BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Penelitian sebelumnya yang dilakukan oleh Firman Hamdani dkk (2017), permasalahan pada penelitian tersebut adalah dengan jumlah pasien yang begitu banyak, tentu data yang disimpan akan semakin bertambah banyak dan sangat rumit apabila data tersebut masih diinput dan dicari secara manual. Kepekaan pasien terhadap rutinitas vaksin yang diberikan untuk hewan peliharaanya masih cukup rendah dan banyak pasien yang terlambat dalam melakukan vaksinasi rutin yang dimana hal ini dapat menyebabkan resiko hewan peliharaannya dapat terserang berbagai penyakit virus dan bakteri. Sistem ini dibuat bertujuan untuk mengelola data dan memudahkan bagi dokter karena terdapat kesulitan dalam mengingat apakah pasien sudah pernah di vaksinasi atau belum di klik hewan PamPam do dan kapan akan divaksin kembali. Metode yang digunakan dalam pengembagan aplikasi adalah *Prototyping* dengan metodologi pengujian *blackbox* testing, tahap-tahapan metodenya yaitu : analisis, desain, implementasi, testing, *maintenance*, keluaran, pengolahan sistem, dan sasaran sistem. Aplikasi pengingat vaksinasi pada toko hewan PamPam Do ini sesuai dengan namanya nantinya mampu mengingatkan vaksinasi kepada pasien dengan adanya sistem reminder vaksin. Dokter dapat mengirimkan pesan kepada pasien yang akan disampaikan melalui email sistem ke email pasien yang telah terdaftar untuk memberitahukan pasien dengan pesan bahwa batas waktu jatuh tempo vaksinasi hewan pasien sebentar lagi akan habis dan dimohon untuk segera kembali dan melakukan vaksinasi kembali. [3].

Penelitian yang lain di kabupaten Kepulauan Sangihe dengan tujuan untuk mempermudah BP4K Kabupaten Kepulauan Sangihe yang dilakukan pada tahun 2019 dalam menyampaikan informasi mengenai pemberian vaksinasi. Permasalahan pada penelitian tersebut adalah jumlah hewan peliharaan di Kabupaten Kepulauan Sangihe meningkat sebanyak lebih dari 1000 ekor dalam kurun waktu 2 tahun. Di Kabupaten Sangihe sendiri BP4K masih menyampaikan informasi pemberian vaksinasi dengan cara manual sehingga kerap terjadi misskomunikasi. Badan Pelaksana Penyuluhan Pertanian Perikanan dan Kehutanan di Kepulauan Sangihe bertugas memberikan pelayanan vaksinasi menggunakan cara surat menyurat ke setiap kecamatan dan kelurahan

dalam menyampaikan informasi mengenai pengadaan vaksinasi yang tidak dapat tersampaikan secara merata kepada masyarakat, serta memakan proses yang cukup panjang dan memakan waktu cukup lama. Dalam membuat Sistem Informasi ini penulis menggunakan *Metode Unified Software Development Process* (USDP). Tahapan yang ada di metode tersebut adalah : identifikasi masalah, pengumpulan data, desain penelitian, pengembangan aplikasi, penyusuan laporan. Sistem Informasi Vaksinasi Hewan Peliharaan dan Ternak di Kab. Kep. Sangihe dianggap perlu, dibuktikan dengan setelah dijalankannya kuesioner kepada pegawaai di BP4K Bagian Peternakan 10/10 (100%) responden menjawab "Ya". Sistem Informasi telah berhasil dibuat sesuai dengan permintaan pengguna dengan menggunakan Metode USDP dan dapat digunakan sebagai media penyampaian informasi mengenai vaksinasi oleh BP4K Bagian peternakan, juga dapat mempermudah proses pencarian data populasi hewan [4].

Penelitian sejenis yang dilakukan pada tahun 2020 oleh Naila Elma Nuarisya melakukan penelitian dengan, permasalahan pada penelitian tersebut adalah pemilik hewan kurang mengetahui kapan saja atau terkadang lupa akan jadwal vaksinasi hewan kesayangannya harus mendapatkan vaksinasi sehingga merugikan kesehatan hewan tersebut. Untuk menyelesaikan permasalahan yang dimiliki, diperlukan adanya sarana untuk mengingatkan pemilik hewan tentang vaksinasi yang harus diberikan salah satunya yaitu aplikasi penggerak. Penelitian ini menggunakan HCD yaitu *framework* dari perancangan suatu sistem yang berfokus pada seseorang yang akan menggunakan sistem yang akan dirancang. Dari aplikasi yang telah dibuat tedapat sekelompok pemangku kepentingan yag dipengaruhi yaitu rumah sakit hewan atau klinik hewan. Sedangkan kelompok pengguna yang akan menggunakan aplikasi yaitu pemilik hewan kucing atau anjing dan dokter hewan. Dari kelompok pemilik hewan, tujuan utama yang ingin dicapai adalah mendapatkan peringatan jadwal vaksinasi hewan yang harus dilakukan. Dari kelompok dokter hewan, tujuan utamanya adalah menerima permintaan vaksinasi hewan dan melalukan pencatatan rekam medis pasien. Berdasarkan aplikasi yang diusulkan, terdapat kebutuhan teknis untuk mendukung jalannya aplikasi yaitu dari segi perangkat keras, perangkat lunak, serta kebutuhan lainnya untuk mengakses internet [5].

Penelitian serupa yang dilakukan oleh Saghifa Fitriana dkk pada tahun 2021. Permasalahan yang ada pada penelitian tersebut adalah

beberapa klinik hewan di Purwokerto belum memanfaatkan kecanggihan teknologi yang ada sehingga proses pengelolaan data masih secara manual untuk setiap proses yang berjalan. Untuk merancang aplikasi ini menggunakan metode *waterfall* dan untuk implementasi menggunakan android supaya dapat digunakan secara *mobile* melalui *smartphone* dan lebih fleksibel dalam penggunaannya. Perancangan Sistem Informasi dirancang sesuai kebutuhan dari salah satu klinik hewan, oleh karena itu dapat dijadikan sebagai dasar untuk membuat aplikasi mobile. Pengguna aplikasi ini akan dipermudah kinerjanya dapat fleksibel menggunakan *smartphone*. Dari sisi pasien akan sangat terbantu sekali dengan adanya aplikasi ini [6].

Penelitian yang sejenis dilakukan oleh Khariri Khariri, Sri Wahyuni pada tahun 2022. Permasalahan yang ada adalah penyakit rabies yang banyak menyerang sistem saraf pusat yang dapat menyebabkan kematian dan penularan menyebar melalui gigitan terutama anjing dan kucing. Penelitian bertujuan untuk memberikan layanan vaksinasi rabies bagi anjing dan kucing sekaligus mengedukasi pemilik hewan. Pendataan dilakukan dengan mengamati dan mendokumentasikan kegiatan untuk analisis data. Dalam kegiatan puncak *World Rabies Day*, Dinas Pangan, Peternakan dan Kesehatan Hewan Provinsi Kalimantan Barat menggandeng Perhimpunan Dokter Hewan Indonesia (PDHI) dan Paramedik Veteriner dan Inseminator Indonesia (*Paravetindo*) untuk memberikan fasilitas layanan vaksin rabies dan sterilisasi secara gratis kepada masyarakat yang memiliki hewan peliharaan anjing dan kucing [7].

Berbeda dengan penelitian sebelumnya, pada penelitian ini penulis bermaksud membangun sistem yang berjudul "Sistem Informasi Vaksinasi Kucing Berbasis *Website*" dengan studi kasus di Klinik Sadewa. Sistem ini dapat diakses oleh admin, kasir, dokter, dan pemilik. Sistem ini bertujuan untuk mempermudah dokter dalam pemberian vaksinasi yang sudah terjadwal, memudahkan pemilik kucing dalam mengetahui penjadwalan vaksinasi, memudahkan pengelola dalam mengetahui jumlah pasien kucing dan jadwal dokter, serta memudahkan pemilik kucing dalam melihat jadwal vaksin yang bisa dilihat melalui hp yang dikirim via *email*.

Tabel 2. 1 Perbandingan Sistem Sebelumnya

No	Judul	Peneliti	Persamaan	Perbedaan
1.	Aplikasi pengingat vaksinasi hewan berbasis desktop pada toko hewan pampam do	Firman Hamdani, Mira Ziveria	Metodologi pengujian yang digunakan yaitu blackbox testing.	**Penelitian sebelumnya :** Metode yang digunakan dalam pengembangan aplikasi yaitu prototype. **Penelitian yang dirancang :** Metode yang digunakan dalam pengembangan aplikasi yaitu metode *waterfall*
2.	Sistem informasi vaksinasi hewan peliharaan dan ternak di kabupaten kepulauan sangihe	Stefanie Mauren Ekaristy Lolaroh, Steven Ray Sentinuwo, Stanley David Sualang Karouw	Tujuan dibangunnya penelitian adalah untuk mempermudah pihak yang bersangkutan dalam penyampaian informasi mengenai jadwal vaksin	**Penelitian sebelumnya :** Metode yang digunakan dalam penelitian yaitu metode *unified software development process* (USDP). **Penelitian yang dirancang :** Metode pengembangan

				sistem yang digunakan *waterfall*, dikarenakan dalam proses perancangan penelitian dikerjakan secara beruntun serta proses pengerjaan diselesaikan tahap demi tahap.
3.	Peracangan antarmuka pengguna aplikasi pengingat jadwal vaksinasi hewan peliharaan menggunakan *human-centred design* (HCD)	Naila Elma Nuarisya	Tujuan penelitian untuk merancang sistem untuk mengingatkan jadwal vaksinasi hewan peliharaan	**Penelitia sebelumnya :** Pendekatan yang digunakan untuk perancangan antarmuka pengguna aplikasi adalah *human centred-design* (HCD). **Penelitian yang dirancang :** Metode pengembangan sistem yang digunakan adalah waterfall.

4.	Perancangan sistem informasi klinik hewan berbasis android	Saghifa Fitriana, Yustina Meisella Kristania	Metode pengembangan dalam perancangan sistem yang digunakan yaitu waterfall	**Penelitian sebelumnya :** - Berbasis android - Jumlah aktor yang terlibat yaitu pasien, petugas dan dokter **Penelitian yang dirancang :** - Berbasis website - Jumlah aktor yang terlibat yaitu dokter, kasir, admin, pemilik
5.	*World rabies day* 2020 : kolaborasi berkualitas dan vaksinasi tuntas untuk Kalimantan barat bebas rabies	Khariri Khariri, Sri Wahyuni	Metode pengumpulan data yang dilakukan adalah observasi dan pengamatan langsung ke studi kasus	**Penelitian sebelumnya :** Metode yang digunakan dalam pelaksanaan kegiatan adalah sosialisasi atau edukasi tentang rabies kepada masyarakat pemilik hewan peliharaan yaitu anjing dan kucing.

				Penelitian yang dirancang : Metode pengembangan sistem yang digunakan waterfall, dikarenakan dalam proses perancangan penelitin dikerjakan secara beruntun serta proses pengerjaan diselesaikan tahap demi tahap.

2.2. Landasan Teori

Landasan teori berisi hal-hal atau teori yang berkaitan dengan permasalahan yang ada pada studi kasus sebagai landasan dalam pembuatan laporan ini.

2.2.1. Sistem Informasi

Sistem adalah suatu kumpulan unsur unsur yang bergabung menjadi satu kesatuan dan mempunyai tujuan yang sama. Unsur-unsur dalam sistem tersebut saling berhubungan satu sama lain untuk memudahkan arus informasi agar dicapai suatu tujuan yang sama[8].

Sistem dapat didefinisikan sebagai serangkaian unit yang masing masing unit tersebut saling terhubung untuk mencapai suatu sasaran yang suda ditentukan[2].

Informasi adalah sekumpulan data-data yang awalnya tidak memiliki sebuah nilai atau tidak bermanfaat bagi orang lain, namun setelah diolah menjadi sebuah bentuk kesatuan yang baru dan bentuk

tersebut memiliki nilai yang dapat digunakan dalam suatu proses pengambilan sebuah keputusan[2].

Sistem informasi yaitu suatu sistem yang menyediakan informasi manajemen dalam mengambil keputusan untuk menjalankan operasional perusahaan tersebut[8].

2.2.2. Vaksinasi Kucing

Vaksin adalah produk biologi yang berisi antigen berupa mikroorganisme yang sudah mati atau masih hidup yang dilemahkan, masih utuh atau bagiannya, atau berupa mikroorganisme yang telah diolah menjadi toksoid atau protein rekombinan, yang ditambahkan dengan zat lainnya. Vaksin digunakan dalam proses imunisasi dengan cara bekerja menimbulkan atau meningkatkan kekebalan seseorang secara aktif terhadap suatu penyakit, sehingga jika suatu saat kucing terkena penyakit maka akan terlindungi. Sama dengan manusia, vaksinasi pada kucing bertujuan untuk membantu mempersiapkan sistem kekebalan tubuh agar dapat memberikan perlindungan saat kucing terinfeksi virus atau bakteri tertentu [9].

Vaksinasi kucing atau pemberian vaksin kepada kucing diberikan secara berkala dan terjadwal. Vaksin kucing dikategorikan menjadi 2 jenis, yaitu vaksin dasar dan vaksin tambahan. Berdasarkan dari pengertiannya, vaksin dasar adalah jenis vaksin yang wajib diberikan ke semua kucing, vaksin tambahan adalah antigen yang diberikan apabila kucing peliharaan kita memiliki beberapa faktor tertentu, contoh : faktor usia, lingkungan, dan interaksi dengan kucing lain. Setelah di lakukan vaksinasi, kucing akan mengalami demam, lesu dan kurang nafsu makan.

2.2.3. Rekayasa Perangkat Lunak

Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi. Pengertian Rekayasa Perangkat Lunak adalah suatu disiplin ilmu yang membahas tentang semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu Analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, desain, pengkodean, pengujian, sampai pemeliharaan sistem setelah dikembangkan [10].

2.2.4. Metodologi *Waterfall*

Metode *Waterfall* disebut juga model sekuensial linier atau alur hidup klasik. Model *Waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari :

1. Rekayasa sistem dan Analisis
   Tahap dimana mulai menyusun kebutuhan untuk sistem dan mengalokasikan beberapa subset dari kebutuhan tersebut pada *software* yang harus berinteraksi pada sistem lainnya, seperti *hardware*, *user* dan *database*.
2. Analisis Kebutuhan Perangkat Lunak
   Analisis kebutuhan menggunakan metode pengumpulan data dan informasi yang diperoleh dari observasi, wawancara, dan sebagainya tentang kebutuhan dari sistem.
3. Perancangan (*design*)
   Pada tahap perancangan dilakukan untuk memberikan gambaran mengenai desain yang harus dibuat. Proses desain menterjemahkan kebutuhan menjadi suatu representasi perangkat lunak yang dapat diakses sebelum pengodingan dimulai.
4. Pembuatan *Coding*
   Pada pembuatan *coding* dilakukan dengan menerapkan hasil desain ke dalam bahasa yang dimengerti komputer menggunakan bahasa pemrograman.

5. Pengujian (*Testing*)
   Pengujian yang digunakan adalah *Black Box*, pada proses pengujian dilakukan untuk mengidentifikasi kemungkinan terjadinya *error* atau kesalahan pada sistem dan memastikan bahwa input yang ditentukan memberikan hasil sesuai dengan harapan.
6. Perawatan (*Maintenance*)
   Pada tahap perawatan jika perangkat lunak sudah melalui tahap pengujian dan sudah selesai dikembangkan, maka sistem sudah bisa digunakan. Perangkat lunak mungkin akan mengalami kesalahan yang tidak diharapkan, oleh karena itu dilakukan tahapan perawatan yang bertujuan untuk melakukan perbaikan dan pengembangan pada sistem.

2.2.5. *Blackbox*

Terkait dengan pengujian program, pengujian dilakukan menggunakan pengujian *black-box.* Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Pengujian *black-box* memiliki dua jenis pengujian yaitu pengujian fungsional dan pengujian *non* fungsional. Pengujian *black-box* (fungsionalitas) menguji bug hanya berdasarkan kegagalan fungsi perangkat lunak yang terungkap dalam bentuk output yang salah. *Black Box* Testing cenderung untuk menemukan hal-hal berikut yaitu[11] :

1) Fungsi yang tidak benar atau tidak ada.
2) Kesalahan antarmuka (*interface errors*).
3) Kesalahan pada struktur data dan akses basis data.
4) Kesalahan performansi
5) Kesalahan inisialisasi dan terminasi.

Tidak seperti pengujian *white-box*, yang dilakukan pada saat awal proses pengujian, pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur struktur kontrol, maka perhatian berfokus pada domain informasi.

Pengujian *black-box* ini terdapat beberapa proses. Proses-proses yang ada dalam pengujian ini diantaranya :

a. Menganalisa kebutuhan dan spesifikasi dari perangkat lunak.
b. Pemilihan jenis *input* yang memungkinkan menghasilkan *output* dengan benar serta jenis *input* yang memungkinkan *output* salah pada perangkat lunak yang sedang diuji.

c. Menentukan *output* untuk satu jenis *input.*
d. Pengujian dilakukan dengan *input-input* yang telah benar-benar diseleksi.
e. Melakukan pengujian
f. Perbandingan *output* yang dihasilkan dengan *output* yang diharapkan.
g. Menentukan fungsionalitas yang seharusnya ada pada perangkat lunak yang sedang diuji.

Keuntungan utama pengujian *Black Box* adalah :

1) Sumber daya yang dibutuhkan yang relatif lebih sedikit dibandingkan dengan pengujian *white box*
2) Efektivitas sumber daya dapat dilakukan dengan pengujian secara otomatis maka berkontribusi pada periode pengujian yang lebih singkat.
3) Kemampuan untuk melakukan hampir semua kelompok *test case*, seperti *availability* (*response time*) *reability*, *load durability* dan kelompok pengujian yang terkait dengan *operation*, *revision*, dan *transition factors*.

2.2.6. Struktur Data

Struktur data adalah cara menyimpan atau merepresentasikan data di dalam komputer agar bisa dipakai secara efisien. Berikut adalah bagian-bagian yang ada pada struktur data :

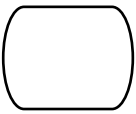
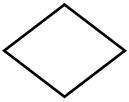
1. Flowchart

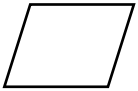
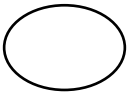
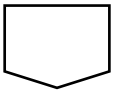
Flowchart adalah sebuah diagram yang menjelaskan alur dari proses sebuah program. Dalam pembuatan sebuah program, *flowchart* berguna untuk menerjemahkan proses berjalannya sebuah program agar mudah dipahami. Setiap langkah digambarkan dengan garis atau arah panah. *Flowchart* terdiri dari 5 jenis, yang masing-masing mempunyai karakteristik dalam penggunaanya, diantara lain yaitu :

1) *Flowchart* dokumen
2) *Flowchart* program
3) *Flowchart* proses
4) *Flowchart* sistem
5) *Flowchart* skematik

Pada dasarnya simbol-simbol dalam *flowchart* mempunyai arti masing-masing. Berikut adalah simbol-simbol yang sering digunakan dalam proses pembuatan *flowchart* seperti yang terdapat pada Tabel 2. 2.

Tabel 2. 2 Simbol *Flowchart*

No	Simbol	Nama Simbol	Fungsi
1.		*Terminator*	Simbol untuk menunjukan awal atau akhir dari aliran proses. Biasanya menggunakan kata-kata '*Start', 'End'*, 'Mulai', 'Selesai'.
2.		*Garis Alir (Flow Line)*	Tanda panah yang menunjukan arah aliran dari suatu proses ke proses yang lain
3.		*Process*	Simbol untuk menunjukan sebuah langkah proses atau operasi. Umumnya, menggunakan kata kerja dalam deskripsi yang singkat dan jelas.
4.		*Decision*	Simbol untuk menunjukan sebuah langkah pengambilan keputusan. Umumnya, menggunakan bentuk pertanyaan dan biasanya jawabannya terdiri dari '*Yes'* dan '*No'* atau 'Ya' dan 'Tidak' yang menentukan alur dalam *flowchart* berjalan selanjutnya berdasarkan

			kriteria atau pertanyaan tersebut.
5.		*Document*	Simbol untuk menunjukan proses atau keberadaan dokumen.
6.		*Input/Output*	Simbol untuk menunjukan data yang menjadi *input* atau *ouput* proses.
7.		*Connector (On-page)*	Simbol untuk menunjukan hubungan simbol dalam *flowchart* sebagai pengganti garis untuk menyederhanakan bentuk saat simbol yang akan dihubungkan jaraknya berjauhan dan rumit jika dihubungkan dengan garis.
8.		*Off-page Connector*	Fungsinya sama dengan *connector*, akan tetapi digunakan untuk menghubungkan simbol simbol yang berada pada halaman yang berbeda. Label untuk *connector* dapat menggunakan huruf dan *Off Page*

			Connector menggunakan angka.

2. *Unified Modeling Language* (UML)

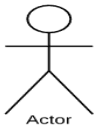
Unified Modeling Language merupakan salah satu metode pemodelan visual yang digunakan dalam perancangan dan pembuatan sebuah *software* yang berorientasikan pada objek. UML merupakan sebuah standar penulisan atau semacam *blue print* dimana didalamnya termasuk sebuah bisnis proses, penulisan kelas-kelas dalam sebuah bahasa yang spesifik[12]. Terdapat beberapa diagram UML yang digunakan dalam pengembangan sebuah sistem, yaitu :

a. *Use Case*

Use case adalah sebuah interaksi yang saling berkaitan antara actor dengan sistem. *Use case* diagram adalah proses penggambaran yang dilakukan untuk menunjukkan hubungan antara pengguna dengan sistem yang sedang dirancang[13].

Terdapat 2 elemen penting dalam *use case* diagram, yaitu Actor dan UC. Actor adalah segala sesuatu yang berinteraksi langsung dengan sistem. Actor dinotasikan dengan simbol gambar orang-orangan *(stick-man)* dengan nama kata benda di bagian bawah yang menyatakan peran atau sistem. *Use case* dinotasikan dengan simbol elips dengan nama kata kerja aktif di bagian dalam yang menyatakan aktivitas dari perspektif actor. Seperti pada tabel 2. 3*Use Case* dibawah ini.

Tabel 2. 3 Simbol *Use Case Diagram*

No	Gambar	Nama Gambar	Keterangan
1.		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*
2.		*Dependency*	Hubungan dimana perubahan yang

			terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri (*dependent*).
3.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk
4.		*Include*	Menspesifikasikan bahwa *use case* sumber secara eksplisit
5.		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.

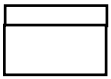
7.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8.		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil terukur bagi suatu aktor.
9.		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10.		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

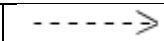
b. *Class Diagram*

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk

memanipulasi keadaan tersebut (metoda/fungsi)[14]. Seperti yang dapat dilihat gambar dan keterangan *class diagram* pada Tabel 2. 4.

Tabel 2. 4 Simbol *Class Diagram*

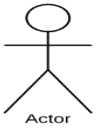
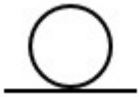
No	Gambar	Nama Gambar	Keterangan
1.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancestor*)
2.		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek
3.		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek

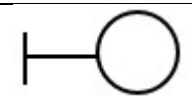
6.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri
7.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

c. *Sequence Diagram*

Sequence Diagram adalah tool yang sering digunakan dalam pengembangan sistem informasi secara *object-oriented* untuk menampilkan interaksi antar objek. *Sequence Diagram* digunakan sebagai perkakas dalam perancangan antarmuka pemakai. *Sequence Diagram* lebih ditujukan untuk memperlihatkan semua bagian/divisi pada sebuah organisasi yang terlibat dalam proses bisnis[15]. Berikut beberapa simbol *Sequence Diagram* seperti pada Tabel 2.5

Tabel 2. 5 Simbol *Sequence Diagram*

No	Gambar	Nama Gambar	Keterangan
1.		*Actor*	Menggambar orang yang sedang berinteraksi dengan sistem.
2.		*Entity Class*	Menggambarkan hubungan yang akan dilakukan.

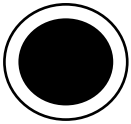
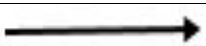
3.		*Boundary Class*	Menggambarkan sebuah gambaran dari *form*
4.		*Control Class*	Menggambarkan penghubung antara *boundary* dengan tabel
5.		*A Focus of Control & A Life Line*	Menggambarkan tempat mulai dan berakhirnya *massage*
6.		*A Massage*	Menggambarkan Pengiriman Pesan

d. *State Diagram*

State diagram digunakan untuk mendokumentasikan beragaam kondisi/keadaan yang bisa terjadi terhadap sebuah *class* dan kegiatan apa saja yang dapat merubah kondisi / keadaan tersebut. Pada umunya statechart diagram menggambarkan *class* tertentu (satu *class* dapat memliki lebih dari satu *statechart diagram*) [16]. Dibawah ini adalah jenis gambar/simbol yang terdapat pada *state diagram* pada Tabel 2. 6.

Tabel 2. 6 Simbol *State Diagram*

No	Gambar	Nama Gambar	Keterangan
1.		*State*	Digambarkan berbentuk segi empat dengan sudut membulat memiliki nama sesuai kondisi saat itu

2.		Titik awal *(start)*	Digunakan untuk menggambarkan awal dari kejadian dalam suatu diagram
3.		Titik Akhir *(end)*	Digunakan untuk menjelaskan/menggambarkan akhir dari kejadian dalam suatu diagram
4.	[*Guard*]	*Guard*	Merupakan syarat transisi yang bersangkutan
5.		*Point*	Digunakan untuk menggambarkan /menjelaskan apakah akan masuk ke dalam status atau keluar
6.	*event*	*Event*	Digunakan untuk menjelaskan kondisi yang menyebabkan sesuatu pada status

2.2.6. Pemrograman Berorientasi Objek (PBO)

Pemrograman Berorientasi Objek (PBO) atau *Object Oriented Programming* (OOP) merupakan suatu pendekatan pemrograman yang menggunakan *object* dan *class*. OOP memberikan kemudahan dalam pembuatan sebuah program, keuntungan yang didapat apabila membuat PBO atau OOP antara lain :

1) *Reusability*, kode yang dibuat dapat digunakan kembali.
2) *Extensibility*, pemrogram dapat membuat metode baru atau mengubah yang sudah ada sesuai yang diinginkan tanpa harus membuat kode dari awal.
3) *Maintainability*, kode yang sudah dibuat lebih mudah untuk dikelola apabila aplikasi yang dibuat berskala besar yang memungkinkan

adanya error dalam pengembangannya hal tersebut dapat diatasi dengan OOP karena pemrograman OOP yang lain yaitu : Alamiah, dapat diandalkan (*reliable*), dapat dipakai kembali (*reusable*), mudah dirawat (*maintainable*), dapat diperluas (*extendable*),efisiensi waktu.

Objek dalam OOP adalah unit terkecil pemrograman yang masih memiliki data (sifat karakteristik) dan fungsi. Objek merupakan entitas dari sebuah keadaan, peilaku dan identitas yang tugasnya dirumuskan dalam suatu lingkup masalah, pendeklarasian objek dari sebuah class disebut dengan instance. *Class* adalah wadah yang berisi pemodelan suatu objek, mendeskripsikan karakteristik dan fungsi objek tersebut. Karena *class* merupakan wadah yang digunakan untuk menciptakan suatu objek, maka *class* harus dibuat terlebih dahulu[14].

2.2.7. Basis Data

Basis data adalah kumpulan data yang saling berhubungan secara logis dan didesain untuk mendapatkan data yang dibutuhkan oleh suatu organisasi. Basis data merupakan komponen penting dalam sistem informasi modern. *Database* merupakan suatu kumpulan data terhubung (*integrated*) yang disimpan dengan cara tertentu sehingga mudah untuk digunakan sehingga proses modifikasi data dapat dilakukan dengan mudah dan terkontrol. *Database system* mempunyai elemen penting yaitu :

1) *Database* sebagai inti dari sistem basis data
2) Program aplikasi untuk manajemen basis data
3) Perangkat keras sebagai penunjang operasi manajemen data
4) *Brainware* yang mempunyai peran penting dalam sistem tersebut

MySQL dikembangkan oleh suatu perusahaan yang terletak di Swedia yang bernama MySQL AB, yang pada saat itu Bernama TcX Data Konsult AB sekitar tahun 1994-1995. My SQL sudah ada sejak 1979 termasuk jenis *Relation Database Management System* (RDBMS) digunakan oleh banyak batas internet sebagai basis data dari informasi yang ditampilkan pada situs web. Terkenalnya MySQL dimungkinkan karena mudahnya untuk digunakan cepat secara kinerja *query*, dan mencukupi kebutuhan basis data terhadap perusahaan skala menengah dan kecil. Sebuah basis data terdapat pada MySQL yang mengandung satu atau beberapa tabel yang berisi sejumlah baris dan kolom.

MySQL merupakan salah satu turunan konsep utama dalam database, yaitu SQL(*Structured Query Language*). SQL adalah bahasa yang

digunakan dalam mengakses data di sebuah database relasional. SQL sering dikenal dengan istilah query, dan bahasa SQL secara prakteknya digunakan sebagai bahasa standar dalam manajemen database relasional. Dalam penggunaan SQL terdapat beberapa perintah yang bertujuan untuk mengakses dan memanajemen data yang terdapat dalam database. Secara garis besar, SQL Server terdapat 3 jenis perintah yaitu :

***1) Data Definition Languange* (DDL)**

DDL merupakan sub perintah dari bahasa SQL yang digunakan untuk membangun rangkaian sebuah database. Terdapat tiga perintah penting dalam DDL, yaitu :

a) *CREATE*, merupakan perintah yang digunakan untuk membuat, seperti membuat database baru, tabel baru, dan kolom baru. Contoh : *CREATE TABEL* nama_tabel.
b) *ALTER*, merupakan perintah yang digunakan untuk mengubah struktur tabel yang telah dibuat. Didalamnya terdapat menambah kolom, menghapus kolom, mengubah nama tabel, dan memberikan atribut pada kolom. Contoh : *ALTER TABEL* nama_tabel *ADD* nama_kolom *datatype*.

***2) Data Manipulation Language* (DML)**

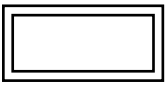
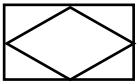
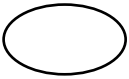
DML adalah sub perintah dari bahasa SQL yang digunakan untuk memanipulasi data dalam database yang telah dibuat. Terdapat 4 (Empat) perintah penting dalam DML, yaitu :

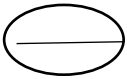
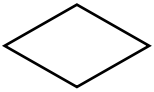
a) *INSERT*: perintah ini digunakan untuk memasukkan data baru ke dalam sebuah tabel. Perintah ini tentu saja bisa dijalankan ketika database dan tabel sudah dibuat. Contoh: *INSERT INTO* nama_tabel *VALUES* (data1, data2, dst…);
b) *SELECT*: perintah ini digunakan untuk mengambil dan menampilkan data dari tabel atau bahkan dari beberapa tabel dengan penggunaan relasi. Contoh: *SELECT* nama_kolom1, nama_kolom2 *FROM* nama_tabel;
c) *UPDATE*: perintah update digunakan untuk memperbaharui data pada sebuah tabel. Contoh: *UPDATE* nama_tabel *SET* kolom1=data1, kolom2=data2,… *WHERE* kolom=data;
d) *DELETE*: perintah delete digunakan untuk menghapus data dari sebuah tabel. Contoh: *DELETE FROM* nama_tabel *WHERE* kolom=data

***3) Entity Relationship Diagram* (ERD)**

ERD adalah suatu model untuk menjelaskan mengenai hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang memiliki hubungan antar relasi. *Entity Relationship Diagram* digunakan untuk memodelkan struktur data serta hubungan antar data, untuk dapat menggambarkannya digunakan beberapa notasi serta simbol. Untuk tabel diagram ERD dapat dilihat pada Tabel 2.7.

Tabel 2. 7 Diagram *Entity Relationship Diagram*

No	Gambar	Nama Gambar	Keterangan
1.		*Entity*	Suatu *entity* digambarkan sebagai sebuah persegi panjang yang memiliki nama entity tersebut.
2.		*Weak Entity*	Suatu *entity* yang tidak dapat diidentifikasikan melalui atributnya dengan sendirinya. Keberadaan *weak entity* bergantung kepada entity lain yang disebut *owner entity*.
3.		*Associative Entity*	Entity yang digunakan pada *many to many relationship*.
4.		*Attribute*	Dalam notasi *Chen*, sebuah atribut digambarkan sebagai sebuah oval yang memuat nama atribut tersebut.

5.		*Key Attribute*	Suatu atribut yang mengidentifikasi suatu *entity* dengan sangat spesifik atau unik. Nama dalam *Key Attribute* selalu di *underscore*
6.		*Multivalued Attribute*	*Attribute* yang dapat memuat lebih dari satu nilai *(Multivalued). Multivalued Attribute* digambarkan dengan dua oval.
7.		*Strong Relationship*	Hubungan dimana sebuah keberadaan entity bergantung dengan entity lain, dan *Primary Key* dari *Child Entity* tidak memuat komponen PK *Parent Entity*.
8.		*Weak Relationship*	Suatu relationship dimana keberadaan *Child entity* bergantung pada induknya, dan PK *Child entity* memuat komponen PK *Parent entity*.

Komponen dalam menyusun ERD antara lain sebagai berikut :

a) Entitas merupakan suatu objek dalam dunia nyata yang bisa dibedakan dengan objek lain, sebagai contoh pengelola, dokter dan

pemilik. Entitas tersebut terdiri dari beberapa atribut sebagai contoh : nama, alamat, umur dan lain sebagainya.

b) Atribut merupakan entitas pasti yang memiliki elemen yang berfungsi untuk dapat mendeskripsikan karakteristik dari suatu entitas tersebut seperti contoh di atas. Isi dari atribut tersebut memiliki sesuatu yang biasa mengidentifikasikan isi elemen yang satu dengan yang lainnya. Terdapat dua jenis Atribut antara lain sebagai berikut :

1. *Identifier* (*key*) yang berfungsi sebagai penentu *entity* secara unik (*primary key*).
2. *Descriptor* (*nonkey attribute*) digunakan untuk dapat menspesifikasikan karakteristik dari sebuah *entity* yang tidak unik.

c) Relasi adalah suatu hubungan antara beberapa entitas himpunan relasi antar entitas pemetaan kardinalitas terdiri dari :

1. One-to-one

Setiap baris data pada tabel pertama dihubungkan hanya ke satu baris data pada tabel ke dua. Hubungan antara *file* pertama dan *file* kedua adalah satu berbanding satu.

2. One-to-many

Setiap baris data dari tabel pertama dapat dihubungkan ke satu baris atau lebih data pada tabel ke dua. Hubungan antara *file* pertama dan *file* kedua adalah satu berbanding banyak atau banyak berbanding satu.

3. Many-to-one

Kebalikan dari *relation One To Many* dimana setiap baris data dari tabel pertama dihubungkan lebih dari satu baris ke tabel kedua. Hubungan antara *file* pertama dan *file* kedua adalah banyak berbanding.

~Halaman ini sengaja dikosongkan~