BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Tabel 2. 1 Penelitian Terdahulu

Jurnal 1 [16]	
Penulis Jurnal	Sofiansyah Fadli
Judul Artikel	Model *Rapid Application Development* Dalam Pengembangan Sistem Reservasi dan Penyewaan Kamar Hotel
Nama Jurnal	Jurnal Informatika & Rekayasa Elektronika, Volume 1, No 1, April 2018
Hasil Jurnal	Sistem ini efektif untuk mengelola data reservasi dan penyewaan kamar hotel, memudahkan petugas untuk mengelola data sesuai dengan tanggung jawab mereka, menyederhanakan manajemen transaksi, mengurangi kesalahan, dan memfasilitasi pemeliharaan data terkait reservasi dan penyewaan kamar hotel.
Adopsi Jurnal	Referensi dalam pengembangan sistem menggunakan metode RAD.
Permasalahan	Hotel Queen, yang menjadi fokus penelitian ini, merupakan salah satu dari banyak hotel yang telah beroperasi namun belum menerapkan teknologi komputer, khususnya dalam hal reservasi dan penyewaan kamar, untuk mengelola data dan memberikan informasi kepada tamu atau pengunjung hotel. Saat ini, proses transaksi reservasi dan penyewaan kamar di Hotel Queen masih dilakukan secara manual dengan mencatat semua informasi pada buku, yang berpotensi menimbulkan kesalahan dan kehilangan data, serta mengakibatkan laporan yang dihasilkan kurang akurat.
Pembahasan	i. Analisis Sistem Untuk mengatasi masalah tersebut, sistem reservasi dan penyewaan kamar hotel telah dirancang dan dikembangkan di Hotel Queen. Tujuannya adalah untuk menyederhanakan dan mempercepat proses kerja, termasuk penginputan data, pencarian transaksi, dan pembuatan laporan. ii. Tujuan Pembuatan Sistem Membangun sistem baru untuk menangani

	masalah yang ada atau memperbarui sistem yang sudah ada yang belum menggunakan komputerisasi. Fokus utama pengembangan sistem adalah pada reservasi dan penyewaan kamar hotel. iii. Analisis Kebutuhan Sistem Dalam bagian ini terdapat beberapa pembahasan dengan menggunakan diagram, yaitu *Data Flow Diagram* (DFD); diagram konteks; *Entity Relationship Diagram* (ERD); dan arsitektur program.
Metode	Pengumpulan data: a. Observasi b. Studi Pustaka c. Wawancara d. Dokumentasi Pengembangan Sistem: a. *Rapid Application Development*
Jurnal 2 [17]	
Penulis Jurnal	Tania Jovita Wibowo, Suryasari, Arnold Aribowo, Andree E. Widjaja
Judul Artikel	Sistem Informasi Penunjang Proses Pemesanan dan Desain Kue Pada Toko Kue Artisan *Online* Berbasis *Web*
Nama Jurnal	ULTIMA InfoSys, Volume X, No. 1, Juni 2019
Hasil Jurnal	Hasil akhir dari penelitian adalah sebuah website yang dapat mendukung proses pemesanan dan desain kue pada Toko Kue Artisan *Online*.
Adopsi Jurnal	Referensi dalam pengembangan sistem menggunakan metode RAD dan pemodelan sistem.
Permasalahan	Pemesanan kue artisan di toko umumnya masih dilakukan secara manual melalui dua tahap utama. Pertama, pelanggan memilih jenis dan desain kue melalui platform media sosial seperti Instagram atau Facebook. Kemudian, pemesanan kue dilakukan melalui aplikasi perpesanan seperti Line atau WhatsApp. Ketergantungan pada kedua media ini mengakibatkan kurang efisiennya sistem pemesanan. Permintaan khusus untuk modifikasi kue hanya dapat dilakukan melalui pesan teks, yang sering mengakibatkan kesalahpahaman dan ketidakjelasan antara pelanggan dan pembuat kue karena kurangnya visualisasi yang jelas. Kendala yang muncul dari pesanan khusus atau custom mencakup

	perubahan desain dan ketidakmampuan kedua belah pihak untuk melihat gambaran visual dari permintaan atau ide yang diajukan, sehingga menimbulkan perbedaan persepsi.
Pembahasan	A. Tahap Analisis Dalam bagian ini terdapat beberapa pembahasan dengan menggunakan diagram, yaitu *use case diagram* dan *class diagram*. B. Tahap Perancangan - Perancangan manajemen data menggunakan *Entity Relationship Diagram* (ERD). - Perancangan lapisan antarmuka dengan memberikan gambaran sederhana berupa sketsa. - Pengujian dan implementasi dengan *setup* sketsa yang telah dibuat.
Metode	Pengumpulan data: a. Wawancara b. Survei Pelanggan c. Observasi d. Studi Literatur Pengembangan sistem: a. *Rapid Application Development*
Jurnal 3 [18]	
Penulis Jurnal	Shruti Dhanotia, Ruchi Goyal
Judul Artikel	*Rapid Application Development (Rad) Approach with Halt Points*
Nama Jurnal	International Journal of Engineering Sciences & Research Technology
Hasil Jurnal	Hasil dari penelitian adalah versi baru dari RAD dengan ditambahkannya Halt Point S (Hps) untuk menyediakan efisiensi yang lebih baik bagi produk perangkat lunak.
Adopsi Jurnal	Referensi dalam pengembangan sistem menggunakan metode RAD.
Permasalahan	Permasalahan terdapat pada metode RAD: - ketika proyek dalam skala besar membutuhkan *engineer* dengan keahlian tinggi. - antara *customer* dengan pengembang harus memiliki komitmen terhadap waktu. - jika komitmen tidak berjalan, maka proyek dengan RAD akan mudah menjadi gagal.
Pembahasan	Pembahasan terdiri dari *system architecture* (*business modeling, data modeling, process modeling, application generation, testing and turn over*), masalah dengan RAD, dan

	iterative application development methodology.
Metode	*Rapid Application Development*
Jurnal 4 [19]	
Penulis Jurnal	Neelu Lalband, D. Kavith
Judul Artikel	*Software Engineering for Smart Healthcare Applications*
Nama Jurnal	International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume 8, Issue 6S4, April 2019
Hasil Jurnal	Hasil dari penelitian adalah analisis dari model SDLC yang sudah ada dan menawarkan model SDLC terbaik untuk aplikasi *Smart Healthcare* yang berfokus pada peningkatan kualitas. Penelitian ini juga menghasilkan identifikasi dari rintangan yang dihadapi pada *software engineering* untuk *smart applications*.
Adopsi Jurnal	Referensi dalam penggunaan SDLC.
Permasalahan	Kurangnya fokus pada proses pengembangan *software* berakibat pada meningkatnya kelemahan dalam implementasi yang menyebabkan berkurangnya kualitas, biaya, dan rasa kepercayaan.
Pembahasan	Kurangnya fokus pada proses pengembangan *software* berakibat pada meningkatnya kelemahan dalam implementasi yang menyebabkan berkurangnya kualitas, biaya, dan rasa kepercayaan.
Metode	Pengumpulan data: a. *Research Question* b. Studi Pustaka
Jurnal 5 [20]	
Penulis Jurnal	Ririn Ikana Desanti, Carolyn Feiby Supit, Andree E. Widjaja
Judul Artikel	Aplikasi Perekrutan dan Penilaian Karyawan Berbasis *Web* Pada PT. XYZ
Nama Jurnal	ULTIMA InfoSys, Vol. VIII, No. 2
Hasil Jurnal	Hasil dari penelitian ini adalah sebuah aplikasi berbasis *web* yang mampu mendukung, mengotomatisasi, dan mempercepat proses rekrutmen dan penilaian kinerja pada PT. XYZ
Adopsi Jurnal	Referensi dalam pengembangan RAD.

Permasalahan	Pada PT. XYZ, rekrutmen karyawan dan penilaian kinerja masih menggunakan cara manual. Cara manual rawan mengalami kesalahan dan pengambilan keputusan yang tidak adil.
Pembahasan	a. Tahap Analisis Menggunakan *use case diagram* untuk aplikasi perekrutan karyawan dan penilaian kinerja. b. Tahap Perancangan Pendekatan AHP digunakan dengan melakukan perbandingan dua-perspektif antara faktor-faktor yang dievaluasi untuk menentukan bobot dari setiap faktor dan antara opsi satu dengan opsi lainnya untuk mengevaluasi faktor-faktor tersebut. c. Perancangan Lapisan Antarmuka Aplikasi terbagi menjadi dua bagian, yakni front-end dan back-end. Bagian front-end digunakan oleh para pelamar untuk mengajukan lamaran pada lowongan pekerjaan yang ditawarkan oleh PT.XYZ. Di sini, mereka juga dapat melakukan tes secara online. Selain itu, bagian front-end juga digunakan oleh kepala departemen untuk menilai kinerja karyawan. Sedangkan bagian back-end digunakan untuk mengajukan permintaan karyawan baru oleh kepala departemen, mengelola lowongan pekerjaan oleh kepala HRD, mengelola soal tes oleh kepala HRD dan staf HRD, serta mengelola indikator penilaian oleh staf HRD. d. Pengujian dan Implementasi Tahap pengujian dalam penelitian ini dilakukan dengan *black-box* testing dengan pendekatan *scenario* dan *functional testing*.
Metode	*Rapid Application Development* dengan *prototyping method, Analytic Hierarchy Process* (AHP)

Tabel 2.1 menunjukkan penelitian terdahulu yang digunakan sebagai referensi untuk penelitian rancang bangun sistem informasi berikut. Untuk mengembangkan sebuah sistem informasi, diperlukan pemahaman dasar mengenai *System Development Life Cycle* (SDLC) agar pengembangan yang dilakukan menjadi terstruktur dan mudah dikendalikan [19]. Pada tahapan *design* (perancangan) yang

termasuk dalam SDLC, terdapat tahap pemodelan sistem yang dibuat dalam bentuk diagram-diagram atau jenis visualisasi lainnya [17]. SDLC merupakakn gambaran umum dari siklus pengembangan sistem dan tahapan yang ada di dalamnya, namun banyak metode-metode yang dikembangkan dari SDLC, salah satunya adalah metode *Rapid Application Development* (RAD) [16] [18] [20]. Semua referensi yang menggunakan metode RAD memiliki waktu pengembangan yang singkat dan menghasilkan sistem informasi atau aplikasi yang dapat berjalan dengan baik [16] [18] [20].

2.2. *Rapid Application Development* (RAD)

Pada SDLC terdapat beberapa model yang sudah diciptakan selama ini, yaitu *waterfall*, *spiral*, *prototype*, dan *rapid application development* (RAD) [21]. Salah satu yang umum digunakan adalah model RAD untuk pengembangan sistem [22]. RAD merupakan kumpulan dari metodologi yang merupakan jawaban dari kelemahan metode pengembangan *waterfall* dan segala bentuk variasinya [23]. Dalam pengembangan sistem, RAD menggunakan metode iteratif atau berulang dimana *working model* (model kerja) sistem dibangun pada awal tahapan pengembangan dengan tujuan menetapkan *requirements* atau kebutuhan pengguna [16]. RAD sangat memungkinkan untuk digunakan pada proyek skala kecil dengan sumber daya yang sedikit dan kebutuhan pengembangan yang singkat [24].

Gambar 2. 1 Tahapan Rapid Application Development
Sumber: [24]

Gambar 2.1 menunjukkan diagram tahapan dari metode RAD. Dalam RAD juga dibagi menjadi beberapa tahapan yang terbagi menjadi empat (4), yaitu

requirements planning (perencanaan kebutuhan), *user design* (perancangan pengguna), *construction* (konstruksi), dan *cutover*. Pada tahap perencanaan kebutuhan, mendefinisikan keseluruhan kebutuhan untuk sistem, tim proyek untuk pengembangan sistem, dan analisis kelayakan (*feasibility*). Selanjutnya, tahap perancangan pengguna merepresentasikan kolaborasi antara analis sistem bersama dengan *designer* dan *programmer* untuk membuat rancangan sistem. Jika perancangan pengguna sudah selesai, maka dilanjutkan pada proses konstruksi dimana pengembang aplikasi bekerja bersama dengan pengguna untuk membangun versi-versi selanjutnya dari sistem secara interaktif. Tahapan ini dieksekusi secara paralel dengan perancangan pengguna secara berulang hingga versi dari sistem yang dapat diterima oleh pengguna dikembangkan. Tahap terakhir adalah *cutover* yang dapat dikatakan mirip dengan tahap implementasi pada SDLC. Semua tahapan yang dibutuhkan untuk proses perpindahan dari sistem lama ke sistem baru diselesaikan pada tahap *cutover* [24].

2.3. *Unified Modeling Language* (UML)

UML merupakan kumpulan dari standar teknik pembuatan diagram yang dibuat oleh Grady Booch, Ivar Jacobson, dan James Rumbaugh. Tujuan dari dibuatnya UML adalah untuk menyediakan kosakata bersifat umum untuk istilah-istilah berbasis objek dan teknik pembuatan diagram sebagai model dari proyek pengembangan sistem manapun, dimulai dari tahap analisis hingga perancangan (*design*).

Diagram Name	Used to	Primary Phase
Structure Diagrams		
Class	Illustrate the relationships between classes modeled in the system.	Analysis, Design
Object	Illustrate the relationships between objects modeled in the system. Function when actual instances of the classes will better communicate the model.	Analysis, Design
Package	Group other UML elements together to form higher level constructs.	Analysis, Design, Implementation
Deployment	Show the physical architecture of the system. Can also be used to show software components being deployed onto the physical architecture.	Physical Design, Implementation
Component	Illustrate the physical relationships among the software components.	Physical Design, Implementation
Composite Structure	Illustrate the internal structure of a class—i.e., the relationships among the parts of a class.	Analysis, Design
Behavioral Diagrams		
Activity	Illustrate business work flows independent of classes, the flow of activities in a use case, or detailed design of a method.	Analysis, Design
Sequence	Model the behavior of objects within a use case. Focuses on the time-based ordering of an activity.	Analysis, Design
Communication	Model the behavior of objects within a use case. Focuses on the communication among a set of collaborating objects of an activity.	Analysis, Design
Interaction Overview	Illustrate an overview of the flow of control of a process.	Analysis, Design
Timing	Illustrate the interaction that takes place among a set of objects and the state changes that they go through along a time axis.	Analysis, Design
Behavioral State Machine	Examine the behavior of one class.	Analysis, Design
Protocol State Machine	Illustrate the dependencies among the different interfaces of a class.	Analysis, Design
Use Case	Capture business requirements for the system and to illustrate the interaction between the system and its environment.	Analysis

Gambar 2. 2 Jenis Diagram UML
Sumber: [23]

Gambar 2.2 menunjukkan jenis diagram UML. Berikut ini adalah penjelasan dari diagram UML yang dibagi menjadi dua jenis utama, yaitu [18]:

1. *Structure diagrams*, merupakan diagram yang digunakan untuk merepresentasikan data dan relasi statis yang ada di dalam sistem informasi. Diagram yang termasuk ke dalam jenis ini adalah *class diagram*, *object diagram*, *package diagram*, *deployment diagram*, *component diagram*, dan *composite structure diagram*.
2. *Behavior diagrams*, merupakan diagram yang menyediakan cara untuk menggambarkan hubungan dinamis antara *instances* atau objek yang merepresentasikan bisnis dari sistem informasi. Diagram yang termasuk ke dalam jenis ini adalah *activity diagram*, *sequence diagram*, *communication diagram*, *interaction overview diagram*, *timing diagram*, *behavioral state machine diagram*, *protocol state machine diagram*, dan *use case diagram*.

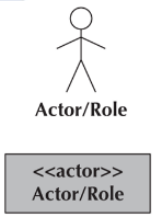
2.3.1. *Use Case Diagram*

Use case dapat memberitahu pada *high level* apa yang butuh dilakukan oleh sistem dan setiap dari teknik pembuatan diagram UML menggunakan *use*

case dengan presentasi yang berbeda karena setiap tampilan memiliki kegunaan yang berbeda. *Use case* diagram biasanya digunakan untuk meringkas rangkaian *user cases* untuk bagian logis dari sistem atau keseluruhan sistem. Sebuah *use case* dapat menjadi representasi beberapa alur atau cara yang digunakan ketika pengguna berinteraksi dengan sistem dimana alur yang dimaksud adalah *scenario*.

Sebuah *use case diagram* mengilustrasikan secara sederhana fungsi utama dari sistem dan jenis-jenis pengguna yang akan berinteraksi dengan sistem tersebut [23]. Berikut ini adalah tabel 2.2 yang berisi elemen-elemen yang terdapat di dalam sebuah *use case diagram* [23]:

Tabel 2. 2 Elemen Use Case Diagram

Istilah dan Penjelasan	Simbol
Aktor: ● Orang atau sistem yang memperoleh manfaat dari sistem dan berada di luar sistem. ● Dilabeli dengan perannya. ● Dapat dikaitkan dengan aktor lain dengan asosiasi spesialisasi atau *superclass*, yang dilambangkan dengan panah (kepala panah berongga). ● Ditempatkan di luar batas dari sistem.	
Use Case: ● Bagian utama dari fungsionalitas sistem. ● Dapat memperpanjang *use case* lainnya. ● Dapat menggunakan *use case* lainnya. ● Ditempatkan di dalam batas sistem. ● Diberi label dengan frase kata kerja dan kata benda yang deskriptif.	
Batasan Sistem: ● Termasuk nama sistem di dalam atau di atas simbol. ● Mewakili ruang lingkup sistem.	
Hubungan Asosiasi: ● Menghubungkan aktor dengan *use case* yang beinteraksi dengan aktor terkait.	

Sumber: [23]

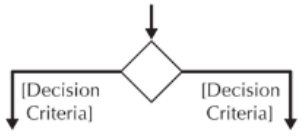
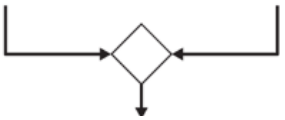
Berikut ini adalah tahapan yang dibutuhkan ketika membuat sebuah *use case diagram* [23]:

1. *Identify use case* berarti melakukan proses identifikasi terhadap *use cases* yang sesuai dengan fungsi utama dari sistem dan menyatukan semuanya menjadi satu dokumentasi *use cases*.
2. *Draw the system boundary* berarti membuat batasan sistem yang memisahkan *use cases* dari *actors*.
3. *Place the use case on the diagram* berarti menambahkan *use cases* ke dalam *system boundary* yang sebelumnya telah dibuat. *Use case* dapat dikelompokkan dengan *packages* ketika jumlah *use case* cukup banyak sehingga diagram akan lebih mudah dipahami dan menjaga model dalam tingkatan kompleksitas yang masih diterima.
4. *Identify the actors* berarti mengidentifikasi aktor-aktor yang akan berperan di dalam *use case diagram*. Identifikasi dilakukan juga dengan memperhatikan tujuan dari *input* dan *output* yang sudah ada dari proses sebelumnya.
5. *Add association relationships* berarti menghubungkan aktor dengan *use cases* dimana aktor berinteraksi dengan menggunakan garis-garis. Dalam menambahkan asosiasi, tidak terdapat keharusan untuk mengurutkan di dalam diagram, namun mungkin ada beberapa hal tambahan yang dapat dilakukan seperti mengatur ulang simbol yang ada sehingga diagram menjadi lebih mudah dipahami.

2.3.2. *Activity Diagram*

Activity diagram mengilustrasikan alur bisnis yang tidak bergantung pada kelas, alur aktivitas dalam *use case*, atau rancangan dari metode [16]. Diagram jenis ini digunakan untuk memodelkan perilaku (*behavior*) dalam suatu proses bisnis secara independen dari objek. *Activity diagram* dapat digunakan untuk memodelkan banyak hal mulai dari alur kerja bisnis tingkat tinggi yang melibatkan banyak *use case* yang berbeda, hingga detail dari sebuah *use case*, bahkan sampai ke detail spesifik dari sebuah metode. Singkatnya, diagram ini dapat digunakan untuk memodelkan jenis proses apapun. Berikut ini adalah tabel 2.3 yang berisi elemen-elemen yang terdapat di dalam *activity diagram* beserta penjelasannya [23]:

Tabel 2. 3 Elemen Activity Diagram

Istilah dan Penjelasan	Simbol
Aksi (*Action*): ● Sebuah perilaku yang sederhana dan tidak dapat diuraikan. ● Diberikan label berdasarkan namanya.	
Aktivitas (*Activity*): ● Aktivitas digunakan untuk mewakili kumpulan dari aksi (*action*). ● Diberikan label berdasarkan namanya.	
Control flow ● *Control flow* digunakan untuk menunjukkan urutan dari eksekusi sebuah *use case*.	
Initial node: ● Merupakan awal dari sekelompok *action* dan atau *activity*.	
Final-activity node: ● Merupakan akhir untuk menghentikan *control flow*.	
Decision node: ● Digunakan untuk mewakili sebuah kondisi uji yang memastikan bahwa *control flow* hanya mengikuti satu jalur. ● Diberikan label dengan kriteria keputusan untuk melanjutkan ke jalur yang lebih spesifik.	
Merge node: ● Digunakan untuk menyatukan kembali berbagai jalur keputusan yang berbeda-beda yang merupakan hasil dari *decision node*.	

Sumber: [23]

Berikut ini adalah tahapan yang dibutuhkan untuk dapat membuat sebuah *activity diagram*:

1. *Choose a business process* berarti memilih proses bisnis yang yang sebelumnya sudah diidentifikasi untuk dimodelkan ke dalam *activity diagram*. Untuk dapat dapat membuat diagram jenis ini, perlu untuk meninjau definisi kebutuhan dan *use case diagram* yang menjelaskan kebutuhannya, juga dengan dokumentasi yang dibuat selama proses pengumpulan kebutuhan, seperti wawancara atau observasi; analisis dari kuisioner; atau daftar *task* yang sudah disusun.
2. *Identify activities* berarti mengidentifikasi kumpulan dari aktivitas yang diperlukan untuk mendukung proses bisnis. Proses ini dapat dilakukan dengan meninjau *use case diagram* untuk dapat mengetahui detail dari aktivitas dari sebuah *use case* yang lebih besar.

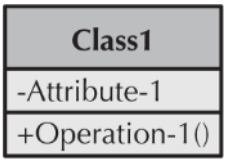
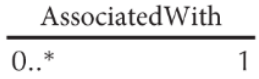
3. *Identify control flows and nodes* diperlukan untuk dapat mendokumentasikan logika dari proses bisnis. *Activity diagram* harus mencakup beberapa keputusan dan penggabungan dari *node*.
4. *Identify object flows and nodes* diperlukan untuk mendukung logika dari proses bisnis. Umumnya *control flows* dan *nodes* tidak ditunjukkan dalam *activity diagram*, kecuali ketika informasi yang ditangkap oleh sistem dalam satu aktivitas digunakan juga oleh aktivitas yang dilakukan kemudian yang tidak langsung.

   Lay out and draw diagram berarti menyusun dan menggambar diagram yang diperlukan untuk mendokumentasikan proses bisnis yang sudah melewati beberapa tahapan sebelumnya. Dalam proses ini, perlu meminimalkan potensi garis yang menyilang diantara proses sehingga estetika dan pemahaman dapat terjaga dengan baik.

2.3.3. *Class Diagram*

Class diagram adalah salah satu teknik pembuatan diagram yang merupakan model statis yang mendukung tampilan statis dari sistem yang berevolusi. Diagram ini menunjukkan kelas-kelas dan relasi-relasi diantara kelas yang bersifat konstan dari waktu ke waktu. *Class diagram* menggambarkan kelas, yang mencakup atribut (*attributes*), perilaku (*behaviors*), dan status (*states*) [23]. Berikut ini adalah tabel 2.4 yang berisi elemen-elemen yang terdapat di dalam *class diagram* beserta penjelasannya [23]:

Tabel 2. 4 Elemen Class Diagram

Istilah dan Penjelasan	Simbol
Kelas (*Class*): ● Merepresentasikan orang, tempat, atau hal yang informasinya harus disimpan oleh sistem. ● Memiliki nama yang dicetak tebal dan berada di bagian atas *compartment*. ● Memiliki sebuah daftar atribut yang terdapat di bagian tengah *compartment*. ● Memiliki sebuah daftar operasi (*operations*) yang terdapat di bagian bawah *compartment*. ● Tidak secara eksplisit menujukkan operasi yang tersedia untuk semua kelas.	
Atribut (*Attribute*): ● Properti yang menggambarkan keadaan suatu objek. ● Dapat diturunkan dari atribut lainnya, ditunjukkan dengan menempatkan *slash* atau garis miring sebelum nama atribut.	attribute name /derived attribute name
Metode (*Method*) ● Mewakili tindakan atau fungsi yang dapat dilakukan oleh sebuah kelas. ● Dapat diklasifikasikan sebagai *constructor*, *query*, atau *update operation*. ● Termasuk tanda dalam kurung yang dapat berisi parameter khusus atau informasi yang dibutuhkan untuk menjalankan operasi.	operation name ()
Hubungan Asosiasi: ● Merepresentasikan hubungan antar beberapa kelas, atau sebuah kelas dan kelas itu sendiri. ● Diberi label dengan frase kata kerja atau nama peran sesuai dengan hubungannya. ● Dapat berada diantara satu atau lebih kelas. ● Berisi simbol *multiplicity* yang menggambarkan waktu minimum dan maksimum dari sebuah *instance class* yang dapat dikaitkan dengan *instance class*.	
Agregasi (*Aggregation*): ● Mewakili hubungan *a-part-of* secara logis antara beberapa kelas atau antara kelas dengan dirinya sendiri. ● Merupakan bentuk khusus dari sebuah asosiasi.	
Generalisasi (*Generalization*): ● Mewakili hubungan jenis-kelas antara beberapa kelas.	
Komposisi (*Composition*): ● Mewakili hubungan *a-part-of* secara fisik antara beberapa kelas atau antara kelas dengan dirinya sendiri. ● Merupakan bentuk khusus dari sebuah asosiasi.	

Sumber: [23]

Class diagram dapat menjadi kompleks ketika semua kelas dan asosisasi untuk sistem digambarkan menjadi satu sehingga memerlukan proses

penyederhanaan dengan menggunakan tampilan dengan tujuan membatasi jumlah informasi yang ditampilkan [23].

Berikut ini adalah tahapan untuk membuat sebuah *class diagram*:

1. *Indentify classes* berarti mengidentifikasi kelas yang harus ditempatkan pada diagram secara menyeluruh.
2. *Identify attributes and operations* berarti mengidentifikasi atribut dan operasi yang akan dijalankan oleh sistem.
3. *Draw associations between the classes* berarti menggambar garis asosiasi untuk menghubungkan kelas-kelas yang memiliki relasi satu sama lainnya.

2.4. Basis Data

Database (basis data) merupakan kumpulan data dan deskripsinya yang terkait secara logika yang dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi. *Database* adalah penyimpanan data tunggal yang memungkinkan berukuran skala besar yang dapat digunakan secara bersamaan oleh banyak departemen dan pengguna.

Pendekatan sistem *database* memisahkan definisi data dengan program aplikasi yang membutuhkan *database* juga dengan pengguna yang hanya mengetahui tampilan eksternal tanpa menyadari bagaimana data didefinisikan beserta fungsinya [25]. Berikut ini adalah keuntungan dari pendekatan *database* [25]:

1. Sebuah definisi objek internal dapat diubah tanpa mempengaruhi pengguna dari objek terkait dengan catatan definisi eksternal dari objek tidak mengalami perubahan.
2. Pendekatan ini memisahkan struktur data dari program aplikasi yang digunakan dan penyimpanan dilakukan di dalam sebuah *database*. Jika struktur data ditambahkan atau terjadi modifikasi di dalam struktur data yang sudah ada sebelumnya, maka program aplikasi tidak akan berubah atau terpengaruh dengan catatan program aplikasi tidak bergantung secara langsung terhadap struktur data.

3. *Logically related* berarti ketika proses analisis terkait dengan kebutuhan informasi suatu organisasi, maka identifikasi terhadap entitas, atribut, dan hubungan perlu dijalankan. Entitas adalah *distinct object* dalam organisasi yang akan direpresentasikan di dalam *database*, misalnya orang; tempat; benda; dan lainnya. Atribut adalah properti yang menjelaskan beberapa aspek dari objek yang ingin direkam. Hubungan adalah asosiasi antar entitas itu sendiri.

2.5. *Entity Relationship Modeling*

ER *modeling* adalah pendekatan *top-down* untuk perancangan *database* yang dimulai dengan melakukan identifikasi terhadap data yang disebut entitas dan hubungan yang terdapat diantara data yang harus direpresentasikan ke dalam bentuk sebuah model. Di dalamnya dapat ditambahkan beberapa informasi penting lainnya yang dikenal dengan istilah atribut dan batasan. Untuk merepresentasikan konsep dari ER *modeling*, terdapat banyak notasi yang dapat digunakan dan paling umum menggunakan UML (*Unified Modeling Language*) [25]. Berikut ini adalah penjelasan mengenai setiap elemen dalam ER *modeling* [25]:

1. *Entity Types* (tipe entitas)

   Tipe entitas adalah sebuah kumpulan dari objek yang memiliki properti yang sama yang diidentifikasikan oleh perusahaan. Setiap objek yang dapat diidentifikasikan secara unik dari tipe entitas dikenal dengan istilah *entity occurrence*.

2. *Relationship Types* (tipe hubungan)

   Tipe hubungan merupakan sebuah kumpulan hubungan antara satu atau lebih tipe entitas yang berpartisipasi dimana setiap tipe hubungan memiliki nama yang mendeskripsikan fungsinya. Hubungan yang dapat diidentifikasi secara unik yang berisi satu kejadian dari setiap tipe entitas yang ada dikenal dengan istilah *relationship occurrence*.

3. Atribut

   Atribut adalah properti yang menjelaskan beberapa aspek dari objek yang ingin direkam dimana objek yang dimaksud adalah entitas. Atribut memiliki nilai

yang mendeskripsikan setiap dari *entity occurrence* dan merepresentasikan bagian utama dari data yang disimpan di dalam *database*. Setiap atribut memiliki hubungan dengan kumpulan nilai yang disebut dengan *domain*.

4. *Keys* (kunci)

   Berikut ini adalah pembagian jenis dari *keys* yang terdapat dalam ER *modeling*:

   a. *Primary key* merupakan *candidate key* yang dipilih secara unik untuk mengidentifikasikan setiap kejadian dari tipe entitas.

   b. *Foreign key* merupakan atribut atau kumpulan atribut dalam hubungan yang dibandingkan dengan *candidate key* pada beberapa relasi.

   c. *Alternate key*, merupakan *candidate key* yang tidak terpilih sebagai *primary key*.

2.6. PHP (*Hypertext Preprocessor*)

PHP merupakan bahasa skrip yang dapat ditanamkan atau disisipkan ke dalam HTML yang berguna untuk membuat aplikasi *web* atau halaman *web* [12]. PHP berguna untuk memproses pengolahan data-data yang dikumpulkan dari *form* HTML untuk diproses dan dikeluarkan kembali menjadi suatu informasi baru [26]. Bahasa pemrograman PHP ini dirancang khusus untuk membentuk sebuah *website* yang dinamis [1]. PHP memiliki beberapa kelebihan, yaitu[27]:

1. *Performance* karena PHP dapat dikatakan sangat cepat karena menggunakan sebuah *inexpensive server* (*server* yang tidak mahal) yang memungkinkan jutaan aktivitas didalamnya.
2. *Scalability* karena PHP memiliki arsitektur *shared-nothing* (tidak berbagi) yang berarti dapat dengan efektif dan dengan biaya yang murah untuk menerapkan *horizontal scaling* dengan sejumlah besar *server*.
3. *Database integration* karena PHP memiliki koneksi *native* yang tersedia untuk banyak sistem *database*, salah satunya tersedia untuk MySQL.
4. *Built-in libraries* karena PHP dirancang untuk digunakan pada *web* memiliki banyak fungsi bawaan untuk melakukan banyak tugas terkait dengan *web*, misalnya *generate* gambar dengan cepat, menghasilkan dokumen PDF, dan lainnya dengan menggunakan beberapa baris *code*.

5. *Cost* karena PHP disediakan gratis dan dapat diunduh kapanpun tanpa biaya tambahan.

2.7. *Hypertext Markup Language* (HTML)

HTML adalah fondasi dari *World Wide Web* (WWW), sebuah bahasa markup yang memungkinkan pembentukan halaman web. Dokumen web, yang juga dikenal sebagai dokumen HTML, adalah file teks yang telah dienkripsi menggunakan HTML, sehingga dapat ditampilkan dengan format yang sesuai di peramban *web* [28]. Berikut adalah rincian tentang setiap elemen HTML [28]:

1. *Hypertext Text* adalah teks yang diklik untuk melompat atau berpindah dari satu dokumen ke dokumen lainnya.
2. *Markup Tag* menerapkan *layout* (tata letak) dan format ke dalam teks polos dimana teks biasanya ditandai dengan *tags*.
3. *Language* atau bahasa mengartikan bahwa HTML dianggap sebagai sebuah bahasa pemrograman.

2.8. *Cascading Style Sheets* (CSS)

Cascading Style Sheets (CSS) adalah bahasa yang digunakan untuk mengatur tampilan dokumen yang ditulis dalam bahasa *markup*. Secara esensial, CSS berfungsi sebagai alat untuk menentukan penampilan atau desain halaman HTML [29]. Dengan CSS, pengguna dapat mengontrol berbagai aspek tampilan seperti jenis huruf, warna, gambar latar belakang, jarak baris, dan tata letak halaman. Selain berlaku untuk *browser desktop*, CSS juga dapat mengatur tampilan dokumen untuk berbagai media lainnya, termasuk perangkat dengan layar kecil. *Style sheets*, dalam konteks ini, adalah instrumen yang efektif untuk meningkatkan efisiensi produksi, karena memungkinkan perubahan tampilan secara konsisten pada seluruh halaman situs hanya dengan melakukan modifikasi pada satu dokumen *style sheets* [30].

2.9. XAMPP

XAMPP adalah paket perangkat lunak sumber terbuka yang terdiri dari PHP dan MySQL, yang digunakan sebagai alat bantu dalam pengembangan aplikasi berbasis PHP. Di dalamnya terdapat tiga komponen utama, yakni Apache sebagai server web, PHP sebagai bahasa pemrograman, dan MySQL sebagai sistem manajemen basis data [1]. XAMPP merupakan distribusi Apache yang ringan dan

kompak, menggabungkan teknologi pengembangan *web* yang umum digunakan dalam satu paket. XAMPP merupakan paket perangkat lunak yang digunakan untuk mempelajari pemrograman *web* [12].

2.10. MySQL

MySQL (*My Structure Query Language*) adalah sistem *database* populer paling lazim digunakan di PHP digunakan di lingkungan *web*, yang memiliki sifat *scalable* artinya bisa dipakai oleh program kecil hingga program besar. MySQL adalah perangkat lunak RDBMS yang dapat mengolah *database* dengan cepat, dapat menampung data dalam jumlah besar, dapat diakses oleh banyak *user* dan dapat melakukan suatu proses secara sinkron atau bersamaan [12]. MySQL merupakan perangkat lunak yang bersifat *open source* [31]. Tingkat tertinggi dari struktur MySQL adalah *database* yang memungkinkan pengguna untuk memiliki satu atau lebih tabel yang berisi data-data dari pengguna [32].

2.11. Pembelian

Pembelian adalah langkah bisnis dalam memilih, memesan, dan mendapatkan barang atau jasa yang diperlukan [1]. Pembelian mencakup transaksi untuk memperoleh barang atau mengeluarkan uang untuk memperoleh produk yang akan dijual, dengan transaksi terjadi antara pembeli dan pemasok barang. Pembelian adalah tindakan mengeluarkan sejumlah uang untuk mendapatkan barang atau produk sesuai dengan kebutuhan yang diinginkan [12].

2.12. Penjualan

Penjualan merupakan aktivitas pemasaran yang bertujuan untuk memperoleh pendapatan dan memenuhi kebutuhan pembeli [12]. Penjualan merupakan upaya konkret untuk mentransfer produk, baik barang maupun jasa, dari produsen kepada konsumen. Tujuan utamanya adalah untuk menghasilkan keuntungan atau laba dari produk yang diproduksi oleh produsen dengan manajemen yang efisien [21].

2.13. *Software Testing*

Testing atau pengujian adalah sebuah tindakan untuk menjalankan perangkat lunak dengan menggunakan *test cases* [33]. *Software testing* merupakan bagian dari SDLC untuk mengidentifikasi *defects* (cacat) atau *errors* yang terdapat di dalam

aplikasi dengan tujuan untuk menegaskan kualitas dari sistem [34]. *Software testing* merupakan alat dari *quality assurance* perangkat lunak yang pertama kali diaplikasikan untuk mengendalikan kualitas dari produk perangkat lunak sebelum dilakukan instalasi bagi pengguna sistem [35].

Berikut ini adalah istilah umum dalam *testing*, yaitu [35]:

1. *Test plan* adalah sebuah dokumen yang memberikan informasi mengenai tujuan, ruang lingkup, pendekatan, dan berbagai atribut yang harus menjad fokus dari proyek pengujian.
2. *Test case* adalah sebuah dokumen dan merupakan unit terkecil dari pengujian. Tujuan dari *test cases* adalah memastikan bahwa fitur dari aplikasi yang diuji berfungsi seperti yang diharapkan

Pengujian juga dibagi menjadi dua jenis utama yang umum digunakan, yaitu [36]:

1. Pengujian kotak putih *(white box testing)* adalah metode pengujian yang memeriksa rincian desain, menggunakan struktur kontrol program yang terinci secara prosedural untuk membagi pengujian menjadi berbagai kasus uji.
2. Pengujian kotak hitam *(black box testing)* adalah jenis pengujian yang berfokus pada pengujian spesifikasi fungsional perangkat lunak di mana pengujian dapat menentukan serangkaian kondisi masukan dan menguji spesifikasi fungsional program tersebut.

2.13.1. *Acceptance Testing*

Acceptance testing adalah sebuah proses membandingkan persyaratan awal program dengan kebutuhan *end users* di masa sekarang [37]. *Acceptance testing* merupakan salah satu tingkatan dari *testing* berdasarkan pada aktivitas dari *software* yang melakukan penilaian terhadap *software* sehubungan dengan persyaratan atau kebutuhan dari pengguna. Pengujian ini dirancang untuk menentukan apakah *software* secara nyata sudah memenuhi kebutuhan pengguna. Dengan kata lain, pengujian ini menyelidiki apakah *software* sudah melakukan apa yang diinginkan oleh pengguna [38].