BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Rancang Bangun

Rancang bangun merupakan konsep yang mencakup dua tahap utama dalam pengembangan sistem atau produk, yaitu perancangan (rancang) dan pembangunan (bangun), yang saling berkaitan dalam menciptakan solusi yang efektif dan efisien. Menurut Kamus Besar Bahasa Indonesia (KBBI), rancang berarti proses merencanakan atau memikirkan sesuatu secara matang sebelum dilaksanakan. Dalam konteks teknologi informasi, rancang mengacu pada tahap awal pengembangan sistem yang mencakup perencanaan, pemodelan, dan pembuatan desain berdasarkan kebutuhan pengguna dan spesifikasi teknis yang ditetapkan.

Sementara itu, bangun menurut KBBI berarti tindakan membangun atau mewujudkan sesuatu, baik secara fisik maupun abstrak. Dalam pengembangan sistem, bangun mengacu pada proses realisasi desain menjadi sistem yang siap digunakan, termasuk pengkodean, pengintegrasian, serta uji coba untuk memastikan fungsionalitas sistem sesuai perencanaan. Bangun tidak hanya mencakup pembangunan fisik, tetapi juga implementasi perangkat lunak yang dilakukan secara bertahap dan sistematis.

Ketika digabungkan, rancang bangun merujuk pada proses menyeluruh dari perencanaan hingga implementasi sistem atau produk. menjelaskan bahwa dalam konteks rekayasa perangkat lunak, rancang bangun melibatkan aktivitas untuk merancang struktur sistem, antarmuka, serta komponen yang diperlukan agar sistem dapat berfungsi sesuai kebutuhan. Dengan kata lain, rancang bangun adalah proses terstruktur yang mencakup perencanaan, desain, dan pembangunan untuk menciptakan solusi yang dapat diimplementasikan secara efektif (Abdur, 2021).

Proses rancang bangun melibatkan tahapan analisis kebutuhan, desain sistem dan desain antar muka, implementasi, dan pengujian. Pada tahap analisis kebutuhan, pengembang mengidentifikasi kebutuhan pengguna dan

mendefinisikan fungsi sistem yang diharapkan. Tahap ini diikuti dengan perancangan sistem, di mana dibuat model atau blueprint sistem, baik secara visual maupun logis, sebagai panduan untuk pengembangan lebih lanjut.

Tahap bangun pada rancang bangun melibatkan implementasi desain ke dalam sistem aktual melalui pengkodean, integrasi komponen, dan pengujian untuk memastikan kesesuaian dengan desain yang telah disepakati. Proses ini sering kali mencakup penyesuaian berdasarkan hasil pengujian untuk mengoptimalkan kinerja dan kegunaan sistem. Setelah implementasi, pengujian menyeluruh dilakukan untuk memastikan sistem bebas dari kesalahan dan memenuhi persyaratan yang telah ditentukan.

2.1.2 Aplikasi

Aplikasi adalah perangkat lunak yang dirancang oleh pengembang atau pengguna untuk menjalankan tugas tertentu secara spesifik (Kadir, 2003). Pada konteks ini aplikasi merupakan program siap pakai yang dirancang untuk melaksanakan fungsi tertentu, baik untuk pengguna maupun untuk mendukung kinerja aplikasi lainnya. Dalam dunia teknologi informasi, aplikasi juga dapat diartikan sebagai penerapan suatu konsep atau sistem yang bertujuan membantu pengguna menyelesaikan tugas-tugas tertentu secara efisien.

Aplikasi perangkat lunak secara umum dapat dikategorikan ke dalam dua kelompok utama, yaitu:

a. Aplikasi perangkat lunak spesialis: Program yang dilengkapi dengan dokumentasi terintegrasi dan dirancang untuk menyelesaikan tugas-tugas spesifik, seperti perangkat lunak untuk desain grafis, pengelolaan data, atau simulasi teknis.
b. Aplikasi perangkat lunak paket: Program yang dirancang dengan dokumentasi terintegrasi untuk menangani jenis masalah tertentu, misalnya perangkat lunak pengolah kata atau aplikasi akuntansi.

Berdasarkan pengertian tersebut, aplikasi dapat diartikan sebagai sekumpulan instruksi atau kode yang disusun secara sistematis untuk

melaksanakan perintah tertentu. Perintah ini diberikan oleh pengguna dan dijalankan melalui perangkat keras komputer, menghasilkan solusi atau informasi yang diharapkan. Dengan kata lain, aplikasi dirancang untuk mempermudah pengguna dalam menyelesaikan tugas-tugas tertentu sekaligus menawarkan solusi praktis untuk berbagai kebutuhan.

Aplikasi tidak hanya terbatas pada perangkat lunak sederhana, tetapi juga mencakup sistem yang kompleks untuk mendukung aktivitas spesifik dalam skala yang lebih besar. Keberadaan aplikasi menunjukkan peran penting teknologi informasi dalam kehidupan manusia, membantu berbagai aspek seperti peningkatan produktivitas, hiburan, hingga pengelolaan data yang rumit (Lutfia & Waryanto, 2017).

2.1.3 Kustomisasi

Kustomisasi merupakan proses penyesuaian suatu produk, layanan, atau sistem agar sesuai dengan kebutuhan, preferensi, atau keinginan individu atau kelompok tertentu. Dalam pengertian umum, kustomisasi tidak terbatas pada konteks teknologi informasi, tetapi mencakup berbagai bidang, seperti manufaktur, pemasaran, hingga layanan jasa. Kustomisasi adalah strategi untuk menyediakan produk atau layanan yang unik bagi setiap pelanggan tanpa kehilangan efisiensi skala produksi massal (Pine II. 1993). Strategi ini memungkinkan penyedia memenuhi kebutuhan pelanggan dengan lebih baik dibandingkan pendekatan produk standar.

Menurut Gilmore dan Pine II (1997), kustomisasi dapat dibagi menjadi empat tipe:

a. ***Collaborative Customization***

Penyedia bekerja sama dengan pelanggan untuk menentukan produk atau layanan yang sesuai kebutuhan mereka. Contoh: layanan desain interior yang melibatkan klien dalam proses perancangan.

b. ***Adaptive Customization***

Produk atau layanan dirancang dengan fitur fleksibel yang dapat disesuaikan oleh pelanggan sendiri. Contoh: furnitur modular yang

memungkinkan pengguna mengatur ulang komponen sesuai kebutuhan.

c. Cosmetic Customization

Penyesuaian dilakukan pada aspek visual atau kemasan produk agar lebih sesuai dengan preferensi pelanggan tanpa mengubah fungsi inti. Contoh: kemasan dengan nama atau pesan pribadi.

d. Transparent Customization

Penyedia menyesuaikan produk atau layanan berdasarkan informasi yang diperoleh tentang pelanggan, tanpa keterlibatan langsung pelanggan. Contoh: layanan streaming yang merekomendasikan konten berdasarkan riwayat tontonan.

Kustomisasi juga relevan dalam industri kreatif, seperti sablon atau desain grafis, di mana pelanggan menginginkan produk yang mencerminkan identitas atau preferensi mereka. Dalam konteks ini, proses kustomisasi tidak hanya meningkatkan kepuasan pelanggan, tetapi juga memperkuat loyalitas dan nilai produk di mata konsumen.

Kustomisasi dalam konteks aplikasi merujuk pada kemampuan menyesuaikan perangkat lunak agar sesuai dengan kebutuhan dan preferensi pengguna. Dalam dunia teknologi informasi, kustomisasi dapat mencakup berbagai aspek, mulai dari antarmuka pengguna hingga fungsi dan fitur yang ditawarkan oleh aplikasi. Dengan kustomisasi, pengguna dapat memodifikasi pengalaman penggunaan sesuai dengan cara mereka berinteraksi dengan aplikasi, sehingga meningkatkan kepuasan dan efisiensi.

Kustomisasi dapat dikelompokkan menjadi beberapa kategori, yaitu perubahan pada antarmuka pengguna (UI), laporan, dokumen, dan formulir, alur kerja, integrasi dengan aplikasi atau modul lain, ekstensi fungsi, dan modifikasi fungsi yang telah ada. Kustomisasi pada konteks rekayasa perangkat lunak adalah adaptasi kode program yang digunakan sesuai dengan bahasa pemrograman yang dipakai untuk mengembangkan sistem (Dittrich et al., 2009).

2.1.4 Sablon

Sablon merupakan teknik cetak yang menggunakan kain berpori atau screen untuk mentransfer tinta ke permukaan bahan seperti tekstil, kertas, plastic, atau kaca sebagai media cetaknya. Proses ini memungkinkan pembuatan desain yang detail dan beragam pada berbagai media (Tobroni, M.I. 2011). Sablon memiliki keunikan tersendiri dalam dunia desain, terutama dalam pembuatan kaos, karena mampu menghasilkan karya dengan karakteristik khusus yang tidak dapat dicapai oleh teknik cetak lainnya.

Teknik sablon dibagi menjadi beberapa jenis, antara lain:

a. Sablon Manual: Proses cetak dilakukan secara manual dengan bantuan screen dan rakel. Teknik ini memerlukan keterampilan khusus dan ketelitian tinggi.
b. Sablon Digital (*Direct to Garment/DTG*): Menggunakan printer khusus untuk mencetak desain langsung ke permukaan kain. Metode ini lebih cepat dan cocok untuk produksi dalam jumlah kecil dengan desain kompleks.
c. Sablon Flocking: Menghasilkan cetakan dengan tekstur berbulu pada permukaannya, memberikan efek visual dan tekstur yang menarik.
d. Sablon Plastisol: Menggunakan tinta plastisol yang menghasilkan cetakan dengan warna cerah dan daya tahan tinggi, namun memerlukan proses pemanasan untuk mengeringkan tinta.

Setiap teknik sablon memiliki kelebihan dan kekurangan, serta aplikasi yang berbeda tergantung pada kebutuhan desain, jenis bahan, dan anggaran produksi. Pemilihan teknik yang tepat akan mempengaruhi kualitas akhir produk dan kepuasan pelanggan.

2.2 Teori Khusus

2.2.1 System Development Life Cycle (SDLC)

System Development Life Cycle (SDLC) adalah proses terstruktur yang digunakan dalam pengembangan dan pemeliharaan sistem informasi. SDLC berfungsi sebagai kerangka kerja yang memandu seluruh tahap pengembangan, mulai dari perencanaan hingga pemeliharaan sistem.

Metodologi ini membantu tim proyek dalam mengelola proyek secara lebih terorganisir dan memastikan bahwa sistem yang dibangun sesuai dengan kebutuhan pengguna dan tujuan bisnis perusahaan (Valacich & George, 2017). SDLC dapat diterapkan baik untuk pengembangan sistem baru maupun untuk pembaruan atau perbaikan sistem yang sudah ada.

Secara umum, SDLC terdiri dari beberapa tahap, antara lain perencanaan, analisis, desain, dan implementasi. Meskipun tahapan ini tampak dilakukan secara linier, pada kenyataannya proses SDLC dapat bersifat iteratif dan fleksibel, artinya jika ada kesalahan atau perubahan kebutuhan, tim dapat kembali ke fase sebelumnya untuk melakukan perbaikan (Dennis et al., 2015).

1. Tahap Perencanaan (*Planning*)

   Tahap perencanaan dalam SDLC adalah langkah pertama dan sangat penting. Pada tahap ini, tim proyek mencoba memahami mengapa sistem baru perlu dibangun atau mengapa sistem yang ada perlu diperbarui. Keputusan yang diambil pada fase ini akan mempengaruhi seluruh siklus pengembangan sistem.

   Langkah-langkah utama dalam tahap ini meliputi:

   a. Identifikasi tujuan dan ruang lingkup proyek guna memastikan pemahaman yang jelas tentang apa yang perlu dicapai.
   b. Identifikasi stakeholder dilakukan dengan mengidentifikasi pemangku kepentingan yang terlibat, termasuk *end user*, manajer proyek, dan pihak-pihak lain yang terkait dalam pengembangan sistem.
   c. Studi kelayakan dilakukan untuk menganalisis kelayakan teknis, ekonomi, dan operasional dari proyek, dengan tujuan untuk menentukan apakah proyek tersebut dapat dilanjutkan.
   d. Penilaian risiko dengan mengidentifikasi risiko yang mungkin muncul selama pengembangan sistem dan merencanakan langkah mitigasi untuk mengurangi dampaknya (Valacich & George, 2017).

2. Tahap Analisis (*Analysis*)

Tahap analisis bertujuan untuk mengidentifikasi dan mendokumentasikan kebutuhan sistem. Dalam tahap ini, fokus utamanya adalah memahami lebih dalam mengenai proses bisnis yang ada serta menentukan kebutuhan pengguna yang harus dipenuhi oleh sistem yang baru.

Langkah-langkah dalam tahap analisis meliputi:

a. Pengumpulan kebutuhan dari semua stakeholder melalui metode seperti wawancara, survei, observasi, dan studi dokumentasi. Kebutuhan yang dikumpulkan ini mencakup kebutuhan fungsional dan non-fungsional.
b. Analisis proses bisnis yang sedang berjalan untuk memastikan bahwa sistem yang akan dibangun mampu mendukung proses tersebut atau bahkan meningkatkan efisiensinya.
c. Dokumentasi kebutuhan harus jelas agar tim pengembang dapat mengikuti panduan yang tepat dalam desain dan implementasi sistem (Dennis et al., 2015). Dokumentasi kebutuhan ini menjadi acuan dalam mendesain sistem yang memenuhi tujuan dan kebutuhan bisnis.

3. Tahap Desain (*Design*)

Tahap desain adalah langkah di mana sistem mulai dibentuk secara rinci. Desain yang dihasilkan akan mencakup struktur sistem, arsitektur perangkat keras dan perangkat lunak, serta tampilan antarmuka pengguna.

Tahap ini terdiri dari beberapa elemen penting:

a. Desain arsitektur sistem mencakup desain server, database, jaringan, dan perangkat lunak utama.
b. Desain data termasuk data yang akan digunakan oleh sistem dirancang, termasuk perancangan database, tabel, dan hubungan antara berbagai entitas.
c. Desain antarmuka pengguna dirancang agar ramah pengguna dan intuitif. Desain yang baik memastikan bahwa pengguna dapat

berinteraksi dengan sistem dengan mudah tanpa perlu keahlian teknis yang mendalam (Dennis et al., 2015).

Hasil dari tahap desain adalah *blue print* yang jelas untuk implementasi sistem yang akan dibangun.

4. Tahap Implementasi (*Implementation*)

Tahap implementasi adalah saat di mana desain sistem diubah menjadi kode atau perangkat lunak yang berfungsi. Pada tahap ini, tim pengembang akan melakukan pengkodean, pengujian, dan instalasi sistem di lingkungan produksi.

Langkah-langkah dalam tahap ini meliputi:

a. Pengkodean (*Coding*) yaitu mengubah desain menjadi kode program yang dapat dijalankan. Biasanya, bahasa pemrograman yang dipilih akan disesuaikan dengan kebutuhan dan platform yang digunakan.
b. Pengujian (*Testing*) yaitu pengujian Sistem secara menyeluruh untuk memastikan bahwa semua fungsi berjalan dengan baik sesuai spesifikasi yang telah ditetapkan.
c. Penerapan (*Deployment*) yaitu proses Setelah sistem lulus uji, di mana sistem dipasang di lingkungan produksi dan mulai digunakan oleh pengguna. Pada tahap ini, data dari sistem lama juga dapat dimigrasikan ke sistem baru jika diperlukan (Dennis et al., 2015).

5. Tahap Pemeliharaan (*Maintenance*)

Pemeliharaan adalah tahap yang berlangsung setelah sistem diimplementasikan dan digunakan. Pada tahap ini, segala masalah yang ditemukan akan diperbaiki, dan penyesuaian atau peningkatan sistem dilakukan seiring dengan perubahan kebutuhan bisnis.

Langkah-langkah utama dalam tahap ini meliputi:

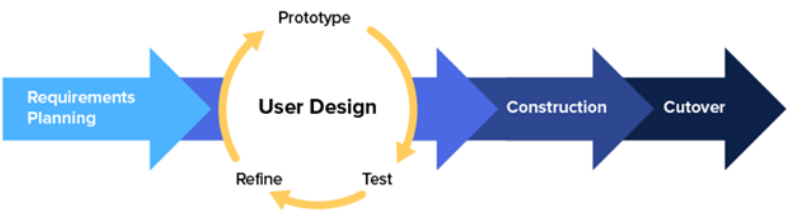
a. Perbaikan kesalahan Jika ditemukan *bug* atau masalah dalam sistem.

b. Pembaruan sistem dilakukan ketika ada perubahan dalam kebutuhan bisnis atau teknologi baru tersedia, sistem akan diperbarui untuk mengakomodasi perubahan tersebut.
c. Pemantauan kinerja secara berkelanjutan untuk memastikan sistem tetap berfungsi dengan baik dan efisien (Valacich & George, 2017).

Meskipun SDLC sering kali dianggap sebagai proses linier, kenyataannya setiap fase dalam SDLC dapat bersifat iteratif. Proyek dapat kembali ke fase sebelumnya jika diperlukan, terutama jika ditemukan masalah yang memerlukan revisi. Misalnya, jika selama tahap pengujian ditemukan masalah besar, tim pengembang mungkin perlu kembali ke tahap desain untuk memperbaiki struktur sistem. Selain itu, beberapa aktivitas dapat dilakukan secara bersamaan di beberapa fase untuk mempercepat pengembangan sistem.

2.2.2 *Metode Rapid Application Development* (RAD)

Metode *Rapid Application Development* (RAD) adalah metodologi pengembangan perangkat lunak yang menekankan pada pengembangan yang cepat dan iteratif, dengan tujuan menghasilkan produk yang dapat memenuhi kebutuhan pengguna dalam waktu singkat. RAD berfokus pada penggunaan prototipe dan umpan balik pengguna yang cepat untuk menyempurnakan produk yang sedang dikembangkan. Dengan melibatkan pengguna secara aktif dalam proses pengembangan, pengembang dapat mengidentifikasi kebutuhan dan masalah lebih awal sehingga memungkinkan penyelesaian yang lebih efisien dan sesuai harapan.



Gambar 2.1 *Rapid Application Development Methodology*

Sumber : Hussain, Tyagi, & Peng, 2023, p. 6

Pada Gambar 2.1 menunjukkan bahwa menurut (Husain, 2023) proses RAD terdiri dari empat tahap utama, termasuk analisis kebutuhan atau *requirements planning*, pembuatan prototipe atau *user design*, implementasi atau *construction*, dan pengujian atau *cutover*.

1. Analisis Kebutuhan Pengguna (*Requirements Planning*)

   Tahapan ini dilakukan untuk mendefinisikan tujuan dan mengidentifikasi kebutuhan dasar pengguna. Pada tahap ini pengembang dan pelanggan (*User*) banyak berkomunikasi untuk menetapkan tujuan dalam memenuhi kebutuhan fungsional dan non-fungsional dalam aplikasi yang akan dikembangkan. Hasil dari tahap ini adalah dokumen yang merangkum kebutuhan dan spesifikasi awal sistem. Dokumen tersebut mencakup masalah yang teridentifikasi, penentuan persyaratan sistem, persyaratan user dan semua pihak yang terlibat.
2. Membuat Prototipe (*User Design*)

   Tahap ini berfokus pada pengembangan prototipe sistem untuk merepresentasikan fungsi inti dan desain awal sistem. Prototipe dibuat secara cepat untuk memvisualisasikan alur kerja dan antarmuka pengguna (UI), dengan masukan langsung dari pengguna. Iterasi dilakukan secara berulang untuk memperbaiki desain berdasarkan umpan balik yang diterima, sehingga menghasilkan prototipe yang mendekati bentuk akhir sistem. Fase ini membantu memastikan bahwa kebutuhan pengguna terpenuhi sebelum sistem dibangun sepenuhnya.
3. Implementasi (*Construction*)

   Pada tahap ini, sistem dibangun berdasarkan prototipe yang telah disepakati. Kegiatan pada tahap ini meliputi pengembangan fitur kompleks, pengintegrasian sistem dengan komponen lain serta penulisan dokumentasi teknis. Proses pengembangan dilakukan dengan memperhatikan kualitas kode untuk memastikan sistem yang stabil dan andal. Implementasi juga mencakup persiapan untuk pengujian dan migrasi data ke dalam sistem baru.
4. Pengujian (*Cutover*)

Tahap ini bertujuan untuk memastikan sistem berjalan dengan baik sebelum diterapkan sepenuhnya. Berbagai jenis pengujian dilakukan, seperti *functional testing*, *performance testing*, dan *user acceptance testing*. Selain itu, data dari sistem lama dimigrasikan ke sistem baru jika diperlukan serta pelatihan pengguna dan penyediaan dokumentasi pendukung dilakukan untuk memastikan pengguna dapat mengoperasikan sistem dengan baik.

2.2.3 *Object-Oriented Analysis and Design* (OOAD)

Object-Oriented Analysis and Design (OOAD) adalah pendekatan dalam pengembangan sistem perangkat lunak yang berfokus pada objek sebagai unit dasar untuk analisis dan desain. Pendekatan ini memperkenalkan cara berpikir yang berbeda dalam membangun perangkat lunak, di mana sistem dianggap sebagai kumpulan objek yang berinteraksi satu sama lain untuk mencapai tujuan tertentu (Wazlawick, 2014). Langkah-langkah Utama dalam OOAD:

1. Analisis Berorientasi Objek (*Object-Oriented Analysis*)

   Analisis berorientasi objek adalah tahap di mana entitas atau objek dalam sistem diidentifikasi berdasarkan persyaratan yang telah dikumpulkan. Ini mencakup beberapa langkah kunci:

   a. Identifikasi semua objek yang relevan dalam domain aplikasi. Proses ini melibatkan pemahaman tentang apa yang harus dilakukan oleh sistem dan bagaimana objek-objek ini berinteraksi.
   b. Pemodelan interaksi dengan menggambarkan interaksi antar objek melalui *use case diagram* memahami alur kerja sistem secara keseluruhan.
   c. Spesifikasi kebutuhan fungsional dan non-fungsional, serta memastikan bahwa semua kebutuhan pengguna telah dipertimbangkan.

2. Desain Berorientasi Objek (*Object-Oriented Design*)

Setelah tahap analisis, desain berorientasi objek berfokus pada pembuatan struktur rinci dari sistem berdasarkan hasil analisis. Beberapa aspek yang harus diperhatikan dalam tahap ini meliputi:

a. Membuat *class diagrams* yang menunjukkan kelas-kelas dalam sistem, atribut, metode, dan hubungan antar kelas.
b. Mengembangkan *sequence diagrams* untuk menunjukkan bagaimana objek berinteraksi satu sama lain selama eksekusi fungsionalitas tertentu.
c. Mendefinisikan desain antarmuka yang harus digunakan oleh objek untuk berinteraksi, sehingga memungkinkan enkapsulasi dan modularitas.

OOAD tidak hanya melibatkan penggunaan bahasa pemrograman berorientasi objek, tetapi juga mengadopsi pola pikir dan teknik yang mendukung prinsip-prinsip berorientasi objek. Beberapa prinsip kunci dalam OOAD mencakup:

1. Enkapsulasi yaitu menyembunyikan detail implementasi objek dan hanya mengekspos apa yang diperlukan melalui antarmuka yang ditentukan. Hal ini membantu menjaga integritas data dan memudahkan perubahan tanpa mempengaruhi bagian lain dari sistem.
2. Abstraksi merupakan prinsip yang berfokus pada identifikasi esensi dari objek dengan mengabaikan detail yang tidak relevan. Abstraksi memungkinkan pengembang untuk model objek secara lebih sederhana dan jelas.
3. Pewarisan (*Inheritance*) memungkinkan kelas-kelas baru (kelas anak) untuk mewarisi atribut dan metode dari kelas yang sudah ada (kelas induk). Ini membantu dalam mengurangi duplikasi kode dan meningkatkan pengorganisasian kode.
4. Polimorfisme memungkinkan objek dari kelas yang berbeda untuk merespons terhadap masukan yang sama dengan cara yang berbeda. Ini memberi fleksibilitas dalam penggunaan objek di dalam sistem.

Pendekatan OOAD juga menekankan pentingnya penggunaan teknik seperti delegasi dan penugasan tanggung jawab untuk menciptakan kode yang lebih modular, dapat digunakan kembali, dan memiliki kopling rendah antara komponen-komponen sistem. Dengan cara ini, perubahan pada satu komponen tidak akan mempengaruhi komponen lain, yang penting dalam pengembangan perangkat lunak yang berskala besar.

Dalam praktiknya, OOAD tidak hanya tentang membuat diagram atau menggunakan alat *Computer-Aided Software Engineering* (CASE), tetapi juga tentang memahami dan menerapkan prinsip-prinsip berorientasi objek secara konsisten. Hal ini termasuk penggunaan berbagai teknik pemodelan, seperti *Unified Modeling Language* (UML), untuk memvisualisasikan berbagai aspek dari sistem yang sedang dikembangkan (Booch et al., 2005).

2.2.4 *Unified Modelling Language* (UML)

Unified Modeling Language (UML) merupakan standar industri yang digunakan dalam pemodelan sistem perangkat lunak untuk memvisualisasikan, menentukan, membuat, dan mendokumentasikan elemen-elemen dari sistem yang kompleks. UML membantu pengembang dan pemangku kepentingan lainnya dalam memahami struktur dan perilaku dari suatu sistem melalui penggunaan berbagai diagram standar yang mudah dipahami. Berdasarkan fungsinya, UML dibagi menjadi dua kelompok utama, yaitu Structure Diagrams dan Behavioral Diagrams, yang berfokus pada aspek struktural dan perilaku dari sistem.

UML pertama kali dikembangkan oleh Grady Booch, Ivar Jacobson, dan James Rumbaugh pada tahun 1990-an dengan tujuan untuk menyatukan berbagai metode pemodelan perangkat lunak yang saat itu masih beragam. Standar UML telah diadopsi oleh *Object Management Group* (OMG) dan menjadi alat yang penting dalam rekayasa perangkat lunak modern (Dennis et al., 2015). Berikut adalah penjelasan dari masing-masing diagram yang ada dalam UML dijelaskan pada Tabel 2.1.

Tabel 2.1 Jenis diagram UML

Nama Diagram	Kegunaan	Fase
	Structure Diagram	
Class	Mengilustrasikan hubungan antar kelas yang dimodelkan dalam sistem.	*Analysis, Design*
	Behavioral Diagrams	
Activity	Mengilustrasikan alur kerja bisnis yang independen dari kelas, alur aktivitas dalam kasus penggunaan, atau desain rinci dari sebuah metode.	*Analysis, Design*
Sequence	Memodelkan perilaku objek dalam sebuah kasus penggunaan; berfokus pada urutan berbasis waktu dari suatu aktivitas.	*Analysis, Design*
Use Case	Menangkap kebutuhan bisnis untuk sistem dan menggambarkan interaksi antara sistem dengan lingkungannya.	*Analysis*

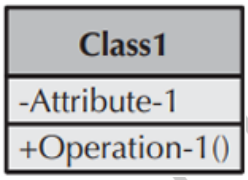
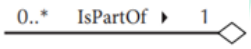
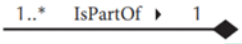
Sumber : Dennis et al., 2015

2.2.4.1 *Class Diagram*

Class Diagram merupakan salah satu diagram dalam *Unified Modeling Language* (UML) yang menggambarkan struktur dari sebuah sistem dengan merepresentasikan kelas-kelas yang ada di dalam sistem tersebut beserta hubungan antar kelas. Setiap kelas pada *Class Diagram* memiliki atribut dan metode yang mencerminkan karakteristik dan perilaku objek dalam sistem. Kelas-kelas ini dihubungkan melalui asosiasi, agregasi, komposisi, atau pewarisan yang menunjukkan bagaimana objek-objek ini saling berinteraksi (Dennis et al., 2015). Diagram ini digunakan untuk memodelkan objek-objek dalam suatu sistem dan bagaimana mereka saling terhubung atau berinteraksi secara hierarkis. Menurut (Booch et al., 2005), *Class Diagram* berperan penting dalam memberikan gambaran struktur yang mendasari sistem perangkat lunak, terutama dalam pengembangan berbasis objek.

Setiap *Class Diagram* terdiri dari beberapa komponen utama, dijelaskan pada Tabel 2.2 *Komponen Class Diagram*

Tabel 2.2 Komponen *Class Diagram*

Simbol	Nama Simbol	Keterangan
	Kelas	Kelas adalah template atau *blueprint* untuk objek dalam sistem. Bagian atas mencantumkan nama kelas (Class1), bagian tengah memuat properti atau variable (Attribute-1), dan bagian bawah mendeskripsikan metode yaitu fungsi atau operasi yang dimiliki kelas (Operation-1()).
	Generalisasi	Relasi ini menunjukkan pewarisan antar kelas. Kelas anak mewarisi atribut dan metode dari kelas induk.
	Agregasi	Menunjukkan hubungan "has-a" di mana kelas induk adalah kumpulan dari kelas lainnya.
	Komposisi	Hubungan "part-of" yang kuat, di mana bagian (part) tidak dapat berdiri sendiri tanpa kelas induk.

Sumber : (Dennis et al., 2015)

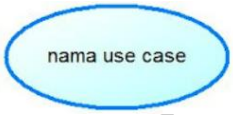
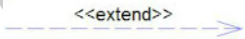
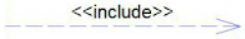
2.2.4.2 *Use Case Diagram*

Use Case Diagram merupakan model yang menjelaskan fungsi-fungsi utama yang disediakan oleh sistem untuk pengguna atau aktor eksternal. Diagram ini menyajikan skenario penggunaan sistem dari sudut pandang pengguna dan menggambarkan apa yang harus dilakukan oleh sistem, tanpa mendetailkan cara sistem melaksanakan tugas tersebut. Setiap *Use Case* mewakili sebuah fungsi yang berguna bagi aktor dan dapat mencakup satu atau beberapa langkah operasi (Dennis et al., 2015).

Use Case Diagram memiliki peran penting dalam memahami persyaratan fungsional sistem karena diagram ini menggambarkan semua kemungkinan interaksi antara aktor dan sistem. Selain itu, diagram ini membantu pengembang untuk mengidentifikasi fungsi-fungsi utama yang perlu dipenuhi oleh sistem (Dennis et al., 2015).

Berikut merupakan penjelasan singkat terkait komponen dalam *Use Case* dijelaskan pada Tabel 2.3.

Tabel 2.3 Komponen *Use Case Diagram*

Simbol	Nama Simbol	Keterangan
	*Use Case*	Use case menggambarkan fungsionalitas yang ditawarkan oleh sistem dalam bentuk unit-unit yang saling berinteraksi dengan aktor atau unit lainnya. Biasanya, penamaan use case diawali dengan kata kerja untuk menjelaskan fungsi atau tindakan yang dilakukan.
	Aktor	Aktor merepresentasikan entitas eksternal yang berinteraksi dengan sistem.
	Association Relationship	Asosiasi merujuk pada hubungan antara aktor dan *use case* yang terlibat dalam *use case* diagram, atau *use case* yang berinteraksi dengan aktor. Simbol asosiasi digunakan untuk menghubungkan elemen-elemen yang saling terkait.
	*Extend Relationship*	Simbol <<*extend*>> pada use case diagram digunakan untuk menunjukkan tambahan yang memperluas fungsionalitas tertentu dari use case utama. Notasi ini digambarkan dengan garis putus-putus yang diakhiri panah mengarah ke use case yang diperluas.
	*Include Relationship*	Simbol <<*include*>> menunjukkan bahwa use case utama selalu melibatkan perilaku dari use case tambahan. Notasi ini digambarkan dengan garis putus-putus yang diakhiri panah mengarah ke use case yang dilibatkan.

Simbol	Nama Simbol	Keterangan
	*Generalization Relationship*	Simbol panah segitiga pada use case diagram merepresentasikan hubungan generalisasi atau pewarisan, di mana use case atau aktor di ujung bawah panah mewarisi karakteristik dari use case atau aktor di ujung atas panah.

Sumber : (Dennis et al., 2015)

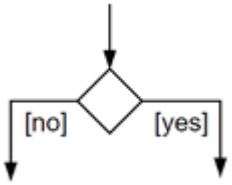
2.2.4.3 *Activity Diagram*

Activity Diagram adalah model grafis yang menggambarkan alur aktivitas atau aliran kerja dalam sistem dari satu tahap ke tahap berikutnya. Diagram ini digunakan untuk memodelkan proses bisnis atau langkah-langkah dalam suatu operasi yang melibatkan beberapa aktor atau sistem, memberikan gambaran yang jelas mengenai bagaimana aktivitas berjalan dalam urutan tertentu. *Activity diagram* juga menekankan transisi antar aktivitas, yang mencerminkan keputusan, alur berurutan, dan alur paralel dalam sistem.

Activity Diagram digunakan untuk menganalisis dan merancang sistem, karena memungkinkan pengembang untuk memahami proses bisnis atau operasional yang rumit, termasuk aktivitas, kondisi, dan alur alternatif (Dennis et al., 2015). Dengan demikian, *activity diagram* memfasilitasi identifikasi kebutuhan dan potensi masalah dalam alur kerja sistem.

Setiap *Activity Diagram* memiliki komponen-komponen utama yang memungkinkan penggambaran alur kerja dalam sistem dengan lebih terstruktur. Berikut adalah komponen-komponen utama *Activity Diagram* dijelaskan pada Tabel 2.4.

Tabel 2.4 Komponen *Activity Diagram*

Simbol	Nama Simbol	Keterangan
	*Swimlane*	Swimlane digunakan untuk merepresentasikan aktor yang melaksanakan berbagai aktivitas, dan jika terdapat beberapa swimlane, mereka digambarkan sejajar untuk menunjukkan keterkaitan antar aktivitas.
	*Initial Node*	Mewakili titik awal sistem berjalan, dengan hanya memiliki satu garis yang terhubung ke aktivitas berikutnya.
	*Final Node*	Merepresentasikan akhir dari alur sistem atau aktivitas.
	*Activity*	Merepresentasikan aktivitas-aktivitas individu dalam suatu alur kerja.
	*Synchronization bar*	dilambangkan dengan garis tebal dan solid. Komponen ini berfungsi untuk mengatur alur aktivitas yang berjalan secara paralel
	*Decision activity*	Merepresentasikan titik di mana alur kerja bercabang berdasarkan kondisi tertentu.
	*Transition arrow*	Direpresentasikan oleh panah yang menghubungkan berbagai aktivitas, menunjukkan urutan eksekusi dari satu aktivitas ke aktivitas berikutnya.

Sumber : (Satzinger et al., 2016)

2.2.4.4 *Sequence Diagram*

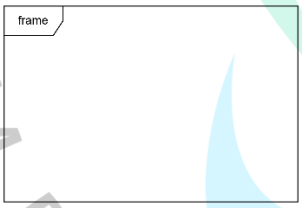
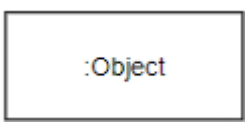
Sequence Diagram adalah diagram interaksi yang mengilustrasikan aliran logis dari suatu proses melalui serangkaian pesan yang dikirim antar objek dalam sistem. Diagram ini

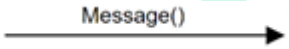
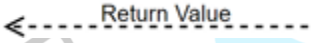
menggambarkan objek yang terlibat dalam suatu proses, pesan-pesan yang ditukar di antara objek tersebut, serta urutan waktu terjadinya pesan. *Sequence diagram* menekankan bagaimana objek-objek dalam sistem berkolaborasi untuk mencapai suatu tujuan atau menghasilkan suatu keluaran yang diinginkan (Satzinger et al., 2016).

Sequence diagram sangat efektif untuk menggambarkan proses-proses yang kompleks atau skenario yang melibatkan banyak interaksi antar objek. Diagram ini memungkinkan pengembang untuk memahami alur logika secara rinci dan menganalisis bagaimana setiap objek berkontribusi terhadap tujuan sistem.

Setiap *sequence diagram* memiliki beberapa komponen utama yang memungkinkannya untuk menggambarkan interaksi antar objek dengan lebih jelas dijelaskan pada Tabel 2.5.

Tabel 2.5 Komponen Sequence Diagram

Simbol	Nama Simbol	Keterangan
	*Frame*	Frame dalam *Sequence diagram* digunakan untuk menandai bagian tertentu, seperti bingkai *loop* untuk menampilkan putaran pesan, frame alternatif untuk logika *if-else* atau *switch*, dan frame opt untuk pemanggilan opsional pesan.
	Aktor	Merepresentasikan individu atau peran yang berinteraksi dengan sistem, berpartisipasi dalam alur dengan mengirim atau menerima pesan, dan digambarkan menggunakan simbol berupa figur batang.
	Objek	Objek yang berpartisipasi dalam suatu urutan direpresentasikan dengan simbol persegi panjang yang memuat nama objek. Notasi tersebut dapat mencakup titik dua sebelum nama kelas yang digarisbawahi, yang bersifat

Simbol	Nama Simbol	Keterangan
		opsional, untuk mengindikasikan bahwa objek tersebut merupakan instance dari kelas yang tidak secara eksplisit disebutkan. Objek ini berperan dengan mengirim atau menerima pesan dalam alur interaksi.
	*Object lifeline*	*Sequence message* ditampilkan dari bagian atas ke bawah dan digambarkan dengan garis putus-putus vertikal.
	*Activity Lifeline*	*Activity lifeline* menunjukkan saat sebuah objek mengirim atau menerima pesan, dan digambarkan dengan simbol persegi panjang sempit yang berada di atas *Object lifeline*.
	*Message*	Digunakan untuk mengirim informasi dari satu objek ke objek lainnya, dengan panggilan operasi dilabeli sesuai pesan yang dikirim dan direpresentasikan oleh panah solid.
	*Return*	Pesan *return* digunakan untuk mengirim kembali nilai dari satu objek ke objek lainnya, dilabeli dengan keterangan nilai yang dikembalikan dan direpresentasikan dengan panah putus-putus.

Sumber : (Satzinger et al., 2016).

2.2.5 *Database Management System* (DBMS)

Sistem Pengelolaan Basis Data atau *Database Management System* (DBMS) adalah perangkat lunak yang digunakan untuk membuat, mengelola, dan memelihara basis data. DBMS menyediakan antarmuka untuk pengguna dalam mengelola data secara aman dan efektif serta menyediakan fitur untuk mengakses, memanipulasi, dan mengendalikan data yang disimpan di dalam basis data (Satzinger et al., 2016). Sistem ini dirancang untuk mendukung berbagai fungsi, termasuk penyimpanan,

pemeliharaan integritas, keamanan, dan pengelolaan data secara bersamaan di lingkungan multi-pengguna.

DBMS memungkinkan pengguna untuk menyimpan data secara terstruktur serta mengatur data dengan menggunakan model tertentu, seperti model relasional, model hierarki, atau model jaringan. Model relasional adalah yang paling umum digunakan, di mana data disimpan dalam bentuk tabel-tabel yang memiliki hubungan dengan tabel lain dalam basis data.

DBMS terdiri dari beberapa komponen utama yang mendukung pengelolaan data secara efisien dan terstruktur, yaitu:

1. ***Data Definition Language*** **(DDL)** merupakan bagian dari bahasa *query database*. DDL digunakan untuk mendefinisikan struktur dan skema database yang memungkinkan pengguna untuk membuat, memodifikasi, dan menghapus objek database seperti tabel, indeks, tampilan, dan konstrain.
2. ***Data Manipulation Language*** **(DML)** merupakan bagian dari bahasa *query* database yang digunakan untuk memanipulasi data dalam tabel, memungkinkan operasi penyisipan *(insert)*, pembaruan (*update*), penghapusan (*delete*), dan pengambilan (*select*) data dari tabel.
3. ***Software Controller*** merupakan komponen perangkat lunak yang memiliki tanggung jawab untuk mengendalikan atau mengatur perilaku sistem basis data secara keseluruhan, termasuk manajemen koneksi, manajemen transaksi, pengelolaan keamanan, optimisasi kinerja, dan pemantauan serta pemeliharaan.

Implementasi DBMS dalam bisnis modern sangat penting untuk meningkatkan efisiensi operasional dan pengambilan keputusan berbasis data. Dengan mengadopsi DBMS, organisasi dapat memastikan integritas data, mengurangi redundansi, dan memfasilitasi akses data yang lebih cepat dan akurat.

Ada beberapa jenis DBMS yang umum digunakan dalam berbagai aplikasi, di antaranya adalah:

1. ***Relational Database Management System* (RDBMS)**

   Sistem ini menyimpan data dalam bentuk tabel (relasi) yang saling berhubungan. Contoh RDBMS yang populer adalah MySQL, PostgreSQL, dan Oracle Database. DBMS jenis ini banyak digunakan karena kemudahan dalam manipulasi data dan fleksibilitas dalam membangun hubungan antar tabel.

2. ***Hierarchical Database Management System***

   Dalam sistem ini, data disusun secara hierarkis dengan struktur pohon. Setiap entitas memiliki satu parent dan bisa memiliki beberapa child. DBMS hierarkis digunakan pada aplikasi yang memerlukan struktur data yang tetap, seperti pada sistem.

3. ***Network Database Management System***

   DBMS jaringan menggunakan model data yang memungkinkan satu child memiliki beberapa parent. Model ini lebih fleksibel dibandingkan model hierarkis dan digunakan pada beberapa sistem bisnis yang kompleks.

4. ***DBMS NoSQL***

   DBMS NoSQL adalah jenis DBMS yang dirancang untuk menangani data tidak terstruktur atau semi-terstruktur. NoSQL populer dalam aplikasi yang membutuhkan skalabilitas tinggi dan pengelolaan data yang besar, seperti big data. Contoh DBMS NoSQL meliputi MongoDB, Cassandra, dan Redis.

Tipe data merupakan konsep fundamental dalam sistem basis data yang menentukan jenis nilai yang dapat disimpan dalam suatu kolom tabel. Pemilihan tipe data yang tepat sangat penting untuk memastikan integritas, efisiensi penyimpanan, dan kecepatan akses data. Tipe data dalam basis data dapat diklasifikasikan sebagai berikut:

4. **Tipe Data Numerik:**

   a. **Integer**: Digunakan untuk menyimpan bilangan bulat, seperti 1, 2, atau -3. Tipe data ini cocok untuk menyimpan data yang tidak memerlukan komponen desimal, seperti jumlah barang atau usia.

b. **Float/Double**: Digunakan untuk menyimpan bilangan desimal atau pecahan, seperti 0.1 atau -0.1. Tipe data ini ideal untuk data yang memerlukan presisi tinggi, seperti nilai mata uang atau hasil pengukuran ilmiah.

5. **Tipe Data Karakter:**
   a. **Char**: Menyimpan satu karakter tunggal, seperti 'A' atau 'b'. Tipe data ini digunakan ketika panjang data sudah diketahui dan tetap.
   b. **Varchar**: Menyimpan deretan karakter dengan panjang variabel, cocok untuk teks seperti nama atau alamat. Penggunaan tipe data ini lebih efisien dalam hal penyimpanan karena hanya menggunakan ruang sesuai dengan panjang data yang disimpan.
6. **Tipe Data Tanggal dan Waktu:**
   a. **Date**: Menyimpan data tanggal, misalnya '2023-01-01'. Tipe data ini memastikan bahwa hanya nilai tanggal yang valid yang dapat disimpan, sehingga membantu dalam menjaga integritas data.
   b. **Time**: Menyimpan data waktu, misalnya '12:30:45'. Digunakan untuk merekam waktu terjadinya suatu peristiwa atau transaksi.
   c. **Datetime**: Menyimpan kombinasi tanggal dan waktu, memungkinkan pencatatan momen spesifik secara lengkap.
7. **Tipe Data Boolean:**

   Menyimpan nilai logika benar (*true*) atau salah (*false*). Tipe data ini sering digunakan dalam operasi logika dan pengambilan keputusan dalam aplikasi.

Pemilihan tipe data yang tepat dalam perancangan basis data sangat penting untuk memastikan efisiensi penyimpanan dan kecepatan akses data. Selain itu, penentuan tipe data yang sesuai juga membantu dalam menjaga integritas data dan mencegah kesalahan input yang tidak diinginkan. Sebagai contoh, penggunaan tipe data 'Date' memastikan bahwa hanya nilai tanggal yang valid yang dapat disimpan dalam kolom tersebut.

Dalam implementasinya, berbagai sistem manajemen basis data (DBMS) mungkin memiliki variasi atau penamaan berbeda untuk tipe data

yang serupa. Oleh karena itu, penting untuk merujuk pada dokumentasi DBMS yang digunakan untuk memahami detail spesifik mengenai tipe data yang didukung.

2.2.6 *Relational Database Management System* (RDBMS)

Relational Database Management System (RDBMS) merupakan jenis DBMS yang bertugas mengatur penyimpanan data dalam bentuk struktur tabel atau relasi. Tabel dalam RDBMS memiliki kemiripan dengan tabel konvensional, berupa struktur data dua dimensi yang terdiri dari baris dan kolom. Namun, istilah dalam RDBMS sedikit berbeda. Baris dalam tabel relasional disebut sebagai tupel, catatan, atau baris, sedangkan kolom dikenal sebagai atribut atau field. Setiap elemen data di dalam sel tabel disebut sebagai nilai atribut atau nilai field. Proses akses dan pengeditan data dalam RDBMS dilakukan melalui penggunaan *Structured Query Language* (SQL) (Satzinger et al., 2016).

Setiap tabel dalam RDBMS diwajibkan memiliki kunci unik yang disebut *Primary Key*. *Primary Key* adalah atribut atau kombinasi atribut yang memiliki nilai unik di seluruh baris tabel, yang bertujuan untuk mengidentifikasi setiap entitas dalam tabel secara eksklusif. Selain itu, tabel juga dapat memiliki atribut atau kumpulan atribut yang berfungsi sebagai *Foreign Key*, yang digunakan untuk menghubungkan tabel satu dengan tabel lainnya dengan mengacu pada *Primary Key* di tabel terkait. Jika terdapat beberapa atribut unik yang dapat berfungsi sebagai *Foreign Key*, perancang database harus menentukan salah satu yang paling sesuai untuk digunakan.

RDBMS menggunakan tabel sebagai unit dasar untuk menyimpan data. Setiap tabel dalam RDBMS terdiri dari baris (rekaman) dan kolom (atribut), di mana setiap baris merepresentasikan satu entitas yang berbeda. Hubungan antara tabel-tabel ini dilakukan melalui penggunaan primary key dan foreign key.

1. ***Primary Key**: Primary key* adalah atribut unik dalam tabel yang digunakan untuk mengidentifikasi setiap baris secara eksklusif.

Dalam satu tabel, hanya boleh ada satu *primary key*, yang menjamin bahwa tidak ada dua baris yang memiliki nilai *primary key* yang sama.

2. ***Foreign Key**: Foreign key* adalah atribut dalam satu tabel yang menjadi referensi ke *primary key* di tabel lainnya. Melalui *foreign key*, tabel-tabel dapat dihubungkan untuk menunjukkan relasi antar entitas. Hal ini memungkinkan RDBMS untuk menjaga integritas data dan memastikan hubungan antar tabel tetap konsisten.
3. **Relasi (*Relationship*)**: Relasi adalah keterkaitan atau hubungan antara dua atau lebih tabel dalam basis data. Relasi dalam RDBMS dapat berupa:
    a. **One-to-One**: Setiap entitas di tabel pertama hanya memiliki satu hubungan dengan entitas di tabel kedua, dan begitu pula sebaliknya.
    b. **One-to-Many**: Satu entitas di tabel pertama dapat terhubung dengan beberapa entitas di tabel kedua, tetapi setiap entitas di tabel kedua hanya terhubung dengan satu entitas di tabel pertama.
    c. **Many-to-Many**: Banyak entitas di tabel pertama dapat berhubungan dengan banyak entitas di tabel kedua, dan hubungan ini bersifat dua arah.

2.2.7 Black Box Testing

Blackbox Testing adalah pendekatan pengujian perangkat lunak yang menilai fungsionalitas aplikasi tanpa memperhatikan struktur internal atau implementasi kode. Dalam *black-box testing*, penguji tidak memiliki pengetahuan tentang bagaimana sistem dikembangkan atau detail teknis dari kode yang digunakan. Pengujian ini terutama digunakan untuk memastikan bahwa aplikasi bekerja sesuai dengan spesifikasi atau persyaratan yang ditetapkan (Myers et al., 2011).

Black-box testing menekankan pada pengujian masukan dan keluaran aplikasi, di mana penguji memberikan data input tertentu dan memverifikasi bahwa hasil yang diperoleh sesuai dengan harapan atau spesifikasi. Tujuan utama dari *black-box testing* adalah untuk mendeteksi kesalahan

fungsionalitas, seperti kesalahan dalam penanganan data, perhitungan, validasi input, dan output yang tidak sesuai.

Black-box Testing tidak memerlukan pemahaman tentang kode atau logika internal perangkat lunak yang diuji. Pengujian ini didasarkan pada persyaratan fungsional dan spesifikasi perangkat lunak, dengan pendekatan yang berfokus pada apa yang seharusnya dilakukan oleh sistem, bukan bagaimana cara kerjanya (Myers et al., 2011). Teknik ini cocok untuk pengujian validasi, di mana penguji memeriksa apakah perangkat lunak memenuhi persyaratan dan harapan pengguna akhir.

Dalam *black-box testing*, beberapa teknik umum digunakan untuk merancang kasus uji yang efektif, di antaranya adalah:

1. Pengujian Berbasis Persyaratan (*Requirement-Based Testing*): Teknik ini melibatkan pembuatan kasus uji berdasarkan persyaratan fungsional yang dijelaskan dalam dokumentasi perangkat lunak. Kasus uji dirancang untuk memverifikasi bahwa setiap fungsi yang diuraikan dalam spesifikasi bekerja dengan benar.
2. Pengujian Berdasarkan Kasus Penggunaan (*Use Case Testing*): Teknik ini menggunakan use case untuk memahami bagaimana perangkat lunak seharusnya berperilaku dari sudut pandang pengguna. Setiap use case mewakili skenario yang berbeda, di mana pengguna berinteraksi dengan perangkat lunak, sehingga membantu penguji memastikan bahwa perangkat lunak merespons dengan benar dalam berbagai situasi.
3. Pembagian Kelas Ekivalen (*Equivalence Partitioning*): Teknik ini membagi data input ke dalam beberapa kelas ekivalen atau kelompok yang dianggap menghasilkan output yang sama. Hanya satu kasus uji dari setiap kelas yang diuji untuk mengurangi jumlah pengujian yang harus dilakukan, namun tetap memastikan cakupan pengujian yang memadai.
4. Analisis Nilai Batas (*Boundary Value Analysis*): Teknik ini menguji nilai-nilai di batas antara kelas ekivalen, yaitu pada titik terendah dan

tertinggi dari setiap kelas. Karena kesalahan sering kali terjadi pada batasan data, teknik ini efektif untuk mendeteksi cacat di area tersebut.

Beberapa jenis *black-box testing* yang umum digunakan dalam pengembangan perangkat lunak meliputi:

1. *Functional Testing*: Pengujian ini fokus pada validasi fungsi perangkat lunak sesuai dengan spesifikasi. Functional testing mencakup pengujian antarmuka, database, dan alur kerja aplikasi.
2. *Non-Functional Testing*: Pengujian ini berfokus pada aspek non-fungsional dari perangkat lunak, seperti performa, keandalan, dan kegunaan. Non-functional testing sering mencakup pengujian beban, stres, dan keamanan untuk memastikan aplikasi bekerja optimal di berbagai kondisi.
3. *Regression Testing*: Jenis pengujian ini dilakukan setelah perbaikan atau modifikasi perangkat lunak untuk memastikan bahwa perubahan tersebut tidak memengaruhi fungsionalitas lain dalam sistem. Regression testing penting untuk menjaga kualitas perangkat lunak selama siklus pengembangan.
4. *Acceptance Testing*: Pengujian ini adalah tahap terakhir sebelum perangkat lunak diserahkan ke pengguna akhir. *Acceptance testing* dilakukan untuk memeriksa apakah perangkat lunak memenuhi kebutuhan dan harapan pengguna dan sering kali melibatkan pengguna akhir atau pemangku kepentingan lainnya.

2.3 Analisa PIECES

Analisis PIECES adalah metode yang digunakan dalam evaluasi sistem informasi untuk mengidentifikasi aspek-aspek yang perlu ditingkatkan dalam suatu sistem. PIECES merupakan akronim dari enam dimensi kunci: *Performance* (Kinerja), *Information* (Informasi), *Economics* (Ekonomi), *Control* (Pengendalian), *Efficiency* (Efisiensi), dan *Services* (Layanan). Analisis PIECES adalah alat yang efektif dalam mengevaluasi kinerja sistem informasi karena memfokuskan pada aspek fungsional dan operasional

sistem, memungkinkan organisasi untuk mengenali area yang membutuhkan perbaikan atau pengembangan lebih lanjut (Bentley et al, 2000).

1. ***Performance*** **(Kinerja)**

   Kinerja dalam analisis PIECES merujuk pada seberapa baik sistem menjalankan fungsinya dan seberapa cepat sistem dapat memproses tugas yang diminta. Evaluasi kinerja sistem mencakup analisis terhadap waktu respon, kecepatan pemrosesan, serta kapasitas sistem dalam menangani jumlah pengguna atau data yang besar. meningkatkan kinerja sistem informasi sering kali melibatkan optimalisasi perangkat keras, perangkat lunak, atau prosedur operasional untuk memastikan sistem mampu menangani beban kerja dengan efektif. Kinerja yang buruk dapat menyebabkan penurunan produktivitas dan pengalaman pengguna yang kurang optimal.

2. ***Information*** **(Informasi)**

   Aspek informasi dalam analisis PIECES berfokus pada kualitas dan keakuratan data yang dihasilkan oleh sistem. Informasi yang baik harus relevan, akurat, tepat waktu, dan dapat dipahami oleh pengguna. Sistem informasi yang memberikan informasi yang berkualitas akan membantu pengguna dalam pengambilan keputusan yang lebih baik dan meningkatkan produktivitas. Informasi yang tidak akurat atau terlambat dapat menyebabkan pengambilan keputusan yang salah, yang berdampak negatif pada operasional dan keberhasilan organisasi.

3. ***Economics*** **(Ekonomi)**

   Dimensi ekonomi dalam analisis PIECES menilai biaya-biaya yang terkait dengan pengembangan, pemeliharaan, dan operasional sistem. Evaluasi ekonomi mencakup analisis terhadap biaya investasi awal, biaya operasional berkelanjutan, serta potensi penghematan biaya yang dihasilkan dari penerapan sistem. Analisis ini juga mempertimbangkan manfaat ekonomi yang dapat diperoleh organisasi, seperti peningkatan produktivitas atau efisiensi yang mengarah pada pengurangan biaya.

4. ***Control*** **(Pengendalian)**

Pengendalian dalam analisis PIECES adalah aspek yang terkait dengan kemampuan sistem dalam mengendalikan akses terhadap data dan operasional sistem. Sistem yang memiliki kontrol yang baik akan mampu melindungi data dari akses yang tidak sah serta menjaga integritas dan keamanan informasi. Sistem pengendalian yang baik sangat penting untuk menjaga data tetap aman dan mengurangi risiko kebocoran atau manipulasi data.

5. ***Efficiency* (Efisiensi)**

Efisiensi dalam analisis PIECES merujuk pada kemampuan sistem dalam memanfaatkan sumber daya secara optimal, baik sumber daya manusia, perangkat keras, maupun perangkat lunak. Efisiensi tinggi berarti sistem dapat menyelesaikan tugas dengan sumber daya minimal, mengurangi waktu dan biaya yang diperlukan. Sistem yang efisien mengurangi pemborosan sumber daya dan meningkatkan produktivitas organisasi secara keseluruhan.

6. ***Services* (Layanan)**

Layanan atau services dalam analisis PIECES berfokus pada kemampuan sistem dalam menyediakan layanan yang memenuhi kebutuhan pengguna, termasuk layanan purna jual, dukungan teknis, dan ketersediaan sistem untuk diakses kapan saja. Layanan yang baik berarti sistem memberikan akses yang andal dan memberikan dukungan yang memadai kepada pengguna. Kepuasan pengguna akhir sangat bergantung pada layanan yang disediakan oleh sistem, termasuk dukungan teknis dan pelatihan pengguna yang memadai.

2.4 Tinjauan Studi

Dalam penulisan penelitian ini dilakukan perbandingan dengan beberapa penelitian sebelumnya antara lain

1. Tinjauan studi pertama diambil dari penelitian yang berjudul "**Rancang Bangun Sistem E-commerce Clothing Store dan Modul Custom Design Lab Terintegrasi**" yang ditulis oleh Paulus Lucky Tirma Irawan dan David Rozando pada SMATIKA Jurnal Volume 10, No. 1, Juni Tahun 2020. Penelitian ini berfokus pada pengembangan sistem *E*-

commerce yang menyediakan fitur kustomisasi produk bagi konsumen, terutama dalam toko pakaian online. Modul utama yang dibahas dalam penelitian ini adalah *Custom Design Lab*, sebuah fitur yang memungkinkan konsumen membuat desain pakaian mereka sendiri secara *online*, yang kemudian diintegrasikan dengan proses pemesanan hingga pengiriman barang. Dalam mengembangkan sistem ini, penelitian menggunakan framework PyroCMS sebagai basis pengembangan aplikasi dan menerapkan konfigurasi modul yang mendukung fitur *E-commerce* secara lengkap.

Pada bab metodologi, penelitian ini menjelaskan langkah-langkah pengembangan sistem, termasuk perancangan alur bisnis E-commerce yang mencakup proses registrasi, pemesanan, negosiasi harga, pembayaran, dan pengiriman. Hasil penelitian menunjukkan bahwa sistem yang dirancang mampu menyediakan fitur yang mendukung interaksi konsumen dengan sistem secara fungsional, termasuk fitur *Custom Design Lab* yang berhasil diimplementasikan untuk memfasilitasi permintaan kustomisasi produk. Dengan demikian, penelitian ini berhasil mengembangkan sistem yang tidak hanya memenuhi kebutuhan dasar *E-commerce* tetapi juga menghadirkan inovasi dalam bentuk kustomisasi produk, yang merupakan nilai tambah bagi pengguna. Selain itu, penelitian ini juga mengaitkan temuan dengan studi terdahulu, memperkaya pemahaman tentang pengembangan fitur *E-commerce modern* dengan elemen kustomisasi. Kesimpulan dari penelitian ini menekankan bahwa integrasi antara E-commerce dan fitur kustomisasi produk dapat meningkatkan kepuasan pengguna dan memberikan keunggulan kompetitif bagi toko pakaian online.

2. Tinjauan studi kedua diambil dari penelitian yang berjudul "**Pengembangan Aplikasi Berbasis Web untuk Pemesanan Jasa Sablon Pakaian**" yang ditulis oleh Tiffani Maria Bernadetta dan Arnold Aribowo pada jurnal Seminar Nasional TEKNOKA Vol. 5, tahun 2020. Penelitian ini bertujuan untuk mengembangkan aplikasi

berbasis web yang dapat memfasilitasi proses pemesanan jasa sablon pakaian, baik untuk produk siap pakai maupun untuk pesanan khusus. Latar belakang penelitian ini adalah untuk mengatasi kendala efisiensi dalam proses pemesanan jasa sablon yang umumnya dilakukan secara langsung di toko atau melalui pesan WhatsApp, yang menyebabkan kesulitan dalam pencatatan pesanan ketika jumlah pesanan meningkat. Dengan menggunakan metode pengembangan sistem waterfall, penelitian ini merancang aplikasi yang dapat meningkatkan efisiensi pemesanan dengan menyediakan antarmuka yang memudahkan pengguna dalam membuat pesanan serta memonitor status pesanan mereka.

Penelitian ini menggunakan *framework* CodeIgniter berbasis PHP dan menerapkan berbagai pemodelan diagram, seperti *use case, activity diagram, class diagram, dan entity relationship diagram* untuk mendeskripsikan alur dan struktur sistem. Hasil dari penelitian ini adalah aplikasi web yang memiliki fitur utama, termasuk halaman pemesanan untuk produk sablon siap pakai dan kustom, konfirmasi pembayaran, pengelolaan inventaris, dan pengelolaan data pengguna bagi admin. Berdasarkan pengujian yang dilakukan dengan metode *blackbox testing*, aplikasi ini terbukti berhasil mendukung kebutuhan transaksi penjualan sablon secara efektif.

3. Tinjauan studi ketiga diambil dari penelitian yang berjudul "**Pengembangan Sistem Manajemen Perusahaan Sablon Kaos Berbasis Website Menggunakan Metode Prototyping (Studi Kasus: Perusahaan Sablon di Kota Malang)**" yang ditulis oleh Muhammad Rayyan Abhad, Achmad Arwan, dan Djoko Pramono pada Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer Vol. 3, No. 9, September Tahun 2019. Penelitian ini berfokus pada pengembangan sistem manajemen berbasis web untuk perusahaan sablon kaos di Kota Malang, yang menghadapi masalah dalam proses pemesanan, produksi, dan pencatatan keuangan. Metode *prototyping* dipilih karena kebutuhan pengguna belum terdefinisi dengan jelas, memungkinkan pengguna

untuk berinteraksi langsung dengan prototipe sistem sehingga kebutuhan yang spesifik dapat teridentifikasi secara bertahap. Sistem ini dirancang untuk mempermudah manajemen proses pemesanan, pelacakan produksi, dan pengelolaan keuangan perusahaan sablon.

Penelitian ini melalui beberapa tahapan, termasuk analisis kebutuhan dengan wawancara yang menghasilkan 30 kebutuhan *fungsional* dan 1 kebutuhan *non-fungsional*, serta desain sistem yang mencakup *sequence diagram*, *class diagram*, dan *entity relationship diagram*. Implementasi sistem dilakukan menggunakan PHP, MySQL, CodeIgniter, dan API RajaOngkir untuk memfasilitasi pengiriman. Hasil pengujian menunjukkan bahwa sistem berfungsi sesuai dengan kebutuhan, dengan hasil valid pada pengujian unit, integrasi, validasi, dan kompatibilitas. Pengujian validasi menggunakan metode *blackbox testing* menunjukkan bahwa sistem memenuhi semua skenario uji.

4. Tinjauan studi keempat diambil dari penelitian yang berjudul "**Aplikasi Toko Kardus Online untuk Kustomisasi Kardus dan Manajemen Stok pada PT XYZ**" yang ditulis oleh Willy Fransidy dan Arief Agus Sukmandhani pada Jurnal Komunikasi Sains dan Teknologi Vol. 2, No. 2, Desember Tahun 2023. Penelitian ini bertujuan untuk mengembangkan aplikasi berbasis web yang dapat memfasilitasi pemesanan kustom kardus serta manajemen stok secara digital untuk PT XYZ. Latar belakang penelitian ini adalah adanya masalah komunikasi dan pencatatan stok yang kurang akurat di perusahaan, sehingga mengakibatkan kendala ketika konsumen ingin memesan kardus yang ternyata tidak tersedia. Dengan menggunakan metodologi Scrum, pengembangan aplikasi dilakukan secara iteratif, memungkinkan pembaruan yang cepat dan respon yang lebih baik terhadap kebutuhan perusahaan.

   Hasil penelitian menunjukkan bahwa aplikasi yang dikembangkan dapat memfasilitasi pemesanan kustom kardus dengan menyediakan antarmuka yang intuitif bagi konsumen serta fitur manajemen stok bagi pihak perusahaan. Implementasi sistem meliputi

halaman login, *dashboard* untuk admin, laporan pesanan, serta halaman *checkout* yang memungkinkan proses transaksi lebih mudah dan terintegrasi. Pengujian dilakukan dengan metode *blackbox testing* yang menunjukkan bahwa aplikasi dapat berfungsi sesuai harapan pada berbagai skenario uji.

5. Tinjauan studi pertama diambil dari penelitian yang berjudul "**Online T-shirts Store: E-Commerce Web Application**" yang ditulis oleh Thinh Nguyen pada *Oulu University of Applied Sciences* sebagai Tugas Akhir Sarjana bidang Teknologi Informasi, tahun 2022. Penelitian ini berfokus pada pengembangan aplikasi web *E-commerce* untuk penjualan kaos yang menyediakan keranjang belanja bagi pelanggan. Latar belakang penelitian ini adalah meningkatnya popularitas platform *E-commerce B2C*, yang memungkinkan pelanggan untuk memilih dan membeli produk secara online. Tujuan utama dari penelitian ini adalah membangun aplikasi yang mudah digunakan dan dapat memenuhi kebutuhan dasar operasional toko *E-commerce* kaos dengan pengalaman antarmuka pengguna yang baik.

   Metode pengembangan yang digunakan mencakup teknologi seperti React.js untuk front-end, Node.js dan Express.js untuk back-end, serta MongoDB sebagai basis data. Penelitian ini juga melibatkan pembuatan RESTful API untuk komunikasi data antara front-end dan back-end, yang memungkinkan pelanggan untuk menambahkan produk ke keranjang dan melakukan pembayaran. Hasil dari penelitian menunjukkan bahwa aplikasi yang dibangun berhasil menyediakan fitur utama seperti tampilan produk, fungsi keranjang belanja, dan autentikasi pengguna yang aman.

Berikut ini merupakan rangkuman yang mencakup tinjauan studi dari berbagai penelitian terdahulu yang relevan untuk mendukung landasan teori serta memperkuat arah dan fokus dari penelitian ini.

Tabel 2.6 *Literature Review*

No	Judul	Penulis, Media Publikasi, Tahun	Tujuan	Metodologi	Saran dan Kelemahan	Perbandingan
1	Rancang Bangun Sistem E-commerce Clothing Store dan Modul Custom Design Lab Terintegrasi	Paulus Lucky Tirma Irawan, David Rozando, SMATIKA Jurnal, 2020	Mengembangkan sistem E-commerce dengan fitur Custom Design Lab untuk toko pakaian online.	Menggunakan framework PyroCMS untuk pengembangan aplikasi dan modul Custom Design Lab berbasis ImageMagick.	• Kustomisasi desain masih bisa dikembangkan lebih interaktif. • Disarankan untuk menambah fitur integrasi dengan sosial media.	Menekankan kustomisasi produk dalam E-commerce pakaian, serupa dengan studi yang mengutamakan personalisasi produk, tetapi menggunakan PyroCMS sebagai basis pengembangan.
2	Pengembangan Aplikasi Berbasis Web untuk Pemesanan Jasa Sablon Pakaian	Tiffani Maria Bernadetta, Arnold Aribowo, Seminar Nasional TEKNOKA, 2020	Mengembangkan aplikasi pemesanan jasa sablon berbasis web untuk memudahkan proses pemesanan dan pencatatan transaksi.	Menggunakan metode pengembangan waterfall dengan beberapa diagram pemodelan seperti use case, activity diagram, dan class diagram.	• Sistem terbatas pada layanan sablon. • Disarankan menambah fitur estimasi waktu produksi.	Fokus pada pemesanan jasa kustom produk cetak (sablon), mirip dengan studi lainnya tentang pemesanan, tetapi dengan pendekatan use case yang lebih sederhana.
3	Pengembangan Sistem Manajemen Perusahaan Sablon Kaos Berbasis Website Menggunakan Metode Prototyping	Muhammad Rayyan Abhad, Achmad Arwan, Djoko Pramono, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2019	Merancang sistem manajemen berbasis web untuk perusahaan sablon kaos, khususnya untuk memudahkan pemesanan dan pelacakan produksi.	Menggunakan metode prototyping yang melibatkan wawancara untuk mengidentifikasi kebutuhan, diikuti dengan pengembangan dan pengujian prototipe.	• Sistem masih terbatas pada kebutuhan perusahaan sablon. • Disarankan untuk menambah fitur integrasi pembayaran online.	Fokus pada manajemen perusahaan sablon dengan fitur pencatatan keuangan, dibandingkan studi lain yang lebih berfokus pada fitur kustomisasi produk atau pesanan.
4	Aplikasi Toko Kardus Online untuk Kustomisasi Kardus dan	Willy Fransidy, Arief Agus Sukmandhani, Jurnal Komunikasi Sains	Mengembangkan aplikasi berbasis web untuk pemesanan kardus kustom dan	Menggunakan metodologi Scrum dengan tahapan iteratif untuk pengembangan modul pemesanan	• Sistem masih terbatas pada produk kardus. • Disarankan menambah fitur analisis stok	Menyediakan fitur manajemen stok yang lebih mendetail dibandingkan dengan studi lainnya yang

No	Judul	Penulis, Media Publikasi, Tahun	Tujuan	Metodologi	Saran dan Kelemahan	Perbandingan
	Manajemen Stok pada PT XYZ	dan Teknologi, 2023	pengelolaan stok di PT XYZ.	dan manajemen stok secara bertahap.	berdasarkan permintaan konsumen.	berfokus pada proses kustomisasi produk.
5	Online T-shirts Store: E-Commerce Web Application	Thinh Nguyen, Oulu University of Applied Sciences, 2022	Mengembangkan aplikasi E-commerce yang mendukung fungsi keranjang belanja dan kustomisasi produk.	Menggunakan RESTful API, front-end berbasis React.js, back-end dengan Node.js dan Express.js, serta MongoDB untuk basis data.	• Tidak ada integrasi dengan fitur kustomisasi produk. • Disarankan menambah fitur kustomisasi atau personalisasi produk untuk pengalaman pengguna yang lebih baik.	Mengutamakan fitur *E-commerce* dasar seperti keranjang belanja, berbeda dari studi lain yang memiliki fitur kustomisasi lebih canggih.