

BAB II

TINJAUAN PUSTAKA

2.1 Teori Terkait

Penelitian terdahulu merupakan salah satu acuan penting yang digunakan peneliti dalam melaksanakan penelitian ini. Kajian terhadap penelitian terdahulu yang relevan bertujuan untuk memperkaya wawasan dan landasan teori, mengetahui posisi penelitian ini di antara penelitian yang sudah ada, serta mengidentifikasi persamaan dan perbedaan dengan penelitian sebelumnya sehingga penelitian ini dapat memberikan kontribusi yang lebih spesifik dan terarah.

Berikut adalah beberapa penelitian terdahulu yang relevan dengan penelitian ini:

Zulkarnain dan Zulkarnain (2023)[1] melakukan penelitian berjudul "Rancang Bangun Sistem Informasi *Onsite Service* Komputer Berbasis *Web* (Studi Kasus: *Cardi Computer & Services*)". Penelitian ini mengembangkan sistem informasi berbasis *web* untuk mengelola layanan *onsite* servis komputer secara digital. Metode pengembangan yang digunakan adalah *waterfall* dengan pengujian *black box testing* dan menghasilkan tingkat keberhasilan 100%. Persamaan dengan penelitian ini adalah pada objek penelitian berupa sistem informasi servis komputer berbasis *web*. Perbedaannya terletak pada studi kasus dan fitur tambahan booking online yang dikembangkan dalam penelitian ini.

Irawansyah dkk. (2025)[11] dalam penelitian berjudul "Perancangan Sistem Informasi Pelayanan Servis Elektronik Berbasis *Website* pada Sahabat Phone Metode *Prototype*" mengembangkan sistem informasi yang mencakup pemesanan layanan servis *online* dan pengecekan status servis secara *real-time*. Persamaan dengan penelitian ini adalah pada fitur pemesanan *online* dan pengecekan status servis. Perbedaannya terletak pada objek penelitian, di mana penelitian tersebut berfokus pada servis handphone, sedangkan penelitian ini berfokus pada servis komputer.

Sari dkk. (2022)[8] dalam penelitian "Implementasi *Internet of Things* Berbasis *Website* dalam Pemesanan Jasa Rumah *Service* Teknisi Komputer" mengembangkan sistem pemesanan jasa *service* teknisi komputer berbasis *web* dengan integrasi *IoT*. Persamaan dengan penelitian ini adalah pada fitur pemesanan layanan servis komputer berbasis *web*. Perbedaannya terletak pada penggunaan teknologi *IoT* pada penelitian tersebut, sementara penelitian ini tidak mengintegrasikan *IoT*.

Andika dan Sedyono (2023)[6] dalam penelitian "Perancangan Sistem Informasi Servis di Toko Oz Computer Salatiga Menggunakan Metode *Prototype* Berbasis *Web*" merancang sistem informasi servis komputer berbasis *web* dengan metode *prototype*. Persamaan dengan penelitian ini adalah pada objek penelitian berupa toko servis komputer berbasis *web*. Perbedaannya terletak pada metode pengembangan yang digunakan, di mana penelitian ini menggunakan metode *waterfall*.

Ardiansyah dkk. (2022)[12] dalam penelitian "Sistem Informasi Jasa Servis Printer Dengan Metode *Waterfall*" mengembangkan sistem informasi jasa servis printer berbasis *web* menggunakan metode *waterfall*. Persamaan dengan penelitian ini adalah pada penggunaan metode *waterfall* dan teknologi *PHP-MySQL*. Perbedaannya terletak pada objek penelitian, di mana penelitian tersebut berfokus pada servis printer saja, sementara penelitian ini mencakup servis komputer secara keseluruhan.

Berikut adalah ringkasan perbandingan penelitian terdahulu yang disajikan dalam bentuk tabel:

Tabel 2.1 Perbandingan Penelitian Terdahulu

No.	Nama Peneliti & Tahun	Judul Penelitian	Metode	Persamaan	Perbedaan
1	Zulkarnain & Zulkarnain (2023)	Rancang Bangun SI <i>Onsite Service</i> Komputer Berbasis <i>Web</i> (Cardi Computer)	<i>Waterfall, Black Box Testing</i>	Objek penelitian berupa sistem informasi servis komputer berbasis <i>web</i>	Objek studi kasus berbeda; penelitian ini di Jet Computer
2	Irawansyah dkk. (2025)	Perancangan SI Pelayanan Servis Elektronik Berbasis <i>Website</i> (Sahabat Phone)	<i>Prototype</i>	Sistem mencakup pemesanan <i>online</i> dan pengecekan status servis	Fokus pada servis <i>handphone</i> ; penelitian ini pada servis komputer
3	Sari dkk. (2022)	Implementasi <i>IoT</i> Berbasis <i>Website</i> dalam Pemesanan Jasa Rumah <i>Service</i> Teknisi Komputer	<i>Prototype + IoT</i>	Sistem <i>booking</i> servis komputer berbasis <i>web</i>	Menggunakan integrasi <i>IoT</i> ; penelitian ini tanpa <i>IoT</i>

No.	Nama Peneliti & Tahun	Judul Penelitian	Metode	Persamaan	Perbedaan
4	Andika & Sedyono (2023)	Perancangan SI Servis di Toko Oz Computer Salatiga Berbasis <i>Web</i>	<i>Prototype</i>	Objek penelitian toko servis komputer berbasis <i>web</i>	Metode pengembangan berbeda; penelitian ini menggunakan <i>Waterfall</i>
5	Ardiansyah dkk. (2022)	Sistem Informasi Jasa Servis Printer Dengan Metode <i>Waterfall</i>	<i>Waterfall</i>	Menggunakan metode <i>waterfall</i> , <i>PHP</i> , dan <i>MySQL</i>	Objek berupa servis printer; penelitian ini servis komputer secara keseluruhan

2.2 Landasan Teori

2.2.1 Pengertian Sistem

Sistem adalah sekumpulan elemen yang saling berhubungan dan bekerja sama untuk mencapai tujuan tertentu. Menurut Pressman dan Maxim (2019), sistem perangkat lunak merupakan kumpulan komponen yang saling terhubung untuk menghasilkan *output* yang berguna bagi pengguna, di mana setiap komponen harus dirancang dengan baik agar sistem secara keseluruhan dapat berfungsi optimal[13]. Sistem terdiri dari tiga komponen utama, yaitu *input* (masukan), proses, dan *output* (keluaran) yang saling berkaitan membentuk satu kesatuan yang utuh dan terstruktur. Rosa dan Shalahuddin (2018) juga menjelaskan bahwa sistem dapat didefinisikan sebagai kumpulan objek yang terintegrasi dengan hubungan antara tiap objek dan sifat-sifatnya yang diikat oleh sebuah tujuan yang ingin dicapai[14]. Dengan demikian, setiap sistem memiliki batas (*boundary*) yang memisahkannya dari

lingkungan luar, serta memiliki sub-sistem yang lebih kecil sebagai bagian dari keseluruhan sistem. Karakteristik sistem yang baik mencakup:

1. Memiliki komponen-komponen yang saling berinteraksi
2. Memiliki batas sistem yang jelas
3. Memiliki lingkungan luar sistem
4. Memiliki penghubung antar sub-sistem
5. Memiliki tujuan yang jelas
6. Mampu melakukan proses pengendalian (kontrol) untuk menjaga agar sistem tetap berjalan sesuai tujuannya.

2.2.2 Pengertian Informasi

Informasi adalah data yang telah diolah dan diproses sehingga memiliki makna dan nilai guna bagi penerimanya dalam pengambilan keputusan. Rosa dan Shalahuddin (2018) mendefinisikan informasi sebagai data yang telah dikelola dan diproses untuk memberikan arti serta meningkatkan pengetahuan bagi yang menggunakannya[14]. Kualitas informasi ditentukan oleh tiga aspek utama:

1. Akurasi, bebas dari kesalahan
2. Ketepatan waktu, tersedia saat dibutuhkan
3. Relevansi, sesuai dengan kebutuhan penggunaanya.

Menurut O'Brien dan Marakas, informasi merupakan data yang telah dikonversi ke dalam konteks yang bermakna dan berguna

bagi pengguna akhir tertentu. Informasi yang berkualitas harus memenuhi beberapa kriteria tambahan, antara lain:

1. Lengkap, artinya informasi yang diberikan harus mencakup semua data yang dibutuhkan.
2. Mudah dipahami, artinya penyajian informasi harus jelas dan tidak membingungkan pengguna.
3. Konsisten, artinya informasi yang sama tidak boleh memberikan makna yang berbeda pada kondisi yang sama.

Dalam konteks sistem informasi servis komputer di Jet Computer, informasi yang dihasilkan mencakup data *booking* pelanggan, status pengerjaan servis, riwayat perbaikan, dan laporan rekapitulasi yang semuanya harus memenuhi kriteria kualitas informasi tersebut agar berguna bagi pengelola maupun pelanggan.

2.2.3 Pengertian Sistem Informasi

Sistem informasi adalah suatu sistem yang mengumpulkan, memproses, menyimpan, dan mendistribusikan informasi guna mendukung pengambilan keputusan dalam suatu organisasi. Zulkarnain dan Zulkarnain (2023) menyatakan bahwa sistem informasi berbasis *web* mampu mengintegrasikan berbagai proses bisnis menjadi satu platform yang efisien, akurat, dan mudah diakses[1]. Rahman dan La Saudin (2022) menyatakan bahwa sistem informasi manajemen berfungsi menyediakan informasi yang diperlukan bagi pengambilan keputusan manajemen pada berbagai

tingkatan organisasi. Sistem informasi terdiri dari beberapa komponen yang bekerja secara sinergis, yaitu:

1. Perangkat keras (*hardware*) sebagai infrastruktur fisik.
2. Perangkat lunak (*software*) sebagai instruksi pengolahan data.
3. Basis data (*database*) sebagai tempat penyimpanan data terstruktur.
4. Jaringan (*network*) sebagai media komunikasi antar komponen.
5. Sumber daya manusia (*brainware*) sebagai pengguna dan pengelola sistem.
6. Prosedur sebagai aturan dan panduan penggunaan sistem.

Dalam penelitian ini, sistem informasi servis komputer di Jet Computer dirancang untuk mengotomasi seluruh proses bisnis mulai dari penerimaan *booking*, pengelolaan data servis, pemantauan status pengerjaan, hingga pelaporan kepada manajemen, sehingga meningkatkan efisiensi operasional dan kualitas pelayanan kepada pelanggan secara signifikan.

2.2.4 Aplikasi Berbasis *Web*

Aplikasi berbasis *web* adalah aplikasi yang diakses melalui *web browser* tanpa memerlukan instalasi khusus pada perangkat pengguna. Dawis dkk. (2023) mendefinisikan aplikasi berbasis *web* sebagai perangkat lunak yang berjalan di *server* dan diakses melalui *browser* dari berbagai perangkat yang terkoneksi internet[15]. Komponen utama pengembangan aplikasi berbasis *web* meliputi:

HTML/CSS (struktur dan tampilan), *PHP* (*server-side scripting*), *JavaScript* (*client-side scripting*), dan *MySQL* (manajemen basis data). Aplikasi berbasis *web* memiliki beberapa keunggulan dibandingkan aplikasi *desktop* tradisional, antara lain:

1. Aksesibilitas tinggi, dapat diakses dari mana saja dan kapan saja menggunakan perangkat yang terhubung internet.
2. Kemudahan pemeliharaan, pembaruan sistem dilakukan di sisi *server* tanpa perlu update manual di setiap perangkat klien.
3. Kompatibilitas lintas *platform*, dapat dijalankan di berbagai sistem operasi selama memiliki *web browser*.
4. Skalabilitas, kapasitas sistem dapat ditingkatkan sesuai kebutuhan tanpa mengganggu pengalaman pengguna.

Arsitektur aplikasi berbasis *web* umumnya mengikuti pola *client-server*, di mana *client* (*browser* pengguna) mengirimkan permintaan *HTTP* ke *server*, *server* memproses permintaan dan mengakses basis data, kemudian mengembalikan respons dalam bentuk halaman *HTML* yang ditampilkan pada *browser*. Dalam penelitian ini, arsitektur tersebut diimplementasikan menggunakan *framework laravel 10* yang mengadopsi pola *MVC* (*Model-View-Controller*) untuk memisahkan logika bisnis, tampilan antarmuka, dan pengelolaan data secara terstruktur.

2.2.5 *PHP (Hypertext Preprocessor)*

PHP (Hypertext Preprocessor) merupakan bahasa pemrograman sisi *server* yang paling banyak digunakan dalam pengembangan aplikasi web dinamis. *PHP* bersifat *open source* dan dapat diintegrasikan dengan berbagai sistem basis data termasuk *MySQL*. Ardiansyah dkk. (2022) menggunakan *PHP* dan *MySQL* sebagai teknologi utama dalam pengembangan sistem informasi jasa servis berbasis *web* dengan hasil yang fungsional dan efisien[16]. Kelebihan *PHP* meliputi performa tinggi, dokumentasi lengkap, dukungan komunitas besar, dan kompatibilitas lintas platform. *PHP* pertama kali dikembangkan oleh Rasmus Lerdorf pada tahun 1994 dan sejak saat itu terus berkembang menjadi salah satu bahasa pemrograman *web* yang paling dominan di dunia. Beberapa karakteristik utama *PHP* yang menjadikannya pilihan tepat dalam pengembangan sistem informasi berbasis *web* adalah:

1. Sintaks yang sederhana dan mudah dipelajari, terutama bagi pengembang yang sudah familiar dengan bahasa *C* atau *Java*.
2. Kemampuan *embedding* langsung ke dalam kode *HTML* sehingga memudahkan pengembangan halaman web dinamis.
3. Dukungan terhadap berbagai protokol jaringan dan format data.
4. Kemampuan penanganan file, *session*, dan *cookie* secara *built-in*.

5. Tersedianya ribuan *library* dan *framework* seperti *CodeIgniter*, *Laravel*, dan *Symfony* yang mempercepat proses pengembangan.

Dalam penelitian ini, *PHP* digunakan bersama *framework Laravel* 10 untuk membangun logika bisnis sistem informasi servis komputer Jet Computer secara terstruktur dan efisien.

2.2.6 *MySQL (Database Management System)*

MySQL adalah sistem manajemen basis data relasional (*RDBMS*) *open source* yang paling banyak digunakan dalam pengembangan aplikasi *web*. *MySQL* menggunakan *SQL* untuk mengakses dan mengelola data. Zein dkk. (2023) menjelaskan bahwa *MySQL* merupakan *RDBMS* yang handal untuk menyimpan data terstruktur dengan kemampuan menangani volume data besar, mendukung transaksi *ACID*, dan menyediakan mekanisme keamanan yang baik[10]. Dalam penelitian ini, *MySQL* menyimpan seluruh data sistem servis meliputi data pelanggan, servis, teknisi, dan *spare part*. *MySQL* dikembangkan oleh *MySQL AB* dan saat ini dikelola oleh *Oracle Corporation*. Sebagai *RDBMS*, *MySQL* mengorganisasi data dalam bentuk tabel-tabel yang saling berelasi menggunakan kunci primer (*primary key*) dan kunci asing (*foreign key*). Beberapa keunggulan *MySQL* yang menjadikannya pilihan utama dalam pengembangan sistem informasi berbasis *web* antara lain:

1. Performa tinggi dengan kemampuan pemrosesan *query* yang cepat bahkan untuk dataset besar.
2. Dukungan *multi-user* dan *multi-threading* sehingga mampu melayani banyak koneksi secara bersamaan.
3. Fitur replikasi data untuk kebutuhan *backup* dan *high availability*.
4. dukungan penuh terhadap standar *SQL ANSI/ISO*.
5. integrasi yang sangat baik dengan bahasa pemrograman *PHP* melalui ekstensi *MySQLi* dan *PDO*.

Dalam penelitian ini, *MySQL* digunakan untuk menyimpan dan mengelola seluruh data sistem meliputi tabel pelanggan, tabel servis, tabel teknisi, tabel *spare part*, dan tabel laporan yang saling berelasi membentuk struktur basis data yang terintegrasi.

2.2.7 Framework Laravel 10

Laravel adalah *framework PHP* bersifat *open source* yang menggunakan arsitektur *MVC (Model-View-Controller)* dan dirancang untuk memudahkan pengembangan aplikasi *web* yang ekspresif, elegan, dan terstruktur. *Framework* ini dikembangkan pertama kali oleh Taylor Otwell dan dirilis pada tahun 2011, kemudian terus berkembang hingga versi *Laravel 10* yang digunakan dalam penelitian ini. Penggunaan *Laravel* dalam pengembangan sistem informasi berbasis *web* terbukti meningkatkan kecepatan pengembangan, kerapian struktur kode,

kemudahan pemeliharaan, serta keamanan aplikasi melalui fitur proteksi bawaan terhadap *SQL injection*, *Cross-Site Scripting (XSS)*, dan *Cross-Site Request Forgery (CSRF)*. *Laravel* dipilih dalam penelitian ini karena menyediakan ekosistem yang lengkap dan modern sehingga mempercepat pembangunan sistem informasi servis komputer Jet Computer secara efisien.

Arsitektur *MVC* yang diterapkan *Laravel* membagi aplikasi menjadi tiga lapisan utama, yaitu:

1. *Model*, yang bertanggung jawab mengelola logika data dan interaksi dengan basis data *MySQL* melalui *Eloquent ORM (Object Relational Mapping)*.
2. *View*, yang bertanggung jawab menampilkan antarmuka pengguna berupa halaman *HTML* dinamis menggunakan *template engine Blade*.
3. *Controller*, yang berfungsi sebagai penghubung antara *Model* dan *View* dengan memproses permintaan pengguna (*request*) dan menentukan respons (*response*) yang sesuai.

Pemisahan tanggung jawab ini membuat kode lebih terorganisir, mudah diuji, dan mudah dipelihara. Selain itu, *Laravel* menyediakan berbagai fitur bawaan yang siap pakai, seperti *Eloquent ORM* untuk operasi *CRUD* yang aman, *Blade* untuk penyusunan tampilan, *Form Request Validation* untuk validasi input pengguna, serta *Middleware* untuk manajemen autentikasi dan

keamanan akses. Fitur-fitur tersebut secara signifikan mempercepat proses pengembangan sistem informasi servis komputer Jet Computer dalam penelitian ini.

2.2.8 Metode Pengembangan Sistem *Waterfall*

Metode *waterfall* merupakan model pengembangan perangkat lunak yang bersifat *linier* dan *sekuensial*, di mana setiap tahapan diselesaikan sebelum beralih ke tahapan berikutnya. Pressman dan Maxim (2019) mendeskripsikan *waterfall* sebagai pendekatan yang mengalir secara sistematis dari analisis kebutuhan, perancangan, implementasi, pengujian, hingga pemeliharaan[10]. Siburian dan Latifah (2023) membuktikan bahwa *waterfall* menghasilkan sistem yang terstruktur dan sesuai kebutuhan apabila diterapkan pada proyek dengan kebutuhan yang sudah terdefinisi jelas[2]. Metode *waterfall* pertama kali diperkenalkan oleh Winston W. Royce pada tahun 1970 dan hingga kini masih banyak diterapkan dalam pengembangan sistem informasi skala kecil hingga menengah. Keunggulan utama metode ini adalah kemudahan pengelolaan karena setiap fase memiliki *deliverable* yang jelas dan jadwal yang terdefinisi. Widiyanti dan Darussalam (2023) menegaskan bahwa penerapan *waterfall* secara berurutan dan disiplin terbukti menghasilkan sistem yang menggantikan proses manual secara menyeluruh sesuai kebutuhan pengguna[17]. Metode *waterfall* dipilih dalam penelitian ini karena kebutuhan sistem Jet

Computer sudah terdefinisi dengan jelas sejak awal berdasarkan hasil observasi dan wawancara, sehingga perubahan kebutuhan di tengah proses pengembangan dapat diminimalisir. Tahapan metode *waterfall* dalam penelitian ini:

1. Perencanaan (*Planning*): Identifikasi masalah, ruang lingkup, tujuan, dan penyusunan jadwal.
2. Analisis Kebutuhan: Identifikasi dan dokumentasi kebutuhan fungsional dan non-fungsional sistem.
3. Perancangan (*Design*): Rancangan arsitektur, diagram *UML*, *ERD*, dan *wireframe* antarmuka.
4. Implementasi: Pengkodean menggunakan *PHP*, *laravel* 10, *Tailwind CSS*, dan *MySQL*.
5. Pengujian: Pengujian *black box testing* dengan teknik *equivalence partitioning*.
6. Pemeliharaan: Perbaikan dan pembaruan sistem berdasarkan masukan pengguna.

2.2.9 UML (Unified Modeling Language)

UML (Unified Modeling Language) merupakan bahasa pemodelan standar yang digunakan untuk merancang dan mendokumentasikan sistem perangkat lunak secara visual. Rosa dan Shalahuddin[18] mendefinisikan *UML* sebagai alat bantu yang handal dalam pengembangan sistem berorientasi objek karena menyediakan bahasa pemodelan visual yang memungkinkan

pengembang membuat cetak biru sistem dalam bentuk yang baku dan mudah dikomunikasikan. *UML* dikembangkan oleh Grady Booch, Ivar Jacobson, dan James Rumbaugh pada pertengahan 1990-an dan selanjutnya ditetapkan sebagai standar oleh *Object Management Group (OMG)*[19]. Diagram *UML* yang digunakan dalam penelitian ini meliputi:

1. *Use Case Diagram*

Use Case Diagram merupakan salah satu jenis diagram dalam *UML* yang berfungsi untuk menggambarkan interaksi antara aktor dengan sistem. Menurut Rosa dan Shalahuddin[20], *Use Case Diagram* digunakan untuk mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat, sehingga dapat diketahui fungsi-fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Aktor dalam *Use Case Diagram* dapat berupa manusia, sistem lain, maupun perangkat yang berinteraksi dengan sistem yang sedang dikembangkan.

Komponen utama dalam *Use Case Diagram* meliputi aktor (*actor*), *use case* yang digambarkan dengan simbol *elips*, serta relasi yang menghubungkan keduanya. Relasi yang umum digunakan antara lain *include*, *extend*, dan *generalization*. Relasi *include* menunjukkan bahwa suatu *use case* selalu memanggil *use case* lainnya, sedangkan relasi *extend* menunjukkan bahwa

suatu *use case* dapat diperluas oleh *use case* lain dalam kondisi tertentu[21].

2. *Activity Diagram*

Activity Diagram merupakan diagram UML yang digunakan untuk menggambarkan alur kerja (*workflow*) atau proses bisnis dalam suatu sistem secara mendetail. Rosa dan Shalahuddin[22] menjelaskan bahwa *Activity Diagram* menggambarkan aktivitas-aktivitas yang terjadi dalam sistem, termasuk aktivitas yang dilakukan oleh aktor maupun sistem itu sendiri secara berurutan. Diagram ini sangat berguna untuk memodelkan logika prosedural, proses bisnis, dan alur kerja yang bersifat paralel maupun sekuensial.

Komponen utama dalam *Activity Diagram* meliputi initial *node* sebagai titik awal proses, *action* atau aktivitas yang digambarkan dengan simbol persegi panjang bersudut bulat, *decision node* untuk menggambarkan percabangan kondisi, *fork* dan join untuk alur paralel, serta *activity final node* sebagai titik akhir proses. Dengan menggunakan *Activity Diagram*, alur proses dalam sistem dapat dipahami secara menyeluruh oleh seluruh pemangku kepentingan (*stakeholder*) yang terlibat dalam pengembangan sistem.

3. *Sequence Diagram*

Sequence Diagram merupakan diagram UML yang menggambarkan urutan interaksi antar objek dalam sistem untuk satu skenario tertentu berdasarkan urutan waktu. Menurut Rosa dan Shalahuddin[22], *Sequence Diagram* digunakan untuk menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Setiap skenario yang digambarkan dalam *Sequence Diagram* merepresentasikan satu alur eksekusi tertentu, misalnya proses login, proses penginputan data, atau proses pencetakan laporan.

Komponen utama dalam *Sequence Diagram* terdiri atas *lifeline* yang merepresentasikan objek atau komponen sistem, *activation bar* yang menunjukkan periode aktif suatu objek, pesan (*message*) yang dikirimkan antar objek, serta *return message* yang menggambarkan nilai kembalian dari suatu pesan[23]. Diagram ini sangat bermanfaat dalam memahami bagaimana objek-objek dalam sistem saling berkolaborasi untuk menyelesaikan suatu proses tertentu.

4. *Class Diagram*

Class Diagram merupakan diagram UML yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Rosa dan

Shalahuddin[23] menjelaskan bahwa *Class Diagram* mendeskripsikan jenis-jenis objek dalam sistem beserta berbagai macam hubungan statis yang terdapat di antara objek-objek tersebut, termasuk atribut dan operasi dari suatu kelas serta batasan yang terdapat pada hubungan antar objek. *Class Diagram* menjadi acuan utama bagi pengembang dalam proses implementasi kode program.

Komponen utama dalam *Class Diagram* terdiri atas kelas (*class*) yang memiliki tiga bagian yaitu nama kelas, atribut, dan metode atau operasi. Hubungan antar kelas dapat berupa *association*, *aggregation*, *composition*, *dependency*, maupun *inheritance* atau generalisasi[24]. Selain itu, setiap atribut dan metode dalam kelas memiliki tingkat akses (*visibility*) yang ditandai dengan simbol + untuk *public*, - untuk *private*, dan # untuk *protected*.

5. *Entity Relationship Diagram* (ERD) untuk rancangan basis data

Entity Relationship Diagram (ERD) merupakan diagram yang secara khusus digunakan untuk merancang struktur basis data (*database*) suatu sistem. Menurut Rosa dan Shalahuddin[23], *ERD* digunakan untuk pemodelan basis data relasional yang menggambarkan entitas, atribut yang dimiliki setiap entitas, serta relasi antar entitas. ERD berfungsi sebagai cetak biru dalam proses pembuatan tabel-tabel pada basis data

sehingga struktur penyimpanan data dapat terorganisir dengan baik dan terhindar dari redundansi data.

Komponen utama dalam ERD terdiri atas entitas (*entity*) yang merepresentasikan objek atau tabel dalam basis data, atribut yang menggambarkan karakteristik dari setiap entitas, serta relasi yang menghubungkan satu entitas dengan entitas lainnya[25]. Relasi antar entitas memiliki tingkat kardinalitas yang menunjukkan jumlah hubungan antar entitas, yaitu satu ke satu (1:1), satu ke banyak (1:N), dan banyak ke banyak (M:N). Penggunaan ERD dalam perancangan sistem bertujuan agar basis data yang dibangun memiliki struktur yang efisien, konsisten, dan mudah untuk dikembangkan lebih lanjut.

UML dikembangkan oleh Grady Booch, Ivar Jacobson, dan James Rumbaugh pada pertengahan 1990-an dan ditetapkan sebagai standar oleh *Object Management Group (OMG)*. Setiap jenis diagram UML memiliki fungsi yang spesifik dalam menggambarkan aspek sistem yang berbeda. *Use Case Diagram* digunakan untuk menggambarkan interaksi antara aktor (pengguna) dengan sistem, menjelaskan apa saja yang dapat dilakukan oleh setiap jenis pengguna. *Activity Diagram* menggambarkan alur kerja atau proses bisnis dalam sistem secara mendetail. *Sequence Diagram* menggambarkan urutan interaksi antar objek dalam sistem untuk satu skenario tertentu. *Class Diagram* menggambarkan struktur

kelas-kelas dalam sistem beserta atribut dan metodenya. Sementara ERD (*Entity Relationship Diagram*) secara khusus digunakan untuk merancang struktur basis data, menggambarkan entitas, atribut, dan relasi antar tabel. Penggunaan UML secara konsisten dalam penelitian ini bertujuan agar sistem yang dibangun terdokumentasi dengan baik dan mudah dipelihara atau dikembangkan lebih lanjut di masa mendatang.

2.2.10 Black Box Testing

Black box testing adalah metode pengujian yang berfokus pada fungsionalitas sistem dari sudut pandang pengguna, tanpa memperhatikan struktur internal kode program. Ridwan dan Nuryasin (2024) menyatakan bahwa *black box testing* dengan teknik *equivalence partitioning* efektif untuk menguji fungsionalitas sistem *web* secara menyeluruh dan mengidentifikasi potensi kesalahan[26]. Hudi dan Karyanti (2024) juga menegaskan bahwa *black box testing* pada sistem *web* di bidang jasa memastikan seluruh fungsi berjalan sesuai kebutuhan pengguna[10]. *Equivalence partitioning* membagi *input* ke dalam kelas-kelas ekuivalen dan menguji setiap kelas untuk memverifikasi *respons* sistem yang benar. *Black box testing* disebut juga pengujian fungsional atau behavioral testing karena penguji hanya perlu memahami apa yang seharusnya dilakukan oleh sistem, bukan bagaimana sistem melakukannya secara internal. Metode ini sangat sesuai diterapkan pada tahap akhir pengembangan untuk

memverifikasi kesesuaian sistem dengan spesifikasi kebutuhan yang telah ditetapkan. Teknik *equivalence partitioning* yang digunakan dalam penelitian ini membagi data masukan ke dalam kelas-kelas ekuivalen, yakni:

1. Kelas valid, yaitu data masukan yang sesuai dengan batasan yang ditetapkan dan diharapkan menghasilkan *respons* sistem yang benar.
2. Kelas tidak valid, yaitu data masukan yang di luar batasan yang ditetapkan dan diharapkan menghasilkan pesan kesalahan yang tepat dari sistem.

Dengan membagi *input* ke dalam kelas-kelas tersebut, pengujian dapat dilakukan secara efisien karena cukup menguji satu representasi dari setiap kelas tanpa harus menguji semua kemungkinan kombinasi input. Dalam penelitian ini, *black box testing* diterapkan pada seluruh modul sistem meliputi modul *login*, *booking* servis, pemantauan status, manajemen data, dan pelaporan untuk memastikan semua fungsionalitas berjalan sesuai kebutuhan operasional Jet Computer.

2.2.11 Sistem *Booking Online*

Sistem *booking online* memungkinkan pelanggan melakukan pemesanan layanan secara digital melalui internet tanpa harus hadir secara fisik. Putra dkk. (2023) membuktikan bahwa sistem *booking* antrian berbasis *website* mengintegrasikan proses pemesanan,

mengurangi antrian, dan meningkatkan kepuasan pelanggan secara signifikan[27]. Widianti dan Darussalam (2023) menegaskan bahwa digitalisasi proses melalui sistem berbasis *web* meningkatkan aksesibilitas layanan bagi masyarakat secara menyeluruh[28]. Dalam penelitian ini, fitur *booking online* memungkinkan pelanggan Jet Computer mendaftarkan keluhan dan menjadwalkan servis dari mana saja melalui *website*.

2.2.12 Payment Gateway (Midtrans)

Payment gateway adalah layanan pihak ketiga yang menjembatani transaksi pembayaran *online* antara penjual, pembeli, dan *bank*/penyedia metode pembayaran secara aman dan *real-time*[29]. *Midtrans* merupakan salah satu penyedia *payment gateway* di Indonesia yang menyediakan produk *Snap* (tampilan pembayaran siap pakai berupa *popup*) dan *Core API* (API yang memberi kontrol penuh untuk membangun alur pembayaran sendiri, termasuk pembuatan *QRIS* dinamis)[30]. *Midtrans* menyediakan mode *Sandbox* untuk pengujian dan *Production* untuk transaksi nyata, serta mekanisme *webhook* yang mengirim notifikasi status transaksi secara otomatis ke *server merchant*.

Pada penelitian ini, *Midtrans* diimplementasikan untuk dua skenario:

1. Pembayaran *online* melalui *Snap*, di mana sistem membuat *Snap* Token berdasarkan data *invoice* lalu menampilkannya sebagai popup pembayaran.
2. Pembayaran *QRIS* melalui *Core API*, di mana sistem membuat kode QR yang dapat dipindai pelanggan menggunakan *e-wallet* atau *mobile banking*.

Setiap perubahan status pembayaran diverifikasi melalui *signature key* dan diperbarui secara otomatis pada tabel *invoice* tanpa konfirmasi manual *admin*.