

**IMPLEMENTASI APLIKASI INVENTARIS DAN  
PELACAKAN ASET PERUSAHAAN MENGGUNAKAN  
DATA MATRIX BERBASIS ANDROID  
(STUDI KASUS: PT FUJIYAMA TECHNOLOGY  
SOLUTION)**

**LAPORAN SKRIPSI**



**Disusun oleh:**

**NIM : 1122140051**  
**Nama : Rizqiansyah Ramadhan**

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI  
INSTITUT TEKNOLOGI DAN BISNIS  
BINA SARANA GLOBAL  
TANGERANG  
2025**

**INSTITUT TEKNOLOGI DAN BISNIS  
BINA SARANA GLOBAL**

**LEMBAR PERSETUJUAN PEMBIMBING**

Nim	: 1122140051
Nama	: Rizqiansyah Ramadhan
Program Studi	: Teknik Informatika
Fakultas	: Teknologi Informasi Dan Komunikasi
Jenjang Studi	: Strata 1
Judul	: Implementasi Aplikasi Inventaris Dan Pelacakan Aset Perusahaan Menggunakan Data Matrix Berbasis Android

Disetujui untuk dipertahankan dalam sidang Skripsi Periode Semester Ganjil Tahun Akademik 2025/2026.

Tangerang, 10 September 2025

Pembimbing



**Dr. M. Ramaddan Julianti, M.T**

**NUPTK 0053758659130163**

**INSTITUT TEKNOLOGI DAN BISNIS  
BINA SARANA GLOBAL**

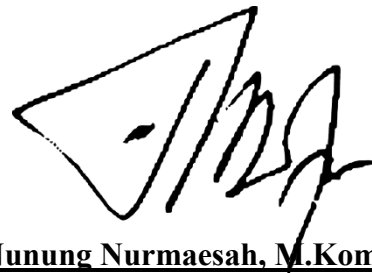
**LEMBAR PERSETUJUAN PENGUJI**

Nim : 1122140051  
Nama : Rizqiansyah Ramadhan  
Program Studi : Teknik Informatika  
Fakultas : Teknologi Informasi Dan Komunikasi  
Jenjang Studi : Strata 1  
Judul : Implementasi Aplikasi Inventaris Dan  
Pelacakan Aset Perusahaan Menggunakan  
Data Matrix Berbasis Android

Disetujui untuk dipertahankan dalam sidang Skripsi Periode Semester Ganjil Tahun Akademik 2025/2026.

Tangerang, 10 September 2025

Penguji,

A handwritten signature in black ink, appearing to read 'Nunung Nurmaesah', is written over a large, stylized, hand-drawn triangle.

**Nunung Nurmaesah, M.Kom**

**NUPTK 8744766667230332**



## KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan laporan skripsi ini. Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, keluarga, para sahabat, dan seluruh umat beliau hingga akhir zaman, aamiin.

Skripsi ini disusun sebagai salah satu syarat kelulusan program Strata 1 (S1) di Institut Teknologi dan Bisnis Bina Sarana Global. Dalam proses penyusunannya, penulis tidak terlepas dari berbagai hambatan dan tantangan. Namun, berkat bimbingan, bantuan, saran, dan kerja sama dari berbagai pihak, khususnya kedua orang tua dan keluarga yang senantiasa memberikan dukungan moral maupun material, motivasi yang luar biasa, serta doa yang tak henti-hentinya, penulis akhirnya dapat menyelesaikan laporan skripsi ini.

Dengan penuh ketulusan, penulis menyampaikan terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. Bapak Assoc. Prof. Dr. H. Dedi, M.Si. selaku Rektor Institut Teknologi dan Bisnis Bina Sarana Global.
2. Bapak Muchamad Iqbal, S.E., M.Kom. selaku Wakil Rektor Bidang Akademik.
3. Bapak Muhammad Iqbal Hanafri, S.Pi., M.Kom selaku Wakil Rektor Bidang Non Akademik.
4. Bapak M. Ramaddan Julianti, M.T. selaku Dekan Fakultas Teknologi Informasi dan Komunikasi
5. Ibu Nunung Nurmaesah, M.Kom. selaku Ketua Program Studi Teknik Informatika.
6. Bapak Dr. M. Ramaddan Julianti, M.T. selaku pembimbing yang selalu bijaksana memberikan bimbingan, nasihat serta waktunya selama penelitian dan penulisan skripsi ini.
7. Ibu Nunung Nurmaesah, M.Kom. selaku Penguji Seminar Skripsi.
8. Seluruh Civitas Akademik Institut Teknologi dan Bisnis Bina Sarana Global yang telah banyak membantu penulis selama mengikuti perkuliahan dan penulisan skripsi ini.
9. Rekan-rekan yang telah membantu penulis dan telah meluangkan waktu untuk mendukung penulis.
10. Semua pihak yang tidak bisa disebutkan satu persatu yang telah banyak membantu penulis dalam menyelesaikan skripsi ini.

Semoga Allah SWT membalas semua kebaikan dengan pahala yang berlipat ganda. Penulis dengan senang hati menerima kritik dan saran yang membangun demi perbaikan di

masa mendatang. Akhirnya, penulis menyerahkan segala hasil dan usaha ini kepada Allah SWT, dengan harapan semoga laporan skripsi ini dapat memberi manfaat, baik bagi penulis sendiri maupun bagi pembaca pada umumnya.

Tangerang, 10 September 2025

Penulis,

Rizqiansyah Ramadhan

# DAFTAR ISI

LEMBAR PERSETUJUAN PEMBIMBING

LEMBAR PERSETUJUAN PENGUJI

KATA PENGANTAR .....	i
DAFTAR ISI.....	iii
DAFTAR GAMBAR .....	vi
DAFTAR TABEL .....	vii
DAFTAR SIMBOL .....	viii
DAFTAR LAMPIRAN.....	xiii
BAB I PENDAHULUAN .....	1
1.1    Latar Belakang .....	1
1.2    Identifikasi Masalah .....	2
1.3    Rumusan Masalah .....	3
1.4    Ruang Lingkup .....	3
1.5    Tujuan Dan Manfaat.....	4
1.5.1    Tujuan Penelitian.....	4
1.5.2    Manfaat penelitian.....	4
1.6    Metode Penelitian.....	5
1.6.1    Jenis penelitian .....	5
1.6.2    Metode pengumpulan data .....	5
1.6.3    Metode Analisis dan Perancangan.....	5
1.7    Sistematika Penulisan.....	6
BAB II LANDASAN TEORI.....	8
2.1    Teori-Teori Dasar/Umum .....	8
2.1.1    Konsep Dasar Sistem .....	8
2.1.1.1    Definisi Sistem .....	8
2.1.1.2    Karakteristik Sistem .....	8
2.1.1.3    Klasifikasi Sistem.....	10
2.1.2    Konsep Dasar Informasi.....	11
2.1.2.1    Definisi Data dan Informasi .....	11
2.1.2.2    Kualitas Informasi .....	12
2.1.3    Konsep Dasar Sistem Informasi.....	12
2.1.3.1    Definisi Sistem Informasi.....	12
2.1.3.2    Komponen Sistem Informasi.....	13

2.1.4	Metodologi Penelitian ( <i>Research and Development</i> ) .....	15
2.1.5	Metodologi Pengembangan Sistem.....	16
2.1.5.1	Definisi SDLC ( <i>System Development Life Cycle</i> ) .....	16
2.1.5.2	Metode Waterfall.....	17
2.1.6	Konsep Perancangan Berorientasi Objek.....	19
2.1.6.1	Analisis dan Perancangan Berorientasi Objek (OOAD).....	19
2.1.6.2	Definisi UML (Unified Modeling Language).....	20
2.1.6.3	Jenis-jenis Diagram UML .....	20
2.1.7	Konsep Dasar Jaringan dan Arsitektur.....	27
2.1.7.1	Arsitektur Client-Server .....	27
2.1.7.2	Definisi API (Application Programming Interface).....	27
2.1.7.3	Konsep REST (Representational State Transfer).....	27
2.1.8	Konsep Internasionalisasi dan Lokalisasi Perangkat Lunak .....	28
2.1.8.1	Internasionalisasi (i18n).....	28
2.1.8.2	Lokalisasi (L10n).....	28
2.2	Teori-Teori Khusus .....	29
2.2.1	Konsep Dasar Manajemen Aset .....	29
2.2.1.1	Definisi Aset dan Inventaris .....	29
2.2.1.2	Definisi Manajemen Aset.....	29
2.2.1.3	Siklus Hidup Aset ( <i>Asset Lifecycle</i> ) .....	30
2.2.2	Konsep Dasar Kode Dua Dimensi (2D Code) .....	31
2.2.2.1	Pengertian Kode Dua Dimensi.....	31
2.2.2.2	Prinsip Kerja dan Keunggulan Umum .....	32
2.2.3	Teknologi Kode Dua Dimensi Pilihan .....	32
2.2.3.1	Karakteristik dan Struktur <i>Data Matrix</i> .....	32
2.2.3.2	Karakteristik dan Struktur QR Code .....	33
2.2.3.3	Analisis Perbandingan Teknis: <i>Data Matrix</i> vs. QR Code .....	33
2.2.3.4	Justifikasi Pemilihan <i>Data Matrix</i> untuk Manajemen Aset .....	34
2.2.4	Teknologi Sisi Klien (Client-Side).....	35
2.2.4.1	Platform Android .....	35
2.2.4.2	Bahasa Pemrograman Dart.....	36
2.2.4.3	<i>Framework</i> Flutter .....	36
2.2.4.4	Arsitektur Perangkat Lunak: Clean Architecture .....	37
2.2.5	Teknologi Sisi Server ( <i>Server-Side</i> ) .....	39
2.2.5.1	Bahasa Pemrograman Go (Golang) .....	39



2.2.5.2	<i>Framework</i> Fiber .....	40
2.2.5.3	Object-Relational Mapping (ORM): GORM .....	41
2.2.6	Teknologi Basis Data dan Notifikasi.....	42
2.2.6.1	Sistem Manajemen Basis Data: PostgreSQL .....	42
2.2.6.2	Layanan Notifikasi: Firebase <i>Cloud</i> Messaging (FCM) .....	43
2.2.7	Format Pertukaran Data .....	44
2.2.7.1	JSON (JavaScript Object Notation) .....	44
2.3	Literature Review .....	46
BAB III ANALISIS SISTEM YANG BERJALAN .....		52
3.1	Gambaran Umum Objek .....	52
3.1.1	Sejarah Singkat.....	52
3.1.2	Struktur Organisasi.....	52
3.1.3	Wewenang dan Tanggung Jawab.....	53
3.2	Tata Laksana Sistem Yang Berjalan .....	55
3.2.1	Prosedur Sistem yang Berjalan .....	55
3.2.2	Activity Diagram Sistem Yang Berjalan .....	55
3.3	Masalah yang Dihadapi .....	56
3.4	Alternatif Pemecahan Masalah.....	57
3.5	User Requirement (Elisitasi) .....	57
3.5.1	Elisitasi tahap I .....	57
3.5.2	Elisitasi tahap II.....	59
3.5.3	Elisitasi tahap III .....	60
3.5.4	Final Elisitasi.....	62
DAFTAR PUSTAKA .....		64
DAFTAR LAMPIRAN.....		70

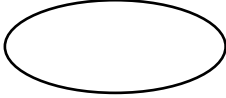

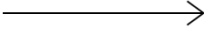
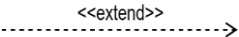
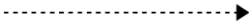
## DAFTAR GAMBAR


Gambar 2. 1 Alur Sistem .....	10
Gambar 2. 2 Building Blocks .....	15
Gambar 2. 3 Metode Waterfall .....	18
Gambar 2. 4 <i>Client Server</i> .....	27
Gambar 2. 5 <i>Asset Lifecyle</i> .....	31
Gambar 2. 6 <i>Barcode</i> 1D vs 2D .....	31
Gambar 2. 7 <i>Data Matrix</i> .....	32
Gambar 2. 8 Clean Architecturede .....	39
Gambar 2. 9 Arsitektur <i>Server</i> .....	46
Gambar 3. 1 Struktur Organisasi.....	52
Gambar 3. 2 Activity Diagram Sistem yang Berjalan.....	56



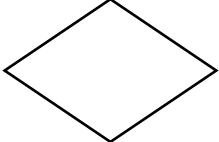


## DAFTAR TABEL

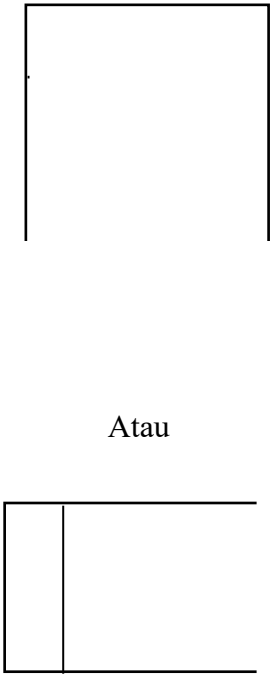
Tabel 2. 1 Simbol-simbol Use Case Diagram.....	21
Tabel 2. 2 Simbol-simbol Activity Diagram.....	22
Tabel 2. 3 Simbol-Simbol Sequence Diagram.....	24
Tabel 2. 4 Simbol-simbol Class Diagram.....	25
Tabel 2. 5 Perbandingan <i>Data Matrix</i> dan QR Code.....	33
Tabel 2. 6 <i>Literature Review</i> .....	46
Tabel 3. 1 Wewenang dan tanggung jawab.....	53
Tabel 3. 2 Elisitasi Tahap 1 .....	58
Tabel 3. 3 Elisitasi Tahap 2.....	59
Tabel 3. 4 Elisitasi Tahap 3 .....	61
Tabel 3. 5 <i>Final Draft</i> Elisitasi .....	62



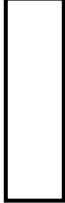
## DAFTAR SIMBOL

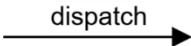
Simbol	Nama	Deskripsi
	Use Case	Merepresentasikan sebuah fungsionalitas spesifik yang disediakan oleh sistem untuk mencapai tujuan tertentu bagi aktor.
	Aktor	Mewakili peran pengguna, sistem eksternal, atau perangkat keras yang berinteraksi secara langsung dengan sistem.
	Asosiasi	Garis yang menghubungkan Aktor dengan Use Case, menandakan bahwa aktor tersebut terlibat dalam fungsionalitas yang digambarkan.
	Ekstensi/ <i>Extend</i>	Relasi yang menunjukkan fungsionalitas opsional atau tambahan. Use case yang menjadi ekstensi dapat berjalan secara mandiri.
	Generalisasi/ <i>Generalization</i>	Hubungan hierarkis antara elemen yang lebih umum (induk) dengan yang lebih spesifik (anak), di mana anak mewarisi properti induk.

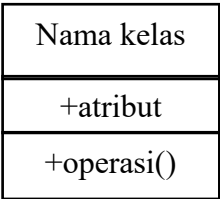

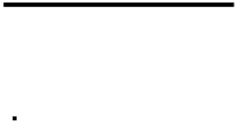
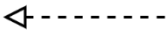
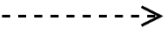
	<p>Menggunakan</p> <p>/ <i>Include</i> /</p> <p><i>uses</i></p>	<p>Relasi yang menunjukkan bahwa sebuah use case memerlukan fungsionalitas dari use case lain untuk dapat dieksekusi.</p>
---	---	---

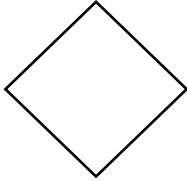
Gambar	Nama	Deskripsi
	Status Awal	Titik mula dari keseluruhan alur kerja dalam diagram. Setiap diagram aktivitas hanya memiliki satu status awal.
	Aktivitas	Representasi dari sebuah tugas atau langkah spesifik yang dilakukan dalam alur kerja.
	Percabangan/ <i>decision</i>	Titik dalam alur di mana terdapat beberapa kemungkinan jalur berdasarkan kondisi tertentu.
	Penggabungan/ <i>join</i>	Titik di mana beberapa alur kerja paralel atau alternatif yang sebelumnya terpisah, disatukan kembali menjadi satu alur.
	Status Akhir	Titik akhir yang menandakan selesainya keseluruhan alur kerja yang digambarkan.

 <p>Atau</p>	Swimlane	Partisi visual yang mengelompokkan aktivitas berdasarkan aktor atau unit yang bertanggung jawab atas pelaksanaannya.
---	----------	--

Simbol	Nama	Deskripsi
 <div>Nama aktor</div>	Aktor	Entitas eksternal seperti pengguna, perangkat, atau sistem lain yang memulai atau berpartisipasi dalam urutan interaksi.
	Garis hidup / <i>lifeline</i>	Garis vertikal yang merepresentasikan eksistensi sebuah objek selama periode interaksi.
<div>Nama objek: nama kelas</div>	Objek	Merepresentasikan sebuah instansi dari kelas yang terlibat dalam interaksi.
	Waktu aktif	Persegi panjang pada lifeline yang menandakan periode waktu saat objek sedang aktif memproses sebuah tugas.

	Pesan	Pesan yang mengindikasikan bahwa sebuah objek membuat objek lainnya.
---	-------	--

Gambar	Nama	Deskripsi
	Kelas / <i>Class</i>	Cetak biru ( <i>blueprint</i> ) untuk membuat objek. Merepresentasikan entitas utama dalam sistem, lengkap dengan atribut dan operasinya.
	Antarmuka / <i>interface</i>	Sebuah kontrak yang mendefinisikan sekumpulan operasi atau metode tanpa implementasi. Kelas yang mengimplementasikan antarmuka ini wajib menyediakan implementasi untuk semua metodenya.
	Asosiasi / <i>Association</i>	Hubungan struktural yang menunjukkan bahwa objek dari satu kelas terhubung dengan objek dari kelas lain.
	Asosiasi berarah / <i>directed association</i>	Hubungan di mana satu kelas mengetahui atau menggunakan kelas lain, tetapi tidak sebaliknya.
	Ketergantungan / <i>Dependency</i>	Hubungan di mana perubahan pada satu kelas dapat memengaruhi kelas lain,

		tanpa adanya hubungan struktural yang kuat
	Agregasi / <i>Agregation</i>	Bentuk khusus dari asosiasi "memiliki-sebuah" ( <i>has-a</i> ), di mana satu kelas (keseluruhan) terdiri dari kelas lain (bagian), namun bagian tersebut dapat ada secara mandiri.



## DAFTAR LAMPIRAN

Lampiran 1 Form Revisi Skripsi .....	<b>Error! Bookmark not defined.</b>
--------------------------------------	-------------------------------------



# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Di era yang serba digital saat ini, perusahaan mau tidak mau harus menggunakan teknologi untuk meningkatkan efisiensi dan daya saing. Manajemen aset adalah salah satu bagian penting dari operasional yang sering menjadi masalah. Aset perusahaan, baik berupa kendaraan operasional, furnitur, atau perangkat keras teknologi, merupakan investasi besar yang membutuhkan pengelolaan dan pemantauan yang cermat. Metode pelacakan aset konvensional yang masih dilakukan secara manual dapat digantikan dengan teknologi identifikasi kontemporer seperti *Data Matrix*. Penggunaan *smartphone* oleh karyawan secara merata memungkinkan implementasi sistem ini menjadi lebih mudah dan terjangkau, memungkinkan pemindaian dan pembaruan data secara instan.

Kelebihan utama dari penggunaan *Data Matrix* dalam inventarisasi aset adalah kecepatan, akurasi, dan efisiensi biaya. *Data Matrix* dipilih karena memiliki keunggulan dalam kepadatan data yang lebih tinggi pada ukuran fisik yang lebih kecil dan tingkat toleransi kesalahan yang lebih superior dibandingkan kode dua dimensi lainnya. Namun, mengimplementasikannya juga memiliki beberapa tantangan, seperti ketergantungan pada perangkat pemindai (Android). Meskipun demikian, jika dibandingkan dengan risiko kesalahan manusia (*human error*), kehilangan data, dan lambatnya proses audit pada sistem manual, keunggulan teknologi ini jauh lebih dominan.

Fujiyama Technology Solution (FTS) adalah unit/brand layanan teknologi dari PT Fujiyama Marketing yang berfokus pada pengembangan aplikasi (*web & mobile*), konsultasi TI, jaringan & *hardware*, solusi POS & *barcode*, serta layanan *cloud* dan pelatihan. Sebagai perusahaan teknologi yang dinamis dan terus berkembang, jumlah aset teknologi seperti laptop, *server*, dan perangkat jaringan terus bertambah seiring dengan pertumbuhan skala proyek dan jumlah karyawan. Pertumbuhan ini menuntut adanya sebuah sistem manajemen aset yang tidak hanya andal dan akurat, tetapi juga mampu beradaptasi dengan lingkungan kerja yang serba cepat. Pengelolaan aset yang baik sangat penting untuk meningkatkan kelancaran operasional, mempertahankan nilai investasi, dan memastikan bahwa setiap aset yang dimiliki dipertanggungjawabkan.

Saat ini, proses inventarisasi dan pelacakan aset di PT Fujiyama Technology Solution masih mengandalkan metode semi-manual, yaitu dengan menggunakan *spreadsheet* yang diperbarui secara periodik. Proses ini sangat bergantung pada entri data manual oleh staf administrasi. Akibatnya, sering terjadi berbagai kendala seperti ketidaksesuaian data antara catatan dengan kondisi fisik aset di lapangan, kesulitan melacak lokasi dan status aset terutama dalam melacak riwayat perpindahan aset dari satu lokasi ke lokasi lain atau dari satu pengguna ke pengguna lainnya, serta memonitor jadwal perawatan preventif yang sering terlewat. Keterlambatan dalam pembaruan data juga meningkatkan risiko kehilangan aset atau ketidakjelasan status kepemilikan aset.

Berdasarkan permasalahan tersebut, maka teridentifikasi adanya kebutuhan mendesak untuk mengembangkan sebuah sistem yang dapat mengotomatisasi dan menyederhanakan proses inventarisasi di PT Fujiyama Technology Solution. Oleh karena itu, penulis berkeinginan untuk melakukan penelitian dengan judul "**Implementasi Aplikasi Inventaris dan Pelacakan Aset Perusahaan Menggunakan Data Matrix Berbasis Android**" sebagai solusi yang diharapkan mampu mengatasi permasalahan tersebut. Dengan menggunakan *smartphone* untuk memindai, aplikasi ini memungkinkan setiap aset diberi nama unik, dan setiap perubahan status, seperti peminjaman, pengembalian, atau perpindahan lokasi, dapat dicatat secara *real-time*. Sistem ini diharapkan dapat meminimalkan *human error*, mempercepat proses audit, dan menyediakan data aset yang terpusat, akurat, dan mudah diakses.

## 1.2 Identifikasi Masalah

Berdasarkan penjelasan yang telah diuraikan pada latar belakang, penulis dapat menemukan dan mengidentifikasi masalah yang ada pada PT Fujiyama Technology Solution, diantaranya yaitu:

1. Proses pendataan aset masih dilakukan secara semi-manual dengan menggunakan *spreadsheet* sehingga rentan terhadap kesalahan *input* data yang dilakukan oleh manusia (*human error*).
2. Masih terdapat kesulitan untuk memantau status, lokasi, dan riwayat penggunaan aset secara *real-time*.
3. Proses pemeriksaan fisik aset berjalan lambat dan tidak efisien karena masih menggunakan daftar *list asset* memerlukan verifikasi manual satu per satu.

4. Tidak adanya sistem terpusat yang menyebabkan data aset tidak konsisten dan sulit untuk diakses oleh pihak yang berkepentingan.
5. Meningkatnya risiko kehilangan atau kerusakan aset karena kurangnya kontrol dan pengawasan yang sistematis.
6. Jadwal perawatan preventif aset sering terlewat karena tidak adanya sistem pengingat otomatis.
7. Proses audit aset secara fisik memakan waktu lama dan tidak efisien, sehingga memperlambat proses pelaporan inventaris.

### 1.3 Rumusan Masalah

Penulis dapat merumuskan masalah berikut berdasarkan latar belakang masalah yang telah dijelaskan:

1. Bagaimana merancang dan membangun sebuah aplikasi inventaris berbasis Android yang mendukung antarmuka multibahasa (Inggris, Jepang) untuk mengakomodasi kebutuhan pengguna yang beragam?
2. Bagaimana mengimplementasikan teknologi *Data Matrix* untuk proses identifikasi, pembaruan, dan pelacakan status aset yang cepat dan akurat, serta mengintegrasikannya dengan pelacakan lokasi geografis (GPS)?
3. Bagaimana merancang arsitektur sistem dengan pemisahan peran (*role-based access*) yang jelas antara Admin, Staff, dan Employee, serta alur kerja yang spesifik untuk setiap peran?
4. Bagaimana mengintegrasikan Google Maps API untuk memvisualisasikan lokasi aset secara *real-time* dan melacak riwayat pergerakan aset guna meningkatkan pengawasan?
5. Bagaimana sistem dapat secara proaktif mengelola dan memberikan notifikasi untuk jadwal perawatan aset agar tidak terlewatkan?
6. Bagaimana aplikasi dapat mempercepat proses audit aset fisik dan menghasilkan laporan inventaris yang akurat secara efisien?

### 1.4 Ruang Lingkup

Untuk memastikan alur pembahasan yang sistematis dan terfokus, penulis membatasi cakupan masalah. Berdasarkan rumusan masalah dan keterbatasan waktu penelitian, maka fokus utama dari pembahasan ini adalah sebagai berikut:

1. Aplikasi ini dirancang dan dikembangkan pada platform Android menggunakan *framework* Flutter, dan *backend server* dibuat dengan Go (Golang) dan basis data PostgreSQL.

2. Sistem menerapkan lokalisasi (multibahasa) untuk antarmuka dan data master pada tiga bahasa: Inggris (default) dan Jepang.
3. Skripsi ini tidak membahas tentang perhitungan depresiasi nilai aset dan tidak diintegrasikan dengan sistem akuntansi perusahaan.
4. Data yang digunakan hanya pada perusahaan PT Fujiyama Technology Solutions dengan data *asset* yang sudah tervalidasi.

## **1.5 Tujuan Dan Manfaat**

Berdasarkan rumusan masalah yang telah diuraikan, penelitian ini memiliki beberapa tujuan yang ingin dicapai serta manfaat yang diharapkan dapat memberikan kontribusi.

### **1.5.1 Tujuan Penelitian**

Tujuan dari dilaksanakannya penelitian ini adalah:

1. Menghasilkan aplikasi inventaris dan pelacakan aset berbasis Android yang fungsional dan mendukung antarmuka multibahasa (Inggris dan Jepang) sesuai kebutuhan PT Fujiyama Technology Solution.
2. Mengimplementasikan pemindaian *Data Matrix* yang terintegrasi dengan GPS untuk identifikasi, pembaruan data, dan pelacakan lokasi aset secara akurat.
3. Membangun sistem dengan arsitektur berbasis peran yang jelas (Admin, Staff, Employee) untuk meningkatkan keamanan dan efisiensi operasional.
4. Mengintegrasikan Google Maps API untuk menyediakan visualisasi data lokasi dan riwayat pergerakan aset secara *real-time*.
5. Membangun fitur manajemen perawatan aset yang proaktif dengan sistem notifikasi otomatis untuk memastikan jadwal pemeliharaan tidak terlewatkan.
6. Mengembangkan fungsionalitas yang dapat mempercepat proses audit aset fisik dan menghasilkan laporan inventaris secara efisien.

### **1.5.2 Manfaat penelitian**

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Tersedianya sebuah aplikasi terpusat yang fungsional dan multibahasa untuk mengatasi kendala sistem manual sehingga proses pengelolaan aset menjadi lebih efisien.
2. Meningkatnya akurasi data aset melalui pemindaian *Data Matrix* dan GPS, yang secara signifikan meminimalkan risiko kehilangan aset dan kesalahan akibat human error.

3. Terciptanya alur kerja yang lebih aman dan efisien karena setiap pengguna hanya dapat mengakses data dan fitur yang sesuai dengan perannya.
4. Meningkatnya kemampuan pengawasan dan pengambilan keputusan melalui visualisasi lokasi aset secara *real-time* dan pelacakan riwayat pergerakannya.
5. Terjaganya kondisi dan nilai aset secara optimal karena jadwal perawatan dapat terpantau dan terlaksana tepat waktu berkat adanya notifikasi otomatis.
6. Proses audit dan pelaporan inventaris menjadi jauh lebih cepat dan akurat, sehingga menghemat waktu dan sumber daya perusahaan.

## **1.6 Metode Penelitian**

Untuk memastikan penelitian ini berjalan secara sistematis dan dapat mencapai tujuan yang telah ditetapkan, maka digunakan kerangka kerja metodologis yang terstruktur. Metode ini mencakup pemilihan jenis penelitian, teknik pengumpulan data, serta pendekatan dalam analisis dan perancangan sistem, yang akan diuraikan sebagai berikut:

### **1.6.1 Jenis penelitian**

Jenis penelitian yang digunakan adalah Penelitian dan Pengembangan atau *Research and Development* (R&D). Metode ini digunakan untuk menghasilkan sebuah produk tertentu, dalam hal ini aplikasi inventaris aset, dan menguji keefektifan produk tersebut untuk memecahkan masalah praktis di lapangan.

### **1.6.2 Metode pengumpulan data**

Pengumpulan data dilakukan dengan beberapa metode, yaitu:

1. Observasi: Melakukan pengamatan langsung terhadap proses manajemen aset yang sedang berjalan di PT Fujiyama Technology Solution untuk memahami alur kerja dan mengidentifikasi kendala.
2. Wawancara: Melakukan tanya jawab dengan staf administrasi atau manajer yang bertanggung jawab atas pengelolaan aset untuk mendapatkan pemahaman mendalam mengenai kebutuhan dan permasalahan yang dihadapi.
3. Studi Literatur: Mempelajari penelitian terdahulu, jurnal, dan dokumentasi teknis terkait manajemen aset, implementasi *QR Code*, *Data Matrix*, serta teknologi yang digunakan dalam pengembangan.

### **1.6.3 Metode Analisis dan Perancangan**

Metode pengembangan sistem yang digunakan adalah model *Waterfall*, yang memiliki tahapan-tahapan sekuensial dan sistematis sebagai berikut:

1. **Analisis Kebutuhan:** Menganalisis data yang terkumpul untuk mendefinisikan kebutuhan fungsional dan non-fungsional sistem.
2. **Perancangan Sistem:** Merancang arsitektur sistem, basis data, dan antarmuka pengguna (UI/UX). Dalam merancang sistem, penelitian ini mengadopsi pendekatan Analisis dan Perancangan Berorientasi Objek atau *Object-Oriented Analysis and Design* (OOAD). Pendekatan ini dipilih karena sangat sesuai untuk memodelkan entitas dunia nyata yang kompleks seperti aset, pengguna, dan lokasi menjadi objek-objek perangkat lunak yang terstruktur. Dengan demikian, sistem yang dihasilkan menjadi lebih modular, mudah dipelihara, dan fleksibel untuk pengembangan di masa depan. Perancangan akan digambarkan menggunakan *Unified Modeling Language* (UML) seperti *Use Case Diagram*, *Activity Diagram*, dan *Class Diagram*..
3. **Implementasi:** Proses penulisan kode (*coding*) untuk aplikasi *mobile* dan *backend server* berdasarkan desain yang telah dibuat.
4. **Pengujian:** Melakukan pengujian fungsionalitas (*Black Box*) dan pengujian sistem untuk memastikan aplikasi berjalan sesuai harapan dan bebas dari *bug*.
5. **Pemeliharaan:** Penyerahan aplikasi kepada perusahaan dan penyusunan dokumentasi untuk pemeliharaan di masa mendatang.

## 1.7 Sistematika Penulisan

Untuk memberikan gambaran yang jelas dan terstruktur mengenai keseluruhan isi laporan penelitian, maka sistematika penulisan disusun sebagai berikut:

### **BAB I            PENDAHULUAN**

Bab ini berisi gambaran umum mengenai penelitian, yang mencakup latar belakang masalah yang memicu penelitian, identifikasi dan perumusan masalah, batasan ruang lingkup agar penelitian tetap fokus, serta tujuan dan manfaat yang diharapkan dari hasil penelitian ini.

### **BAB II           LANDASAN TEORI**

Bab ini menguraikan berbagai teori yang menjadi fondasi dalam penelitian dan perancangan sistem. Pembahasan mencakup konsep-konsep dasar mengenai sistem informasi, manajemen aset, teknologi *Data Matrix*, serta teori-teori terkait arsitektur dan teknologi yang digunakan dalam pengembangan aplikasi.

### **BAB III          ANALISIS SISTEM YANG BERJALAN**



Bab ini berfokus pada analisis mendalam terhadap objek penelitian. Di dalamnya dibahas mengenai gambaran umum perusahaan, termasuk sejarah dan struktur organisasi, serta analisis terhadap tata laksana sistem manajemen aset yang saat ini berjalan. Bab ini juga merinci masalah yang dihadapi dan kebutuhan pengguna (*user requirement*) melalui elisitasi.

#### **BAB IV RANCANGAN SISTEM YANG DIUSULKAN**

Bab ini berfokus pada perancangan sistem baru yang diusulkan sebagai solusi untuk mengatasi masalah yang telah diidentifikasi pada bab sebelumnya. Pembahasan mencakup usulan prosedur baru, perancangan sistem yang digambarkan melalui diagram UML (Use Case Diagram, Activity Diagram, Sequence Diagram, dan Class Diagram), rancangan basis data, serta perancangan antarmuka pengguna (prototype). Bab ini juga akan menjelaskan spesifikasi teknis sistem yang dibutuhkan untuk implementasi.

#### **BAB V PENUTUP**

Bab ini merupakan bagian akhir dari laporan skripsi yang berisi simpulan dari keseluruhan hasil penelitian dan pembahasan. Simpulan akan menjawab rumusan masalah serta tujuan dan manfaat penelitian yang telah diuraikan pada BAB I. Selain itu, bab ini juga akan menyajikan saran-saran yang dapat digunakan untuk pengembangan sistem lebih lanjut di masa mendatang.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Teori-Teori Dasar/Umum**

##### **2.1.1 Konsep Dasar Sistem**

Menurut Sidik, hlm. (2022, hlm. 3), sistem pada dasarnya adalah kumpulan dari berbagai bagian dan unsur yang saling berinteraksi dan terkait satu sama lain untuk mencapai suatu tujuan bersama. Pemahaman mengenai konsep dasar sistem ini menjadi esensial karena aplikasi yang dikembangkan pada dasarnya adalah sebuah sistem yang dirancang untuk memecahkan masalah spesifik. Sistem ini terdiri dari berbagai komponen yang bekerja sama untuk mencapai tujuan tertentu..

##### **2.1.1.1 Definisi Sistem**

Menurut Sidik, hlm. (2022, hlm. 3), sistem adalah sekumpulan bagian serta unsur yang ada di dalamnya yang saling berinteraksi dan berhubungan satu sama lain. Hubungan tersebut membentuk keterikatan dan komitmen bersama untuk menghasilkan kekuatan yang besar demi tercapainya tujuan yang sama.

Menurut Choudhury dkk., hlm. (2021, hlm. 1), menyatakan bahwa sistem merupakan suatu himpunan atau *assemblage* dari komponen-komponen yang saling berinteraksi, saling berhubungan, dan saling bergantung. Komponen tersebut membentuk suatu kesatuan yang kompleks dan terintegrasi dengan tujuan yang jelas dan seragam.

Berdasarkan kedua definisi tersebut, dapat disimpulkan bahwa sistem bukan hanya sekadar kumpulan komponen yang tersusun tanpa keteraturan, melainkan sebuah kesatuan yang memiliki arah dan tujuan yang pasti. Komponen di dalamnya saling terhubung dan bekerja secara harmonis untuk mengolah masukan (*input*) menjadi keluaran (*output*) yang bermanfaat, melalui proses dan mekanisme yang terstruktur..

##### **2.1.1.2 Karakteristik Sistem**

Untuk dapat mengidentifikasi dan merancang sebuah sistem secara efektif, penting untuk memahami karakteristik yang melekat padanya. Menurut Ayumida dkk., hlm. (2021, hlm. 3) terdapat beberapa karakteristik yang mendefinisikan sebuah sistem:

1. **Komponen Sistem (*Components*):** Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Dalam penerapan umum, komponen dapat mencakup antarmuka pengguna, *server*, basis data, dan perangkat atau teknologi pendukung lainnya.
2. **Batasan Sistem (*Boundary*):** Merupakan area yang membatasi antara satu sistem dengan sistem lainnya atau dengan lingkungan luarnya. Batasan ini mendefinisikan ruang lingkup sistem. Misalnya, batasan dapat mencakup aturan akses pengguna yang menentukan siapa saja yang berhak berinteraksi dengan data atau sumber daya sistem.
3. **Lingkungan Luar Sistem (*Environment*):** Segala sesuatu di luar batas sistem yang dapat memengaruhi operasinya. Lingkungan luar dapat meliputi kebijakan organisasi, budaya kerja, regulasi eksternal, dan kebutuhan operasional yang dinamis yang memengaruhi adopsi serta keberhasilan sistem.
4. **Penghubung Sistem (*Interface*):** Media yang menghubungkan antar subsistem, memungkinkan sumber daya mengalir dari satu subsistem ke subsistem lainnya. Sebagai contoh, *Application Programming Interface* (API) dapat menjadi penghubung utama antara subsistem klien dan subsistem *server*, sementara teknologi identifikasi seperti pemindai kode dapat menjadi penghubung antara objek fisik dan representasi digitalnya.
5. **Masukan Sistem (*Input*):** Energi atau data yang dimasukkan ke dalam sistem untuk diproses. Contoh masukan dapat berupa hasil pemindaian kode identifikasi, *Input* data baru, atau permintaan laporan dari pengguna.
6. **Keluaran Sistem (*Output*):** Hasil dari pemrosesan masukan yang telah diolah menjadi bentuk yang berguna. Keluaran dapat berupa pembaruan data secara *real-time*, laporan terperinci, atau notifikasi otomatis.
7. **Pengolahan Sistem (*Process*):** Bagian yang mengubah masukan menjadi keluaran. Proses ini dapat terjadi pada *server* atau komponen pengolah lainnya, di mana logika bisnis dieksekusi untuk menghasilkan informasi yang dibutuhkan.
8. **Sasaran Sistem (*Objective*):** Setiap sistem memiliki tujuan atau sasaran yang menjadi motivasi utamanya. Sasaran ini umumnya mencakup peningkatan akurasi, efisiensi, dan akuntabilitas dalam pengelolaan sumber daya.

Memandang sebuah sistem melalui kerangka karakteristik ini mengubah perspektif dari sekadar “membangun aplikasi” menjadi “merancang sistem sosio-teknis yang terintegrasi”. Keberhasilan implementasi tidak hanya bergantung pada keunggulan teknis, tetapi juga pada kemampuan sistem untuk berinteraksi dan beradaptasi dengan lingkungan manusia maupun organisasi yang menggunakannya.

### 2.1.1.3 Klasifikasi Sistem

Sistem dapat diklasifikasikan berdasarkan berbagai sudut pandang untuk memahami sifat dan perilakunya. Mengutip dari penelitian Usnaini dkk., hlm. (2021, hlm. 1) dalam Jurnal Manajemen Informatika Jayakarta, klasifikasi sistem informasi secara umum dapat dibedakan menjadi:

1. Sistem Manual dan Sistem Terkomputerisasi: Sistem manual masih mengandalkan pencatatan atau pengolahan data secara langsung oleh manusia, sedangkan sistem terkomputerisasi sudah memanfaatkan teknologi komputer untuk mengotomatisasi berbagai proses.
2. Sistem Berbasis *Web* dan Sistem *Mobile*: Berdasarkan platform implementasinya, sistem dapat berjalan melalui aplikasi web yang diakses lewat browser, maupun aplikasi mobile yang dijalankan di perangkat bergerak.
3. Sistem standalone beroperasi secara independen tanpa keterhubungan dengan sistem lain, sedangkan sistem terintegrasi memiliki keterhubungan dengan sistem lain dalam suatu organisasi sehingga memungkinkan sinkronisasi dan pertukaran data.

Berdasarkan klasifikasi tersebut, suatu sistem tidak terbatas pada satu kategori saja, melainkan dapat memiliki beberapa karakteristik sekaligus. Misalnya, sebuah sistem bisa terkomputerisasi, berbasis mobile, dan terintegrasi pada saat yang sama.



Gambar 2. 1 Alur Sistem

Sumber: <https://www.brainkart.com>

## 2.1.2 Konsep Dasar Informasi

Dalam pengembangan sebuah sistem, pemahaman mengenai konsep informasi menjadi landasan yang krusial. Informasi merupakan hasil dari pengolahan data yang terorganisir sehingga memiliki nilai dan makna bagi penerimanya, terutama dalam konteks pengambilan keputusan. Menurut Fahmi dkk., hlm. (2024, hlm. 1), sistem informasi secara esensial adalah rangkaian komponen yang bekerja untuk mengumpulkan, mengelola, memproses, dan menyebarkan informasi guna mendukung fungsi-fungsi vital dalam sebuah organisasi

### 2.1.2.1 Definisi Data dan Informasi

Dalam Dalam konteks sistem informasi modern, terdapat distinsi fundamental antara data dan informasi yang perlu dipahami dengan baik. Data merujuk pada fakta mentah atau elemen dasar yang belum mengalami pengolahan, sementara informasi merupakan hasil transformasi data yang telah diproses sedemikian rupa sehingga memberikan makna dan nilai bagi penggunanya (Purnomo & Alijoyo, 2024, hlm. 1–2).

Menurut Fahmi dkk., hlm. (2024, hlm. 1–2), sistem informasi adalah serangkaian komponen yang bekerja bersama untuk mengumpulkan, mengelola, menyimpan, memproses, dan menyebarkan informasi yang diperlukan untuk mendukung pengambilan keputusan dalam suatu organisasi. Sejalan dengan itu, Purnomo & Alijoyo, hlm. (2024, hlm. 3) menyatakan bahwa dengan sistem informasi yang efisien, sebuah organisasi dapat mengelola data dengan lebih baik, membuat keputusan yang lebih tepat, dan meningkatkan kinerja secara keseluruhan.

Karakteristik utama dari sistem yang efektif adalah kemampuannya dalam melakukan transformasi nilai, bukan sekadar penyimpanan data. Sistem yang dirancang dengan pendekatan analitik dapat berfungsi sebagai *engine intelligence* yang mampu menghasilkan wawasan bisnis untuk mendukung pengambilan keputusan strategis (Fahmi dkk., 2024, hlm. 2). Dengan demikian, sistem tidak hanya menjadi repository digital, melainkan juga platform yang mengubah raw data menjadi actionable intelligence untuk mengoptimalkan operasional organisasi (Purnomo & Alijoyo, 2024, hlm. 3).

### 2.1.2.2 Kualitas Informasi

Nilai dari sebuah informasi ditentukan oleh kualitasnya, yang menjadi parameter fundamental dalam pengembangan sistem informasi yang efektif. Berdasarkan model DeLone dan McLean yang telah banyak diadopsi dalam penelitian sistem informasi terkini, untuk menilai suatu kualitas informasi dapat menggunakan lima dimensi yaitu akurasi, ketepatan waktu, kelengkapan, relevansi, dan konsistensi (Pasaribu, 2021, hlm. 4)

Dari berbagai dimensi tersebut, terdapat tiga dimensi utama yang dianggap paling krusial untuk mendukung operasional dan pengambilan keputusan dalam sebuah organisasi:

1. Akurat (*Accurate*): Informasi harus bebas dari kesalahan, tidak bias, dan tidak menyesatkan sehingga dapat merepresentasikan keadaan yang sebenarnya. Akurasi menjadi aspek fundamental yang menentukan reliabilitas sebuah sistem informasi.
2. Tepat Waktu (*Timely*): Informasi harus tersedia dan dapat diakses saat dibutuhkan. Informasi yang usang atau terlambat diterima akan kehilangan nilainya, terutama dalam mendukung pengambilan keputusan yang bersifat dinamis dan sensitif terhadap waktu.
3. Relevan (*Relevant*): Informasi harus memberikan manfaat yang sesuai dengan kebutuhan penggunanya. Relevansi memastikan bahwa informasi yang disajikan dapat menjawab pertanyaan, menyelesaikan masalah, atau mendukung tugas spesifik dari pengguna tersebut.

### 2.1.3 Konsep Dasar Sistem Informasi

Setelah memahami konsep sistem dan informasi secara terpisah, langkah selanjutnya adalah menggabungkan keduanya dalam kerangka sistem informasi. Sebuah sistem informasi pada dasarnya adalah fondasi teknologi dan prosedural yang memungkinkan sebuah organisasi untuk mengubah data mentah menjadi informasi yang bermakna untuk mencapai tujuannya (Rajan & Sulthana, 2022, hlm. 2).

#### 2.1.3.1 Definisi Sistem Informasi

Sistem Sistem informasi dapat didefinisikan sebagai perpaduan antara konsep sistem dan informasi yang saling terintegrasi untuk mendukung operasional organisasi. Berdasarkan penelitian terkini, sistem informasi adalah serangkaian komponen yang bekerja bersama untuk mengumpulkan, mengelola, menyimpan,

memproses, dan menyebarkan informasi yang diperlukan untuk mendukung pengambilan keputusan dalam suatu organisasi atau entitas (Fahmi dkk., 2024, hlm. 1).

Dalam konteks pengembangan sistem akademik, sistem informasi berperan penting dalam menyampaikan informasi dan mengelola data organisasi. Sebagaimana dijelaskan dalam penelitian Fahmi dkk., hlm. (2024, hlm. 1) yang dikutip dalam *Journal of Information System Applied Management Accounting and Research*, sistem informasi akademik berbasis *web* dapat digunakan untuk mengelola dan menampilkan informasi seperti data pengguna, data subjek, jadwal, dan informasi terkait lainnya secara efektif dan efisien.

Berdasarkan konsep tersebut, aplikasi inventaris dan pelacakan aset ini dapat diposisikan lebih dari sekadar sistem pencatatan. Aplikasi ini berfungsi sebagai Decision Support System (DSS) atau Sistem Pendukung Keputusan yang terspesialisasi untuk manajemen aset perusahaan. Sistem ini secara langsung menyediakan data yang diperlukan untuk menjawab pertanyaan-pertanyaan manajerial krusial, seperti analisis kebutuhan pengadaan aset baru, identifikasi aset yang memerlukan perawatan preventif, atau tracking utilisasi aset tertentu. Dengan demikian, sistem ini menghasilkan *output* berupa *intelligence* aktif untuk mendukung pengambilan keputusan manajerial (Fahmi dkk., 2024, hlm. 1).

### 2.1.3.2 Komponen Sistem Informasi

Dalam konsep sistem informasi, sebuah sistem terdiri dari komponen-komponen yang saling terintegrasi untuk mencapai tujuan organisasi. Menurut Rajan & Sulthana, hlm. (2022, hlm. 7), komponen utama sistem informasi terdiri dari enam blok yang saling berkaitan, yaitu blok masukan, blok model, blok keluaran, blok teknologi, blok basis data, dan blok pengendalian yang dikenal sebagai *information system building blocks*.

Model *building blocks* ini menyediakan kerangka kerja yang sangat baik, keenam blok tersebut dapat diuraikan sebagai berikut:

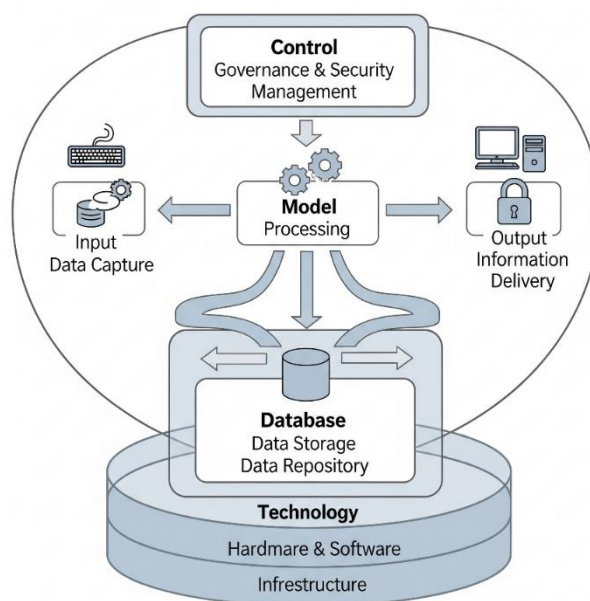
1. Blok Masukan (*Input Block*): Blok ini mencakup metode dan media yang digunakan untuk menangkap dan mengumpulkan data mentah dari lingkungan internal maupun eksternal organisasi.
2. Blok Model (*Model Block*): Blok model terdiri dari kombinasi prosedur, aturan logis, dan algoritma yang mengubah data masukan menjadi informasi yang

berguna. Komponen ini merepresentasikan business logic dan proses bisnis yang menjadi inti dari operasional sistem informasi dalam organisasi.

3. Blok Keluaran (*Output Block*): Blok ini merupakan hasil akhir dari pemrosesan sistem informasi yang disajikan kepada pengguna dalam berbagai format. Keluaran ini dapat berupa laporan ringkasan, dasbor analitik, notifikasi otomatis, atau dokumen yang dapat diekspor untuk mendukung pengambilan keputusan.
4. Blok Teknologi (*Technology Block*): Blok teknologi merupakan seluruh infrastruktur yang mendasari operasional sistem informasi. Komponen ini mencakup perangkat keras (seperti komputer dan *server*), perangkat lunak (seperti sistem operasi dan basis data), serta teknologi komunikasi dan jaringan yang memungkinkan sistem berjalan secara efektif.
5. Blok Basis Data (*Database Block*): Blok Blok basis data adalah kumpulan data terorganisir yang disimpan secara sistematis untuk mendukung operasional sistem informasi. Komponen ini berfungsi sebagai repositori utama yang menyimpan seluruh aset informasi milik organisasi.
6. Blok Kendali (*Control Block*): Blok ini dirancang untuk menjaga integritas, keamanan, dan keandalan sistem informasi dari berbagai ancaman dan kesalahan. Komponen ini sangat krusial untuk memastikan hanya pengguna yang berwenang yang dapat mengakses dan memodifikasi data, serta menjamin akurasi dan keamanan informasi organisasi.

Model building blocks ini berfungsi sebagai *framework* teoretis yang menyediakan landasan akademis untuk keseluruhan arsitektur sistem, memastikan bahwa implementasi praktis didasarkan pada teori sistem informasi yang mapan.





Gambar 2. 2 Building Blocks  
Sumber: <https://link.springer.com>

#### 2.1.4 Metodologi Penelitian (*Research and Development*)

Sebagaimana telah disebutkan pada Bab I, jenis penelitian yang digunakan adalah Penelitian dan Pengembangan atau *Research and Development* (R&D). Menurut Nur Elfi Husda dkk., hlm. (2023, hlm. 11), metode penelitian R&D didefinisikan sebagai cara untuk membuat produk tertentu dan menguji keefektifan produk tersebut. Produk yang dikembangkan tidak selalu harus berupa perangkat keras (*hardware*), tetapi bisa juga berupa perangkat lunak (*software*) seperti aplikasi.

Tujuan dari penelitian R&D adalah untuk menjadi penghubung atau pemutus kesenjangan antara penelitian dasar yang bersifat teoretis dengan penelitian terapan yang bersifat praktis. Dalam konteks penelitian ini, R&D digunakan untuk menghasilkan produk berupa aplikasi inventaris aset yang diharapkan dapat meningkatkan efektivitas dan produktivitas di PT Fujiyama Technology Solution.

Model pengembangan yang diadopsi dalam penelitian ini mengacu pada model yang dijelaskan oleh (Nur Elfi Husda dkk., 2023, hlm. 44), yang terdiri dari serangkaian langkah sistematis. Langkah-langkah tersebut disesuaikan dengan kebutuhan pengembangan aplikasi ini sebagai berikut:

1. Research and Information Collecting (Penelitian dan Pengumpulan Data): Tahap ini merupakan langkah awal untuk mengumpulkan informasi melalui studi literatur terkait penelitian sejenis, serta melakukan pengumpulan data primer di

lapangan melalui observasi dan wawancara untuk memahami permasalahan yang ada.

2. Planning (Perencanaan): Berdasarkan data yang terkumpul, dilakukan perencanaan yang mencakup perumusan keterampilan yang dibutuhkan, penentuan tujuan yang ingin dicapai, dan pelaksanaan studi kelayakan untuk memastikan proyek dapat dijalankan.
3. Develop Preliminary Form of Product (Pengembangan Bentuk Awal Produk): Pada tahap ini, hasil perencanaan diterjemahkan ke dalam bentuk awal produk. Ini mencakup perancangan arsitektur sistem, desain basis data, dan antarmuka pengguna (UI/UX) aplikasi.
4. Preliminary Field Testing dan Main Field Testing (Uji Coba Lapangan): Produk awal yang telah dikembangkan kemudian diuji coba dalam skala terbatas untuk mendapatkan umpan balik awal, yang dilanjutkan dengan uji coba dalam skala yang lebih besar untuk validasi fungsionalitas. Dalam penelitian ini, tahap ini diwujudkan melalui pengujian *Black Box* pada aplikasi.
5. Product Revision (Revisi Produk): Berdasarkan hasil dari uji coba lapangan, dilakukan revisi dan perbaikan terhadap produk untuk menyempurnakan fungsionalitas dan mengatasi kekurangan yang ditemukan.
6. Dissemination and Implementation (Diseminasi dan Implementasi): Merupakan tahap akhir di mana produk yang sudah final diserahkan dan diimplementasikan di lingkungan perusahaan untuk digunakan dalam operasional sehari-hari.
7. Dengan demikian, metode R&D memberikan kerangka kerja yang solid untuk keseluruhan proses penelitian, sementara metode pengembangan sistem seperti Waterfall diterapkan secara lebih spesifik pada tahap ke-3 dan ke-4, yaitu saat pengembangan dan pengujian produk perangkat lunak.

## **2.1.5 Metodologi Pengembangan Sistem**

### **2.1.5.1 Definisi SDLC (*System Development Life Cycle*)**

Menurut Prasetyo dkk., hlm. (2024, hlm. 3) System Development Life Cycle atau yang biasa disebut SDLC adalah pendekatan bertahap untuk menganalisa dan membuat rancangan sistem menggunakan siklus yang spesifik terhadap kegiatan penggunaannya. SDLC merupakan pusat pengembangan sistem informasi yang efisien. SDLC adalah proses memahami bagaimana suatu sistem informasi dapat mendukung kebutuhan bisnis, merancang sistem, membangun sistem yang setelah itu diberikan kepada penggunaannya.

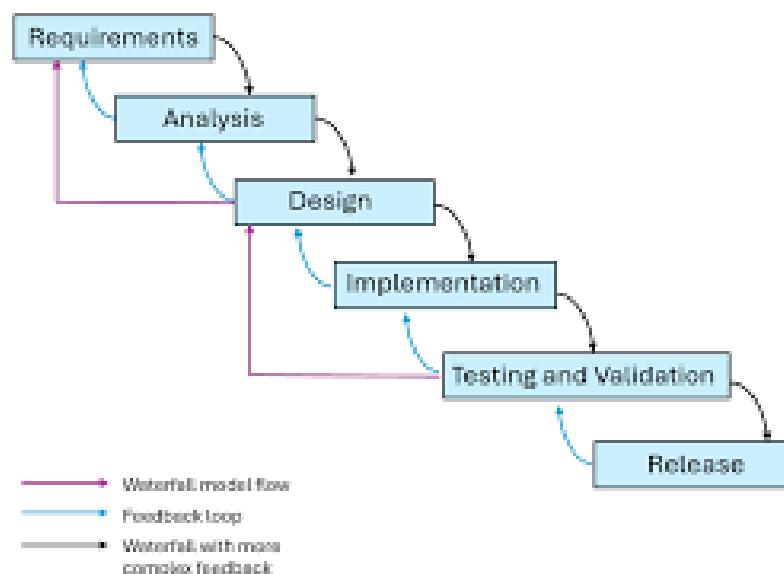
### 2.1.5.2 Metode Waterfall

Menurut Christian & Voutama, hlm. (2024, hlm. 3) Metode Waterfall adalah metode pengembangan sistem yang terstruktur di mana setiap tahapan dilakukan secara bertahap dan tidak boleh dilanjutkan sampai tahapan sebelumnya selesai. Metode ini memiliki beberapa keunggulan, termasuk membuat proses perancangan sistem lebih mudah karena tahapan-tahap ini harus dilakukan secara bertahap sampai dengan selesai sehingga proses penelitian tidak terganggu.

Tahapan-tahapan dalam metode waterfall, sebagai berikut:

1. Analisis Kebutuhan Sistem Setelah menyelesaikan analisis kebutuhan sistem, langkah pertama adalah menemukan masalah yang ada, seperti bagaimana menyesuaikan sistem kegiatan Merdeka Belajar Kampus Merdeka (MBKM) dengan tujuan pembuatan dan mengurangi hambatan yang mungkin muncul selama proses pembuatan sistem. Proses pengumpulan data yang dikenal sebagai analisis kebutuhan melibatkan identifikasi, wawancara, atau penelitian literatur yang berkaitan dengan pengembangan sistem informasi. Data yang dikumpulkan kemudian diolah dan dianalisis untuk mendapatkan data atau informasi lengkap tentang spesifikasi kebutuhan pengguna perangkat lunak yang akan dikembangkan.
2. Sistem dan Software Desain Sesuai dengan tahapan penelitian, perancangan database dan sistem dirancang pada tahap perancangan. Diharapkan, sebagai hasil dari tahapan perancangan sistem, proses pembuatan sistem informasi MBKM akan menjadi lebih mudah dan lebih terarah ke tujuan. Setelah memilih sistem yang akan digunakan, desain tampilan dirancang dengan menarik dan mudah digunakan bahkan untuk orang yang tidak terbiasa. Perancangan database dilakukan untuk menentukan struktur, table, dan field database. Struktur database yang dirancang untuk memenuhi kebutuhan sistem informasi MBKM dan membantu logika pemrograman menyelesaikan masalah. Pada tahap ini, informasi tentang spesifikasi kebutuhan dari tahap analisis persyaratan dievaluasi. Informasi ini kemudian diterapkan pada desain pengembangan. Perancangan desain dilakukan untuk membantu memberikan gambaran menyeluruh tentang apa yang harus dilakukan untuk arsitektur sistem perangkat lunak secara keseluruhan.

3. Pembuatan dan Implementasi Pada tahap ini, implementasi perencanaan dan desain sistem informasi dilakukan. Pengkodean atau perancangan sistem informasi menjadi suatu program dilakukan untuk memenuhi kebutuhan analisis yang dilakukan sebelumnya. Selanjutnya, perangkat lunak dibagi menjadi modul-modul kecil yang akan digabungkan di tahap berikutnya. Selain itu, sebagai bagian dari tahap implementasi sistem informasi yang dibuat, modul yang telah dibuat diuji dan diperiksa untuk memastikan bahwa mereka memenuhi persyaratan.
4. Integration Sistem Testing Setelah proses ini selesai, modul atau unit yang dikembangkan diintegrasikan ke dalam sistem informasi MBKM secara keseluruhan. Proses pengujian sistem informasi dilakukan secara menyeluruh dengan melakukan berbagai percobaan pada sistem informasi yang digunakan untuk menemukan kesalahan atau kesalahan. Tahapan ini dilakukan untuk memastikan bahwa sistem informasi dibuat dengan benar dan sesuai dengan proses bisnis yang telah dipelajari sebelumnya.
5. Operasional dan Maintenance Pemeliharaan sistem dilakukan untuk memperbaiki kesalahan saat implementasi unit sistem dan peningkatan sesuai dengan kebutuhan pengguna. Pemeliharaan sistem juga dilakukan untuk menjalankan tahapan operasional sistem informasi MBKM yang telah ada oleh pengguna dan untuk mengidentifikasi kesalahan pemrograman.



Gambar 2. 3 Metode Waterfall  
Sumber: <https://www.anao.gov.au>

## 2.1.6 Konsep Perancangan Berorientasi Objek

### 2.1.6.1 Analisis dan Perancangan Berorientasi Objek (OOAD)

Analisis dan Perancangan Berorientasi Objek (*Object-Oriented Analysis and Design* - OOAD) adalah pendekatan teknis dalam rekayasa perangkat lunak yang memodelkan sistem sebagai kumpulan objek yang saling berinteraksi untuk menyelesaikan masalah tertentu Weifan Liu, hlm. (2022, hlm. 1). OOAD merupakan metode populer untuk menganalisis, merancang aplikasi, sistem, atau bisnis dengan menerapkan paradigma berorientasi objek dan pemodelan visual sepanjang siklus pengembangan perangkat lunak untuk meningkatkan komunikasi pemangku kepentingan dan kualitas produk. Pendekatan OOAD terdiri dari dua fase utama yang saling berkaitan:

1. **Analisis Berorientasi Objek (OOA):** Fase analisis berfokus pada pemahaman dan pemodelan domain masalah tanpa mempertimbangkan batasan implementasi teknis. Tujuan utamanya adalah mengidentifikasi objek-objek fundamental dari dunia nyata, atribut-atributnya, dan perilakunya berdasarkan kebutuhan fungsional sistem (Weifan Liu, 2022, hlm. 1). Perbedaan utama antara analisis berorientasi objek dengan pendekatan analisis tradisional terletak pada cara struktur kebutuhan diorganisir di sekitar objek, yang mencakup perilaku dan keadaan yang dipola setelah item nyata yang berinteraksi dengan sistem di dunia nyata.
2. **Perancangan Berorientasi Objek (OOD):** Fase perancangan mentransformasikan model konseptual hasil OOA menjadi model teknis yang siap diimplementasikan. OOD menerapkan batasan implementasi pada model konseptual, termasuk platform perangkat keras dan perangkat lunak, kebutuhan kinerja, penyimpanan permanen, transaksi, kegunaan sistem, serta batasan waktu dan anggaran (Weifan Liu, 2022, hlm. 1). Fase ini mendefinisikan bagaimana objek-objek berinteraksi dalam perangkat lunak, struktur kelas, dan arsitektur sistem secara keseluruhan.

Pendekatan OOAD didasari oleh tiga prinsip fundamental:

1. **Enkapsulasi (*Encapsulation*):** Merupakan konsep membungkus data (atribut) dan metode (perilaku) yang beroperasi pada data tersebut ke dalam satu unit tunggal yang disebut objek. Enkapsulasi menyembunyikan detail implementasi internal dari objek lain dan hanya mengekspos fungsionalitas yang diperlukan

melalui antarmuka, sehingga meningkatkan modularitas dan keamanan data (Weifan Liu, 2022, hlm. 1).

2. Pewarisan (*Inheritance*): Mekanisme yang memungkinkan sebuah kelas baru (kelas anak) untuk mewarisi atribut dan metode dari kelas yang sudah ada (kelas induk). Pewarisan mendorong penggunaan kembali kode (*code reusability*) dan menciptakan hierarki kelas yang logis, sehingga mengurangi kompleksitas pengembangan sistem.
3. Polimorfisme (*Polymorphism*): Berasal dari bahasa Yunani yang berarti "banyak bentuk". Prinsip ini memungkinkan objek dari kelas yang berbeda untuk merespons pesan atau pemanggilan metode yang sama dengan cara yang berbeda, memberikan fleksibilitas dalam implementasi dan memungkinkan sistem untuk berperilaku sesuai konteks objek yang spesifik.

#### 2.1.6.2 Definisi UML (Unified Modeling Language)

UML (*Unified Modeling Language*) adalah sebuah bahasa berbasis grafik atau gambar yang digunakan untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan sistem dalam pengembangan perangkat lunak berbasis *Object-Oriented* (OO). UML merupakan metode pemodelan visual yang menjadi sarana perancangan sistem berorientasi objek.

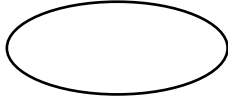

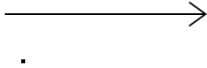
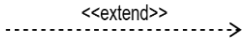
Definisi UML sendiri adalah bahasa standar yang digunakan untuk visualisasi, perancangan, dan pendokumentasian sistem perangkat lunak. Saat ini, UML telah menjadi bahasa standar internasional, dan bukan hanya sekadar bahasa pemrograman visual. UML juga dapat diintegrasikan secara langsung dengan berbagai bahasa pemrograman, seperti Java, C++, Visual Basic, bahkan terhubung dengan *object-oriented database* (Willyana, 2022a, hlm. 2).

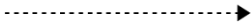
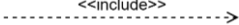
#### 2.1.6.3 Jenis-jenis Diagram UML

Penelitian ini akan memanfaatkan beberapa jenis diagram UML untuk memvisualisasikan rancangan sistem, sebagaimana diamanatkan dalam metode penelitian menurut (Willyana, 2022a, hlm. 2).

1. **Use Case Diagram:** Yaitu salah satu jenis diagram pada UML yang menggambarkan interaksi antara sistem dan aktor, use case diagram juga dapat men-deskripsikan tipe interaksi antara si pemakai sistem dengan sistemnya.



Tabel 2. 1 Simbol-simbol Use Case Diagram

Simbol	Nama	Deskripsi
	Use Case	Merepresentasikan sebuah fungsionalitas spesifik yang disediakan oleh sistem untuk mencapai tujuan tertentu bagi aktor.
	Aktor	Mewakili peran pengguna, sistem eksternal, atau perangkat keras yang berinteraksi secara langsung dengan sistem.
	Asosiasi	Garis yang menghubungkan Aktor dengan Use Case, menandakan bahwa aktor tersebut terlibat dalam fungsionalitas yang digambarkan.
	Ekstensi/ <i>Extend</i>	Relasi yang menunjukkan fungsionalitas opsional atau tambahan. Use case yang menjadi ekstensi dapat berjalan secara mandiri.

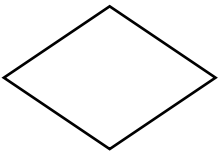


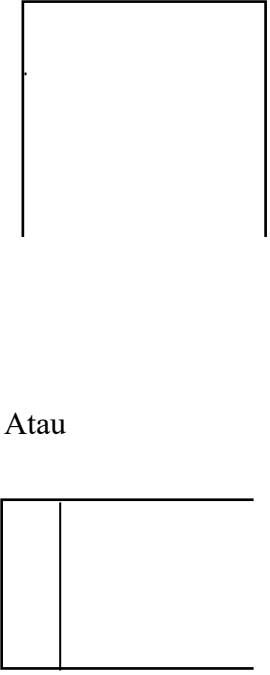
	Generalisasi/ <i>Generalization</i>	Hubungan hierarkis antara elemen yang lebih umum (induk) dengan yang lebih spesifik (anak), di mana anak mewarisi properti induk.
	Menggunakan <i>/ Include /</i> <i>uses</i>	Relasi yang menunjukkan bahwa sebuah use case memerlukan fungsionalitas dari use case lain untuk dapat dieksekusi.

2. **Activity Diagram:** Menggambarkan rangkaian aliran dari aktivitas dan interaksi beberapa use case. Diagram ini sangat mirip dengan flowchart karena memodelkan workflow dari suatu aktifitas ke aktifitas yang lainnya. Pembuatan activity diagram pada awal pemodelan proses dapat membantu memahami keseluruhan proses.

Tabel 2. 2 Simbol-simbol Activity Diagram

Gambar	Nama	Deskripsi
	Status Awal	Titik mula dari keseluruhan alur kerja dalam diagram. Setiap diagram aktivitas hanya memiliki satu status awal.
	Aktivitas	Representasi dari sebuah tugas atau langkah spesifik yang dilakukan dalam alur kerja.

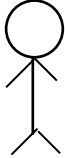

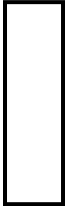
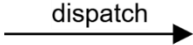


	Percabangan/ <i>decision</i>	Titik dalam alur di mana terdapat beberapa kemungkinan jalur berdasarkan kondisi tertentu.
	Penggabungan/ <i>join</i>	Titik di mana beberapa alur kerja paralel atau alternatif yang sebelumnya terpisah, disatukan kembali menjadi satu alur.
	Status Akhir	Titik akhir yang menandakan selesainya keseluruhan alur kerja yang digambarkan.
 <p>Atau</p>	Swimlane	Partisi visual yang mengelompokkan aktivitas berdasarkan aktor atau unit yang bertanggung jawab atas pelaksanaannya.

3. **Sequence diagram:** adalah jenis diagram yang memvisualisasikan interaksi antar objek berdasarkan urutan waktu. Diagram ini menggunakan garis vertikal yang disebut *lifeline* untuk mewakili keberadaan sebuah objek selama proses

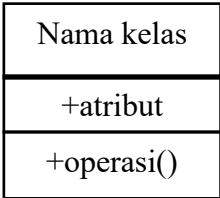

berlangsung, dan panah horizontal yang disebut *message* untuk menggambarkan komunikasi, seperti pemanggilan metode atau pertukaran informasi, antar objek tersebut.


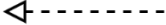
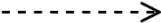
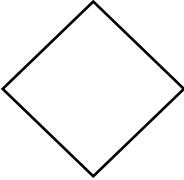
Tabel 2. 3 Simbol-Simbol Sequence Diagram

Simbol	Nama	Deskripsi
 atau <div style="border: 1px solid black; padding: 2px; display: inline-block;">Nama aktor</div>	Aktor	Entitas eksternal seperti pengguna, perangkat, atau sistem lain yang memulai atau berpartisipasi dalam urutan interaksi.
	Garis hidup / <i>lifeline</i>	Garis vertikal yang merepresentasikan eksistensi sebuah objek selama periode interaksi.
<div style="border: 1px solid black; padding: 2px; display: inline-block;">Nama objek: nama kelas</div>	Objek	Merepresentasikan sebuah instansi dari kelas yang terlibat dalam interaksi.
	Waktu aktif	Persegi panjang pada lifeline yang menandakan periode waktu saat objek sedang aktif memproses sebuah tugas.
	Pesan	Pesan yang mengindikasikan bahwa sebuah objek membuat objek lainnya.

4. **Class Diagram:** Diagram ini merepresentasikan struktur statis sistem dengan menunjukkan kelas-kelas, atribut, metode, dan hubungan antar kelas.

Tabel 2. 4 Simbol-simbol Class Diagram

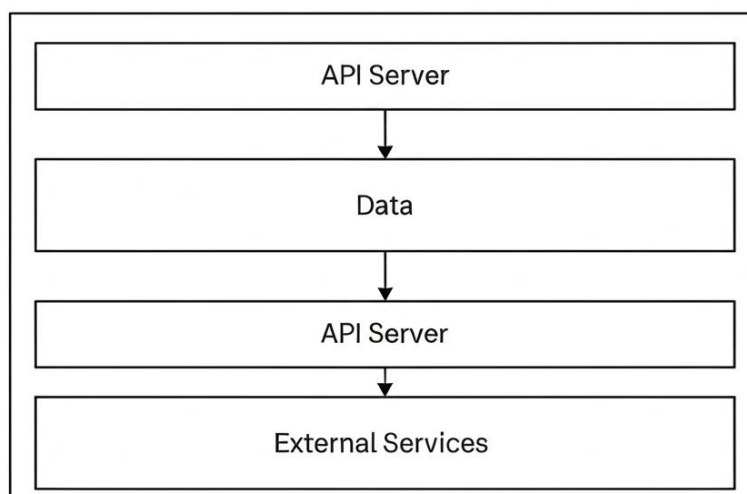
Gambar	Nama	Deskripsi
	Kelas / <i>Class</i>	<p>Cetak biru (<i>blueprint</i>) untuk membuat objek.</p> <p>Merepresentasikan entitas utama dalam sistem, lengkap dengan atribut dan operasinya.</p>
	Antarmuka / <i>interface</i>	<p>Sebuah kontrak yang mendefinisikan sekumpulan operasi atau metode tanpa implementasi. Kelas yang mengimplementasikan antarmuka ini wajib menyediakan implementasi untuk semua metodenya.</p>

	Asosiasi / <i>Association</i>	Hubungan struktural yang menunjukkan bahwa objek dari satu kelas terhubung dengan objek dari kelas lain.
	Asosiasi berarah / <i>directed association</i>	Hubungan di mana satu kelas mengetahui atau menggunakan kelas lain, tetapi tidak sebaliknya.
	Ketergantungan / <i>Dependency</i>	Hubungan di mana perubahan pada satu kelas dapat memengaruhi kelas lain, tanpa adanya hubungan struktural yang kuat
	Agregasi / <i>Agregation</i>	Bentuk khusus dari asosiasi "memiliki-sebuah" ( <i>has-a</i> ), di mana satu kelas (keseluruhan) terdiri dari kelas lain (bagian), namun bagian tersebut dapat ada secara mandiri.

## 2.1.7 Konsep Dasar Jaringan dan Arsitektur

### 2.1.7.1 Arsitektur Client-Server

Menurut Mukti dkk., hlm. (2023, hlm. 3) Arsitektur client-server adalah pendekatan di mana aplikasi terbagi menjadi dua bagian: sisi klien (client-side) dan sisi server (server-side). Sisi klien berjalan di perangkat pengguna dan bertanggung jawab untuk tampilan dan interaksi antarmuka pengguna, sedangkan sisi server berjalan di server dan bertanggung jawab untuk pemrosesan logika bisnis, pengolahan data, dan penyimpanan.



Gambar 2. 4 *Client Server*

Sumber: <https://www.tutorialspoint.com>

### 2.1.7.2 Definisi API (Application Programming Interface)

Menurut Ferry Gunawan dkk., hlm. (2024, hlm. 3) *Application Programming Interface* (API) adalah seperangkat aturan dan protokol yang memungkinkan aplikasi perangkat lunak yang berbeda untuk berkomunikasi satu sama lain. API bertindak sebagai perantara yang mendefinisikan metode komunikasi yang diizinkan antara klien dan server.

### 2.1.7.3 Konsep REST (Representational State Transfer)

REST adalah gaya arsitektur yang banyak digunakan untuk merancang API jaringan. Menurut Dalimunthe dkk., hlm. (2023, hlm. 1–2), "An Application Programming Interface (API) that follows the REST style is called a RESTful API". RESTful API menggunakan metode HTTP standar (seperti GET, POST, PUT, DELETE) untuk berinteraksi dengan sumber daya. Mencatat bahwa arsitektur ini menggunakan skema, kueri, dan *resolver* untuk komunikasi data berbasis HTTP.

Salah satu prinsip utamanya adalah *statelessness*, yang berarti setiap permintaan dari klien ke *server* harus berisi semua informasi yang diperlukan untuk memahami dan memproses permintaan tersebut, tanpa *server* perlu menyimpan status klien apa pun dari permintaan sebelumnya.

## **2.1.8 Konsep Internasionalisasi dan Lokalisasi Perangkat Lunak**

Di era global, permintaan untuk dukungan multibahasa dalam perangkat lunak terus meningkat, mendorong lahirnya industri lokalisasi sejak tahun 1980-an (Pirrone & D'Ulizia, 2024, hlm. 1). Untuk menghasilkan produk digital yang dapat diterima secara global, diperlukan dua proses fundamental yang saling terkait, yaitu internasionalisasi dan lokalisasi. Proses penerjemahan sederhana tidak lagi memadai karena perangkat lunak modern mengandung teks non-linear dan konten multimedia yang kompleks (Pirrone & D'Ulizia, 2024, hlm. 1–2).

### **2.1.8.1 Internasionalisasi (i18n)**

Internasionalisasi adalah tahap rekayasa perangkat lunak yang berfokus pada perancangan dan pengembangan produk agar mudah dilokalkan untuk target audiens yang beragam tanpa perlu rekayasa ulang (Pirrone & D'Ulizia, 2024, hlm. 10). Pada intinya, ini adalah proses "mengaktifkan" perangkat lunak untuk lokalisasi (Pirrone & D'Ulizia, 2024, hlm. 16). Praktik utamanya meliputi:

1. Memisahkan teks antarmuka (UI strings) dari kode sumber (source code).
2. Memberikan dukungan untuk berbagai set karakter (seperti Unicode).
3. Merancang antarmuka yang dapat mengakomodasi panjang teks yang berbeda-beda antar bahasa.
4. Mengabstraksi format data yang spesifik secara budaya, seperti tanggal, waktu, angka, dan mata uang.

Internasionalisasi yang baik merupakan prasyarat krusial, karena internasionalisasi yang buruk akan berdampak besar pada tahap lokalisasi, sering kali menyebabkan peningkatan biaya dan waktu (Pirrone & D'Ulizia, 2024, hlm. 16).

### **2.1.8.2 Lokalisasi (L10n)**

Lokalisasi adalah proses mengambil produk yang telah diinternasionalisasi dan mengadaptasinya secara linguistik dan budaya agar sesuai dengan target lokal (wilayah/negara dan bahasa) di mana produk tersebut akan digunakan dan dijual (Pirrone & D'Ulizia, 2024, hlm. 2). Proses ini lebih dari sekadar penerjemahan. Lokalisasi bertujuan untuk memberikan sebuah produk

"tampilan dan nuansa seperti produk yang dibuat secara nasional" (Pirrone & D'Ulizia, 2024, hlm. 2). Selain isu linguistik, lokalisasi juga harus mempertimbangkan masalah sosio-kultural dan teknologi untuk merilis produk digital yang terasa familiar bagi pengguna lokal (Pirrone & D'Ulizia, 2024, hlm. 2).

## **2.2 Teori-Teori Khusus**

### **2.2.1 Konsep Dasar Manajemen Aset**

#### **2.2.1.1 Definisi Aset dan Inventaris**

Penting untuk membedakan antara aset dan inventaris. Aset adalah sumber ekonomi yang diharapkan memberikan manfaat usaha dikemudian hari, aset dapat berbentuk uang, barang ataupun sumber daya manusia (Tuti Amiasih & Andiani, 2022, hlm. 1). Di sisi lain, inventaris, menurut Christian & Voutama, hlm. (2024, hlm. 2), adalah daftar yang memuat semua barang milik suatu organisasi, baik itu perusahaan, sekolah, kantor, ataupun instansi pemerintah.

Perbedaan ini bersifat mendasar. Inventarisasi adalah kegiatan mencatat "apa yang dimiliki", sedangkan manajemen aset merupakan disiplin yang lebih luas yang berfokus pada pengelolaan "nilai dari apa yang dimiliki". Sistem berbasis *spreadsheet* umumnya hanya berfungsi sebagai alat inventaris sederhana. Sebaliknya, sistem yang dilengkapi kemampuan untuk melacak riwayat, penggunaan, serta jadwal perawatan aset dapat bergerak melampaui inventarisasi dasar menuju manajemen aset yang komprehensif. Pendekatan ini mencerminkan sebuah proses yang lebih canggih dibandingkan sekadar pencatatan inventaris.

#### **2.2.1.2 Definisi Manajemen Aset**

Manajemen Aset adalah suatu ilmu untuk memandu pengelolaan kekayaan yang mencakup suatu proses perencanaan kebutuhan aset, mendapatkan, inventarisasi, legal audit, menilai, mengoperasikan, memelihara, membaharukan atau menghapuskan, hingga mengalihkan aset secara efektif dan efisien (Gunawan dkk., 2024, hlm. 2).

Menurut Pasaribu, hlm. (2021, hlm. 4) Pada dasarnya aset adalah istilah ekonomi dan dengan demikian aset merupakan sesuatu yang mempunyai nilai ekonomis. Secara umum aset adalah barang (thing) atau sesuatu barang (*anything*) yang mempunyai nilai ekonomi (*economic value*), nilai komersial (*commercial value*) atau nilai tukar (*exchange value*) yang dimiliki oleh badan usaha, instansi atau individu (perorangan). Aset adalah benda yang terdiri dari benda tidak

bergerak dan benda bergerak. Barang yang dimaksud meliputi barang yang tidak bergerak (tanah dan atau bangunan) dan barang bergerak (mobil, motor, dsb), baik yang berwujud (*tangible*) maupun tidak terwujud (*intangible*), yang tercakup dalam aktiva/kekayaan atau harta kekayaan dari suatu perusahaan, badan usaha, institusi atau individu perorangan. Aset adalah barang yang dalam pengertian hukum disebut benda, yang terdiri dari benda tidak bergerak dan benda bergerak.

### 2.2.1.3 Siklus Hidup Aset (*Asset Lifecycle*)

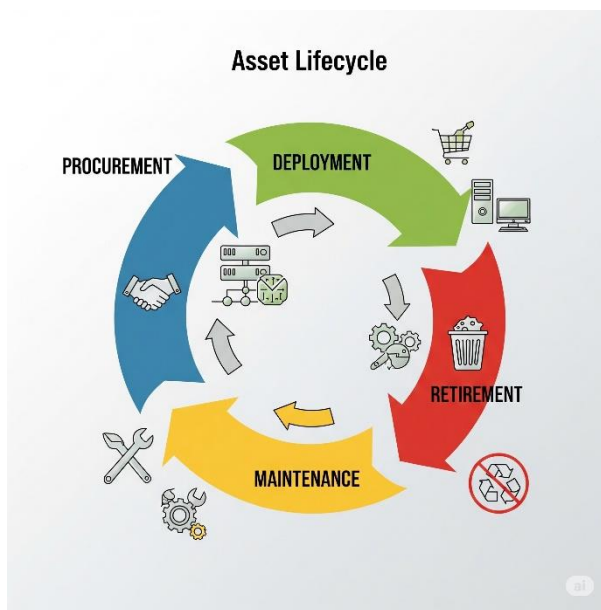
Konsep siklus hidup aset, yang disinggung oleh (Hartomo dkk., 2023, hlm. 2), menyatakan bahwa setiap aset melewati serangkaian tahapan dari pengadaan hingga penghapusan. Ruang lingkup penelitian ini secara eksplisit mencakup "Pengelolaan siklus hidup aset secara lengkap", yang umumnya terdiri dari tahap-tahap berikut:

1. **Pengadaan (*Procurement*):** Proses pembelian atau perolehan aset.
2. **Penerapan (*Deployment*):** Penugasan aset kepada pengguna atau lokasi tertentu.
3. **Pemeliharaan (*Maintenance*):** Kegiatan perawatan dan perbaikan untuk menjaga kondisi dan nilai aset.
4. **Penghapusan (*Retirement/Disposal*):** Proses penarikan aset dari penggunaan aktif.

Konsep siklus hidup ini bukan sekadar teori, melainkan harus tercermin secara langsung dalam desain sistem, terutama dalam struktur basis data dan fungsionalitas aplikasi.

1. Untuk tahap Pengadaan, basis data harus memiliki kolom untuk tanggal pembelian, harga, dan informasi garansi.
2. Untuk tahap Penerapan, sistem harus mampu mencatat histori penugasan aset kepada pengguna dan perpindahan antar lokasi.
3. Untuk tahap Pemeliharaan, sistem harus menyediakan fitur untuk mencatat aktivitas perbaikan dan menjadwalkan perawatan preventif.
4. Untuk tahap Penghapusan, harus ada mekanisme untuk mengubah status aset menjadi "dihapuskan" atau "tidak aktif".



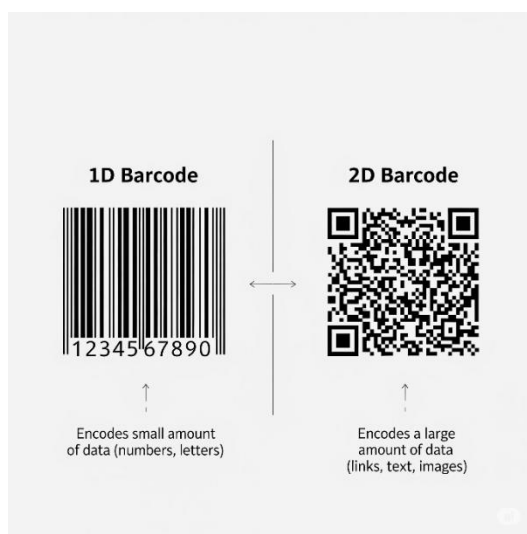


Gambar 2. 5 *Asset Lifecycle*  
 Sumber: <https://shipglobal.in/blogs>

## 2.2.2 Konsep Dasar Kode Dua Dimensi (2D Code)

### 2.2.2.1 Pengertian Kode Dua Dimensi

Kode dua dimensi (2D), yang dikenal juga sebagai kode matriks, menyimpan data secara horizontal dan vertikal, memberikan kapasitas jauh lebih besar ketimbang *barcode* satu dimensi. Misalnya, dalam QR code, digunakan kode Reed–Solomon untuk menyisipkan informasi koreksi kesalahan dan menjamin keandalan pembacaan meski sebagian bagian rusak (Bajaj, 2024, hlm. 10). Bajaj, hlm. (2024, hlm. 1–13) menjelaskan bagaimana mekanisme ini bekerja dan potensi kerentanannya terhadap manipulasi citra QR.



Gambar 2. 6 *Barcode 1D vs 2D*

Sumber: <https://bagaz.medium.com>



Gambar 2. 7 *Data Matrix*  
Sumber: <https://bagaz.medium.com>

#### 2.2.2.2 Prinsip Kerja dan Keunggulan Umum

Berbeda dari scanner laser pada *barcode* linier, kode 2D seperti QR dibaca dengan pemindai berbasis citra (camera imager). Keunggulannya adalah kapasitas tinggi dan kemampuan koreksi kesalahan yang kuat melalui algoritma seperti Reed–Solomon. Selain itu, konstruksi dua-dimensional RS codes dapat mengatasi error insersi dan deletions hingga batas optimal  $(n-3)$ . Con dkk., hlm. (2024, hlm. 4–8) dalam penelitiannya memperlihatkan desain algoritma decoding yang efisien dan optimal untuk kode 2D semacam ini.

#### 2.2.3 Teknologi Kode Dua Dimensi Pilihan

##### 2.2.3.1 Karakteristik dan Struktur *Data Matrix*

*Data Matrix* adalah kode matriks D yang terdiri dari sel-sel hitam dan putih (atau modul) yang disusun dalam pola persegi atau persegi panjang. Menurut Karrach & Pivarčiová, hlm. (2021, hlm. 1–2) *Data Matrix* adalah kode matriks 2-dimensi yang terbentuk dari modul hitam dan putih dalam pola persegi atau persegi panjang. Versi ECC 200 modern menggunakan penyusunan “*L-shaped*” *finder* (dua sisi solid) dan timing pattern alternasi di sisi berlawanan untuk orientasi dan penentuan jumlah baris/kolom. Simbol bervariasi dari  $10 \times 10$  hingga  $144 \times 144$  modul, dengan kapasitas hingga 2.335 karakter alfanumerik (atau hingga 3.116 numerik / 1.556 byte). Kemampuan koreksi kesalahan berbasis Reed–Solomon dapat memperbaiki hingga sekitar 30 % kerusakan.

### 2.2.3.2 Karakteristik dan Struktur QR Code

*Quick Response* (QR) Code adalah jenis kode D lain yang sangat populer, terutama di aplikasi konsumen. Menurut Bahat Nauli, hlm. (2024, hlm. 3) QR Code adalah kode matriks 2-dimensi dengan modul kotak hitam dan putih dan struktur yang khas. Terdapat tiga finder pattern di sudut (proporsi 1:1:3:1:1), separator margin tipis di sekitarnya, timing patterns horizontal dan vertikal, serta alignment patterns versi 2 ke atas (strukturnya: 5×5 gelap, 3×3 terang, pusat gelap) untuk koreksi distorsi. Simbol QR berkisar dari 21×21 hingga 177×177 modul, dan dapat menyimpan hingga 7.089 karakter numerik, 4.296 alfanumerik, atau 1.817 Kanji.

### 2.2.3.3 Analisis Perbandingan Teknis: *Data Matrix* vs. QR Code

Meskipun *Data Matrix* dan QR Code merupakan jenis kode dua dimensi (2D) yang paling umum dan populer, keduanya memiliki karakteristik teknis dan struktural yang berbeda secara fundamental. Perbedaan ini membuat masing-masing lebih cocok untuk jenis aplikasi yang berbeda. Menurut Karrach & Pivarčiová, hlm. (2021, hlm. 2), perbandingan teknis antara keduanya dapat dirangkum sebagai berikut:

Tabel 2. 5 Perbandingan *Data Matrix* dan QR Code

Fitur	<i>Data Matrix</i>	QR Code
Ukuran	Dari 10x10 hingga 144x144 modul (penambahan +2 modul)	Dari 21x21 hingga 177x177 modul (penambahan +4 modul)
Kapasitas	3.116 karakter numerik, atau 2.335 alfanumerik, atau 1.556 byte	7.089 karakter numerik, atau 4.296 alfanumerik, atau 2.953 byte (pada tingkat koreksi kesalahan terendah)
Tingkat Koreksi Kesalahan	Satu level standar (ECC 200) yang mampu menoleransi kerusakan hingga sekitar 30%	Empat level yang dapat dikonfigurasi: L (Low) 7%, M (Medium) 15%, Q (Quartile) 25%, dan H (High) 30%.
Pola Penemu (Finder Pattern)	Berbentuk "L" yang terletak di tepi kode	Terletak di tiga sudut kode, masing-masing

		dibentuk oleh kotak gelap di dalam bingkai yang lebih besar.
<b>Pola Waktu (Timing Pattern)</b>	Terdiri dari modul gelap dan terang yang berselang-seling di sepanjang tepi yang berlawanan dengan <i>finder pattern</i> .	Terdiri dari modul gelap dan terang yang berselang-seling dan menghubungkan ketiga <i>finder pattern</i>

Secara struktural, perbedaan paling mencolok terletak pada *Finder Pattern*. *Data Matrix* menggunakan sebuah pola penemu berbentuk "L" yang solid pada dua sisinya untuk menentukan posisi dan orientasi. Sebaliknya, QR Code menggunakan tiga pola penemu identik yang ditempatkan di sudut-sudutnya, yang memungkinkannya dibaca dari berbagai orientasi dengan sangat cepat.

Keduanya memanfaatkan algoritma koreksi kesalahan Reed-Solomon untuk memastikan keandalan data, bahkan jika sebagian kode mengalami kerusakan atau kotor. Namun, implementasinya berbeda. QR Code menawarkan empat level koreksi kesalahan yang dapat dipilih (Low, Medium, Quartile, High), yang memungkinkan pengembang untuk menyeimbangkan antara kepadatan data dan ketahanan terhadap kesalahan. Di sisi lain, *Data Matrix* (khususnya versi modern ECC 200) umumnya menggunakan satu tingkat koreksi kesalahan standar yang sangat kuat.

#### 2.2.3.4 Justifikasi Pemilihan *Data Matrix* untuk Manajemen Aset

Pemilihan antara *Data Matrix* dan QR Code untuk sebuah aplikasi spesifik harus didasarkan pada analisis keunggulan teknis masing-masing dalam konteks kebutuhan aplikasi tersebut. Berdasarkan perbandingan teknis, justifikasi pemilihan teknologi dapat dibangun di atas beberapa pilar utama:

1. **Kepadatan Data dan Ukuran Fisik:** Salah satu keunggulan utama *Data Matrix* adalah kemampuannya untuk menyimpan data dalam jumlah besar pada area fisik yang sangat kecil. Karrach & Pivarčiová, hlm. (2021, hlm. 2) menyatakan bahwa *Data Matrix* sering digunakan untuk menandai barang-barang kecil seperti komponen elektronik karena membutuhkan lebih sedikit ruang untuk menyandikan jumlah data yang sama dibandingkan QR Code. Hal ini

menjadikannya pilihan ideal untuk aplikasi manajemen aset di mana banyak aset memiliki permukaan terbatas untuk penandaan.

2. **Ketahanan dan Aplikasi Industri:** Struktur *Data Matrix* yang lebih sederhana dan kepadatannya yang tinggi membuatnya sangat cocok untuk lingkungan industri. *Data Matrix* adalah standar yang diadopsi secara luas untuk *Direct Part Marking* (DPM), yaitu proses penandaan kode secara permanen langsung ke permukaan aset (misalnya, dengan etsa laser) daripada menggunakan stiker. Penandaan permanen ini krusial untuk melacak aset sepanjang siklus hidupnya, terutama pada aset yang sering terpapar kondisi lingkungan yang keras. Penggunaan *Data Matrix* dalam proses produksi dan logistik telah terdokumentasi dengan baik.
3. **Sifat Koreksi Kesalahan:** Meskipun kedua kode menggunakan algoritma Reed-Solomon, filosofi implementasinya berbeda. Fleksibilitas empat level koreksi pada QR Code sangat bermanfaat untuk aplikasi yang menghadap konsumen, seperti pemasaran atau tautan informasi, di mana pengembang mungkin ingin memaksimalkan jumlah data (misalnya URL yang panjang) dengan mengorbankan sedikit ketahanan. Sebaliknya, standar koreksi kesalahan *Data Matrix* yang tunggal dan kuat (ECC 200) memberikan tingkat keandalan yang konsisten dan tinggi. Untuk aplikasi kritis seperti manajemen aset, di mana integritas data lebih penting daripada fleksibilitas kapasitas, ketahanan yang terstandarisasi ini merupakan sebuah keuntungan signifikan.

Pada Secara ringkas, pilihan antara keduanya mencerminkan prioritas aplikasi. QR Code unggul dalam skenario yang membutuhkan kapasitas data besar dan interaksi cepat dengan konsumen. Sebaliknya, *Data Matrix* lebih superior untuk aplikasi industri dan manajemen aset karena kepadatan datanya yang tinggi pada ukuran kecil, ketahanannya terhadap lingkungan industri, dan tingkat koreksi kesalahan yang kuat dan terstandarisasi.

## 2.2.4 Teknologi Sisi Klien (Client-Side)

### 2.2.4.1 Platform Android

Android dipilih sebagai platform target untuk aplikasi klien. Kinari dkk., hlm. (2024, hlm. 1–3), Android adalah "sistem operasi *mobile* yang paling banyak diminati dan digunakan pada perangkat *smartphone*" dan bersifat *open source*. Popularitas dan aksesibilitasnya yang luas di kalangan karyawan membuatnya

menjadi platform yang ideal untuk penerapan sistem ini, karena tidak memerlukan pengadaan perangkat keras khusus.

#### 2.2.4.2 Bahasa Pemrograman Dart

Dart adalah bahasa pemrograman yang menjadi fondasi dari kerangka kerja Flutter, keduanya dikembangkan oleh Google Kinari dkk., hlm. (2024, hlm. 5), Dart adalah bahasa berorientasi objek berbasis kelas dengan sintaks yang sangat mirip dengan JavaScript, sehingga memudahkan transisi bagi pengembang yang sudah terbiasa dengan bahasa tersebut .

Aung dkk., hlm. (2024, hlm. 6) menambahkan bahwa Dart terintegrasi secara mulus dengan Flutter dan dirancang untuk menawarkan kesederhanaan, efisiensi, dan performa. Salah satu keunggulan utamanya adalah kemampuannya untuk dikompilasi menjadi kode JavaScript, yang memungkinkannya berjalan di platform *web* yang tidak mendukung Dart secara *native* (Kinari dkk., 2024, hlm. 5). Karakteristik inilah yang menjadikan Dart sebagai pilihan yang kuat untuk pengembangan aplikasi lintas-platform.

#### 2.2.4.3 Framework Flutter

Flutter adalah sebuah *Software Development Kit* (SDK) atau *toolkit* antarmuka pengguna (UI) bersifat *open-source* dari Google yang memungkinkan pengembang membangun aplikasi yang dikompilasi secara *native* untuk platform seluler, *web*, dan desktop dari satu basis kode tunggal (Kinari dkk., 2024, hlm. 2). Flutter memungkinkan konversi kode yang efisien menjadi kode *native* untuk aplikasi seluler dan JavaScript yang dioptimalkan untuk peramban web (Aung dkk., 2024, hlm. 1).

Keunggulan utama Flutter yang relevan dengan penelitian ini meliputi:

1. Pengembangan Pengembangan Lintas Platform: Flutter memungkinkan pengembangan aplikasi untuk berbagai platform dari satu basis kode, yang secara signifikan menghemat waktu, upaya, dan biaya pengembangan (Kinari dkk., 2024, hlm. 2).
2. Antarmuka yang Ekspresif dan Fleksibel: Menurut Kinari dkk., hlm. (2024, hlm. 2–4), Flutter menyediakan seperangkat widget dan alat siap pakai yang kaya, yang membantu pengembang menciptakan antarmuka pengguna yang menarik, responsif, dan fleksibel. Arsitektur berbasis widget ini menyederhanakan pengembangan antarmuka yang kompleks.

3. Kinerja Tinggi: Aplikasi yang dibangun dengan Flutter menunjukkan kinerja yang mendekati aplikasi native karena menggunakan mesin rendering-nya sendiri (Skia) dan arsitektur berbasis widget yang dapat disesuaikan (Kinari dkk., 2024, hlm. 4).
4. Hot Reload: Fitur ini, seperti yang disorot oleh Aung dkk., hlm. (2024, hlm. 6), memungkinkan pengembang melihat perubahan kode secara *real-time* di aplikasi yang sedang berjalan tanpa perlu memulai ulang seluruh aplikasi. Ini mempercepat iterasi dan proses eksperimen secara drastis, menjadikannya sangat cocok untuk tujuan edukasi dan pengembangan yang cepat.

#### 2.2.4.4 Arsitektur Perangkat Lunak: Clean Architecture

Clean Architecture menekankan pemisahan yang tegas antara logika inti (domain/business rules) dan detail implementasi (UI, database, perangkat, jaringan). Tujuannya adalah kode yang mudah diuji, mudah dipelihara, serta tahan perubahan pada platform maupun dependensi (Sinatria dkk., 2023, hlm. 2). Studi-studi berbasis Flutter menunjukkan pola tiga lapis yang konsisten Presentation, Domain, dan Data dengan dependency rule satu arah menuju domain sebagai pusat (Sinatria dkk., 2023, hlm. 8; Wijayanto dkk., 2023). Berikut adalah detailnya:

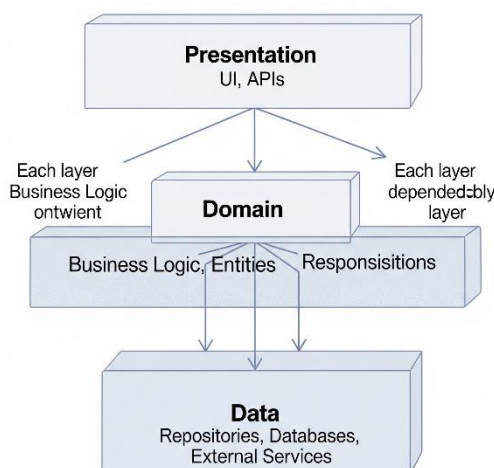
1. Use Cases / Interactors Use case adalah inti perilaku aplikasi: mereka mengenkapsulasi aturan bisnis spesifik dan urutan langkah untuk sebuah fitur (mis. *createOrder*, *getProfile*). Di Flutter, use case ditempatkan di lapisan domain sehingga tak bergantung pada UI atau sumber data (Sinatria dkk., 2023, hlm. 11); hal ini memudahkan pengujian unit dan menjaga stabilitas logika saat detail implementasi berubah.
2. Entities / Value Objects Entities adalah objek domain yang merepresentasikan konsep bisnis (mis. User, Product) dan value objects memastikan invariant bisnis (mis. Email, Money). Menjaga entities bebas dari dependensi *framework* Flutter membuatnya mudah dipakai ulang dan diuji secara independen (Sinatria dkk., 2023, hlm. 11).
3. Repository Interfaces Repository interface didefinisikan di domain sebagai kontrak akses data (mis. UserRepository) sehingga use case bergantung pada abstraksi, bukan implementasi (Sinatria dkk., 2023, hlm. 11). Implementasi konkret disediakan oleh lapisan data, memungkinkan pergantian sumber data tanpa mengubah domain.

4. Data Sources (Remote / Local) Lapisan data mengimplementasikan repository dengan data source terpisah untuk remote (API) dan local (SQLite/SharedPreferences). Di Flutter, pola ini mempermudah mock untuk pengujian dan penggantian library HTTP atau DB tanpa merusak domain/presentation (Sinatria dkk., 2023, hlm. 11).
5. DTO & Mapper Data Transfer Objects (DTO) dan mapper bertugas menerjemahkan format eksternal (JSON/API) ke entity domain dan sebaliknya (Sinatria dkk., 2023, hlm. 11); ini mencegah “bocornya” struktur API ke domain, sehingga perubahan bentuk respons *backend* hanya mempengaruhi mapper dan DTO, bukan logika bisnis.
6. Presentation (Controllers / Presenters / ViewModels / BLoC) Presentation layer (widgets + state management seperti BLoC/Cubit/Provider) bertanggung jawab untuk UI, mapping entity → view model, dan menangani event pengguna; karena dipisahkan, UI bisa diubah ulang tanpa memengaruhi use case atau entity. Banyak studi Flutter merekomendasikan BLoC untuk menjaga boundaries yang jelas (Sinatria dkk., 2023, hlm. 12).
7. Dependency Injection (Composition Root) DI (mis. `get_it`, `provider`) disusun di composition root saat aplikasi diinisialisasi, mengikat implementasi repository, data source, dan use case (Sinatria et al., 2023, p. 12). Pendekatan ini mempertahankan aturan dependensi satu arah dan memudahkan swapping implementasi untuk testing atau perubahan produksi.
8. Error Handling & Failure Models Alih-alih melempar error mentah ke UI, pola Clean Architecture menganjurkan model Failure/Either untuk mempropagasi kesalahan secara eksplisit melalui lapisan; ini membuat penanganan error terstruktur dan lebih mudah diuji di level use case. Studi Flutter menunjukkan praktik ini meningkatkan keandalan dan prediktabilitas alur kesalahan (Sinatria et al., 2023, p. 12).
9. Testing (Unit, Integration, Widget) Dengan boundary yang jelas, testing menjadi tersegmentasi: unit test untuk use case/entities, integration test untuk repository+data source, dan widget test untuk presentation. Literatur Flutter yang menerapkan Clean Architecture melaporkan percepatan deteksi regresi dan kemudahan maintenance berkat isolasi pengujian.

Penerapan *Clean Architecture* adalah investasi awal untuk keuntungan jangka panjang. Dengan memisahkan lapisan Presentasi, Domain, dan Data,



aplikasi menjadi modular sehingga UI bisa diubah tanpa menyentuh logika inti, dan sumber data dapat diganti dengan dampak minimal (Sinatria dkk., 2023, hlm. 14). Pendekatan ini mencerminkan pola pikir pengembangan tingkat perusahaan membangun sistem yang fleksibel, mudah dipelihara, dan siap berevolusi sesuai kebutuhan di masa depan.



Gambar 2. 8 Clean Architecturede  
Sumber: <https://medium.com>

## 2.2.5 Teknologi Sisi *Server* (*Server-Side*)

### 2.2.5.1 Bahasa Pemrograman Go (Golang)

*Backend* untuk sistem ini dikembangkan menggunakan bahasa pemrograman Go (juga dikenal sebagai Golang), sebuah bahasa modern yang populer untuk membangun sistem microservice berskala besar di level enterprise. Menurut Saioc dkk., hlm. (2023, hlm. 1), popularitasnya didorong oleh kombinasi kesederhanaan, dukungan konkurensi bawaan, garbage collection, pengetikan statis (static typing), dan performa yang baik.

Fondasi utama yang membuat Go sangat cocok untuk layanan *backend* adalah model konkurensinya. Go dibangun di atas prinsip ***Communicating Sequential Processes (CSP)***, yang lebih mengutamakan komunikasi melalui pengiriman pesan (*message-passing*) daripada menggunakan memori bersama (*shared memory*) untuk sinkronisasi (Malhotra, 2025, hlm. 1). Filosofi ini terangkum dalam moto terkenalnya: "Jangan berkomunikasi dengan berbagi memori; sebaliknya, bagikan memori dengan berkomunikasi" (Malhotra, 2025, hlm. 1).

Implementasi konkurensi di Go menggunakan goroutine, yaitu *thread* ringan yang dikelola langsung oleh Go *runtime*, bukan oleh sistem operasi (Saioc dkk., 2023, hlm. 1). *Goroutine* sangat efisien dari segi sumber daya, di mana setiap *goroutine* hanya mengonsumsi sekitar 2KB memori pada saat inisialisasi. Efisiensi ini memungkinkan sebuah *server* untuk menangani puluhan ribu koneksi konkuren misalnya, 50.000 koneksi hanya dengan memori sekitar 200MB (Malhotra, 2025, hlm. 2). Untuk sinkronisasi dan pertukaran data antar *goroutine*, Go menyediakan channel, sebuah saluran komunikasi yang aman secara tipe (*type-safe*) (Malhotra, 2025, hlm. 2; Saioc dkk., 2023, hlm. 1). Penggunaan model ini terbukti dapat mengurangi kompleksitas kode secara signifikan pada sistem terdistribusi sambil tetap mempertahankan performa tinggi yang sebanding dengan bahasa tingkat rendah seperti C++ (Malhotra, 2025, hlm. 1).

#### 2.2.5.2 **Framework Fiber**

Di atas bahasa Go, framework **Fiber** digunakan untuk mempercepat pengembangan web API. Fiber adalah “*an Express inspired web framework built on top of Fasthttp, the fastest HTTP engine for Go*” (Fiber Documentation, 2025). Framework ini dirancang untuk memberikan kinerja tinggi dengan antarmuka yang sederhana serta efisiensi memori yang maksimal:

1. **Kinerja Ekstrem yang Terukur:** Berdasarkan data resmi pada halaman Benchmarks dokumentasi Fiber, framework ini mampu mencapai hingga 13.509.592 permintaan per detik dengan rata-rata latensi hanya 0,9 ms, menjadikannya salah satu framework web tercepat di ekosistem Go (Fiber Documentation, 2025). Benchmark tersebut menggunakan fasthttp sebagai HTTP engine utama, yang diakui sebagai engine HTTP tercepat di dunia Go.
2. **Efisiensi Sumber Daya:** Fiber secara eksplisit dirancang dengan prinsip “zero memory allocation and performance in mind” untuk meminimalkan overhead sistem dan memaksimalkan throughput (Fiber Documentation, 2025). Pendekatan ini menjadikan Fiber sangat ringan dan efisien, bahkan pada konfigurasi perangkat keras dengan sumber daya terbatas.
3. **Ukuran Deployment yang Minimal :** Fiber tidak memerlukan runtime tambahan seperti Node.js atau JVM, sehingga menghasilkan ukuran binary dan image Docker yang relatif kecil. Dokumentasi Fiber menyoroti keunggulan ini sebagai salah satu alasan utama mengapa framework ini banyak dipilih untuk

aplikasi berbasis container (containerized environments) (Fiber Documentation, 2025).

4. **Konsistensi Performa:** Hasil benchmark resmi menunjukkan bahwa performa Fiber tetap stabil pada beban tinggi dengan latensi di bawah 1 ms secara konsisten, bahkan ketika jumlah koneksi meningkat (Fiber Documentation, 2025). Hal ini membuktikan bahwa Fiber mampu mempertahankan stabilitas dan kecepatan respons di berbagai skenario produksi.

Kombinasi antara bahasa Go dan framework Fiber merupakan manifestasi dari prioritas utama sistem ini: kecepatan, efisiensi, dan skalabilitas tinggi yang telah dibuktikan secara empiris melalui pengujian resmi. Backend yang dibangun dengan Fiber dioptimalkan untuk merespons permintaan pengguna secara instan, bahkan ketika terjadi banyak permintaan simultan..

### 2.2.5.3 Object-Relational Mapping (ORM): GORM

Untuk berinteraksi dengan basis data, GORM digunakan. GORM adalah “*the fantastic ORM library for Golang, aims to be developer friendly*” (GORM Documentation, 2025). Sebagai sebuah ORM, GORM memetakan struct (struktur data) di Go ke tabel di basis data relasional. Ini memungkinkan pengembang untuk melakukan operasi basis data (Create, Read, Update, Delete) menggunakan objek Go yang familier, alih-alih menulis kueri SQL mentah. Keunggulannya adalah:

1. **Keamanan Terintegrasi:** Dokumentasi resmi GORM menegaskan bahwa “GORM uses the database/sql’s argument placeholders to construct the SQL statement, which will automatically escape arguments to avoid SQL injection”. (GORM Documentation, 2025).
2. **Optimisasi Performa yang Komprehensif:** Dokumentasi menyebut beberapa fitur performa, misalnya *Prepared Statement Caching* (“...GORM allows cache prepared statement to increase performance”) dan dukungan untuk mode-generik (Generics API) yang meningkatkan keamanan tipe dan mempermudah pemeliharaan kode. (GORM Documentation, 2025).
3. **Produktivitas Pengembangan:** Dokumentasi menyebut bahwa GORM adalah “developer friendly” dengan fitur lengkap seperti Associations, Hooks, Transactions, Batch Insert, database resolver (read/write splitting) yang membantu pengembang fokus pada logika aplikasi. (GORM Documentation, 2025)**Dukungan Asosiasi yang Kuat:** GORM menangani semua jenis relasi (one-to-one, one-to-many, many-to-many) secara natural dan intuitif,

memungkinkan definisi model kompleks dengan tabel yang saling terkait tanpa kompleksitas SQL manual.

#### **4. Dukungan Asosiasi yang Kuat & Fitur Lanjutan untuk Skalabilitas:**

Dokumentasi menunjukkan bahwa GORM menangani relasi seperti Has One, Has Many, Many To Many, Polymorphism, dan mendukung plugin seperti Database Resolver (multiple databases, read/write splitting) yang menunjang skalabilitas. (GORM Documentation, 2025).

Meskipun menulis SQL mentah sangat kuat, penggunaan ORM seperti GORM secara signifikan mempercepat pengembangan dan mengurangi risiko kerentanan keamanan seperti SQL injection, sebagaimana dikonfirmasi oleh dokumentasi resmi di atas. Pilihan ini mencerminkan keseimbangan pragmatis antara kinerja mentah (dari Go/Fiber) dan produktivitas serta keamanan pengembang (dari GORM), yang didukung oleh optimisasi performa bawaan dan arsitektur yang telah matang.

### **2.2.6 Teknologi Basis Data dan Notifikasi**

#### **2.2.6.1 Sistem Manajemen Basis Data: PostgreSQL**

PostgreSQL dipilih sebagai sistem manajemen basis data relasional (RDBMS) untuk proyek ini. PostgreSQL adalah "a database system designed by Postgres to *Process* relational databases" dan dikenal sebagai sistem yang kuat dan kaya fitur. Tidak seperti RDBMS lain yang lebih sederhana, PostgreSQL dikenal karena kepatuhannya yang ketat terhadap standar SQL, keandalan, dan integritas data yang kuat. Keunggulan postgresql diantaranya adalah:

1. Dukungan JSON/JSONB yang superior menjadi salah satu alasan teknis utama pemilihan PostgreSQL, sebagaimana dikonfirmasi oleh studi perbandingan sistematis Taipalus, hlm. (2023, hlm. 34–36), adalah dukungannya untuk data JSON. PostgreSQL menunjukkan performa yang "commendable efficiency" dalam menangani data JSON, terutama pada operasi single insert yang secara signifikan mengungguli MongoDB. Kemampuan untuk menyimpan dan mengindeks tipe data JSONB secara native memungkinkan fleksibilitas yang lebih besar, di mana data terstruktur (seperti detail aset) dan data semi-terstruktur (seperti atribut khusus aset) dapat hidup berdampingan secara efisien dalam satu platform basis data.
2. Optimisasi penyimpanan dan performa JSONB juga menjadi keunggulan. Dokumentasi resmi PostgreSQL (2025) menegaskan bahwa JSONB

menggunakan "a decomposed binary format" yang membuatnya "significantly faster to *Process* as it does not involve reparsing". Studi teknis DbVis (2025) menunjukkan bahwa JSONB "wins the performance race as it stores JSON data in a binary format, making it more efficient for querying compared to json". Format biner ini mendukung operasi-operasi canggih seperti existence operator dan GIN (Generalized Inverted Index) yang memberikan kemampuan pencarian yang efisien pada dokumen JSON dalam volume besar.

3. Validasi empiris dari studi perbandingan juga mendukung hal ini. Studi *benchmarking* yang dianalisis oleh (Taipalus, 2023, hlm. 33–35) menunjukkan bahwa dalam beberapa skenario, PostgreSQL "demonstrated significantly better performance in processing single inserts of JSON data compared to MongoDB"..
4. Konsistensi performa dan skalabilitas juga menjadi pertimbangan. Meskipun studi yang sama menunjukkan bahwa performa select PostgreSQL menurun pada dataset yang sangat besar (>500K rows), MongoDB menunjukkan konsistensi yang lebih baik dalam skenario ini (Taipalus, 2023, hlm. 33–35). Namun, untuk sebagian besar aplikasi dengan volume data menengah, PostgreSQL memberikan keseimbangan optimal antara performa dan fleksibilitas fitur.
5. Dalam posisinya pada landscape database modern, berdasarkan systematic literature review (Taipalus, 2023, hlm. 9–12) yang menganalisis 117 studi perbandingan database, PostgreSQL konsisten muncul sebagai salah satu RDBMS yang paling sering dibandingkan dan menunjukkan performa kompetitif. Survey Stack Overflow yang dikutip dalam studi Stormatics (2024) juga menunjukkan "PostgreSQL's consistent ascent in popularity" sejak 2019.

Pilihan PostgreSQL dengan dukungan JSONB native ini mencerminkan keseimbangan strategis antara kekuatan relational database tradisional dan fleksibilitas NoSQL modern. Sebagaimana dikonfirmasi oleh berbagai studi empiris, ini adalah pilihan yang mendukung keandalan, skalabilitas di masa depan, dan kemampuan untuk menangani beragam tipe data dalam satu platform yang konsisten.

#### 2.2.6.2 Layanan Notifikasi: Firebase *Cloud* Messaging (FCM)

Layanan notifikasi Firebase *Cloud* Messaging (FCM) diintegrasikan untuk membuat sistem ini benar-benar *real-time* dari perspektif pengguna. FCM adalah "a cross-platform messaging solution that lets you reliably send messages at no

cost". Fungsinya adalah untuk mengirimkan notifikasi push dari *server* ke aplikasi klien.

Berdasarkan studi yang dipublikasikan dalam jurnal CMC (Computers, Materials & Continua) oleh (Sung dkk., 2023, hlm. 20), Firebase telah terbukti sebagai platform yang sangat efektif untuk komunikasi *real-time* dalam sistem IoT. Penelitian mereka menunjukkan bahwa "Firebase *cloud* data, which can be accessed directly in *real-time* via Android-based *mobile* apps for end users" memberikan solusi yang handal dengan akurasi 95.51% dalam transmisi data *real-time*.

Sung dkk., hlm. (2023, hlm. 5) juga mendeskripsikan dalam penelitiannya bagaimana Firebase bekerja melalui protokol MQTT (Message Queuing Telemetry Transport) yang "works as a broker or *server* that has the function of filtering incoming messages and distributing them to clients who are interested in receiving them". Dalam konteks sistem tracking aset, IoT devices bekerja sebagai clients yang mempublish data dari sensor, dan MQTT protocol meneruskan data *real-time* ke Firebase melalui *cloud* gateway.

Selain itu, studi Sung dkk., hlm. (2023, hlm. 19) menganalisis aspek cost-effectiveness dari penggunaan Firebase dalam sistem IoT, menunjukkan bahwa implementasi berbasis IoT dengan Firebase memiliki "cost-effectiveness of 94.21% compared to labor costs". Hal ini menunjukkan bahwa selain keandalan teknis, FCM juga memberikan efisiensi ekonomi yang signifikan untuk sistem enterprise.

## 2.2.7 Format Pertukaran Data

### 2.2.7.1 JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) dipilih sebagai format standar industri untuk pertukaran data antara aplikasi Flutter dan *backend* Go. Berikut adalah keunggulan JSON:

1. Standarisasi dan Ubiquitas: Coelho & Yannou-Medrala, p. (2023, p. 2) dalam penelitian mereka yang dipublikasikan oleh Mines Paris - PSL University mengonfirmasi bahwa "JSON is a simple de facto standard cross-language textual format used to represent, exchange and store structured data in computer systems". Penelitian ini menekankan bahwa JSON telah "become in recent years an ubiquitous cross-language de facto standard to represent, exchange and store data between computer applications, partially replacing XML".

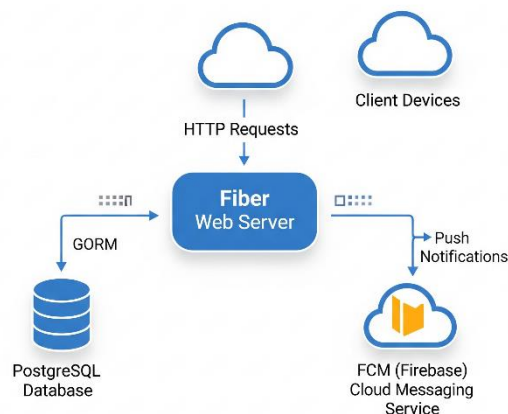
2. Efisiensi dan Kemudahan Pemrosesan: Studi akademik Coelho & Yannou-Medrala, hlm. (2023, hlm. 2) menunjukkan bahwa JSON bersifat ringan dan mudah diproses karena strukturnya yang "built upon the null value, booleans, numbers, Unicode strings, arrays (aka *list*, tuple, sequence, set) and objects (aka struct, record, dict, map, association, key-value pairs)". Karakteristik ini membuatnya "lightweight and featureful" untuk aplikasi modern.
3. Dukungan Cross-Platform Native: Penelitian tersebut juga mengonfirmasi bahwa JSON didukung secara native oleh hampir semua bahasa pemrograman modern, dengan "a wide range of libraries and tools are available for many programming languages and systems beyond JavaScript including Python, Java, Shell, and SQL" (Coelho & Yannou-Medrala, 2023, hlm. 2). Hal ini sejalan dengan penggunaan Dart (Flutter) dan Go dalam arsitektur sistem ini.
4. Optimisasi untuk REST API: Berdasarkan analisis Coelho & Yannou-Medrala, hlm. (2023, hlm. 2), JSON memiliki dua use case utama yang relevan dengan sistem ini: "API Data for simple structures exchanged between computer systems at API interfaces, for instance between *web* front-ends and back-ends in multitier architectures"<sup>2</sup>. Ini menegaskan bahwa pemilihan JSON untuk REST API dalam arsitektur Flutter-Go adalah praktik yang telah divalidasi secara akademik.
5. Keamanan dan Standardisasi: Gómez dkk., hlm. (2022, hlm. 1) dalam penelitiannya yang dipublikasikan di Computer and Information Technology menyatakan bahwa "Due to its simplicity, JSON objects are the most common way for sending information over the HTTP protocol"<sup>1</sup>. Penelitian ini juga mengakui pentingnya keamanan dalam pertukaran JSON, yang mendukung implementasi sistem yang aman dan reliable.

JSON berfungsi optimal dalam arsitektur multitier seperti yang dijelaskan dalam penelitian (Coelho & Yannou-Medrala, 2023, hlm. 2), khususnya untuk "API Data for simple structures exchanged between computer systems at API interfaces". Dalam konteks sistem tracking aset, struktur berbasis pasangan kunci-nilai (*key-value pairs*) JSON memungkinkan representasi data yang intuitif dan efisien.

Sebagaimana dikonfirmasi oleh research terbaru, JSON memiliki karakteristik yang sangat sesuai untuk aplikasi *mobile* dan *web* modern: "JSON data are mostly generated and processed automatically from a programming language",

yang sejalan dengan otomatisasi dalam sistem Flutter-Go yang dikembangkan (Coelho & Yannou-Medrala, 2023, hlm. 2).

Pemilihan JSON memastikan interoperabilitas yang mulus dan efisien, didukung oleh standarisasi yang kuat dan adopsi universal dalam ekosistem teknologi modern. Ini menjadi bagian akhir yang melengkapi arsitektur sistem yang modern, kohesif, dan berkinerja tinggi, sebagaimana divalidasi oleh penelitian-penelitian akademik terkini.



Gambar 2. 9 Arsitektur *Server*  
 Sumber: <https://medium.com/@amarjitkr93>

## 2.3 Literature Review

Tabel di bawah ini merangkum penelitian-penelitian yang telah dilakukan dan memiliki korelasi dengan topik "Implementasi Aplikasi Inventaris Dan Pelacakan Aset Perusahaan Menggunakan Data Matrix Berbasis Android".

Tabel 2. 6 *Literature Review*

No	Judul	Penulis	Metode	Hasil
1	Pengembangan Sistem Manajemen Aset Pada PT. PELINDO IV Cabang	Mardani dkk. (2023)	Menggunakan metode Prototyping untuk aplikasi Android dengan QR-Code. Pengujian fungsionalitas	Berhasil membuat pencatatan aset lebih akurat dan terpusat, sehingga proses pengelolaan data



	Balickpapan Berbasis Android		dengan <i>Black Box</i> dan <i>User Acceptance Testing</i> (UAT).	menjadi lebih cepat dan efisien.
2	Sistem Monitoring Data Aset dan Inventaris IT Berbasis <i>Web</i> pada PT. Pan Brothers Tbk	Rakhma dkk. (2022)	Menggunakan metode <i>Rapid Application Development</i> (RAD). Perancangan sistem memakai UML dan diuji dengan <i>Blackbox Testing</i> untuk platform <i>web</i> (PHP & MySQL).	Menghasilkan sistem informasi berbasis <i>web</i> yang dapat menangani manajemen aset IT, mempermudah pengawasan dan proses pelaporan.
3	Perancangan dan Implementasi Sistem Informasi Inventory Barang Berbasis <i>Web</i> pada PT XYZ	Muhamad Yunus Aliviyudin dkk. (2025)	Menerapkan model <i>Waterfall</i> (SDLC). Aplikasi <i>web</i> dikembangkan dengan PHP, MySQL, dan Bootstrap, lalu diuji fungsionalitasnya menggunakan <i>Black Box Testing</i> .	Menghasilkan program yang menyederhanakan manajemen inventaris, mempercepat pencarian data, dan otomatisasi pembuatan laporan.
4	Pengembangan Aplikasi Perangkat Bergerak Sistem Pendataan Inventaris UKM	M. Fadhli Alpharajasa dkk. (2023)	Menggunakan metode <i>Prototyping</i> dengan arsitektur MVVM. Aplikasi Android (Kotlin)	Aplikasi terbukti lebih praktis dari Excel dengan efektivitas 100%. Responden setuju aplikasi ini

	FILKOM UB berbasis Android		menggunakan Firebase sebagai <i>backend</i> . Diuji dengan <i>Black Box</i> dan <i>Usability Testing</i> .	mempermudah pendataan inventaris.
5	The Design of Goods Data Storage Application Based on Android using <i>Barcode Scanner</i>	Ipinuwati dkk. (2022)	Menggunakan model <i>Waterfall</i> (SDLC). Aplikasi Android dibangun menggunakan App Inventor dengan basis data lokal TinyDB dan fitur <i>Barcode Scanner</i> .	Berhasil membangun aplikasi penyimpan data barang via <i>barcode scan</i> yang menyederhanakan akses data (kode, nama, harga) untuk pemilik toko.
6	Perancangan sistem informasi inventarisasi aset berbasis <i>web</i> menggunakan metode <i>waterfall</i>	Usnaini dkk. (2021)	Menggunakan metode <i>Waterfall</i> dan perancangan UML. Sistem dibangun berbasis <i>web</i> dengan PHP dan MySQL untuk mengelola data aset secara terkomputerisasi.	Sistem yang dirancang berhasil menyediakan informasi aset yang cepat dan akurat, serta mempermudah pembuatan laporan aset secara efisien.
7	Sistem Peminjaman Barang Menggunakan QR	Purnomo & Alijoyo (2024)	Menggunakan metode <i>System Development Life Cycle</i> (SDLC).	Aplikasi yang dihasilkan dapat mengelola data barang dan

	Code Berbasis Aplikasi Android		Aplikasi dibangun untuk platform Android dan memanfaatkan pemindaian QR Code untuk proses peminjaman.	peminjaman secara efisien, serta mengurangi risiko kesalahan dan kehilangan data.
8	Sistem Informasi Pendataan dan Peminjaman Aset Perusahaan Berbasis <i>Web</i> Pada PT. Vadhana International	Willyana (2022b)	Menggunakan metode <i>Waterfall</i> . Sistem berbasis <i>web</i> dirancang menggunakan UML dan dikembangkan dengan PHP, <i>framework</i> Laravel, dan database MySQL.	Sistem berhasil menangani masalah pendataan dan peminjaman aset yang sebelumnya manual, sehingga menjadi lebih terstruktur dan terdokumentasi dengan baik.
9	PERANCANGAN SISTEM INFORMASI BERBASIS <i>WEB</i> PENGELOLAAN INVENTARIS ASET KANTOR DI PT. MPM FINANCE BANDUNG	Pasaribu (2021)	Menggunakan metode <i>Waterfall</i> . Menganalisis sistem berjalan untuk merancang sistem usulan berbasis <i>web</i> dengan perancangan UML dan database.	Menghasilkan sistem informasi pengelolaan inventaris aset yang dapat memberikan informasi akurat, relevan, dan tepat waktu untuk mendukung operasional perusahaan.
10	RANCANG BANGUN APLIKASI	Christian & Voutama (2024)	Menggunakan metode <i>Waterfall</i> . Sistem	Menghasilkan aplikasi inventaris berbasis <i>web</i> yang

	SISTEM INFORMASI INVENTARIS BERBASIS <i>WEBSITE</i>		dikembangkan untuk platform <i>web</i> dengan menggunakan bahasa pemrograman PHP dan database MySQL.	mempermudah pengolahan data barang, mempercepat pembuatan laporan, dan mengurangi kesalahan pencatatan.
--	---	--	---	---

Berdasarkan tinjauan literatur terhadap penelitian-penelitian relevan yang telah dilakukan, terdapat beberapa persamaan dan perbedaan dengan sistem yang akan penulis bangun, yaitu "Implementasi Aplikasi Inventaris dan Pelacakan Aset Perusahaan Menggunakan Data Matrix Berbasis Android".

1. Menurut penelitian yang dilakukan oleh Mardani dkk. (2023), terdapat persamaan dalam pembangunan aplikasi berbasis Android untuk manajemen aset di lingkungan perusahaan. Namun, perbedaannya terletak pada teknologi pemindaian yang menggunakan QR-Code, sementara penelitian penulis akan menggunakan *Data Matrix* yang berkapasitas data lebih padat.
2. Menurut penelitian dari Rakhma dkk. (2022), tujuannya serupa yaitu menangani data aset dan inventaris untuk menggantikan sistem manual. Perbedaan signifikan terletak pada platform yang berbasis *Web* (PHP dan MySQL), sedangkan penelitian penulis berfokus pada platform Android dengan teknologi pemindaian *Data Matrix*.
3. Menurut penelitian dari Muhamad Yunus Aliviyudin dkk. (2025), kesamaannya adalah tujuan untuk menggantikan proses manual dengan Excel di lingkungan perusahaan. Akan tetapi, penelitian tersebut berfokus murni pada inventaris barang berbasis *Web*, berbeda dengan penelitian penulis yang cakupannya lebih luas (inventaris dan pelacakan aset) pada platform Android.
4. Menurut penelitian oleh M. Fadhli Alpharajasa dkk. (2023), terdapat persamaan teknis yang relevan seperti penggunaan platform Android dan teknologi modern (Kotlin dan Firebase). Perbedaan utamanya adalah studi kasus untuk UKM yang kebutuhannya berbeda dari perusahaan, serta tidak diterapkannya teknologi pemindaian spesifik.

5. Menurut penelitian dari Ipinuwati dkk. (2022), persamaannya adalah penggunaan platform Android dan teknologi pemindaian (*Barcode Scanner*). Namun, studi kasusnya adalah toko (skala UMKM) dengan fokus pada pengecekan harga, bukan pelacakan aset perusahaan yang kompleks, serta teknologinya lebih sederhana (App Inventor dan TinyDB).
6. Menurut penelitian dari Usnaini dkk. (2021), terdapat kesamaan dalam tujuan untuk mengelola inventarisasi aset secara terkomputerisasi. Perbedaan utamanya adalah platform yang dikembangkan berbasis *Web* dengan metode Waterfall, sedangkan penelitian penulis berfokus pada aplikasi Android.
7. Menurut penelitian oleh Purnomo & Alijoyo (2024), ditemukan kesamaan dalam pemanfaatan platform Android dan teknologi pemindaian (QR Code). Akan tetapi, fokusnya lebih spesifik pada sistem peminjaman barang, tidak mencakup keseluruhan siklus hidup manajemen aset yang diusulkan penulis.
8. Menurut penelitian yang dilakukan Willyana (2022b), sistem yang dibangun juga menargetkan pendataan dan peminjaman aset di lingkungan perusahaan. Perbedaan utamanya adalah platform yang dikembangkan berbasis *Web* (PHP dan Laravel), sementara penelitian penulis berfokus pada solusi *mobile*.
9. Menurut penelitian dari Pasaribu (2021), terdapat kesamaan dalam konteks pengelolaan inventaris aset kantor untuk sebuah perusahaan. Namun, perbedaannya adalah platform yang diimplementasikan berbasis *Web*, bukan aplikasi Android.
10. Terakhir, penelitian oleh Christian & Voutama (2024) juga memiliki tujuan serupa untuk menggantikan sistem inventaris manual. Perbedaannya adalah platformnya yang berbasis *web* dan fokusnya lebih umum pada inventaris barang, tidak secara spesifik pada pelacakan aset perusahaan menggunakan aplikasi *mobile*.

Secara keseluruhan, meskipun sudah banyak penelitian yang mengembangkan sistem inventaris dan aset, penelitian yang akan penulis lakukan memiliki kebaruan pada penerapan teknologi *Data Matrix* sebagai media identifikasi dan pelacakan aset di lingkungan perusahaan dalam sebuah aplikasi berbasis Android.

## BAB III

### ANALISIS SISTEM YANG BERJALAN

#### 3.1 Gambaran Umum Objek

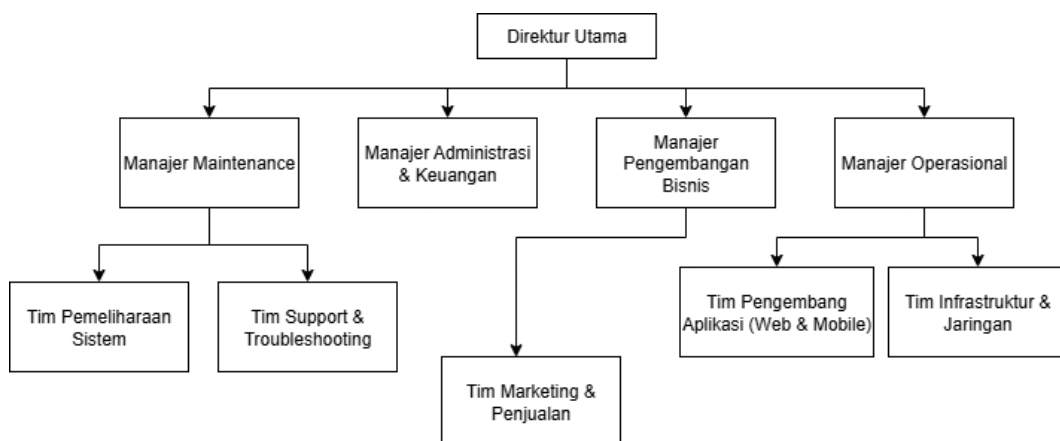
Pada bab ini, akan diuraikan gambaran umum mengenai objek penelitian, yaitu PT Fujiyama Technology Solution. Pembahasan mencakup sejarah singkat perusahaan, prosedur sistem yang sedang berjalan, analisis terhadap masalah yang dihadapi, serta alternatif pemecahan masalah yang diusulkan untuk meningkatkan efisiensi manajemen aset perusahaan.

##### 3.1.1 Sejarah Singkat

Fujiyama Technology Solution (FTS) merupakan sebuah unit atau merek layanan teknologi yang berada di bawah naungan PT Fujiyama Marketing. PT Fujiyama Marketing sendiri adalah perusahaan *powertools* yang telah beroperasi di Indonesia sejak tahun 1996. FTS memfokuskan layanannya pada solusi teknologi yang komprehensif, mencakup pengembangan aplikasi *web* dan seluler, konsultasi TI, jaringan dan perangkat keras, solusi Point of Sale (POS) dan *barcode*, serta layanan *cloud* dan pelatihan. Sebagai perusahaan teknologi yang dinamis, FTS terus mengalami pertumbuhan aset seiring dengan berkembangnya skala proyek dan penambahan jumlah karyawan, sehingga memerlukan sistem manajemen aset yang andal.

##### 3.1.2 Struktur Organisasi

Struktur organisasi dalam sebuah perusahaan menggambarkan hierarki, alur koordinasi, serta pembagian tugas dan wewenang yang jelas antar setiap bagian. Hal ini menjadi landasan untuk memastikan operasional perusahaan berjalan secara efektif dan teratur.



Gambar 3. 1 Struktur Organisasi

### 3.1.3 Wewenang dan Tanggung Jawab

Setiap jabatan dalam struktur organisasi memiliki wewenang dan tanggung jawab yang spesifik untuk memastikan semua fungsi perusahaan berjalan dengan baik. Penjelasan wewenang dan tanggung jawab ini penting untuk memberikan batasan yang jelas mengenai tugas setiap peran.

Tabel 3. 1 Wewenang dan tanggung jawab

No	Jabatan	Wewenang	Tanggung Jawab
1	Direktur Utama	Mengambil keputusan strategis tertinggi, menyetujui anggaran tahunan, dan menentukan arah kebijakan perusahaan.	Memimpin dan mengelola perusahaan secara keseluruhan untuk mencapai visi dan misi, serta bertanggung jawab kepada para pemangku kepentingan.
2	Manajer Maintenance	Mengalokasikan sumber daya untuk tim pemeliharaan dan support, serta menyetujui jadwal perawatan sistem.	Memastikan semua sistem dan aplikasi yang dikelola perusahaan berjalan dengan optimal dan andal, serta mengawasi tim teknis terkait.
3	Tim Pemeliharaan Sistem	Melakukan akses ke <i>server</i> dan sistem produksi untuk keperluan pemeliharaan dan pembaruan sesuai jadwal yang disetujui.	Menjalankan kegiatan perawatan preventif pada infrastruktur dan aplikasi untuk mencegah terjadinya gangguan atau downtime.
4	Tim Support & Troubleshooting	Menghubungi klien secara langsung untuk memberikan bantuan teknis dan melakukan analisis pada sistem yang bermasalah.	Memberikan respons cepat dan solusi efektif terhadap setiap laporan masalah atau bug yang masuk dari pengguna atau klien.

5	Manajer Administrasi & Keuangan	Menyetujui pengeluaran operasional perusahaan, mengelola arus kas, dan membuat kebijakan administratif internal.	Mengelola seluruh aspek keuangan, sumber daya manusia (SDM), dan administrasi umum perusahaan, termasuk pengelolaan aset.
6	Manajer Pengembangan Bisnis	Menegosiasikan kontrak dan kerja sama dengan klien baru, serta menyetujui strategi dan anggaran kampanye pemasaran.	Mendorong pertumbuhan pendapatan perusahaan dengan mencari peluang bisnis baru dan memimpin tim marketing & penjualan.
7	Tim Marketing & Penjualan	Menjalankan kampanye pemasaran yang telah disetujui dan memberikan penawaran harga kepada calon klien sesuai standar perusahaan.	Mempromosikan layanan perusahaan untuk menghasilkan leads berkualitas dan mencapai target penjualan yang telah ditetapkan.
8	Manajer Operasional	Mengatur prioritas proyek, mengalokasikan developer dan sumber daya teknis, serta menyetujui arsitektur teknis proyek.	Mengawasi kelancaran semua proyek teknis dari awal hingga akhir, memastikan proyek selesai tepat waktu, sesuai anggaran, dan berkualitas.
9	Tim Pengembang Aplikasi	Memilih teknologi ( <i>library/framework</i> ) yang sesuai untuk proyek dan melakukan	Merancang, mengembangkan, dan menguji aplikasi <i>web &amp; mobile</i> sesuai dengan kebutuhan klien dan spesifikasi proyek.



		deployment aplikasi ke lingkungan <i>server</i> .	
10	Tim Infrastruktur & Jaringan	Mengelola akses dan hak-hak pada <i>server</i> , serta mengimplementasikan kebijakan keamanan jaringan perusahaan.	Memastikan ketersediaan, keamanan, dan performa seluruh infrastruktur IT perusahaan, termasuk <i>server</i> , database, dan jaringan.

## 3.2 Tata Laksana Sistem Yang Berjalan

### 3.2.1 Prosedur Sistem yang Berjalan

Berdasarkan hasil observasi, proses manajemen aset di PT Fujiyama Technology Solution saat ini masih berjalan secara semi-manual dengan mengandalkan *spreadsheet* dan koordinasi antar-divisi melalui aplikasi pesan. Alur sistem yang berjalan dapat diringkas sebagai berikut:

#### 1. Registrasi & Pencatatan Awal Aset

Admin mencatat data pembelian (nama aset, merk/tipe, tanggal beli, garansi, penanggung jawab, lokasi awal) pada *spreadsheet* induk. Label fisik belum distandardisasi; sebagian aset belum memiliki penanda unik yang konsisten.

#### 2. Perpindahan/Peminjaman Aset

Saat aset dipinjam atau berpindah lokasi, staf memperbarui *spreadsheet* secara manual. Perubahan kadang tercatat terlambat karena bergantung pada pelapor manusia.

#### 3. Pemeriksaan Fisik/Audit Berkala

Tim melakukan pengecekan fisik periodik menggunakan daftar cetak/*spreadsheet*; verifikasi dilakukan satu-per-satu (matching kode/nama vs kondisi/lokasi), sehingga memakan waktu.

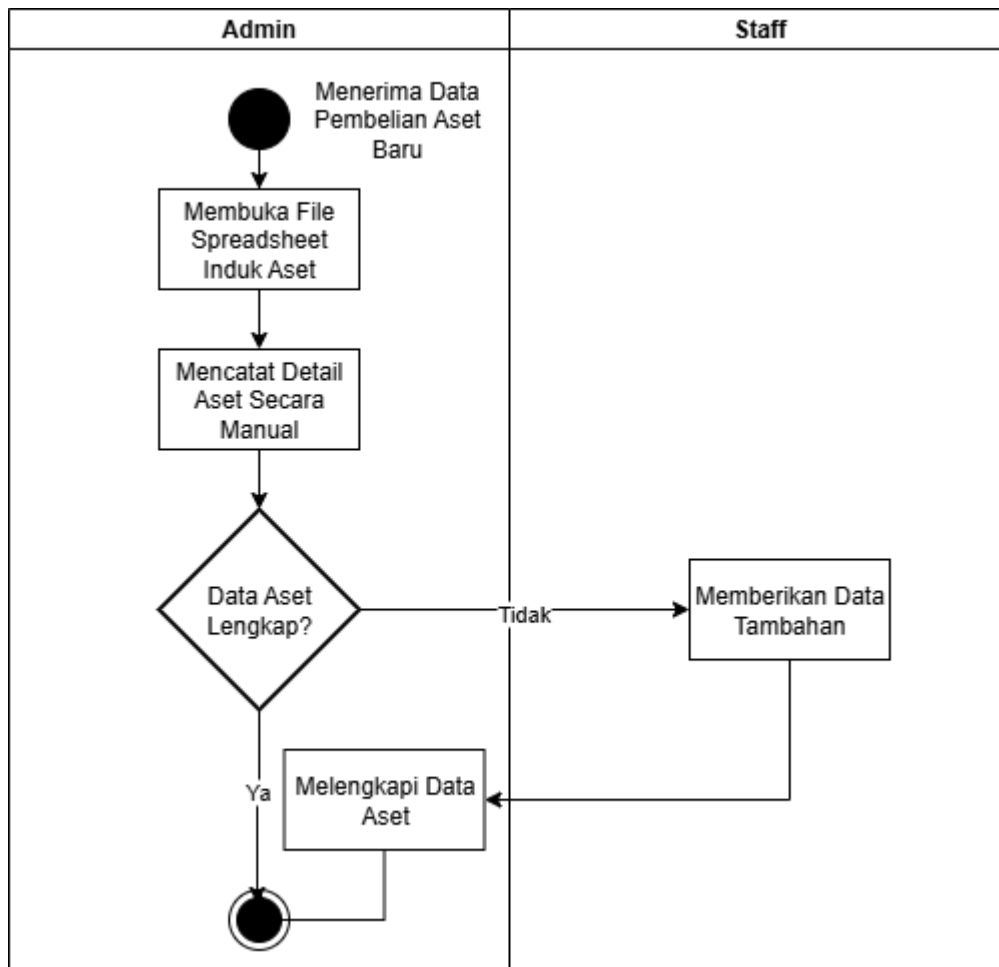
#### 4. Perawatan & Perbaikan

Jadwal perawatan dicatat manual; histori perbaikan tidak selalu terdokumentasi utuh dalam satu sumber terpusat.

### 3.2.2 Activity Diagram Sistem Yang Berjalan

Untuk memberikan gambaran visual yang lebih jelas mengenai alur kerja sistem manajemen aset yang sedang berjalan, dibuatlah sebuah *Activity Diagram*.

Diagram ini memodelkan alur aktivitas dari setiap aktor yang terlibat dalam proses pengelolaan aset saat ini.



Gambar 3. 2 Activity Diagram Sistem yang Berjalan

### 3.3 Masalah yang Dihadapi

Berdasarkan analisis terhadap sistem yang berjalan, teridentifikasi beberapa masalah utama yang dihadapi oleh PT Fujiyama Technology Solution, antara lain:

1. Proses pendataan yang semi-manual rentan terhadap kesalahan *Input* data oleh manusia (*human error*).
2. Sulitnya memantau status, lokasi, dan riwayat penggunaan aset secara *real-time*.
3. Proses pemeriksaan fisik aset berjalan lambat dan tidak efisien karena verifikasi dilakukan secara manual satu per satu.
4. Tidak adanya sistem terpusat yang menyebabkan data aset menjadi tidak konsisten dan sulit diakses oleh pihak yang berkepentingan.

5. Meningkatnya risiko kehilangan atau kerusakan aset akibat kurangnya kontrol dan pengawasan yang sistematis.

### 3.4 Alternatif Pemecahan Masalah

Mengacu pada masalah yang dihadapi dan kompetensi yang dimiliki oleh PT Fujiyama Technology Solution, alternatif pemecahan masalah yang diusulkan adalah pengembangan sebuah "Aplikasi Inventaris dan Pelacakan Aset Berbasis Android dengan Identifikasi *Data Matrix*" yang didukung oleh *backend* terpusat. Solusi ini mencakup beberapa poin utama:

1. Penanda Unik: Setiap aset akan diberikan penanda unik berupa *Data Matrix* yang ditempelkan secara fisik. Kode ini dapat dipindai menggunakan *smartphone* untuk berbagai keperluan seperti registrasi, mutasi, peminjaman, pengembalian, dan audit.
2. Pencatatan Real-time: Setiap perubahan status atau lokasi aset akan langsung dicatat dan disimpan ke dalam basis data terpusat, sehingga dapat dipantau melalui dashboard secara *real-time*.
3. Kontrol Akses Berbasis Peran: Sistem akan menerapkan hak akses yang berbeda untuk setiap peran (misalnya Admin, Staf, dan Karyawan) untuk menjamin keamanan dan relevansi data.
4. Fitur Tambahan: Aplikasi akan dilengkapi dengan fitur untuk menjadwalkan perawatan preventif, mencatat riwayat perbaikan, serta mengirimkan notifikasi untuk kejadian penting.
5. Pelaporan Otomatis: Sistem dapat menghasilkan laporan inventaris, mutasi, dan pemeliharaan yang dapat diekspor untuk kebutuhan audit dan pengambilan keputusan.

### 3.5 User Requirement (Elisitasi)

Tahap elisitasi dilakukan untuk mengumpulkan dan mendefinisikan kebutuhan pengguna secara rinci. Proses ini dibagi menjadi beberapa tahapan untuk memastikan semua kebutuhan teridentifikasi dan dapat diimplementasikan dalam sistem yang baru.

#### 3.5.1 Elisitasi tahap I

Tahap ini berisi daftar kebutuhan awal yang diperoleh dari hasil analisis alur kerja sistem yang diusulkan untuk setiap peran pengguna (Admin, Staff, dan Employee).

Tabel 3. 2 Elisitasi Tahap 1

No	Keterangan
	Fungsional
1	Sistem dapat menampilkan Splash Screen dengan logo perusahaan.
2	Sistem menyediakan otentikasi pengguna (Login/Logout) yang aman menggunakan JWT dan menerapkan kontrol akses berbasis peran (Admin, Staff, Employee).
3	Menampilkan dashboard dinamis yang menyajikan informasi dan menu yang relevan sesuai dengan hak akses masing-masing peran.
4	(Admin) Mengelola data master (Kategori, Lokasi, Pengguna) dan melakukan registrasi aset baru, termasuk pembuatan kode Data Matrix unik secara otomatis.
5	(Admin) Menghasilkan berbagai jenis laporan inventaris dan aktivitas aset yang dapat diekspor ke format PDF.
6	(Staff) Melakukan pemindaian kode Data Matrix menggunakan kamera untuk identifikasi dan pembaruan status aset secara cepat.
7	(Staff) Mencatat aktivitas aset secara real-time, meliputi perpindahan (mutasi), serah terima, dan riwayat perawatan, dengan menyertakan data koordinat GPS.
8	(Employee) Melihat daftar aset yang menjadi tanggung jawabnya dan dapat melaporkan masalah atau kerusakan pada aset tersebut melalui aplikasi.
9	Memvisualisasikan lokasi aset secara real-time di peta (Google Maps).
10	Melacak dan menampilkan riwayat pergerakan sebuah aset di peta (Google Maps).
11	Menyediakan fitur pencarian aset yang komprehensif berdasarkan berbagai kriteria (nama, merek, model).
12	Sistem dapat mengirimkan notifikasi push (via Firebase Cloud Messaging) untuk pengingat penting seperti jadwal perawatan atau status laporan.
13	Mendukung antarmuka multibahasa (Inggris dan Jepang) di seluruh bagian aplikasi.
	Non-Fungsional

1	Aplikasi memiliki antarmuka yang intuitif, mudah digunakan (user-friendly), dan responsif.
2	Sistem memiliki tingkat keamanan yang tinggi untuk melindungi data aset perusahaan.
3	Aplikasi harus berjalan dengan performa yang cepat dan andal pada berbagai perangkat Android.

### 3.5.2 Elisitasi tahap II

Pada tahap ini, kebutuhan dari Tahap I diklasifikasikan menggunakan metode MDI (*Mandatory, Desirable, Inessential*) untuk menentukan prioritas.

Tabel 3. 3 Elisitasi Tahap 2

No	Keterangan	M	D	I
	Fungsional			
1	Sistem dapat menampilkan Splash Screen dengan logo perusahaan.			✓
2	Sistem menyediakan otentikasi pengguna (Login/Logout) yang aman menggunakan JWT dan menerapkan kontrol akses berbasis peran (Admin, Staff, Employee).	✓		
3	Menampilkan dashboard dinamis yang menyajikan informasi dan menu yang relevan sesuai dengan hak akses masing-masing peran.	✓		
4	(Admin) Mengelola data master (Kategori, Lokasi, Pengguna) dan melakukan registrasi aset baru, termasuk pembuatan kode Data Matrix unik secara otomatis.	✓		
5	(Admin) Menghasilkan berbagai jenis laporan inventaris dan aktivitas aset yang dapat diekspor ke format PDF.	✓		
6	(Staff) Melakukan pemindaian kode Data Matrix menggunakan kamera untuk identifikasi dan pembaruan status aset secara cepat.	✓		

7	(Staff) Mencatat aktivitas aset secara real-time, meliputi perpindahan (mutasi), serah terima, dan riwayat perawatan, dengan menyertakan data koordinat GPS.	✓		
8	(Employee) Melihat daftar aset yang menjadi tanggung jawabnya dan dapat melaporkan masalah atau kerusakan pada aset tersebut melalui aplikasi.	✓		
9	Memvisualisasikan lokasi aset secara real-time di peta (Google Maps).		✓	
10	Melacak dan menampilkan riwayat pergerakan sebuah aset di peta (Google Maps).	✓		
11	Menyediakan fitur pencarian aset yang komprehensif berdasarkan berbagai kriteria (nama, merek, model).	✓		
12	Sistem dapat mengirimkan notifikasi push (via Firebase Cloud Messaging) untuk pengingat penting seperti jadwal perawatan atau status laporan.	✓		
13	Mendukung antarmuka multibahasa (Inggris dan Jepang) di seluruh bagian aplikasi.		✓	
	Non-Fungsional			
1	Aplikasi memiliki antarmuka yang intuitif, mudah digunakan (user-friendly), dan responsif.	✓		
2	Sistem memiliki tingkat keamanan yang tinggi untuk melindungi data aset perusahaan.	✓		
3	Aplikasi harus berjalan dengan performa yang cepat dan andal pada berbagai perangkat Android.	✓		

### 3.5.3 Elisitasi tahap III

Kebutuhan yang lolos dari Tahap II dianalisis kembali menggunakan metode TOE (*Technical, Operational, Economy*) dengan skala *Low (L)* dan *Middle (M)*. Tidak ada yang dinilai *High (H)* karena semua dianggap dapat dikerjakan dalam lingkup laporan skripsi.

Tabel 3. 4 Elisitasi Tahap 3

No	Keterangan	T			O			E		
		H	M	L	H	M	L	H	M	L
	Fungsional									
1	Otentikasi Pengguna dengan JWT dan Kontrol Akses Berbasis Peran.			✓			✓			✓
2	Dashboard Dinamis Berbasis Peran.		✓				✓			✓
3	(Admin) Pengelolaan Data Master dan Registrasi Aset (Termasuk Generate Data Matrix).		✓				✓			✓
4	(Admin) Generasi dan Ekspor Laporan ke PDF.			✓			✓			✓
5	(Staff) Fungsionalitas Pemindaian Data Matrix via Kamera.		✓			✓				✓
6	(Staff) Pencatatan Aktivitas Aset (Mutasi, Perawatan) dengan GPS.		✓				✓			✓
7	(Employee) Fitur Melihat Aset dan Melaporkan Masalah.			✓			✓			✓
8	Visualisasi Lokasi Aset Real-time di Google Maps.		✓			✓				✓
9	Pelacakan Riwayat Pergerakan Aset di Peta.			✓			✓			✓
10	Fitur Pencarian Aset yang Komprehensif.	✓					✓			✓
11	Notifikasi Push via Firebase Cloud Messaging (FCM).	✓					✓		✓	
12	Dukungan Antarmuka Multibahasa.			✓			✓			✓

	Non-Fungsional								
1	Antarmuka <i>User-Friendly</i> dan Responsif.			✓			✓		✓
2	Keamanan Sistem yang Tinggi.			✓			✓		✓
3	Performa Aplikasi Cepat dan Andal.		✓				✓		✓

### 3.5.4 Final Elisitasi

Final Elisitasi adalah daftar akhir dari kebutuhan yang akan diimplementasikan dalam sistem, yang telah disetujui setelah melalui seluruh tahapan analisis.

Tabel 3. 5 *Final Draft* Elisitasi

No	Keterangan
	Fungsional
1	Sistem menyediakan otentikasi pengguna (Login/Logout) yang aman dan menerapkan kontrol akses berbasis peran (Admin, Staff, Employee).
2	Menampilkan dashboard dinamis yang menyajikan menu relevan sesuai hak akses peran.
3	(Admin) Mengelola data master (Kategori, Lokasi, Pengguna) dan melakukan registrasi aset baru dengan pembuatan kode Data Matrix otomatis.
4	(Admin) Menghasilkan laporan inventaris yang dapat diekspor ke format PDF.
5	(Staff) Melakukan pemindaian kode Data Matrix untuk identifikasi dan pembaruan status aset.
6	(Staff) Mencatat aktivitas aset secara real-time, seperti perpindahan (mutasi) dan perawatan, dengan data koordinat GPS.
7	(Employee) Melihat daftar aset yang menjadi tanggung jawabnya dan dapat melaporkan masalah atau kerusakan.
8	Memvisualisasikan lokasi aset secara real-time pada peta (Google Maps).



9	Menyediakan fitur pencarian aset berdasarkan berbagai kriteria.
10	Mendukung antarmuka multibahasa (Inggris dan Jepang) di seluruh aplikasi.
	Non-Fungsional
1	Aplikasi memiliki antarmuka yang intuitif dan responsif.
2	Sistem memiliki keamanan yang tinggi untuk melindungi data.
3	Aplikasi harus berjalan dengan performa yang cepat dan andal.
Tangerang, .....	
Penyusun	
Rizqiansyah Ramadhan	
1122140051	
Mengetahui	
Stakeholder	Pembimbing Utama

## **BAB IV**

### **RANCANGAN SISTEM YANG DIUSULKAN**

## **BAB V**

### **PENUTUP**

## DAFTAR PUSTAKA

- Aung, S. T., Funabiki, N., Aung, L. H., Kinari, S. A., Mentari, M., & Wai, K. H. (2024). A Study of Learning Environment for Initiating Flutter App Development Using Docker. *Information (Switzerland)*, 15(4). <https://doi.org/10.3390/info15040191>
- Ayumida, S., Hakim, L., Azis, M. S., Mahaulika, C., Bina, U., Informatika, S., & Mandiri, U. N. (2021). SISTEM INFOMASI PENYEWAAN LAPANGAN OLAHRAGA MENGGUNAKAN METODE WATERFALL PADA GREEN GARDEN SPORT CENTER. Dalam *Ijns.org Indonesian Journal on Networking and Security* (Vol. 10). Online.
- Bahat Nauli, S. (2024). PERANCANGAN APLIKASI PERSEDIAAN BARANG MENGGUNAKAN BARCODE QUICK RESPONSE DENGAN METODE FIRST-IN FIRST-OUT BERBASIS MOBILE (STUDI KASUS: PT KOPI KENANGAN). Dalam *Jurnal Riset Ilmiah* (Vol. 3, Nomor 4).
- Bajaj, B. S. (2024). *Reliability on QR codes and Reed-Solomon codes*. <http://arxiv.org/abs/2407.17364>
- Choudhury, A., Asan, O., & Mansouri, M. (2021). *Role of Artificial Intelligence, Clinicians & Policymakers in Clinical Decision Making: A Systems Viewpoint*.
- Christian, C., & Voutama, A. (2024). RANCANG BANGUN APLIKASI SISTEM INFORMASI INVENTARIS BERBASIS WEBSITE. *Jurnal Informatika dan Teknik Elektro Terapan*, 12(2). <https://doi.org/10.23960/jitet.v12i2.4259>
- Coelho, F., & Yannou-Medrala, C. (2023). *JSON Model: a Lightweight Featureful Description Language for JSON Data Structures Fabien Coelho, Claire Yannou-Medrala. JSON Model: a Lightweight Featureful Description Language for JSON Data Structures. Mines JSON Model: a Lightweight Featureful Description Language for JSON Data Structures*. <https://minesparis-psl.hal.science/hal-04415527v1>
- Con, R., Shpilka, A., & Tamo, I. (2024). *Optimal Two-Dimensional Reed--Solomon Codes Correcting Insertions and Deletions*. <http://arxiv.org/abs/2311.02771>
- Dalimunthe, S., Hasri Putra, E., & Fadhly Ridha, M. A. (2023). Restful API Security Using JSON Web Token (JWT) With HMAC-Sha512 Algorithm in Session Management. *IT Journal Research and Development*, 8(1), 81–94. <https://doi.org/10.25299/itjrd.2023.12029>
- Fahmi, M., Manajemen, A., Keimigrasian, T., & Imigrasi, P. (2024). Analisis Implementasi Sistem Informasi: Studi Literatur Analysis Of Information System Implementation: Literature Review. Dalam *JTSI* (Vol. 5, Nomor 1).
- Ferry Gunawan, Grace Martha, & G. Bororing. (2024). *Sistem Pemesanan dan Pembayaran Makanan berbasis Web Terintegrasi dengan Application Programming Interface (API)*.
- Fiber Documentation. (2025, Oktober 23). *Benchmarks | Fiber*. GoFiber. <https://docs.gofiber.io/>

- Gómez, O. S., Rosero, R. H., Estrada-Gutiérrez, J. C., & Jiménez-Rodríguez, M. (2022). An Approach for Securing JSON Objects through Chaotic Synchronization. *Cybernetics and Information Technologies*, 22(4), 23–34. <https://doi.org/10.2478/cait-2022-0037>
- Gunawan, R., Jumadhi, N., & Bakhri, A. S. (2024). Rancang Bangun Sistem Informasi Manajemen Aset Berbasis Web (Studi Kasus : SDN Ciwaringin 3). *Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi dan Komunikasi*, 19(1), 34–40. <https://doi.org/10.35969/interkom.v19i1.345>
- Hartomo, K. D., Setiyawati, N., & Bangkalang, D. H. (2023). Implementasi dan Pelatihan Aplikasi Manajemen Aset Gereja Berbasis Progressive Web Application. *ABDIMASKU : JURNAL PENGABDIAN MASYARAKAT*, 6(3), 735. <https://doi.org/10.62411/ja.v6i3.1560>
- Ipinuwati, S., Maselena, A., & Prayoga, B. D. (2022). The Design of Goods Data Storage Application based on Android using Barcode Scanner. *International Journal of Information technology and Computer Engineering*, 23, 1–12. <https://doi.org/10.55529/ijitc23.1.12>
- Karrach, L., & Pivarčiová, E. (2021). Comparative Study of Data Matrix Codes Localization and Recognition Methods. *Journal of Imaging*, 7(9), 163. <https://doi.org/10.3390/jimaging7090163>
- Kinari, S. A., Funabiki, N., Aung, S. T., Wai, K. H., Mentari, M., & Puspitaningayu, P. (2024). An Independent Learning System for Flutter Cross-Platform Mobile Programming with Code Modification Problems. *Information (Switzerland)*, 15(10). <https://doi.org/10.3390/info15100614>
- M. Fadhli Alpharajasa, Agi Putra Kharisma, & Aryo Pinandito. (2023). *Pengembangan Aplikasi Perangkat Bergerak Sistem Pendataan Inventaris UKM FILKOM UB berbasis Android*.
- Malhotra, A. (2025). *Concurrency Patterns in Golang: Real-World Use Cases and Performance Analysis*. <https://doi.org/10.32996/jcsts>
- Mardani, R. M., Suharso, W., & Nuryasin, I. (2023). Pengembangan Sistem Manajemen Aset Pada PT. PELINDO IV Cabang Balikpapan Berbasis Android. *REPOSITOR*, 5(3), 737–746.
- Muhamad Yunus Aliviyudin, Mega Tri Kurnia, & Febri Dolis Herdiani. (2025). *Perancangan dan Implementasi Sistem Informasi Inventory Barang Berbasis Web pada PT XYZ*.
- Mukti, A., Hadiyanti, A. D., Nurlaela, A., & Panjaitan, J. (2023). *Sistem Analisa Sentiment Bakal Calon Presiden 2024 Menggunakan Metode NLP Berbasis Web The Sentiment Analysis System For the 2024 Presidential Candidates Uses Web-Based NLP Method* (Vol. 6, Nomor 1).
- Nur Elfi Husda, Suhardi Suhardi, Inda Sukati, & Welly Sugianto. (2023). *Metodologi Penelitian: Kualitatif, Kuantitatif dan Research and Development (R&D)*.

- Pasaribu, J. S. (2021). PERANCANGAN SISTEM INFORMASI BERBASIS WEB PENGELOLAAN INVENTARIS ASET KANTOR DI PT. MPM FINANCE BANDUNG. Dalam *Jurnal Ilmiah Teknologi Informasi Terapan* (Vol. 7, Nomor 3).
- Pirrone, M., & D'Ulizia, A. (2024). The Localization of Software and Video Games: Current State and Future Perspectives. *Information (Switzerland)*, 15(10). <https://doi.org/10.3390/info15100648>
- Prasetyo, D., Matani Raya, J., Klp Lima, K., Kupang, K., & Tenggara Timur, N. (2024). *Rancang Bangun Sistem Informasi Manajemen Persediaan Barang Menggunakan Metode System Development Life Cycle (SDLC) Di Sma Negeri 1 SoE-(Dwi Prasetyo)* RANCANG BANGUN SISTEM INFORMASI MANAJEMEN PERSEDIAAN BARANG MENGGUNAKAN METODE SYSTEM DEVELOPMENT LIFE CYCLE (SDLC) DI SMA NEGERI 1 SoE (Vol. 14, Nomor 2).
- Purnomo, S., & Alijoyo, F. A. (2024). Sistem Peminjaman Barang Menggunakan QR Code Berbasis Aplikasi Android. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, 6(2), 322–328. <https://doi.org/10.47233/jteksis.v6i2.1350>
- Rajan, R., & Sulthana, S. (2022). *Inventory Management System in Mobile-Based Point of Sale* CENTRAL ASIAN JOURNAL OF THEORETICAL AND APPLIED SCIENCES *Inventory Management System in Mobile-Based Point of Sale*. <https://www.researchgate.net/publication/360877453>
- Rakhma, S. A., Tullah, R., & Mustafa, S. M. (2022). *Sistem Monitoring Data Aset dan Inventaris IT Berbasis Web pada PT. Pan Brothers Tbk* (Vol. 1, Nomor 2). *Jurnal Teknologi*.
- Saioc, G.-V., Shirchenko, D., & Chabbi, M. (2023). *Unveiling and Vanquishing Goroutine Leaks in Enterprise Microservices: A Dynamic Analysis Approach*. <http://arxiv.org/abs/2312.12002>
- Sidik, F. (2022). INPUT, PROCESS AND OUTPUT SYSTEM THEORY APPROACH IN EDUCATIONAL INSTITUTIONS. *Irfani*, 18(1), 34–40. <https://doi.org/10.30603/ir.v18i1.2658>
- Sinatria, M. B., Oman Komarudin, & Kamal Prihamdani. (2023). PENERAPAN CLEAN ARCHITECTURE DALAM MEMBANGUN APLIKASI BERBASIS MOBILE DENGAN FRAMEWORK GOOGLE FLUTTER. *INFOTECH journal*, 9(1), 132–146. <https://doi.org/10.31949/infotech.v9i1.5237>
- Sung, W. T., Isa, I. G. T., & Hsiao, S. J. (2023). An IoT-Based Aquaculture Monitoring System Using Firebase. *Computers, Materials and Continua*, 76(2), 2180–2200. <https://doi.org/10.32604/cmc.2023.041022>
- Taipalus, T. (2023). *Database management system performance comparisons: A systematic literature review*. <https://doi.org/10.1016/j.jss.2023.111872>
- Tuti Amiasih, & Andiani. (2022). *SISTEM INFORMASI MANAJEMEN ASET (STUDI KASUS PERUSAHAAN Y)*.

- Usnaini, M., Yasin, V., & Sianipar, A. Z. (2021). Perancangan sistem informasi inventarisasi aset berbasis web menggunakan metode waterfall. *Jurnal Manajemen Informatika Jayakarta*, 1(1), 36. <https://doi.org/10.52362/jmijayakarta.v1i1.415>
- Weifan Liu. (2022). *The Implications of Object-Oriented Analysis and Design*.
- Wijayanto, R. A., Hajar, R. R., & Sejati, P. (2023). Implementing Flutter Clean Architecture for Mobile Tourism Application Development. Dalam *International Journal of Computer Applications* (Vol. 185, Nomor 39).
- Willyana, J. (2022a). Sistem Informasi Pendataan dan Peminjaman Aset Perusahaan Berbasis Web Pada PT.Vadhana International. *Jurnal Mahasiswa Aplikasi Teknologi Komputer dan Informasi*, 4(3), 123–127.
- Willyana, J. (2022b). Sistem Informasi Pendataan dan Peminjaman Aset Perusahaan Berbasis Web Pada PT.Vadhana International. *Jurnal Mahasiswa Aplikasi Teknologi Komputer dan Informasi*, 4(3), 123–127.

## **DAFTAR LAMPIRAN**