BAB II
TINJAUAN PUSTAKA

2.1 Konsep Sistem Informasi

Sistem merupakan interaksi antar elemen yang saling berkaitan, terhubung dan saling mempengaruhi satu sama lain untuk mencapai suatu tujuan tertentu. Secara konseptual suatu sistem memiliki minimal tiga elemen yang terdiri dari *input*, proses, dan *output*. Akan tetapi didalam beberapa studi mengatakan jika suatu sistem memiliki elemen keempat yaitu umpan balik. Elemen fisik pada sistem dapat berupa data, prosedur, *software, hardware*, dan manusia [3]. Informasi adalah sebuah data terstruktur yang sudah diolah sedemikian rupa agar sehingga memiliki makna dan nilai bagi penggunanya, serta memungkinkan untuk digunakan dalam pengambilan keputusan. Berdasarkan pengertian tersebut, maka secara fundamental sistem informasi dapat dipahami sebagai suatu sistem yang tidak hanya terdiri elemen fisik tetapi juga secara khusus dirancang untuk memproses data menjadi sebuah nilai yang berharga dalam menentukan pengambilan keputusan strategis. Sistem inforamsi tidak hanya sekedar teknologi, tetapi juga bagaimana teknolgi tersebut dimanfaatkan untuk mendukung proses bisnis, operasional, analisis, dan pengambilan keputusan [4].

Sistem informasi lebih dari sekedar teknologi, karena sistem informasi merupakan suatu sistem terintegrasi yang melibatkan interaksi kompleks antara komponen teknologi dengan proses bisnis yang dirancang untuk mencapai tujuan strategis dan operasional [5]. Oleh karena itu untuk dapat beroperasi secara maksimal, maka sebuah sistem informasi dibangun atas beberapa komponen yang saling terintegrasi. Komponen – komponen utama sistem informasi terdiri dari:

1. Blok Masukan

   Blok masukan merupakan blok penting yang menjadi langkah awal sistem untuk memproses sebuah data didalam sistem. Blok masukan dapat menerima jenis masukan berupa angka, gambar, dan data penting lainnya.

2. Blok Model

   Blok ini terdiri dari serangkaian prosedur, logika, dan model matematika yang berguna memproses dan memanipulasi data yang sudah diinputkan sebelumnya, sehingga nanti menghasilkan keluaran yang diinginkan.

3. Blok Keluaran

Setelah sistem berhasil memproses data, maka akan mengeluarkan hasil akhir melalui blok keluaran. Hasil akhir ini menjadi sebuah informasi penting yang dapat digunakan perusahaan untuk pengambilan keputusan.

4. Blok Teknologi

   Teknologi pada sistem informasi menjadi seperti sebuah wadah yang berfungsi untuk menerima inputan, menjalankan model dan proses, hingga menghasilkan keluaran. Blok teknologi membantu pengendalian dari sistem secara keseluruhan.

5. Blok Basis Data

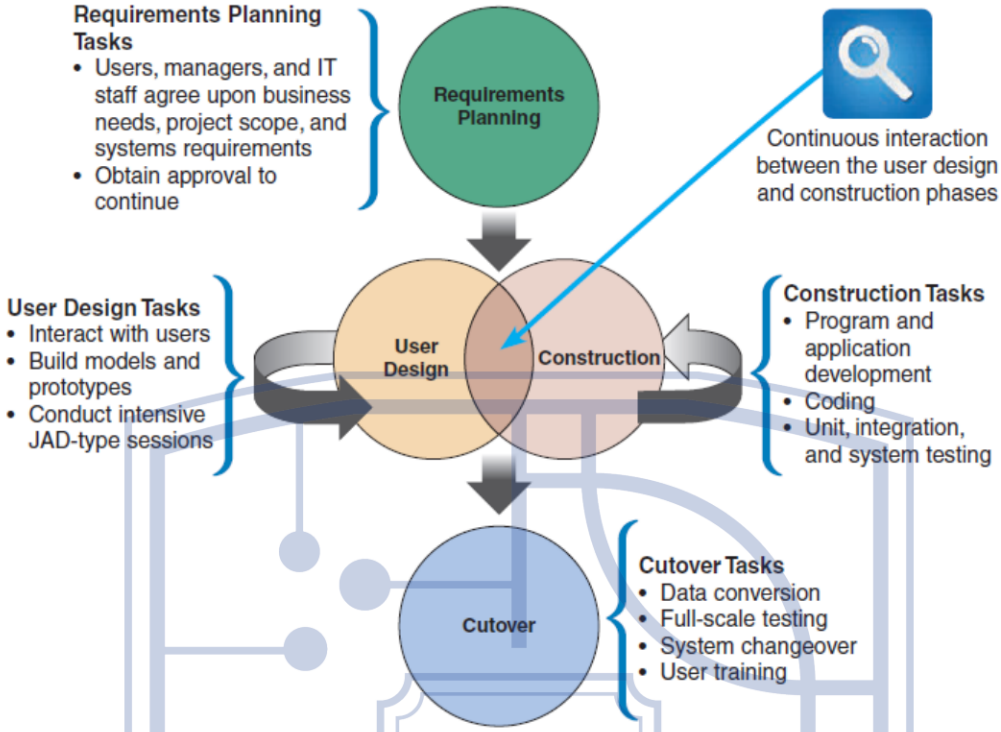
   Basis data merupakan sebuah kumpulan data terstruktur dan saling berhubungan yang disimpan secara digital agar mudah diakses, dikelola, dan diproses. Sistem informasi dapat memproses dan memanipulasi data pada basis data melalui perangkat lunak dan biasa dikenal sebagai *Database Management System* (DBMS)

6. Blok Kendali

   Blok kendali dirancang dan diterapkan untuk mengatasi terjadinya pemeliharaan agar menghindari sistem dari kerusakan dan ketika terjadi kesalahan, maka dapat dengan cepat diatasi.

2.2 RAD (*Rapid Application Development*)

Rapid Application Development atau RAD merupakan metodologi pengembangan sistem yang menekankan siklus pengembangan cepat dan singkat dibandingkan dengan metodologi pengembangan sistem tradisional. RAD merupakan salah satu metodologi pengembangan sistem yang berorientasi objek dengan menggunakan pendekatan iteratif dan *component-based construction* [3]. RAD merupakan metodologi yang diadaptasi dari prinsip air terjun yang dirancang mencapai kecepatan pengembangan maksimal [6]. Berdasarkan kedua opini penulis tersebut maka kita dapat mengambil kesimpulan jika RAD merupakan metodologi pengembangan sistem yang lengkap dan dapat dikembangkan dalam waktu yang singkat dibandingkan metodologi tradisional, caranya melalui pembuatan prototipe yang dapat digunakan kembali dan keterlibatan pengguna yang intensif.



Gambar 2. 1 Model Pengembangan RAD

Sebagai sebuah metodologi, RAD memiliki tahapan – tahapan yang dirangkum menjadi 4 fase utama secara berurutan. Berikut tahapan - tahapan pengembangan sistem dengan menggunakan metodologi *Rapid Application Development* (RAD) [6]:

1. *Requirements Planning*

   Fase ini merupakan tahapan awal dimana perencanaan, dan analisis kebutuhan dilakukan secara bersamaan. Tahapan ini bersifat kolaboratif karena melibatkan manajer, tim IT, dan pengguna untuk menetapkan kebutuhan bisnis, ruang lingkup sistem, tujuan sistem, serta kendala - kendala yang mungkin dihadapi dikemudian hari. Tim IT melakukan observasi untuk mendeskripsikan kendala yang dihadapi pengguna dan manajer sehingga pengembangan sistem dibutuhkan. Fase perencanaan ini berakhir ketika tim menyepakati masalah utama dan memperoleh persetujuan manajemen untuk melanjutkan.

2. *User Design*

   Fase ini merupakan inti dari metodologi RAD, karena pengguna terlibat secara aktif dengan analis sistem untuk memastikan model dan prototipe sistem. Durasi berjalannya fase ini bergantung dengan seberapa kompleksnya akan kebutuhan sistem. Fase ini bersifat interaktif dan memungkinkan terjadinya modifikasi untuk menemukan prototipe terbaik, hingga pengguna menyetujui cara kerja sistem.

3. *Construction*

Fase ini berfokus pada pengembangan sistem secara teknis berdasarkan desain prototipe yang telah disetujui sebelumnya, lalu pengembang akan melakukan integrasi dan pengujian secara interaktif. Salah satu hal yang membuat RAD berbeda dengan metodologi lain adalah karena pada fase ini pengguna dan analis sistem masih terlibat aktif sehingga dapat memberikan perubahan saat sistem dikembangkan. Biasanya fase *contruction* dan *user design* berjalan bersamaan sehingga dikenal sebagai fase iterasi.

4. *Cutover*

   Fase ini mirip dengan fase implementasi pada metodologi SDLC (*System Development Life Cycle*), akan tetapi pada RAD setiap kegiatan implementasi yang ada di SDLC justru dipadatkan sehingga proses terjadi dalam waktu yang lebih singkat. Fase cutover meliputi kegiatan transisi dari sistem lama ke sistem baru.

Dengan empat fase yang sudah dijelaskan sebelumnya, maka dapat diketahui bahwa RAD merupakan metodologi pengembangan sistem yang mengedepankan pengembangan sistem dengan waktu yang singkat. Jika bandingkan dengan teknik pengembangan sistem lainnya seperti *waterfall*, RAD justru menawarkan fleksibilitas dalam melakukan perubahan dan adaptasi saat proses pengembangan dilakukan, berbeda dengan *waterfall* yang menggunakan teknik pengembangan linier. Berbeda dengan metode *iterative* yang membutuhkan beberapa fase pengembangan, RAD justru membuat fase iterasi yang mengedepankan kecepatan sehingga dapat langsung diimplementasikan sesegera mungkin. Tidak seperti metodologi spiral yang memerlukan analisis risiko, RAD justru lebih ringan dan praktis sehingga lebih hemat dalam penggunaan sumber daya, sehingga cocok untuk pengembangan proyek skala kecil [6].

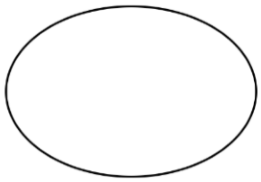
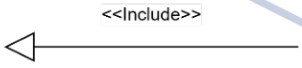
2.3 Teknik Pengembangan Sistem

2.3.1 *Use Case Diagram*

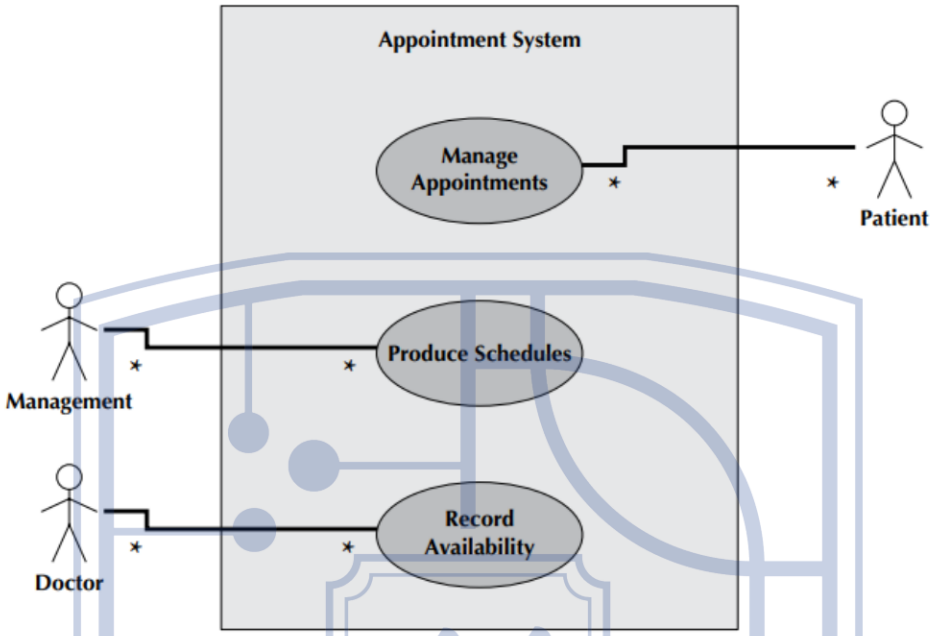
Use case diagram adalah sebuah teknik pemodelan yang sering digunakan pada fase *requirements planning*. *Use case* merupakan salah satu jenis diagram perilaku (*behavior diagram*) dalam UML (*Unified Modeling Language*) yang menunjukan interaksi pengguna dengan sistem [6]. Model *use case* memudahkan pengguna untuk memahami interaksi dengan sistem hanya dengan melihat gambaran sistem tanpa harus mendetailkan cara kerja sistem. Representasi *visual* dari model *use case* memberikan ilustrasi hubungan pengguna dengan cara kerja sistem.

Simbol – simbol yang digunakan pada diagram *use case* adalah sebagai berikut [6]:

Tabel 2. 1 Simbol *Use Case Diagram*

No	Simbol	Keterangan
1		*Use Case*: Menggambarkan interaksi sistem dengan aktor dalam sebuah proses.
2		*Actor*: Merepresentasikan pengguna, baik pengguna individu maupun sistem eksternal, atau alat ketika berkomunikasi dengan *use case*.
3		*Generalization*: Menggambarkan hubungan aktor dengan *use case*. Interaksi antara aktor dan *use case* tersebut terjadi secara pasif dengan sistem.
4		*Extend*: Menunjukkan jika sebuah *use case* secara bersyarat dapat memperluas perilaku dari *use case* lain. Contohnya: "Melakukan pembayaran" <<extend>> "Kartu kredit"
5		*Include*: Menjelaskan jika sebuah *use case* wajib menyertakan perilaku dari *use case* lain. Contohnya: "Melakukan pembayaran" <<include>> "*Login*"
6		*Association*: Menjelaskan hubungan antara aktor dengan *use case* yang sedang berinteraksi.

Berikut adalah contoh dari *use case diagram* yang digunakan untuk mengembangkan sistem informasi untuk melakukan janji temu dengan dokter [7]:



Gambar 2. 2 Contoh *Use Case Diagram*

Berikut ini adalah contoh isian dari narasi *use case* dan fungsi dari setiap *field* narasi *use case* [6]:

Tabel 2. 2 Fungsi Dari Setiap Field Pada Narasi *Use Case*

Nama use case	Disini nama *use case* didefinisikan dengan jelas.	
Aktor	Disini nama setiap aktor dijelaskan.	
Description	*Field* ini menjelaskan proses yang akan dilakukan *use case*.	
Precondition	Bagian ini menjelaskan kondisi yang dilakukan sebelum *use case*.	
Main Flow	Bagian ini menjelaskan aktivitas yang dilakukan aktor.	
	Kegiatan aktor	Respon Sistem
	Aktivitas aktor	Aktivitas sistem
	Bagian ini menjelaskan kegiatan aktor.	Bagian ini menjelaskan kegiatan sistem.
Alternatif flow	Kegiatan lain yang dapat dilakukan aktor.	
	Kegiatan aktor	Respon sistem
	Aktivitas aktor	Aktivitas sistem
	Bagian ini menjelaskan kegiatan alternatif aktor.	Bagian ini menjelaskan kegiatan alternatif sistem.

Postcondition	Kondisi setelah *use case* berhasil dijalankan.
Assumption	Asumsi yang terjadi setelah aktivitas *use case* selesai dijalankan.

Dalam penggunaannya, *use case* biasanya selalu dilengkapi dengan narasi yang berfungsi mendeskripsikan tugas dan fungsi dari *use case* yang digunakan, berikut salah satu contoh narasi *use case* dari contoh *use case* diatas [6]:

Tabel 2. 3 Contoh Narasi *Use Case*

<table>
<tr><td>Nama use case</td><td colspan="2">Membuat Jadwal Konsultasi</td></tr>
<tr><td>Aktor</td><td colspan="2">Manajer.</td></tr>
<tr><td>Description</td><td colspan="2">Mendeskripsikan jadwal dokter yang tersedia.</td></tr>
<tr><td>Precondition</td><td colspan="2">Manajer menjadwalkan konsultasi.</td></tr>
<tr><td rowspan="5">Main Flow</td><td colspan="2">Bagian ini menjelaskan aktivitas yang dilakukan aktor.</td></tr>
<tr><td>Kegiatan aktor</td><td>Respon Sistem</td></tr>
<tr><td>1. Memilih menu jadwal dokter.</td><td>2. Menampilkan daftar jadwal dokter.</td></tr>
<tr><td>3. Memilih jadwal dokter terkait.</td><td>4. Menampilkan menu jadwalkan dokter.</td></tr>
<tr><td>5. Menyimpan jadwal konsultasi.</td><td>6. Menginformasikan dokter mengenai jadwal konsultasi.</td></tr>
<tr><td rowspan="4">Alternatif flow</td><td colspan="2">Kegiatan lain yang dapat dilakukan aktor.</td></tr>
<tr><td>Kegiatan aktor</td><td>Respon sistem</td></tr>
<tr><td>3. Mencari jadwal dokter terbaik.</td><td>4. Menampilkan daftar dokter sesuai yang dicari.</td></tr>
<tr><td>5. Menyimpan jadwal konsultasi.</td><td>6. Menginformasikan dokter terkait.</td></tr>
<tr><td>Postcondition</td><td colspan="2">Konsultasi berhasil dijadwalkan.</td></tr>
<tr><td>Assumption</td><td colspan="2">-</td></tr>
</table>

Inti dari diagram ini adalah memvisualisasikan kebutuhan fungsional dari perusahaan terhadap sistem informasi dengan menggunakan perspektif pengguna. Dengan memvisualisasikan menggunakan *use case*, maka akan lebih mudah menetapkan ruang lingkup sistem yang akan dikembangkan. *Use case diagram* biasanya menjadi diagram pertama yang

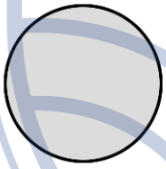
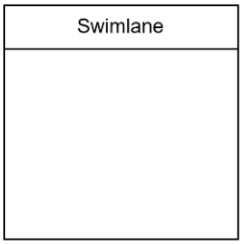
dibuat dalam UML, karena komponen seperti aktor, dan lainnya dapat kembali digunakan sebagai masukan untuk membuat diagram yang lebih rinci seperti *activity diagram*.

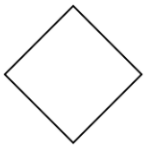
2.3.2 *Activity Diagram*

Activity diagram dikenal juga sebagai diagram perilaku yang didefinisikan sebagai alat untuk memodelkan alur kerja dan urutan aktivitas dalam sebuah sistem, baik dilakukan oleh orang, sistem, atau kombinasi dari keduanya [6]. Model dari *activity diagram* merepresentasikan setiap aktivitas – aktivitas yang saling terhubung, bercabang, atau berjalan secara paralel. Diagram ini sangat membantu analis sistem untuk menganalisis dan merancang ulang proses bisnis, bahkan lebih kompleksnya dapat membantu mengidentifikasi langkah atau aktivitas berlebihan yang mengakibatkan alur kerja menjadi lamban. *Activity diagram* bertugas mendetailkan langkah – langkah lebih dari satu *use case*, oleh karena itu biasanya *activity diagram* dibuat setelah *use case* selesai didefinisikan sebelumnya.

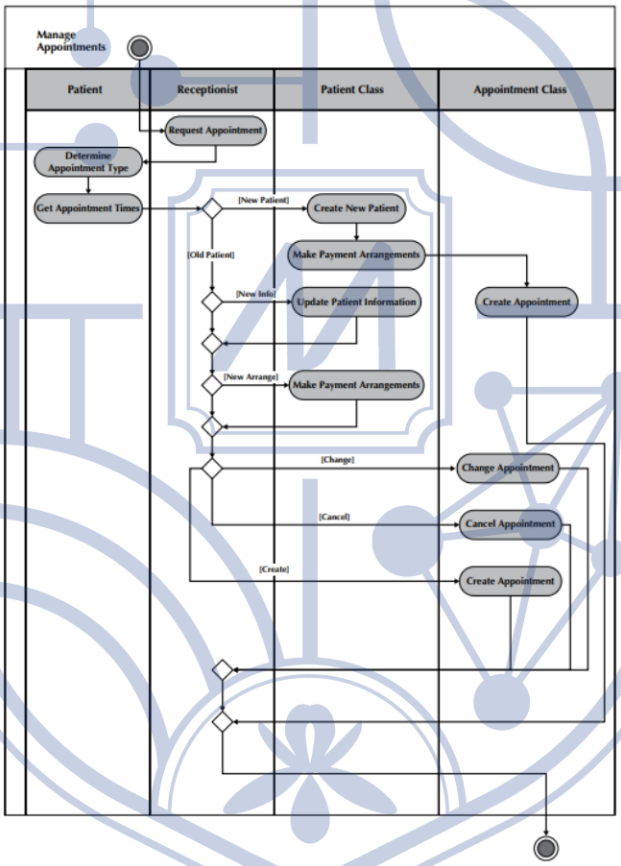
Sebagai salah satu model visualisasi aktivitas pengguna terhadap sistem, maka *activity diagram* juga memiliki simbol yang berfungsi sebagai representasi tindakan yang dilakukan. Berikut simbol – simbol yang umum digunakan pada *activity diagram* [6]:

Tabel 2. 4 Simbol *Activity Diagram*

No	Simbol	Keterangan
1		*Start*: Didefinisikan sebagai titik awal untuk memulai aktivitas di dalam sistem.
2		Aktivitas: Setiap kegiatan yang dilakukan pengguna akan dideskripsikan pada blok aktivitas.
3		*Arrow*: Menunjukkan alir aktivitas yang akan dilakukan sistem apabila blok aktivitas selesai dieksekusi.
4		*Swimlane*: Menunjukkan sebuah entitas yang bertanggung jawab atas kegiatan atau aktivitas yang terjadi padanya.

5		*Decision:* Menunjukkan sebuah kondisi yang memiliki aktivitas lebih dari satu.
6		*End Point*: Menandakan sebagai titik akhir, dimana sistem sudah selesai menjalankan setiapkan aktivitas berdasarkan penggunanya.

Berikut ini adalah contoh *activity diagram* sederhana [7]:



Gambar 2. 3 Contoh *Activity Diagram*

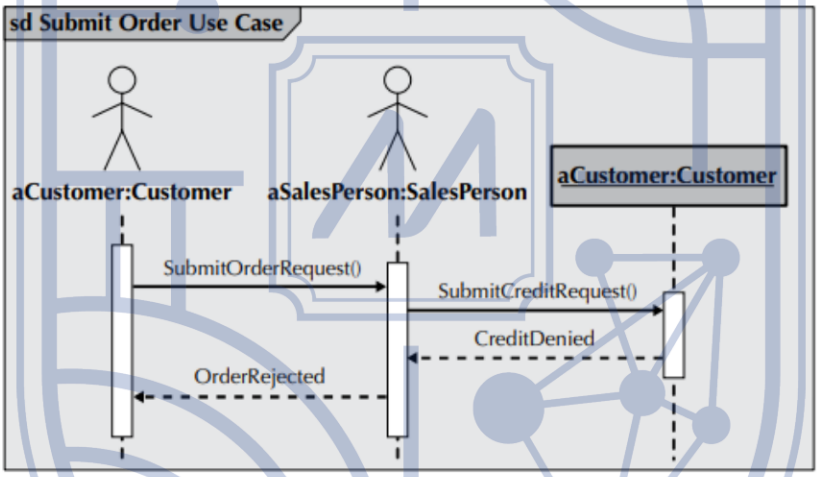
Model *activity diagram* dianggap memiliki ciri khas tersendiri karena memiliki *swimlane* yang berguna untuk memisahkan hubungan setiap entitas pengguna terhadap sistem. Penggunaan *swimlane* dianggap sangat efektif dalam memodelkan proses bisnis karena jelas menunjukkan arah dan tanggung jawab setiap pengguna yang terlibat, sehingga memudahkan kolaborasi dan analisis peran dalam sistem. Seperti yang diketahui, *activity diagram* tidak bisa berdiri sendiri tanpa menggunakan use case sebelumnya, karena *activity diagram* memiliki

kelebihan dalam menganalisis dan merincikan aktivitas setiap pengguna, sementara use case diagram justru fokus pada apa yang dilakukan sistem.

2.3.3 *Sequence Diagram*

Sequence diagram merupakan salah satu jenis *Unified Modeling Language* (UML) yang sering digunakan untuk menggambarkan interaksi pengguna dengan sistem berdasarkan urutan waktu. Diagram ini sering digunakan untuk mendetailkan aktivitas yang terjadi dari *use case diagram*, selain itu diagram ini juga dapat membantu pengembang untuk menunjukkan logika dan metode dari sebuah proses bisnis. Biasanya *sequence diagram* berfokus pada urutan pesan (*message sequence*) yang ditukarkan antar objek selama eksekusi sebuah *use case* atau operasi.

Berikut adalah contoh penggunaan *sequence diagram* sederhana [7]:

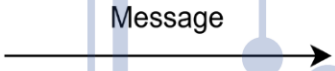
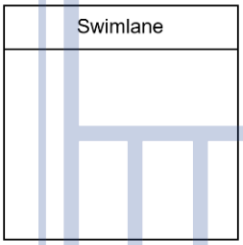


Gambar 2. 4 Contoh *Sequence Diagram*

Selain contoh tersebut diatas, berikut simbol – simbol yang sering digunakan dalam memvisualisasikan *sequence diagram* [6]:

Tabel 2. 5 Simbol *Sequence Diagram*

No	Simbol	Keterangan
1		*Class*: Berfungsi untuk mengirimkan atau menerima pesan yang ditampilkan dari bagian atas *sequence diagram*.

2		*Lifeline*: Simbol ini menandakan waktu selama objek dibagian atasnya dapat berinteraksi dengan objek lain dalam sebuah aktivitas dengan sistem.
3		*Message*: Simbol ini menampilkan pesan dan informasi yang dikirimkan antar objek dalam setiap aktivitas. Selain itu simbol ini juga dapat membawa informasi tambahan tentang isi pesan.
4		*Swimlane*: Menunjukkan sebuah entitas yang bertanggung jawab atas kegiatan atau aktivitas yang terjadi padanya.

Sequence diagram sangat berperan penting bagi analis dan pengembang sistem untuk memahami alur logika yang kompleks secara visual. *Sequence diagram* diposisikan sebagai alat pengembangan sistem yang sangat penting, efektif, dan efisien untuk memodelkan kegiatan dan aktivitas dinamis dari sistem. Selain itu *sequence diagram* juga memiliki peran untuk menjembatani kesenjangan antara kebutuhan perusahaan dengan desain teknis yang berorientasi dengan objek yang akan dikembangkan oleh pengembang atau dalam hal ini tim IT.

2.3.4 *Class Diagram*

Class diagram merupakan sebuah fondasi dari model desain sistem berorientasi objek, dimana diagram ini merepresentasikan struktur statis sistem dengan menggunakan hubungan antar kelas. Selain memiliki kelas untuk saling berhubungan, diagram ini juga memiliki atribut dan metode yang saling berhubungan. Sebuah diagram kelas biasanya digambarkan sebagai sebuah kotak yang terdiri dari tiga bagian utama, yaitu; nama kelas, atribut, dan metode. Atribut

merepresentasikan sebagai karakteristik data, sedangkan metode menjelaskan tentang perilaku atau fungsi yang dapat dilakukan oleh kelas [6].

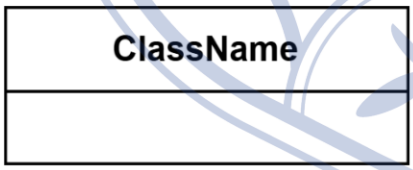
Selain menampilkan kelas dan komponennya, diagram kelas juga menggambarkan hubungan yang disebut dengan *relationships*. Hubungan tersebut terdiri atas tiga jenis yang dikenal sebagai *association, aggregation*, dan *composition* atau *inheritance.* Biasanya setiap hubungan dilengkapi dengan notasi kardinalitas seperti berikut [6]:

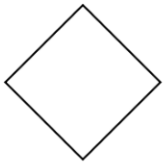
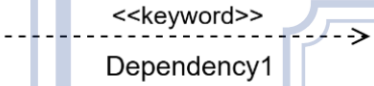
Tabel 2. 6 Kardinalitas *Class Diagram*

No	Notasi UML	Sifat Hubungan	Contoh
1	0..*	*Zero to Many*	Karyawan memiliki atau tidak memiliki potongan gaji.
2	0..1	*Zero to One*	Seorang karyawan hanya dapat memiliki satu pasangan atau tidak.
3	1	*One to One*	Seorang manajer hanya dapat mengelola sebuah perusahaan saja.
4	1..*	*One to Many*	Sebuah orderan dapat memiliki banyak produk yang dipesan.

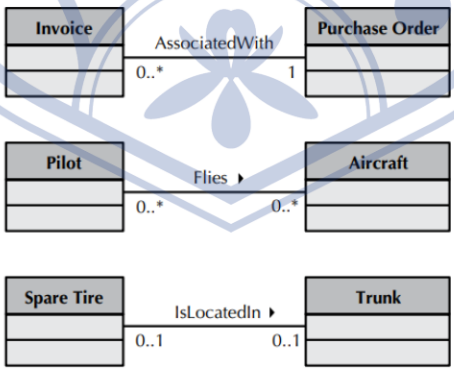
Selain memiliki hubungan kardinalitas, *class diagram* juga memiliki simbol – simbol yang digunakan, berikut simbolnya [7]:

Tabel 2. 7 Simbol *Class Diagram*

No	Simbol	Keterangan
1		*Class*: Merupakan respresentasi dari entitas atau objek. Sebuah class memiliki atribut yang merepresentasikan data dari objek tersebut.
2		*Generalization*: Hubungan antara objek anak (*descendent*) yang mewarisi sifat atau perilaku dari objek orang tua (*ancestor*)

3		*Navy Association*: Merepresentasikan pembagian hubungan asosiasi yang melibatkan lebih dari dua kelas.
4		*Collaboration*: Merepresentasikan bagaimana objek bekerja sama untuk menghasilkan suatu hasil yang terukur.
5		*Realization*: Merepresentasikan operasi yang dilakukan sebuah objek.
6		*Dependency*: Hubungan dimana perubahan terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya.
7		*Association*: Menunjukkan hubungan struktural antar dua kelas.

Berikut adalah contoh *class diagram* untuk sebuah kegiatan penjualan pada sebuah sistem informasi penjualan [7]:



Gambar 2. 5 Contoh *Class Diagram*

Class diagram memiliki peran penting pada siklus pengembangan sistem karena diagram ini memberikan visual yang general tentang struktur sistem sehingga tim IT dapat memahami komponen – komponen sistem yang akan dibangun. Setiap kesalahan dapat

diminimalkan dengan menggunakan *class diagram* karena struktur data relasi yang divisualisasikan dengan jelas mulai dari awal pengembangan.

2.4 Basis Data

Basis data atau *database* merupakan sekumpulan data terstruktur yang saling berhubungan dan disimpan pada suatu media penyimpanan digital. Di dalam sebuah *database*, setiap data disusun dalam bentuk tabel atau yang lebih dikenal sebagai entitas, dimana setiap entitas tersebut memiliki *field* atau atribut [8]. Entitas adalah objek yang dapat didefinisikan sebagai orang, divisi, tempat, benda, dan hal lain yang datanya bisa disimpan dan dianggap bernilai. Sedangkan *field* adalah informasi – informasi dari entitas yang akan disimpan, seperti contohnya: nama, usia, minat, alamat, nomor telepon, dan informasi lainnya. Setiap entitas didalam *database* memiliki hubungan satu sama lain, setiap hubungan bisa berupa one-to-many (1:N), one-to-one (1:1), dan many-to-many (M:N). Konsep hubungan antar entitas ini sering disebut dengan kardinalitas relasi atau sebagai menyebutnya dengan relasi entitas (*entitiy relationship*) [8].

Sistem database memiliki beberapa komponen seperti data, dan perangkat lunak untuk mengelola data tersebut. Perangkat lunak untuk mengelola database dikenal sebagai *Database Management System* (DBMS) [9]. Ada banyak jenis DBMS yang sering digunakan dalam pengembangan sistem, biasanya sebuah pengembang akan menggunakan *database* yang tepat dan cocok dengan sistem yang sedang dikembangkannya. Beberapa *database* yang terkenal seperti berikut:

1. MySQL

   MySQL merupakan salah satu jenis *database* yang paling populer digunakan untuk pengembangan sistem berbasis *website*. MySQL menjadi pilihan utama dikarenakan cara penggunaan dan cara konfigurasi yang mudah. Konfigutasi MySQL dengan berbagai bahasa pemrograman dapat dilakukan dengan mudah tanpa harus mengubah pengaturan yang ada pada sebuah *framework* bahasa pemrograman.
2. Microsoft SQL Server

   Selain MySQL, *Microsoft SQL Server* juga menjadi salah satu jenis *database* yang paling populer dikalangan pengembang sistem informasi. *SQL Server* mendukung pengembangan aplikasi berbasis desktop, *website*, dan aplikasi lainnya.
3. PostgreSQL

PostgreSQL adalah salah satu jenis *database* yang sedang naik pamornya, hal ini terjadi karena tingkat kekuatan dan keamanan data yang ditawarkan juga lebih unggul dibandingkan jenis database lain.

2.5 Aplikasi Web

Aplikasi web adalah salah satu jenis perangkat lunak komputer yang dapat diakses dengan melalui internet menggunakan *web browser*. Tidak seperti aplikasi *desktop*, aplikasi *website* dapat bekerja tanpa dilakukan proses instalasi pada perangkat pengguna, sehingga dapat memudahkan pengguna untuk mengaksesnya tanpa harus membuka komputer, atau artinya pengguna dapat menggunakan *handphone* untuk mengaksesnya. Aplikasi web bekerja melibatkan arsitektur klien-server, dimana *web browser* sebagai klien mengirimkan request kepada *server*, lalu kemudian server mengirimkan data berupa HTML (*HyperText Markup Language*). CSS (*Cascading Style Sheets*), dan skrip *Javascript*. Data yang dikirimkan *server* diubah sebuah *web browser* menjadi sebuah objek berupa *User Interface* (UI) yang dapat dilihat pengguna [10].

Pengembang aplikasi web saat ini sudah dikategorikan menjadi 2, yaitu *Front-End* dan *Back-End Development*. *Front-End Development* lebih fokus menciptakan desain pengalaman pengguna lebih responsif, menarik, dan interaktif. Sedangkan *Back-End Development* justru lebih banyak berhubungan dengan *server*, aliran data aplikasi, dan logika matematika yang diterapkan untuk aplikasi web. Saat ini banyak kerangka kerja yang sudah berkembang untuk mendukung pengembangan *website* lebih responsif, dan interaktif. Berikut beberapa kerangka kerja yang populer digunakan dalam mengembangkan aplikasi *website*:

1. Kerangka Kerja *Front-End*

   HTML dan CSS sebagai kerangka dasar *website* saat ini sudah mencapai versi 5 yang sudah banyak digunakan oleh pengembang. Selain itu beberapa kerangka kerja yang mendukung perkembangan tersebut seperti: React.js, Vue.js, dan Angular.js milik javascript serta kerangka kerja Laravel milik PHP.

2. Kerangka Kerja *Back-End*

   Banyak bahasa pemrograman saat ini mendukung kinerja *back-end development*, seperti: Javascript, PHP (*Hypertext Preprocessor*), Python, TypeScript, dan Go atau Golang. Selain bahasa pemrograman tersebut, banyak juga berkembang kerangka kerja yang mendukung seperti: Laravel milik PHP, Nodejs milik javascript, Django milik python, dan kerangka kerja lain yang dapat dikombinasikan sesuai kebutuhan.

2.6 Sumber Daya Manusia Perusahaan

2.6.1 *Recruitment*

Recruitment adalah sebuah proses untuk menemukan kandidat karyawan yang memenuhi kualifikasi dari sebuah pekerjaan yang terbuka dalam organisasi [11]. Fase akhir dari proses rekrutmen adalah seleksi untuk menentukan apakah kemampuan yang dituliskan calon karyawan sesuai dengan kualifikasi yang dibutuhkan organisasi. Proses rekrutmen tidak hanya sekedar memposting lowongan pekerjaan dan menerima lamaran, tetapi juga harus dilakukan secara sistematis untuk mendapatkan orang yang tepat ditempat yang tepat. Rekrutmen adalah sebuah sub sistem yang terhubung dengan pasar tenaga kerja eksternal dan kebutuhan internal organisasi [12]. Rekrutmen merupakan hal penting yang dapat menjaga keseimbangan, keselarasan, dan keanekaragaman sistem, oleh karena itu perencanaan SDM calon karyawan harus anggap sebagai hal penting yang tidak boleh dianggap remeh.

Proses rekrutmen dapat dikelompokkan menjadi dua proses yang berbeda, yaitu proses rekrut interntal dan eksternal. Proses rekrut internal terjadi ketika seorang karyawan potensial memiliki kesempatan untuk naik jabatan mengisi pekerjaan yang terbuka. Sedangkan proses rekrut eksternal terjadi ketika organisasi mencari kandidat karyawan untuk menempati sebuah posisi terbuka didalam organisasi. Selain rekrutmen, sebuah organisasi dapat melakukan berbagai cara untuk mendapatkan kandidat yang sesuai kualifikasi, seperti kegiatan: promosi *internal*, *transfer* antar divisi, atau bahkan menggunakan jasa *oursourcing* tenaga kerja untuk mengurangi biaya atau untuk mengisi posisi terbuka yang bersifat sementara.

Dalam menjalankan sebuah proses rekrutmen, terdapat beberapa tahapan yang harus dilakukan organisasi dalam mendapatkan kandidat terbaik, berikut langkah – langkahnya [11]:

1. *Preparation*

   Tahapan persiapan merupakan langkah awal yang mana departemen SDM akan menentukan kerangka kerja dalam mendapatkan karyawan dengan kompetensi yang sesuai dengan kualifikasi pekerjaan. Selain itu departemen SDM juga menentukan biaya, dan saluran komunikasi untuk menyebarkan informasi lowongan kerja. Ditahap ini departemen SDM juga menetapkan jumlah biaya yang akan dihabiskan untuk mendapatkan kandidat yang sesuai dengan kualifikasi pekerjaan.

2. *Receive Applicants*

   Departemen SDM yang telah membuka lowongan pekerjaan akan menerima surat lamaran dari calon karyawan. Setiap calon karyawan yang memiliki kompetensi yang sesuai dengan kualifikasi pekerjaan akan mendapatkan undangan untuk melakukan seleksi dan tahapan yang lain sesuai dengan peraturan organisasi.

3. *Selection Stage 1*

   Departemen SDM melakukan seleksi untuk memisahkan calon karyawan yang sesuai dengan kualifikasi dengan calon karyawan yang jelas tidak sesuai dengan kualifikasi berdasarkan surat lamaran yang sebelumnya sudah diterima departemen SDM. Setiap calon karyawan potensial akan mendapatkan undangan untuk melanjutkan proses berikutnya sesuai dengan kebutuhan organisasi.

4. *Selection Stage 2: Interview Rounds*

   Calon karyawan yang memiliki potensi akan diinterview oleh perwakilan departemen SDM untuk mengetahui sejauh mana kompetensi yang dimiliki calon karyawan tersebut. Jika sesuai dengan kualifikasi perusahaan, maka calon karyawan akan mendapatkan undangan untuk melanjutkan prses berikutnya sesuai aturan organisasi.

5. *Offer and Joining Formalities*

   Pada banyak organisasi biasanya pekerjaan akan diberikan kepada kurang dari lima kandidat terpilih yang paling berkualitas. Untuk menyelesaikan proses rekrutmen, calon karyawan akan mendapatkan dokumen yang diperlukan dan melakukan tanda tangan kontrak kerja dengan organisasi. Setelah itu barulah calon karyawan ditentukan kapan akan mulai bekerja.

Proses perbandingan kompetensi calon karyawan dan kualifikasi pekerjaan organisasi dikenal sebagai *profile matchup*. Akan tetapi mekanisme yang sama dimana perusahaan akan menyesuaikan profil calon karyawan dengan kebutuhan pekerjaan lewat *role profile* dan *person spesification* [11]. Penyesuaian kompetensi dan kualifikasi pekerjaan membantu departemen SDM mendapatkan karyawan potensial.

2.6.2 Pencatatan Jam Kerja

Pencatatan jam kerja karyawan atau biasa disebut sebagai presensi merupakan salah satu aspek penting dalam manajemen sumber daya manusia (SDM) yang memiliki peran untuk mengawasi, mengatur, dan mengevaluasi kinerja serta produktivitas karyawan [13]. Proses ini mencatatkan waktu kedatangan, jam istirahat, hingga waktu kepulangan karyawan setiap harinya. Tujuannya adalah untuk mengetahui bahwa setiap aktivitas kerja yang dilakukan sesuai jadwal yang telah ditetapkan perusahaan. Pencatatan jam kerja ini juga memiliki fungsi untuk menghitung upah, lembur, serta tunjangan lainnya. Dengan memiliki sistem pencatatan jam kerja yang baik, maka dapat dipastikan perusahaan dapat menghindari kesalahan administrasi dan juga meningkatkan transparansi antara perusahaan dan karyawan [11].

Pada proses pencatatan jam kerja saat ini dapat dilakukan menggunakan metode digital yang mendukung kemudahan dan efisiensi dalam pelaksanaannya. Metode digital yang berkembang saat ini justru juga meningkatkan transparansi dan meminimalisir kesalahan memasukkan data. Oleh karena itu saat ini banyak perusahaan yang sudah mencoba beralih menggunakan metode digital dalam pencatatan jam kerjanya [11].

Selain menjadi alat kontrol, pencatatan jam kerja karyawan juga memiliki peran penting dalam pengelolaan strategis karyawan. Data jam kerja yang disimpan dengan baik dapat menjadi alat bagi manajer untuk melakukan analisis terkait produktivitas, disiplin, hingga tren yang mempengaruhi kinerja karyawan [12]. Berdasarkan data tersebut manajer dapat mengambil keputusan strategis untuk memberikan insentif tambahan, atau justru menambah upah yang akan diterima karyawan.

2.6.3 *Compensation and Reward Management*

Pencatatan jam kerja merupakan salah satu fungsi operasional dan administrasi yang paling penting dalam sumber daya manusia perusahaan yang tujuannya mencatat waktu masuk, waktu keluar, cuti, izin, dan lembur karyawan agar perusahaan dapat menghitung upah dengan benar. Pada dasarnya pencatatan jam kerja dan kompensasi karyawan merupakan dua hal yang tidak bisa dilepaskan, karena secara konsep untuk menghitung upah seorang karyawan didapatkan dari pencatatan jam kerja [12].

Kompensasi merupakan bentuk imbalan yang diterima karyawan atas kinerja sebagai kontribusi yang sudah dilakukan dalam kurun waktu tertentu pada organisasi. Secara umum, kompensasi tidak hanya sekedar gaji pokok, tetapi juga bonus, insentif, tunjangan, serta penghargaan non finansial seperti kesempatan berkembang, dan lingkungan kerja yang mendukung. Kompensasi memiliki peran strategis dalam memotivasi karyawan untuk tetap menjaga kualitas kinerja tetap tinggi [11]. Mengelola kompensasi dengan baik dapat mendukung perusahaan dalam menjaga produktivitas untuk tetap meningkat, serta menghindari terjadinya *turnover absensi* yang meningkat. Pemberian kompensasi yang baik membuat karyawan merasa dihargai atas kerja keras yang sudah dilakukannya, sehingga justru menimbulkan jiwa loyalitas karyawan terhadap organisasi. Loyalitas karyawan menjadi hal penting bagi organisasi untuk bisa mencapai tujuan tujuan organisasi dengan maksimal dan dengan jangka waktu yang telah ditentukan.

Reward atau penghargaan tidak hanya sebatas tentang uang dan jabatan yang akan diterima karyawan. Penghargaan dapat berupa apresiasi, pengembangan karir, sertifikat penghargaan, atau bahkan memberikan tugas istiimewa yang justru akan membuat seorang

karyawan merasa sangat dihargai. Perlu diingat bahwa sistem penghargaan yang diterapkan organisasi tidak hanya meningkatkan kepuasan karyawan bekerja, tetapi juga memperkuat budaya organisasi berbasis kinerja, artinya setiap karyawan tidak akan merasa keberatan jika mengerjakan tugas mereka [11].

Dalam kompensasi, terdapat prinsip yang harus dijadikan landasan untuk menetapkan kompensasi setiap karyawan, agar menghindari terjadinya kecurangan atau rasa tidak adil bagi pribadi karyawan. Prinsip dan mekanisme dalam menyusun sistem kompensasi harus berasaskan keadilan, dimana kompensasi yang diterima karyawan dihitung berdasarkan kontribusi dan tanggung jawabnya. Performa juga dapat dijadikan salah satu aspek untuk memberikan kompensasi agar setiap karyawan tidak menurunkan kualitas kerjanya terhadap organisasi. Oleh karena itu transparansi menjadi hal penting yang akan membuat karyawan merasa tidak dicurangi, sehingga kompensasi tidak hanya dianggap sebagai apresiasi terhadap karyawan terpilih saja. Jika transparansi sudah diterapkan, maka organisasi juga harus secara konsisten menentukan kompensasi karyawan secara seragam tanpa diskriminasi.