BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Sistem Informasi

Berbagai unsur yang meliputi data, dokumen, proses, mesin, dan individu saling berhubungan untuk menjalankan fungsi tertentu secara terpadu. Kesatuan dari unsur-unsur tersebut membentuk sistem yang diarahkan untuk mencapai tujuan yang sama (Bili dkk., 2025:234). Makna suatu pesan dapat disampaikan melalui konsep, fakta, data maupun simbol yang telah diolah menjadi informasi. Informasi dapat disajikan dalam berbagai bentuk dan pemanfaatannya semakin berkembang sejalan dengan kemajuan teknologi informasi serta komunikasi elektronik (Bili dkk., 2025:234).

Pengelolaan informasi dalam organisasi dilakukan melalui perpaduan sumber daya manusia, prosedur kerja, *hardware,* dan *software* yang saling terintegrasi. Keseluruhan unsur tersebut membentuk sistem informasi yang berfungsi mengolah, menyimpan, dan menyebarkan informasi guna menunjang kegiatan organisasi (Kholid dkk., 2025:100). Sistem informasi juga dapat dipahami sebagai kesatuan antara data, teknologi informasi, prosedur, dan sumber daya manusia yang dirancang secara terstruktur untuk mendukung pencapaian tujuan organisasi (Maryana & Yulina, 2025:224). Selain mendukung pemrosesan transaksi harian dan kegiatan operasional, sistem informasi berperan dalam

menyediakan laporan bagi pihak yang membutuhkan serta membantu manajemen mengambil keputusan secara lebih efektif (Ramadhan dkk., 2025:129).

Berbagai komponen, seperti sumber daya manusia, prosedur kerja, *hardware,* dan *software*, saling terintegrasi dalam membentuk sistem informasi. Kesatuan tersebut berfungsi mengolah data menjadi informasi yang bermanfaat untuk mendukung kegiatan operasional, manajerial, dan strategis organisasi. Penerapan sistem informasi juga membantu organisasi mengelola data secara lebih efektif, memperlancar komunikasi antarbagian, mempercepat penyediaan informasi, serta meningkatkan kualitas pengambilan keputusan guna mencapai tujuan yang telah ditetapkan.

2. *Inventory*

Barang yang disimpan untuk memenuhi kebutuhan produksi maupun dijual pada waktu yang akan datang disebut sebagai persediaan atau *inventory.* Keberadaan persediaan menjadi bagian dari aset perusahaan yang menunjang kelancaran kegiatan operasional sesuai dengan kebutuhan dan permintaan pasar (Putri & Tristianto, 2024:97; Prasetyo, 2025:17). Oleh karena itu, perusahaan perlu menerapkan pengelolaan persediaan secara terkomputerisasi agar pengendalian stok dapat berlangsung lebih efektif dan efisien (Prasetyo, 2025:17). Pengelolaan persediaan tersebut dapat didukung oleh sistem informasi *inventory* yang berfungsi mencatat barang keluar dan masuk, serta

menyajikan laporan stok secara terstruktur. Penerapan sistem ini membantu mengurangi kesalahan pencatatan, mempercepat penyusunan laporan, dan memudahkan pemantauan ketersediaan barang secara *real-time* (Ilhamsyah dkk., 2025:2).

Maka dikatakan persediaan atau *inventory* mencakup barang yang disimpan perusahaan untuk menunjang kegiatan operasional, proses produksi, maupun penjualan di masa mendatang. Agar pengelolaannya berjalan secara efektif dan efisien, perusahaan memerlukan sistem informasi *inventory* yang mampu mengatur data persediaan secara sistematis, mulai dari pencatatan barang masuk dan barang keluar hingga penyusunan laporan stok. Penerapan sistem tersebut membantu mempercepat pengelolaan persediaan, meningkatkan ketepatan pencatatan, menunjang dalam mengambil kebijakan dalam perusahaan serta memudahkan pemantauan stok.

3. PHP *Hypertext Preprocessor*

Kemampuan mengolah data di sisi *server* menjadikan PHP (*Hypertext Preprocessor*) sebagai salah satu bahasa pemrograman yang banyak dimanfaatkan dalam pengembangan aplikasi berbasis *web*. Bahasa pemrograman ini dikenal memiliki sintaks yang sederhana, fleksibel, serta mudah dipahami karena karakteristik penulisannya memiliki kemiripan dengan bahasa C, Java, dan Perl, meskipun PHP menyediakan berbagai fungsi yang lebih spesifik (Indriyani dkk., 2024:70). Sejak diperkenalkan oleh Rasmus Lerdorf pada tahun 1995, PHP terus mengalami

perkembangan dan menjadi salah satu bahasa yang populer dalam pengembangan *web* (Mahendra, 2023:65). Selain itu, PHP dapat diintegrasikan dengan berbagai sistem manajemen basis data, termasuk MySQL, sehingga mampu mengelola data secara dinamis dan mendukung pembangunan *website* maupun sistem informasi yang interaktif, efisien, ringan, serta mudah dikembangkan sebagaimana keperluan pengguna (Ismail dkk., 2025:1470).

4. *JavaScript*

Dalam pengembangan *website*, *JavaScript* digunakan sebagai bahasa skrip yang dijalankan langsung oleh *web browser* pada sisi klien (*client-side*). Seluruh instruksi yang ditulis dalam *JavaScript* diproses oleh perangkat pengguna tanpa memerlukan pemrosesan dari *server* sehingga perubahan pada halaman *web* dapat ditampilkan secara langsung sesuai dengan interaksi yang dilakukan pengguna (Saribu dkk., 2025:151). *JavaScript* diterapkan bersama dokumen HTML untuk mengatur perilaku dan tampilan halaman *web* melalui kumpulan skrip yang dieksekusi oleh *browser* (Rizal & Bakhri, 2025:3). Kemampuan tersebut memungkinkan berbagai elemen pada halaman *web* merespons tindakan pengguna secara dinamis dan *real-time*. Oleh sebab itu, *JavaScript* menjadi salah satu teknologi utama dalam pengembangan aplikasi *web* karena mampu menciptakan tampilan yang lebih interaktif, responsif, serta memberikan pengalaman penggunaan yang lebih baik.

5. *Bootstrap*

Perancangan antarmuka *website* dapat dilakukan dengan lebih mudah melalui *Bootstrap,* yaitu *framework* berbasis CSS (*Cascading Style Sheet*) yang menyediakan berbagai komponen siap pakai. *Framework* ini menawarkan beragam elemen, seperti tombol, navigasi, tipografi, serta komponen visual lainnya sehingga proses pembuatan tampilan *web* menjadi lebih cepat, rapi, dan efisien. Selain memanfaatkan CSS, *Bootstrap* juga dilengkapi dengan komponen *JavaScript* yang mendukung terciptanya antarmuka yang lebih interaktif dan responsif. Dengan berbagai fitur tersebut, *Bootstrap* banyak digunakan untuk mengembangkan *website* maupun aplikasi *web* yang mampu menyesuaikan tampilannya pada berbagai ukuran perangkat tanpa memerlukan proses perancangan dari awal (Rizal & Bakhri, 2025:4).

Pengembangan antarmuka *website* dapat dilakukan dengan lebih cepat dan efisien menggunakan *Bootstrap,* yaitu *framework front-end* yang menggabungkan HTML, CSS, dan *JavaScript*. *Framework* ini menyediakan berbagai komponen siap pakai, seperti *grid system*, tipografi, formulir, tombol, navigasi, serta elemen antarmuka lainnya sehingga proses perancangan halaman *web* menjadi lebih mudah (Dirgantara & Andrian, 2023:434). Selain menyediakan beragam komponen, *Bootstrap* dirancang dengan pendekatan *responsive web design* yang mengutamakan kompatibilitas pada perangkat bergerak, seperti *smartphone,* tanpa mengabaikan tampilan pada komputer maupun *tablet* (Gunawan dkk.,

2021:297). Oleh karena itu, *Bootstrap* banyak dimanfaatkan oleh pengembang *web* untuk menghasilkan tampilan *website* yang responsif, menarik, dan mudah dikembangkan sesuai kebutuhan.

6. *Laravel*

Sejak diperkenalkan oleh Taylor Otwell pada tahun 2011, *Laravel* berkembang menjadi salah satu *framework* PHP yang banyak digunakan dalam pengembangan aplikasi *web*. *Framework* ini tersedia secara gratis dan semakin diminati oleh para pengembang karena menawarkan sintaks yang sederhana serta berbagai fitur pendukung, seperti *routing*, *middleware,* dan *Object Relational Mapping* (ORM) *Eloquent,* yang membantu menyederhanakan proses pengembangan aplikasi (Abdulloh, 2022:10; Hilmi dkk., 2025:12). Selain mudah digunakan, *Laravel* juga dikenal memiliki tingkat keamanan yang baik serta mampu mendukung pengembangan aplikasi berskala kecil maupun besar sehingga banyak diterapkan pada pengembangan aplikasi *web* modern (Ilhamsyah dkk., 2025:2).

Dalam proses pengembangan antarmuka, *Laravel* dapat dipadukan dengan *Bootstrap,* yaitu *library framework* berbasis HTML, CSS, dan *JavaScript* yang digunakan untuk membangun tampilan website. *Bootstrap* menyediakan berbagai komponen antarmuka sehingga memudahkan pembuatan halaman *web* yang responsif dan menjadi salah satu *framework* yang banyak digunakan oleh pengembang *web* (Fauziah dkk., 2025:33).

Pengembangan aplikasi *web* dapat dilakukan secara lebih terstruktur dengan memanfaatkan *Laravel,* yaitu *framework* PHP yang menyediakan sintaks sederhana dan mudah dipahami oleh pengembang. Berbagai fitur, seperti *routing*, *middleware*, autentikasi, dan *Object Relational Mapping* (ORM) *Eloquent*, turut mendukung proses pembangunan aplikasi agar lebih efisien, aman, dan mudah dikelola. Sebagai *framework* yang bersifat *open source*, Laravel dapat digunakan tanpa biaya serta mampu mengakomodasi pengembangan aplikasi dengan kebutuhan yang beragam, mulai dari skala kecil hingga skala besar. Berkat fleksibilitas, keamanan, dan kemudahan yang ditawarkannya, Laravel menjadi salah satu *framework* PHP yang banyak dimanfaatkan dalam pengembangan sistem informasi berbasis *web* modern.

7. Sistem Basis Data

Basis data (*database*) merupakan kumpulan data yang disusun berdasarkan struktur atau skema tertentu sehingga setiap data memiliki hubungan yang terorganisasi dan dapat dikelola oleh sistem komputer (Haviz dkk., 2024:165). Pengaturan tersebut memungkinkan data disimpan dalam bentuk tabel-tabel yang tersusun secara sistematis sehingga proses penyimpanan, pencarian, maupun penggunaan kembali dapat dilakukan dengan lebih mudah dan cepat (Siswanto dkk., 2025:239). Selain berfungsi sebagai media penyimpanan, database juga mendukung pengolahan data menjadi informasi yang diperlukan oleh pengguna. Oleh karena itu, keberadaan basis data membantu organisasi mengelola data

secara lebih teratur, meningkatkan ketepatan informasi yang dihasilkan, serta mendukung pelaksanaan kegiatan operasional dan pengambilan keputusan secara lebih efektif (Gunawan dkk., 2021:297).

8. *PostgreSQL*

Postgres sebutan lain dari *PostgreSQL* yang artinya sistem manajemen basis data relasional (*Relational Database Management System*/RDBMS) yang bersifat *open source* dan banyak dimanfaatkan dalam pengembangan aplikasi *web*, aplikasi seluler, maupun kebutuhan analisis data (Salahuddin dkk., 2024:3). Pengembangan *PostgreSQL* dimulai pada tahun 1986 melalui proyek *POSTGRES* di Jurusan Ilmu Komputer, Universitas California, Berkeley. Selanjutnya, pada tahun 1996 nama proyek tersebut diubah menjadi *PostgreSQL* untuk menunjukkan dukungannya terhadap standar SQL (Salahuddin dkk., 2024:3). Selain mendukung kueri SQL untuk basis data relasional, *PostgreSQL* juga mampu mengelola data berformat JSON sehingga dapat digunakan pada kebutuhan nonrelasional. Pengembangannya yang telah berlangsung selama lebih dari dua dekade menjadikan *PostgreSQL* dikenal memiliki tingkat kestabilan, keandalan, serta konsistensi yang tinggi dalam menangani transaksi yang kompleks dan data dalam jumlah besar (Salahuddin dkk., 2024:3; Pradana & Chotijah, 2025:3157).

Dengan demikian bisa disimpulkan terkait kemampuan mengelola data dalam jumlah besar dan menangani transaksi yang kompleks menjadikan *PostgreSQL* sebagai salah satu sistem manajemen basis data

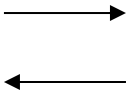
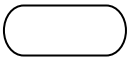
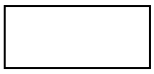
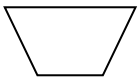
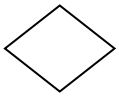
relasional (*Relational Database Management System*/*RDBMS*) yang banyak digunakan dalam pengembangan sistem informasi. Basis data yang bersifat *open source* ini mendukung penerapan standar SQL serta mampu mengelola data relasional maupun nonrelasional secara andal dengan kinerja yang tetap stabil. Selain menawarkan tingkat keamanan dan keandalan yang tinggi, *PostgreSQL* juga memiliki skalabilitas yang baik sehingga dapat memenuhi berbagai kebutuhan pengelolaan data pada aplikasi modern secara efektif dan efisien.

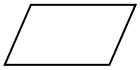
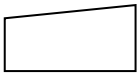
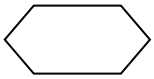
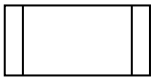
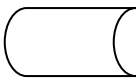
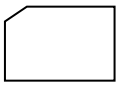
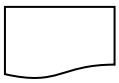
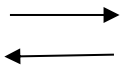
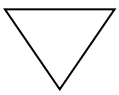
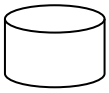
9. *Flowchart*

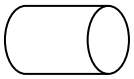
Flowchart atau bagan alir menyajikan tahapan langkah dan alur logika dalam bentuk diagram yang menggunakan simbol-simbol grafis sehingga setiap tahapan dapat dipahami secara sistematis (Listyoningrum dkk., 2023:103). Melalui penyajian tersebut, aliran data maupun proses dalam suatu program dapat digambarkan secara runtut sejak tahap awal hingga tahap akhir pemrosesan (Putra dkk., 2021:2611). Kejelasan alur yang ditampilkan menjadikan *flowchart* sebagai alat bantu yang banyak dimanfaatkan pada tahap menganalisis, merancang, mengkodekan, dan menyelesaikan permasalahan dengan lebih rinci pada rangkaian aktivitas operasional. Selain memudahkan pengembang dalam memahami mekanisme kerja suatu program, *flowchart* juga membantu proses identifikasi permasalahan dan evaluasi sehingga penyusunan solusi dapat dilakukan secara lebih terarah. Dengan demikian, penggunaan *flowchart* mampu mendukung pengembangan sistem yang lebih terstruktur,

mempermudah komunikasi mengenai alur kerja sistem, serta memberikan gambaran proses yang jelas bagi pengembang maupun pengguna. Simbol-simbol *Flowchart* dapat dilihat pada Tabel 2.1.

Tabel 2.1. Simbol-simbol *Flowchart*

Simbol	**Nama**	**Keterangan**
	Arus	Perpindahan alur dari satu tahapan ke tahapan berikutnya pada *flowchart* ditunjukkan menggunakan simbol arus atau *connecting line*. Simbol ini berfungsi menghubungkan setiap simbol yang terdapat dalam bagan sehingga urutan dan arah proses dapat digambarkan secara jelas serta sistematis.
	Terminal	Sebagai simbol *stop* dan *start* dalam suatu sistem.
	penghubung	Penghubung bagian-bagian *flowchart* yang terpisah sehingga alur proses tetap berkesinambungan tanpa perlu membuat garis penghubung yang panjang.
	baris penghubung	menghubungkan alur proses yang berpindah ke lembar atau halaman yang berbeda.
	Proses	Suatu proses pengolahan data dilakukan oleh komputer atau *personal computer* (PC).
	kegiatan manual	Menggambarkan proses yang dikerjakan secara manual tanpa melibatkan komputer.
	keputusan	Titik pengambilan keputusan untuk menentukan alur proses berdasarkan kondisi atau kriteria tertentu.

	keluar-masuk	Kegiatan masukan (*input*) maupun keluaran (*output*) tanpa bergantung pada jenis perangkat yang digunakan.
	Manual *Input*	Menggambarkan proses pemasukan data secara manual melalui papan ketik (*keyboard*) yang terhubung secara *online*.
	persiapan	Proses persiapan media penyimpanan (*storage*) yang akan digunakan dalam pengolahan data.
	proses terdifinisi	Pelaksanaan suatu subprogram atau prosedur yang menjadi bagian dari proses utama.
	penyimpanan *online*	Menunjukkan bahwa data diterima dari media penyimpanan *disk* atau disimpan ke dalam *disk*.
	unit pita magnetik	Data berasal dari pita magnetik atau hasil keluaran disimpan pada pita magnetik.
	kartu plong	Menunjukkan bahwa data masukan berasal dari kartu atau hasil keluaran ditulis ke media kartu.
	dokumen	Data diterima dari dokumen berbentuk kertas atau hasil keluaran dicetak ke media kertas.
	Garis Alir	Sebagai penanda arah alur menuju langkah atau instruksi berikutnya dalam suatu algoritma.
	penyimpanan *offline*	Menyatakan bahwa data akan disimpan pada media penyimpanan.
	magnetik *Disk*	Proses *input* atau *output* yang menggunakan *disk magnetik*.

	magnetik drum	Menggambarkan proses *input* atau *output* yang memanfaatkan *drum magnetik*.

Sumber: Simanungkalit dan Lubis (2023:18)

Keterangan

Flowchart disusun menggunakan berbagai simbol standar yang masing-masing memiliki fungsi dan makna tertentu dalam menggambarkan alur suatu proses atau sistem. Setiap simbol digunakan untuk merepresentasikan aktivitas, keputusan, aliran proses, masukan (*input*), keluaran (*output*), maupun penyimpanan data sehingga alur kerja sistem dapat dipahami dengan lebih jelas dan sistematis. Adapun simbol-simbol *flowchart* yang digunakan dalam penelitian ini beserta nama dan keterangannya disajikan pada Tabel 2.1. Simbol-simbol *Flowchart*.

10. *Unified Modeling Language* (UML)

UML atau *Unified Modeling Language* adalah bahasa pemodelan standar yang digunakan untuk menggambarkan, merancang, dan mendokumentasikan berbagai komponen penting dalam sistem perangkat lunak. Komponen ini dapat berupa model, struktur sistem, maupun bagian lain dari perangkat lunak. UML juga digunakan dalam pemodelan proses bisnis serta sistem *non-software*. Bahasa ini dikembangkan oleh James Rumbaugh, Ivar Jacobson, dan Grady Booch melalui *Rational Software Corporation* dan berlandaskan pendekatan berorientasi objek. UML menyediakan berbagai notasi visual yang memungkinkan perancang

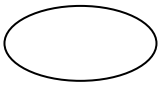
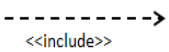
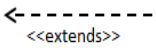
sistem melihat dan memahami sistem dari berbagai perspektif. Penggunaannya pun tidak terbatas pada rekayasa perangkat lunak saja, tetapi meluas ke berbagai bidang yang memerlukan pemodelan (Irfan, dkk, 2023:84).

UML, atau *Unified Modeling Language*, merupakan bahasa pemodelan yang digunakan secara global untuk membuat rancangan sistem perangkat lunak. Dengan UML, pengembang dapat menggambarkan, merancang, membangun, dan mencatat berbagai elemen sistem secara terstruktur. Sama seperti arsitek merancang bangunan sebelum konstruksi, pengembang sistem menggunakan diagram UML untuk merancang struktur dan alur kerja dalam proses pengembangan *software*. Pemahaman terhadap istilah dan simbol dalam UML akan memudahkan pengembang dalam memahami spesifikasi sistem secara menyeluruh (Irfan, dkk, 2023:84).

a. *Use case diagram*

   Use case diagram adalah bentuk pemodelan yang bertujuan untuk menunjukkan hubungan atau interaksi antara aktor (pengguna) dengan sistem yang sedang dirancang. Diagram ini digunakan untuk mengenali peran-peran penting dalam sistem serta menentukan pihak-pihak yang terlibat dalam proses tersebut memiliki otoritas untuk menggunakan fungsi-fungsi tertentu dalam sistem tersebut (Irfan, dkk, 2023:84). Simbol-simbol *Use Case* dapat dilihat pada Tabel 2.2.

Tabel 2.2. Tabel *Use Case*

Gambar	Nama	Keterangan
	*Use Case*	Fungsi-fungsi sistem tergambar ketika terjadi pertukaran pesan antar komponen, baik antar unit maupun antara unit dengan *actor*.
	Aktor	Entitas yang terlibat dalam interaksi dengan sistem informasi, seperti orang, proses, atau sistem eksternal, berada di luar cakupan sistem tersebut. Karena itu, aktor tidak selalu merujuk pada manusia meskipun sering digambarkan dengan simbol orang; umumnya, penamaan aktor diawali dengan kata benda untuk merepresentasikan fungsinya
	Asosiasi	Aktor dapat berinteraksi dengan *use case*, begitu juga *use case* dapat berinteraksi dengan aktor yang terlibat.
	*Include*	Relasi *include* menunjukkan keterkaitan antara *use case* utama dengan fungsi tambahan yang hanya bisa dijalankan jika dipanggil oleh *use case* lainnya sebagai bagian dari proses. Terdapat dua interpretasi umum dalam konsep *include* pada *use case*: 1. *Use case* tambahan secara otomatis dipanggil setiap kali *use case* utama dijalankan. 2. Sistem akan terlebih dahulu memverifikasi apakah *use case* yang disertakan telah dieksekusi sebelum melanjutkan ke fungsi tambahan tersebut.
	*Extend*	Interaksi antar *use case* dapat mencerminkan fungsi yang diperluas, di mana *use case* pelengkap tetap memiliki kemampuan untuk beroperasi secara mandiri tanpa kehadiran *use case* utama-konsep ini menyerupai pewarisan (*inheritance*) dalam paradigma pemrograman berbasis objek. Biasanya, nama *use case* tetap konsisten, dengan panah menunjukkan arah perluasan; *use*

		case yang diperluas biasanya masih dalam kategori fungsi yang sama dengan induknya.
⟶	*Generalization*	Relasi antara generalisasi dan spesialisasi merepresentasikan struktur hierarkis yang menghubungkan dua *use case* dalam sistem, di mana salah satunya merupakan bentuk umum dari yang lain. Panah biasanya diarahkan ke *use case* yang bersifat generik untuk menunjukkan struktur pewarisan fungsi.

Sumber: Irfan, dkk, (2023:84)

Keterangan:

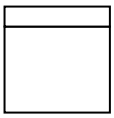
Tabel simbol *Use Case Diagram* digunakan untuk menjelaskan makna setiap simbol yang digunakan dalam pemodelan kebutuhan fungsional sistem menggunakan *Unified Modeling Language* (UML). *Use Case Diagram* menggambarkan interaksi antara aktor dengan sistem serta fungsi-fungsi yang dapat dijalankan sesuai dengan hak akses dan peran masing-masing aktor. Selain itu, diagram ini juga menunjukkan hubungan antar *use case*, seperti hubungan *association*, *include*, *extend*, dan *generalization*, yang menggambarkan keterkaitan antar fungsi dalam sistem. Dengan adanya *Use Case Diagram*, kebutuhan fungsional Sistem Informasi *Inventory* dapat divisualisasikan secara jelas sehingga memudahkan proses analisis, perancangan, implementasi, dan pengembangan sistem

b. *Activity diagram*

Activity diagram merupakan jenis diagram yang digunakan untuk merepresentasikan alur aktivitas dan proses kerja dalam suatu sistem,

baik dalam konteks bisnis maupun perangkat lunak. Fokus utama dari diagram ini adalah pada aktivitas-aktivitas yang dilakukan oleh sistem, bukan pada tindakan pengguna (aktor) (Irfan, dkk, 2023:85). Simbol-simbol *Actvity Diagram* dapat dilihat pada Tabel 2.3.

Tabel 2.3. Simbol *Actvity Diagram*

Gambar	Nama	Keterangan
	Status awal	Diagram aktivitas memiliki titik awal yang menunjukkan kondisi awal dari proses sistem yang akan dijalankan.
	Aktivitas	Aktivitas yang dilakukan oleh sistem biasanya diawali dengan kata kerja yang menunjukkan tindakan tertentu.
	Percabangan	Cabang asosiasi digunakan ketika terdapat beberapa pilihan aktivitas yang dapat dijalankan.
	Penggabungan	Penggabungan asosiasi dilakukan untuk menyatukan beberapa aktivitas menjadi satu kesatuan proses.
	Status akhir	Diagram aktivitas juga mencerminkan titik akhir eksekusi sistem, yang menunjukkan bahwa seluruh aktivitas telah selesai dilakukan.
	Swimlane	Pembagian tanggung jawab antar unit dalam organisasi yang mengelola pelaksanaan setiap aktivitas dalam proses bisnis.

Sumber: Irfan, dkk, (2023:85)

Keterangan:

Tabel simbol *Activity Diagram* digunakan untuk menjelaskan arti dari setiap simbol yang digunakan dalam pemodelan alur aktivitas pada *Unified Modeling Language* (UML). *Activity Diagram* menggambarkan urutan aktivitas, pengambilan keputusan, serta aliran proses yang terjadi dalam suatu sistem

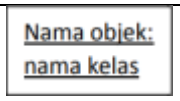
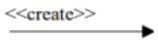
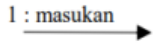
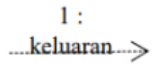
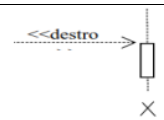
dari awal hingga akhir. Diagram ini memudahkan dalam memahami proses bisnis yang berlangsung, pembagian tanggung jawab setiap pihak yang terlibat, serta hubungan antar aktivitas dalam menjalankan fungsi-fungsi sistem. Dengan adanya *Activity Diagram*, alur kerja Sistem Informasi *Inventory* dapat divisualisasikan secara sistematis sehingga mempermudah proses analisis, perancangan, dan implementasi sistem

c. *Sequence diagram*

Sequence diagram adalah representasi visual yang menunjukkan bagaimana objek-objek dalam suatu *use case* berinteraksi satu sama lain melalui pesan yang dikirim dan diterima, serta bagaimana urutan aktivitas tersebut berlangsung sesuai dengan siklus hidup objek. Untuk menyusun diagram ini, penting untuk memahami objek-objek yang terlibat dalam skenario serta peran yang dimainkan masing-masing objek dalam kelas (Irfan, dkk, 2023:86). Simbol-simbol *Sequence diagram* dapat dilihat pada Tabel 2.4.

Tabel 2.4. Simbol *Sequence diagram*

Gambar	Nama	Keterangan
	Aktor	Entitas seperti orang, proses, atau sistem eksternal yang terlibat dalam interaksi dengan sistem informasi berada di luar lingkup sistem utama yang dibangun. Karena itu, aktor tidak selalu merepresentasikan manusia walaupun disimbolkan dengan ikon manusia. Nama aktor umumnya diawali dengan kata benda untuk menunjukkan peran atau fungsinya.

	*Lifeline*	Menggambarkan tahapan atau fase dari keberadaan suatu objek selama masa hidupnya.
	Objek	Menandai objek yang terlibat dalam pertukaran pesan selama proses interaksi.
	Waktu aktif	Menunjukkan bahwa objek dalam kondisi aktif dan sedang melakukan aktivitas tertentu, di mana setiap tindakan yang terjadi selama periode aktif ini direpresentasikan sebagai bagian dari proses.
	Pesan tipe *Create*	Mengilustrasikan bahwa suatu objek bertanggung jawab dalam pembuatan objek baru, dengan panah diarahkan ke objek hasil ciptaan. Panah juga menunjukkan objek yang menjalankan metode tertentu, yang harus sesuai dengan metode yang terdapat dalam diagram kelas dari objek yang dimaksud.
	Pesan tipe *Send*	Sebuah objek yang berfungsi untuk mengirimkan data, masukan, atau informasi kepada objek lain, ditunjukkan dengan panah yang menunjuk ke arah objek penerima informasi
	Pesan tipe *return*	Menggambarkan proses pengembalian dari satu objek ke objek lain setelah menyelesaikan eksekusi metode tertentu, di mana panah diarahkan ke penerima hasil pengembalian.
	Pesan tipe *destroy*()	Menunjukkan bahwa suatu objek mengambil tindakan untuk menghapus atau mengakhiri objek lain, ditandai dengan panah yang mengarah ke objek yang dihilangkan. Idealnya, setiap objek yang dibuat memiliki proses penghancuran (*destroy*) yang menyertainya.

Sumber: Irfan, dkk, (2023:87)

Keterangan:

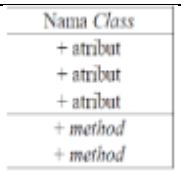
Tabel simbol *Sequence Diagram* digunakan untuk menjelaskan makna setiap simbol yang digunakan dalam pemodelan interaksi antar objek pada *Unified Modeling Language* (UML). *Sequence Diagram* menggambarkan urutan komunikasi atau pertukaran pesan antar aktor dan objek dalam menjalankan suatu proses sesuai dengan urutan waktu (*sequence*). Melalui diagram ini, alur eksekusi setiap fungsi dalam sistem dapat dipahami dengan lebih jelas, mulai dari aktor yang memulai proses, objek yang terlibat, hingga pesan yang dipertukarkan selama proses berlangsung. Dengan demikian, *Sequence Diagram* membantu menggambarkan perilaku dinamis sistem serta hubungan antar objek dalam menyelesaikan suatu fungsi pada Sistem Informasi *Inventory* yang dibangun.

d. *Class diagram*

Class diagram digunakan untuk menampilkan struktur internal dari sistem dengan menjelaskan kelas-kelas utama yang menyusun sistem tersebut. Setiap kelas mencakup atribut (properti) dan operasi (metode) yang dapat dijalankan dalam konteks sistem (Irfan, dkk, 2023:86). Diagram kelas digunakan untuk merepresentasikan deskripsi kelas, atribut, objek, serta keterkaitannya satu sama lain. Diagram ini menyajikan gambaran menyeluruh terhadap struktur sebuah sistem melalui penggambaran hubungan antar kelas. Umumnya, dalam satu sistem terdapat lebih dari satu *class diagram* yang menunjukkan

berbagai aspek struktur sistem. Diagram ini sangat berguna untuk memvisualisasikan struktur internal sistem berbasis objek. Selain itu, *class diagram* berperan dalam menjelaskan jenis-jenis objek yang digunakan dalam sistem beserta relasi yang terjadi antar objek tersebut (Suharni, dkk, 2023:3). Simbol-simbol *Class diagram* dapat dilihat pada Tabel 2.5.

Tabel 2.5. Simbol *Class diagram*

Gambar	Nama	Keterangan
	*Class*	Himpunan objek-objekdari berbagai atributyang me-miliki operasi yang sama.
	Association	Hubungan antara dua kelas yang bersifat umum, dan umumnya mencantumkan jumlah (*multiplicity*) objek yang terlibat dalam relasi tersebut.
	Directed Association	Hubungan antara kelas yang menunjukkan bahwa satu kelas memanfaatkan atau menggunakan fungsi dari kelas lainnya.
	Aggregation	Hubungan yang menggambarkan keterkaitan antara bagian dan keseluruhan suatu objek dikenal sebagai relasi bagian-keseluruhan (*whole-part relationship*).
	*Composition*	Relasi *composition* menunjukkan ketergantungan penuh suatu kelas terhadap kelas induknya, di mana bagian tidak dapat berdiri sendiri tanpa keseluruhannya.
	*Dependency*	Menggambarkan metode atau fungsi dalam sebuah kelas yang melibatkan pemanfaatan atau interaksi dengan kelas lainnya.

Sumber: Suharni, dkk, (2023:3)

Keterangan:

Tabel simbol *Class Diagram* digunakan untuk menjelaskan arti dari setiap simbol yang digunakan dalam pemodelan sistem berorientasi objek menggunakan *Unified Modeling Language* (UML). Simbol-simbol tersebut berfungsi untuk menggambarkan struktur sistem, hubungan antar kelas, serta interaksi yang terjadi di antara objek-objek dalam sistem. Dengan memahami fungsi masing-masing simbol, proses perancangan sistem menjadi lebih mudah dipahami dan dapat memberikan gambaran yang jelas mengenai struktur serta hubungan antar komponen pada Sistem Informasi *Inventory* yang dibangun

11. *BlackBox*

Pengujian *Blackbox* dilakukan dengan menitikberatkan pada fungsi yang dimiliki perangkat lunak tanpa melibatkan pemeriksaan terhadap struktur kode maupun logika program. Dalam pelaksanaannya, penguji memberikan sejumlah masukan (*input*) ke dalam sistem, kemudian membandingkan keluaran (*output*) yang dihasilkan dengan spesifikasi atau hasil yang telah ditentukan sebelumnya (Sunaryo & Untari, 2025:35). Melalui pendekatan tersebut, berbagai kesalahan pada fungsi sistem, seperti antarmuka, validasi data, maupun integrasi antarmodul, dapat diidentifikasi tanpa mengetahui proses internal yang terjadi pada program. Oleh karena itu, metode *Blackbox* banyak diterapkan pada tahap akhir pengembangan perangkat lunak untuk memastikan bahwa setiap fungsi

aplikasi telah berjalan sesuai dengan kebutuhan dan harapan pengguna (Tambay dkk., 2025:45).

Kesesuaian fungsi suatu perangkat lunak dengan kebutuhan pengguna dapat diketahui melalui *Black Box Testing*, yaitu metode pengujian yang menilai kinerja sistem berdasarkan respons yang diberikan terhadap berbagai masukan (*input*). Pada metode ini, penguji tidak meninjau kode program atau mekanisme internal sistem, tetapi memusatkan perhatian pada keluaran (*output*) yang dihasilkan untuk memastikan bahwa setiap fungsi telah bekerja sesuai spesifikasi. Pendekatan tersebut membantu menemukan kesalahan yang berkaitan dengan validasi data, antarmuka pengguna, fungsi sistem, maupun integrasi antarmodul. Dengan demikian, *Black Box Testing* menjadi salah satu metode yang banyak diterapkan untuk memverifikasi kesiapan sistem sebelum digunakan atau diimplementasikan oleh pengguna.

12. *System Usability Scale* (SUS)

System Usability Scale (SUS) pertama kali diperkenalkan oleh Brooke pada tahun 1986 sebagai alat evaluasi yang cepat dan reliabel untuk menilai tingkat kegunaan suatu sistem dari sudut pandang pengguna. Metode ini melibatkan 10 pernyataan yang dinilai menggunakan skala Likert dari 1 (Sangat Tidak Setuju) hingga 5 (Sangat Setuju). Keunggulan metode SUS terletak pada kesederhanaannya, karena hanya memerlukan waktu singkat untuk mendapatkan hasil yang dapat diandalkan. SUS memberikan hasil yang dapat diinterpretasikan secara kuantitatif dan

kualitatif untuk mengidentifikasi kekuatan dan kelemahan suatu sistem. Nilai rata-rata SUS secara global adalah 68, yang dianggap sebagai batas minimal untuk sistem yang dapat diterima pengguna (Putra, dkk, 2025:123).

System Usability Scale (SUS) merupakan metode evaluasi pengujian langsung yang digunakan untuk menilai tingkat usability suatu system atau aplikasi berdasarkan pengalaman pengguna. *System Usability Scale* (SUS) memiliki tingkat keandalan tinggi, banyak digunakan, efektif dalam mengukur pengalaman pengguna, serta efisien dari segi biaya dan implementasi. Metode ini pertama kali dikembangkan oleh John Brooke pada tahun 1986 dan hingga kini menjadi standar dalam pengukuran usability di berbagai bidang. SUS terdiri dari sepuluh pernyataan dengan skala penilaian Likert, di mana pengguna diminta untuk memberikan tanggapan mengenai kemudahan penggunaan, efisiensi, serta kenyamanan dalam mengoperasikan suatu sistem. SUS memiliki keunggulan dalam mengukur usability secara cepat dan efektif, sehingga banyak digunakan dalam berbagai penelitian terkait pengalaman pengguna. Metode ini dapat diterapkan pada berbagai jenis sistem, termasuk aplikasi berbasis *web*, sistem informasi, hingga perangkat lunak lainnya. Salah satu alasan utama penggunaan SUS adalah kemampuannya dalam memberikan hasil evaluasi yang objektif dan mudah dipahami oleh pengembang sistem (Jayanti dan Putra, 2025:840).

Nilai rata-rata = $\sum \frac{xi}{N}$ …….(2.1)

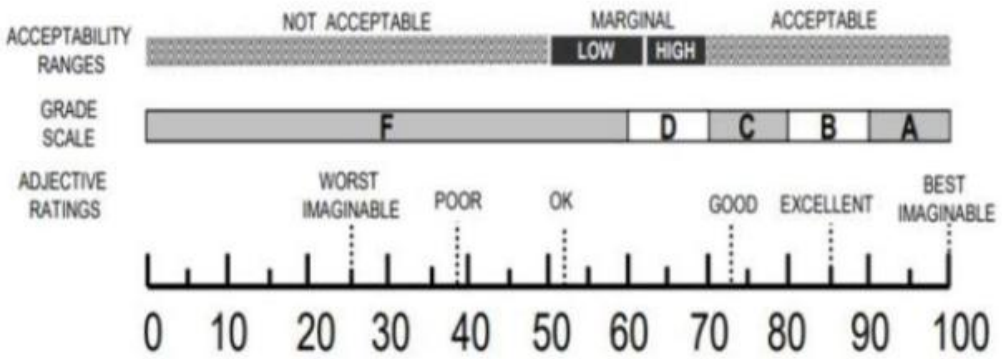
xi: nilai score responden

N: Jumlah Responden

Rumus pada persamaan 2.1 memastikan bahwa nilai keseluruhan diperoleh dengan membagi total nilai responden dengan jumlah responden. Langkah ini penting untuk mengukur representasi rata-rata dari skor yang diperoleh.

Skor SUS = (∑(Skor Positif-1) + ∑(5-Skor Negatif) x 2.5 …..(2.2)

Dalam perhitungan *System Usability Scale* (SUS), metode penskalaan skor dilakukan secara sistematis untuk memperoleh penilaian komprehensif terhadap usabilitas suatu aplikasi. Untuk pernyataan bernomor ganjil (seperti pernyataan 1, 3, 5, 7, 9), skor responden akan dikurangi dengan angka 1, yang memungkinkan penilaian kontributif dari aspek positif. Sementara untuk pernyataan bernomor genap (seperti pernyataan 2, 4, 6, 8, 10), perhitungan dilakukan dengan mengurangkan skor responden dari nilai 5, yang membantu mengidentifikasi persepsi negatif pengguna. Setelah melakukan perhitungan individual untuk setiap pernyataan, seluruh skor tersebut dijumlahkan dan dikalikan dengan konstanta 2.5. Proses ini menghasilkan nilai akhir dalam rentang 0 hingga 100, yang secara akurat menggambarkan tingkat usabilitas aplikasi dari perspektif pengguna. Interpretasi Skor SUS dapat dilihat pada Gambar 2.1.



Gambar 2.1. Interpretasi Skor SUS

Skor SUS diinterpretasikan dengan menggunakan beberapa tingkatan penerimaan. Skor dengan rentang 0–50 dianggap *not acceptable* (tidak dapat diterima), 51–70 masuk dalam kategori marginal (batas bawah), dan skor di atas 70 masuk ke dalam kategori *acceptable* (dapat diterima). Selain itu, skor SUS juga diklasifikasikan berdasarkan tingkat kualitas, yaitu mulai dari *worst imaginable* (terburuk yang bisa dibayangkan) hingga *best imaginable* (terbaik yang bisa dibayangkan). Rentang ini memberikan pemahaman yang jelas mengenai tingkat usabilitas sistem berdasarkan pengalaman pengguna (Shalihah, dkk, 2025:2087).

Berdasarkan kutipan-kutipan diatas, dapat disimpulkan bahwa *System Usability Scale* (SUS) merupakan metode evaluasi *usability* yang digunakan untuk mengukur tingkat kemudahan, kenyamanan, dan efektivitas penggunaan suatu sistem berdasarkan pengalaman pengguna. SUS terdiri dari 10 pernyataan dengan skala Likert yang menghasilkan penilaian secara kuantitatif dan kualitatif. Metode ini memiliki keunggulan

karena sederhana, cepat, reliabel, mudah diterapkan, serta mampu memberikan hasil evaluasi yang objektif sehingga banyak digunakan dalam pengujian berbagai sistem dan aplikasi berbasis teknologi informasi.

B. Kajian Empiris

Kajian empiris pada penelitian ini mengacu pada beberapa penelitian terdahulu yang membahas pengembangan sistem informasi *inventory* berbasis *web*. Penelitian yang dilakukan oleh Zhafira dkk. (2025:2498) menunjukkan bahwa penerapan sistem informasi *inventory* yang dilengkapi *barcode scanner* mampu mempermudah proses pencatatan dan pelacakan persediaan. Selain itu, sistem tersebut memberikan kemudahan bagi manajer dalam memantau ketersediaan stok serta menyusun laporan secara *real-time* melalui berbagai perangkat, sehingga efisiensi dan produktivitas operasional PT Matosa mengalami peningkatan.

Hasil penelitian lain yang dilakukan oleh Yanti dan Kurniawan (2026:1432) juga membuktikan bahwa sistem informasi *inventory* berbasis *web* dapat mengatasi kendala pengelolaan persediaan yang sebelumnya masih dilakukan secara manual di PT BPI Jakarta. Sistem yang dikembangkan mampu mengintegrasikan pengelolaan data barang, data pengguna, transaksi barang masuk dan keluar, proses peminjaman serta pengembalian alat, hingga penyusunan laporan *inventory* secara otomatis. Integrasi tersebut menghasilkan informasi stok yang lebih akurat dan dapat dipantau secara *real-time*, sehingga kegiatan pengawasan serta pengendalian persediaan menjadi lebih efektif.

Penelitian yang dilakukan oleh Ilhamsyah dkk. (2025:6) melalui "Rancang Bangun Sistem Manajemen *Inventory* Gudang di CV. Ada Trading Indonesia Berbasis *Website*" menunjukkan bahwa penerapan sistem berbasis *website* mampu meningkatkan efektivitas pengelolaan gudang. Sistem tersebut memudahkan petugas gudang dalam mencatat data barang masuk dan barang keluar secara lebih cepat dan akurat sehingga dapat mengurangi kesalahan pencatatan. Selain itu, ketersediaan informasi stok secara *real-time* membantu pengguna dalam melakukan pemantauan persediaan dan mendukung pengambilan keputusan terkait manajemen stok. Proses pendataan barang juga menjadi lebih terstruktur sehingga pencarian data lebih mudah dilakukan tanpa bergantung pada dokumen fisik.

Penelitian lain yang dilakukan oleh Prasetyo (2025:24) dengan judul "Perancangan Sistem Informasi *Inventory* Sparepart Mobil Berbasis *Desktop* pada PT Nusantara Bekasi" menyatakan bahwa perancangan sistem informasi penjualan dan persediaan barang mampu meningkatkan proses pengolahan data pada perusahaan. Penerapan sistem tersebut diharapkan dapat mempercepat pengelolaan data, khususnya pada aktivitas penjualan dan pengendalian persediaan, sehingga pelaksanaan pekerjaan menjadi lebih efektif dan efisien.

Penelitian ini memiliki kesamaan dengan beberapa penelitian terdahulu karena sama-sama mengembangkan sistem informasi *inventory* yang bertujuan meningkatkan efektivitas dan efisiensi dalam pengelolaan persediaan barang. Meskipun memiliki tujuan yang serupa, terdapat sejumlah perbedaan pada

metode pengembangan sistem, teknologi yang digunakan, maupun objek penelitian.

Penelitian yang dilakukan oleh Zhafira dkk. menerapkan metode *Waterfall*, sedangkan penelitian ini menggunakan *Rapid Application Development* (RAD) yang menekankan proses pengembangan secara cepat melalui pendekatan iteratif. Perbedaan juga terlihat pada penelitian Yanti dan Kurniawan yang mengembangkan sistem menggunakan metode *Waterfall*, *framework CodeIgniter*, dan basis data MySQL, sedangkan penelitian ini memanfaatkan metode RAD, *framework* Laravel, serta basis data PostgreSQL. Sementara itu, penelitian Ilhamsyah dkk. sama-sama mengembangkan sistem informasi *inventory*, tetapi menggunakan metode *Waterfall* dengan basis data MySQL, berbeda dengan penelitian ini yang menerapkan metode RAD dan PostgreSQL. Adapun penelitian Prasetyo menggunakan metode *Waterfall*, bahasa pemrograman Java, serta basis data MySQL, sedangkan penelitian ini dibangun menggunakan *framework* Laravel dengan basis data PostgreSQL melalui metode RAD.

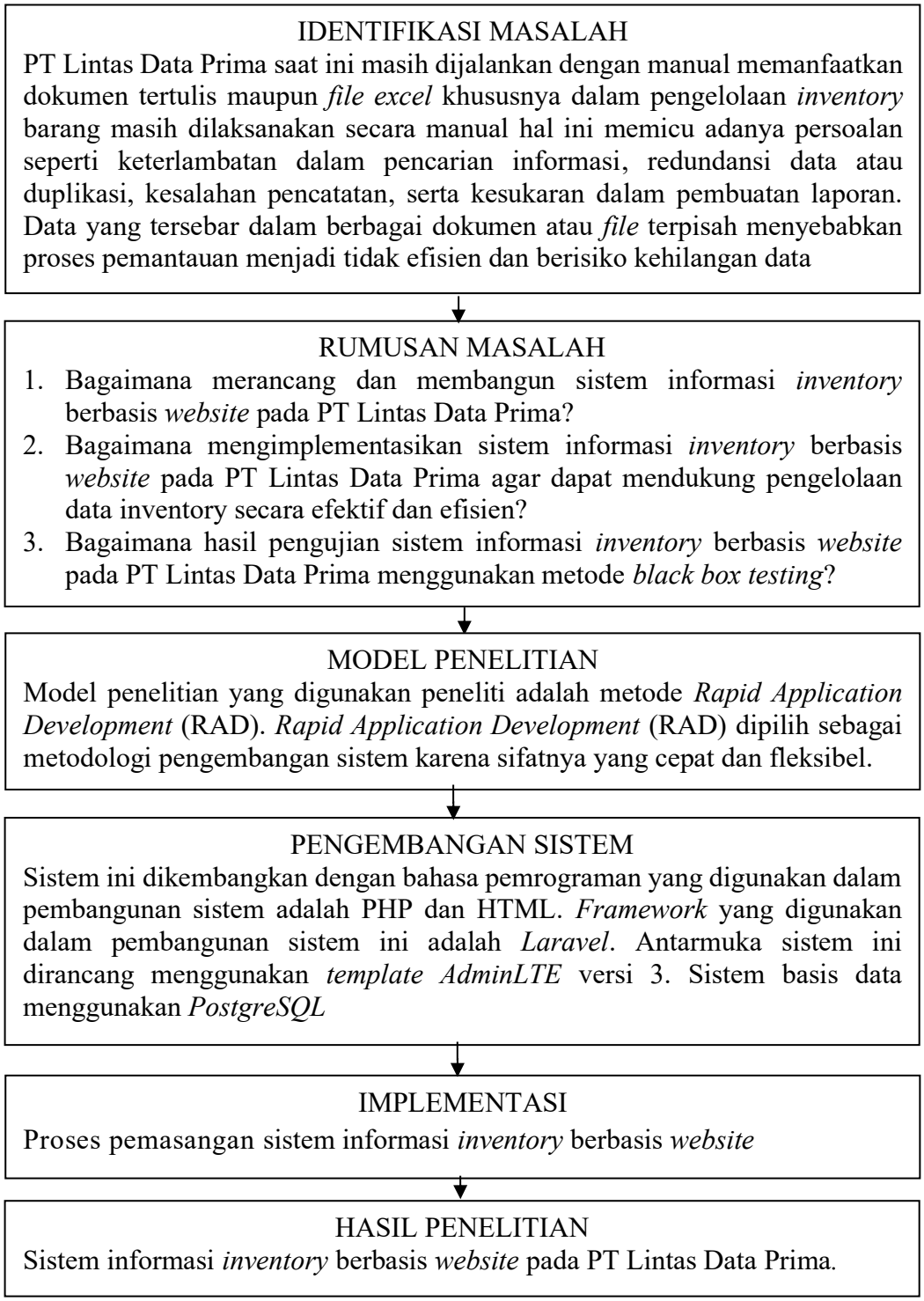
Selain aspek metodologi dan teknologi, objek penelitian juga menunjukkan karakteristik yang berbeda. Penelitian Zhafira dkk. dilaksanakan di PT Matosa yang bergerak pada bidang distribusi, sedangkan penelitian Yanti dan Kurniawan dilakukan di PT BPI Jakarta yang bergerak pada bidang *cleaning service*. Penelitian Ilhamsyah dkk. mengambil objek CV Ada Trading Indonesia yang bergerak di bidang perdagangan, sementara penelitian Prasetyo dilakukan di PT Nusantara Bekasi yang bergerak pada bidang penjualan

sparepart mobil. Berbeda dengan penelitian-penelitian tersebut, penelitian ini dilaksanakan di PT Lintas Data Prima, yaitu perusahaan penyedia layanan internet (*Internet Service Provider*/*ISP*) dan teknologi jaringan. Aktivitas operasional perusahaan ini melibatkan pengelolaan berbagai perangkat jaringan, seperti *router*, *switch*, kabel serat optik, serta perangkat pendukung lainnya yang digunakan pada proses instalasi, pemeliharaan, dan operasional layanan internet. Kondisi tersebut menyebabkan kebutuhan pengelolaan persediaan memiliki karakteristik yang berbeda sehingga sistem yang dikembangkan disesuaikan dengan proses bisnis perusahaan agar mampu mengelola stok, transaksi barang masuk dan keluar, serta penyusunan laporan secara lebih efektif.

Berdasarkan perbandingan tersebut, kebaruan penelitian ini terletak pada penerapan metode *Rapid Application Development* (RAD) yang mendukung tahap pengembangan sistem secara lebih cepat melalui siklus iteratif. Selain itu, penggunaan *framework* Laravel yang dipadukan dengan basis data *PostgreSQL* diharapkan mampu menghasilkan sistem informasi *inventory* yang lebih terstruktur, aman, serta memiliki kinerja yang optimal.

C. **Kerangka Berpikir**

Kerangka berpikir dalam penelitian ini dapat dilihat pada Gambar 2.2.



Gambar 2.2. Kerangka Berpikir