

## **BAB II**

### **TINJAUAN PUSTAKA DAN LANDASAN TEORI**

#### **2.1 Tinjauan Pustaka**

Penelitian tentang sistem informasi sebelumnya telah dilakukan dengan judul “Rancang Bangun Sistem Informasi Ekstrakurikuler”. Sistem ini berfungsi untuk pendaftaran ekstrakurikuler secara *online*, serta memperlancar pengelolaan data anggota ekstrakurikuler. Metode yang digunakan dalam pembuatan sistem ini dengan *Unified Approach* yang terdiri dari tahapan, *Object Oriented Analysis*, *Object Oriented Design*, *Object Oriented Programming* dan pemodelan data menggunakan *Unified Modelling Language*[5].

Penelitian lainnya dilakukan dengan judul “Rancang Bangun Sistem Informasi Penerimaan Siswa Baru”. Fungsi sistem ini untuk meningkatkan kinerja sistem penerimaan siswa baru. Sistem berbasis *website*, dibangun menggunakan metode *Rapid Application Development* (RAD)[6].

Penelitian lain juga melakukan penelitian dengan judul “Rancang Bangun Sistem Informasi Unit Kegiatan Mahasiswa Seni dan Budaya Sekolah Tinggi Teknologi Garut”. Sistemnya digunakan untuk proses pendaftaran menjadi lebih praktis, serta memperlancar pengelolaan data anggota organisasi. Metode yang digunakan dalam pembuatan adalah *Rational Unified Process* yang menggunakan *Unified Modeling Language* sebagai bahasa pemodelan selama periode pengembangan dan menggunakan metode *Black-box Testing* sebagai pengujian terhadap aplikasi[7].

Berdasarkan penelitian-penelitian terkait dengan sistem informasi, maka perbedaan penelitian ini dengan penelitian sebelumnya yaitu adanya fitur *Email gateway* yang digunakan untuk mengirimkan pesan pemberitahuan penerimaan pendaftaran OSIM kepada siswa, kemudian memudahkan ketua OSIM dan ketua ekstrakurikuler dalam pengajuan proposal. Sistem dibuat menggunakan *framework Laravel* dan dalam perancangannya menggunakan Metode *waterfall* dengan

metode pengujiannya menggunakan *Blackbox Testing* untuk memudahkan dalam menyelesaikan penelitian yang dibuat.

## **2.2 Landasan Teori**

### **2.2.1 Sistem Informasi**

Sistem merupakan kumpulan dari perangkat keras dan perangkat lunak yang dibuat oleh manusia guna untuk mengolah data informasi yang bertujuan untuk pencapaian dari sebuah organisasi tersebut[1]. Sistem memiliki beberapa karakteristik atau sifat yang terdiri dari komponen sistem, batasan sistem, lingkungan luar sistem, penghubung sistem, masukan sistem, keluaran sistem, pengolahan sistem dan sasaran sistem. Sedangkan, informasi adalah data yang diolah menjadi lebih berguna dan berarti bagi penerimanya, serta untuk mengurangi ketidakpastian dalam proses pengambilan keputusan mengenai suatu keadaan[8].

Jadi, sistem informasi merupakan suatu kombinasi teratur dari orang-orang, *hardware*, *software*, jaringan komunikasi dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi. Sistem informasi dapat bersifat formal maupun informal. Sistem informasi akuntansi, produksi dan penjualan merupakan contoh sistem informasi formal yang memang secara resmi memiliki tanggung jawab untuk menghasilkan informasi yang akurat. Sedangkan sistem informasi informal berasal dari bagian-bagian organisasi yang tidak secara resmi memberikan informasi, seperti misalnya bagian *legal*[8].

### **2.2.2 Organisasi Intra Sekolah (OSIS) dan Organisasi Siswa Intra Madrasah (OSIM)**

Organisasi Siswa Intra Sekolah (OSIS) merupakan organisasi yang sah di lingkungan sekolah, wajib dibentuk setiap sekolah sebagai perpanjangan dari tangan pemerintah dalam pembinaan generasi muda. Tujuan didirikan OSIS ialah diharapkan para siswa mampu meneruskan perjuangan bangsa dan pembangunan nasional dengan memberi bekal keterampilan, kepemimpinan, keseragaman jasmani, kreativitas, patriotisme, kepedulian soisial dan kepribadian serta budi luhur[9].

Organisasi Siswa Intra Madrasah (OSIM) adalah organisasi siswa yang berfungsi sebagai wadah untuk menampung semua aspirasi dan kegiatan-kegiatan siswa diluar kurikulum atau ekstrakurikuler. Di samping itu juga OSIM berfungsi sebagai salah satu bentuk pembinaan terhadap siswa, sehingga dia menjadi warga negara yang lebih baik. Karena dengan adanya OSIM tersebut siswa akan mendapat pengalaman-pengalaman yang sangat berguna bagi pribadi siswa, misalkan pengalaman menjadi pemimpin, pengalaman bekerja sama, pengalaman hidup demokratis, pengalaman berjiwa toleransi dan pengalaman mengendalikan organisasi[4].

Sebagai suatu organisasi yang ada di sekolah, OSIS dan OSIM ini mempunyai struktur kepengurusan yang sama yang dimana kepengurusan tersebut terdiri dari [9] :

a. Pembina

Pembina memiliki tugas sebagai berikut :

1. Bertanggung jawab atas seluruh pengolahan, pembinaan dan pengembangan OSIS atau OSIM di sekolah.
2. Memberi nasehat dan saran pada perwakilan kelas dan pengurus.
3. Mengesahkan dan melantik anggota.
4. Mengarahkan penyusunan Anggaran Rumah Tangga dan program kerja.
5. Menghadiri, mengawasi serta mengevaluasi kinerja OSIS atau OSIM.

b. Ketua

Ketua memiliki tugas sebagai berikut :

1. Memimpin organisasi dengan baik.
2. Menegoordinasi seluruh anggota.
3. Memimpin rapat.
4. Mengevaluasi kinerja seluruh anggota.

c. Wakil Ketua

Wakil Ketua memiliki tugas sebagai berikut :

1. Bekerjasama dengan ketua pada setiap kegiatan.

2. Memberi saran dan nasehat kepada ketua ketika akan mengambil keputusan.
  3. Menggantikan ketua saat ketua berhalangan.
  4. Bertanggung jawab kepada ketua.
- d. Sekretaris
- Sekretaris memiliki tugas sebagai berikut :
1. Sebagai notulen pada setiap kegiatan.
  2. Mengevaluasi kegiatan dan menyiapkan agenda.
- e. Bendahara
- Bendahara memiliki tugas sebagai berikut :
1. Bertanggung jawab mengetahui pemasukan dan pengeluaran uang.
  2. Mengatur inventaris dan perbendaharaan.
  3. Membuat kuitansi untuk tiap transaksi.
- f. Anggota
- Anggota memiliki tugas sebagai berikut :
1. Menyusun dan melaksanakan program kerja.
  2. Menyampaikan laporan pertanggungjawaban kepada ketua.

### 2.2.3 *Short Message Service (SMS)*

SMS merupakan fitur *Global System for Mobile (GSM)* yang paling populer hingga saat ini. Dimulai dengan diperkenalkannya sistem telepon *wireless/seluler digital* memberikan beberapa kelebihan, seperti kemampuan optimasi sistem yang ditunjukkan dengan kemampuan kompresi dan pengkodean data *digital*. *Handset* yang diperlukan untuk sistem ini juga menjadi sangat simpel, kecil, dan ringan, karena digunakannya *chip digital* untuk *Subscriber Identification module (SIM)*. Teknologi *chip digital* juga memungkinkan penambahan fitur-fitur baru sebagai layanan tambahan, seperti *voice mail*, *call waiting*, dan *Short Message Service (SMS)*. SMS dimaksudkan untuk menjadi alat pertukaran informasi antara dua *mobile subscriber*. Elemen-elemen utama pada arsitektur SMS terdiri dari *Short Message Entity (SME)*, *SMS Service Centre (SMSC)* dan *Email Gateway* yang terkoneksi dengan elemen-elemen pada GSM sebagai *channel* penghantar [10].

**a. *Short Message Entity (SME)***

Short Message Entity (SME) adalah elemen yang dapat mengirim atau menerima pesan singkat. SME dapat berupa *software* aplikasi pada mobile handset, dapat juga berupa perangkat facsimile, perangkat telex, remote internet *server*, dll. Sebuah SME dapat berupa *server* yang terkoneksi dengan SMS center secara langsung atau melalui *gateway*. Dikenal juga *External SME (ESME)* yang merepresentasikan sebuah WAP *proxy/server, Email Gateway* atau *Voice Mail server*[10].

**b. *SMS Service Centre (SMSC)***

SMS Service Centre (SMSC) memegang peran kunci dalam arsitektur SMS. Fungsi utama SMSC adalah menyampaikan pesan singkat antara SME dengan MS, juga menyimpan dan meneruskan pesan singkat (menyimpan pesan jika penerima SME tidak tersedia). SMSC dapat terintegrasi sebagai bagian dari *mobile network* (contoh: terintegrasi dengan MSC) atau sebagai entitas *network independen*[10].

**c. *Email Gateway***

*Email Gateway* memungkinkan sebuah *email* beroperasi menjadi SMS dengan interkoneksi SMSC pada internet. Dengan *email gateway*, pesan dapat dikirim dari sebuah SME menuju sebuah host internet dan sebaliknya. Peran *email gateway* adalah mengubah format pesan (dari SMS ke email dan sebaliknya) dan *merelay* pesan antara SMS dan domain internet[10]

**2.2.4 *Rekayasa Perangkat Lunak***

Rekayasa perangkat lunak merupakan program yang berisi kumpulan intruksi untuk melakukan proses pengolahan data. Rekayasa perangkat lunak sebagai penghubung antara manusia sebagai pengguna dengan perangkat keras komputer, berfungsi menerjemahkan bahasa mesin sehingga perangkat keras komputer memahami keinginan pengguna dan menjalankan intruksi yang diberikan dan selanjutnya memberikan hasil yang diinginkan oleh manusia[11].

## A. Metode Pengembangan Sistem

Metode Pengembangan Sistem adalah suatu proses pengembangan sistem yang mendefinisikan serangkaian aktivitas atau metode dalam rangka mengembangkan dan merawat sistem secara keseluruhan. Terdapat beberapa Metode Pengembangan Sistem yaitu, metode Pengembangan Sistem yang diantaranya adalah Metode *System Development Life Cycle* (SLDC), metode *Waterfall*, metode *Prototyping*, metode *Rapid Application Development* (RAD), metode *Spiral*, metode *Object Oriented Technology* dan Metode *End-user Development*[12].

Dari beberapa metode pengembangan tersebut, penelitian yang dilakukan menggunakan metode *waterfall*. Metode *waterfall* merupakan salah satu model untuk perencanaan Perangkat Lunak. Metode *waterfall* adalah salah satu model klasik yang bersifat sistematis. Mengapa disebut sistematis? Karena model ini dikerjakan secara berurutan dalam penerapannya.

Berikut langkah-langkah pada metode *waterfall* [12] :

### 1. Analisis kebutuhan

Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau studi literatur. Sistem analis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen ini lah yang akan menjadi acuan sistem analis untuk menerjemahkan ke dalam bahasa pemrogram.

### 2. Desain Sistem

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka dan prosedur pengodean. Tahapan dimana dilakukan penuangan pikiran dan

perancangan sistem terhadap solusi dari permasalahan yang ada dengan menggunakan perangkat pemodelan sistem seperti diagram alir data (*data flow diagram*), diagram hubungan entitas (*entity relationship diagram*) serta struktur dan bahasan data.

### 3. Penulisan Kode Program

Penulisan kode program atau *coding* merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan ini lah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah untuk menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

### 4. Pengujian Program

Tahapan dimana sistem yang baru diuji kemampuan dan keefektifannya sehingga didapatkan kekurangan dan kelemahan sistem yang kemudian dilakukan pengkajian ulang dan perbaikan terhadap aplikasi menjadi lebih baik dan sempurna. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

### 5. Penerapan dan Pemeliharaan Program

Tahapan perangkat lunak yang dimana sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (*peripheral* atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional.

## **B. Metode Pengujian Sistem**

Metode pengujian sistem merupakan suatu pengujian perangkat lunak yang terintegrasi. Biasanya perangkat lunak dihubungkan dengan

perangkat lainnya. Pengujian perangkat lunak dapat dibedakan menjadi dua yaitu pengujian *Black-box* dan pengujian *White-box*.

Pengujian *black-box* merupakan teknik pengujian perangkat lunak yang berfokus pada spesifikasi fungsional dari perangkat lunak. Pengujian *black-box* bekerja dengan mengabaikan struktur kontrol sehingga perhatiannya difokuskan pada informasi domain. Pengujian *black-box* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program [13].

Teknik pengujian ini dapat menjadi tes fungsional dan non-fungsional, meskipun biasanya fungsional. Perancang memilih input yang valid dan tidak valid dan menentukan *output* yang benar. Metode uji dapat diterapkan pada semua tingkat pengujian. Metode uji coba *black-box* memfokuskan pada keperluan fungsional karena memungkinkan pengembangan suatu *software* untuk membuat kondisi yang akan melatih syarat fungsional [14].

Kategori untuk menemukan kesalahan dalam uji coba *black-box* [14] :

1. Kesalahan suatu antarmuka atau *interface*.
2. Fungsi yang salah atau hilang.
3. Kesalahan dalam struktur data atau database eksternal.
4. Kesalahan kinerja.
5. Kesalahan inisialisasi dan terminasi.

Ciri uji coba *black-box* [14] :

1. Uji coba *black-box* berfokus pada kebutuhan fungsional.
2. Uji coba *black-box* bukan alternatif dari uji coba *white-box*. ini merupakan pendekatan pelengkap yang mencakup *error* dengan kelas berbeda dari metode uji coba *white-box*.
3. Uji coba *black-box* melakukan pengujian tanpa pengetahuan yang detail.

Kelebihan uji coba *black-box* [14] :

1. Uji coba *black-box* dapat menguji keseluruhan fungsionalitas suatu perangkat lunak.



2. Uji coba *black-box* dapat memilih subset test secara efektif dan efisien.

Kekurangan uji coba *black-box* [14] :

1. Penguji tidak yakin perangkat lunak diuji dengan benar.
2. Penguji tidak yakin bahwa pengujian telah lolos pengujian.

### 2.2.5 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau dikenal juga sebagai *Object Oriented Programming* (OOP), merupakan bagian dari rangkaian aktivitas metode berorientasi objek untuk membangun suatu sistem. Sistem berorientasi objek merupakan sebuah sistem yang komponennya dibungkus (*dienkapsulasi*) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat, dan komponen lainnya serta dapat berinteraksi satu sama lain [15].

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat suatu permasalahan dan sistem baik sistem perangkat lunak, sistem informasi atau sistem lainnya. Ada banyak cara untuk melakukan pengabstraksian dan memodelkan objek, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Sistem berorientasi objek merupakan sebuah sistem yang komponennya dibungkus atau dienskapsulasi menjadi kelompok data dan fungsi. Setiap komponen ini nantinya dalam suatu sistem akan mewarisi atribut dan sifat dari komponen lain dan dapat berinteraksi satu sama lain [15].

Berikut adalah konsep dasar yang harus dipahami yaitu [15] :

1. Kelas (*class*)

Kelas adalah kumpulan objek dengan karakteristik yang sama. Sebuah kelas akan memiliki sifat atau atribut, metode atau operasi, hubungan dan arti. Suatu kelas dapat diturunkan dalam kelas lain, dimana atribut dan kelas dapat diwariskan pada kelas yang baru. Secara teknis, kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia atau hal lainnya. Objek merupakan suatu

entitas yang mampu menyimpan informasi dan operasi yang dapat berpengaruh pada status objeknya. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat suatu objek.

3. Metode (*method*)

Metode atau operasi pada suatu kelas hampir sama dengan fungsi pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode. Metode atau operasi dapat berasal dari *event*, aktivitas atau aksi keadaan dan fungsi.

4. Atribut (*attribute*)

Atribut adalah suatu variabel yang dimiliki oleh sebuah kelas. Atribut dapat berupa nilai atau elemen data yang dimiliki objek dalam suatu kelas. Atribut ini dimiliki secara individual oleh sebuah objek dan sebaiknya bersifat privat untuk menjaga konsep enkapsulasi.

5. Abstraksi (*abstraction*)

Prinsip untuk merepresentasi dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek lainnya yang tidak sesuai dengan masalah.

6. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan untuk menyembunyikan implementasi sehingga objek tidak mengetahui cara kerjanya.

7. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan objek mewarisi sebagian atau seluruh definisi dan objek lainnya.

8. Antarmuka (*interface*)

Antarmuka sangat mirip dengan kelas, tetapi tanpa atribut kelas memiliki metode yang dideklarasikan tanpa ada isi. Deklarasi metode sebuah antarmuka dapat diimplementasi oleh kelas lainnya. Sebuah kelas dapat mengimplementasikan lebih dari satu antarmuka dimana kelas akan mendeklarasikan metode antarmuka yang dibutuhkan oleh kelas itu. Antarmuka atau *interface* biasanya digunakan agar kelas lain tidak mengakses langsung ke suatu kelas.

9. *Reusability*

Pemanfaatan kembali suatu objek sudah didefinisikan untuk suatu masalah yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan objek dan kelas.

11. Komunikasi Antar Objek

Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirim dari suatu objek ke objek yang lainnya.

12. Polimorfisme (*polymorphism*)

Kemampuan objek yang digunakan untuk banyak tujuan yang berbeda dengan nama yang sama sehingga dapat menghemat baris program.

13. *Package*

*Package* adalah sebuah kemasan yang digunakan untuk mengelompokkan kelas sehingga memungkinkan kelas yang bernama sama disimpan dalam *package* yang berbeda.

**A. *Unified Modelling Language (UML)***

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk membangun sebuah perangkat lunak, yaitu *Unified Modelling Language (UML)*. *Unified Modeling Language (UML)* merupakan suatu metode permodelan secara visual untuk perancangan sistem berorientasi objek, atau definisi UML yaitu sebagai Bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem *software*[16].


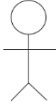
Penggunaan UML bergantung pada level penggunaannya. Jadi belum tentu pandangan yang berbeda itu berarti salah, tetapi perlu dipahami UML itu digunakan dalam hal apa yang divisualkan. Sistem informasi bukanlah ilmu yang pasti, maka banyak perbedaan dan interpretasi dalam bidang sistem informasi merupakan hal yang sangat wajar [15].

### 1. *Use Case Diagram*

*Use case diagram* merupakan interaksi antara *actor* eksternal, relasi dan sistem yang fungsional. Tujuan utama *use case diagram* adalah memutus dan mendeskripsikan kebutuhan fungsional, mendeskripsikan dengan jelas, konsisten dan menyeluruh, menyediakan basis pengujian sistem dan mempunyai kemampuan penelusuran kelas dalam perancangan dan implementasi.

Berikut Tabel 2.1 adalah simbol-simbol yang ada pada *use case diagram* [17] :

**Tabel 2. 1** Simbol *Use Case Diagram*

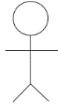



No.	Simbol	Nama	Keterangan
1.		<i>Use case</i>	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau <i>actor</i> ; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i> .
2.		<i>Actor</i> / Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat; biasanya dinyatakan menggunakan kata benda di awal frase nama <i>actor</i> .

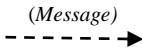



2. <i>Sequence Diagram</i> bersifat dinamis untuk	3.		<i>Association</i> / Asosiasi	Komunikasi antara aktor dan <i>use case</i> yang berpartisipasi atau interaksi antara aktor dan <i>use case</i> .
	4.		<i>Include</i>	Relasi <i>use case</i> tambahan ke <i>use case</i> utama, dimana <i>use case</i> utama memerlukan <i>use case</i> tambahan untuk menjalankan fungsinya atau sebagai syarat dijalkannya <i>use case</i> utama.
	5.		<i>Extend</i> / Ekstensi	Relasi <i>use case</i> tambahan ke <i>use case</i> utama, dimana <i>use case</i> utama dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu.
	6.		<i>Generalization</i> / Generalisasi	Hubungan umum-khusus antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

mengambarkan interaksi antara objek di dalam dan sekitar sistem berupa pesan terhadap waktu digunakan untuk menggambarkan rangkaian sebuah *event* untuk menghasilkan *output* tertentu. Dalam menggambarkan *sequence diagram* perlu memperhatikan objek-objek yang terlibat di dalam *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Berikut Tabel 2.2 adalah simbol-simbol yang ada pada diagram *sequence diagram* [17] :

**Tabel 2. 2** Simbol *Sequence Diagram*

No.	Simbol	Nama	Keterangan
1.		<i>Actor</i> / Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat; biasanya dinyatakan menggunakan kata benda di awal frase nama <i>actor</i> .
2.		<i>Lifeline</i> / Garis hidup	Menyatakan kehidupan suatu objek.
3.		<i>Activation/</i> Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.
4.	 ( <i>Message</i> )	<i>Message</i> / Pesan tipe <i>send</i>	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.

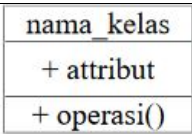
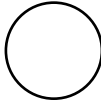


No.	Simbol	Nama	Keterangan
5.		<i>Message /</i> Pesan tipe <i>return</i>	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian tertentu, arah panah mengarah pada objek yang menerima kembalian.
6.		<i>Boundary</i>	Objek yang menggambarkan sebuah <i>form</i> .
7.		<i>Control</i> <i>Class</i>	Objek yang menghubungkan <i>boundary</i> dengan <i>entity</i> .
8.		<i>Entity</i> <i>Class</i>	Objek yang menggambarkan hubungan kegiatan yang akan dilakukan.

### 3. *Class Diagram*

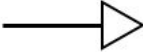
Diagram *Class* (*Class diagram*) adalah sebuah spesifikasi yang apabila diinstantiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Diagram *Class* menggambarkan keadaan, (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi). Hubungan antar class umumnya statis artinya harus mengetahui eksistensi bagian kelas lain. *Class* dapat diturunkan dari *class* lain dan mewarisi *class* asal dan menambahkan fungsionalitas baru.

Berikut Tabel 2.3 adalah simbol-simbol yang ada pada *class diagram* [17] :

**Tabel 2. 3** Simbol *Class Diagram*

No.	Simbol	Nama	Keterangan
1.		Kelas	Kelas pada struktur sistem
2.		<i>Interface</i>	Tampilan antarmuka dalam pemrograman berorientasi objek
3.		<i>Association</i>	Trelasi antar <i>class</i> dengan arti umum, biasanya juga disertai dengan <i>multiplicity</i>
4.		<i>Directed association</i>	Relasi antar <i>class</i> dengan makna <i>class</i> yang digunakan oleh kelas yang lain




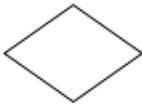









5.		Generalisasi	Relasi antar <i>class</i> dengan makna generalisasi-spesialisasi
----	---	--------------	--

## B. *Flowchart*

*Flowchart* adalah bagan-bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urutan-urutan dari prosedur-prosedur yang ada didalam sistem. Bagan alur sistem menunjukkan apa yang kerjakan sistem Berikut Tabel 2.4 adalah simbol-simbol yang ada pada *flowchart* [18] :

**Tabel 2. 4** Simbol *Flowchart*

No.	Simbol	Nama	Keterangan
1.		<i>Flow</i> / <i>Arus</i>	Menyatakan jalannya arus suatu proses
2.		<i>Connector</i>	Menyatakan sambungan dari suatu proses ke proses lainnya dalam halaman/lembar yang sama
3.		<i>Manual</i>	Menyatakan suatu tindakan (proses) yang tidak dilakukan oleh <i>computer</i>
4.		<i>Decision</i>	Menunjukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, ya/tidak

5.		<i>Process</i>	Menunjukkan pengolahan yang dilakukan oleh computer
6.		<i>Terminal</i>	Menyatakan awal atau akhir suatu program
7.		<i>Manual Input</i>	Untuk memasukkan data secara manual
8.		<i>Document</i>	Untuk mencetak laporan ke <i>printer</i>
9.		<i>Input-Output</i>	Untuk menyatakan proses <i>input</i> dan <i>output</i> tanpa tergantung jenis peralatannya
10.		<i>Disk Storage</i>	Menyatakan input berasal dari <i>disk</i> atau <i>output</i> disimpan ke <i>disk</i>
11.		<i>Database</i>	Menyatakan data input atau <i>output</i> disimpan dalam <i>database</i>

### 2.2.6 Rekayasa Web

Rekayasa *Web* adalah sebuah aplikasi yang menggunakan pendekatan sistematis, disiplin, dan terukur untuk pengembangan, operasi dan pemeliharaan aplikasi berbasis *Web* (*Web-based applications*). Rekayasa *Web* adalah subdisiplin dari rekayasa perangkat lunak yang membantu menyediakan metodologi untuk merancang, mengembangkan, memelihara, dan melibatkan aplikasi *web*. Rekayasa

*Web* membantu para pengembang sistem di bawah *control*, memeperkecil risiko-risiko yang akan terjadi dan meningkatkan kualitas, dapat dipelihara, dan memiliki skalabilitas aplikasi *Web* [19].

### 2.2.7 Basis Data

*Database* atau basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematis sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut *database management system* (DBMS)[20].

Kebutuhan basis data dalam sistem informasi adalah sebagai berikut [15] :

- Memasukkan data, menyimpan data dan mengambil data.
- Membuat laporan berdasarkan data yang disimpan.

Tujuan dari dibuat tabel adalah untuk menyimpan data ke dalam tabel agar mudah diakses. Oleh karena itu, dalam merancang tabel dibutuhkan pola pikir penyimpanan data dimana setiap baris terdiri dari beberapa kolom.

#### A. *Database Management System (DBMS)*

*Database Management System* (DBMS) atau dalam bahasa Indonesia sering disebut sebagai sistem manajemen basis data adalah *software* atau *tools* dari basis data yang dibangun untuk melakukan pengelolaan basis data dengan operasi-operasi yang telah disediakan menggunakan Bahasa tertentu yang telah disepakati (dalam hal ini disebut *SQL*). *SQL* merupakan singkatan dari *Structure Query Language*. Bahasa ini merupakan standar yang digunakan untuk mengakses *database relasional*[21].

#### B. *Data Definition Language (DDL)*

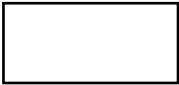


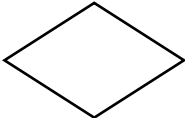
*Data definition Language* (DDL) merupakan suatu metode query *SQL* yang digunakan untuk mendefinisikan data pada suatu *database*.

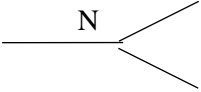
DDL juga dapat digunakan untuk membuat dan memodifikasi suatu obyek *database*[22].

### 2.2.8 Entity Relationship Diagram

*Entity Relationship Diagram* adalah suatu model untuk menggambarkan data dalam bentuk entitas, atribut, dan hubungan antarentitas. Perlu diketahui bahwa model seperti ini tidak mencerminkan bentuk fisik yang nantinya akan disimpan dalam database, melainkan hanya bersifat konseptual. Berikut Tabel 2.5 adalah simbol-simbol yang ada pada *entity relationship diagram* [23]:

**Tabel 2. 5** Simbol *Entity Relationship Diagram*

No.	Simbol	Nama	Keterangan
1.		<i>Entity</i> / Entitas	Merupakan data inti yang akan disimpan atau bakal tabel pada basis data. Penamaan etitas biasanya lebih mengarah pada kata benda dan belum merupakan nama <i>table</i>
2.		Atribut	<i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas
3.		<i>Multivalued</i> / Atribut multinilai	<i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu
4.		Relasi	Relasi yang menghubungkan antar entitas; biasanya diawali kata kerja

5.		<i>Associatio n</i> / Asosiasi	Penghubung antar relasi dan entitas dimana di kedua ujungnya memiliki <i>multiplicity</i> kemungkinan jumlah pemakaian (kardinalitas)
----	---	---------------------------------------	---