BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian yang berjudul "Implementasi dan Pelatihan *Aplikasi Point of Sale* Pada UMKM Nabila Sembako" menggunakan teknologi *website* seperti *PHP*, *MySQL, HTML, CSS,* dan *JavaScript* untuk mendukung pengelolaan transaksi dan stok barang. Hasil penelitian menunjukkan bahwa sistem mampu meningkatkan efisiensi operasional, pengelolaan stok, serta pelaporan keuangan secara otomatis. Selain itu, penggunaan sistem juga mendorong literasi digital pelaku usaha dalam mengelola bisnisnya (Safitri dkk., 2024).

Penelitian lain yang berjudul "Rancang Bangun Sistem *Point Of Sales* (POS) Berbasis *Web* dan *Mobile* untuk UMKM" merancang sistem *POS* berbasis web dan mobile dengan model *Waterfall* yang mampu mengurangi kesalahan transaksi, mempercepat pelayanan, serta meningkatkan kepuasan pengguna. Sistem tersebut dirancang untuk membantu proses transaksi penjualan menjadi lebih terstruktur dan efisien sehingga dapat meningkatkan kualitas pelayanan kepada pelanggan. (Terusan Ryacudu dkk., 2025).

Penelitian yang berjudul "Inovasi Teknologi dalam Manajemen Penjualan: *Aplikasi Point of Sales* Berbasis *Web* untuk UMKM" mengenai *aplikasi POS* berbasis *web* menggunakan *PHP Native* yang terintegrasi dengan layanan pembayaran digital menunjukkan bahwa sistem mampu meningkatkan akurasi, efisiensi penjualan, serta mendukung pengambilan keputusan berbasis data. Integrasi pembayaran digital pada sistem juga membantu mempercepat proses transaksi dan mempermudah pengelolaan data penjualan secara otomatis(Andy dan Widiono, 2024).

Penelitian berjudul "Perancangan Sistem Informasi Manajemen Stok Dan Penjualan Pada Toko Oli Joni" menggunaan *JavaScript* dalam pengembangan *aplikasi web* juga memiliki peran signifikan dalam menciptakan antarmuka yang interaktif dan dinamis. *JavaScript* memungkinkan sistem merespons aksi pengguna secara *real time*, seperti validasi input, pemrosesan data tanpa memuat ulang halaman, serta peningkatan interaktivitas pada *aplikasi web*

sehingga mampu meningkatkan efisiensi dan kenyamanan pengguna dalam menggunakan sistem (Hardi dkk., 2022).

Pada halaman admin, penggunaan template antarmuka juga menjadi faktor penting. Penelitian berjudul "Pengenalan *Framework Css* Dalam Pembuatan Halaman Admin *Aplikasi Web* Menggunakan Admin Lte" menyatakan bahwa AdminLTE merupakan *framework CSS* yang efektif digunakan untuk membangun halaman admin *aplikasi web* yang responsif dan terstruktur. AdminLTE menyediakan berbagai komponen antarmuka siap pakai seperti tabel, formulir, menu navigasi, dan grafik yang dapat mempercepat proses pengembangan serta menghasilkan tampilan admin yang profesional dan mudah digunakan. Penelitian lain juga menyebutkan bahwa AdminLTE membantu pengembangan antarmuka sistem menjadi lebih efektif dan responsif (Nata dkk., 2023).

Selain itu, penelitian berjudul " *Aplikasi Invertory* Yang Terintegrasi Dengan Inventaris Barang Menggunakan *Barcode Scanner* Berbasis *Web* " Menunjukkan bahwa penerapan *barcode scanner* pada sistem *inventory* berbasis web mampu mempermudah proses pelacakan barang, meningkatkan akurasi input data, serta mempercepat proses pencarian barang dalam sistem. Penelitian tersebut juga menjelaskan bahwa *barcode scanner* dapat terhubung langsung dengan *database* sehingga proses identifikasi barang menjadi lebih efektif dan efisien (Suseno dan Supratman, 2021).

Penelitian ini berfokus pada pengembangan sistem *Point of Sale (POS)* berbasis web yang dilengkapi dengan *fitur barcode scanner* pada UMKM CV. Sumber Barokah Sentosa. Penelitian sebelumnya lebih banyak diterapkan pada toko sembako, sistem penjualan umum, atau sistem *inventory* barang, sedangkan penelitian ini diterapkan pada bengkel motor yang memiliki variasi dan kompleksitas sparepart lebih tinggi berdasarkan merk, tipe kendaraan, dan kategori barang. Selain itu, *fitur barcode scanning* pada penelitian ini digunakan tidak hanya untuk pelacakan barang, tetapi juga untuk mendukung proses transaksi penjualan dan pengelolaan stok secara real time sehingga proses pencatatan barang menjadi lebih cepat, akurat, dan efisien. Penelitian

ini juga menerapkan antarmuka berbasis *HTML*, *CSS*, *JavaScrip*t, serta template AdminLTE yang *responsif* guna mendukung kemudahan penggunaan sistem oleh admin maupun karyawan

2.2 Kerangka Pemikiran

Kerangka pemikiran ini menggambarkan tahapan yang dilakukan dalam merancang dan membangun *aplikasi* pelayanan pelanggan berbasis *website* di UMKM CV.Sumber Barokah Sentosa, Pedan, Klaten. Kerangka pemikiran bisa dilihat pada Gambar 2.1.



Gambar 2. 1 Kerangka Pemikiran

2.3 Landasan Teori

2.3.1 Implementasi

Implementasi adalah suatu tindakan nyata atau proses penerapan dari sebuah rencana, desain, atau konsep ke dalam bentuk yang dapat dioperasikan. implementasi merupakan tahap penting dalam siklus pengembangan sistem informasi, karena pada tahap ini rancangan sistem yang telah dibuat diubah menjadi bentuk nyata sehingga dapat digunakan sesuai tujuan. (Amelina, 2024).

2.3.2 Aplikasi

Aplikasi adalah suatu perangkat lunak atau program komputer yang dibuat khusus untuk memenuhi kebutuhan berbagai aktivitas dan pekerjaan tertentu. *Aplikasi* memanfaatkan kemampuan komputer secara langsung untuk melakukan suatu tugas yang diinginkan pengguna. *Aplikasi* membuat proses yang sebelumnya rumit bisa menjadi lebih sederhana dan cepat, dan dapat berbentuk *mobile app*, *web app*, atau *desktop app* (Darin Farhanah dkk., 2022)

2.3.3 Pelayanan Kepada Pelanggan *(Customer Service)*

Pelayanan kepada pelanggan (*customer service*) adalah setiap kegiatan atau tindakan yang dilakukan oleh sebuah perusahaan atau organisasi untuk memberikan kepuasan kepada pelanggan. Melalui pelayanan ini, keinginan dan kebutuhan pelanggan dapat terpenuhi. Pelayanan yang baik dapat menciptakan hubungan jangka panjang antara perusahaan dan pelanggan, meningkatkan loyalitas, dan citra merek. Pelayanan dapat diberikan melalui berbagai saluran seperti panggilan telepon, *live chat* di *website*, *email*, atau tatap muka(I Gde O, 2022).

2.3.4 Usaha Mikro, Kecil, dan Menengah (UMKM)

UMKM (Usaha *Mikro*, Kecil, dan Menengah) merupakan usaha yang dijalankan oleh individu, rumah tangga, atau badan usaha berskala kecil yang memiliki kontribusi signifikan terhadap perekonomian suatu negara. Penggolongan UMKM umumnya ditentukan berdasarkan omzet tahunan, Jumlah aset, serta Jumlah tenaga kerja sesuai dengan kriteria yang ditetapkan dalam peraturan perundang-undangan. Namun, kendala utama yang sering dihadapi UMKM adalah pengelolaan operasional yang masih manual. Oleh

karena itu, penerapan teknologi informasi sangat diperlukan untuk mempercepat proses transaksi, meminimalkan kesalahan pencatatan, dan meningkatkan efisiensi pelayanan secara keseluruhan (Septiani dkk., 2024).

2.3.5 *Website*`

Website atau situs *web* adalah kumpulan halaman dalam suatu *domain* yang memuat berbagai informasi seperti teks, gambar, video, suara, dan elemen interaktif lainnya, yang saling terhubung melalui hyperlink, dan dapat diakses melalui *browser* (Wororomi dkk., 2025).

2.3.6 *PHP* Native

Penerapan *PHP* Native dalam pengembangan sistem informasi masih relevan digunakan dalam membangun *aplikasi* berbasis *web* yang sederhana namun efektif. Penelitian mereka pada perancangan Sistem Informasi Manajemen Perpustakaan menunjukkan bahwa *PHP* Native mampu mendukung kebutuhan fungsional pengguna, seperti pengelolaan data buku, anggota, serta transaksi peminjaman dan pengembalian. Keunggulan utama penggunaan *PHP Native* adalah kemudahan dalam implementasi tanpa harus bergantung pada *Framework* tertentu, sehingga lebih ringan, mudah dipahami, dan sesuai untuk skala institusi pendidikan maupun organisasi kecil menengah (Zonries Siahaan dan Agustina Dalimunthe, 2023).

2.3.7 Java Script

JavaScript adalah salah satu bahasa pemrograman tingkat tinggi yang berperan penting dalam pengembangan *web* modern, terutama untuk menciptakan konten yang interaktif dan dinamis pada halaman *web*. Dalam konteks pembuatan *aplikasi web*, *JavaScript* berjalan di sisi klien (*client-side*) sehingga memungkinkan elemen visual seperti menu, animasi, dan validasi formulir bereaksi secara *real-time* tanpa perlu memuat ulang halaman, meningkatkan pengalaman pengguna secara signifikan. Selain itu, *JavaScript* banyak digunakan bersama *HTML* dan *CSS* untuk menyusun struktur, desain, dan fungsi interaktif sebuah *website*, serta mendukung berbagai pustaka dan *Framework* modern. Dalam penelitian sistem informasi berbasis *web*, *JavaScript* terbukti efektif dalam mengelola data serta memaksimalkan interaksi digital,

seperti peningkatan layanan *website* melalui mekanisme pemrosesan input dan validasi data secara dinamis (Raehan dkk., 2025).

2.3.8 AdminLTE

AdminLTE merupakan *template* antarmuka (*User Interface*) berbasis *web* yang digunakan untuk membangun halaman admin pada *aplikasi web* secara responsif dan terstruktur. AdminLTE dikembangkan dengan memanfaatkan *Framework CSS* seperti *Bootstrap*, sehingga menyediakan berbagai komponen siap pakai, antara lain menu navigasi, tabel data, formulir, ikon, serta grafik yang mendukung kebutuhan pengelolaan sistem. Penggunaan AdminLTE dalam pengembangan *aplikasi web* dapat mempercepat proses pembuatan halaman admin serta menghasilkan tampilan yang konsisten dan profesional. AdminLTE terbukti efektif sebagai *Framework CSS* dalam pembuatan halaman admin *aplikasi web* yang responsif dan mudah digunakan, sehingga mendukung efisiensi pengembangan sistem dan meningkatkan kenyamanan pengguna dalam mengelola data (Nata dkk., 2023).

2.3.9 *CSS*

Cascading Style Sheets (*CSS*) merupakan komponen penting dalam pengembangan *website* karena berfungsi mengatur tata letak, tampilan, serta desain visual agar lebih menarik dan responsif. Penggunaan *CSS* yang tepat dapat meningkatkan kualitas antarmuka pengguna *(UI)* sekaligus memberikan pengalaman pengguna *(UX)* yang lebih baik. Melalui pemanfaatan *CSS* , elemen-elemen pada halaman *web* dapat disesuaikan seperti warna, tipografi, dan tata letak, sehingga menghasilkan tampilan yang konsisten dan profesional pada berbagai perangkat (Gunawan dan Firdaus, 2024).

2.3.10 *Database* (Basis Data)

Database , atau basis data, adalah kumpulan data yang saling berhubungan dan disimpan secara elektronik dalam sistem komputer, yang memungkinkan penyimpanan, pengelolaan, dan pengambilan data secara efisien. Selain itu, *database* sering dioperasikan melalui sistem manajemen basis data *(DBMS)* yang mengatur integritas, keamanan, serta relasi antar data (Pradipta dkk., 2022).

2.3.11 MySQL

MySQL adalah sistem manajemen basis data relasional *(RDBMS) open source* yang sangat populer digunakan dalam pengembangan aplikasi *web*, terutama karena kemampuannya dalam mengelola, menyimpan, dan mempertahankan integritas data dengan efisien. *MySQL* mendukung operasi CRUD (*Create, Read, Update, Delete*), penggunaan indeks, query optimization, serta dukungan untuk full-text search yang berguna untuk pencarian teks. Berdasarkan penelitian *Optimization of MySQL Database* Šušter & Ranisavljević, (2023), teknik fisik pengaturan penyimpanan dan tuning data dapat meningkatkan performa *MySQL* secara signifikan. Penelitian lain *Perbandingan Performa Pencarian Data Berbasis Teks dengan Dan Tanpa Full-text Index pada Basis Data MySQL* menunjukkan bahwa penggunaan *full-text index* mempercepat proses pencarian tekstual dibandingkan tanpa *index*, khususnya pada *database* dengan volume data besar (Imam Thoib dkk., 2024).

2.3.12 *Barcode*

Barcode adalah representasi data dalam bentuk garis-garis paralel dengan ketebalan dan jarak tertentu yang dapat dibaca oleh mesin. *barcode* digunakan untuk mengidentifikasi suatu objek dengan cepat dan akurat menggunakan *Barcode scanner*. *Barcode* banyak digunakan dalam berbagai bidang seperti perdagangan, logistik, industri, hingga bidang kesehatan untuk mempermudah pelacakan dan manajemen data (Berkat Zega, 2025).

2.3.13 *Barcode Scanner*

Barcode scanner merupakan perangkat input yang digunakan untuk membaca kode *barcode* pada suatu barang agar data dapat diproses secara otomatis oleh sistem. Penggunaan *barcode scanner* bertujuan untuk mempercepat proses pencarian dan pendataan barang serta mengurangi kesalahan input data secara manual. Menggunakan *Barcode Scanner* Berbasis *Web*, "Informasi pada *Barcode* berisi enkripsi dari sejumlah digit angka. Saat *barcode* tersebut di scan dengan alat *barcode scanner*, maka kode tersebut secara otomatis terhubung ke data barang yang sudah disimpan dalam *database* (Alda, 2021).

2.3.14 Retur

Retur adalah proses pengembalian barang dari konsumen kepada penjual yang biasanya terjadi ketika barang yang diterima tidak sesuai pesanan, rusak, atau cacat produksi. Pengembalian barang dalam transaksi *online* di Indonesia telah diatur dalam *self regulation* marketplace, termasuk pada metode pembayaran *Cash on Delivery (COD)*. *Self regulation* tersebut disusun untuk melindungi hak konsumen agar dapat mengajukan retur apabila barang yang diterima tidak sesuai, sekaligus memberi kejelasan kewajiban penjual dalam menindaklanjuti pengembalian. Namun, dalam praktiknya masih ditemukan berbagai kendala seperti penjual yang menolak retur atau konsumen yang enggan membayar barang COD, sehingga menimbulkan permasalahan hukum maupun sosial. Oleh karena itu, pengaturan retur melalui *self regulation* marketplace menjadi penting sebagai upaya perlindungan konsumen sesuai dengan Undang-Undang Perlindungan Konsumen di Indonesia (Diurnal dkk., 2022).

2.3.15 Point of sale

Point of sale (POS) adalah sistem yang digunakan untuk mencatat transaksi penjualan, mengelola stok barang, serta menghasilkan laporan keuangan secara otomatis. *POS* modern biasanya berbasis *web* sehingga dapat digunakan secara *real-time* dan terintegrasi dengan *database* . *Aplikasi POS* berbasis *web* yang dirancang untuk UMKM mampu meningkatkan efisiensi manajemen penjualan, mengurangi kesalahan pencatatan transaksi, serta memberikan kemudahan dalam penyusunan laporan. Dengan demikian, penerapan *POS* tidak hanya membantu pelaku UMKM dalam operasional sehari-hari, tetapi juga berperan dalam mendukung pengambilan keputusan bisnis yang lebih akurat (Andy dan Widiono, 2024).

2.3.16 Analisis Dan Desain Sistem Berorientasi Objek

Analisis dan desain sistem berorientasi objek adalah pendekatan dalam rekayasa perangkat lunak yang memandang sistem sebagai kumpulan objek yang saling berinteraksi. Analisis objek bertujuan mengidentifikasi kebutuhan fungsi sistem dan entitas yang terlibat, sedangkan desain objek memetakan

kebutuhan tersebut ke dalam struktur kelas, relasi, dan interaksi yang dapat diimplementasikan. Pendekatan ini mendukung modularitas, enkapsulasi, dan reuse kode, memudahkan pemeliharaan dan pengembangan sistem di masa mendatang (Randa, 2022).

2.3.16 *UML*

UML atau *Unified Modeling Language* adalah bahasa pemodelan visual standar yang digunakan untuk merancang dan memodelkan sistem perangkat lunak berbasis objek. *UML* memungkinkan pengembang untuk menyajikan struktur dan perilaku sistem melalui Diagram seperti *Use Case* , *Class Diagram* , *Activity Diagram* , *Sequence Diagram* , dan lain-lain. *UML* merupakan komunikasi antara analis sistem, pengembang, dan pemangku kepentingan menjadi lebih jelas dan sistematis (Siska Narulita dkk., 2024).

UML digunakan untuk membantu proses perancangan sistem dengan menggambarkan interaksi, alur aktivitas, serta struktur dari sistem yang akan dikembangkan. *UML* terdiri dari beberapa jenis Diagram yang memiliki fungsi dan tujuan masing-masing dalam memodelkan sistem secara menyeluruh. Adapun jenis-jenis Diagram yang digunakan antara lain:

a. *Use Case* Diagram

Use Case menggambarkan interaksi antara pengguna dan sistem, serta menunjukkan jenis-jenis interaksi yang terjadi dalam suatu program. Dengan kata lain, *Use Case*adalah rangkaian interaksi yang saling terkait antara sistem dan aktor. Oleh karena itu, *Use Case* biasanya direpresentasikan dalam bentuk Diagram untuk memperlihatkan perilaku (*behaviour*) sistem informasi yang akan dikembangkan (Sandfreni dkk., 2021).

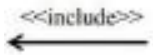
Selain itu, *Use Case* Diagram juga berfungsi sebagai alat komunikasi antara analis sistem, *developer*, dan *stakeholder*, karena Diagram ini menggambarkan kebutuhan fungsional secara sederhana namun *informatif*. Dengan adanya *Use Case* , kebutuhan pengguna dapat diidentifikasi lebih awal sehingga meminimalkan kesalahan dalam proses pengembangan sistem.

Use Case Diagram biasanya terdiri dari beberapa elemen utama seperti aktor, *Use Case* , batas sistem (*system boundary*), serta hubungan antar elemen

seperti *association, include, dan extend*. Aktor merepresentasikan pengguna atau sistem lain yang berinteraksi, sedangkan *Use Case* menggambarkan proses atau layanan yang disediakan sistem. Hubungan include digunakan untuk menggambarkan proses yang selalu disertakan, sementara extend menunjukkan proses tambahan yang bersifat opsional.

Diagram ini juga membantu dalam memetakan alur aktivitas secara keseluruhan sebelum masuk ke tahap perancangan yang lebih detail seperti *Activity* Diagram atau *Sequence* Diagram . *Use Case* Diagram menjadi dasar yang penting dalam proses analisis dan perancangan sistem karena mampu memberikan gambaran umum dari fungsi-fungsi yang akan dikembangkan. Simbol *Use Case* bisa dilihat pada Tabel 2.1.

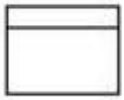
Tabel 2. 1 Simbol *Use Case*

Simbol	Keterangan
	Aktor: Mewakili peran orang, sistem yang lain, atau alat Ketika berkomunikasi
	*Use Case :* Abstraksi dan interaksi antara sistem dan aktor.
	Association: Abstraksi dari penghubung antara aktor dengan *Use Case* .
	Generalisasi: Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *Use Case* .
	Menunjukkan bahwa suatu *Use Case*seluruhnya merupakan fungsionalitas dari *Use Case*lainnya.
	Menunjukkan bahwa suatu *Use Case*seluruhnya merupakan fungsionalitas dari *Use Case*lainnya jika suatu kondisi terpenuhi.

b. *Activity* Diagram

Activity Diagram merupakan salah satu Diagram perilaku (behavioral *Diagram*) dalam *UML* yang digunakan untuk memodelkan alur aktivitas atau proses bisnis dalam sistem dengan menampilkan langkah-langkah kerja yang dilakukan baik oleh sistem maupun pengguna, lengkap dengan kondisi percabangan, pengulangan, serta aliran kontrol dari satu aktivitas ke aktivitas lainnya. *Activity Diagram* berperan penting dalam memahami dinamika sistem dan dapat diberi semantik formal agar dapat diverifikasi secara *runtime* guna memastikan sistem berjalan sesuai dengan model yang dirancang. Selain itu, *Activity* Diagram juga membantu pengembang menganalisis efisiensi proses, mendeteksi potensi *bottleneck*, serta memberikan visualisasi yang jelas terhadap alur kerja sistem sebelum tahap implementasi dilakukan (Siewe dan Ngounou, 2025). Simbol *Activity* bisa dilihat pada Tabel 2.2.

Tabel 2. 2 *Activity Diagram*

Simbol	Nama	Keterangan
	Status awal	Sebuah Diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitasi biasanya diawali dengan kata kerja
	Percabangan/ *Decision*	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Penggabungan / Join	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	Status akhir	Status akhir yang dilakukan sistem, sebuah Diagram aktivitas memiliki sebuah status akhir.
	*Swimlane*	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

c. *Class* Diagram

Class Diagram merupakan Diagram struktur (*structural Diagram*) dalam *UML* yang digunakan untuk menggambarkan struktur kelas dalam sistem, termasuk atribut, operasi (metode), serta hubungan antar kelas seperti *asosiasi, generalisasi, dan agregasi*. Diagram ini berfungsi untuk menjelaskan bagaimana data dan logika bisnis dikelola dalam sistem. tata letak dan struktur *Class* Diagram berpengaruh besar terhadap tingkat pemahaman pengembang terhadap sistem. *Class* Diagram yang terorganisasi dengan baik membantu dalam mengidentifikasi dependensi antar kelas dan meningkatkan kualitas desain perangkat lunak, *Class* Diagram menjadi dasar dalam tahap desain sistem karena mempresentasikan kerangka logis yang akan diimplementasikan ke dalam kode program menggunakan paradigma berorientasi objek

Selain itu, *Class* Diagram juga mempermudah proses dokumentasi sistem karena menggambarkan hubungan hierarki dan struktur internal dari setiap objek yang berperan dalam sistem. Diagram ini mencakup informasi penting seperti visibilitas atribut dan metode *(public, private, protected*), tipe data, dan hubungan antar kelas yang menentukan bagaimana data mengalir dalam sistem. Dengan adanya Diagram ini, pengembang dapat memastikan bahwa struktur perangkat lunak telah dirancang secara konsisten sebelum masuk ke tahap implementasi.

Class Diagram juga sering digunakan sebagai acuan dalam pengembangan basis data karena struktur kelas memiliki keterkaitan erat dengan model tabel relasional. Misalnya, atribut dalam kelas dapat diterjemahkan menjadi kolom dalam tabel, sedangkan asosiasi antar kelas dapat diimplementasikan sebagai relasi antar tabel. Oleh karena itu, *Class* Diagram tidak hanya penting dalam desain berorientasi objek, tetapi juga mendukung proses normalisasi data dan pemodelan *database* yang lebih terstruktur.

Dengan penyajian yang jelas dan terperinci, *Class* Diagram membantu menjaga keselarasan antara kebutuhan fungsional sistem dan implementasi teknisnya (Bergström dkk., 2022). Simbol *Class* Diagram bisa dilihat pada Tabel 2.3.

Tabel 2. 3 *Class* Diagram

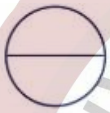
Simbol	Nama	Keterangan
	Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
	Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	Collaboration	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
	Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.

d. Sequence Diagram

Sequence Diagram merupakan alat yang sangat populer dalam pengembangan sistem informasi berorientasi objek untuk menampilkan

interaksi antar objek dalam suatu sistem secara berurutan berdasarkan waktu. Diagram ini membantu pengembang memahami alur komunikasi antar objek dan memastikan proses berjalan sesuai dengan logika sistem yang dirancang (Sopriani dan Purwanto, 2023). Simbol *Sequence* Diagram bisa dilihat pada Tabel 2.4.

Tabel 2. 4 *Sequence* Diagram

Simbol	Nama	Keterangan
	Entity *Class*	Gambaran sistem sebagi landasan dalam Menyusun basis data
	Boundary *Class*	Menangani komunikasi antar lingkungan sistem
	Control *Class*	Bertanggung jawab terhadap kelas kelas terhadap objek yang berisi logika.
	Recursive	Pesan untuk dirinya.
	Activation	Mewakili proses durasi aktivasi sebuah operasi.
	Life Line	Komponen yang digambarkan garis putus-putus dengan objek.

e. *Component* Diagram

Component Diagram digunakan untuk menggambarkan struktur dan hubungan antar komponen dalam sistem perangkat lunak. Diagram ini menunjukkan bagaimana setiap komponen saling berinteraksi melalui antarmuka dan dependensi, sehingga mempermudah proses pemodelan arsitektur sistem secara modular dan terstruktur (Puspita Sari dkk., 2022). Simbol *Component* Diagram bisa dilihat pada Tabel

Tabel 2. 5 *Component* Diagram

Simbol	Nama	Keterangan
	Package	Package merupakan sebuah bungkusan dari satu atau lebih *Component*
	*Component*	Bagian aplikasi berbentuk modul yang menjalankan fungsi tertentu
	dependency	Arah panah mengarah pada *Component* yang dipakai
	Interface	Layanan yang disediakan komponen

f. *Deployment* Diagram

Deployment Diagram digunakan untuk menggambarkan bagaimana komponen perangkat lunak diimplementasikan pada lingkungan fisik seperti server, *client*, dan perangkat jaringan lainnya. Diagram ini menunjukkan hubungan antar node serta distribusi komponen sistem, sehingga membantu pengembang memahami arsitektur penyebaran sistem secara keseluruhan. Melalui Diagram ini, pemetaan antara perangkat lunak dan perangkat keras menjadi lebih jelas, termasuk spesifikasi protokol komunikasi yang digunakan antar perangkat. Untuk memahami representasi visual yang digunakan dalam Diagram tersebut, simbol-simbol standar dalam (Asra dkk., 2023) . *Deployment* Diagram dapat dilihat pada Tabel 2.6.

Tabel 2. 6 *Deployment* Diagram

Simbol	Deskripsi
	*Package* merupakan sebuah bungkusan dari satu atau lebih
	Biasanya mencakup pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika didalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang telah didefinisikan sebelumnya pada Diagram komponen.
	Ketergantungan antar *node*, arah panah mengarah pada *node* yang dipakai
	Relasi antar *node*

2.3.16 Pengujian Grey Box

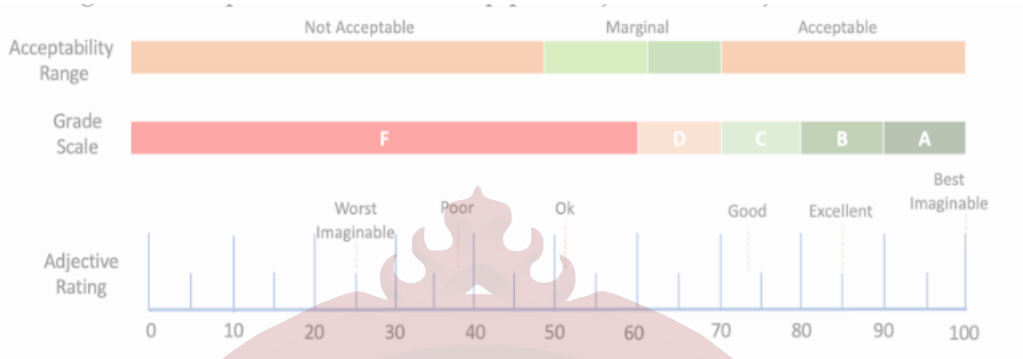
Grey Box merupakan metode pengujian perangkat lunak yang mengombinasikan pendekatan *black-box testing* dan *white-box testing*. Pada metode ini, penguji memiliki pengetahuan sebagian terhadap struktur internal sistem, seperti alur proses, struktur data, atau logika program, namun tetap melakukan pengujian berdasarkan fungsi dan perilaku sistem dari sudut pandang pengguna. *Grey Box* testing dilakukan dengan memanfaatkan pemahaman internal sistem untuk merancang skenario pengujian yang lebih efektif, tanpa harus menganalisis kode program secara menyeluruh. Pendekatan ini memungkinkan penguji untuk mengidentifikasi kesalahan fungsional sekaligus mendeteksi potensi kesalahan logika yang muncul akibat interaksi antar modul sistem. Pada tahap pengujian sistem, metode *Grey Box* Testing digunakan untuk memastikan bahwa setiap *fitur* aplikasi berjalan sesuai dengan spesifikasi yang telah ditetapkan, sekaligus memperhatikan mekanisme kerja internal sistem.

Dalam penelitian yang dilakukan pada aplikasi *F1Math*, *Grey Box* testing untuk menguji kestabilan sistem setelah dilakukan perubahan atau pengembangan *fitur.* Hasil pengujian menunjukkan bahwa metode ini efektif dalam menemukan bug pada tahap awal pengujian serta meningkatkan stabilitas dan keandalan aplikasi secara keseluruhan, Pengujian *Grey Box* umumnya disajikan dalam bentuk tabel yang merangkum *fitur* sistem, aktor yang terlibat, serta alur proses pengujian, sehingga pengujian dapat dilakukan secara terstruktur dan sistematis.(Kharisma Makassar, 2023).

2.3.17 SUS (*System Usability Scale*)

System Usability Scale (SUS) merupakan metode pengujian usability yang digunakan untuk mengukur tingkat kemudahan penggunaan suatu sistem atau aplikasi berdasarkan persepsi pengguna. Metode ini menggunakan instrumen kuesioner yang terdiri dari 10 pernyataan dengan skala *Likert* lima tingkat. Pernyataan bernomor ganjil bersifat Positif, sedangkan pernyataan bernomor genap bersifat negatif. Metode *SUS* banyak digunakan karena bersifat sederhana, cepat, tidak memerlukan jumlah responden yang besar, serta mampu menghasilkan skor kuantitatif yang mudah dipahami dan diinterpretasikan. Metode SUS melibatkan pengguna secara langsung tanpa memerlukan kriteria khusus dalam pemilihan responden. Pengujian ini dapat diterapkan pada berbagai jenis sistem, seperti aplikasi berbasis Android, *website*, maupun sistem informasi lainnya. Keunggulan metode SUS adalah kemampuannya untuk memberikan gambaran tingkat usability sistem secara efektif serta dapat langsung menunjukkan apakah sistem dapat diterima atau tidak oleh pengguna. Pengukuran Dalam metode SUS mencakup lima aspek utama, yaitu *learnability, efficiency, memorability, errors*, dan *satisfaction. Learnability* menunjukkan kemudahan sistem untuk dipelajari oleh pengguna, *efficiency* menggambarkan kemampuan sistem dalam membantu pengguna menyelesaikan tugas dengan efektif, *memorability* berkaitan dengan kemudahan pengguna dalam mengingat kembali cara penggunaan sistem, error menunjukkan tingkat kesalahan yang terjadi serta kemudahan pengguna dalam memperbaikinya, dan *satisfaction* menggambarkan tingkat kenyamanan serta kepuasan pengguna dalam

menggunakan sistem (Utami dkk., 2025). Interpretasi skor SUS bisa di lihat pada Gambar 2.2.



Gambar 2. 2 Skala Interpretasi Hasil Skor SUS

Sumber: (Utami dkk., 2025)

Berdasarkan skala tersebut, *acceptability range* menunjukkan tingkat penerimaan pengguna terhadap sistem, yang diklasifikasikan menjadi tiga kategori, yaitu:

Not Acceptable untuk skor SUS 0–58

Marginal untuk skor SUS 59–70

Acceptable untuk skor SUS 71–100.

Selain itu, hasil pengujian SUS juga dikonversikan ke dalam *grade scale* dan *adjective rating*. Pembagian interpretasi skor SUS disajikan pada Tabel 2.7.

Tabel 2. 7 Skala Interpretasi Hasil Skor SUS (Utami et al., 2025).

Rentang Skor SUS	***Grade***	***Adjective Rating***
≥ 85	A	*Best Imaginable*
80,3 – < 85	A	*Excellent*
74 – < 80,3	B	*Good*
68 – < 74	C	*OK*
51 – < 68	D	*Poor*
≤ 51	F	*Worst Imaginable*

Sumber: (Utami dkk., 2025)

Keterangan:

a) *Grade* menunjukkan tingkat kualitas usability sistem.
b) *Adjective* Rating merupakan penilaian kualitatif terhadap sistem berdasarkan persepsi pengguna.
c) *Acceptability* menunjukkan tingkat penerimaan sistem oleh pengguna.

Rumus Perhitungan *System Usability Scale* (SUS)

Perhitungan skor SUS dilakukan berdasarkan jawaban responden terhadap 10 pernyataan dengan ketentuan sebagai berikut:

1. Untuk pernyataan ganjil (1, 3, 5, 7, dan 9):

$$S_i = X_i - 1$$

2. Untuk pernyataan genap (2, 4, 6, 8, dan 10):

$$S_i = 5 - X_i$$

3. Skor Per Responden dengan rumus:

$$\text{Skor SUS} = \left(\sum_{i=1}^{10} S_i\right) \times 2{,}5$$

4. Skor akhir SUS dihitung dengan Rumus

$$\bar{x} = \frac{\sum x}{n} \quad \text{Persamaan 2.1}$$

Keterangan:

X_i= nilai jawaban responden

S_i= skor hasil konversi

$i = 1$ = urutan pertanyaan *SUS*

n= jumlah responden

Skor SUS berada pada rentang 0–100, di mana skor yang lebih tinggi menunjukkan tingkat usability sistem yang semakin baik.

Pertanyaan *System Usability Scale (SUS)*

Responden diminta untuk memberikan penilaian terhadap setiap pernyataan berikut berdasarkan pengalaman menggunakan sistem, dengan skala penilaian sebagai berikut:

1 = Sangat Tidak Setuju

2 = Tidak Setuju

3 = Netral

4 = Setuju

5 = Sangat Setuju

Tabel 2. 8 Kuesioner *System Usability Scale (SUS)*

No	Pernyataan Pengujian Beta Testing	Skala Penilaian (1–5)
1	Saya berpikir bahwa saya akan sering menggunakan sistem ini.	1 2 3 4 5
2	Saya merasa sistem ini terlalu rumit untuk digunakan.	1 2 3 4 5
3	Saya merasa sistem ini mudah digunakan.	1 2 3 4 5
4	Saya merasa membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini.	1 2 3 4 5
5	Saya merasa berbagai fungsi dalam sistem ini terintegrasi dengan baik.	1 2 3 4 5
6	Saya merasa terdapat terlalu banyak ketidakkonsistenan dalam sistem ini.	1 2 3 4 5
7	Saya membayangkan sebagian besar orang akan dapat mempelajari sistem ini dengan sangat cepat.	1 2 3 4 5
8	Saya merasa sistem ini sangat merepotkan untuk digunakan.	1 2 3 4 5
9	Saya merasa percaya diri saat menggunakan sistem ini.	1 2 3 4 5
10	Saya merasa perlu mempelajari banyak hal terlebih dahulu sebelum dapat menggunakan sistem ini.	1 2 3 4 5