Pengolah Data

Setelah pengguna memberikan input melalui antarmuka, data diteruskan ke lapisan proses atau aplikasi. Bagian ini menjalankan logika sistem dan aturan bisnis untuk menentukan apakah data dapat

diproses. Misalnya, ketika mahasiswa memilih mata kuliah, sistem dapat melakukan pemeriksaan:

- apakah mahasiswa masih aktif;
- apakah mata kuliah tersedia;
- apakah prasyarat telah terpenuhi;
- apakah jumlah SKS tidak melebihi batas;
- apakah jadwal mata kuliah tidak bentrok.

Hasil pemeriksaan tersebut kemudian menentukan tindakan sistem, seperti menyimpan data, menolak input, atau menampilkan pesan kesalahan.

3. Database sebagai Penyimpanan Data

Database berfungsi menyimpan data yang digunakan dan dihasilkan oleh sistem secara terstruktur. Database tidak hanya menyimpan data, tetapi juga menyediakan data ketika dibutuhkan oleh proses aplikasi. Pada Sistem Informasi Akademik, database dapat menyimpan:

- data mahasiswa;
- data dosen;
- data mata kuliah;
- data program studi;
- data KRS;
- data nilai;
- data jadwal perkuliahan.

Database menjadi **sumber data utama** yang mendukung proses sistem.

4. Alur Hubungan Ketiga Komponen

Hubungan antarmuka, proses, dan database dapat digambarkan secara sederhana sebagai berikut:

Pengguna → Antarmuka → Proses/Aplikasi → Database

Kemudian hasilnya dikembalikan melalui:

Database → Proses/Aplikasi → Antarmuka → Pengguna

Contohnya pada proses pengisian KRS:

- Pengguna memilih mata kuliah melalui halaman KRS.
- Antarmuka mengirimkan data pilihan kepada aplikasi.
- Proses aplikasi memvalidasi data berdasarkan aturan akademik.
- Database diperiksa untuk memperoleh data mahasiswa, mata kuliah, prasyarat, dan jadwal.
- Proses aplikasi menentukan apakah pengisian KRS valid.
- Jika valid, data KRS disimpan ke database.
- Antarmuka menampilkan hasil kepada pengguna, misalnya *"KRS berhasil disimpan"*.

5. Hubungan dalam Arsitektur Sistem

Secara konseptual, ketiga komponen tersebut dapat ditempatkan dalam arsitektur berlapis:

Komponen	Fungsi Utama	Contoh
Antarmuka (UI)	Menerima input dan menampilkan output	Form KRS, dashboard, tabel
Proses/Aplikasi	Menjalankan logika dan aturan bisnis	Validasi KRS, perhitungan SKS
Database	Menyimpan dan menyediakan data	Tabel mahasiswa, mata kuliah, KRS

Pemisahan tersebut membuat sistem lebih terstruktur, mudah dikembangkan, dipelihara, dan diuji. Antarmuka tidak seharusnya menangani seluruh logika bisnis secara langsung, sedangkan database berfokus pada pengelolaan data. Proses aplikasi menjadi penghubung utama antara keduanya.

6. Contoh Sederhana

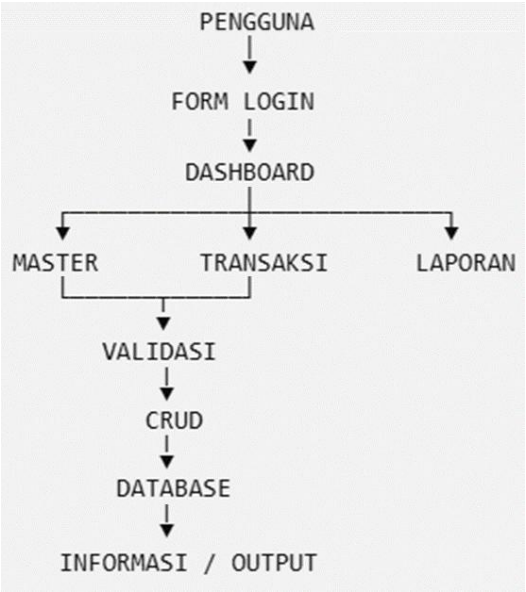
Pada form login, hubungan ketiganya adalah:

UI Login → Proses Autentikasi → Database Pengguna

1. Pengguna memasukkan *username* dan *password*.
2. UI mengirimkan data ke proses autentikasi.
3. Proses mencari akun yang sesuai di database.
4. Database mengembalikan informasi akun.
5. Proses memverifikasi kredensial.
6. UI menampilkan dashboard jika berhasil atau pesan kesalahan jika gagal.

1. Integrasi antarmuka dengan basis data merupakan tahap penting dalam pembangunan sistem informasi karena menghubungkan kebutuhan pengguna dengan data yang disimpan dalam database.
2. Dalam Sistem Informasi Akademik, integrasi tersebut dapat diwujudkan melalui komponen utama, yaitu:
    - Alur input data, untuk memasukkan data ke sistem.
    - Validasi data, untuk memastikan data benar dan sesuai aturan.
    - CRUD, untuk mengelola data melalui Create, Read, Update, dan Delete.
    - Form Login, untuk autentikasi dan pengaturan akses pengguna.
    - Form Master Data, untuk mengelola data referensi utama.
    - Dashboard, untuk menyajikan informasi ringkas dan indikator kinerja.
    - Laporan, untuk menghasilkan informasi terstruktur yang dapat digunakan dalam monitoring, evaluasi, dan pengambilan keputusan.

Hubungan seluruh komponen tersebut dapat diringkas:



Gambar 3. 18 Hubungan Komponen Sistem Informasi Akademik

Dengan rancangan integrasi yang baik, database tidak hanya menjadi tempat penyimpanan data, tetapi menjadi bagian dari sebuah sistem yang mampu mengubah data → informasi → pengetahuan untuk mendukung pengambilan keputusan.

3.20 Studi Kasus Perancangan S.I. Perpustakaan

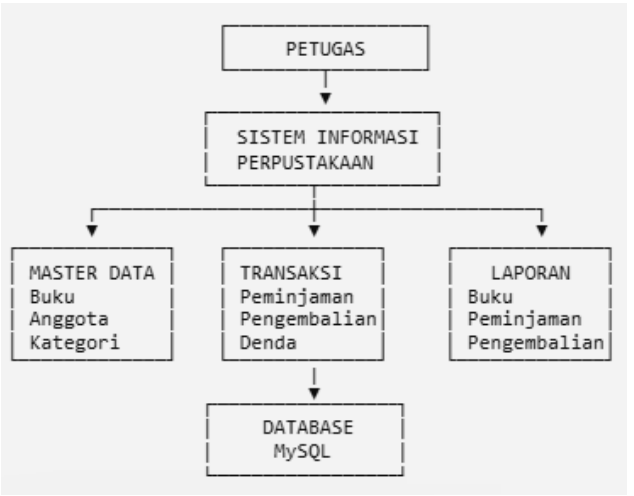
3.21.1 Gambaran Umum Studi Kasus

Perpustakaan merupakan salah satu unit pelayanan yang membutuhkan pengelolaan data secara terstruktur. Aktivitas perpustakaan tidak hanya berkaitan dengan penyimpanan data buku, tetapi juga mencakup

pengelolaan anggota, kategori buku, penulis, penerbit, peminjaman, pengembalian, denda, serta penyusunan laporan. Pada sistem manual, proses tersebut sering dilakukan menggunakan buku pencatatan atau lembar kerja sehingga berpotensi menimbulkan beberapa permasalahan, seperti:

- pencarian buku membutuhkan waktu relatif lama;
- data buku dapat tercatat lebih dari satu kali;
- sulit mengetahui status buku secara cepat;
- pencatatan peminjaman dan pengembalian kurang terintegrasi;
- perhitungan keterlambatan dan denda dapat mengalami kesalahan;
- pembuatan laporan membutuhkan waktu;
- sulit mengetahui jumlah buku yang tersedia secara real-time.

Oleh karena itu, diperlukan **Sistem Informasi Perpustakaan** yang mampu mengintegrasikan seluruh proses tersebut dalam satu sistem. Secara konseptual, sistem dapat digambarkan:



Gambar 3. 19 Konsep Sistem Informasi Perpustakaan

3.21.2 Analisis Kebutuhan

A. Pengertian

Analisis kebutuhan merupakan tahap untuk mengidentifikasi kebutuhan pengguna dan fungsi yang harus disediakan oleh sistem.

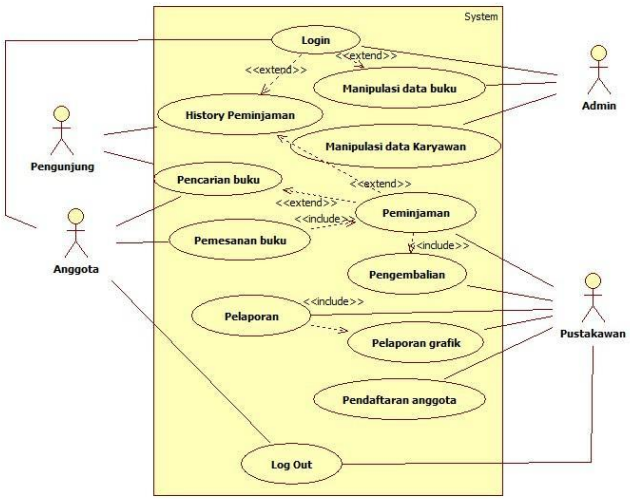
Dalam studi kasus ini terdapat tiga kelompok pengguna utama:

1. **Administrator**
2. **Petugas perpustakaan**
3. **Anggota perpustakaan**

Masing-masing pengguna memiliki kebutuhan dan hak akses berbeda.

B. Identifikasi Aktor

Aktor	Deskripsi
Administrator	Mengelola pengguna dan seluruh konfigurasi sistem
Petugas	Mengelola buku, anggota, peminjaman, pengembalian, dan laporan
Anggota	Melihat katalog dan riwayat peminjaman



Gambar 3. 20 Aktor Sistem

C. Kebutuhan Fungsional

Kebutuhan fungsional menunjukkan fungsi yang harus disediakan sistem.

1. **Modul Login**

   Sistem harus menyediakan:

   - login pengguna;
   - logout;
   - validasi username dan password;
   - pengaturan hak akses.

2. **Modul Data Anggota**

   Petugas dapat:

   - menambah anggota;
   - melihat anggota;
   - mencari anggota;
   - mengubah anggota;
   - menghapus anggota.

3. **Modul Data Buku**

   Petugas dapat:

   - menambah buku;
   - mengubah buku;
   - menghapus buku;
   - mencari buku;
   - melihat stok buku.

4. **Modul Kategori :** Petugas dapat mengelola kategori buku.

5. **Modul Peminjaman,** Sistem dapat:

   - mencatat transaksi peminjaman;
   - memilih anggota;
   - memilih buku;

- menentukan tanggal peminjaman;
- menentukan tanggal jatuh tempo.

6. **Modul Pengembalian,** Sistem dapat:
   - mencatat pengembalian;
   - menghitung keterlambatan;
   - menghitung denda;
   - mengubah status buku.
7. **Modul Laporan,** Sistem dapat menghasilkan:
   - laporan data buku;
   - laporan anggota;
   - laporan peminjaman;
   - laporan pengembalian;
   - laporan keterlambatan;
   - laporan denda.

D. Kebutuhan Nonfungsional

Selain fungsi utama, sistem harus memenuhi kebutuhan nonfungsional.

Aspek	Kebutuhan
Keamanan	Login dan hak akses
Performance	Pencarian data cepat
Usability	Antarmuka mudah digunakan
Reliability	Data tersimpan secara konsisten
Maintainability	Sistem mudah dikembangkan
Availability	Sistem digunakan sesuai kebutuhan operasional

Aspek	Kebutuhan
Backup	Database dapat dicadangkan secara berkala

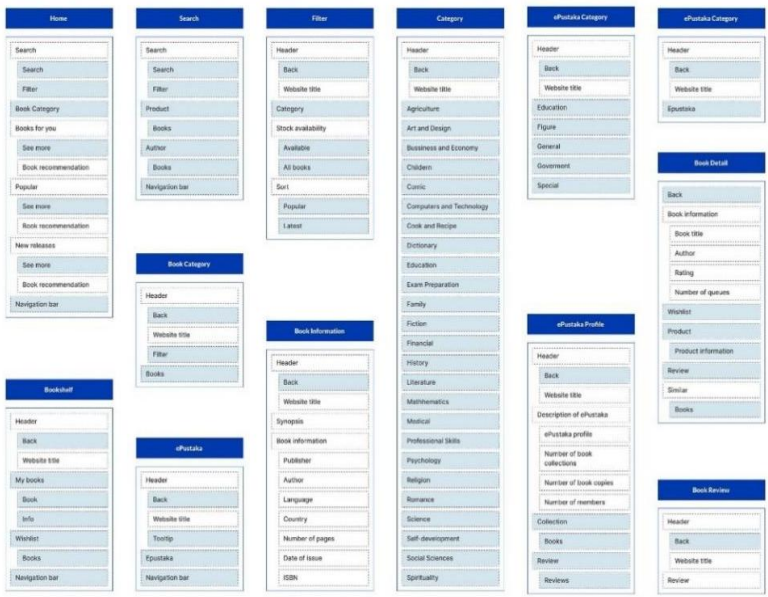
3.21.3 *Desain Wireframe*

A. Pengertian *Wireframe*

Wireframe merupakan rancangan awal struktur antarmuka yang menunjukkan posisi komponen tanpa berfokus pada warna, ikon, gambar, atau detail visual.

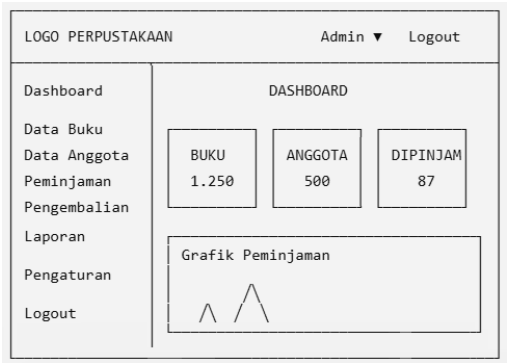
Wireframe digunakan untuk menjawab pertanyaan:

"Di mana setiap elemen antarmuka akan ditempatkan?"



Gambar 3. 21 Contoh *Wireframe* Sistem Perpustakaan

B. *Wireframe Dashboard*



Gambar 3. 22 Contoh Rancangan

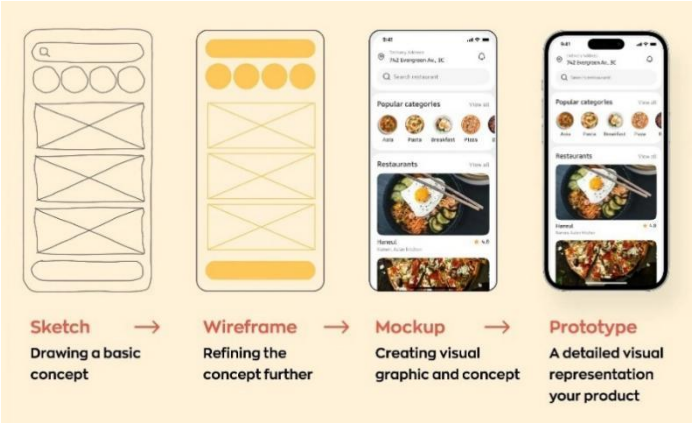
3.21.4 *Prototype*

A. Pengertian *Prototype*

Prototype adalah model awal sistem yang dibuat untuk menggambarkan bagaimana sistem akan digunakan sebelum sistem dikembangkan secara penuh. Prototype memiliki tingkat detail yang lebih tinggi dibandingkan wireframe.

Perbedaannya:

Wireframe	***Prototype***
Struktur dasar	Simulasi interaksi
Fokus layout	Fokus pengalaman pengguna
Relatif sederhana	Lebih realistis
Tidak selalu interaktif	Dapat interaktif



Gambar 3. 23 Perbedaan Wireframe dan Prototype

B. Contoh Prototype Form Peminjaman

```
┌──────────────────────────────────────────┐
│          TRANSAKSI PEMINJAMAN            │
├──────────────────────────────────────────┤
│ No. Transaksi : PJM-2026-0001            │
│                                          │
│ Anggota                                  │
│ [ Cari NIM / Nama ▼ ]                    │
│ Nama       : Andi Pratama                │
│ NIM        : 2026001001                  │
│                                          │
│ Buku                                     │
│ [ Cari Kode / Judul ▼ ]                  │
│ Judul      : Analisis Sistem Informasi   │
│ Kode       : BK001                       │
│                                          │
│ Tgl Pinjam : [ 11-08-2026 ]              │
│ Jatuh Tempo: [ 18-08-2026 ]              │
│                                          │
│             [Simpan] [Batal]             │
└──────────────────────────────────────────┘
```

Gambar 3. 24 Perbedaan Wireframe dan Prototype

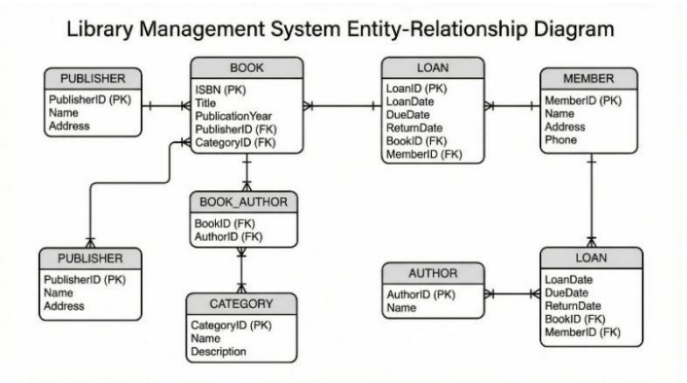
Prototype tersebut dapat dikembangkan menjadi aplikasi aktual setelah mendapatkan persetujuan dari pengguna.

3.21.5 Perancangan ERD

A. Identifikasi Entitas

Berdasarkan analisis kebutuhan, beberapa entitas utama yang diperlukan adalah:

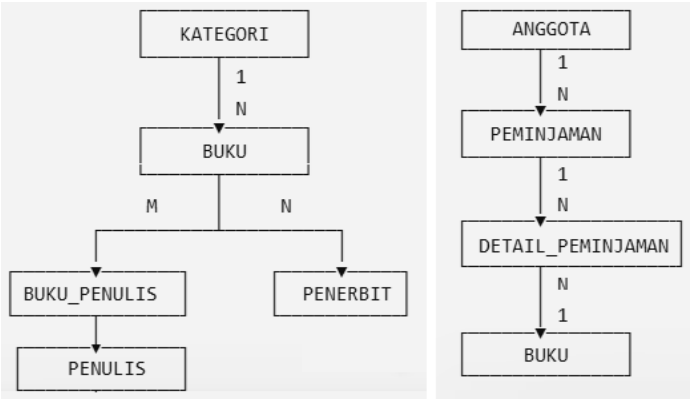
1. user
2. anggota
3. buku
4. kategori
5. penerbit
6. penulis
7. buku_penulis
8. peminjaman
9. detail_peminjaman
10. pengembalian
11. denda



Gambar 3. 25 ERD Sistem Informasi Perpustakaan

B. Relasi Antarentitas

Secara konseptual:



Gambar 3. 26 Relalsi Antar entitas Sistem Informasi Perpustakaan

3.21.6 Normalisasi

Normalisasi dilakukan untuk mengurangi redundansi dan meningkatkan integritas data.

Misalkan pada awalnya data peminjaman disimpan dalam satu tabel:

No Pinjam	NIM	Nama	Buku	Penulis	Tgl Pinjam	Tgl Kembali
PJ001	1	Andi	Basis Data	Kadir	01-Aug	05-Aug
PJ001	1	Andi	Pemrograman	Siregar	01-Aug	05-Aug

Struktur tersebut mengandung redundansi.

A. Bentuk Tidak Ternormalisasi

PEMINJAMAN

- ✓ NoPinjam
- ✓ NIM
- ✓ Nama
- ✓ Buku2
- ✓ Penulis2
- ✓ TanggalPinjam

- ✓ Buku1
- ✓ Penulis1
- ✓ TanggalKembali

Masalah:

- terdapat kelompok data berulang;
- sulit menambah buku ketiga;
- terjadi redundansi;
- struktur tidak fleksibel.

B. *First Normal Form* (1NF)

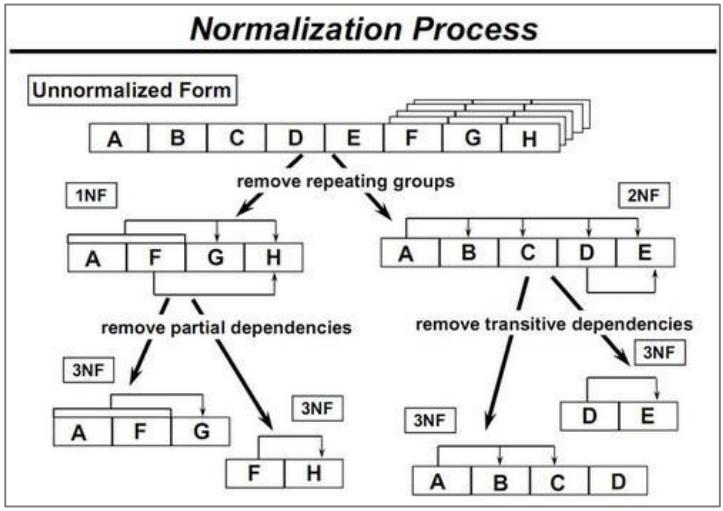
Data harus mempunyai nilai atomik.

PEMINJAMAN

- ✓ NoPinjam
- ✓ NIM
- ✓ Nama
- ✓ TanggalPinjam
- ✓ TanggalKembali

DETAIL_PEMINJAMAN

- ✓ NoPinjam
- ✓ KodeBuku
- ✓ JudulBuku
- ✓ Penulis



Gambar 3. 27 Proses Normalisasi 1NF

C. *Second Normal Form* (2NF)

Semua atribut non-key harus bergantung penuh pada primary key.

Informasi buku seperti judul_buku tidak bergantung pada kombinasi transaksi, tetapi bergantung pada kode_buku.

Maka dipisahkan:

BUKU

- ✓ KodeBuku
- ✓ JudulBuku
- ✓ IdKategori
- ✓ IdPenerbit

dan:

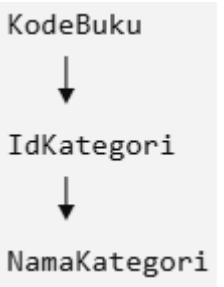
DETAIL_PEMINJAMAN

- ✓ NoPinjam
- ✓ KodeBuku

D. *Third Normal Form* (3NF)

Tidak boleh terdapat ketergantungan transitif.

Contoh:



Maka kategori dipisahkan:

KATEGORI

- ✓ IdKategori
- ✓ NamaKategori

Hasil akhirnya menjadi:

- ✓ ANGGOTA
- ✓ KATEGORI
- ✓ PENERBIT
- ✓ PENULIS
- ✓ BUKU
- ✓ BUKU_PENULIS
- ✓ PEMINJAMAN
- ✓ DETAIL_PEMINJAMAN
- ✓ PENGEMBALIAN
- ✓ DENDA

3.21.7 Struktur Tabel

Berikut rancangan struktur tabel utama.

A. Tabel Anggota

Field	Tipe	Constraint	Keterangan
id_anggota	INT	PK, AI	Identitas anggota
nomor_anggota	VARCHAR(20)	UNIQUE	Nomor anggota
nama	VARCHAR(100)	NOT NULL	Nama anggota
alamat	VARCHAR(255)	-	Alamat
no_telp	VARCHAR(20)	-	Nomor telepon
email	VARCHAR(100)	UNIQUE	Email
status	VARCHAR(20)	DEFAULT	Status anggota

B. Tabel Buku

Field	Tipe	Constraint	Keterangan
id_buku	INT	PK, AI	Identitas buku
isbn	VARCHAR(20)	UNIQUE	ISBN
judul	VARCHAR(200)	NOT NULL	Judul
tahun_terbit	YEAR	-	Tahun terbit
stok	INT	CHECK	Jumlah stok
id_kategori	INT	FK	Kategori
id_penerbit	INT	FK	Penerbit

C. Tabel Peminjaman

Field	Tipe	Constraint
id_peminjaman	BIGINT	PK, AI
id_anggota	INT	FK
tanggal_pinjam	DATE	NOT NULL
tanggal_jatuh_tempo	DATE	NOT NULL
status	VARCHAR(20)	DEFAULT

D. Tabel Detail Peminjaman

Field	Tipe	Constraint
id_detail	BIGINT	PK, AI
id_peminjaman	BIGINT	FK
id_buku	INT	FK
jumlah	INT	CHECK

3.21.8 Kamus Data

Kamus data menjelaskan struktur setiap atribut yang digunakan dalam database.

Tabel 3. 2 Kamus Data Anggota

Nama Field	Tipe	Panjang	Key	Null	Keterangan
id_anggota	INT	11	PK	Tidak	ID anggota
nomor_anggota	VARCHAR	20	UK	Tidak	Nomor anggota
nama	VARCHAR	100	-	Tidak	Nama anggota
alamat	VARCHAR	255	-	Ya	Alamat
no_telp	VARCHAR	20	-	Ya	Telepon
email	VARCHAR	100	UK	Ya	Email

Nama Field	Tipe	Panjang	Key	Null	Keterangan
status	VARCHAR	20	-	Tidak	Status

Tabel 3. 3 Kamus Data Buku

Nama Field	Tipe	Panjang	Key	Keterangan
id_buku	INT	11	PK	ID buku
isbn	VARCHAR	20	UK	ISBN
judul	VARCHAR	200	-	Judul buku
tahun_terbit	YEAR	4	-	Tahun terbit
stok	INT	-	-	Stok buku
id_kategori	INT	-	FK	Kategori
id_penerbit	INT	-	FK	Penerbit

3.21.9 Implementasi Database

Database dapat dibuat dengan MySQL.

A. Membuat Database

```
CREATE DATABASE perpustakaan;
USE perpustakaan;
```

B. Tabel Kategori

```
CREATE TABLE kategori (
  id_kategori INT AUTO_INCREMENT,
  nama_kategori VARCHAR(100) NOT NULL,

  PRIMARY KEY (id_kategori),
  UNIQUE (nama_kategori)
) ENGINE=InnoDB;
```

C. Tabel Penerbit

```
CREATE TABLE penerbit (
  id_penerbit INT AUTO_INCREMENT,
  nama_penerbit VARCHAR(150) NOT NULL,
  alamat VARCHAR(255),
  email VARCHAR(100),
  PRIMARY KEY (id_penerbit)
) ENGINE=InnoDB;
```

D. Tabel Anggota

```
CREATE TABLE anggota
(
  id_anggota INT AUTO_INCREMENT,
  nomor_anggota VARCHAR(20) NOT NULL,
  nama VARCHAR(100) NOT NULL,
  alamat VARCHAR(255),
  no_telp VARCHAR(20),
  email VARCHAR(100),
  status VARCHAR(20) NOT NULL DEFAULT 'Aktif',
  PRIMARY KEY (id_anggota),
  UNIQUE (nomor_anggota),
  UNIQUE (email),
  CHECK (status IN ('Aktif', 'Nonaktif'))
) ENGINE=InnoDB;
```

E. Tabel Buku

```
CREATE TABLE buku (
```

```
  id_buku INT AUTO_INCREMENT,
  isbn VARCHAR(20) NOT NULL,
  judul VARCHAR(200) NOT NULL,
  tahun_terbit YEAR,
  stok INT NOT NULL DEFAULT 0,
  id_kategori INT NOT NULL,
  id_penerbit INT NOT NULL,
  PRIMARY KEY (id_buku),
  UNIQUE (isbn),
  FOREIGN KEY (id_kategori)
    REFERENCES kategori(id_kategori)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  FOREIGN KEY (id_penerbit)
    REFERENCES penerbit(id_penerbit)
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
  CHECK (stok >= 0)
) ENGINE=InnoDB;
```

F. Tabel Penulis

```
CREATE TABLE penulis (
  id_penulis INT AUTO_INCREMENT,
  nama_penulis VARCHAR(150) NOT NULL,
  PRIMARY KEY (id_penulis)
) ENGINE=InnoDB;
```

G. Tabel Buku_Penulis

Karena hubungan buku dan penulis adalah **many-to-many**, diperlukan tabel penghubung.

```
CREATE TABLE buku_penulis (
  id_buku INT NOT NULL,
  id_penulis INT NOT NULL,
  PRIMARY KEY (id_buku, id_penulis),
  FOREIGN KEY (id_buku)
    REFERENCES buku(id_buku)
    ON DELETE CASCADE,
  FOREIGN KEY (id_penulis)
    REFERENCES penulis(id_penulis)
    ON DELETE CASCADE
) ENGINE=InnoDB;
```

H. Tabel Peminjaman

```
CREATE TABLE peminjaman (
  id_peminjaman BIGINT AUTO_INCREMENT,
  id_anggota INT NOT NULL,
  tanggal_pinjam DATE NOT NULL,
  tanggal_jatuh_tempo DATE NOT NULL,
  status VARCHAR(20) NOT NULL DEFAULT 'Dipinjam',
  PRIMARY KEY (id_peminjaman),
  FOREIGN KEY (id_anggota)
    REFERENCES anggota(id_anggota)
    ON UPDATE CASCADE
```

```
    ON DELETE RESTRICT,
  CHECK (
    status IN ('Dipinjam', 'Selesai', 'Terlambat'))
) ENGINE=InnoDB;
```

I. Tabel Detail Peminjaman

```
CREATE TABLE detail_peminjaman (
  id_detail BIGINT AUTO_INCREMENT,
  id_peminjaman BIGINT NOT NULL,
  id_buku INT NOT NULL,
  jumlah INT NOT NULL DEFAULT 1,
  PRIMARY KEY (id_detail),
  FOREIGN KEY (id_peminjaman)
    REFERENCES peminjaman(id_peminjaman)
    ON DELETE CASCADE,
  FOREIGN KEY (id_buku)
    REFERENCES buku(id_buku)
    ON DELETE RESTRICT,
  CHECK (jumlah > 0),
  UNIQUE (id_peminjaman, id_buku)
) ENGINE=InnoDB;
```

J. Tabel Pengembalian

```
CREATE TABLE pengembalian (
  id_pengembalian BIGINT AUTO_INCREMENT,
  id_peminjaman BIGINT NOT NULL,
  tanggal_kembali DATE NOT NULL,
```

```
  terlambat_hari INT NOT NULL DEFAULT 0,
  PRIMARY KEY (id_pengembalian),
  UNIQUE (id_peminjaman),
  FOREIGN KEY (id_peminjaman)
    REFERENCES peminjaman(id_peminjaman)
    ON DELETE RESTRICT,
  CHECK (terlambat_hari >= 0)
) ENGINE=InnoDB;
```

K. Tabel Denda

```
CREATE TABLE denda (
  id_denda BIGINT AUTO_INCREMENT,
  id_pengembalian BIGINT NOT NULL,
  jumlah_denda DECIMAL(12,2) NOT NULL
DEFAULT 0,
  status_pembayaran VARCHAR(20)
    NOT NULL DEFAULT 'Belum Lunas',
  PRIMARY KEY (id_denda),
  UNIQUE (id_pengembalian),
  FOREIGN KEY (id_pengembalian)
    REFERENCES pengembalian(id_pengembalian)
    ON DELETE CASCADE,
  CHECK (jumlah_denda >= 0),
  CHECK (
    status_pembayaran IN
    ('Belum Lunas', 'Lunas'))
) ENGINE=InnoDB;
```

3.21.10 Contoh Data

Data Kategori

```
INSERT INTO kategori (nama_kategori)
VALUES
('Teknologi Informasi'),
('Sistem Informasi'),
('Manajemen'),
('Novel');
```

Data Anggota

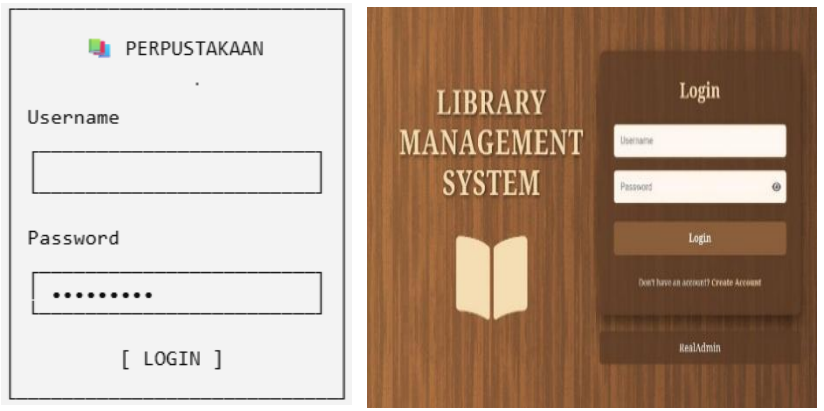
```
INSERT INTO anggota
(nomor_anggota, nama, alamat, no_telp, email)
VALUES
('AGT001', 'Andi Pratama','Malang','081234567890',
'andi@email.com');
```

Data Buku

```
INSERT INTO buku
(isbn, judul, tahun_terbit, stok, id_kategori, id_penerbit)
VALUES('9781234567890','Analisis dan Perancangan
Sistem Informasi',2025,5,2,1);
```

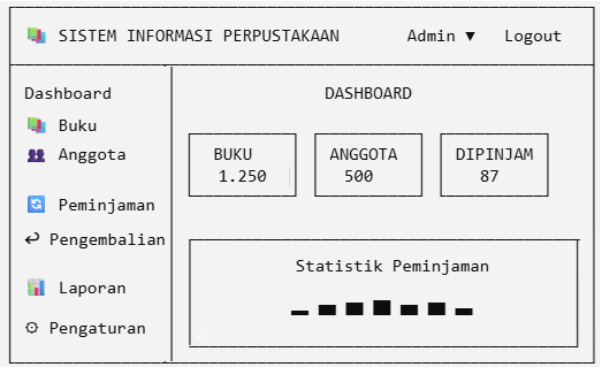
3.21.11 Rancangan Antarmuka Utama

Halaman Login



Gambar 3. 28 Rancangan Login

3.21.12 Dashboard Utama



Gambar 3. 29 Dashboard Perpustakaan

3.21.13 Halaman Data Buku

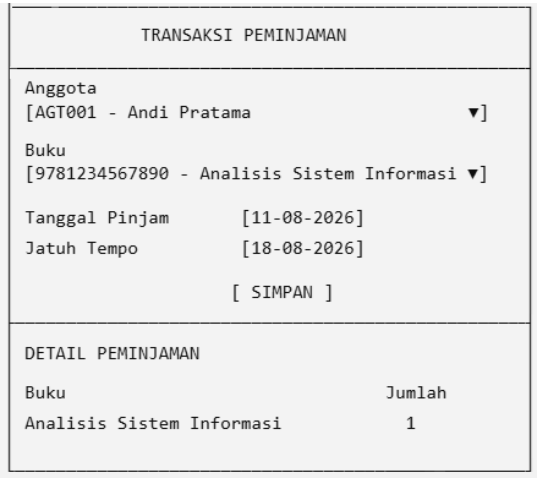
DATA BUKU [Cari Judul/ISBN]

[+ Tambah Buku]

ISBN	Judul	Stok	Aksi
9781	Analisis Sistem	5	Edit | Hapus
9782	Basis Data	3	Edit | Hapus
9783	Pemrograman Web	8	Edit | Hapus

Gambar 3. 30 Antarmuka Data Buku

3.21.14 Halaman Peminjaman



Gambar 3. 31 Antarmuka Transaksi Peminjaman

3.21.15 Halaman Pengembalian

Pada proses pengembalian, sistem dapat menghitung keterlambatan secara otomatis.

Contoh:

Tanggal Pinjam : 01-08-2026

Jatuh Tempo : 08-08-2026

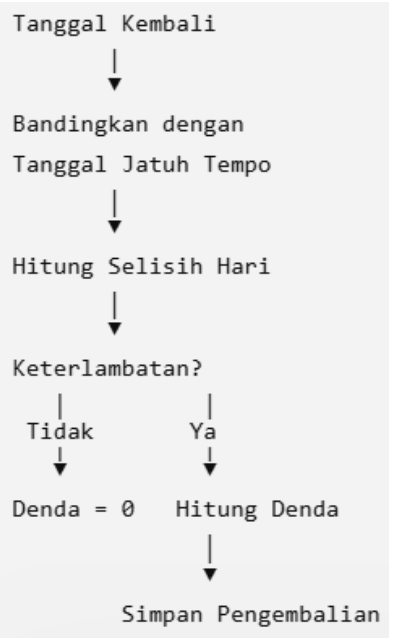
Tanggal Kembali: 11-08-2026

Keterlambatan : 3 hari

Denda per hari : Rp2.000

Total Denda : Rp6.000

Secara konseptual:



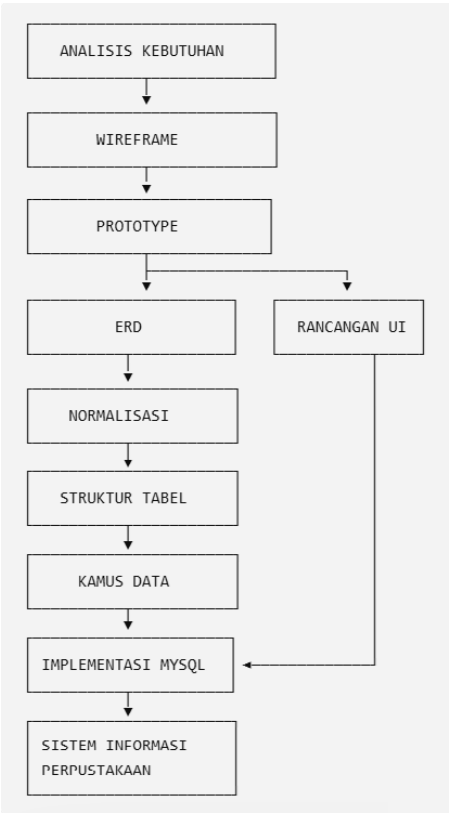
3.21.16 Halaman Laporan



Gambar 3. 32 Rancangan Laporan

3.21.17 Integrasi Seluruh Hasil Perancangan

Studi kasus ini memperlihatkan bahwa setiap tahapan perancangan saling berhubungan.



Gambar 3. 33 Alur Lengkap Perancangan Sistem Informasi Perpustakaan

3.21.18 Contoh Alur Operasional Sistem

Sebagai contoh, seorang anggota ingin meminjam buku.

1. Anggota login

   ↓

2. Sistem memverifikasi akun

↓

3. Anggota mencari buku

↓

4. Sistem menampilkan stok

↓

5. Petugas memproses peminjaman

↓

6. Sistem mencatat transaksi

↓

7. Stok buku berkurang

↓

8. Sistem menentukan tanggal jatuh tempo

↓

9. Buku dikembalikan

↓

10. Sistem menghitung keterlambatan

↓

11. Sistem menghitung denda

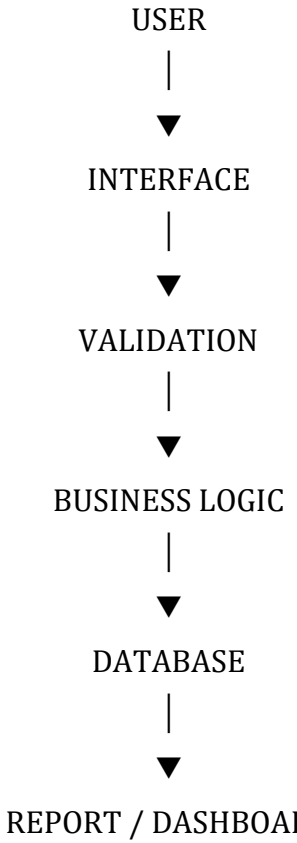
↓

12. Stok buku bertambah

↓

13. Laporan diperbarui

Dengan demikian, sebuah aktivitas sederhana seperti **peminjaman buku** sebenarnya melibatkan banyak komponen sistem:



3.21.19 Kesimpulan Studi Kasus

Studi kasus Sistem Informasi Perpustakaan menunjukkan bagaimana berbagai konsep yang telah dibahas pada subbab sebelumnya dapat diterapkan secara terpadu.

Tahap pertama adalah **analisis kebutuhan** untuk menentukan aktor, kebutuhan fungsional, dan

kebutuhan nonfungsional. Hasil analisis kemudian digunakan sebagai dasar pembuatan **wireframe** dan **prototype** untuk menggambarkan rancangan antarmuka.

Selanjutnya, kebutuhan data diterjemahkan menjadi **ERD** yang menggambarkan entitas, atribut, dan hubungan antarentitas. ERD kemudian dinormalisasi untuk mengurangi redundansi dan menghasilkan struktur relasional yang lebih baik.

Hasil normalisasi diterjemahkan menjadi **struktur tabel** yang dilengkapi dengan primary key, foreign key, constraint, index, unique, default value, dan check constraint. Selanjutnya, **kamus data** digunakan untuk mendokumentasikan setiap atribut secara sistematis.

Tahap berikutnya adalah implementasi rancangan ke dalam **MySQL**. Database yang telah dibuat kemudian diintegrasikan dengan antarmuka melalui mekanisme CRUD, validasi, autentikasi, dan proses bisnis. Dengan demikian, rangkaian perancangan dapat dirumuskan sebagai:

Kebutuhan → Wireframe → Prototype → ERD → Normalisasi → Struktur Tabel → Kamus Data → Implementasi Database → Antarmuka → Sistem Informasi Perpustakaan.

Studi kasus ini sekaligus memperlihatkan bahwa perancangan sistem informasi bukan hanya kegiatan membuat database atau tampilan aplikasi secara terpisah, melainkan proses terpadu yang menghubungkan kebutuhan pengguna, proses bisnis, data, antarmuka, dan teknologi implementasi.

DAFTAR PUSTAKA

Benyon, D. (2019). Designing user experience: A guide to HCI, UX and interaction design (4th ed.). Pearson.

Connolly, T., & Begg, C. (2015). Database systems: A practical approach to design, implementation, and management (6th ed.). Pearson.

Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). About face: The essentials of interaction design (4th ed.). Wiley.

Coronel, C., & Morris, S. (2019). Database systems: Design, implementation, & management (13th ed.). Cengage Learning.

Date, C. J. (2019). Database design and relational theory (2nd ed.). O'Reilly Media.

Elmasri, R., & Navathe, S. B. (2016). Fundamentals of database systems (7th ed.). Pearson.

Fathansyah. (2020). Basis data (Edisi revisi). Informatika Bandung.

Harrington, J. L. (2016). Relational database design and implementation (4th ed.). Morgan Kaufmann.

International Organization for Standardization. (2018). ISO 9241-11:2018: Ergonomics of human-system interaction—Part 11: Usability: Definitions and concepts. ISO.

International Organization for Standardization. (2019). ISO 9241-210:2019: Ergonomics of human-system interaction—Part 210: Human-centred design for interactive systems. ISO.

Jaspers, M. W. M., Steen, T., van den Bos, C., & Geenen, M. (2004). The think aloud method: A guide to user

interface design. International Journal of Medical Informatics, 73(11–12), 781–793.

Kendall, K. E., & Kendall, J. E. (2019). Systems analysis and design (10th ed.). Pearson.

Kroenke, D. M., & Auer, D. J. (2018). Database processing: Fundamentals, design, and implementation (15th ed.). Pearson.

Krug, S. (2014). Don't make me think, revisited: A common sense approach to web usability (3rd ed.). New Riders.

Lemahieu, W., vanden Broucke, S., & Baesens, B. (n.d.). Principles of database management: The practical guide to storing, managing and analyzing big and small data. Cambridge University Press.

MySQL. (2026). MySQL 8.4 reference manual. Oracle Corporation.

Nielsen, J. (1993). Usability engineering. Academic Press.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 152–158). Association for Computing Machinery.

Nielsen Norman Group. (n.d.). 10 usability heuristics for user interface design.

Norman, D. A. (2013). The design of everyday things: Revised and expanded edition. Basic Books.

Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill Education.

Ramakrishnan, R., & Gehrke, J. (2003). Database management systems (3rd ed.). McGraw-Hill.

Rubin, J., & Chisnell, D. (2008). Handbook of usability testing: How to plan, design, and conduct effective tests. Wiley.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). Designing the user interface: Strategies for effective human-computer interaction (6th ed.). Pearson.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database system concepts (7th ed.). McGraw-Hill Education.

Sommerville, I. (2016). Software engineering (10th ed.). Pearson.

Teorey, T. J., Lightstone, S., Nadeau, T., & Jagadish, H. V. (2011). Database modeling and design: Logical design (5th ed.). Morgan Kaufmann.

Tidwell, J., Brewer, C., & Valencia, A. (2020). Designing interfaces: Patterns for effective interaction design (3rd ed.). O'Reilly Media.

W3C. (2023). Web content accessibility guidelines (WCAG) 2.2.

Watt, A., Eng, N., & Strickland, F. (n.d.). Database design (3rd ed.). BCcampus Open Textbook.

Yakub. (2012). Pengantar sistem informasi. Graha Ilmu.

BAB 4

PENGUJIAN SISTEM DAN IMPLEMENTASI

Oleh Heliawati Hamrul

Pengujian dan implementasi sistem merupakan dua tahapan penting dalam pengembangan sistem informasi. Keduanya saling berkaitan untuk memastikan sistem siap digunakan dan mampu memberikan manfaat bagi organisasi. Pengujian sistem merupakan proses mengevaluasi sistem yang telah lengkap dan terintegrasi untuk memastikan bahwa sistem memenuhi persyaratan serta berfungsi sesuai dengan kebutuhan pengguna. Pengujian dilakukan melalui berbagai kasus uji untuk memeriksa fungsi, kinerja, dan kesesuaian sistem sekaligus menemukan cacat sebelum sistem digunakan. Sementara itu, implementasi sistem merupakan proses menerapkan sistem yang telah diuji ke lingkungan operasional agar dapat digunakan oleh pengguna akhir. Implementasi mencakup instalasi, konfigurasi, pelatihan pengguna, transisi dari sistem lama ke sistem baru, serta pemeliharaan dan pengembangan sistem secara berkelanjutan.

Pengujian dan implementasi memiliki peran strategis dalam menjamin kualitas sistem informasi. Pengujian memberikan keyakinan bahwa sistem telah bebas dari cacat kritis, memenuhi persyaratan, dan sesuai dengan tujuan

penggunaannya. Kualitas hasil pengujian pada akhirnya berpengaruh langsung terhadap keandalan, keamanan, kinerja, dan kepuasan pengguna terhadap sistem. Namun, sistem yang telah berhasil melewati pengujian belum tentu memberikan manfaat apabila proses implementasinya tidak dilakukan dengan benar. Implementasi memastikan bahwa sistem yang telah diuji dapat dipasang, dikonfigurasi, dioperasikan, dan digunakan secara tepat dalam lingkungan organisasi. Oleh sebab itu, pengujian dan implementasi perlu dipahami sebagai rangkaian proses yang tidak terpisahkan dalam menghasilkan sistem informasi yang berkualitas sekaligus mampu mendukung pencapaian tujuan organisasi.

4.1 Pengujian Sistem

4.1.1 Konsep Dasar Pengujian Sistem

Bayangkan sebuah pesawat baru yang akan digunakan untuk mengangkut penumpang. Sebelum dinyatakan layak terbang, pesawat tersebut harus melalui berbagai pengujian untuk memastikan mesin, sistem navigasi, keamanan, dan komponen lainnya dapat berfungsi dengan baik. Hal yang sama dilakukan ketika seseorang membeli mobil baru melalui *test drive* untuk memastikan kendaraan bekerja sesuai harapan. Dalam bidang kesehatan, obat baru juga harus melalui serangkaian pengujian sebelum dapat digunakan oleh masyarakat. Ilustrasi tersebut menunjukkan bahwa suatu produk perlu diuji terlebih dahulu sebelum digunakan untuk memastikan bahwa produk tersebut aman, berfungsi dengan baik, dan sesuai dengan tujuan penggunaannya.

Prinsip yang sama berlaku pada perangkat lunak dan sistem informasi. Sistem yang telah dikembangkan tidak dapat langsung dianggap benar hanya karena proses pemrogramannya telah selesai. Semakin kompleks suatu perangkat lunak, semakin sulit memastikan perilakunya hanya dengan memeriksa kode atau rancangan secara langsung. Kesalahan dapat muncul karena kekeliruan manusia, kesalahpahaman terhadap kebutuhan pengguna, maupun kelemahan dalam proses pengembangan. Oleh karena itu, pengujian sistem diperlukan untuk menemukan kesalahan atau cacat yang menyebabkan sistem tidak berfungsi sebagaimana mestinya.

Pengujian sistem adalah proses sistematis untuk memeriksa dan mengevaluasi perangkat lunak atau sistem informasi dengan menggunakan kondisi dan data tertentu. Proses ini bertujuan untuk menemukan kesalahan serta memastikan bahwa sistem bekerja sesuai dengan kebutuhan dan rancangan yang ditetapkan. Pengujian merupakan bagian penting dari proses penghapusan cacat perangkat lunak, bersama dengan teknik lain seperti inspeksi, pembuatan prototipe, dan pemeriksaan sintaks secara otomatis.

Tujuan utama pengujian sistem adalah menemukan kesalahan sedini mungkin, memastikan kesesuaian sistem dengan kebutuhan pengguna, meningkatkan kualitas dan keandalan sistem, serta mengurangi risiko kegagalan ketika sistem digunakan. Pengujian juga perlu dilakukan sejak tahap awal pengembangan, bukan hanya setelah sistem selesai dibangun. Hal ini karena semakin lama suatu kesalahan tidak terdeteksi, semakin besar biaya dan usaha yang diperlukan untuk memperbaikinya. Kesalahan pada tahap kebutuhan, misalnya, dapat diperbaiki dengan relatif mudah ketika ditemukan sejak awal. Sebaliknya, apabila kesalahan

tersebut baru ditemukan setelah sistem digunakan, perbaikannya dapat memerlukan perubahan besar, bahkan menyebabkan sebagian sistem harus dibangun ulang. Oleh karena itu, pengujian bukan sekadar tahap untuk mencari kesalahan, tetapi merupakan upaya untuk mencegah kesalahan berkembang menjadi masalah yang lebih besar, mahal, dan berdampak pada pengguna maupun organisasi.

4.1.2 Prinsip-Prinsip Pengujian

Pengujian sistem merupakan proses yang dilakukan secara terarah untuk memastikan bahwa perangkat lunak memiliki kualitas yang sesuai dengan kebutuhan dan dapat berfungsi sebagaimana yang diharapkan. Berikut ini adalah prinsip dalam pengujian sistem informasi:

1. Deteksi Cacat dan Jaminan Kualitas

   Prinsip utama pengujian sistem adalah mendeteksi dan mengidentifikasi *defect, error,* atau *bug* yang terdapat dalam perangkat lunak. Selain menemukan kesalahan, pengujian juga digunakan untuk mengevaluasi kualitas, akurasi, dan efisiensi sistem perangkat lunak. Proses ini membantu memastikan bahwa perangkat lunak tidak hanya dapat berjalan, tetapi juga memenuhi kebutuhan pengguna dan persyaratan fungsional yang telah ditetapkan (Gaur dkk., 2016)

2. Verifikasi dan Validasi

   Verifikasi berkaitan dengan pemeriksaan apakah perangkat lunak telah memenuhi spesifikasi yang ditetapkan. Proses ini memastikan kesesuaian perangkat lunak terhadap spesifikasi yang menjadi dasar pengembangannya. Sementara itu, validasi berfokus pada apakah perangkat lunak benar-benar

mampu memenuhi tujuan penggunaannya. Kedua aspek tersebut saling melengkapi dalam pengujian. Oleh karena itu, pengujian tidak hanya mempertanyakan apakah sistem telah dibangun sesuai spesifikasi, tetapi juga apakah sistem tersebut dapat memenuhi tujuan yang dimaksudkan.

3. Proses yang Sistematis dan Metodis

   Pengujian dilakukan sebagai suatu proses yang sistematis dan bertahap *(step-by-step).* Pendekatan ini diperlukan agar berbagai jenis cacat dapat dideteksi secara efisien dengan penggunaan waktu dan sumber daya yang optimal. Dengan proses yang terstruktur, penguji dapat melakukan pengujian secara lebih efisien dalam menemukan berbagai kemungkinan cacat pada perangkat lunak (Jin & Zeng, 2011).

4. Manajemen Risiko

   Manajemen Risiko dalam pengujian sistem informasi merupakan proses identifikasi sejak dini permasalahan yang berpotensi muncul sehingga masalah tersebut dapat dicegah sebelum perangkat lunak diimplementasikan dan digunakan oleh penggunanya. Prinsip manajemen risiko menunjukkan bahwa kegiatan pengujian berkontribusi terhadap upaya menjaga sistem agar tetap andal dan memiliki kualitas yang sesuai sebelum diimplementasikan (Panagiotis Leloudas, 2025) .

4.1.3 Strategi dan Level Pengujian

Strategi pengujian sistem dapat dipahami sebagai pendekatan yang digunakan untuk menguji perangkat lunak secara bertahap, mulai dari komponen terkecil

hingga keseluruhan sistem dan penerimaan pengguna. Pengujian terdiri atas level *Unit Testing, Integration Testing, System Testing, User Acceptance Testing* (UAT). Keempat jenis pengujian tersebut memiliki fokus, tujuan, waktu pelaksanaan, serta pihak yang melaksanakannya masing-masing. Strategi dan level pengujian membentuk rangkaian pengujian yang berkembang dari lingkup yang paling kecil menuju lingkup yang lebih luas.



Gambar 4. 1 *Workflow* Pengujian Sistem

1. *Unit Testing* merupakan pengujian terhadap unit atau komponen individual perangkat lunak, umumnya pada tingkat kode atau modul. Pengujian ini dilakukan pada tahap awal pengembangan dengan tujuan memastikan bahwa setiap unit dapat berfungsi dengan benar dan sesuai dengan rancangan. Fokus utamanya adalah menemukan kesalahan logika dan kesalahan fungsional sedini mungkin. Karena pengujian dilakukan terhadap komponen yang masih terisolasi, pengembang dapat lebih mudah mengidentifikasi bagian sistem yang mengalami masalah sebelum komponen tersebut digabungkan dengan komponen lainnya (Chandrasekaran dkk., 2019).
2. *Integration Testing*, dilakukan setelah unit-unit dinyatakan berfungsi dengan benar. Pengujian integrasi dilakukan terhadap kombinasi unit atau modul untuk menemukan permasalahan pada

antarmuka dan interaksi di antara unit-unit yang telah digabungkan. Dengan demikian, cakupan pengujian mulai diperluas dari komponen individual menuju hubungan antarkomponen (Tan dkk., 2022).

3. *System Testing*, yaitu pengujian terhadap sistem yang telah terintegrasi secara lengkap untuk memastikan bahwa sistem secara keseluruhan memenuhi persyaratan yang telah ditentukan. Berbeda dengan unit testing yang berfokus pada bagian individual dan integration testing yang berfokus pada interaksi antarmodul, *system testing* mengevaluasi keseluruhan sistem dan dapat menggunakan skenario yang mensimulasikan kondisi dunia nyata (Guy Wingate, 2010).
4. *User Acceptance Testing* (UAT) dilakukan untuk menentukan apakah sistem telah memenuhi kebutuhan pengguna dan kriteria penerimaan. UAT dilakukan oleh atau untuk pengguna akhir sehingga hasilnya menjadi dasar untuk memastikan bahwa perangkat lunak telah memenuhi kebutuhan pengguna dan siap untuk diterapkan.

Perbedaan masing-masing pengujian diuraikan dalam tabel 4.1.

Tabel 4. 1 Level Pengujian

Jenis Pengujian	Area Fokus	Tujuan Utama	Waktu Pelaksanaan
Unit Testing	Unit/modul yang telah digabungkan	Memastikan setiap unit berfungsi dengan benar	Tahap awal pengembangan

Jenis Pengujian	Area Fokus	Tujuan Utama	Waktu Pelaksanaan
Integration Testing	Keseluruhan sistem	Mendeteksi kesalahan pada antarmuka dan interaksi antarsatuan yang telah diintegrasikan	Setelah *unit testing*
System Testing	Kebutuhan pengguna	Memvalidasi sistem secara keseluruhan terhadap persyaratan	Setelah *integration testing*
User Acceptance Testing	Kode/sistem yang mengalami perubahan	Memastikan perangkat lunak memenuhi kebutuhan pengguna akhir dan siap diterapkan	Setelah *system testing*

4.1.4 Metode dan Teknik Pengujian Sistem

Pengujian secara efektif dan efisien dilakukan dengan berbagai metode dan teknik tertentu. Pemilihan teknik yang tepat sangat bergantung pada aspek sistem yang ingin diuji, ketersediaan dokumentasi, serta sejauh mana akses penguji terhadap struktur kode program di dalam sistem. Subbab berikut akan membahas ketiga

metode tersebut beserta teknik-teknik perancang skenario uji yang sering digunakan dalam pengembangan sistem informasi.

Pengujian Black-Box

Pengujian *Black-Box* merupakan teknik pengujian perangkat lunak yang dilakukan dengan mengevaluasi fungsi aplikasi tanpa memerlukan pengetahuan mengenai struktur kode sumber (*source code*)Pengujian berfokus pada input yang diberikan dan output yang dihasilkan. Pengujian *Black-Box* menempatkan fungsi yang dapat diamati dari perangkat lunak sebagai fokus utama pengujian.

Terdapat tiga karakteristik utama dalam pengujian *Black-Box* yaitu;

1. Tidak memerlukan akses terhadap kode sumber sehingga penguji tidak harus memahami atau melihat cara kerja internal perangkat lunak.
2. Pengujian berfokus pada fungsionalitas, yaitu memeriksa apakah sistem telah memenuhi persyaratan fungsional sebagaimana dijelaskan dalam spesifikasi.
3. Pengujian bersifat berbasis *input* dan *output*, sehingga kasus uji dirancang berdasarkan kemungkinan input yang diberikan kepada sistem dan output yang diharapkan. Dengan demikian, keberhasilan pengujian ditentukan dengan membandingkan perilaku aktual perangkat lunak terhadap perilaku yang diharapkan berdasarkan spesifikasi.

Pengujian *Black-Box* dilakukan melalui beberapa tahapan yang terstruktur yang terdiri dari:

1. Tahap pertama adalah memahami spesifikasi dengan meninjau kebutuhan dan spesifikasi

perangkat lunak untuk menentukan bagian yang perlu diuji.

2. Tahap kedua adalah mengidentifikasi skenario pengujian dengan menentukan kategori, pilihan, dan batas nilai input yang memungkinkan, termasuk penggunaan *Boundary Value Analysis.*
3. Tahap ketiga adalah merancang kasus uji berdasarkan skenario yang telah ditentukan dengan menekankan output yang diharapkan dari setiap input.
4. Tahap keempat adalah menjalankan pengujian dengan menjalankan kasus uji pada perangkat lunak dan mengamati output yang dihasilkan.
5. Tahap kelima adalah membandingkan hasil, yaitu memeriksa kesesuaian antara output aktual dan output yang diharapkan serta melaporkan perbedaan sebagai *defect.*

Beberapa teknik yang umum digunakan dalam pengujian *Black-Box*. *Boundary Value Analysis (BVA), Equivalence Partitioning, Decision Table Testing, Use Case Testing* yang dijelaskan secara detail pada pembahasan selanjutnya.

Boundary Value Analysis (BVA)

BVA merupakan teknik pengujian perangkat lunak yang berfokus pada pembuatan kasus uji menggunakan nilai input yang berada pada batas-batas *(boundaries)* rentang input. Fokus tersebut didasarkan pada pengamatan bahwa perilaku perangkat lunak lebih mungkin mengalami perubahan pada batas rentang input dan kesalahan lebih sering muncul pada area tersebut. Kasus uji dirancang menggunakan nilai minimum dan maksimum yang valid, nilai yang berada tepat di bawah dan tepat di atas batas, serta dalam

beberapa kondisi dapat menggunakan nilai tengah *(midpoint)* sebagai pembanding. Fokus pada batas menjadi penting karena perangkat lunak dapat mengalami kesalahan seperti *off-by-one error,* kesalahan logika kondisi, maupun kegagalan dalam menangani kondisi yang berada di ujung rentang input. Sebagai contoh, apabila suatu sistem menetapkan nilai maksimum 100, maka pengujian tidak cukup hanya menggunakan nilai yang berada jauh di bawah 100. Nilai 100, nilai tepat di bawahnya, dan nilai tepat di atasnya perlu diperhatikan untuk mengetahui apakah sistem benar-benar membedakan nilai yang masih valid dengan nilai yang sudah berada di luar batas.

Penggunaan BVA memberikan beberapa keuntungan dalam proses pengujian diantara nya :

1. Kemampuan deteksi cacat yang tinggi pada area batas, karena secara khusus mengarahkan pengujian pada bagian input yang berpotensi menimbulkan kesalahan.
2. Relatif efisien dalam pemilihan kasus uji, karena dapat mengurangi jumlah kasus uji dibandingkan dengan pengujian secara menyeluruh dengan tetap mempertahankan tingkat deteksi cacat yang tinggi.
3. Membantu meningkatkan cakupan pengujian dengan memastikan bahwa perubahan perilaku sistem pada titik-titik kritis telah diuji.

Contoh pengujian BVA diuraikan dalam Tabel 4.2 :

Tabel 4. 2 Contoh Pengujian BVA

Jenis Cacat	Deskripsi	Contoh
Off-by-one errors	Kesalahan yang terjadi ketika nilai batas secara tidak tepat dimasukkan atau dikeluarkan,	Gagal pada n=0, n=1, n=max

Jenis Cacat	Deskripsi	Contoh
	misalnya penggunaan < dibandingkan <=	
Incorrect handling of limits	Kegagalan ketika perangkat lunak tidak memproses nilai minimum, maksimum, atau nilai tepat di luar batas dengan benar	Menerima n=101 ketika maksimum adalah 100
Edge case failures	Cacat yang hanya muncul ketika input berada pada atau dekat batas rentang input yang valid	Sistem mengalami *crash* pada input batas
Negative testing failures	Masalah yang muncul ketika input tidak valid atau berada di luar rentang tidak ditangani sesuai harapan	Tidak menghasilkan *error* untuk n=-1
Domain partitioning errors	Masalah ketika perangkat lunak tidak membedakan subdomain input dengan benar	Output salah pada batas antarsubdomain

Namun, penerapan BVA secara manual dapat menjadi lebih memakan waktu ketika sistem memiliki domain input yang kompleks. Oleh karena itu, pengembangan *machine learning* dan *automated tools* dapat digunakan untuk membantu mengotomatisasi proses BVA.

Pengujian *Decision Table*

Pengujian *Decision Table* merupakan teknik pengujian perangkat lunak yang sistematis untuk menghasilkan kasus uji berdasarkan kebutuhan sistem dengan mengidentifikasi seluruh kemungkinan kombinasi kondisi *(input)* dan tindakan *(output).* Teknik ini sangat berguna pada sistem yang memiliki beberapa kondisi input yang dapat menghasilkan perilaku atau keluaran sistem yang berbeda (Joosten dkk., 2020). Berikut ini adalah urutan tahapan pengujian *Decision Table*:

1. Mengekstraksi seluruh kondisi *(input)* dan tindakan *(output)* yang relevan dari kebutuhan perangkat lunak.
2. Membuat tabel dengan kondisi dan kemungkinan kombinasi sebagai dasar untuk menggambarkan berbagai skenario *input*.
3. Menentukan tindakan atau hasil yang diharapkan untuk setiap kombinasi kondisi.
4. Menyederhanakan aturan dengan menggabungkan aturan yang serupa atau menghilangkan kombinasi yang tidak mungkin terjadi untuk mengurangi jumlah kasus uji tanpa kehilangan cakupan.
5. Menggunakan setiap baris pada *decision table* sebagai dasar untuk membuat kasus uji yang akan dieksekusi.

Contoh pengujian *Decision Table* pada sistem login dengan dua kondisi, yaitu *username* valid dan *password* valid. Masing-masing kondisi memiliki dua kemungkinan nilai, yaitu *Yes* atau *No*. Dari kedua kondisi tersebut terbentuk empat kemungkinan kombinasi. Contoh ini menunjukkan bagaimana kombinasi kondisi input dapat digunakan untuk menentukan tindakan atau hasil yang diharapkan dari sistem.

Tabel 4. 3 Contoh Pengujian *Decision Table*

Condition 1 (Username)	***Condition 2 (Password)***	***Action (Result)***
Yes	*Yes*	Login successful
Yes	*No*	Login failed
No	*Yes*	Login failed
No	*No*	Login failed

Berdasarkan tabel tersebut, setiap baris merepresentasikan satu kasus uji yang perlu dieksekusi. Baris pertama menguji kondisi ketika *username* dan *password* valid sehingga hasil yang diharapkan adalah keberhasilan login. Baris kedua menguji *username* valid tetapi *password* tidak valid, sehingga login harus gagal. Baris ketiga menguji username tidak valid dengan *password* valid dan menghasilkan kegagalan login. Sementara itu, baris keempat menguji kondisi ketika *username* dan *password* sama-sama tidak valid dan hasilnya juga berupa kegagalan login. Dengan menguji seluruh kombinasi tersebut, berbagai kemungkinan input yang telah diidentifikasi dalam kebutuhan dapat diperiksa terhadap hasil yang diharapkan.

Pengujian *Use Case*

Pengujian *Use Case* merupakan pendekatan pengujian perangkat lunak yang berfokus pada validasi fungsi dan batasan sistem berdasarkan *use case*, yaitu deskripsi mengenai bagaimana pengguna berinteraksi dengan sistem untuk mencapai tujuan tertentu. Pendekatan ini sangat berguna terutama dalam pengujian pada tingkat sistem karena pengujian

dilakukan dengan melihat perilaku sistem berdasarkan kebutuhan dan skenario interaksi pengguna (Sabharwal dkk., 2011).

Pelaksanaan pengujian *Use Case* dilakukan melalui beberapa tahapan yang saling berkaitan.

1. Tahap pertama adalah mengidentifikasi *use case* dengan mengumpulkan dan mendokumentasikan use case yang menggambarkan kebutuhan fungsional serta interaksi antara pengguna (*actor*) dan sistem.
2. Tahap kedua adalah menurunkan skenario yaitu memecah setiap *use case* menjadi beberapa skenario berupa urutan tindakan atau kejadian yang menggambarkan kemungkinan interaksi pengguna.
3. Tahap ketiga adalah menghasilkan kasus uji berdasarkan skenario yang dihasilkan pada tahap 2. Setiap kasus uji dirancang untuk mencakup jalur atau perilaku tertentu yang dijelaskan dalam use case. Dengan demikian, use case menjadi dasar untuk menentukan apa yang perlu diuji, sedangkan skenario memberikan gambaran mengenai bagaimana pengujian tersebut dilakukan.
4. Setelah kasus uji disusun, tahap berikutnya adalah mengeksekusi pengujian. Kasus uji dijalankan terhadap sistem, umumnya melalui *Graphical User Interface* (GUI), untuk memverifikasi apakah sistem berperilaku sesuai dengan yang telah dijelaskan dalam *use case*.
5. Hasil aktual dari pelaksanaan pengujian kemudian dibandingkan dengan hasil yang diharapkan melalui tahap validasi hasil. Apabila perilaku aktual sistem

sesuai dengan hasil yang telah ditentukan dalam *use case* dan skenario, maka sistem menunjukkan perilaku yang diharapkan. Sebaliknya, perbedaan antara perilaku aktual dan hasil yang diharapkan menunjukkan adanya kondisi yang perlu diperhatikan dalam proses pengujian.

Untuk memahami penerapan pengujian *use case*, berikut ini adalah contoh pengujian aplikasi belanja daring dengan *use case* "*Place an Order"*.

Tabel 4. 4 Contoh *Use Case "Place an Order"*

Kasus Uji	Skenario yang diuji	Langkah dalam *use case*
Place an order with valid payment details	Menguji proses pemesanan ketika informasi pembayaran valid	1. Pengguna login 2. Pengguna memilih barang untuk dimasukkan ke keranjang 3. Pengguna melanjutkan ke tahap pembayaran 4. Pengguna memasukkan data pengiriman dan informasi pembayaran 5. Pengguna mengkonfirmasi pesanan
Place an order with invalid payment details	Menguji proses pemesanan ketika informasi pembayaran tidak valid	
Place an order with an empty cart	Menguji proses pemesanan ketika keranjang dalam keadaan kosong	

Contoh di atas menunjukkan bahwa pengujian *Use Case* menghubungkan kebutuhan pengguna, interaksi pengguna dengan sistem, skenario, dan kasus uji dalam satu alur pengujian.

Pengujian White-Box

Pengujian *White-Box* merupakan teknik pengujian perangkat lunak yang dilakukan dengan pengetahuan penuh terhadap struktur internal, logika, dan kode program yang diuji.

Karakteristik utama pengujian *White-Box* adalah adanya akses ke kode sumber. Penguji menggunakan kode sumber untuk memahami struktur dan logika program, kemudian merancang kasus uji berdasarkan bagian-bagian kode tersebut.

Pengujian *White-Box* memiliki keunggulan dalam mendeteksi cacat lebih awal, khususnya pada tahap unit testing dan integration testing. Dalam penerapannya, pengujian *White-Box* dapat digunakan untuk memverifikasi modul atau fungsi individual melalui unit pengujian, memeriksa interaksi antarkomponen melalui *integration testing*, serta memastikan bahwa perubahan pada kode tidak merusak fungsi yang sebelumnya telah berjalan melalui *regression testing* (Sasmito, 2020).

Pelaksanaan pengujian *White-Box* dilakukan melalui beberapa tahapan yang terstruktur yang terdiri dari :

1. Memahami kode sumber. Pada tahap ini, penguji meninjau kode sumber, unit program, dan batas-batas program untuk memperoleh pemahaman menyeluruh mengenai struktur dan logika internal sistem.

2. Menentukan kriteria cakupan. Penguji memilih ukuran cakupan kode yang sesuai, seperti *statement coverage*, *branch coverage*. Kriteria tersebut digunakan sebagai pedoman dalam merancang pengujian sekaligus mengukur sejauh mana kode telah tercakup oleh kasus uji. Dengan adanya kriteria cakupan, penguji dapat mengetahui bagian kode yang telah diuji maupun bagian yang masih membutuhkan pengujian tambahan.
3. Merancang kasus uji. Kasus uji dibuat untuk menjalankan berbagai *path*, kondisi, dan *branch* yang terdapat dalam kode secara sistematis.
4. Menjalankan pengujian baik pada tingkat unit untuk menguji modul individual maupun pada tingkat integrasi untuk menguji beberapa modul yang telah digabungkan.
5. Menganalisis hasil. Penguji mengevaluasi bagian kode yang telah dijalankan dan mengidentifikasi elemen kode yang belum tercakup. Apabila ditemukan cacat atau kegagalan, dilakukan tahap perbaikan cacat. Setelah perbaikan dilakukan, *regression testing* dapat dijalankan untuk memastikan bahwa perbaikan tersebut tidak menimbulkan masalah baru pada bagian sistem lainnya.

Pengujian *Grey-Box*

Pengujian *Grey-Box* merupakan pendekatan pengujian perangkat lunak yang menggabungkan unsur-unsur pengujian *Black-Box* dan pengujian *White-Box*. Pada pendekatan ini, penguji memiliki pengetahuan sebagian mengenai cara kerja internal sistem. Pengetahuan tersebut lebih banyak dibandingkan

pengujian *Black-Box*, tetapi tidak sampai pada tingkat akses dan pemahaman penuh sebagaimana pada pengujian *White-Box* (Wang & Fang, 2020). Dengan adanya pengetahuan internal yang terbatas, penguji dapat merancang kasus uji yang lebih terarah dibandingkan pengujian yang hanya berfokus pada fungsi, tanpa harus memiliki akses penuh terhadap kode sumber. Pengujian *Grey-Box* merupakan kombinasi teknik, karena memanfaatkan strategi berbasis spesifikasi seperti pengujian *Black-Box* dan strategi berbasis kode sumber seperti pada pengujian *White-Box* (Choi dkk., 2019).

Pelaksanaan pengujian *Grey-Box* dilakukan melalui beberapa tahapan yakni:

1. Tahap pertama yaitu memperoleh informasi tingkat tinggi mengenai sistem, seperti diagram arsitektur, API, atau skema basis data, tanpa memperoleh seluruh kode sumber.
2. Tahap kedua yaitu menggunakan informasi tersebut untuk mengidentifikasi titik integrasi yang kritis, aliran data, atau bagian sistem yang berpotensi mengalami kesalahan.
3. Tahap ketiga yaitu membuat kasus uji yang menargetkan antarmuka pengguna sekaligus proses yang mendasarinya, dengan memperhatikan bagaimana berbagai komponen saling berinteraksi.
4. Tahap keempat yaitu menjalankan pengujian sambil memantau output dan, apabila memungkinkan, kondisi internal sistem.
5. Tahap terakhir yaitu menggunakan perilaku yang diamati dan informasi internal yang tersedia secara terbatas untuk membantu mendiagnosis serta melaporkan defek.

4.2 Implementasi Sistem

Setelah melalui berbagai skenario pengujian, sistem informasi selanjutnya harus melalui proses implementasi agar dapat digunakan oleh pengguna sesuai dengan kebutuhan yang telah ditetapkan. Implementasi bukan sekadar tahap akhir setelah sistem selesai dikembangkan, melainkan proses yang memerlukan perencanaan dan persiapan sejak jauh sebelum proses akuisisi selesai. Perencanaan yang tidak dilakukan secara cermat dan tidak mempertimbangkan berbagai faktor yang berkaitan dengan penerapan sistem dapat menimbulkan berbagai permasalahan. Sebaliknya, implementasi yang direncanakan dengan baik diperlukan untuk memastikan bahwa perangkat lunak yang telah dikembangkan dan sesuai dengan kebutuhan pengguna dapat diterima serta digunakan secara efektif.

Implementasi sistem informasi merupakan tahap ketika produk yang selesai dikembangkan dan diuji dinyatakan siap untuk diterapkan di lingkungan pengguna. Proses rilis dan implementasi sistem informasi selanjutnya mencakup sejumlah aktivitas yang mendukung kesiapan sistem dan pengguna. Aktivitas tersebut antara lain penyusunan manual pengguna akhir dan manual teknis, penyediaan pelatihan bagi pengguna akhir, serta instalasi sistem di lokasi pelanggan. Penyebaran sistem dapat mencakup pembuatan data master dan beberapa data transaksi yang diperlukan agar sistem dapat mulai digunakan. Selain itu, pengujian kelayakan juga diperlukan untuk memastikan bahwa sistem yang akan diterapkan telah memenuhi kondisi yang diperlukan sebelum digunakan.

4.2.1 Strategi Implementasi Sistem

Setelah sistem informasi selesai dikembangkan dan melalui proses pengujian, tahap berikutnya adalah menentukan strategi implementasi yang tepat untuk menerapkan sistem ke lingkungan pengguna. Pemilihan strategi implementasi perlu mempertimbangkan kesiapan sistem, kondisi organisasi, kebutuhan pengguna, serta risiko yang mungkin muncul selama proses peralihan dari sistem lama ke sistem baru. Strategi yang dipilih akan menentukan bagaimana sistem baru diperkenalkan, digunakan, dan diterima oleh pengguna, sekaligus memengaruhi waktu, sumber daya, dan tingkat risiko dalam proses implementasi. Oleh karena itu, implementasi perlu direncanakan secara cermat agar proses peralihan dapat berlangsung sesuai dengan tujuan yang telah ditetapkan.

Secara umum, terdapat beberapa pendekatan yang dapat digunakan dalam implementasi sistem, yaitu implementasi langsung, implementasi paralel, dan implementasi pilot Ketiga strategi tersebut memiliki karakteristik yang berbeda dalam hal cara penerapan sistem, proses transisi, kebutuhan sumber daya, serta tingkat risiko yang dihadapi yang akan dibahas pada subbab ini.

Implementasi Langsung

Implementasi langsung merupakan pendekatan implementasi sistem dengan cara menghentikan sistem lama dan menggantinya dengan sistem baru pada tanggal yang telah ditentukan. Pada saat peralihan dilakukan, sistem baru mulai digunakan dan pengguna tidak lagi dapat menggunakan sistem lama. Pendekatan ini memiliki karakteristik utama berupa peralihan yang tegas dan langsung. Setelah tanggal peralihan ditetapkan, sistem lama tidak lagi menjadi pilihan bagi

pengguna sehingga seluruh aktivitas harus dilakukan menggunakan sistem baru. Oleh karena itu, sebelum peralihan dilakukan, sistem baru harus telah dipersiapkan secara menyeluruh. Persiapan tersebut mencakup pengujian terhadap sistem baru beserta seluruh antarmuka yang digunakan, sehingga dapat dipastikan bahwa sistem benar-benar dapat diterima dan digunakan sebagaimana yang diharapkan (Elvis Foster & Bradford Towle Jr., 2021) .

Syarat utama penerapan peralihan langsung adalah sistem baru telah diuji secara menyeluruh dan terbukti sepenuhnya dapat diterima. Hal ini penting karena setelah peralihan berlangsung, pengguna tidak memiliki akses terhadap sistem lama sebagai alternatif. Dengan kata lain, keberhasilan peralihan sangat bergantung pada tingkat kesiapan dan penerimaan sistem baru sebelum tanggal penerapan ditetapkan. Apabila sistem telah memenuhi kondisi tersebut, pendekatan ini diharapkan dapat menghasilkan proses perpindahan dengan penundaan yang minimal.

Implementasi Paralel

Implementasi paralel merupakan pendekatan implementasi yang dilakukan dengan menjalankan sistem lama dan sistem baru secara bersamaan selama periode transisi. Kedua sistem tetap digunakan hingga pengguna merasa puas dan yakin bahwa sistem baru telah dapat berfungsi sebagaimana yang diharapkan. Setelah sistem baru dinilai memadai, sistem lama kemudian dihentikan dan tidak lagi digunakan. Pendekatan ini memberikan kesempatan kepada pengguna untuk beradaptasi dengan sistem baru secara bertahap sekaligus menyediakan sistem lama sebagai pendukung selama proses perubahan berlangsung.

Secara tradisional, konversi paralel digunakan dalam proses perpindahan dari sistem manual ke sistem terkomputerisasi. Namun, pendekatan yang sama juga dapat diterapkan ketika organisasi berpindah dari sistem terkomputerisasi lama ke sistem yang lebih baru. Dengan menjalankan kedua sistem secara bersamaan, pengguna memperoleh waktu untuk menggunakan dan menilai sistem baru sebelum sistem lama sepenuhnya dihentikan. Hal tersebut memberikan rasa aman bagi pengguna karena sistem lama masih tersedia selama proses transisi. Apabila pengguna belum merasa puas dengan sistem baru, sistem lama masih dapat digunakan sehingga proses perubahan tidak langsung bergantung sepenuhnya pada sistem baru (Elvis Foster & Bradford Towle Jr., 2021).

Meskipun memberikan keamanan selama masa transisi, konversi paralel memiliki konsekuensi berupa kebutuhan usaha dan biaya yang besar. Hal ini terjadi karena organisasi harus menjalankan dua sistem dalam periode yang sama. Dalam beberapa kondisi, diperlukan dua kelompok pekerja untuk menangani masing-masing sistem secara paralel. Pada kondisi lain, pekerja yang sama mungkin harus menangani pekerjaan dari kedua sistem sehingga diperlukan kerja lembur. Dengan demikian, keamanan yang diperoleh melalui keberadaan sistem lama selama masa transisi harus dipertimbangkan bersama dengan tambahan usaha dan biaya yang muncul akibat pelaksanaan kedua sistem secara bersamaan.

Implementasi Pilot

Implementasi pilot merupakan implementasi terhadap suatu sistem, proses, atau proyek baru dalam skala terbatas sebelum diterapkan secara penuh.

Pendekatan implementasi pilot memberikan kesempatan untuk melihat bagaimana sistem bekerja dalam lingkungan dan konteks penggunaan yang sebenarnya. Pengguna yang terlibat dalam tahap ini dapat memberikan umpan balik berdasarkan pengalaman mereka ketika menggunakan sistem. Informasi tersebut dapat digunakan untuk mengidentifikasi *design flaws* maupun permasalahan implementasi yang sebelumnya mungkin belum terlihat. Karena dilakukan dalam skala terbatas dan lingkungan yang terkendali, organisasi memiliki kesempatan untuk menangani permasalahan tersebut sebelum sistem diterapkan kepada pengguna dalam skala yang lebih luas (Bansler & Havn, 2009).

Mengapa Menggunakan Implementasi Pilot?

Terdapat tiga alasan utama menggunakan implementasi pilot yaitu:

1. Mengurangi risiko. Dengan menemukan dan menyelesaikan permasalahan sejak tahap awal, organisasi dapat menghindari kesalahan yang lebih mahal ketika sistem telah diterapkan secara penuh.
2. Meningkatkan kegunaan, karena umpan balik dari pengguna nyata membantu memastikan bahwa sistem sesuai dengan kebutuhan aktual dan konteks pekerjaan. Penggunaan sistem secara langsung memungkinkan organisasi memperoleh pemahaman mengenai kesesuaian sistem dengan lingkungan tempat sistem akan digunakan.
3. Berperan dalam mendukung manajemen perubahan. Keterlibatan pemangku kepentingan selama tahap pilot dapat membantu organisasi mengelola perubahan yang muncul akibat penerapan sistem baru sekaligus meningkatkan penerimaan terhadap sistem. Ketika pemangku kepentingan memperoleh kesempatan untuk

terlibat dan memberikan umpan balik, organisasi dapat menggunakan pengalaman tersebut untuk memperbaiki proses penerapan sebelum dilakukan secara luas.

4.2.2 Pelatihan Pengguna, Dokumentasi dan Manual Pengguna

Keberhasilan implementasi sistem tidak hanya ditentukan oleh kesiapan perangkat lunak secara teknis, tetapi juga oleh kesiapan pengguna dalam memahami dan menggunakan sistem serta tersedianya informasi yang dapat menjadi panduan selama penggunaan. Sistem yang telah dikembangkan dan dinyatakan siap diterapkan tetap dapat menghadapi kendala apabila pengguna tidak memiliki pengetahuan dan keterampilan yang memadai atau tidak memperoleh dokumentasi yang jelas. Oleh karena itu, pelatihan pengguna, dokumentasi, dan manual pengguna menjadi bagian penting dalam proses implementasi untuk membantu memastikan bahwa sistem dapat digunakan secara efektif, efisien, dan sesuai dengan fungsi yang telah dirancang.

Pelatihan Pengguna

Pelatihan pengguna merupakan salah satu aktivitas inti dalam proses implementasi sistem informasi. Pelatihan diperlukan agar pengguna mampu menggunakan perangkat lunak secara efektif dan efisien sesuai dengan fungsi yang tersedia. Strategi pelatihan perlu disesuaikan dengan kondisi organisasi, terutama ukuran dan persebaran lokasi pengguna. Pada organisasi yang berukuran kecil dan berada dalam satu wilayah, pelatihan dapat diberikan secara langsung kepada seluruh pengguna akhir. Sebaliknya, pada organisasi

besar yang memiliki pengguna yang tersebar secara geografis, pelatihan kepada seluruh pengguna secara langsung dapat membutuhkan waktu dan biaya yang besar. Salah satu pendekatan yang dapat digunakan adalah mengidentifikasi beberapa pengguna kunci (key user) untuk memperoleh pelatihan terlebih dahulu. Selanjutnya, pengguna kunci tersebut meneruskan pengetahuan dan keterampilan yang diperoleh kepada pengguna akhir lainnya dalam organisasi.

Media pembelajaran berbasis video juga dapat digunakan sebagai bagian dari pelatihan pengguna, terutama ketika produk perangkat lunak didistribusikan atau diterapkan melalui web. Alat untuk membuat video dan gambar yang mampu merekam tampilan layar pengguna dapat membantu menghasilkan materi pelatihan yang menunjukkan langkah-langkah penggunaan perangkat lunak secara visual. Materi tersebut dapat digunakan untuk membantu pengguna memahami cara menggunakan berbagai fungsi perangkat lunak secara efektif.

Pelatihan pengguna akhir merupakan bagian integral dari rekayasa perangkat lunak. Dalam menentukan strategi pelatihan, terdapat beberapa pertanyaan yang perlu dipertimbangkan, yaitu siapa yang akan dilatih, siapa yang menyelenggarakan pelatihan, sumber daya apa yang tersedia, dan di mana pelatihan akan dilaksanakan. Pelatihan dapat dilakukan oleh insinyur perangkat lunak, vendor, maupun pengguna perangkat lunak internal. Pemilihan pihak yang memberikan pelatihan perlu disesuaikan dengan kondisi organisasi dan kebutuhan peserta. Pelatihan juga dapat dilaksanakan di pusat pelatihan yang telah

disiapkan secara khusus atau di lokasi organisasi atau departemen yang menerima pelatihan. Pelatihan di lokasi pengguna memberikan kemudahan bagi organisasi, tetapi pilihan ini memiliki biaya yang lebih tinggi.

Tujuan pelatihan perlu disampaikan secara jelas kepada peserta sejak awal. Kejelasan tersebut membantu peserta memahami kemampuan yang diharapkan setelah mengikuti pelatihan dan mempersiapkan diri dengan baik. Tujuan pelatihan juga perlu disesuaikan dengan karakteristik peserta. Sebagai contoh, staf administrasi yang menggunakan sistem untuk entri data membutuhkan pendekatan pelatihan yang berbeda dengan manajer yang menggunakan sistem untuk kebutuhan organisasi dan pengambilan keputusan. Oleh karena itu, metode pelatihan perlu mempertimbangkan kebutuhan peserta dengan mengombinasikan unsur visual, pendengaran, dan praktik langsung sesuai kebutuhan.

Keberhasilan pelatihan juga dipengaruhi oleh kesiapan materi yang digunakan. Materi pelatihan perlu dipersiapkan dengan baik dan umumnya dapat berupa panduan pengguna, lembar ringkasan, dan buku kerja. Materi tersebut membantu peserta mengikuti proses pembelajaran sekaligus menyediakan referensi yang dapat digunakan ketika perangkat lunak mulai digunakan dalam pekerjaan sehari-hari. Apabila diperlukan evaluasi terhadap peserta, evaluasi harus dilakukan secara objektif berdasarkan kriteria kinerja yang telah ditetapkan sebelumnya. Dalam kondisi tertentu, instruktur juga dapat dievaluasi oleh peserta sebagai bagian dari upaya penjaminan mutu yang dilakukan oleh

perusahaan atau departemen yang menyelenggarakan pelatihan. Dengan demikian, pelatihan pengguna tidak hanya bertujuan memperkenalkan cara penggunaan sistem, tetapi juga memastikan bahwa pengguna memiliki kemampuan yang sesuai dengan tujuan dan standar kinerja yang telah ditetapkan.

Dokumentasi dan Manual Pengguna

Dokumentasi merupakan bagian penting dari setiap produk perangkat lunak karena penggunaan sistem yang tidak tepat dapat menyebabkan kerusakan pada produk, data, maupun sistem lain yang terhubung dengannya. Salah satu penyebab penggunaan perangkat lunak yang tidak tepat adalah tidak tersedianya manual pengguna *(user manual)* yang baik. Manual pengguna berfungsi sebagai referensi yang menjelaskan cara menggunakan produk perangkat lunak beserta berbagai fungsi yang disediakan. Dengan adanya manual yang lengkap dan mudah dipahami, pengguna dapat mempelajari fungsi sistem dan menggunakannya sesuai dengan prosedur yang telah ditentukan. Sebaliknya, tanpa dokumentasi yang memadai, pengguna berpotensi mengalami kesulitan dalam memahami dan memanfaatkan seluruh fungsi perangkat lunak secara tepat.

Manual pengguna yang baik harus menyediakan instruksi langkah demi langkah untuk setiap fitur atau fungsi yang tersedia. Apabila suatu transaksi, misalnya transaksi penjualan, terdiri atas lima langkah, maka seluruh langkah tersebut beserta berbagai pilihan yang tersedia perlu dijelaskan secara lengkap. Dokumentasi juga perlu mencantumkan alternatif penyelesaian (walk-around) apabila terdapat cacat perangkat lunak yang

belum dapat diperbaiki dan menyebabkan suatu fungsi tidak dapat digunakan sebagaimana mestinya.

Untuk membantu pengguna memahami prosedur dengan lebih jelas, manual pengguna sebaiknya dilengkapi dengan tangkapan layar sehingga pengguna dapat melihat secara langsung bagian antarmuka yang harus digunakan. Dalam penyusunannya, tersedia pula berbagai alat otomatisasi, seperti Activedocs dan Report Lab, yang dapat membantu meningkatkan produktivitas dalam pembuatan manual pengguna. Namun, manual yang dihasilkan secara otomatis tetap perlu ditinjau dan disempurnakan secara manual agar benar-benar mudah digunakan.

Manual teknis (technical manual) memiliki peran penting dalam pemeliharaan produk perangkat lunak karena memberikan informasi yang membantu tim pemeliharaan memahami struktur dan cara kerja perangkat lunak. Dokumentasi teknis mencakup informasi mengenai arsitektur perangkat lunak dan bagaimana rancangan tersebut diimplementasikan. Selain dokumentasi teknis secara umum, komentar pada kode sumber memiliki nilai penting. Dalam kode sumber terdapat pemanggilan terhadap kelas, metode, dan variabel yang didefinisikan di bagian lain. Komentar pada bagian pemanggilan tersebut dapat menjelaskan alasan pemanggilan dilakukan. Demikian pula, ketika terdapat bagian kode sumber yang besar atau memiliki logika yang kompleks, komentar dapat membantu tim pemeliharaan memahami logika yang digunakan.

Dokumentasi kode sumber menjadi semakin penting karena memahami kode yang ditulis oleh orang lain sering kali tidak mudah. Bahkan pengembang yang

menulis kode sumber tersebut dapat mengalami kesulitan mengingat kembali logika program yang dibuatnya setelah waktu yang cukup lama. Oleh karena itu, pendokumentasian kode sumber merupakan bagian penting dalam mendukung pemeliharaan perangkat lunak. Dokumentasi teknis juga menjadi sangat penting apabila produk perangkat lunak dirancang untuk diintegrasikan dengan produk perangkat lunak lain. Dalam kondisi tersebut, diperlukan dokumentasi *Application Programming Interface* (API) yang lengkap untuk menjelaskan antarmuka yang tersedia dan bagaimana antarmuka tersebut dapat digunakan untuk melakukan integrasi.

DAFTAR PUSTAKA

Bansler, J. P., & Havn, E. C. (2009). Pilot implementation of health information systems: Issues and challenges. *15th Americas Conference on Information Systems 2009, AMCIS 2009*, *6*, 4044–4050. https://doi.org/https://doi.org/10.1016/j.ijmedinf.2010.05.004

Chandrasekaran, G., Neethidevan, V., & Murugachandravel, J. (2019). Impact of Unit Testing in Web Automation Testing. *International Journal of Recent Technology and Engineering*, *8*(3), 1011–1013. https://doi.org/10.35940/ijrte.C4064.098319

Choi, J., Jang, J., Han, C., & Cha, S. K. (2019). Grey-Box Concolic Testing on Binary Code. *Proceedings - International Conference on Software Engineering*, *2019-May*, 736–747. https://doi.org/10.1109/ICSE.2019.00082

Elvis Foster, & Bradford Towle Jr. (2021). *Software Engineering A Methodical Approach, 2nd Edition* (2nd Edition). Auerbach Publications.

Gaur, J., Goya, A., Choudhury, T., & Sabitha, S. (2016, November). A Walk Through of Software Testing Techniques. *Proceedings of the SMART*. https://doi.org/10.1109/SYSMART.2016.7894499

Guy Wingate. (2010). *Pharmaceutical Computer Systems Validation Quality Assurance, Risk Management and Regulatory Compliance*. CRC Press Taylor & Francis Group.

Jin, H., & Zeng, F. (2011). Research on the Definition and Model of Software Testing Quality. *The Proceedings of 2011 9th International Conference on Reliability,*

Maintainability and Safety, 639–644. https://doi.org/10.1109/ICRMS.2011.5979352

Joosten, J., Permanasari, A. E., & Adji, T. B. (2020). The Use of Decision Table for Reducing Complex Rules in Software Testing. *IOP Conference Series: Materials Science and Engineering*, *732*(1). https://doi.org/10.1088/1757-899X/732/1/012086

Panagiotis Leloudas. (2025). *Software Testing Strategies: Accelerating Software Delivery With Continuous Testing and Integration*. BPB Publication.

Sabharwal, S., Sibal, R., & Sharma, C. (2011). A Genetic Algorithm Based Approach for Prioritization of Test Case Scenario in Static Testing. *International Conference on Computer & Communication Technology (ICCCT)*. https://doi.org/10.1109/ICCCT.2011.6075160

Sasmito, G. W. (2020). White Box Testing with Basis Path Technique in the Demography Administration Website. *ICSECC 2020 - 2nd International Conference on Sustainable Engineering and Creative Computing, Proceedings*, 86–92. https://doi.org/10.1109/ICSECC51444.2020.9557428

Tan, Z. Y. J., Hasa, M. M., Wong, M. Y., & Ramasamy, R. K. (2022). Implementation Approach of Unit and Integration Testing Method Based on Recent Advancements in Functional Software Testing. *Journal of System and Management Sciences*, *12*(4), 85–100. https://doi.org/10.33168/JSMS.2022.0406

Wang, D., & Fang, J. (2020). A Gray Box for Visualizing Instruction Sequence Based on Improved Suffix Tree. *IEEE Access*, *8*, 132157–132170. https://doi.org/10.1109/ACCESS.2020.3009990

BIODATA PENULIS



Marniyati H. Botutihe, S.Kom., M.Kom.

Dosen Program Studi Sistem Informasi

Fakultas Ilmu Komputer Universitas Pohuwato

Penulis lahir di Parigi tanggal 07 Oktober 1990. Penulis adalah dosen tetap pada Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Pohuwato. Menyelesaikan pendidikan S1 pada Teknik Informatika dan melanjutkan S2 pada Jurusan Teknik Informatika. Penulis menekuni bidang Menulis pada bidang Ilmu Komputer.

BIODATA PENULIS



Sitti Najmia Rifai, S.T., M.T.

Dosen Program Studi Komunikasi dan Penyiaran Islam
Fakultas Ushuluddin, Adab, dan Dakwah
Universitas Islam Negeri Kendari

Penulis lahir pada 17 Desember 1995. Saat ini, penulis merupakan dosen pada Program Studi Komunikasi dan Penyiaran Islam, Fakultas Ushuluddin, Adab, dan Dakwah, Institut Agama Islam Negeri Kendari. Penulis menyelesaikan pendidikan Sarjana Teknik pada bidang Teknik Informatika di Universitas Halu Oleo dan melanjutkan pendidikan Magister Teknik pada Program Magister Informatika, Universitas Hasanuddin. Dalam kegiatan pendidikan dan penelitian, penulis menekuni bidang teknologi informasi, teknologi pendidikan, teknologi komunikasi, *data mining*, kecerdasan buatan, serta desain UI/UX. Beberapa kajian yang telah dilakukan berhubungan dengan pengembangan media pembelajaran interaktif berbasis web, pemanfaatan metode *backpropagation* untuk prediksi data, komputasi awan, dan pengembangan sistem informasi.

BIODATA PENULIS



Taufik Rachman, S.Kom., M.T.

Dosen Program Studi Teknik Informatika

STT STIKMA Internasional

Penulis lahir di kota Kendari pada 11 Mei. Penulis adalah dosen tetap pada Program Studi Teknik Informatika Sekolah Tinggi Teknologi STIKMA Internasional dikota Malang. Menyelesaikan pendidikan Strata-1 pada Jurusan Teknik Informatika di Institut Sains & Teknolgo Palapa Malang dan menyelesaikan Studi Magister Jurusan Teknik Industri di Institut Teknologi Nasional Malang. Penulis konsern pada Matakuliah Analisa Perancangan Sistem Informasi, Rekayasa Perangkat Lunak, Basis Data dan Sistem Basis Data Lanjut sesuai bidang keahliannya.

BIODATA PENULIS



Heliawati Hamrul, S.Kom., M.Kom.

Dosen Program Studi Teknik Informatika

Fakultas Teknik Universitas Sulawesi Barat

Dosen di Program Studi Teknik Informatika Fakultas Teknik Universitas Sulawesi Barat. Lahir di Palopo, 15 Oktober 1987. Anak pertama dari tiga bersaudara. Anak dari Bapak Hamrul Kasuak dan Ibu Almh. Martha Pasiangan. Menamatkan pendidikan Program Sarjana (S1) di STMIK Dipanegara Makassar, dan Program Magister-S2 pada STMIK Amikom Jogjakarta. Saat buku ini ditulis, Penulis sedang menempuh pendidikan S3 di Program Studi Doktor Sistem Informasi Universitas Diponegoro. Penulis merupakan Pemegang Sertifikasi bidang IT Analis dari BNSP. Berbagai penelitian yang telah dilakukan dan dipublikasi Jurnal Nasional terindeks Sinta.