

BAB 2

TINJAUAN PUSTAKA

Pada bab ini berisi tentang uraian literatur dan referensi penelitian terdahulu yang telah dilaksanakan oleh peneliti sebelumnya. Studi pustaka yang dilakukan bersumber dari jurnal, artikel, dan laporan hasil penelitian yang berkorelasi dengan penelitian yang akan dilakukan.

2.1 Pencucian Kendaraan A&R

Pencucian Mobil A&R merupakan layanan cuci kendaraan yang mengutamakan kebersihan, kecepatan, dan kepuasan pelanggan. Didirikan dengan tujuan memberikan pengalaman pelayanan terbaik, A&R menyediakan berbagai jenis layanan, seperti cuci mobil biasa, cuci salju (snow wash), poles bodi, serta perawatan interior. Dengan lokasi yang strategis di Jl.I. Abdul Muis Sangatta utara, Kutai Timur, Kalimantan Timur. Didukung oleh tim profesional serta peralatan canggih, A&R menjadi pilihan utama bagi pemilik kendaraan yang menginginkan hasil cuci yang optimal dan aman untuk cat mobil. Dengan harga yang kompetitif dan pelayanan yang ramah, A&R terus berinovasi untuk memenuhi harapan pelanggan dan menjaga standar kualitas layanan yang tinggi.

2.2 Sistem Informasi berbasis *Web*

Sistem merupakan kumpulan elemen atau subjek yang saling berinteraksi dan bekerja sama sesuai aturan tertentu untuk mencapai tujuan yang telah ditetapkan. Setiap sistem memiliki ciri khas, termasuk komponen sistem, batas sistem, keluaran yang dihasilkan, serta tujuan yang ingin dicapai. Sementara itu, informasi adalah data yang telah diproses sehingga memiliki arti dan dapat dipahami oleh penerima informasi, serta membantu mengurangi ketidakpastian dalam pengambilan keputusan. Sistem informasi sendiri merupakan gabungan dari unsur-unsur manusia, perangkat keras, perangkat lunak, dan jaringan komunikasi yang berfungsi untuk mengumpulkan, mengolah, menyimpan, dan

menyebarkan informasi dalam suatu organisasi. *Website* atau situs web adalah kumpulan halaman yang dapat diakses melalui jaringan internet. Setiap *website* memiliki alamat unik yang dikenal sebagai URL, dan halaman-halamannya akan ditampilkan saat URL tersebut dibuka. *Website* berfungsi sebagai media untuk membagikan pengetahuan, minat, produk, maupun berbagai jenis informasi, yang dapat disajikan dalam bentuk teks, gambar, video, audio, atau file multimedia lainnya. Berdasarkan pengertian mengenai sistem informasi dan *website* yang telah dijelaskan sebelumnya, maka sistem informasi berbasis web dapat didefinisikan sebagai suatu kesatuan antara elemen-elemen dan jaringan komunikasi yang beroperasi melalui aplikasi berbasis internet, dengan fungsi untuk menyebarkan, mengolah, mengumpulkan, serta menyimpan informasi dalam suatu organisasi atau lembaga.

Adapun dibawah ini beberapa perbandingan sistem informasi berdasarkan jurnal dengan studi kasus pencucian kendaraan yang bisa dijadikan sebagai acuan dalam pembangunan sistem informasi pencucian A&R. Berikut adalah tabel 2.1.

Tabel 2.1 Perbandingan Sistem Informasi Pencucian Kendaraan.

Fitur Yang Tersedia	Penelitian Terdahulu					
	Jadianan Parhusip, 2017	Meriska Defriani, 2017	Hadiana, 2020	Cahyanti, 2021	Shadiq, 2019	Rhivant hio, 2018
Login	✓	✓	✓	✓	✓	✓
Melihat data karyawan	✓	x	✓	✓	✓	✓
Melihat data kendaraan	✓	✓	✓	✓	✓	✓

Melihat informasi pencucian	✓	✓	✓	✓	✓	✓
Menambah data karyawan	✓	✓	✓	✓	✓	✓
Menambah data kendaraan	✓	✓	✓	✓	✓	✓
Mengubah data karyawan	✓	✓	✓	✓	✓	✓
Mengubah data kendaraan	✓	x	x	x	✓	✓
Melihat laporan keuangan	✓	✓	✓	✓	✓	✓
Logout	✓	✓	✓	✓	✓	✓

2.3 *System Development Life Cycle (SDLC)*

Perangkat lunak dapat dianggap sebagai sebuah produk yang proses pembuatannya dimulai dari sebuah gagasan. Gagasan tersebut kemudian dituangkan ke dalam bentuk dokumen. Dokumen, diagram, atau perangkat lunak itu sendiri berfungsi sebagai artefak yang dihasilkan dalam setiap tahap, dan menjadi pedoman dalam proses pengembangan ke tahap berikutnya hingga akhirnya produk siap digunakan oleh pengguna. Proses bertahap ini dikenal dengan istilah Software Development Life Cycle (SDLC). SDLC merupakan rangkaian tahapan yang terorganisir, tersusun secara sistematis, dan memiliki

definisi yang jelas, yang dijalankan oleh analis sistem atau programmer dalam merancang dan membangun sebuah perangkat lunak (Ali, 2019). Tahap dalam Software Development Life Cycle didefinisikan sebagai :

1. *Requirement Analysis*

Tahapan ini bertujuan untuk mengidentifikasi kebutuhan dari berbagai pemangku kepentingan, serta merumuskan deskripsi sistem dalam bahasa yang dapat dipahami oleh semua pihak yang terlibat. Kebutuhan ini terbagi menjadi dua jenis: kebutuhan fungsional, yaitu fitur atau tindakan yang harus dilakukan oleh sistem, dan kebutuhan non-fungsional, seperti tingkat keandalan, jumlah pengguna yang didukung, serta desain antarmuka sistem.

2. *System Analysis*

Analisis sistem berfungsi untuk memahami kondisi sistem komputer yang ada dan bagaimana sistem tersebut akan berintegrasi dalam lingkungan bisnis. Proses ini dilakukan dengan mempelajari model dan alur kerja bisnis, biasanya melalui wawancara. Informasi mengenai cara kerja sistem dikumpulkan menggunakan use case atau model objek lainnya.

3. *Design System*

Tahapan desain menetapkan bagaimana sistem akan dijalankan dan bagaimana setiap komponennya diatur agar sistem dapat dikembangkan serta digunakan dengan efektif. Fokus utama dari tahap ini adalah mendefinisikan sistem secara menyeluruh sebagai dasar dalam pengembangan lebih lanjut.

4. *Implementation*

Tahapan implementasi melibatkan pembuatan sistem sebagai kumpulan komponen yang saling terhubung. Komponen-komponen baru digabungkan dengan bagian yang telah ada sebelumnya dan diuji untuk memastikan kesesuaiannya dengan spesifikasi yang telah dirancang.

5. *Testing*

Pada tahap ini, sistem diuji untuk memastikan bahwa semua kebutuhan, baik fungsional maupun non-fungsional, telah dipenuhi. Hasil pengujian ini memberikan umpan balik kepada tim pengembang mengenai kualitas perangkat lunak yang dibangun, sehingga perbaikan dapat dilakukan jika diperlukan.

6. *Deployment*

Deployment adalah tahap di mana sistem mulai diimplementasikan secara menyeluruh. Setiap tahap sebelumnya harus diselesaikan terlebih dahulu sebelum sistem benar-benar diterapkan dan digunakan.

7. *Maintenance*

Pemeliharaan bertujuan untuk memastikan sistem tetap beroperasi sesuai kebutuhan. Proses ini mencakup perbaikan atau perubahan kecil yang dilakukan seiring waktu agar sistem tetap relevan dan mampu beradaptasi dengan perkembangan yang terjadi. Singkatnya, maintenance mencakup pengembangan sistem berkelanjutan di masa mendatang.

Adapun dibawah ini perbandingan dari beberapa metode SDLC yang biasa digunakan, yang selanjutnya perbandingan ini akan menjadi acuan mengapa metode tersebut digunakan.

Tahapan Pengembangan Perangkat Lunak	Waterfall	Prototype	RAD
Perencanaan Sistem (Systems Planning)	Berawal dari kebutuhan	Berawal dari kebutuhan	Berawal dari kebutuhan
Analisis Sistem (Systems Analysis)	Kebutuhan data harus dianalisis di awal secara lengkap dan menyeluruh	Kebutuhan data dapat ditambah ataupun dikurangi sesuai dengan	Kebutuhan data dapat ditambah ataupun dikurangi sesuai dengan kebutuhan user, ketika dilakukan

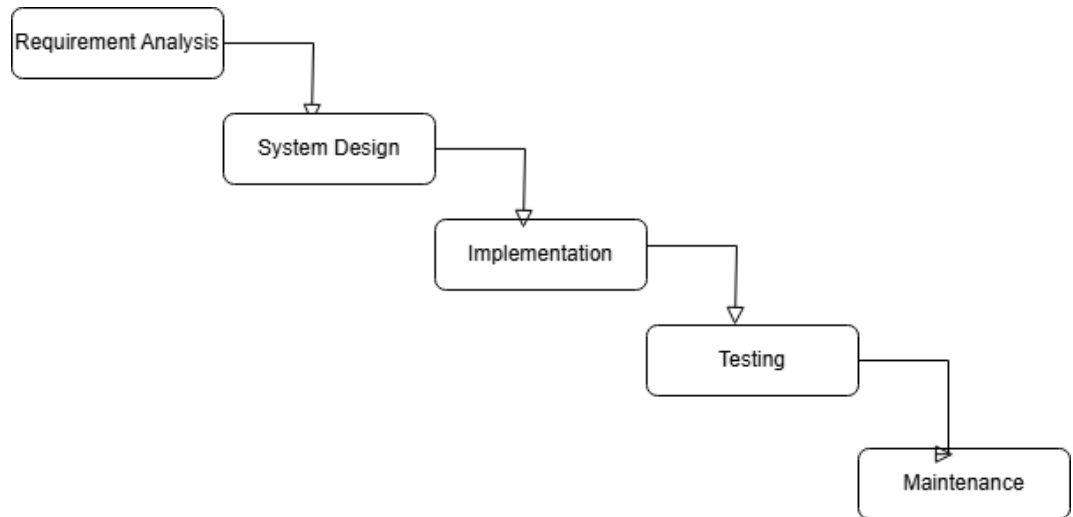
		kebutuhan user, ketika dilakukan testing	testing
	Perubahan data ataupun fungsional akan merubah ke seluruh proses pada tahapan berikutnya	Perubahan dapat dilakukan selama sistem atau perangkat lunak masih dalam bentuk prototype	Kebutuhan fungsi mayor dapat dimodelkan dalam waktu tertentu dan dapat dibicarakan oleh tim RAD yang terpisah
Perancangan Sistem (Systems Design)	Testing dilakukan ketika semua tahapan pada model sudah selesai.	Testing dapat dilakukan ketika prototype telah dibangun, sehingga hasil testing dapat merubah rancangan sistem	Testing dapat dilakukan ketika prototype telah dibangun, sehingga hasil testing dapat merubah rancangan sistem.
	Tidak dapat memberikan gambaran yang jelas mengenai sistem yang dibangun, karena sistem bisa dilihat jika semua tahapan telah dilakukan.	Memberikan prototype sebagai gambaran sistem yang akan dibangun, sehingga a user dapat melihat dan berinteraksi langsung dengan gambaran sistem.	Memberikan prototype sebagai gambaran sistem yang akan dibangun, sehingga user dapat melihat dan berinteraksi langsung dengan gambaran sistem.
		User berperan aktif dalam pengembangan sistem	User berperan aktif dalam pengembangan sistem
		Sistem yang dibangun akan sesuai dengan keinginan user	Sistem yang dibangun akan sesuai dengan keinginan user

			Mempunyai kemampuan untuk menggunakan kembali komponen yang ada (reusable object) sehingga pengembang tidak perlu membuat dari awal lagi dan waktu lebih singkat
Implementasi Sistem (Systems Implement)	Menerapkan proses perancangan yang baik	Tidak menerapkan proses perancangan yang baik	Tidak menerapkan proses perancangan yang baik
	Evaluasi dilakukan setelah sistem telah dibangun	Evaluasi dilakukan ketika prototype telah dibangun	Evaluasi dilakukan ketika prototype telah dibangun
	Mengedepankan kebutuhan fungsional sistem	Mengedepankan aspek kenyamanan user	Mengedepankan aspek kenyamanan user dan kecepatan pembangunan
Pemeliharaan Sistem (Systems Maintenance)	Dilakukan sesuai kesepakatan	Dilakukan sesuai kesepakatan	Dilakukan sesuai kesepakatan

2.4 Metode *Waterfall*

Waterfall merupakan salah satu model dalam *Software Development Life Cycle* (SDLC) yang banyak digunakan dalam proses pengembangan sistem informasi. Model ini bersifat sistematis dan berurutan karena setiap tahap dalam

metode *Waterfall* harus diselesaikan secara bertahap dan tidak dapat dilompati. Metode ini menggambarkan siklus pengembangan perangkat lunak yang dimulai dari tahap perancangan awal sebelum sistem diproduksi, hingga proses setelah sistem selesai dibuat dan digunakan.



Gambar 2.1 Alur Diagram *Waterfall* (Dwanoko, 2019)

Model *Waterfall* adalah salah satu pendekatan dalam SDLC yang kerap digunakan karena setiap tahapannya mudah dipahami dan diikuti. Model ini terdiri dari lima langkah utama, yaitu:

1. *Requirement Analysis*

Tahap ini bertujuan untuk mengidentifikasi kebutuhan sistem melalui proses pengumpulan dan analisis data yang terstruktur. Data dapat dikumpulkan melalui wawancara maupun penyebaran kuesioner. Analisis terstruktur digunakan untuk mengolah data yang telah dikumpulkan guna menemukan permasalahan dan merumuskan solusi yang tepat.

2. *System Design*

Pada tahap perancangan sistem, dilakukan pengembangan desain konseptual, rancangan basis data, serta desain antarmuka pengguna. Di tahap ini, dijelaskan secara rinci fungsi-fungsi sistem dan bagaimana

sistem akan berjalan, termasuk tata letak antarmuka pengguna, diagram alur proses, serta dokumentasi pendukung lainnya.

3. *Implementation*

Tahap implementasi mencakup proses pengkodean atau pembuatan program berdasarkan desain sistem yang telah dirancang sebelumnya.

4. *Testing*

Pada tahap pengujian, sistem diuji dengan menggunakan data nyata untuk memastikan bahwa aplikasi berjalan sesuai dengan harapan dan dapat menghasilkan output yang akurat.

5. *Maintenance*

Tahap ini bertujuan untuk memastikan sistem tetap berjalan optimal dan bebas dari gangguan atau kesalahan. Selain itu, juga termasuk proses serah terima sistem kepada pengguna atau pemangku kepentingan.

2.4.1 Dokumen Spesifikasi Kebutuhan Perangkat Lunak

Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) adalah dokumen yang merinci persyaratan yang harus dipenuhi dalam proses pembuatan atau pengembangan perangkat lunak. Dokumen ini mencakup dua kategori kebutuhan, yaitu kebutuhan pengguna dan kebutuhan sistem. SKPL berperan penting dalam tahap perancangan aplikasi, pelaksanaan, pemantauan proyek, serta proses verifikasi dan validasi (Sommerville, 2003). Adapun karakteristik dokumen SKPL yang baik akan dijelaskan yakni (IEEE, 1984) :

1. Unambiguous (Tidak Ambigu)

Dokumen SKPL harus dirumuskan secara jelas agar memiliki satu makna dan tujuan yang pasti mengenai sifat produk perangkat lunak. Jika terdapat istilah yang berpotensi memiliki banyak arti, maka istilah tersebut harus dimasukkan dalam daftar definisi dengan makna yang telah ditentukan secara khusus.

2. Complete (Lengkap)

Dokumen SKPL disebut lengkap jika mencakup semua kebutuhan yang berkaitan dengan hasil kerja perangkat lunak, termasuk fungsi, atribut, dan antarmuka eksternal. Dokumen ini juga harus menjelaskan respons perangkat lunak terhadap semua jenis input data dan sesuai dengan standar serta label yang ditentukan.

3. Verifiable (Dapat Diverifikasi)

SKPL dikatakan dapat diverifikasi jika setiap persyaratan dapat diuji kebenarannya. Persyaratan seperti “produk harus berjalan dengan baik” tidak dapat diverifikasi karena kata “baik” tidak terdefinisi secara objektif.

4. Consistent (Konsisten)

SKPL harus bebas dari pertentangan internal. Artinya, tidak boleh ada dua kebutuhan yang menggambarkan hal yang sama namun menggunakan istilah berbeda atau bertentangan dalam karakteristiknya.

5. Modifiable (Dapat Dimodifikasi)

SKPL harus mudah diperbarui. Struktur dokumen yang baik memungkinkan perubahan dilakukan secara rinci dan konsisten. Keberadaan indeks, daftar isi, dan referensi memudahkan proses modifikasi. Persyaratan yang sama tidak boleh muncul lebih dari satu kali dalam dokumen.

6. Traceable (Dapat Dilacak)

Setiap kebutuhan dalam SKPL harus memiliki sumber yang jelas. Terdapat dua jenis pelacakan: backward traceability untuk melacak ke dokumen sebelumnya, dan forward traceability untuk memastikan bahwa

setiap kebutuhan diikuti dalam dokumen pengembangan berikutnya. Kedua jenis pelacakan penting jika satu kebutuhan merepresentasikan kelompok kebutuhan lainnya.

7. Usable During the Operation and Maintenance Phase (Dapat Digunakan dalam Fase Operasi dan Pemeliharaan)

SKPL harus mendukung kebutuhan saat perangkat lunak dioperasikan dan dipelihara, termasuk ketika perangkat lunak mengalami perubahan. Pemeliharaan biasanya dilakukan oleh pihak yang bukan pengembang awal. Perubahan kecil dapat dilakukan dengan perbaikan kode, sedangkan perubahan besar memerlukan dokumentasi desain dan kebutuhan perangkat lunak. Pemahaman terhadap fungsi perangkat lunak sangat penting agar proses pemeliharaan berjalan efektif.

1. Pendahuluan

- a. Tujuan
- b. Definisi, Istilah, dan Singkatan
- c. Aturan Penomoran
- d. Referensi
- e. Deskripsi Umum Dokumen

2. Kebutuhan perangkat lunak

- a. Deskripsi Umum Sistem
- b. Kebutuhan Fungsional dan Non-Fungsional
- c. Batasan Sistem
- d. Lingkungan Sistem
- e. *Model Use Case*

3. Lampiran

2.4.2. Dokumen Deskripsi Perancangan Perangkat Lunak

Dokumen Deskripsi Perancangan Perangkat Lunak (DPPL) adalah representasi sistem perangkat lunak yang berfungsi sebagai sarana untuk menyampaikan informasi terkait desain perangkat lunak. Dokumen ini menggambarkan elemen-elemen perangkat lunak yang digunakan dalam proses analisis, perencanaan, implementasi, serta pengambilan keputusan (Society, 1998). DPPL ditujukan bagi manajer proyek, tim penjamin mutu, pemrograman, penguji, dan pengelola proyek agar mereka dapat meninjau apakah desain yang dijelaskan sudah lengkap, ringkas, konsisten, dan mudah dimengerti. Dalam penyusunannya, DPPL mencakup seluruh informasi yang diperlukan sebagaimana ditetapkan oleh IEEE (2009).

1. Pendahuluan
 - a. Tujuan
 - b. Ruang Lingkup
 - c. Aturan Penomoran
 - d. Referensi
 - e. Definisi Istilah
2. Model Analisis
 - a. *System Sequence* Tahap Analisis
 - b. *Class Diagram* Analisis
 - c. *Package Diagram*
 - d. Deskripsi Arsitektur
 - e. Pedoman Peancangan
3. Model Perancangan
 - a. *System Sequence* Tahap Perancangan
 - b. *Class Diagram* Tahap Perancangan
 - c. Perancangan Representasi Kelas Persisten
 - d. Perancangan Antar Muka

2.4.3 Dokumen Uji Perangkat Lunak

Dokumen Uji Perangkat Lunak (DUPL) adalah dokumen yang berisi rincian tentang rancangan pengujian dan berbagai jenis pengujian yang dilakukan terhadap perangkat lunak yang telah dirancang dan diimplementasikan. Dokumen ini berfungsi sebagai acuan teknis bagi para penguji atau peneliti perangkat lunak untuk memastikan bahwa perangkat lunak berjalan sesuai dengan harapan (Society, 1998).

DUPL membantu memastikan setiap bagian perangkat lunak diuji secara menyeluruh dan sesuai standar. Dalam proses penyusunannya, DUPL mencakup berbagai informasi penting sebagaimana diatur dalam standar IEEE (2008).

1. Pendahuluan
 - a. Tujuan
 - b. Ruang Lingkup
 - c. Aturan Penomoran
 - d. Referensi
2. Rencana dan Prosedur Pengujian
 - a. Rencana Pengujian
 - b. Prosedur Pengujian
3. Kasus Uji dan Hasil Uji

2.5 *Unified Modeling Language*

Unified Modeling Language (UML) merupakan bahasa standar yang digunakan dalam perancangan perangkat lunak. Rancangan yang dihasilkan dengan UML akan menggambarkan, menentukan, membangun, dan mendokumentasikan struktur serta arsitektur perangkat lunak yang akan dikembangkan. Penjelasan umum mengenai *Unified Modeling Language* (UML) disajikan pada Tabel 2.2 berikut. (Pressman, 2008)

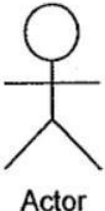
Tabel 2.2 Unified Modeling Language (UML)

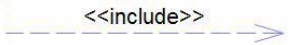
Area Utama	Tampilan	Diagram	Konsep Utama
Struktural	Static view	Class diagram	Class, association, generalization, dependency, realization, interface
	Use case view	Use case diagram	Use case, Use case generalization, actor, association, include, extend
	Implementation view	Component diagram	Dependency, Realization, Interface, Component
	Deployment view	Deployment diagram	Location, Component, Node, dependency
Dinamis	State machine view	Statechart diagram	Action, Event, State, Transition
	Activity view	Activity diagram	Join, Activity, State, Fork, Completion Transition
	Interaction view	Sequence diagram	Activation, Object Interaction, message,
		Collaboration diagram	Message, Interaction, Collaboration, collaboration role
Manajemen model	Model management view	Class diagram	Model, Subsistem, Package
Perpanjangan	Semua	Semua	Stereotype, Constraint

2.5.1 Use Case Diagram

Use case menggambarkan interaksi antara aktor dan sistem. Sebuah use case merujuk pada aktivitas spesifik yang dilakukan oleh aktor terhadap sistem, seperti melakukan login, memasukkan data ke dalam sistem, dan lain sebagainya. Use case menggambarkan hubungan antara satu atau lebih aktor dengan sistem. Pada tahap analisis kebutuhan, use case menggambarkan sistem dalam bentuk kotak yang berisi interaksi antara aktor dan sistem yang disajikan dalam bentuk naratif. (Kusbianto,2010). Beragam simbol yang digunakan dalam Use Case Diagram ditampilkan pada Tabel 2.3 berikut.

Tabel 2.3 Use case Diagram

Simbol	Nama	Deskripsi
	<i>Use case</i>	Use case menggambarkan fungsionalitas yang disediakan oleh sistem sebagai unit yang bertukar pesan antara <i>use case</i> dengan actor.
	<i>Actor</i>	Seorang aktor memulai sebuah <i>use case</i> , dan seorang aktor menerima sesuatu yang memiliki nilai. Untuk mengidentifikasi aktor, perlu menentukan pembagian tugas pada setiap <i>user</i> dari sistem. Aktor merupakan abstraksi dari orang atau sistem yang berinteraksi langsung dengan target sistem

Simbol	Nama	Deskripsi
	<i>Association Relationship</i>	Asosiasi antara actor dan <i>use case</i> yang digambarkan dengan garis lurus tanpa panah yang mengindikasikan adanya interaksi antar actor dan <i>use case</i> .
	<i>Generalization Relationship</i>	Asosiasi antara actor dan <i>use case</i> yang menggunakan simbol panah menandakan apabila actor berinteraksi secara langsung oleh sistem.
	<i>Include Relationship</i>	<i>Include</i> merupakan keadaan di mana suatu <i>use case</i> menggunakan fungsi dari <i>use case</i> lainnya
	<i>Extend Relationship</i>	<i>Extends</i> , yaitu suatu <i>use case</i> yang merupakan bentuk perluasan fungsi dari <i>use case</i> lainnya.

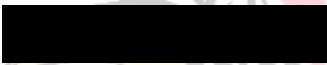
)*Dennis, Tegarden & Wixom, 2015

2.5.2 Activity Diagram

Activity diagram memberikan visualisasi tentang alur kerja atau langkah-langkah dalam sistem maupun tahapan bisnis yang ada dalam perangkat lunak. Hal yang perlu diperhatikan dalam activity diagram adalah bagaimana kegiatan dalam sistem dijelaskan, serta tindakan apa yang dilakukan oleh aktor

selama proses tersebut. Beragam simbol yang digunakan dalam Activity Diagram ditampilkan pada Tabel 2.4 berikut

Tabel 2.4 Activity Diagram

Simbol	Deskripsi
Status Awal 	Awal dari serangkaian Aktivitas
Aktivitas 	- Digunakan untuk merepresentasikan kumpulan action. - Diberi nama sesuai dengan nama activity-nya.
Percabangan 	Notasi yang menggambarkan ketika terdapat alternatif aktivitas melebihi satu
Penggabungan 	Notasi yang menggambarkan penggabungan melebihi satu aktivitas yang dikombinasikan menjadi satu
Status Akhir 	Notasi yang menggambarkan status akhir dalam aktivitas diagram

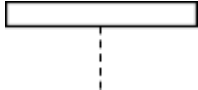
)*Dennis, Tegarden & Wixom, 2015

2.5.3 Sequence Diagram

Sequence diagram menggambarkan interaksi antara objek dan use case melalui deskripsi waktu hidup objek serta pesan yang dikirim dan diterima antar objek. Oleh karena itu, untuk membuat sequence diagram, penting untuk mengetahui berbagai objek yang terlibat dalam use case. Penyusunan sequence diagram diperlukan untuk menganalisis skenario yang terdapat dalam use case

diagram (Maiyendra, 2019). Beragam simbol yang digunakan dalam Use Case Diagram ditampilkan pada Tabel 2.4 berikut ;

Tabel 2.5 Sequence Diagram

Simbol	Deskripsi
<i>LifeLine</i> 	Menunjukkan interaksi suatu objek selama satu urutan.
Message 	Menggambarkan interaksi antar objek yang memuat berbagai informasi terkait aktivitas yang sedang berlangsung

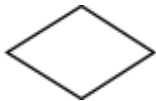
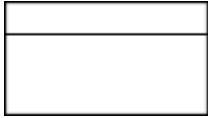
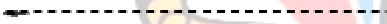
)*(Maiyendra, 2019)

2.5.4 Class Diagram

Class diagram berfungsi sebagai deskripsi dari kelompok objek, properti, tindakan, serta hubungan yang serupa. Dengan adanya class diagram, kita dapat memperoleh gambaran keseluruhan tentang sistem. Hal ini tercermin melalui berbagai kelas yang ada dan hubungan antar kelas tersebut. Biasanya, suatu sistem akan memiliki beberapa class diagram(Maiyendra, 2019). Beragam simbol yang digunakan dalam *Class Diagram* ditampilkan pada Tabel 2.6 berikut ;

Tabel 2.6 Class Diagram

Simbol	Deskripsi
<i>Generalization</i> 	Merepresentasikan hubungan antara berbagai class.

<i>Nary Association</i> 	Menggambarkan upaya guna mencegah terhadap asosiasi lebih dari dua objek
<i>Class</i> 	Menggambarkan pengelompokan berbagai objek yang memiliki kesamaan atribut maupun operasi.
<i>Collaboration</i> 	Menggambarkan urutan aksi yang dijalankan oleh sistem untuk menghasilkan keluaran yang terstruktur bagi seorang aktor.
<i>Realization</i> 	Menggambarkan operasi yang benar-benar dilakukan oleh suatu objek dalam sebuah sistem.
<i>Dependency</i> 	Menggambarkan hubungan perubahan pada elemen independen yang dapat mempengaruhi elemen lain yang bergantung padanya.
<i>Association</i> 	Menggambarkan hubungan antara satu objek dengan objek lainnya dalam suatu sistem.

)*(Maiyendra, 2019)

2.6 Entity Relationship Diagram

Entity Relationship (ER) adalah model database yang menggambarkan logika data yang mendukung kelangsungan bisnis. Entity Relationship terdiri dari sekumpulan objek serupa yang disebut entitas. Hubungan antar entitas ini dapat digambarkan dalam bentuk one-to-one (1:1), one-to-many (1:n), atau many-to-many (n:n). Dalam diagram Entity Relationship, simbol yang digunakan antara lain persegi panjang untuk entitas, elips atau oval untuk atribut, dan belah ketupat untuk hubungan atau relasi antar entitas. Komponen-komponen dalam diagram Entity Relationship meliputi:

1. Entitas

Entitas adalah objek fisik atau peristiwa yang diatur oleh pengguna untuk berada dalam alur sistem. Dengan kata lain, entitas mencakup semua informasi yang disimpan dalam sistem.

2. Atribut

Atribut merujuk pada karakteristik atau bagian dari sebuah entitas. Nilai-nilai dari bagian-bagian entitas ini disebut sebagai atribut entitas tersebut.

3. Relasi

Relasi menggambarkan bagaimana suatu entitas terhubung dengan entitas lainnya. Hubungan antar entitas dalam sistem database bisa berhubungan dengan bagian atau departemen dalam suatu organisasi. Organisasi juga seringkali membatasi hubungan antar entitas. Misalnya, dalam suatu sistem manajemen, entitas Security dapat mengakses data entitas CCTV, namun tidak dapat mengakses data entitas pegawai. Relasi antar entitas dapat dibedakan menjadi:

- a. One-to-one (1:1): Setiap entitas hanya memiliki satu hubungan dengan entitas lainnya.
- b. One-to-many (1:n): Sebuah entitas memiliki lebih dari satu hubungan dengan entitas lainnya.

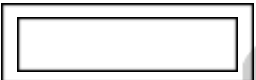
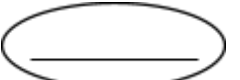
- c. Many-to-many (n:n): Setiap entitas memiliki relasi dengan entitas lainnya, dan sebaliknya.

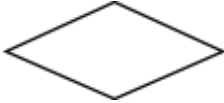
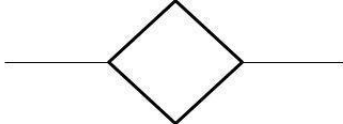
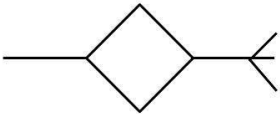
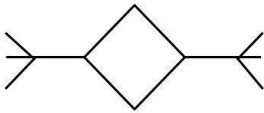
4. Key (Kunci)

Key adalah data yang digunakan komputer untuk mengenali kesatuan data dalam sistem. Key mendefinisikan kombinasi atribut dari sebuah entitas untuk mengidentifikasi item dalam dataset. Jenis-jenis kunci yang umum digunakan antara lain primary key, yang merupakan data unik untuk mengidentifikasi setiap entitas, seperti ID atau nomor induk (Singh, 2011).

Beragam simbol yang digunakan dalam *Entity Relationship Diagram* ditampilkan pada Tabel 2.7 berikut ;

Tabel 2.7 Entity Relationship Diagram

Simbol	Deskripsi
	Entitas adalah objek baik secara fisik maupun konseptual, yang terdapat di dunia nyata dan dapat diidentifikasi secara unik.
	Entitas lemah adalah entitas yang keberadaannya bergantung pada entitas lain untuk dapat diidentifikasi secara lengkap.
	Atribut, yaitu karakteristik dari entitas atau relasi
	Atribut <i>multivalued</i> , yaitu karakteristik dari suatu entitas atau relasi yang memiliki sekelompok nilai
	Atribut <i>primary key</i> , yaitu atribut entitas yang digunakan untuk membedakan set entitas secara unik

Simbol	Deskripsi
	Relasi, yaitu hubungan antara satu entitas dengan entitas lainnya
	<i>One-to-one</i> (1:1), yaitu hubungan antara satu entitas dengan satu entitas
	<i>One-to-many</i> (1:n), yaitu hubungan antara satu entitas dengan beberapa entitas
	<i>Many-to-many</i> (n:n), yaitu entitas yang memiliki lebih dari satu hubungan dengan entitas lainnya

2.7 Codeigniter

CodeIgniter (CI) adalah sebuah framework PHP open-source yang menggunakan pendekatan Model, View, Controller (MVC) dalam pengembangan proyek web. Dengan pendekatan ini, proses pembangunan web dapat dilakukan dengan lebih cepat. Selain itu, CodeIgniter juga dilengkapi dengan sistem keamanan yang kuat dan memiliki waktu muat yang cepat di browser web.

Tujuan utama dari aplikasi CodeIgniter adalah untuk membantu developer dalam mengembangkan proyek mereka dengan lebih cepat. Menurut pengembang internal CodeIgniter, proses pengembangan biasanya memakan waktu lama karena developer harus menulis kode dari awal. Untuk mengatasi hal ini, CodeIgniter menyediakan pustaka bawaan (built-in library) yang berisi

raturan template dan solusi untuk tugas-tugas umum yang biasa dikerjakan oleh developer. Selain itu, CodeIgniter juga menawarkan antarmuka pengguna (UI) dan struktur yang sederhana serta logis untuk mengakses pustaka bawaan tersebut. Singkatnya, CodeIgniter memungkinkan developer untuk lebih fokus pada proyek mereka dengan mengurangi jumlah kode yang diperlukan untuk berbagai tugas.

2.8 *Black Box Testing*

Black box testing adalah metode pengujian yang berfokus pada pendeteksian kesalahan fungsi dalam suatu program dengan cara membandingkan antara data masukan (*input*) dan keluaran (*output*) yang dihasilkan. Sementara itu, black box testing juga merupakan jenis pengujian perangkat lunak yang bertujuan untuk mengevaluasi fungsi-fungsi sistem. Pengujian ini dilakukan untuk menemukan kesalahan fungsi yang tidak berjalan sebagaimana mestinya atau yang hilang, kesalahan pada antarmuka, serta ketidaksesuaian dalam struktur data.

2.9 Penelitian Terdahulu

Dalam melakukan penelitian ini selain dengan melakukan wawancara dan observasi awal, penulis juga melakukan *research* terhadap data-data serupa yaitu dengan melakukan pencarian terhadap jurnal-jurnal serupa yang kemudian dapat dijadikan acuan dari laporan penelitian ini. Setelah dilakukan pengumpulan jurnal-jurnal serupa, kemudian penulis menganalisis terhadap jurnal-jurnal dengan metode serupa. Berikut adalah Tabel 2.8 yang menjelaskan isi rangkuman terhadap penelitian terdahulu.

Tabel 2.8 Penelitian Terdahulu

No	Nama dan Tahun Publikasi	Hasil
1	Jadianan Parhusip, 2017	Pencucian mobil NASCAR belum memiliki sarana untuk menyampaikan informasi

mengenai layanan yang tersedia bagi masyarakat yang menggunakan jasa layanan pencucian mobil. Masalah lainnya adalah pengelolaan input data mobil yang akan diolah menjadi laporan bulanan, serta pengelolaan data bahan-bahan yang digunakan dalam proses pencucian mobil, yang juga akan disusun sebagai laporan pengeluaran bulanan.

Metode : UML(*Unified Modeling Language*).

Hasil : Dihasilkan *website* pencucian yang dapat menyediakan informasi yang berguna bagi masyarakat luas yang ingin mengetahui layanan pencucian yang disediakan oleh Nascar, termasuk informasi mengenai harga jasa, ketersediaan lorong, jumlah pengunjung yang sedang mengantri, serta pengumuman terbaru yang ada di tempat pencucian.

2	Meriska Defriani, 2017	Pitstop <i>Carwash</i> purwakarta masih menggunakan metode manual dalam pengelolaan datanya dimana datanya belum terkomputerisasi dan pencatatan dilakukan dalam sebuah buku. hal tersebut mengakibatkan sulitnya proses rekapitulasi data pelanggan juga administrasi per hari, serta proses pelaporan. Metode : <i>Waterfall</i> dan <i>blackbox testing</i>. Hasil : Dihasilkan sebuah sistem informasi yang dapat melakukan proses pengolahan data transaksi pencucian dimana data yang diolah meliputi data pelanggan dan data karyawan
3	Hadiana, 2020	Babe Car Wash 420 semuanya masih menggunakan sistem manual, terlihat dari pendaftaran pelanggan yang akan mencuci kendaraan, pelayanan pemilihan paket pencucian kendaraan pelanggan yang dilakukan dengan sesi tanya jawab dan transaksi pembayarannya masih dicatat pada

		nota pembayaran menggunakan tulisan tangan dan disimpan pada buku perusahaan. Metode : <i>Prototype</i> Hasil : Dihasilkan sistem yang dapat melakukan input data transaksi, mencetak nota pembayaran dan mampu memberikan sistem nomor antrian.
4	Cahyanti, 2021	Peneliti merencanakan perancangan aplikasi yang dimulai dari saat pelanggan datang ke lokasi pencucian kendaraan. Pada tahap ini, pelanggan akan ditawarkan untuk menjadi anggota dari layanan pencucian tersebut. Jika pelanggan bersedia menjadi anggota, maka data pribadi mereka akan dicatat dan disimpan oleh pihak pengelola usaha. Selanjutnya, proses pencucian kendaraan dilakukan, sementara pelanggan hanya perlu menunggu hingga pencucian selesai. Metode : <i>Waterfall</i> Hasil : sistem dibangun mampu menyelesaikan masalah pada pembuatan laporan, penggajian pegawai, dan data pelanggan yang terintegrasi pada sistem.
5	Shadiq, 2019	Sistem Informasi Pelayanan Pembayaran Cuci Steam Kendaraan Bermotor dimulai dengan membuat Activity Diagram Proses Bisnis Berjalan, Entity Relationship Diagram dan Use Case Diagram. Metode : <i>Waterfall</i> Hasil : Rancangan sistem mampu melakukan pengolahan data, penyimpanan data, dan pencarian data terkait pembayaran
6	Rhivanthio, 2018	Dalam proses pelayanan pencucian mobil, <i>Master Clean</i> menghadapi kendala utama berupa tingginya jumlah antrian kendaraan yang menunggu untuk dicuci. Kondisi ini menyebabkan waktu tunggu pelanggan menjadi cukup lama, yang pada akhirnya menurunkan

		tingkat kepuasan terhadap layanan yang diberikan. Akibatnya, sebagian pelanggan memilih beralih ke tempat pencucian lain yang dinilai lebih cepat dalam pelayanannya. Jika permasalahan ini tidak segera ditangani, maka secara bertahap jumlah pelanggan <i>Master Clean</i> berpotensi mengalami penurunan. Metode : <i>Waterfall</i>. Hasil : Rancangan mampu memberikan informasi data mengenai antrian kendaraan yang sedang dicuci. selain itu sistem juga dapat melakukan rekap data transaksi.
7	Maulidia, 2023	Perancangan Sistem Informasi Pelayanan dan Administrasi Klinik dilakukan karena banyaknya keluhan terhadap waktu tunggu yang lama. Terutama ada saat pasien tidak membawa kartu berobat sehingga pencarian rekam medis akan cukup memakan waktu dan sering terjadi double data dikarenakan kegiatan registrasi masih dilakukan secara manual kertas. Peneliti melakukan survey dengan 89 responden dengan hasil; 85,7% tidak puas, 15% puas. Metode : Systems Development Life Cycle(SDLC). Hasil : Sistem mampu menyimpan semua data rekam medis pasien. sehingga pasien tidak perlu lagi khawatir meskipun tidak membawa kartu berobat pencarian data rekam medis akan semakin efisien menggunakan sistem.
8	Riayansyah, 2021	Sistem Informasi Penggajian dibutuhkan di Lion Parcel Cisoka kota Bandung, dikarena pengelolaan data penggajian karyawan belum terkomputerisasi dan dinilai kurang efisien dalam pengelolaan data penggajian yang ada. Sehingga dibutuhkan suatu sistem yang dapat mencegah redudansi data dan dapat melakukan pengarsipan data penggajian karyawan

		Metode : <i>Waterfall</i> Hasil : <i>Website</i> yang dihasilkan dapat mengelola arsip data penggajian karyawan. Sehingga proses kontrol penggajian menjadi lebih akurat dan efisien karena sudah terkoneksi dengan <i>database</i>.
9	Haryanto, 2020	Kegiatan jual beli yang ada pada toko JAsko An Nazma Moslem Wear masih dilakukan secara manual, pemasaran dan promosi produk masih terbatas pada sekitar toko. pengenalan produk dan promosi masih dilakukan secara <i>offline</i>. Peneliti merasa diperlukannya sebuah sistem yang dapat memberikan layanan promosi dengan cakupan yang lebih luas dan dapat melakukan pemesanan melalui <i>online</i>. Metode : Systems Development Life Cycle (SDLC). Hasil : Sistem Informasi yang dirancang dapat melakukan promosi dan pemesan secara <i>online</i> melalui <i>website</i>.
10	Frieyadie, 2015	Pembangunan Sistem Informasi Inventory dibutuhkan untuk melakukan pengendalian dan perencanaan persediaan pada perusahaan. Sistem yang dibangun dapat digunakan administrasi dalam menginventarisasi aset, meliputi pencatatan, pengolahan, penyimpanan dan pelaporan. Metode : Systems Development Life Cycle(SDLC). Hasil : Aplikasi inventory berbasis web administrasi IT bisa melihat laporan barang masuk dan keluar dengan menampilkan tanggal dan identitas barang tersebut.

Berdasarkan jurnal penelitian terdahulu didapatkan beberapa pemahaman yang dapat dipelajari. Pada jurnal Jadianan Parhusip metode yang digunakan merupakan UML peneliti dapat lebih efisien dalam komunikasi dengan *stakeholder* dalam membangun sistem. Pada jurnal Meriska Defriani, Cahyanti, Shadiq, Rhivanthio, dan Riayansyah menggunakan metode *Waterfall*. Dalam

jurnal mereka menjelaskan dengan menggunakan metode *Waterfall* dapat memudahkan dalam membangun sistem informasi dikarenakan proses dalam metode *Waterfall* tidak terjadi tahapan yang berulang. Pada jurnal Hadiana menggunakan metode *prototype*. Metode *prototype* digunakan karena selain akan digunakan oleh *stakeholder* sistem yang dibuat oleh hadiana juga akan digunakan oleh pelanggan. Sehingga *prototype* digunakan agar dapat membuat sistem sesuai dengan kebutuhan masing-masing pengguna. Pada jurnal Haryanto dan Frieyadie menggunakan metode SDLC. Peneliti menggunakan metode SDLC untuk mengurangi faktor kesalahan dan memastikan bahwa seluruh pihak yang terlibat dalam sistem memahami sistem yang akan dibangun.

