BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Pengertian Rancang Bangun

Menurut pendapat dari Muarie (2015) bahwa rancang bangun merupakan proses menciptakan dan mengembangkan sebuah aplikasi atau sistem yang sebelumnya belum pernah ada pada suatu instansi atau objek penelitian tersebut.

Gunawan et al (2020) menjelaskan bahwa Rancang bangun adalah proses menggambarkan, merencanakan, dan menyusun sketsa atau mengintegrasikan berbagai elemen yang terpisah menjadi satu kesatuan yang utuh dan berfungsi dengan baik. Secara sederhana, rancang bangun dapat diartikan sebagai kegiatan yang menerjemahkan hasil analisis menjadi sebuah paket perangkat lunak, baik untuk menciptakan sistem baru maupun untuk memperbaiki sistem yang sudah ada agar lebih optimal.

Menurut Pressman (2015) perancangan atau rancang adalah suatu proses sistematis yang bertujuan untuk menerjemahkan hasil analisis suatu sistem ke dalam bahasa pemrograman. Proses ini dilakukan untuk menggambarkan secara rinci dan mendalam bagaimana setiap komponen dalam sistem tersebut akan diimplementasikan.

Masih menurut Pressman (2015) pembangunan atau bangun sistem adalah suatu kegiatan yang bertujuan untuk menciptakan sistem baru atau melakukan perubahan, perbaikan, dan pembaruan terhadap sistem yang sudah ada secara menyeluruh.

Berdasarkan pendapat beberapa ahli di atas dapat disimpulkan bahwa rancang bangun adalah proses sistematis yang melibatkan perancangan dan pembangunan suatu aplikasi atau sistem, baik yang baru maupun yang sudah ada, dengan tujuan menciptakan atau memperbaiki sistem agar berfungsi secara optimal.

2.1.2 Pengertian Aplikasi

Menurut Juansyah (2015) aplikasi dapat diartikan sebagai sebuah program yang siap digunakan, dirancang untuk menjalankan fungsi tertentu guna memenuhi kebutuhan pengguna atau mendukung operasi aplikasi lainnya. Aplikasi ini ditujukan untuk target pengguna tertentu sesuai dengan tujuan pengembangannya. Menurut Kamus Komputer Eksekutif, aplikasi juga didefinisikan sebagai solusi pemecahan masalah yang memanfaatkan teknik pengolahan data. Biasanya, aplikasi ini berfokus pada komputasi atau pengolahan data yang diinginkan atau diharapkan untuk mencapai hasil tertentu.

Kamus Besar Bahasa Indonesia menjelaskan pengertian aplikasi adalah merupakan hasil penerapan dari sebuah desain sistem yang dirancang khusus untuk mengolah data. Proses pengolahan ini dilakukan dengan mengikuti aturan atau ketentuan yang ditetapkan dalam bahasa pemrograman tertentu.

Aplikasi juga dapat berfungsi sebagai alat produktivitas, menyediakan kemampuan untuk mengelola waktu, tugas, dan komunikasi, serta sebagai sarana hiburan, menyediakan akses ke konten multimedia, permainan, dan layanan streaming (Gani, 2021).

Berdasarkan pendapat beberapa ahli di atas dapat disimpulkan bahwa aplikasi adalah program perangkat lunak yang dirancang untuk melakukan tugas-tugas tertentu pada komputer atau perangkat digital, bertujuan memenuhi kebutuhan pengguna atau menyelesaikan masalah tertentu.

2.1.3 Pengertian Transportasi

Menurut Undang-Undang Nomor 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan, transportasi didefinisikan sebagai proses perpindahan orang atau barang dari satu lokasi ke lokasi lainnya dengan menggunakan kendaraan yang bergerak di ruang lalu lintas jalan. Transportasi memegang peran penting dalam kehidupan sehari-hari, baik untuk memenuhi kebutuhan mobilitas individu maupun mendukung aktivitas ekonomi, seperti distribusi barang dan jasa. Dengan adanya

transportasi, konektivitas antarwilayah menjadi lebih lancar, sehingga mendorong pertumbuhan sosial dan ekonomi masyarakat.

Menurut Kamus Besar Bahasa Indonesia (KBBI), transportasi adalah proses perpindahan manusia atau barang dari satu tempat ke tempat lain dengan menggunakan kendaraan, baik yang digerakkan oleh tenaga manusia maupun mesin. Kebutuhan akan transportasi, baik untuk mobilitas manusia maupun distribusi barang, merupakan hal yang tidak bisa dipisahkan dari kehidupan sehari-hari.

Dari beberapa penjelasan mengenai pengertian transportasi, dapat disimpulkan bahwa transportasi adalah proses perpindahan manusia atau barang dari satu lokasi ke lokasi lain dengan menggunakan kendaraan, baik yang digerakkan oleh tenaga manusia maupun mesin. Proses ini memegang peran penting dalam kehidupan sehari-hari, baik untuk memenuhi kebutuhan mobilitas individu maupun mendukung aktivitas ekonomi seperti distribusi barang dan jasa. Kebutuhan akan transportasi merupakan hal yang tidak terpisahkan dari kehidupan sehari-hari, karena memfasilitasi mobilitas dan distribusi yang efisien.

2.1.4 Pengertian *Website*

Website adalah koleksi halaman-halaman yang dapat diakses melalui internet yang berisi informasi, gambar, audio, video dan konten lainnya yang ditampilkan dalam bentuk multimedia. *Website* biasanya memiliki alamat URL unik yang digunakan untuk mengidentifikasi lokasi dan memudahkan pengguna untuk mengaksesnya melalui *browser web* (Anniss, 2014).

Menurut Abdullah (2018) *website* dapat didefinisikan sebagai sekumpulan halaman yang menyajikan informasi dalam bentuk data digital. Informasi ini dapat berupa teks, gambar, animasi, suara, video, atau kombinasi dari semuanya. Website disediakan melalui koneksi internet, sehingga memungkinkan siapa saja di seluruh dunia untuk mengakses dan melihatnya.

Menurut Sarwono (2015) *website* adalah sebuah media yang terdiri dari berbagai halaman berisi informasi. Informasi ini dapat diakses melalui

internet dan dinikmati oleh pengguna di seluruh dunia. Pada dasarnya, *website* merupakan sekumpulan kode yang berisi serangkaian perintah. Kode-kode ini kemudian diterjemahkan dan ditampilkan oleh peramban (*browser*) agar dapat dibaca dan dipahami oleh pengguna.

Dari beberapa penjelasan mengenai pengertian website, dapat disimpulkan bahwa *Website* adalah kumpulan halaman digital yang dapat diakses melalui internet, berisi informasi dalam berbagai bentuk seperti teks, gambar, audio, video, atau kombinasi multimedia. Setiap website memiliki alamat URL unik yang memudahkan pengguna untuk mengidentifikasi dan mengaksesnya melalui peramban web. Secara teknis, website terdiri dari serangkaian kode yang diterjemahkan oleh browser agar informasi dapat ditampilkan dan dipahami oleh pengguna. Sebagai media informasi, website memungkinkan akses global, sehingga siapa pun di seluruh dunia dapat melihat dan memanfaatkan konten yang disediakan.

2.1.5 Pengertian *Database*

Menurut Rahimi (2020), pangkalan data (dalam bahasa Inggris disebut *database*) adalah sekumpulan data yang tersusun secara terorganisir. Data ini biasanya disimpan dan diakses secara elektronik melalui sistem komputer. Ketika pangkalan data semakin kompleks, pengembangannya memerlukan teknik perancangan dan pemodelan yang lebih formal. Untuk mengelola pangkalan data, digunakan perangkat lunak khusus yang disebut Sistem Manajemen Basis Data (*Database Management System*) atau disingkat DBMS.

Menurut Connolly & Begg (2015), *database* adalah kumpulan data logis yang saling terhubung, dilengkapi dengan deskripsi atau penjelasan terkait. *Database* dirancang khusus untuk memenuhi berbagai kebutuhan informasi suatu organisasi.

Menurut Satzinger, Jackson, & Burd (2016), *database* adalah kumpulan data yang terintegrasi, terpusat, dan dikelola secara terstruktur. *Database* dirancang untuk menyimpan, mengatur, dan mengelola informasi secara efisien sehingga memudahkan pengguna dalam mengakses dan memanfaatkan data tersebut.

Dari pengertian-pengertian *database* menurut para ahli di atas, dapat disimpulkan bahwa *database* adalah kumpulan data yang terorganisir, terstruktur, dan saling terhubung, dirancang untuk menyimpan dan mengelola informasi secara efisien. Data tersebut disimpan secara elektronik dan dapat diakses melalui sistem komputer, dengan bantuan perangkat lunak khusus yang disebut Sistem Manajemen Basis Data (DBMS). Database dirancang untuk memenuhi kebutuhan informasi suatu organisasi, memungkinkan pengguna untuk menyimpan, mengatur, dan mengakses data dengan mudah. Semakin kompleks suatu database, semakin diperlukan teknik perancangan dan pemodelan yang formal agar data dapat dikelola secara efektif dan terintegrasi.

2.1.6 Pengertian *JavaScript*

Menurut Flanagan (2020), *JavaScript* adalah salah satu bahasa pemrograman utama yang digunakan dalam pengembangan *web*. Hampir sebagian besar situs *website* saat ini memanfaatkan *JavaScript* untuk meningkatkan fungsionalitas dan interaktivitasnya. Selain itu, semua *browser web* modern, baik di perangkat *desktop*, tablet, maupun ponsel, telah mendukung *JavaScript*, menjadikannya sebagai bahasa pemrograman yang paling populer dan banyak digunakan di dunia. *JavaScript* dikenal sebagai bahasa pemrograman yang dinamis dan bersifat interpretatif, artinya kode dapat dijalankan langsung tanpa perlu dikompilasi terlebih dahulu. Bahasa ini sangat fleksibel karena mendukung paradigma pemrograman berorientasi objek (OOP) dan pemrograman fungsional, sehingga cocok untuk berbagai jenis proyek pengembangan, mulai dari aplikasi *web* sederhana hingga kompleks.

Menurut Sunyoto (2007), *JavaScript* sangat populer di dunia internet dan dapat berfungsi dengan baik di hampir semua *browser web* terkenal, seperti Google Chrome, Internet Explorer (IE), Mozilla Firefox, Netscape, dan Opera. Kode *JavaScript* biasanya disisipkan ke dalam halaman *web* menggunakan tag <script>. Sebagai salah satu teknologi inti dari *World Wide Web*, *JavaScript* bekerja bersama HTML dan CSS untuk menciptakan pengalaman *web* yang dinamis. *JavaScript* memungkinkan

pengembang untuk membuat halaman web yang interaktif, sehingga menjadi komponen penting dalam pembuatan aplikasi *web* modern.

Awalnya, *JavaScript* hanya digunakan di sisi klien (*client-side*) dalam browser *web*. Namun, seiring perkembangan teknologi, mesin *JavaScript* kini juga diintegrasikan ke dalam berbagai perangkat lunak lain. Misalnya, *JavaScript* digunakan di sisi server (*server-side*) pada server *web* dan *database*, serta dalam aplikasi non-web seperti perangkat lunak pengolah kata dan pembaca PDF. Selain itu, *JavaScript* juga dapat berfungsi sebagai *runtime environment* yang memungkinkan pengembang untuk membuat aplikasi *desktop* dan *mobile*. Hal ini menjadikan JavaScript sebagai salah satu bahasa pemrograman yang paling serbaguna dan banyak digunakan di berbagai platform.

Dari pengertian-pengertian *JavaScript* menurut para ahli di atas, dapat disimpulkan bahwa *JavaScript* adalah bahasa pemrograman dinamis yang banyak digunakan dalam pengembangan *web* untuk meningkatkan fungsionalitas dan interaktivitas halaman *web*. Dukungan dari semua browser dan fleksibilitasnya dalam mendukung pemrograman berorientasi objek secara fungsional membuat *JavaScript* menjadi pilihan utama untuk proyek pengembangan *web* dan aplikasi modern.

2.1.7 Pengertian *Systems Development Life Cycle* (SDLC)

Menurut Prof. Dr. Sri Mulyani, AK., CA. (2017) SDLC adalah sebuah pendekatan sistematis yang digunakan untuk merancang, mengembangkan, dan mengimplementasikan suatu sistem. Metode ini merupakan serangkaian tahapan logis yang diterapkan oleh seorang pengembang sistem untuk menciptakan sistem informasi yang efektif dan efisien. Proses SDLC meliputi berbagai aspek penting, seperti pengumpulan kebutuhan (*requirements*), validasi sistem, pelatihan pengguna, serta kolaborasi dengan pemilik sistem untuk memastikan bahwa sistem yang dibangun sesuai dengan harapan dan kebutuhan pengguna.

Menurut Wasson (2016) SDLC terdiri dari serangkaian fase kerja yang terdefinisi dengan jelas. Fase-fase ini digunakan oleh para pengembang sistem untuk menciptakan dan menyampaikan sistem

informasi yang efektif. Mirip seperti proses produksi di jalur perakitan, tujuan SDLC adalah menghasilkan sistem berkualitas yang memenuhi harapan pengguna. Hal ini dicapai dengan memastikan sistem dikembangkan sesuai dengan kebutuhan (*requirements*), diselesaikan dalam waktu yang telah ditentukan, dan sesuai dengan perkiraan biaya yang telah direncanakan.

Dari pengertian-pengertian *Systems Development Life Cycle* (SDLC) menurut para ahli di atas, dapat disimpulkan bahwa SDLC adalah pendekatan sistematis yang terdiri dari serangkaian fase terdefinisi untuk merancang, mengembangkan, dan mengimplementasikan sistem informasi yang efektif dan efisien. Proses ini mencakup pengumpulan kebutuhan, validasi sistem, pelatihan pengguna, dan kolaborasi dengan pemilik sistem untuk memastikan hasil akhir sesuai dengan harapan, waktu, dan anggaran yang direncanakan. Tujuannya adalah menghasilkan sistem berkualitas tinggi yang memenuhi kebutuhan pengguna secara tepat.

2.1.8 Pengertian *Waterfall*

Menurut Tristianto (2018), Metode Waterfall adalah salah satu model pengembangan perangkat lunak yang bersifat berurutan, di mana setiap tahap harus diselesaikan sebelum melanjutkan ke tahap berikutnya, layaknya aliran air terjun. Model ini membagi proses pengembangan menjadi beberapa fase utama yang terstruktur dan sistematis, yaitu:

1. *Requirement* (Analisis Kebutuhan)

   Tahap awal di mana kebutuhan sistem dikumpulkan dan dianalisis untuk memastikan bahwa perangkat lunak yang akan dikembangkan sesuai dengan kebutuhan pengguna.
2. *System Design* (Perancangan Sistem)

   Pada tahap ini, arsitektur sistem mulai dirancang berdasarkan kebutuhan yang telah dikumpulkan, termasuk desain struktur data, antarmuka, dan komponen utama lainnya.
3. *Coding & Testing* (Pengkodean dan Pengujian)

Proses implementasi dilakukan dengan menulis kode program sesuai dengan desain yang telah dirancang, kemudian diuji untuk memastikan fungsionalitasnya berjalan dengan baik.

4. *Deployment* (Penerapan Program)

   Setelah pengujian berhasil, perangkat lunak diimplementasikan dan mulai digunakan oleh pengguna.

5. *Maintenance* (Pemeliharaan)

   Tahap terakhir yang melibatkan pemantauan, perbaikan bug, serta pengembangan lebih lanjut jika diperlukan untuk meningkatkan kinerja sistem.

Metode Waterfall banyak digunakan dalam proyek yang memiliki kebutuhan dan spesifikasi yang jelas sejak awal. Karena pendekatannya yang sistematis dan terdokumentasi dengan baik, metode ini cocok untuk proyek dengan cakupan yang stabil dan minim perubahan selama proses pengembangan.

2.1.9 Pengertian *Unified Modeling Language* (UML)

Menurut Munawar (2021), UML adalah salah satu alat bantu yang sangat andal dalam dunia pengembangan sistem berbasis objek. UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem untuk membuat cetak biru (blueprint) dari ide atau visi mereka dalam format yang standar dan mudah dipahami. Selain itu, UML juga dilengkapi dengan mekanisme yang efektif untuk berbagi (sharing) dan mengomunikasikan rancangan sistem kepada pihak lain, seperti tim pengembang, stakeholder, atau klien.

Menurut Booch (2005), UML adalah sebuah bahasa. Seperti halnya bahasa pada umumnya, UML terdiri dari elemen-elemen dasar (seperti kata-kata) dan aturan-aturan yang mengatur bagaimana elemen-elemen tersebut digabungkan untuk menciptakan komunikasi yang efektif. UML adalah bahasa pemodelan yang dirancang khusus untuk menggambarkan sistem, baik secara konsep maupun fisik. Dengan kata lain, UML membantu pengembang untuk menggambarkan struktur, perilaku, dan interaksi dalam sistem secara jelas dan terstruktur.

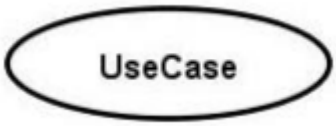
Dari pengertian-pengertian *Unified Modeling Language* (UML) menurut para ahli di atas, dapat disimpulkan bahwa UML adalah bahasa pemodelan visual standar yang digunakan untuk menggambarkan sistem berbasis objek secara terstruktur, baik secara konseptual maupun fisik. Dengan menyediakan elemen dasar dan aturan yang jelas, UML memungkinkan pengembang untuk membuat cetak biru sistem yang mudah dipahami dan dikomunikasikan kepada tim pengembang, stakeholder, atau klien.

Jenis-jenis *Unified Modeling Language* (UML) yaitu sebagai berikut:

a) *Use Case Diagram*

Use case diagram adalah sebuah diagram visual yang digunakan untuk menggambarkan fungsi atau layanan yang ada di suatu sistem, serta interaksi antara pengguna (*actor*) dengan sistem tersebut. Diagram ini membantu pengembang dan *stakeholder* untuk memahami bagaimana pengguna akan berinteraksi dengan sistem dan fitur-fitur apa saja yang dapat diakses. Di dalam use case terdapat simbol-simbol seperti di bawah ini:

- *Use Case*



Gambar 2.1 Simbol *Use Case*.
Sumber: Jurnal Hendini Ade (2016:108)

Use case merepresentasikan fungsi-fungsi yang ditawarkan suatu sistem dalam bentuk serangkaian unit yang saling berkomunikasi, baik antar-unit maupun dengan aktor. Setiap fungsi tersebut dinyatakan menggunakan kata kerja untuk menegaskan tindakan atau layanan yang diberikan.

- *Actor*/Aktor

Gambar 2.2 Simbol *Actor*.
Sumber: Jurnal Hendini Ade (2016:108)

Actor/Aktor adalah gambaran dari individu atau sistem eksternal yang memicu berfungsinya sistem target. Baik individu maupun sistem dapat menjalankan lebih dari satu peran. Perlu diingat, aktor hanya berinteraksi dengan *Use Case* dan tidak mengendalikan jalannya *Use Case* tersebut.

- *Association*/Asosiasi

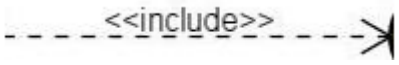
Gambar 2.3 Simbol *Association* Garis Lurus.
Sumber: Jurnal Hendini Ade (2016:108)

Asosiasi antara aktor dan *use case* digambarkan dengan garis lurus tanpa panah untuk menunjukkan pihak yang secara langsung memulai interaksi, bukan aliran data.

Gambar 2.4 Simbol *Association* Berpanah.
Sumber: Jurnal Hendini Ade (2016:108)

Sementara itu, garis dengan panah terbuka menandakan situasi ketika aktor hanya terlibat secara pasif dalam sistem.

- *Include*

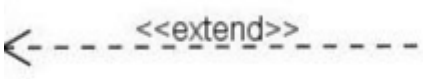


Gambar 2.5 Simbol *Include*.
Sumber: Jurnal Hendini Ade (2016:108)

Relasi *include* menyatakan bahwa suatu *use case* harus menyertakan perilaku *use case* lain; dengan kata lain, sebuah *use*

case memanggil *use case* lainnya seperti sebuah program mengeksekusi fungsi yang diperlukan.

- *Extend*



Gambar 2.6 Simbol *Extend*.
Sumber: Jurnal Hendini Ade (2016:108)

Relasi extend menggambarkan bagaimana sebuah use case dapat diperluas dengan skenario tambahan ketika suatu kondisi atau prasyarat tertentu terpenuhi. Dengan kata lain, use case utama menyediakan kerangka dasar alur kerja, sedangkan use case pelengkap akan dijalankan hanya jika situasi spesifik muncul.

b) *Activity Diagram*

Activity diagram adalah diagram visual yang digunakan untuk menggambarkan alur kerja atau proses bisnis dari serangkaian aktivitas yang dilakukan oleh pengguna (*user*) atau sistem. Diagram ini tidak hanya menunjukkan urutan langkah-langkah yang harus dijalankan, tetapi juga menggambarkan keputusan (*decision*), percabangan (*branching*), dan interaksi antar komponen dalam proses tersebut. Berikut simbol-simbol yang terdapat di activity diagram:

- Start Point

Gambar 2.7 Simbol *Start Point*.
Sumber: Jurnal Hendini Ade (2016:108)

Titik awal (*start point*) pada *activity diagram* diwujudkan sebagai lingkaran hitam kecil yang biasanya ditempatkan di sudut kiri atas, menandai titik masuk pertama ke dalam alur kerja. Simbol ini menunjukkan di mana eksekusi aktivitas dimulai dan dari sinilah semua aliran panah (transisi) pertama keluar menuju langkah-langkah berikutnya.Titik awal (start point) pada activity diagram diwujudkan sebagai lingkaran hitam kecil yang biasanya

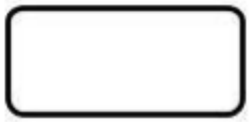
ditempatkan di sudut kiri atas, menandai titik masuk pertama ke dalam alur kerja. Simbol ini menunjukkan di mana eksekusi aktivitas dimulai dan dari sinilah semua aliran panah (transisi) pertama keluar menuju langkah-langkah berikutnya.

- *End Point*

Gambar 2.8 Simbol *End Point*.
Sumber: Jurnal Hendini Ade (2016:108)

Titik akhir (*end point*) dalam *activity diagram* ditandai dengan lingkaran ganda (*bullseye*) yang menandai selesainya seluruh rangkaian aktivitas. Simbol ini memberi tahu bahwa tidak ada transisi lebih lanjut setelah langkah terakhir dijalankan, sehingga alur proses berhenti secara definitif pada titik tersebut.

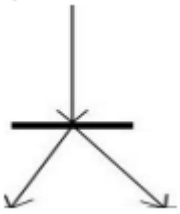
- *Activities*/Aktifitas



Gambar 2.9 Simbol *Activities*.
Sumber: Jurnal Hendini Ade (2016:108)

Simbol aktivitas pada activity diagram merepresentasikan satu unit pekerjaan atau langkah dalam suatu alur proses bisnis. Setiap aktivitas menggambarkan tindakan seperti “Verifikasi Data Pelanggan” atau “Proses Pembayaran”yang harus diselesaikan sebelum alur dapat berlanjut ke langkah berikutnya.

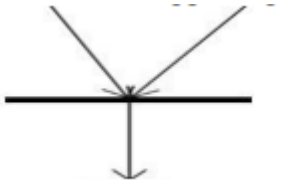
- *Fork*/Percabangan



Gambar 2.10 Simbol *Fork*.
Sumber: Jurnal Hendini Ade (2016:108)

Simbol *fork* pada activity diagram ditampilkan sebagai sebuah garis tebal (bar) yang memisahkan satu alur menjadi beberapa alur paralel, serta sebaliknya dapat menyatukan kembali alur-alur tersebut menjadi satu titik sinkronisasi.

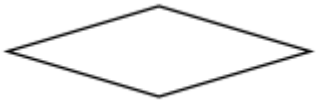
- *Join*/Penggabungan



Gambar 2.11 Simbol *Join*.
Sumber: Jurnal Hendini Ade (2016:108)

Simbol *Join*, sering juga disebut *rake*, digambarkan sebagai garis tebal horizontal atau vertikal yang menggabungkan beberapa alur paralel menjadi satu titik tunggal. Fungsinya adalah menyinkronkan aliran aktivitas yang menunggu hingga semua jalur paralel mencapai bar penggabungan sebelum melanjutkan ke langkah berikutnya.

- *Decision Point*/Pilihan



Gambar 2.12 Simbol *Decision Point*.
Sumber: Jurnal Hendini Ade (2016:108)

Decision point pada *activity diagram* diwakili oleh bentuk wajik (*diamond*) yang menunjukkan titik di mana alur proses bercabang berdasarkan kondisi tertentu, misalnya hasil evaluasi bernilai *true* atau *false*. Saat aliran mencapai simbol ini, sistem melakukan pengecekan logika misalnya “Apakah data valid?” lalu memilih jalur yang sesuai: jika kondisi terpenuhi, alur mengikuti cabang *true*; jika tidak, alur berlanjut melalui cabang *false* atau alternatif lain.

- *Swimlane*

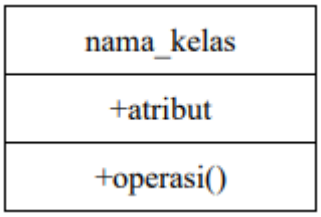
Gambar 2.13 Simbol *Swimlane*.
Sumber: Jurnal Hendini Ade (2016:108)

Swimlane pada *activity diagram* diilustrasikan sebagai kolom atau baris yang membagi diagram menjadi beberapa area, di mana masing-masing area (*lane*) mewakili peran, departemen, atau entitas yang bertanggung jawab atas serangkaian aktivitas.

c) *Class Diagram*

Class diagram adalah diagram visual yang digunakan untuk memetakan struktur kelas, atribut, metode, serta hubungan antar kelas dalam sebuah sistem aplikasi. Diagram ini tidak hanya menunjukkan bagaimana kelas-kelas saling berinteraksi, tetapi juga menjelaskan hierarki (seperti pewarisan/*inheritance*), asosiasi, atau ketergantungan antar kelas. Simbol-simbol yang terdapat di dalam class diagram adalah sebagai berikut:

- *Class*/Kelas



Gambar 2.14 Simbol *Class*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Kelas dalam *class diagram* merepresentasikan cetak biru (*blueprint*) bagi objek-objek yang akan dibangun oleh sistem; ia mendefinisikan atribut (data) dan operasi (metode) yang dimiliki setiap *instance*. Dengan menempatkan kelas-kelas pada diagram,

dapat dipetakan struktur statis sistem, siapa saja komponennya, informasi apa yang mereka simpan, serta metode apa yang dapat mereka kerjakan.

- *Interface*/Antarmuka

Gambar 2.15 Simbol *Interface*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Simbol *interface* pada *class diagram* berfungsi serupa dengan *interface* dalam pemrograman berorientasi objek, yaitu mendefinisikan sekumpulan operasi tanpa mengikat pada implementasi tertentu.

- *Association*/Asosiasi

Gambar 2.16 Simbol *Association*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Relasi asosiasi menghubungkan dua kelas untuk menunjukkan adanya keterkaitan fungsional atau struktural antara keduanya, di mana setiap sisi hubungan dapat memiliki aturan jumlah objek yang terlibat (*multiplicity*).

- *Multiplicity*/Multiplisitas

Gambar 2.17 Simbol *Multiplicity*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Multiplisitas atau Kardinalitas, merujuk pada berapa banyak satu entitas dapat terhubung atau berinteraksi dengan entitas lainnya.

Secara lebih sederhana, ini adalah jumlah minimum dan maksimum suatu objek yang dapat berelasi dengan objek lain.

- *Directed Association*/Asosiasi Berarah

Gambar 2.18 Simbol *Direct Association*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Relasi asosiasi antar kelas menggambarkan bahwa satu kelas memanfaatkan layanan atau atribut kelas lain dalam model, seolah-olah kelas pertama "menggunakan" kelas kedua. Visualisasinya berupa garis penghubung sering kali dilengkapi tanda panah untuk menunjukkan arah penggunaan yang di setiap ujungnya tertera notasi *multiplicity.*

- *Generalization*/Generalisasi

Gambar 2.19 Simbol *Generalization*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Relasi generalisasi pada *class diagram* menunjukkan hubungan "induk-anak" antara dua kelas, di mana kelas umum (*superclass*) mengekspresikan atribut dan perilaku dasar, sedangkan kelas khusus (*subclass*) mewarisi semua elemen tersebut sekaligus menambahkan atau memodifikasi fitur spesifik.

- *Aggregation*/Agregasi

Gambar 2.20 Simbol *Aggregation*.
Sumber: Sukamto dan Shalahuddin (2016:146)

Agregasi menggambarkan hubungan "*whole part*" antar kelas, di mana satu kelas berfungsi sebagai kesatuan (*whole*) yang terdiri dari satu atau beberapa bagian (*parts*) yang berdiri sendiri. Dalam

diagram, agregasi diwakili oleh batang garis yang di salah satu ujungnya terdapat diamond transparan mengarah ke kelas "*whole*"

d) *Sequence Diagram*

Sequence Diagram adalah diagram visual yang menggambarkan interaksi antara objek dalam suatu sistem atau proses tertentu. Diagram ini menampilkan urutan pesan yang dikirimkan antar objek, sehingga memberikan pemahaman yang lebih jelas mengenai bagaimana elemen-elemen dalam sistem saling berkomunikasi. Simbol-simbol yang terdapat di sequence diagram yaitu sebagai berikut:

- *Entity Class*/Kelas Entitas

Gambar 2.21 Simbol *Entity Class*.
Sumber: Jurnal Hendini Ade (2016:110)

Kelas entitas merepresentasikan komponen inti dalam sistem yang berisi kumpulan objek-objek seperti "Pengguna", "Produk", atau "Transaksi" yang menggambarkan struktur data dan hubungan antar entitas pada fase awal perancangan. Sebagai fondasi untuk menyusun *database*, setiap kelas entitas menjabarkan atribut dan relasi yang nantinya akan diterjemahkan ke dalam tabel dan kolom, sekaligus menjadi acuan bagi interaksi di diagram urutan (*sequence diagram*).

- *Boundary Class*/Kelas Pembatas

Gambar 2.22 Simbol *Boundary Class*.
Sumber: Jurnal Hendini Ade (2016:110)

Boundary class menggambarkan sekumpulan kelas yang bertugas sebagai antarmuka bagi aktor saat berinteraksi dengan sistem misalnya komponen tampilan form input, form laporan, atau dialog konfirmasi. Pada *sequence diagram*, *boundary class* muncul sebagai objek yang menerima permintaan dari aktor, memvalidasi

data masukan, lalu meneruskan pesan ke kontroler atau *entity class* untuk diproses lebih lanjut.

- *Control Class*/Kelas Kontrol

Gambar 2.23 Simbol *Control Class*.
Sumber: Jurnal Hendini Ade (2016:110)

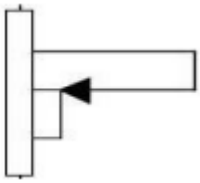
Control class berperan sebagai objek penengah yang memuat logika aplikasi seperti perhitungan, validasi, atau aturan bisnis kompleks tanpa terikat langsung pada penyimpanan data entitas. Dalam *sequence diagram*, *control class* muncul setelah *boundary class* menerima input dari aktor, kemudian meneruskan instruksi ke *entity class* sesuai kebutuhan.

- *Message*/Pesan

Gambar 2.24 Simbol *Message*.
Sumber: Jurnal Hendini Ade (2016:110)

Pesan pada *sequence diagram* diwakili oleh panah yang menunjuk dari satu objek ke objek lain untuk menggambarkan pengiriman informasi atau pemanggilan operasi. Setiap pesan menunjukkan jenis interaksi misalnya sinkron (pemanggilan metode yang menunggu balasan) atau asinkron (pengiriman tanpa menunggu) serta urutan kronologis antar objek.

- *Recursive*



Gambar 2.25 Simbol *Recursive*.
Sumber: Jurnal Hendini Ade (2016:110)

Relasi *recursive* pada *sequence diagram* terjadi ketika sebuah objek mengirimkan pesan kepada dirinya sendiri untuk menjalankan operasi berulang atau memecah tugas menjadi langkah-langkah internal. Dalam diagram, hal ini digambarkan sebagai panah melengkung yang kembali ke *lifeline* objek yang sama, sering kali diikuti oleh kotak eksekusi (*activation bar*) tambahan untuk menandakan bahwa objek itu menunda proses utama demi menyelesaikan sub-prosesnya.

- Activation

Gambar 2.26 Simbol *Activation*.
Sumber: Jurnal Hendini Ade (2016:110)

Activation pada *sequence diagram* ditunjukkan sebagai balok persegi panjang yang menempel pada *lifeline* objek, menggambarkan periode di mana objek tersebut sedang menjalankan sebuah operasi. Panjang balok ini sama dengan lamanya eksekusi semakin panjang balok maka semakin lama objek tersebut untuk memproses perintah.

- *Lifeline*

Gambar 2.27 Simbol *Lifeline*.
Sumber: Jurnal Hendini Ade (2016:110)

Lifeline di *sequence diagram* ditampilkan sebagai garis putus-putus vertikal yang menjulur ke bawah dari ikon objek, menggambarkan rentang waktu keberadaan objek tersebut selama

skenario dijalankan. Sepanjang *lifeline* inilah ditempatkan *activation bar* balok sempit yang menunjukkan kapan objek aktif memproses suatu operasi sehingga kita dapat melihat dengan jelas titik awal dan akhir eksekusi setiap metode.

2.2 Tinjauan Studi

Dalam penulisan skripsi ini, peneliti mengambil informasi dari berbagai penelitian sebelumnya sebagai bahan pembanding. Peneliti mengevaluasi kelebihan dan kekurangan dari penelitian-penelitian tersebut. Selain itu, peneliti juga mengumpulkan informasi dari berbagai sumber seperti buku dan skripsi lain untuk mendapatkan informasi tentang teori yang berkaitan dengan judul penelitian. Hal ini dilakukan untuk memperoleh dasar teori ilmiah yang kuat dan meyakinkan.

1. Penelitian yang telah dilakukan oleh Fenti Dwi Novianti (2019) membahas **Rancang Bangun Aplikasi Peminjaman Mobil Berbasis Website Di CV. Armada Jaya Trans.** Jurnal ini membahas pengembangan aplikasi peminjaman mobil berbasis web untuk mengatasi masalah pencatatan manual dan duplikasi data di CV. Armada Jaya Trans. Perusahaan rental skala menengah ini menghadapi duplikasi data dan kesalahan perhitungan biaya karena mengelola transaksi via Excel. Tujuannya adalah meningkatkan efisiensi pengelolaan data peminjaman, pengembalian, dan pelaporan melalui sistem terdigitalisasi. Metode yang digunakan adalah System Development Life Cycle (SDLC) dengan model waterfall, mencakup analisis kebutuhan, perancangan sistem (termasuk flowchart dan data flow diagram), serta implementasi. Proses yang ada meliputi booking, pinjam, pembayaran, pengembalian, validasi tanggal agar tidak bentrok, kalkulasi denda otomatis, cetak nota, hingga grafik tren peminjaman bulanan untuk owner. Aplikasi yang dibuat akan berjalan di platform web dengan menggunakan PHP (*backend*), MySQL (*database*), dan NetBeans IDE. PHP dan MySQL dipilih karena kemudahan integrasi dengan server lokal (XAMPP) dan kemampuan menangani data transaksional. Kesimpulan penelitian menyatakan

bahwa aplikasi ini berhasil mengurangi waktu pencatatan dan meminimalisir kesalahan laporan.

2. Agung Yulianto Nugroho (2018), membuat **Rancang Bangun Aplikasi Pemesanan Layanan Service Mobil Berbasis Website**. Penelitian ini mengembangkan aplikasi pemesanan layanan bengkel mobil berbasis web dengan model *Business-to-Customer* (B2C). Latar belakang masalah adalah promosi manual via brosur dan kesulitan pelanggan mengakses informasi layanan. Metode *SDLC waterfall* digunakan, dengan tahap analisis kebutuhan, desain UML (*use case diagram*, *activity diagram*), dan pengujian *blackbox*. Aplikasi ini menyediakan katalog suku cadang, pemesanan layanan servis berkala & emergency, keranjang belanja, checkout, riwayat order, dan panel admin (stok, harga, laba rugi). Aplikasi yang dibuat akan berjalan di platform web dengan menggunakan PHP (*backend*), MySQL (*database*), dan *framework* CSS untuk antarmuka. Hasilnya, aplikasi meningkatkan interaksi pelanggan-bengkel melalui pemesanan online dan transparansi harga.
3. Penelitian yang dilakukan oleh Ari Dwi Ani (2015) membahas **Perancangan Sistem Pemesanan (Order System) Mobil dan Follow Up Customers Berbasis Web dan Menggunakan SMS Gateway (Studi Kasus : PT. Nasmoco Salatiga)**. Jurnal ini merancang sistem pemesanan mobil dengan integrasi SMS gateway untuk PT. Nasmoco. Tujuannya menggantikan sistem manual berbasis kertas dan meningkatkan interaksi pelanggan via SMS. Metode *prototyping iterative* digunakan untuk desain berbasis UML, diikuti pengujian *blackbox*. Order mobil dapat dikirim via pola SMS tertentu; sistem mem-parse pesan, memvalidasi stok, lalu menghasilkan e-receipt otomatis. Aplikasi ini dibangun dengan PHP (*backend*), MySQL (*database*), dan Gammu SMS Gateway. Hasilnya, sistem memungkinkan pemesanan via SMS, pelacakan status, dan broadcast promo.

4. Penelitian yang dilakukan oleh Abdi Prialam (2025) membahas **Rancang Bangun Aplikasi Kendaraan Mobil Operasional Berbasis Mobile Menggunakan Metode Waterfall Pada Pt Pelabuhan Indonesia (Persero)** ini berfokus pada Studi ini menyoroti persoalan dokumen fisik (form order, kwitansi BBM, struk E-Toll) di lingkungan BUMN PT Pelabuhan Indonesia yang tersebar di banyak cabang. Dengan kerangka Waterfall, peneliti mengumpulkan kebutuhan via FGD bersama Divisi *General Affairs*, mendesain prototipe antarmuka di Figma, lalu membangun aplikasi mobile lintas *platform* menggunakan Flutter, Provider State-Management, RESTful API, dan database MySQL 8.
5. Penelitian kelima adalah penelitian yang dilakukan oleh Taufiqqurrachman, Amin Suyanto, Ari Rosemalatriasari, dan Dimpo Sinaga pada tahun 2024 dengan judul **Sistem Informasi Peminjaman Kendaraan Dinas Operasional Berbasis Web pada Direktorat Jenderal Energi Baru Terbarukan Dan Konservasi Energi.** Penelitian ini menjelaskan kebutuhan unit kerja pemerintah yang masih mengandalkan formulir kertas untuk setiap pengajuan kendaraan dinas sehingga proses persetujuan berjenjang memakan waktu yang lama, menimbulkan antrean, serta menyulitkan dalam melacak riwayat penggunaan kendaraan. Metode yang digunakan adalah Waterfall dimulai dari *requirement analysis* (wawancara staf sekretariat & observasi alur disposisi), perancangan UML (*use-case*, *activity*, *class*). Aplikasi yang dibuat akan berjalan di platform web dengan menggunakan CodeIgniter 3 (*frontend*), jQuery (*backend*), dan MySQL (*database*). Hasilnya, aplikasi menunjukkan penurunan waktu persetujuan, pengurangan kesalahan input, dan kepuasan pengguna pada survei pasca-pelatihan.