

BAB II

LANDASAN TEORI

Bab ini berfungsi sebagai dasar konseptual untuk mendukung analisis dan pembahasan. Setiap teori yang digunakan harus relevan dengan permasalahan penelitian. Bab ini berisi teori-teori dari para ahli yang telah teruji secara empiris dan digunakan untuk menjelaskan variabel-variabel dalam penelitian. Selain itu, bab ini juga memuat penelitian terdahulu yang relevan sebagai pembanding dan penguat posisi penelitian.

2.1. Kajian Teori

2.1.1. Sistem Informasi Manajemen Gudang (*Warehouse Management*)

Sistem Informasi Manajemen Gudang (*Warehouse Management System*) merepresentasikan sinergi antara teknologi informasi dengan aktivitas operasional manusia guna mendukung pemantauan sirkulasi stok secara komprehensif. Gudang sendiri merupakan fasilitas sentral dan komponen krusial dalam rantai manajemen pasokan dan logistik perusahaan. Fokus utama dari sistem ini terletak pada pengendalian pergerakan barang masuk (*stock in*) dan barang keluar (*stock out*) secara presisi demi menjamin stabilitas ketersediaan inventaris di dalam gudang. Lebih lanjut, sistem manajemen gudang yang dirancang dengan baik tidak hanya mempermudah pengelolaan data barang, tetapi juga bertindak sebagai alat strategis yang mampu meningkatkan efisiensi dan efektivitas operasional, sehingga memungkinkan perusahaan untuk mencapai tujuan bisnis dan mendapatkan keunggulan kompetitif (Satryawati et al., 2023).

2.1.2. Manajemen Bahan Baku (*Raw Material*)

Dalam manajemen logistik dan operasional, bahan baku (*raw material*) merupakan material utama yang dibeli dari pemasok untuk diolah menjadi produk jadi yang dihasilkan oleh perusahaan (Rochman & Waluyowati, 2024). Manajemen bahan baku yang optimal mencakup aspek pengendalian persediaan material dasar untuk meminimalisir biaya simpan sekaligus menghindari hilangnya kesempatan

keuntungan akibat *stockout* (kekosongan persediaan) (Rochman & Waluyowati, 2024).

Tanpa adanya sistem pengendalian persediaan yang baik, perusahaan dihadapkan pada risiko ketidakmampuan untuk memenuhi fluktuasi permintaan, yang pada akhirnya dapat menghambat kontinuitas kegiatan produksi secara keseluruhan, untuk memberikan tingkat proteksi terhadap risiko habisnya persediaan akibat lonjakan permintaan yang tidak menentu.

2.1.3. Metode *Prototype*

Metode *Prototype* adalah metode pengembangan perangkat lunak iteratif yang memungkinkan pengembang dan pengguna untuk berkolaborasi dalam membangun prototipe sistem yang kemudian disempurnakan berdasarkan umpan balik pengguna (Lawolo et al., 2025). Pendekatan ini dikenal sebagai cara yang fleksibel, efisien, dan berpusat pada pengguna, sehingga sangat membantu dalam mengelola ketidakpastian, meminimalkan risiko, dan menghasilkan produk yang lebih sesuai dengan kebutuhan pasar dan pengguna (Lawolo et al., 2025). Tahapan-tahapan sistematis dalam metode ini meliputi:

1. Kebutuhan Awal (*Initial Requirement*), merupakan proses awal untuk memahami kebutuhan dasar pengguna. Pada tahap ini, dilakukan komunikasi intensif dengan pihak PT. Plastindo Pack Printing untuk memetakan spesifikasi sistem secara akurat.
2. Perancangan dan Pembuatan Prototipe (*Design & Prototyping*), merancang bentuk awal sistem dan membangun kerangka sementara yang berfokus pada penyajian alur kerja serta tampilan antarmuka (UI/UX) sebagai representasi aplikasi.
3. Evaluasi dan Pembaruan (*Customer Evaluation & Review*), melibatkan pengguna untuk mengevaluasi apakah prototipe yang dibangun sudah sesuai dengan keinginan. Umpan balik yang konstruktif dari pengguna digunakan untuk melakukan revisi prototipe secara berulang hingga pengguna merasa puas (Lawolo et al., 2025).

4. Pengembangan Sistem Final (*Development*), setelah prototipe dianggap sesuai dan disetujui, sistem final dibangun dengan fitur lengkap dan implementasi teknis secara menyeluruh.

Metodologi ini dipandang sangat efektif bagi kelompok pengguna dengan tingkat literasi digital terbatas, karena memungkinkan visualisasi fungsional aplikasi sebelum tahap finalisasi sistem.

2.1.4. Joint Application Design

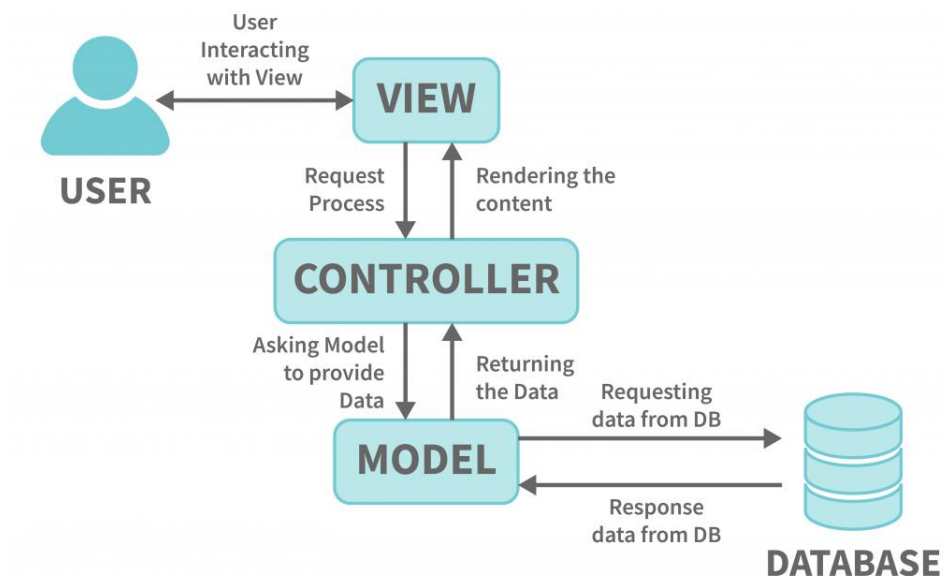
Joint Application Design (JAD) merupakan metode pengembangan sistem yang menekankan keterlibatan aktif pengguna dalam proses analisis dan perancangan sistem. Metode ini dilakukan melalui diskusi terstruktur yang melibatkan berbagai pihak terkait untuk mengidentifikasi kebutuhan sistem secara bersama-sama.

Menurut Aldisa et al. (2022), JAD adalah teknik yang menggabungkan pengguna, analis, dan manajemen dalam satu forum untuk menentukan kebutuhan sistem secara lebih akurat.

Selain itu, metode JAD juga dapat mempercepat proses pengembangan sistem dan meminimalisir kesalahan dalam perancangan karena adanya komunikasi langsung antara pengguna dan pengembang.

2.1.5. Framework Laravel 12

Laravel merupakan *framework* PHP yang banyak digunakan dalam pengembangan aplikasi web karena menyediakan struktur pengembangan yang terorganisir dan mudah dipelihara. Laravel menerapkan konsep *Model-View-Controller* (MVC), yaitu pemisahan antara logika program, tampilan antarmuka, dan pengolahan data, sehingga proses pengembangan aplikasi menjadi lebih terstruktur dan sistematis, serta fleksibilitas dalam konstruksi RESTful API untuk diintegrasikan dengan platform *mobile*. Keunggulan lain dari ekosistem Laravel adalah ketersediaan modul keamanan khusus seperti Laravel Sanctum, yang terbukti handal dalam menangani proses autentikasi atau otorisasi pengguna serta pengelolaan *token* API yang aman untuk melindungi data sistem (Fauzi & Hardjianto, 2022).



Gambar 2.1. Alur kerja Model-View-Controller

(Sumber: www.rumahstudio.com)

Berdasarkan Gambar 2.1, permintaan dari pengguna pertama kali diproses oleh Controller, kemudian Controller akan berkomunikasi dengan Model untuk mengambil atau mengolah data yang diperlukan. Hasil pengolahan data tersebut selanjutnya dikirim kembali ke View untuk ditampilkan kepada pengguna.

Sebagai representasi implementasi MVC pada *backend*, berikut adalah contoh penulisan sintaks PHP pada bagian *Controller* untuk mengolah dan menampilkan data persediaan bahan baku:

```

class SparepartController extends Controller
{
    public function index(Request $request)
    {
        $query = Sparepart::query();

        if ($request->search) {
            $query->where('sparepart', 'like', "%{$request->search}%");
        }
        if ($request->kategori) {
            $query->where('kategori', $request->kategori);
        }
        if ($request->filter === 'low') {
            $query->whereColumn('stok', '<', 'stok_ideal');
        }
    }
}
  
```

Gambar 2.2. Contoh sintaks PHP pada *Controller* Laravel

Pada sisi presentasi data (*View*), antarmuka web dibangun menggunakan elemen dasar HTML yang dikelola oleh *template engine* bawaan Laravel (Blade) untuk menampilkan rekapan data gudang secara visual. Berikut adalah contoh struktur HTML untuk tabel manajemen *sparepart*:

```

1 <div class="row align-items-center">
2   <div class="col-md-6">
3     <h1 class="m-0">Daftar Sparepart</h1>
4   </div>
5   <div class="col-md-6 text-right">
6     <div id="current-time" class="text-muted">
7       <i class="far fa-clock"></i> <span id="clock"></span>
8     </div>
9     <div class="text-muted">
10      <i class="far fa-calendar-alt"></i> <span id="date"></span>
11    </div>
12  </div>
13</div>
14<div class="row align-items-center mt-2">
15  <div class="col-6">
16    <span class="text-muted small" id="live-indicator" title="Data diperbarui otomatis">
17      <span class="live-dot" id="live-dot"></span> Live
18    </span>
19  </div>
20  <div class="col-6 text-right">
21    @if(auth()->user()->role === 'admin')
22      <button class="btn btn-success btn-sm" onclick="openAddModal()">
23        <i class="fas fa-plus"></i> Tambah Sparepart
24      </button>
25    @endif
26  </div>
27</div>

```

Gambar 2.3. Contoh Sintaks HTML

2.1.6. Framework Flutter

Flutter merupakan SDK (*Software Development Kit*) inovasi Google yang menggunakan bahasa pemrograman Dart, dirancang khusus untuk membangun aplikasi mobile hybrid melalui satu basis kode (*single codebase*). Keunggulan Flutter mencakup performa yang setara dengan aplikasi native serta kemudahan dalam pengembangan antarmuka reaktif yang mendukung fungsionalitas identifikasi visual. Penggunaan Flutter yang dikombinasikan dengan integrasi RESTful API terbukti mampu menghasilkan aplikasi mobile (seperti Android) yang berjalan lancar, terstruktur, dan sangat cepat dalam mengeksekusi request data dari server (Fauzi & Hardjianto, 2022).

Penggunaan *framework* seperti Laravel terbukti efektif dalam membangun sistem inventaris yang sistematis dan terstruktur, yang mampu meningkatkan performa pengolahan data aset secara signifikan (Iqbal et al., 2019).

Untuk mengimplementasikan komunikasi data dari perangkat pengguna operasional ke *server* pusat, berikut adalah contoh bahasa pemrograman Dart yang digunakan dalam Flutter saat menarik data barang:

```
Future<Response> getSpareparts({String? search, String? category}) async {
  return await _dio.get('/sparepart', queryParameters: {
    if (search?.isNotEmpty == true) 'search': search,
    if (category?.isNotEmpty == true) 'kategori': category,
  });
}
```

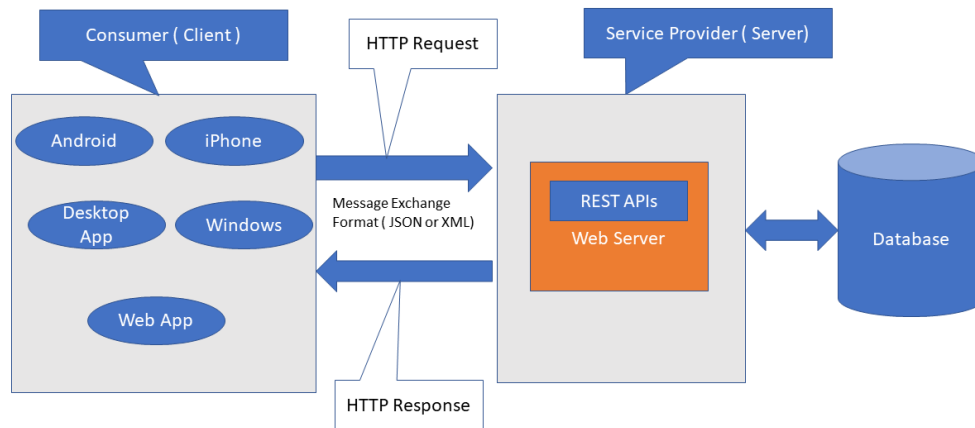
Gambar 2.4. Contoh Sintaks Dart

2.1.7. REST API

REST (*Representational State Transfer*) adalah sebuah gaya arsitektur perangkat lunak yang sederhana, dapat diskalakan, efektif, aman, dan andal yang menggunakan protokol HTTP untuk pertukaran data. Arsitektur ini memungkinkan aplikasi yang berbeda, dalam hal ini aplikasi *mobile* Flutter sebagai *frontend* dan Laravel 12 sebagai *backend*—untuk saling berkomunikasi secara lancar meskipun dibangun dengan bahasa pemrograman yang berbeda.

Dalam praktiknya, REST API menyediakan metode HTTP utama seperti POST, GET, PUT, dan DELETE pada masing-masing *endpoint* yang bertugas untuk mengoperasikan fungsi CRUD (*Create, Read, Update, Delete*) di dalam basis data (Farhandika et al., 2024). Proses pertukaran dan transfer data antara *server* dan *client* ini menggunakan JSON (*JavaScript Object Notation*), yaitu standar format berbasis teks untuk merepresentasikan data secara terstruktur, Format JSON memiliki keunggulan berupa struktur *key-value* yang ringan sehingga mempercepat waktu pemrosesan respons sistem.

REST – Architecture



By Ramesh Fadattare (Java Guides)

Gambar 2.5. Arsitektur REST API

(Sumber: www.javaguides.net)

Salah satu prinsip utama dari REST adalah arsitektur *Client-Server*, yang mendeskripsikan pemisahan pengembangan secara mandiri antara antarmuka pengguna (*Client*) dan pengolah data (*Server*) (Farhandika et al., 2024). Dalam rancangan sistem ini, aplikasi *mobile* bertindak sebagai *client* yang mengirimkan permintaan (*request*), sementara *framework* Laravel bertindak sebagai *server* yang memproses logika serta memberikan balasan (*response*) dari *database* MySQL. Sebagaimana ditegaskan oleh Iqbal et al. (2019), penggunaan API ini sangat krusial dalam sistem informasi modern untuk memastikan integrasi data yang sistematis dan terstruktur antara server dan klien.

```

1  {
2      "message": "Login berhasil",
3      "token": "22|Xi3QX2DF5WjcHGLt9tZsi9ueLX3Ye4bFvTX5PyL692b68f11",
4      "user": {
5          "id": 1,
6          "name": "Administrator",
7          "email": "admin@mail.com",
8          "role": "admin"
9      }
10 }
```

Gambar 2.6. Respons sintaks JSON

Arsitektur Client-Server, REST memisahkan antara antarmuka pengguna (Client) dan pengolah data (Server). Dalam sistem ini, aplikasi mobile Flutter bertindak sebagai client yang mengirim permintaan, sementara framework Laravel 12 bertindak sebagai server yang menyediakan data dari database MySQL.

2.1.8. UML

Unified Modeling Language (UML) adalah sebuah standarisasi bahasa pemodelan berorientasi objek yang digunakan untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak (Aryani et al., 2024). UML juga merupakan kumpulan pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak yang terkait dengan objek.

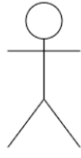
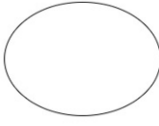
Tujuan utama dari penggunaan UML diagram adalah untuk membantu tim pengembangan proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi desain arsitektur perangkat lunak atau pembuat program.

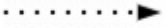
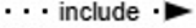
Dalam perancangan sistem informasi manajemen stok ini, beberapa pemodelan UML yang digunakan antara lain:

1. Use Case Diagram

Use case diagram adalah gambaran fungsionalitas suatu sistem yang mendeskripsikan interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Aryani et al., 2024). Diagram ini menunjukkan batasan sistem dan apa saja yang dapat dilakukan oleh pengguna terhadap sistem tersebut.

Tabel 2.1. *Use Case Diagram*

Simbol	Nama	Keterangan
	<i>Actor</i>	Mewakili peran pengguna untuk mewakili interaksi dengan sistem.
	<i>Use Case</i>	merupakan abstraksi dari fungsionalitas, interaksi, atau proses spesifik yang disediakan oleh sistem untuk aktor.

Simbol	Nama	Keterangan
	<i>Association</i>	garis penghubung fisik yang menunjukkan adanya interaksi atau jalur komunikasi langsung antara aktor dengan <i>use case</i> .
	<i>Generalization</i>	Menunjukkan spesialisasi Aktor untuk dapat berpartisipasi dengan <i>use case</i>
	<i>Relationship (Include)</i>	Menyatakan bahwa suatu rangkaian tindakan adalah bagian fungsi yang dipanggil oleh <i>use case</i> lainnya.
	<i>Relationship (Extend)</i>	Menunjukkan bahwa suatu <i>use case</i> dapat menambah fungsi lain dalam kondisi tertentu

2. Class Diagram

Class diagram adalah sebuah diagram yang menunjukkan hubungan antar *class* yang di dalamnya terdapat atribut dan fungsi (operasi) dari suatu objek (Arianti et al., 2022). Diagram ini sangat berguna dalam memberikan gambaran umum tentang bagaimana entitas di dalam sistem (seperti data barang, data pengguna) saling berhubungan dan berinteraksi.

Tabel 2.2. *Class Diagram*

Simbol	Nama	Keterangan
	<i>Generalization</i>	Hubungan antara objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
	<i>Nary Association</i>	menghindari asosiasi dengan lebih dari 2 objek.

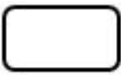
Simbol	Nama	Keterangan
	<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
	<i>Realization</i>	Operasi yang benar benar dilakukan oleh suatu objek.
	<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang berdiri sendiri
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

3. Activity Diagram

Activity diagram merupakan diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem, proses bisnis, maupun navigasi menu yang ada pada perangkat lunak (Aryani et al., 2024). Diagram ini memodelkan konsep aliran kontrol aksi yang terstruktur secara berurutan, dari proses awal hingga akhir (Arianti et al., 2022).

Tabel 2.3. *Activity Diagram*

Simbol	Nama	Keterangan
	Titik Awal/ <i>Start</i>	Menandakan proses tersebut dimulai

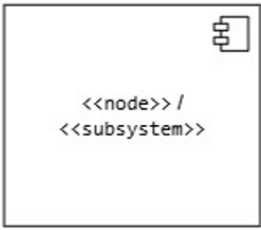
Simbol	Nama	Keterangan
	<i>Activity</i>	aktivitas yang dilakukan atau sedang terjadi dalam sistem.
	<i>Decision</i> atau <i>Percabangan</i>	suatu titik atau point yang mengindikasikan suatu kondisi di mana adanya kemungkinan dalam perbedaan transisi.
	<i>Fork Node</i>	Pada tahap tertentu satu aliran akan berubah menjadi beberapa aliran
	<i>Swimlanes</i>	menjadi kolom dan baris untuk membagi tanggung jawab objek-objek yang melakukan suatu aktivitas.
	Titik Akhir/ <i>Final Node</i>	menandakan proses tersebut berakhir.

4. **Component Diagram**

Component Diagram adalah salah satu diagram UML yang berguna untuk memodelkan aspek logis maupun fisik dari implementasi suatu sistem, digunakan untuk memvisualisasikan organisasi dan hubungan antar komponen dalam suatu sistem (Munawar, 2021).

Tabel 2.4. *Component Diagram*

Simbol	Nama	Keterangan
	<i>Component</i>	Mewakili bagian modular dan independen dari sistem perangkat lunak. Komponen ini menyembunyikan detail kode internalnya dan hanya berinteraksi dengan dunia luar melalui antarmuka (<i>interface</i>).
	<i>Node/Subsystem</i>	Mewakili lingkungan eksekusi fisik atau pengelompokan logis sistem. Digunakan untuk

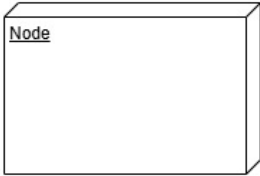
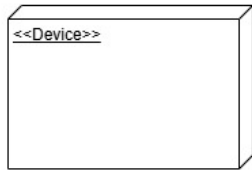
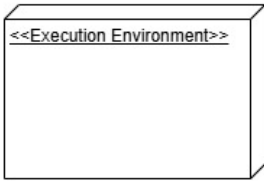
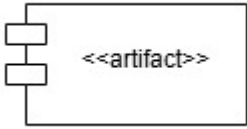
Simbol	Nama	Keterangan
		mengelompokkan komponen-komponen mana saja yang hidup di dalam ruang lingkup tersebut.
	<i>Required Interface</i>	Menandakan bahwa sebuah komponen membutuhkan layanan, data, atau akses dari komponen lain agar bisa berfungsi.
	<i>Provided Interface</i>	Menandakan bahwa sebuah komponen menyediakan layanan atau jalur komunikasi yang bisa diakses dan digunakan oleh komponen lain.
	<i>Dependency</i>	Menunjukkan bahwa satu komponen bergantung pada komponen lain secara internal. Perubahan pada komponen yang ditunjuk panah dapat memengaruhi komponen yang menunjuk. Digunakan untuk komunikasi internal di dalam satu <i>node</i> yang sama.
	<i>Connector / Association</i>	Digunakan untuk menghubungkan sebuah komponen fisik (kotak Component) dengan antarmukanya (Lollipop atau Socket). Garis lurus ini menunjukkan kepemilikan antarmuka tersebut oleh komponen yang bersangkutan.

5. Deployment Diagram

Deployment Diagram adalah jenis diagram UML yang menggambarkan arsitektur fisik di dimana komponen-komponen

perangkat lunak diimplementasikan di dalam perangkat keras (Munawar, 2021).

Tabel 2.5. *Deployment Diagram*

Simbol	Nama	Keterangan
	<i>Node</i>	Elemen fisik atau logis pada saat sistem dijalankan. Mewakili sumber daya komputasi seperti perangkat keras atau lingkungan eksekusi perangkat lunak.
	<i>Device</i>	Jenis <i>node</i> khusus yang mewakili perangkat keras fisik nyata yang mampu melakukan komputasi, seperti <i>Smartphone</i> , Komputer PC, Laptop, atau Mesin Pabrik.
	<i>Execution Environment</i>	Jenis <i>node</i> khusus yang mewakili lingkungan perangkat lunak tempat aplikasi berjalan, seperti Sistem Operasi, <i>Web Server</i> , atau <i>Database Management System</i> .
	<i>Artifact</i>	Mewakili entitas atau produk nyata dari proses pengembangan perangkat lunak yang ditanamkan (<i>deploy</i>) ke dalam <i>node</i> , seperti <i>file .apk</i> , modul <i>backend</i> web, atau kumpulan berkas kode.
	<i>Communication Path</i>	Jalur komunikasi fisik yang menghubungkan antar <i>node</i> . Biasanya diberi label berupa nama protokol jaringan yang digunakan untuk pertukaran data, seperti HTTPS, TCP/IP, atau REST API.

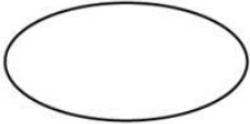
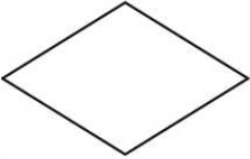
2.1.9. Entity Relationship Diagram (ERD)

Keberhasilan suatu sistem informasi sangat bergantung pada keandalan struktur basis data di baliknya (Pradipta et al., 2022). Oleh karena itu, penggunaan

Entity Relationship Diagram (ERD) menuntut tahapan perancangan yang matang sejak awal (Pradipta et al., 2022). Hal ini bertujuan untuk menghasilkan arsitektur data yang tidak hanya sejalan dengan spesifikasi kebutuhan pengguna, tetapi juga mampu mengelola sirkulasi informasi secara optimal saat sistem benar-benar diimplementasikan (Pradipta et al., 2022).

Untuk menciptakan fondasi data yang sistematis, proses perancangan ERD umumnya dibagi ke dalam tahapan pemodelan berikut (Pradipta et al., 2022):

Tabel 2.6. Simbol *Entity Relationship Diagram* (ERD)

Simbol	Nama	Keterangan
	Entitas	Merupakan objek di dunia nyata yang dapat dibedakan dari objek lain, yang datanya akan disimpan dalam basis data.
	Atribut	Merupakan karakteristik atau properti yang mendeskripsikan sebuah entitas.
	Relasi	Menunjukkan hubungan yang terjadi antara satu atau lebih entitas.
	Link	Berfungsi untuk menghubungkan antara atribut dengan entitas, maupun entitas dengan relasinya.

2.1.10. Black Box Testing

Black Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada evaluasi spesifikasi fungsional dari sebuah sistem, di mana penguji tidak perlu mengetahui struktur internal, logika, atau *source code* dari program yang

diuji (Agung, 2024). Pengujian ini dilakukan sepenuhnya dari sudut pandang pengguna akhir (*end-user*) dengan cara mencoba interaksi langsung pada antarmuka aplikasi, memasukkan data *input*, dan memverifikasi apakah sistem memberikan *output* yang sesuai dengan spesifikasi yang diharapkan (Agung, 2024). Mengingat sistem informasi manajemen stok bahan baku ini telah diimplementasikan dan di-*hosting* agar dapat diakses secara daring, pengujian *Black Box* dikonfigurasi dengan karakteristik dan tujuan yang spesifik terhadap arsitektur aplikasi.

Karakteristik Utama:

- a. Fokus pada Fungsionalitas Antarmuka, dimana pengujian murni berpusat pada interaksi pengguna dengan elemen antarmuka (tombol, formulir, pemindai visual) pada aplikasi *mobile* di sisi lapangan dan *dashboard web-admin* di sisi manajemen.
- b. Pengujian pada lingkungan nyata (*Hosted Environment*), aplikasi telah berjalan di server *cloud*, pengujian ini sekaligus mengevaluasi stabilitas dan respons komunikasi data melalui protokol RESTful API antara aplikasi *client* dan basis data MySQL dalam kondisi jaringan yang sebenarnya.
- c. Validasi *input* dan *output* dimana pengujian pada konsistensi alur kerja. Sebagai contoh, menguji apakah pembaruan data pemakaian bahan baku yang dikirim (*input*) oleh staf dari perangkat selulernya secara instan dan akurat mengubah grafik ketersediaan di layar manajemen (*output*) (Agung, 2024).
- d. Pengujian dapat dilakukan langsung oleh karyawan operasional dan manajemen sebagai *tester* (melalui skenario *User Acceptance Test*), tanpa mereka perlu memahami struktur pemrograman *framework* Flutter maupun Laravel 12 di baliknya.

Tujuan Pengujian pada Sistem Aplikasi

- a. Memastikan fungsionalitas utama, dimana fitur-fitur kritis seperti pembaruan stok digital, manajemen pengeluaran barang, identifikasi

visual produk, dan pelaporan otomatis berfungsi sempurna tanpa kegagalan sistem (*bug*).

- b. Menguji keamanan hak akses dimana validitas proses autentikasi (*login*) dan pembatasan hak akses (*role-based access*), untuk memastikan keamanan data yang tersimpan di *server hosting*.
- c. Meninjau performa transaksi data, untuk memastikan bahwa tidak ada hambatan (*delay*) yang berarti ketika aplikasi *mobile* mengirimkan permintaan (*request*) pembaruan inventaris ke *backend server*, sehingga prinsip pelaporan secara *real-time* dapat diandalkan.

2.1.11. User Acceptance Testing (UAT)

Pengujian Penerimaan Pengguna atau *User Acceptance Testing* (UAT) merupakan tahapan krusial dalam proses pengujian perangkat lunak yang bertujuan untuk memastikan bahwa sistem yang dikembangkan telah benar-benar memenuhi kebutuhan dan harapan pengguna sebelum dirilis secara resmi (Aliyah et. al., 2025). Evaluasi kelayakan sistem ini dilakukan secara langsung oleh pengguna akhir (*end-user*) yang berinteraksi dengan aplikasi guna memvalidasi bahwa seluruh fitur telah beroperasi sesuai dengan persyaratan fungsional perusahaan (Aliyah et al., 2025).

Pada pelaksanaannya, fokus evaluasi UAT umumnya diklasifikasikan ke dalam beberapa variabel utama, yang meliputi fungsionalitas sistem, kinerja atau keandalan sistem, pengalaman dan desain antarmuka (*user interface*), serta efisiensi dan peningkatan produktivitas kerja pengguna (Aliyah et al., 2025).

Untuk mengukur tingkat keberhasilan sistem secara objektif, pengumpulan data dilakukan dengan pengujian skenario langsung oleh pihak operasian PT. Plastindo Pack Printing.

2.2. Penelitian Terdahulu

Penelitian terdahulu merupakan dasar acuan untuk memposisikan penelitian yang diusulkan dan menegaskan kebaruan yang ditawarkan. Berikut adalah tinjauan terhadap beberapa hasil penelitian yang relevan:

Dalam ranah manajemen persediaan dan pergudangan, Rochman dan Waluyowati (2024) menganalisis pengendalian bahan baku menggunakan metode *Economic Order Quantity* (EOQ). Hasilnya menunjukkan bahwa penerapan metode presisi mampu menekan total biaya persediaan hingga 78% serta mencegah terjadinya kekosongan stok. Relevan dengan hal tersebut, Satryawati et al. (2023) mengembangkan *Warehouse Management System* berbasis Laravel dan MySQL yang terbukti meningkatkan efisiensi pemantauan stok, mutasi masuk, dan mutasi keluar secara *real-time* tanpa adanya kendala eror pada pengujian *black box*.

Terkait pemanfaatan teknologi, Iqbal et al. (2019) membuktikan efektivitas digitalisasi sistem informasi inventaris berbasis Laravel pada instansi pemerintah dengan peningkatan kecepatan pencarian data sebesar 99,75% dibandingkan sistem manual. Lebih lanjut, untuk menjembatani komunikasi data, arsitektur REST API terbukti sangat krusial dalam memisahkan pengembangan *client-server* (Farhandika et al., 2024). Hal ini diperkuat oleh penelitian Fauzi dan Hardjianto (2022) yang berhasil mengintegrasikan aplikasi *mobile* berbasis Flutter dengan *backend* menggunakan layanan keamanan Laravel Sanctum. Integrasi ini menghasilkan eksekusi *request* data operasional yang sangat cepat, aman, dan mudah diakses melalui platform *mobile*.

Di sisi metodologi pengembangan, Nova et al. (2022) melakukan tinjauan literatur sistematis (*Systematic Literature Review*) terhadap pengembangan sistem informasi berbasis website. Hasilnya menunjukkan bahwa model Scrum merupakan model dari metode Agile yang paling sering digunakan karena fleksibilitasnya dalam menangani perubahan kebutuhan proyek.

Penelitian yang dilakukan oleh Pirmansyah et al. (2025) menerapkan metode *Waterfall* dalam pengembangan *website* bagi UMKM Pengrajin Pandan Lanji Bersatu. Penelitian tersebut menunjukkan bahwa pendekatan sistematis *Waterfall* efektif untuk membangun sistem yang stabil dengan kebutuhan yang sudah terdefinisi sejak awal. Namun, berbeda dengan penelitian tersebut, penelitian ini memilih Metode *Prototype* untuk memberikan ruang bagi pengguna di PT. Plastindo Pack Printing dalam memberikan umpan balik (*feedback*) secara berulang, mengingat adanya hambatan literasi digital pada staf operasional.

Perbedaan penelitian ini dengan penelitian sebelumnya terletak pada integrasi platform dan target pengguna. Penelitian ini menggunakan pendekatan kombinasi antara aplikasi mobile berbasis Flutter untuk sisi lapangan dan web-admin berbasis Laravel 12 untuk manajemen. Selain itu, keunikan penelitian ini terletak pada penggunaan metode Prototype yang dirancang khusus untuk mengatasi hambatan literasi digital pada karyawan operasional di PT. sehingga pengguna dapat memberikan umpan balik langsung terhadap *interface* sistem sebelum aplikasi benar-benar diimplementasikan secara penuh.

2.3. Kerangka Berpikir

Kerangka berpikir ini disusun untuk menjelaskan alur logika dari identifikasi permasalahan hingga solusi yang ditawarkan dalam pengembangan sistem *warehouses management* berbasis mobile, Alur ini menggambarkan hubungan antara kondisi di lapangan, dan mengimplementasikan teknologi yang mudah diterapkan untuk mengatasi masalah tersebut.

a. kondisi aktual di lapangan

1. Pencatatan sirkulasi bahan baku dan stok barang di PT. Plastindo Pack Printing saat ini masih dilakukan secara manual dan konvensional.
2. Tingkat literasi digital karyawan operasional di rantai produksi masih terbatas, sehingga penggunaan sistem pencatatan yang rumit cenderung dihindari
3. Upaya penerapan sistem inventaris berbasis *website* sebelumnya kurang berhasil karena antarmuka yang kurang intuitif dan minimnya fleksibilitas akses (mobilitas) bagi pekerja gudang.
4. Sistem pelaporan sisa stok belum terintegrasi secara *real-time* ke pihak manajemen.

b. Faktor Penyebab

1. Proses pengelolaan dan pencatatan data yang masih mengandalkan buku fisik atau dokumen kertas..

2. Kurangnya pelibatan pengguna akhir (karyawan gudang) dalam merumuskan kebutuhan antarmuka pada pengembangan sistem sebelumnya.
3. Keterbatasan sarana teknologi yang dapat diakses secara instan dari genggaman (perangkat *mobile*) saat karyawan sedang melakukan aktivitas fisik di area gudang.

c. Dampak

1. Rentannya terjadi kehilangan data, kesalahan perhitungan, dan ketidaksesuaian antara pencatatan fisik dengan stok riil di lapangan.
2. Membutuhkan waktu yang lama untuk mencari informasi detail mengenai riwayat keluar-masuknya bahan baku tertentu.
3. Pihak manajemen kesulitan memantau ketersediaan bahan baku secara akurat, yang berisiko menyebabkan terhentinya proses produksi secara tiba-tiba akibat kekosongan persediaan (*stockout*).

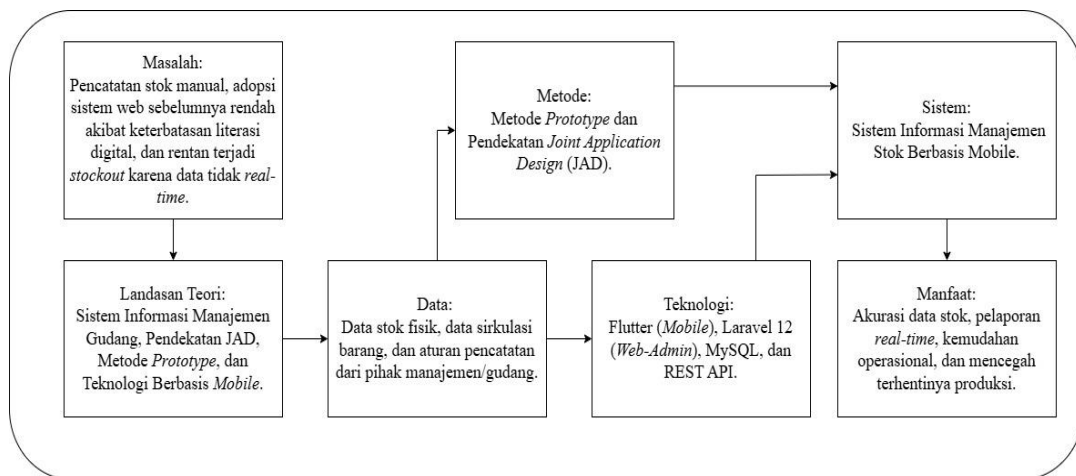
d. Solusi yang ditawarkan

1. Mengembangkan sistem informasi manajemen stok dengan arsitektur *hybrid*, yakni aplikasi *mobile* berbasis Flutter untuk kelancaran operasional di lapangan, dan *web-admin* berbasis Laravel 12 untuk pemantauan manajemen, yang dihubungkan melalui arsitektur REST API.
2. Menerapkan metode *Prototype* agar pengembangan sistem berjalan secara iteratif, memungkinkan visualisasi antarmuka aplikasi sebelum tahap finalisasi sistem.
3. Menggunakan pendekatan *Joint Application Design* (JAD) pada tahap analisis kebutuhan guna memastikan spesifikasi dan fitur aplikasi benar-benar kolaboratif dan disesuaikan dengan kemampuan literasi digital karyawan gudang.

e. Tujuan Akhir

1. Meningkatkan efektivitas, efisiensi, dan akurasi dalam pencatatan stok bahan baku di lingkungan pabrik.

2. Menyediakan antarmuka aplikasi *mobile* yang ramah pengguna (*user-friendly*) sehingga dapat dengan mudah diadopsi oleh karyawan operasional.
3. Mendukung pihak manajemen dengan sistem pelaporan terpusat yang *real-time* untuk mempermudah pengambilan keputusan terkait pengadaan bahan baku.



Gambar 2.7. Flow Chart Kerangka Berpikir