BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

2.1.1 Rancang Bangun

Rancang bangun bertujuan untuk mengubah kebutuhan pengguna yang telah dianalisis menjadi bentuk desain teknis yang dapat diwujudkan dalam sistem. Tahapan ini mencakup berbagai aktivitas seperti perancangan arsitektur sistem, basis data, antarmuka pengguna, serta komponen-komponen sistem yang saling terhubung. Rancang bangun yang baik akan menjadi fondasi yang kuat dalam menghasilkan perangkat lunak yang berkualitas dan sesuai dengan kebutuhan pengguna. Dalam proses ini, terdapat beberapa pendekatan yang biasa digunakan, seperti pendekatan terstruktur dan pendekatan berorientasi objek. Pendekatan terstruktur dilakukan dengan metode yang logis dan sistematis, di mana sistem dipecah menjadi modul-modul kecil yang saling berkaitan. Setiap bagian sistem dirancang secara bertahap dengan fokus pada alur data, kontrol, serta struktur hierarki yang terorganisir. Beberapa alat bantu yang umum digunakan dalam pendekatan ini mencakup diagram alir data (Data Flow Diagram), diagram struktur program, dan tabel keputusan. Tujuan dari pendekatan ini adalah menghasilkan sistem yang mudah dianalisis, diuji, serta dipelihara karena dokumentasi yang lengkap dan standar. Pendekatan ini cocok digunakan untuk sistem dengan alur kerja yang linier dan kebutuhan yang stabil, seperti sistem inventaris atau akuntansi.

Sebaliknya, pendekatan berorientasi objek mendeskripsikan sistem berdasarkan objek nyata yang terdapat dalam proses bisnis. Setiap objek memiliki atribut serta metode yang merepresentasikan data dan perilakunya. Sistem dipandang sebagai kumpulan objek yang saling berinteraksi untuk menjalankan proses bisnis. Keunggulan dari pendekatan ini terletak pada fleksibilitas dan skalabilitasnya, sebab setiap objek bersifat independen dan dapat dikembangkan secara modular. Untuk membantu visualisasi, pendekatan ini menggunakan alat bantu seperti Unified Modeling Language (UML), yang mencakup diagram kelas

(class diagram), diagram use case, dan diagram aktivitas, guna mempermudah komunikasi antar pihak yang berkepentingan. Pendekatan ini sangat relevan untuk pengembangan perangkat lunak modern berbasis web dan mobile karena mampu beradaptasi terhadap perubahan kebutuhan pengguna secara dinamis (Pressman & Maxim, 2019).

2.1.1.1 System Development Life Cycle (SDLC)

Rancang bangun adalah tahap krusial dalam pengembangan sistem informasi yang bertujuan untuk menerjemahkan kebutuhan sistem ke dalam bentuk desain teknis yang dapat diimplementasikan. Menurut Rukmana et al. (n.d.), rancang bangun melibatkan pada perancangan elemen-elemen sistem yang mendukung fungsionalitas, efisiensi, serta keandalan sistem secara keseluruhan. Tahapan ini melibatkan pengembangan sistem yang lebih terstruktur dan fleksibel, memungkinkan penyesuaian dan perbaikan yang cepat.

Dalam model pengembangan sistem informasi, SDLC (System Development Life Cycle) menjadi acuan utama dalam mengadopsi aktivitas-aktivitas yang digunakan, yaitu perencanaan, analisis, desain, dan implementasi. Model Waterfall, sebagai salah satu pendekatan metode pengembangan sistem informasi yang awal, menunjukkan siklus hidup yang linier dan sekuensial (berurutan), di mana setiap tahap aksi dilaksanakan secara berurutan. Siklus ini menunjukkan bahwa setiap tahap pekerjaan harus selesai sebelum tahap berikutnya dimulai, menunjukkan kebutuhan proses yang linier dan terstruktur. Model ini cocok memberikan panduan yang jelas dan terstruktur kepada pengembang sistem dalam menghasilkan perangkat lunak atau aplikasi yang berkualitas dan sesuai dengan kebutuhan pengguna (Rukmana et al., n.d.).

Dalam penelusuran tugas akhir ini, konsep-konsep tersebut dijelaskan dengan lebih rinci dan di bawah penjelasan mengenai SDLC dan model pengembangan sistem informasi lainnya yang sering digunakan. Dengan demikian pemahaman konsep tersebut, diharapkan akan lebih mudah dalam

mengimplementasikannya ke dalam proyek pengembangan sistem informasi yang akan dilakukan.

1. **Perencanaan**

   Perencanaan adalah tahap awal dalam siklus pengembangan sistem informasi. Tahap ini melibatkan identifikasi kebutuhan bisnis dan penentuan ruang lingkup proyek. Dalam perencanaan, dilakukan studi kelayakan (teknis, ekonomis, hukum, operasional), pembentuan tim proyek, serta penyusunan rencana kerja dan anggaran. Hasil utama dari tahap ini adalah dokumen kebutuhan sistem dan rencana proyek. Tahap perencanaan ini penting karena menjadi fondasi untuk semua tahap berikutnya dalam siklus pengembangan.

2. **Analisis**

   Analisis adalah tahap kedua dalam siklus pengembangan sistem informasi. Tahap ini berfokus pada penggalian kebutuhan pengguna melalui observasi, wawancara, atau kuesioner. Hasil dari analisis ini berupa spesifikasi kebutuhan sistem, baik fungsional maupun non-fungsional, serta model sistem berjalan (as-is) dan sistem yang direncanakan (to-be). Analisis ini penting karena membantu dalam mengidentifikasi kebutuhan-kebutuhan yang sebenarnya diperlukan oleh pengguna dan menentukankannya dalam sistem yang akan dibangun.

3. **Desain**

   Desain adalah tahap ketiga dalam siklus pengembangan sistem informasi. Tahap ini melibatkan kebutuhan-kebutuhan yang telah dianalisis ke dalam bentuk desain teknis. Cakupan desain meliputus desain arsitektur sistem, basis data, antarmuka pengguna, dan keamanan. Hasil dari tahap ini adalah blueprint sistem yang siap dikembangkangkan. Desain ini penting karena menjadi panduan utama keberhasilan implementasi sistem, memastikan struktur dan fungsi yang akan dibangun nanti proyek.

4. **Implementasi dan Perawatan**

   Implementasi dan Perawatan adalah tahap keempat dalam siklus pengembangan sistem informasi. Tahap ini mencakupkan proses pengkodean, integrasi sistem, pengujian, dan instalasi sistem. Selain itu, pelatihan pengguna dan migrasi data juga dilakukan untuk memastikan sistem dapat digunakan

secara efektif. Implementasi ini adalah tahap di mana desain teknis yang telah dibuat diterapkan ke dalam lingkungan operasional, sehingga sistem dapat berjalan sesuai dengan kebutuhan dan alur kerja yang diharapkan.

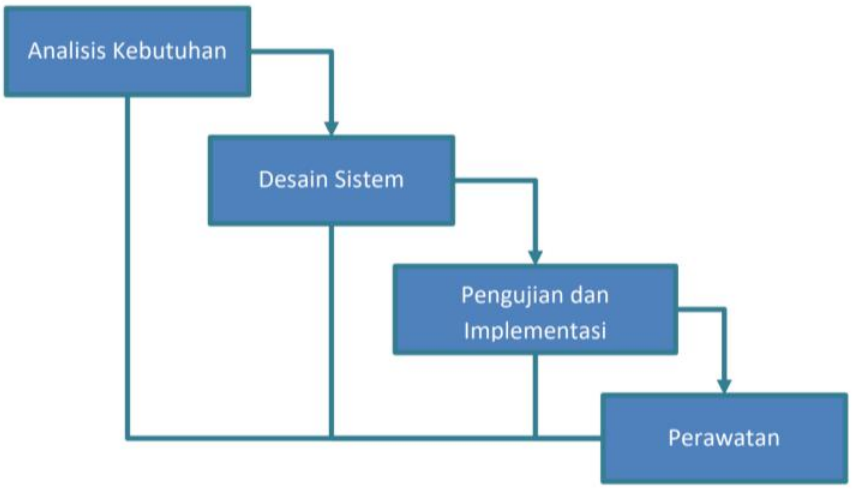
Gambar 2.1 Gambar aktifitas-aktifitas pada SDLC

Dalam implementasi SDLC, pemilihan metodologi pengembangan sangat memengaruhi keberhasilan proyek sistem informasi karena menurut Rukmana et al. (n.d.), metodologi sebagai ilmu (cara teratur) yang digunakan untuk melaksanakan pekerjaan pengembangan agar tercapai sistem informasi sesuai yang dikehendaki. Setiap metodologi menawarkan pendekatan yang berbeda dalam menjalankan tahapan-tahapan SDLC, mulai dari perencanaan hingga pemeliharaan. Beberapa model bekerja secara linier, sementara lainnya lebih fleksibel dan iteratif. Pemilihan metodologi didasarkan pada kompleksitas proyek, sumber daya yang tersedia, serta tingkat keterlibatan pengguna. Berikut ini adalah beberapa metodologi pengembangan sistem yang umum digunakan dalam kerangka SDLC:

1. **Waterfall**

   Metodologi Waterfall merupakan pendekatan klasik dan linier dalam pengembangan sistem informasi yang memperkenalkan proses secara bertahap dan terstruktur. Tahapan-tahapan dilakukan secara berurutan, dimulai dari identifikasi kebutuhan, perancangan sistem, proses implementasi, dilanjutkan dengan pengujian, hingga tahap pemeliharaan sistem. Dalam model ini, setiap fase wajib diselesaikan sepenuhnya sebelum melangkah ke tahap selanjutnya, dan tidak terdapat proses yang dijalankan secara bersamaan antar fase. Model ini dikenal memiliki struktur yang jelas dan mudah dipahami, terutama untuk jenis proyek yang kebutuhannya telah ditetapkan sejak awal serta tidak mengalami banyak perubahan. Oleh karena itu, pendekatan Waterfall sangat sesuai untuk proyek-proyek yang dimulai dari permintaan formal dan memiliki dokumentasi kebutuhan yang terdefinisi dengan baik. Adanya proses verifikasi

di setiap tahapan membantu mendeteksi kesalahan lebih awal sehingga dapat mengurangi potensi kesalahan besar di akhir pengembangan. Kejelasan dalam aktivitas serta keluaran dari setiap tahapan menjadikan model ini ideal untuk proyek skala besar, seperti pengembangan sistem di bidang pemerintahan, kesehatan, maupun industri manufaktur.



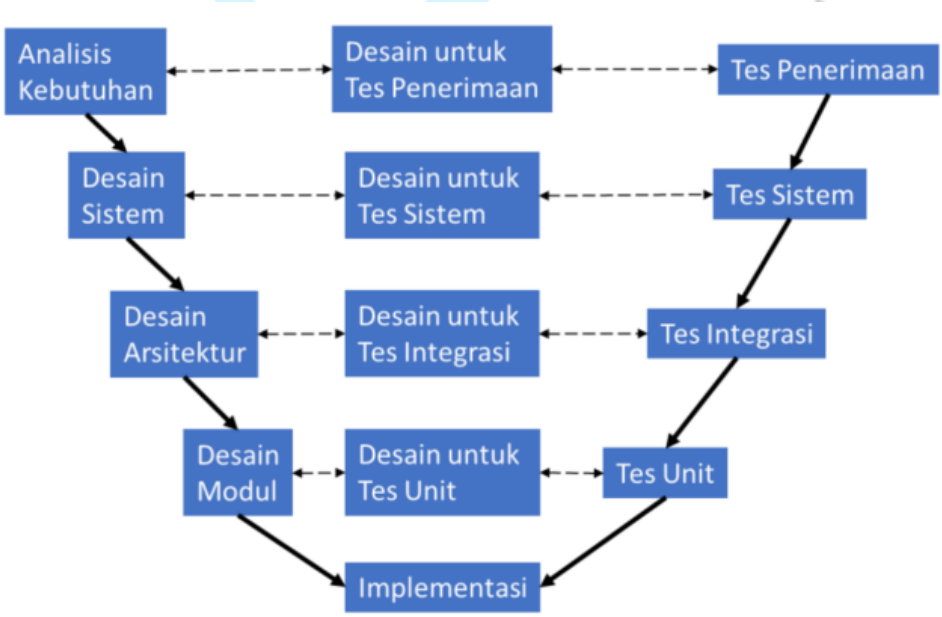
Gambar 2.2 Model Waterfall

Menurut Arief Yanto Rukmana dkk. (2023), model Waterfall menawarkan beberapa keunggulan, seperti kemudahan dalam penjadwalan proyek, pendekatan yang terstruktur, serta tahapan dan aktivitas yang telah terdefinisi dengan baik. Namun, model ini juga memiliki sejumlah kelemahan, di antaranya adalah sifatnya yang kaku dan tidak fleksibel terhadap perubahan kebutuhan, serta sulitnya untuk kembali ke tahap sebelumnya setelah suatu tahap diselesaikan. Selain itu, model ini memerlukan perencanaan yang sangat rinci di awal dan penyelesaian proyek yang relatif lebih lama serta mahal jika dibandingkan dengan pendekatan lainnya. Model Waterfall tidak cocok untuk proyek yang membutuhkan adaptasi cepat terhadap perubahan atau memiliki kebutuhan yang masih dinamis. Oleh karena itu, dalam konteks modern yang seringkali memerlukan fleksibilitas tinggi, pendekatan Waterfall lebih sering digunakan untuk proyek dengan ruang lingkup yang tetap dan kebutuhan yang sudah pasti (Arief Yanto Rukmana, Rahman, R., Hery Afriyadi, Dikwan

Moeis, Setiawan, Z., Nur Subchan, Magdalena, L., Marcello Singadji, Augury El Rayeb, & Adi, T., 2023).

2. Model V

Model V merupakan modifikasi dari model Waterfall yang memberikan penekanan khusus pada aktivitas verifikasi dan validasi di setiap tahapan pengembangan sistem. Meskipun pendekatan ini masih mengadopsi urutan proses seperti pada model Waterfall, Model V membaginya menjadi dua sisi utama, yakni verifikasi dan validasi, yang secara visual menyerupai huruf "V". Pada bagian kiri diagram, tahapan pengembangan dimulai dari identifikasi kebutuhan, perancangan sistem, perancangan arsitektur, dan desain modul, lalu dilanjutkan hingga proses implementasi. Sementara itu, pada bagian kanan terdapat tahapan validasi yang meliputi pengujian unit, integrasi, sistem secara keseluruhan, serta uji penerimaan akhir. Dengan pendekatan ini, setiap proses pada fase verifikasi diimbangi oleh validasi yang ditujukan untuk memastikan kembali bahwa sistem yang dibangun telah mencerminkan kebutuhan awal secara akurat. Pendekatan ini memungkinkan proses pengujian dilakukan lebih awal dan berkelanjutan sepanjang proses pengembangan, sehingga mampu meminimalkan kesalahan yang berpotensi muncul di tahap akhir.

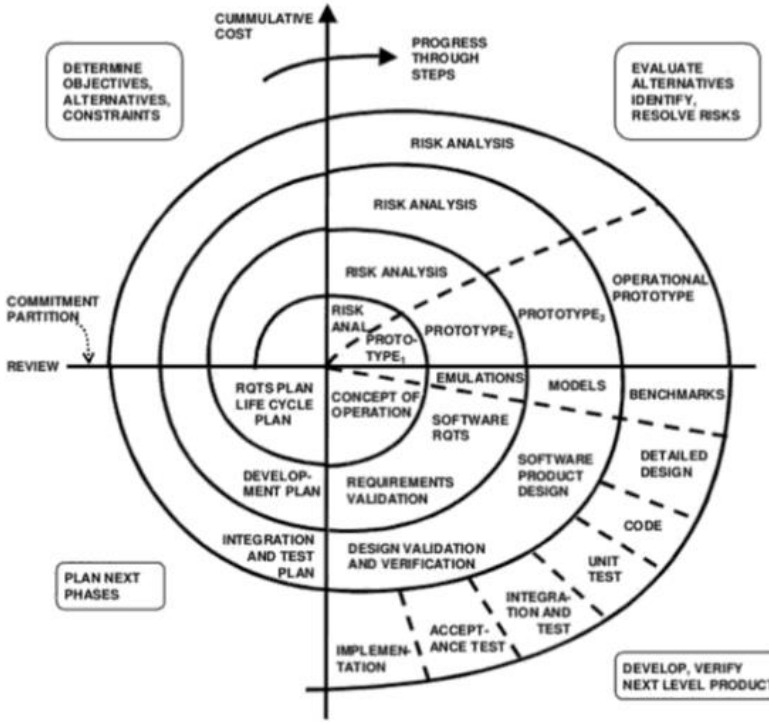


Gambar 2.3 Model V

Menurut Arief Yanto Rukmana dkk. (2023), keunggulan utama dari model V adalah strukturnya yang ringkas dan mudah digunakan, serta memiliki hasil yang sangat spesifik di setiap fase. Model ini juga memungkinkan perencanaan pengujian dilakukan sejak dini sehingga meningkatkan peluang keberhasilan proyek dibandingkan dengan model Waterfall. Selain itu, apabila kebutuhan (requirement) telah didefinisikan secara jelas dan mudah dipahami, model V sangat efektif untuk digunakan. Namun, sama seperti model Waterfall, model V juga memiliki kelemahan utama yaitu kurangnya fleksibilitas dalam menghadapi perubahan kebutuhan, serta tingginya effort dan biaya jika terjadi penyesuaian ruang lingkup proyek. Model ini juga tidak memberikan ruang untuk pembuatan prototype awal, karena pengembangan langsung dilakukan pada fase implementasi. Oleh karena itu, model V sangat tepat digunakan dalam proyek berskala besar dengan kebutuhan yang sudah terdefinisi secara ketat sejak awal (Arief Yanto Rukmana, Rahman, R., Hery Afriyadi, Dikwan Moeis, Setiawan, Z., Nur Subchan, Magdalena, L., Marcello Singadji, Augury El Rayeb, & Adi, T., 2023).

3. **Spiral**

Model Spiral merupakan metode pengembangan sistem informasi yang menggabungkan keunggulan dari dua pendekatan populer sebelumnya, yaitu Waterfall dan Prototyping. Pendekatan ini didesain untuk mendukung proyek-proyek berskala besar yang kompleks dan memiliki tingkat risiko yang tinggi. Model spiral bekerja secara iteratif, dengan tahapan yang dilakukan secara berulang dalam bentuk spiral yang setiap siklusnya terdiri dari perencanaan, analisis risiko, pengembangan prototype, serta evaluasi hasil. Dalam setiap iterasi, risiko dianalisis dan diperkirakan sebelum pengambilan keputusan apakah proses bisa lanjut ke tahap berikutnya atau perlu perbaikan. Dengan demikian, model ini mendukung pendekatan top-down (perencanaan sistem dari atas ke bawah) dan bottom-up (pembangunan sistem dari bawah ke atas) secara bersamaan, sehingga proses pengembangan menjadi lebih adaptif terhadap perubahan dan lebih sistematis dalam menghadapi ketidakpastian.

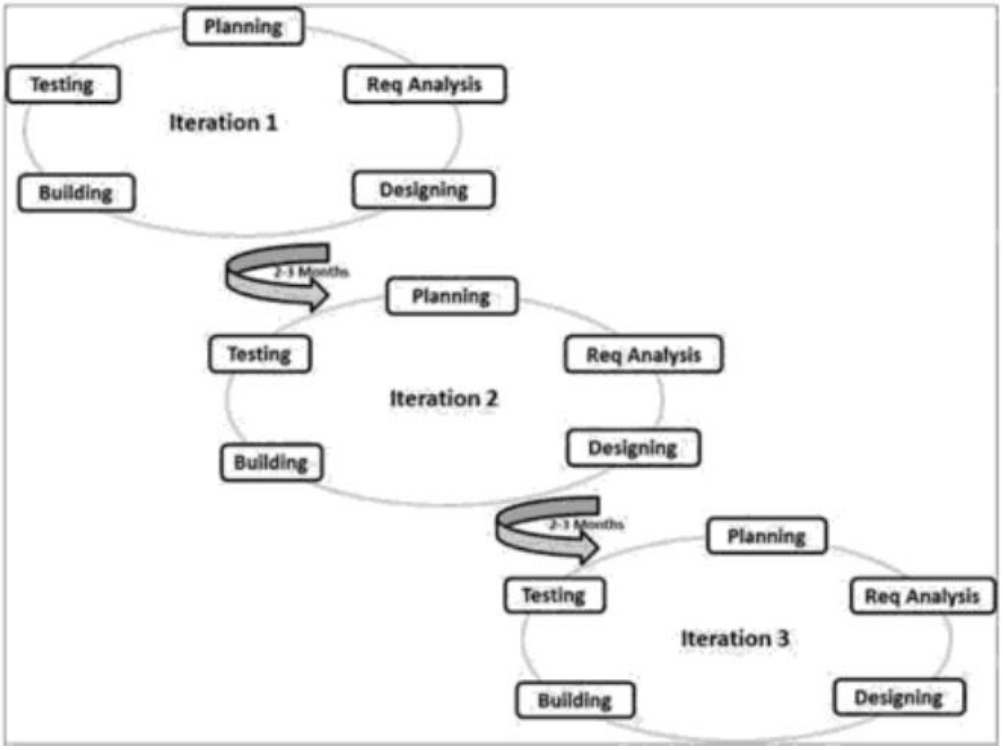


Gambar 2.4 Model Spiral

Menurut Arief Yanto Rukmana dkk. (2023), model spiral mengadopsi banyak fase yang serupa dengan model Waterfall, namun dibedakan dengan adanya fokus yang kuat pada evaluasi risiko dan pengembangan prototype secara berkala. Model ini sangat sesuai digunakan dalam proyek-proyek besar, mahal, dan kompleks karena memungkinkan estimasi biaya, waktu, dan kebutuhan menjadi lebih realistis seiring berjalannya proses. Keterlibatan pengembang sejak awal dan kemampuan untuk membagi proyek ke dalam fase-fase kecil memungkinkan tim lebih mudah mengelola risiko serta menyesuaikan sistem jika terjadi perubahan kebutuhan. Meski demikian, model spiral juga memiliki kekurangan, terutama dalam hal biaya dan waktu yang relatif tinggi untuk mencapai produk akhir. Selain itu, pendekatan ini menuntut keahlian khusus dalam evaluasi risiko dan pengelolaan iterasi proyek. Oleh karena itu, model spiral lebih cocok digunakan oleh tim pengembang yang sudah berpengalaman dalam proyek berskala besar dan membutuhkan kontrol ketat terhadap ketidakpastian.

4. Agile

Model Agile merupakan metode pengembangan perangkat lunak yang membagi proses pembangunan menjadi unit-unit kecil yang disebut iterasi. Setiap iterasi berlangsung dalam waktu singkat, umumnya antara satu hingga tiga minggu, dan melibatkan kolaborasi antara tim pengembang dengan perwakilan pengguna. Dalam setiap siklus iterasi dilakukan sejumlah aktivitas seperti perencanaan, identifikasi kebutuhan, perancangan, pengkodean, pengujian unit, hingga uji penerimaan (acceptance test). Pendekatan ini memungkinkan sistem dikembangkan secara bertahap dan dilakukan evaluasi secara terus-menerus, sehingga setiap fitur yang dihasilkan bisa segera diverifikasi oleh pengguna. Dengan cara ini, produk akhir menjadi lebih sesuai dengan kebutuhan pengguna serta mempercepat proses rilis ke pasar. Kekuatan utama dari model Agile terletak pada fleksibilitas dan kemampuannya dalam beradaptasi terhadap perubahan kebutuhan yang muncul selama proses berlangsung.



Gambar 2.5 Model Agile

Menurut Arief Yanto Rukmana dkk. (2023), model Agile sangat ideal untuk proyek yang membutuhkan interaksi intensif antara pengguna dan pengembang. Model ini memungkinkan peningkatan kualitas produk dan kepuasan pengguna karena setiap iterasi melibatkan umpan balik langsung dari pelanggan. Keunggulan dari pendekatan ini adalah komunikasi yang terus-menerus, kolaborasi aktif dengan pelanggan, serta kemampuan beradaptasi dengan cepat terhadap perubahan kebutuhan. Namun demikian, Agile juga memiliki kelemahan, antara lain membutuhkan tim dengan keterampilan khusus, kolaborasi tinggi untuk memahami kebutuhan pengguna, dan dokumentasi yang sering kali dilakukan di tahap selanjutnya sehingga berpotensi menimbulkan celah informasi. Selain itu, model ini juga dapat mengurangi usability dan skalabilitas pada beberapa komponen jika tidak dirancang dengan baik sejak awal (Arief Yanto Rukmana, Rahman, R., Hery Afriyadi, Dikwan Moeis, Setiawan, Z., Nur Subchan, Magdalena, L., Marcello Singadji, Augury El Rayeb, & Adi, T., 2023). Oleh karena itu, meskipun Agile sangat cocok untuk proyek dinamis dan inovatif, diperlukan kesiapan tim dan sistem yang matang untuk mengoptimalkan manfaatnya.

5. **Rapid Aplication Development (RAD)**

Rapid Application Development (RAD) adalah salah satu metode pengembangan perangkat lunak yang menitikberatkan pada kecepatan proses pengembangan serta keterlibatan aktif pengguna melalui pendekatan iteratif. Berbeda dengan pendekatan Waterfall yang bersifat linier, RAD memungkinkan sistem dikembangkan dalam waktu yang relatif singkat dengan membagi tahapan menjadi beberapa fase singkat yang bisa dijalankan secara bersamaan. Fase-fase utama dalam metode RAD mencakup tahap perencanaan kebutuhan (requirements planning), perancangan bersama pengguna (user design), pembangunan sistem (construction), dan tahap implementasi (cutover). Pada setiap iterasi, prototipe dibuat dan segera diuji oleh pengguna untuk memastikan sistem telah sesuai dengan kebutuhan aktual. Pendekatan ini menawarkan berbagai keuntungan seperti percepatan waktu peluncuran, peningkatan sinergi antara pengembang dan pengguna, serta kemampuan

beradaptasi terhadap perubahan kebutuhan selama pengembangan berlangsung (Pereira, 2022).

Dalam implementasinya, RAD sangat efektif digunakan dalam proyek yang menuntut waktu pengembangan yang singkat serta interaksi intensif dengan pengguna akhir. Penggunaan platform low-code seperti OutSystems bahkan semakin memperkuat keunggulan RAD dengan menyediakan lingkungan visual yang mempercepat proses desain dan implementasi. Pereira (2022) menekankan bahwa RAD mampu meminimalkan risiko kegagalan proyek karena setiap hasil pengembangan langsung dapat divalidasi oleh pengguna melalui prototipe interaktif. Hal ini memungkinkan pengembang untuk segera menyesuaikan sistem apabila ditemukan ketidaksesuaian. Namun, efektivitas RAD sangat bergantung pada keterlibatan aktif pengguna dan kemampuan tim dalam mengelola iterasi dengan cepat dan efisien. Oleh karena itu, RAD lebih tepat diterapkan pada proyek-proyek berskala kecil hingga menengah yang memerlukan solusi cepat, responsif, dan dapat berkembang seiring perubahan kebutuhan.

2.1.1.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan bahasa pemodelan visual standar yang digunakan secara luas dalam rekayasa perangkat lunak untuk mendeskripsikan, merancang, dan mendokumentasikan sistem perangkat lunak berbasis objek. UML hadir sebagai alat bantu utama dalam menyampaikan struktur dan perilaku sistem melalui representasi grafis yang jelas dan terstandar. Dengan UML, pengembang dapat membuat model sistem yang mudah dipahami oleh berbagai pemangku kepentingan, termasuk analis sistem, perancang perangkat lunak, hingga pengguna non-teknis. UML juga memberikan kemudahan dalam proses dokumentasi karena dapat mencerminkan kondisi sistem secara konsisten sepanjang siklus hidup perangkat lunak. Selain itu, penggunaan UML dapat meningkatkan efisiensi dalam proses desain karena setiap diagram memiliki fungsi yang spesifik, seperti memodelkan struktur kelas, interaksi antar objek, atau alur aktivitas sistem. Keunggulan UML terletak pada fleksibilitasnya yang tidak

tergantung pada metode pengembangan tertentu, sehingga dapat digunakan dalam pendekatan Waterfall, Agile, maupun pendekatan lainnya. Berbagai jenis diagram yang tersedia dalam UML menjadikan bahasa ini sangat kaya dan mampu mencerminkan berbagai aspek penting dari sistem perangkat lunak. Oleh karena itu, UML adalah bagian krusial dalam tahapan rekayasa perangkat lunak modern yang bertujuan menghasilkan sistem yang terdokumentasi, terstruktur, dan mudah dipelihara (Siregar, 2023).

UML menyediakan beragam jenis diagram yang dirancang untuk menggambarkan sistem dari berbagai aspek, baik dari sisi struktur maupun perilaku. Diagram-diagram tidak sebatas digunakan untuk proses analisis dan perancangan sistem, tetapi juga berperan penting dalam komunikasi antar anggota tim dan dokumentasi teknis sistem. Secara umum, diagram UML dibagi menjadi dua kelompok besar, yakni diagram struktur serta diagram perilaku. Pembagian ini memungkinkan pengembang untuk menggambarkan sistem secara komprehensif—mulai dari bagaimana elemen sistem disusun secara statis, hingga bagaimana mereka berinteraksi dalam menjalankan fungsinya secara dinamis.

a. Struktural Diagrams

Struktural Diagrams atau diagram struktur menggambarkan komponen-komponen sistem yang bersifat statis, atau tidak mengalami perubahan selama sistem beroperasi. Fokus utama dari jenis diagram ini adalah pada representasi elemen-elemen internal sistem seperti kelas, entitas, bagian-bagian sistem, serta relasi antar elemen tersebut. Diagram semacam ini sangat membantu dalam memperlihatkan bagaimana sistem disusun, diklasifikasikan, dan diorganisasikan. Beberapa contoh diagram struktur adalah:

- Class Diagram – mewakili struktur kelas dalam sistem, mencakup atribut, fungsi (method), dan keterkaitan antar kelas.
- Object Diagram – menyajikan instansiasi dari class diagram pada kondisi tertentu dalam waktu tertentu.
- Component Diagram – menunjukkan bagaimana komponen perangkat lunak saling berinteraksi.

- Deployment Diagram – menggambarkan bagaimana komponen perangkat lunak didistribusikan pada infrastruktur fisik (server, perangkat keras, dll.).

b. Behavioral Diagrams

Behavioral diagrams atau diagram perilaku menggambarkan bagaimana sistem berperilaku atau merespons terhadap aksi-aksi tertentu, serta bagaimana objek dalam sistem saling berinteraksi selama proses berlangsung. Diagram jenis ini digunakan untuk memahami logika proses, urutan kejadian, serta komunikasi antar komponen selama sistem berjalan. Beberapa contoh diagram perilaku adalah:

- Use Case Diagram – menjelaskan keterkaitan antara aktor (baik pengguna maupun sistem eksternal) dengan fungsi-fungsi (use case) dalam sistem.
- Sequence Diagram – menunjukkan urutan komunikasi antara objek seiring waktu untuk menyelesaikan aktivitas tertentu.
- Activity Diagram – mewakili alur kegiatan atau proses operasional dari suatu fungsi maupun aktivitas di dalam sistem.
- State Machine Diagram – menunjukkan perubahan status suatu objek berdasarkan peristiwa yang terjadi.

Dengan memahami perbedaan dan fungsi masing-masing kategori diagram, pengembang sistem dapat memilih diagram yang paling relevan sesuai kebutuhan tahap pengembangan yang sedang dilakukan. Di antara banyaknya jenis diagram yang disediakan UML, beberapa jenis diagram dianggap paling umum dan penting untuk dianalisis dan digunakan dalam praktik, terutama pada tahap awal desain sistem.

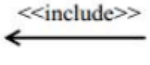
a. Use Case Diagram

Use Case Diagram merupakan salah satu jenis diagram perilaku dalam UML yang digunakan untuk menggambarkan interaksi antara sistem dan aktor yang berperan di dalamnya. Diagram ini memvisualisasikan

bagaimana pengguna (aktor) berinteraksi dengan sistem melalui fungsi-fungsi utama (use case) yang disediakan oleh sistem tersebut. Tujuan utama dari diagram ini adalah untuk mengidentifikasi kebutuhan fungsional dari sistem berdasarkan perspektif pengguna, sehingga menjadi fondasi dalam menyusun desain sistem yang relevan dan sesuai dengan ekspektasi pengguna akhir.

Menurut Siregar (2023), Use Case Diagram merupakan alat bantu yang sangat penting pada tahap awal analisis sistem karena memberikan gambaran menyeluruh tentang fungsionalitas yang diharapkan dari sistem. Diagram ini bersifat sederhana, tetapi sangat efektif dalam menjelaskan batasan sistem (system boundary), aktor yang terlibat (baik internal maupun eksternal), serta hubungan antara aktor dengan berbagai skenario penggunaan. Setiap use case mewakili satu fungsi utama sistem, seperti "Login", "Cetak Laporan", atau "Kelola Data", yang dihubungkan dengan aktor melalui asosiasi. Selain itu, Use Case Diagram juga dapat memuat relasi seperti include (fungsi umum yang dipanggil oleh beberapa use case), dan extend (fungsi tambahan yang bersifat opsional tergantung kondisi tertentu), yang membantu menyusun skenario fungsional dengan lebih rinci dan fleksibel.

Tabel 2.1 **Tabel Komponen Utama Use Case**

Simbol	Keterangan
	Merepresentasikan peran manusia, sistem eksternal, atau perangkat lain yang berinteraksi dengan sistem melalui use case.
	Menggambarkan fungsi atau layanan yang disediakan oleh sistem kepada aktor.
	Menghubungkan aktor dengan use case untuk menunjukkan adanya interaksi atau komunikasi di antara keduanya.
	Menandakan hubungan pewarisan antara aktor, di mana aktor anak mewarisi tanggung jawab dan perilaku dari aktor induk.
	Menggambarkan bahwa sebuah use case selalu menyertakan (mengikutsertakan) fungsionalitas dari use case lain sebagai bagian integral dari proses.

<<extend>>	Menunjukkan bahwa sebuah use case memiliki fungsionalitas tambahan yang dapat dijalankan jika kondisi tertentu terpenuhi (opsional).

Use Case Diagram biasanya digunakan dalam dokumentasi kebutuhan sistem (Software Requirements Specification) sebagai salah satu referensi awal dalam membangun sistem perangkat lunak. Diagram ini juga sering digunakan dalam diskusi antara pengembang dan pengguna untuk menyepakati fungsi-fungsi utama yang akan dikembangkan, sebelum melangkah ke tahap desain teknis lebih lanjut. Karena sifatnya yang abstrak dan tidak teknis, Use Case Diagram dapat dipahami oleh berbagai pihak, termasuk pihak manajerial atau pemilik sistem yang tidak memiliki latar belakang teknis.

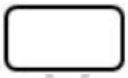
b. Activity Diagram

Activity Diagram adalah salah satu jenis diagram perilaku dalam UML yang berfungsi untuk memodelkan alur aktivitas atau logika proses bisnis yang terjadi dalam suatu sistem. Diagram ini merepresentasikan bagaimana aliran kerja dimulai, bagaimana keputusan dibuat, serta bagaimana aktivitas dilakukan secara sekuensial maupun paralel. Dengan bentuk yang menyerupai flowchart, Activity Diagram menjadi sarana yang sangat efektif untuk memvisualisasikan proses kompleks menjadi lebih terstruktur dan mudah dipahami. Hal ini membuatnya sangat berguna pada tahap analisis maupun desain sistem, khususnya ketika merancang fitur yang melibatkan banyak tahapan, kondisi logis, dan pelaku yang berbeda. Elemen-elemen seperti titik awal (initial node), keputusan (decision), penggabungan alur (merge), aktivitas paralel (fork dan join), hingga titik akhir (final node) membuat diagram ini mampu menangkap berbagai skenario proses dengan baik.

Penggunaan Activity Diagram sangat luas, mulai dari mendeskripsikan prosedur operasional organisasi, alur transaksi sistem, hingga logika internal dari sebuah modul aplikasi. Diagram ini membantu

analis dan pengembang memahami bagaimana suatu proses harus dijalankan secara sistematis, sekaligus memfasilitasi diskusi dengan pengguna non-teknis karena sifat visualnya yang intuitif. Menurut Siregar (2023), Activity Diagram tidak hanya berguna untuk mendeskripsikan alur kerja, tetapi juga berperan penting dalam mengidentifikasi potensi konflik logika atau inefisiensi proses sejak tahap awal pengembangan. Oleh karena itu, diagram ini sering menjadi jembatan penting antara kebutuhan fungsional pengguna dan rancangan teknis yang akan diterapkan oleh tim pengembang.

Tabel 2.2 **Tabel Komponen Activity Diagram**

Simbol	Nama	Keterangan
	Status Awal	Menandakan titik awal dari suatu proses dalam diagram aktivitas. Setiap diagram aktivitas harus dimulai dari status ini.
	Aktivitas	Merepresentasikan tindakan atau proses yang dilakukan oleh sistem atau pengguna. Biasanya diawali dengan kata kerja seperti "memproses".
	Percabangan/Decision	Menunjukkan titik keputusan dalam proses, di mana terdapat lebih dari satu kemungkinan jalannya aktivitas (cabang).
	Penggabungan/Join	Digunakan untuk menyatukan beberapa alur aktivitas menjadi satu jalur tunggal dalam proses.
	Status Akhir	Menunjukkan titik akhir dari alur proses. Diagram aktivitas akan berakhir pada simbol ini.
	Swimlane	Digunakan untuk memisahkan tanggung jawab antar bagian atau aktor dalam proses. Mewakili siapa yang melakukan aktivitas tertentu.

c. Sequence Diagram

Sequence Diagram merupakan salah satu bentuk diagram perilaku dalam UML (Unified Modeling Language) yang digunakan untuk menggambarkan bagaimana objek-objek dalam sistem berinteraksi berdasarkan urutan waktu secara kronologis. Diagram ini memperlihatkan cara objek saling bertukar informasi melalui pengiriman pesan dalam

rangka menyelesaikan suatu aktivitas tertentu, serta memperjelas aliran komunikasi antar objek secara runtut.

Unsur-unsur utama dalam Sequence Diagram meliputi lifeline (garis vertikal yang merepresentasikan eksistensi objek selama waktu tertentu), activation bar (blok vertikal yang menunjukkan aktivitas atau eksekusi suatu operasi), serta message (panah horizontal yang menunjukkan pesan yang dikirim antar objek). Diagram ini biasanya disusun untuk menggambarkan satu skenario fungsional secara spesifik, seperti proses autentikasi pengguna, transaksi pemesanan, atau pembuatan laporan.

Sequence Diagram efektif digunakan untuk mengidentifikasi urutan komunikasi dan potensi kesalahan logika dalam interaksi objek sebelum sistem diimplementasikan. Diagram ini juga membantu dalam dokumentasi kebutuhan sistem, sebagai referensi teknis bagi tim pengembang, serta menjadi alat bantu komunikasi antar stakeholder dengan berbagai latar belakang. Visualisasi yang ditawarkan Sequence Diagram mendukung verifikasi dan validasi sistem sejak tahap perancangan awal hingga pengembangan berlanjut (Siregar, 2023).

d. Class Diagram

Class Diagram merupakan salah satu tipe diagram struktur dalam Unified Modeling Language (UML) yang berfungsi untuk merepresentasikan struktur statis sistem berbasis objek. Diagram ini menunjukkan entitas-entitas kelas dalam sistem beserta atribut, fungsi (method), serta keterkaitan antar kelas, sehingga menjadi dasar penting dalam perancangan sistem informasi berbasis objek. Setiap kelas dalam diagram ini ditampilkan dalam bentuk persegi panjang yang terdiri dari tiga bagian utama, yaitu bagian atas mencantumkan nama kelas, bagian tengah menampilkan atribut, dan bagian bawah mencakup metode atau operasi yang dimiliki oleh kelas tersebut. Selain mendeskripsikan isi dari setiap

kelas, diagram ini juga memperlihatkan relasi antar kelas seperti asosiasi, agregasi, komposisi, maupun generalisasi.

Class Diagram sangat bermanfaat pada tahap perancangan dan analisis sistem karena mampu menyusun struktur sistem menjadi lebih sistematis serta mudah dipahami. Diagram ini mendukung pengembang dalam mengenali objek-objek yang ada di dalam sistem beserta bagaimana entitas-entitas tersebut berinteraksi dan saling bergantung satu dengan lainnya. Dengan demikian, Class Diagram menjadi salah satu alat penting untuk menerjemahkan kebutuhan pengguna menjadi struktur sistem yang nantinya dapat diimplementasikan menjadi kode program (Siregar, 2023).

Tabel 2.3 **Tabel Komponen Class Diagram**

NO	Gambar	Nama	Keterangan
1		Generalization	Relasi di mana objek anak mewarisi sifat dan struktur dari objek induknya.
2		N-ary Association	Digunakan untuk mencegah asosiasi kompleks yang melibatkan lebih dari dua objek.
3		Class	Sekumpulan objek dengan atribut dan operasi yang serupa.
4		Collaboration	Menjelaskan interaksi antar objek dalam sistem untuk mencapai hasil tertentu.
5		Realization	Proses pelaksanaan nyata dari operasi yang dilakukan suatu objek.
6		Dependency	Relasi di mana perubahan pada satu elemen mempengaruhi elemen lain yang bergantung padanya.
7		Association	Koneksi atau hubungan antara dua objek dalam sistem.

Selain itu, Class Diagram tidak hanya relevan bagi pengembang, namun juga digunakan oleh analis dan perancang basis data untuk membuat model sistem secara menyeluruh. Diagram ini membantu memastikan bahwa struktur sistem telah dirancang secara logis, efisien, dan mendukung kebutuhan fungsional yang telah ditentukan sebelumnya.

Unified Modeling Language (UML) merupakan bahasa pemodelan visual standar yang digunakan secara luas dalam rekayasa perangkat lunak untuk mendeskripsikan, merancang, dan mendokumentasikan sistem perangkat lunak berbasis objek. UML hadir sebagai alat bantu utama dalam menyampaikan struktur dan perilaku sistem melalui representasi grafis yang jelas dan terstandar. Dengan UML, pengembang dapat membuat model sistem yang mudah dipahami oleh berbagai pemangku kepentingan, termasuk analis sistem, perancang perangkat lunak, hingga pengguna non-teknis. UML juga memberikan kemudahan dalam proses dokumentasi karena dapat mencerminkan kondisi sistem secara konsisten sepanjang siklus hidup perangkat lunak. Selain itu, penggunaan UML dapat meningkatkan efisiensi dalam proses desain karena setiap diagram memiliki fungsi yang spesifik, seperti memodelkan struktur kelas, interaksi antar objek, atau alur aktivitas sistem. Keunggulan UML terletak pada fleksibilitasnya yang tidak tergantung pada metode pengembangan tertentu, sehingga dapat digunakan dalam pendekatan Waterfall, Agile, maupun pendekatan lainnya. Berbagai jenis diagram yang tersedia dalam UML menjadikan bahasa ini sangat kaya dan mampu mencerminkan berbagai aspek penting dari sistem perangkat lunak. Oleh karena itu, UML merupakan salah satu komponen penting dalam proses rekayasa perangkat lunak modern yang bertujuan untuk menghasilkan sistem yang terdokumentasi, terstruktur, dan mudah dipelihara (Siregar, 2023).

2.1.2 Sistem Informasi

Sistem informasi adalah suatu sistem terintegrasi yang terdiri atas komponen-komponen yang saling berinteraksi untuk mengumpulkan, mengolah, menyimpan, dan mendistribusikan data menjadi informasi yang berguna. Informasi tersebut digunakan untuk mendukung proses pengambilan keputusan, pengendalian, koordinasi, serta analisis dalam suatu organisasi. Dalam konteks bisnis dan organisasi modern, sistem informasi tidak hanya berfungsi sebagai alat bantu administratif, tetapi juga menjadi fondasi strategis untuk menciptakan efisiensi dan meningkatkan daya saing organisasi (Priyanto, 2021).

2.1.2.1 Unsur - Unsur Sistem Informasi

Sistem informasi memiliki lima komponen utama yang saling berhubungan dan membentuk suatu sistem yang utuh:

1. Perangkat keras (hardware): meliputi komputer, server, perangkat input-output, serta perangkat jaringan.
2. Perangkat lunak (software): termasuk aplikasi sistem dan aplikasi khusus yang digunakan untuk menjalankan fungsi-fungsi tertentu.
3. Basis data (database): berisi data terstruktur yang dapat diakses, diubah, dan diolah sesuai kebutuhan pengguna.
4. Prosedur dan kebijakan: aturan yang mengatur cara sistem digunakan oleh pengguna.
5. Manusia (brainware): pengguna sistem informasi seperti operator, analis sistem, dan manajer.

Setiap unsur ini saling bergantung satu sama lain. Misalnya, tanpa data yang benar dan relevan, perangkat lunak tidak akan bisa memberikan informasi yang berguna bagi pengguna. Oleh karena itu, desain sistem informasi harus dilakukan secara holistik dan terintegrasi agar dapat menghasilkan sistem yang andal dan bernilai guna tinggi.

2.1.2.2 Fungsi Sistem Informasi

Sistem informasi memiliki fungsi utama dalam mengubah data mentah menjadi informasi yang bernilai. Fungsi-fungsi tersebut mencakup:

1. Input: pengumpulan data dari sumber internal dan eksternal.
2. Processing: pemrosesan data mentah menjadi informasi melalui berbagai metode, termasuk kalkulasi, pengurutan, atau klasifikasi.
3. Output: hasil pemrosesan data dalam bentuk laporan, grafik, atau notifikasi.
4. Storage: penyimpanan data dan informasi untuk digunakan di masa depan.
5. Control & Feedback: pengawasan terhadap proses sistem dan pemberian umpan balik untuk perbaikan berkelanjutan.

Dalam organisasi modern, sistem informasi telah menjadi aset strategis. Sistem informasi mampu meningkatkan efisiensi operasional, mempercepat arus informasi, serta memberikan keunggulan kompetitif melalui pemanfaatan teknologi informasi. Implementasi sistem informasi yang baik memungkinkan organisasi merespons perubahan pasar lebih cepat, meningkatkan layanan kepada pelanggan, dan meminimalkan risiko operasional.

2.1.3 Database

Database merupakan komponen fundamental dalam sistem informasi modern karena berfungsi sebagai tempat penyimpanan utama semua data yang dibutuhkan oleh sistem untuk menjalankan proses-proses bisnis dan operasional. Dalam konteks teknologi informasi, database didefinisikan sebagai kumpulan data yang saling terhubung dan disusun secara sistematis dalam bentuk digital, sehingga dapat dengan mudah diakses, dimodifikasi, dan dikelola melalui bantuan perangkat lunak manajemen basis data atau yang dikenal sebagai DBMS (Database Management System). Penggunaan database memungkinkan organisasi menyimpan informasi dalam jumlah besar dengan tingkat keakuratan yang tinggi, menghindari redundansi data, serta mendukung kebutuhan penyajian data secara real-time. Tanpa database yang dirancang dan diimplementasikan dengan baik, sistem informasi akan kehilangan kemampuan untuk mengelola data secara efisien dan aman, yang pada akhirnya dapat menghambat proses pengambilan keputusan maupun kinerja operasional harian. Oleh karena itu, database tidak hanya dipandang sebagai fasilitas penyimpanan data, tetapi juga sebagai fondasi logis yang menentukan keberhasilan sistem informasi secara keseluruhan (Kusnadi, 2022).

Secara fungsional, database memiliki beberapa peran utama dalam sistem informasi yang saling mendukung dan tidak dapat dipisahkan satu sama lain. Fungsi utamanya meliputi penyimpanan data secara terstruktur, pengelolaan hak akses untuk menjaga keamanan informasi, pemrosesan data agar dapat diubah menjadi informasi yang bermanfaat, serta penyediaan mekanisme kontrol terhadap integritas dan konsistensi data. Dalam praktiknya, database mendukung pengolahan data yang

bersifat transaksi (transactional processing) maupun analitik (analytical processing), sehingga dapat dimanfaatkan dalam berbagai konteks penggunaan mulai dari operasional harian hingga pengambilan keputusan strategis jangka panjang. Selain itu, database juga memiliki peran penting dalam pengembangan aplikasi karena menjadi media utama komunikasi antara frontend dan backend, di mana seluruh informasi yang diolah dan ditampilkan ke pengguna berasal dari data yang tersimpan dalam sistem database. Dengan struktur tabel, relasi antar entitas, serta aturan validasi yang ketat, database mampu menyediakan data yang tidak hanya tersedia, tetapi juga relevan dan akurat dalam konteks penggunaannya.

Perancangan database dalam sistem informasi membutuhkan pendekatan yang metodologis, mencakup identifikasi kebutuhan data, perancangan model entitas dan relasi, hingga implementasi skema fisik yang sesuai dengan platform DBMS yang digunakan. Tahap awal perancangan biasanya dimulai dari penyusunan diagram Entity Relationship (ERD), yang bertujuan untuk memodelkan bagaimana entitas dalam sistem saling berhubungan dan data apa saja yang diperlukan dari masing-masing entitas tersebut. Setelah desain logis selesai, proses dilanjutkan dengan normalisasi data untuk menghindari redundansi dan memastikan efisiensi penyimpanan. Implementasi skema tersebut dalam bentuk tabel-tabel pada DBMS memungkinkan sistem untuk mengeksekusi operasi seperti penyimpanan, pencarian, penghapusan, dan pembaruan data secara konsisten dan terkontrol. Dalam pengembangannya, pemilihan jenis database – baik relasional (RDBMS) seperti MySQL dan PostgreSQL, maupun non-relasional seperti MongoDB – disesuaikan dengan kebutuhan sistem dari segi volume data, kecepatan akses, serta fleksibilitas struktur data yang dibutuhkan.

Selain sebagai komponen teknis, database juga memiliki implikasi strategis dalam organisasi, terutama dalam hal menjaga keberlangsungan dan keamanan informasi yang menjadi aset penting. Sistem informasi yang andal bergantung pada database yang mampu menangani transaksi secara akurat, melakukan backup secara berkala, dan menyediakan mekanisme pemulihan saat terjadi kegagalan sistem. Dalam banyak kasus, ketahanan dan kecepatan pemrosesan database menjadi

indikator utama dalam menilai kualitas sistem secara keseluruhan, karena performa sistem sangat dipengaruhi oleh bagaimana data disimpan dan dikelola. Oleh karena itu, pengelolaan database tidak hanya menjadi tanggung jawab teknis, tetapi juga bagian dari strategi manajerial yang melibatkan kebijakan keamanan data, pengaturan hak akses, dan audit informasi secara berkala. Peran penting inilah yang menjadikan database sebagai pusat gravitasi dari seluruh sistem informasi yang mendukung aktivitas bisnis, pelayanan publik, maupun sistem digital lainnya di era modern saat ini (Kusnadi, 2022).

2.2 Tinjauan Studi

Tinjauan studi dilakukan untuk menelaah penelitian-penelitian sebelumnya yang relevan sebagai referensi dalam pengembangan sistem pemesanan tiket Travel Bang Udin Cirebon. Penelitian terdahulu memberikan gambaran dan pertimbangan terhadap pendekatan, teknologi, dan metode yang dapat digunakan dalam proyek ini. Berikut adalah beberapa penelitian terdahulu yang dijadikan acuan:

1. Tinjauan studi pertama diambil dari penelitian yang berjudul "PERANCANGAN DASHBOARD MONITORING PENJUALAN PADA WEBSITE PATERON.ID MENGGUNAKAN FRAMEWORK LARAVEL DAN VUE JS" Oleh Rendra Dwi Bima Sakti, Sri Lestanti, Saiful Nur Budiman (2024) Penelitian ini membahas permasalahan tracking penjualan di website Pateron Indonesia yang masih menggunakan metode konvensional (spreadsheet), menyebabkan ketidakefisienan waktu dan risiko kesalahan data. Tujuannya adalah membangun dashboard monitoring berbasis KPI (Key Performance Indicator) penjualan untuk menyajikan data otomatis dan real-time. Metode yang digunakan adalah prototyping dengan pengujian black box testing dan evaluasi kualitas informasi model DeLone and McLean. Hasilnya, dashboard berhasil diimplementasikan dengan tingkat keberhasilan fungsional 91% dan kualitas informasi 89.28%, sehingga membantu tim pemasaran dalam pengambilan keputusan strategis.
2. Tinjauan studi kedua diambil dari penelitian yang berjudul "Penerapan REST API Laravel sebagai Fondasi Back-end Aplikasi G-MOOC 4D" Oleh Iman Nurjaman, Fandy Setyo Utomo, Nandang Hermanto (2024) Penelitian

ini berfokus pada pengembangan aplikasi MOOC inklusif untuk penyandang disabilitas tunanetra. Permasalahan utamanya adalah kebutuhan fondasi back-end yang efisien dan aman untuk pertukaran data antara front-end dan sistem AI. Tujuannya adalah membangun REST API menggunakan framework Laravel untuk mengelola operasi CRUD (Create, Read, Update, Delete) berbasis JSON. Metode pengembangan menggunakan model waterfall (analisis kebutuhan, desain, implementasi, pengujian). Hasilnya, REST API berhasil diimplementasikan dengan semua endpoint teruji valid melalui black box testing menggunakan Postman.

3. Tinjauan studi ketiga diambil dari penelitian yang berjudul "SISTEM INFORMASI PEMESANAN TIKET PADA CV. GLOBAL TRANS SOLUTIONS" Oleh T. Ardiansyah (2020) Penelitian ini mengatasi masalah pemesanan tiket pesawat konvensional (via loket/telepon) yang tidak efisien dan berisiko kesalahan. Tujuannya adalah merancang sistem berbasis web untuk memudahkan konsumen memesan tiket secara online tanpa batas geografis. Metode pengembangan menggunakan waterfall dan pengumpulan data melalui studi lapangan (wawancara, observasi) serta studi pustaka. Sistem dibangun dengan PHP dan MySQL. Hasil implementasi menunjukkan sistem berfungsi optimal dalam menangani pemesanan, pembayaran, dan cetak tiket, serta meningkatkan akurasi data.
4. Tinjauan studi keempat diambil dari penelitian yang berjudul "SISTEM INFORMASI PEMESANAN TIKET PADA INDAH TRAVEL BERBASIS WEB" Oleh Muhammad Khathab, Muh. Rasyid Ridha (2020) Penelitian ini menyoroti keterbatasan pemesanan tiket travel konvensional (loket/telepon) di Indah Travel yang menghambat akses informasi jadwal dan harga. Tujuannya adalah membangun sistem berbasis web untuk memberikan kemudahan pemesanan real-time. Metode analisis menggunakan PIECES (Performance, Information, Economy, Control, Efficiency, Service) dan pemodelan UML (Use Case Diagram, Activity Diagram). Sistem dikembangkan dengan PHP dan MySQL. Hasil pengujian membuktikan semua fitur (pemesanan, pembatalan, laporan) berjalan baik, mengurangi biaya dan waktu pelanggan.

5. Tinjauan studi kelimat diambil dari penelitian yang berjudul “Sistem Informasi Pemesanan dan Pembelian Tiket Bus Online Berbasis Web pada P.O. Budiman” Oleh Joko S Dwi Raharjo, Rahmat Tullah, Hari Setiana – Jurnal Sisfotek Global, Vol. 9 No. 2 (2019) Penelitian ini merancang sebuah sistem informasi berbasis web untuk mendukung pemesanan dan pembelian tiket bus secara online pada perusahaan otobus P.O. Budiman. Permasalahan yang diangkat adalah belum tersedianya sistem digital dalam proses pemesanan tiket, sehingga menyulitkan pelanggan untuk mengetahui ketersediaan tiket dan harus datang langsung ke agen. Dalam pengembangan sistem, digunakan pendekatan berbasis objek dengan Unified Modeling Language (UML), serta pengujian dilakukan menggunakan metode blackbox. Sistem dibagi ke dalam beberapa peran seperti pelanggan, admin, dan pimpinan. Hasil penelitian menunjukkan bahwa sistem dapat membantu pelanggan dalam memperoleh informasi dan melakukan pemesanan tiket secara lebih mudah, serta memudahkan pihak perusahaan dalam manajemen data transaksi.