BAB II
LANDASAN TEORI

1.1 Teori Dasar

2.1.1 Sistem Informasi

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berinteraksi untuk melakukan suatu kegiatan atau mencapai suatu tujuan tertentu. Sistem juga dapat diartikan sebagai serangkaian data atau komponen yang saling berinteraksi dan berhubungan satu sama lain untuk mencapai suatu tujuan. Sistem adalah suatu unit data yang terstruktur dan diatur secara prosedural. (Sallaby & Kanedi, 2020).

Informasi sangat penting dan berguna karena memungkinkan kita untuk memahami kemajuan dan kegagalan proses. Sebuah sistem yang tidak memiliki informasi dapat dianggap lemah dan rentan. (T. Rahman et al., 2020).

Sistem informasi adalah sebuah sistem di dalam sebuah organisasi yang menyediakan laporan-laporan yang diperlukan dan berguna bagi pihak-pihak tertentu. (Asyhari & Arifin, 2021). Menurut Kertahadi (2007), Sistem informasi adalah alat untuk menyajikan informasi dengan cara yang berguna bagi penerimanya (Sutiyono, S.T., M.Kom & Santi, 2020).

2.1.2 Apotek

Dalam Peraturan Menteri Kesehatan Republik Indonesia Nomor 9 Tahun 2017 Tentang Apotek pada Bab I Ketentuan Umum Pasal 1 ayat (1) menjelaskan bahwa Apotek adalah sarana pelayanan kefarmasian tempat dilakukan praktik kefarmasian oleh Apoteker (Menteri Kesehatan Republik Indonesia, 2017).

2.1.3 Pasien

Dalam Peraturan Menteri Kesehatan Republik Indonesia Nomor 24 Tahun 2022 Tentang Rekam Medis pada Bab I Ketentuan Umum Pasal 1 ayat (6) menjelaskan bahwa Pasien adalah setiap orang yang melakukan konsultasi masalah kesehatannya untuk memperoleh pelayanan kesehatan yang diperlukan baik secara langsung maupun tidak langsung di Fasilitasi Pelayanan Kesehatan (Menteri Kesehatan Republik Indonesia, 2022) .

2.1.4 Antrian

Menurut Heizer dan Render (2016:658) Antrian adalah orang atau barang yang berbaris dalam antrean yang menunggu untuk menerima layanan, dan antrean mencakup cara perusahaan mengatur waktu dan memberikan layanan yang baik dan efisien kepada pelanggan.

Sedangkan menurut Handolo & Astuti (2017:106) Antrian adalah proses yang melibatkan pelanggan yang tiba di lokasi tertentu, kemudian menunggu dalam sistem untuk menerima layanan, dan meninggalkan tempat tersebut setelah layanan selesai.

2.1.5 Rekam Medis

Dalam peraturan Menteri kesehatan Republik Indonesia Nomor 24 Tahun 2022 Tentang Rekam Medis pada Bab I Ketentuan Umum Pasal 1 ayat (1) menjelaskan bahwa Rekam Medis adalah dokumen yang berisikan data identitas pasien, pemeriksaan, pengobatan, tindakan, dan pelayanan lain yang telah diberikan kepada pasien (Menteri Kesehatan Republik Indonesia, 2022).

2.1.6 Rekam Medis Elektronik

Dalam peraturan Menteri kesehatan Republik Indonesia Nomor 24 Tahun 2022 Tentang Rekam Medis pada Bab I tentang Rekam Medis Pasal 1 ayat (2) menjelaskan bahwa Rekam Medis Elektronik adalah Rekam Medis yang dibuat dengan menggunakan sistem elektronik yang diperuntukkan bagi penyelenggaraan Rekam Medis. Pasal V menjelaskan: Rekam Medis Elektronik merupakan salah satu subsistem dari sistem informasi Fasilitas Pelayanan Kesehatan yang terhubung dengan subsistem informasi lainnya di Fasilitas Kesehatan (Menteri Kesehatan Republik Indonesia, 2022)

2.1.7 Basis Data

Basis data adalah struktur penyimpanan data yang memungkinkan Anda untuk menambah, mengakses, dan memproses data yang tersimpan dalam basis data komputer. Hal ini memerlukan sistem manajemen basis data seperti *MySQL Server*. (Artanto, 2022).

2.1.8 Normalisasi Data

Normalisasi basis data adalah proses pengorganisasian data dalam basis data relasional untuk menghilangkan redundansi dan mencegah anomali pemrosesan data. Normalisasi basis data bertujuan untuk memperbaiki struktur tabel dan hubungan antar tabel agar lebih efisien dan lebih mudah dikelola. Normalisasi database dilakukan dengan cara memecah sebuah tabel besar menjadi beberapa tabel yang lebih kecil dan saling berhubungan erat sesuai dengan aturan normalisasi.

ujuan dari normalisasi database adalah untuk menciptakan struktur database yang optimal dan efisien. Dalam normalisasi, database diorganisasikan ke dalam tabel-tabel dan setiap tabel memiliki sekumpulan atribut yang terkait dengan entitas atau hubungan tertentu. Tujuan dari normalisasi adalah untuk menghilangkan redundansi data dan meminimalisir inkonsistensi data, sehingga mengurangi kemungkinan terjadinya anomali data seperti update, penyisipan, dan penghapusan. Dalam proses normalisasi, perlu diketahui definisi dari tahapan atau bentuk-bentuk normalisasi, yaitu:

1. Tidak Normal *(unnormalize)*

Bentuk abnormal adalah sekumpulan data yang direkam tanpa mengikuti aturan atau format tertentu. Bentuk tidak normal ini berisi kelompok yang berulang, yang dapat menyebabkan masalah saat memanipulasi data, seperti anomali penyisipan, pembaruan, dan penghapusan. Anomali penyisipan terjadi ketika data ditambahkan, tetapi elemen kosong masih tersisa. Anomali update terjadi ketika perubahan dilakukan pada beberapa data di dalam tabel, tetapi tidak semua data diperbarui.

Sementara itu, delete anomalies terjadi ketika sebuah baris atau record yang tidak terpakai dihapus, yang dapat menyebabkan data lain yang terkait hilang.

Tabel 2. 1 Tabel *Unnormalize*

No. Faktur	Tanggal	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
A-101	08/12/2023	R-112	Jora	J-010	Kabel	102.000,-	2
				J-100	Handuk	50.000,-	4
B-102	13/12/2016	R-113	Ratu	J-090	Tali	5.000,-	5
				J-132	Teh	23.000,-	1

2. Normal pertama

Pada tingkat ini, setiap kolom dalam tabel hanya memiliki nilai atomik dan tidak dapat dipecah lebih lanjut. Setiap baris dalam tabel harus unik dan tidak boleh ada kolom variabel dalam tabel. Sebuah tabel dikatakan dalam bentuk normal pertama (1NF) jika setiap atribut hanya memiliki satu nilai di setiap baris. Oleh karena itu, tabel yang belum dinormalisasi perlu dimodifikasi untuk memenuhi kriteria 1NF agar dapat ditampilkan dalam bentuk berikut:

Tabel 2. 2 Tabel Normal Pertama

No. Faktur	Tanggal	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	M Fikri Setiadi	B-001	Sampo	12.000,-	1
F-001	12/12/2016	P-001	M Fikri Setiadi	B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-002	Kopi	15.000,-	1
F-002	13/12/2016	P-002	Jack	B-003	Teh	7.000	2

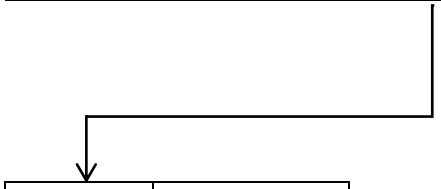
3. Normal kedua

Persyaratan untuk 2NF adalah bahwa “ketergantungan fungsional” parsial tidak diperbolehkan untuk kunci utama dalam tabel. Apa yang dimaksud dengan “ketergantungan fungsional”? Ketergantungan fungsional adalah setiap atribut

bukan kunci yang secara fungsional bergantung pada kunci utama. Intinya adalah bahwa dalam langkah normalisasi 2NF ini, tabel harus dipartisi berdasarkan kunci utama. Dengan demikian, bentuk normalisasi tabel 2NF adalah sebagai berikut:

Tabel 2. 3 Tabel Normal Kedua

No. Faktur	**Tanggal**	**Kode Pelanggan**	**Kode Barang**	**Qty**
F-001	12/12/2016	P-001	B-001	1
F-001	12/12/2016	P-001	B-002	1
F-002	13/12/2016	P-002	B-002	1
F-002	13/12/2016	P-002	B-003	2



Kode Pelanggan	**Nama**
P-001	M Fikri Setiadi
P-002	Jack

Kode Barang	**Nama Barang**	**Harga**
B-001	Sampo	12.000,-
B-002	Kopi	15.000,-
B-003	Teh	7.000,-

4. Normal ketiga

Dalam bentuk normal ketiga (3NF), ketergantungan transitif tidak diperbolehkan dalam tabel. Apa yang dimaksud dengan ketergantungan transitif? Ketergantungan transitif biasanya terjadi pada tabel yang dihasilkan dari hubungan yang memiliki tiga atribut, yaitu A, B, dan C. Dalam hal ini, jika A ⇒ B dan B ⇒ C, maka C disebut ketergantungan transitif dari A melalui B. Pada dasarnya, dalam 3NF, jika ada atribut yang tidak bergantung secara langsung pada primary key tetapi bergantung pada atribut lain, maka atribut tersebut perlu dialokasikan ke tabel baru. Sebagai contoh, atribut qty tidak bergantung secara langsung pada kunci utama "kode_suara" tetapi bergantung pada kolom "kode_barang". Oleh karena itu, setelah normalisasi dalam 3NF, tabel akan terbentuk sebagai berikut:

Tabel 2. 4 Tabel Normal Ketiga

No. Faktur	Tanggal	Kode Pelanggan
F-001	12/12/2016	P-001
F-002	13/12/2016	P-002

Tabel Detail

No.Faktur*	Kode Barang	Qty
F-001	B-001	1
F-001	B-002	1
F-002	B-002	1
F-002	B-003	2

Tabel Barang

Kode Pelanggan*	Nama
P-001	M Fikri Setiadi
P-002	Jack

Tabel Pelanggan

Kode Barang*	Nama Barang	Harga
B-001	Sampo	12.000,-
B-002	Kopi	15.000,-
B-003	Teh	7.000

Ada beberapa jenis kunci yang digunakan dalam proses pencarian, penyaringan, dan penghapusan dalam pengelompokan database, yaitu:

1. Kunci Super (*Super Key*: adalah sekumpulan satu atau beberapa atribut yang dapat digunakan untuk mengidentifikasi sebuah objek secara unik dalam sekumpulan objek.
2. Kunci Primer (*Primary Key*): Ini adalah atribut atau sekumpulan atribut minimal yang tidak hanya mengidentifikasi secara unik peristiwa tertentu, tetapi juga dapat mewakili setiap kejadian suatu objek.

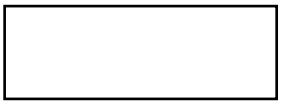
3. Kunci Tamu (*Foreign Key*): Ini adalah atribut atau sekumpulan atribut yang melengkapi hubungan yang menunjukkan hubungannya dengan entitas induk.
4. Kunci Calon (*Candidate Key*): Ini adalah atribut atau sekumpulan atribut minimum yang dapat digunakan untuk mengidentifikasi secara unik peristiwa tertentu dalam suatu entitas.
5. Kunci Alternatif (*Alternate Key*): Ini adalah atribut atau sekumpulan atribut minimum yang dapat digunakan untuk mengidentifikasi secara unik peristiwa tertentu dalam suatu entitas.
6. Kunci Gabungan (*Composite Key*): Jika tidak ada atribut yang dapat digunakan sebagai kunci utama, beberapa atribut dapat digabungkan menjadi satu kunci gabungan. Kunci ini digunakan untuk mengidentifikasi objek dalam tabel secara unik.

2.1.9 *Entity Relationship Diagram* (ERD)

ERD (*Entity-Relationship Diagram*) merupakan pemodelan basis data awal yang dikembangkan dari teori himpunan di bidang matematika untuk merancang basis data relasional. ERD digunakan untuk menggambarkan entitas, atribut dan hubungan antar entitas dalam sebuah sistem, sehingga memudahkan untuk memahami dan mengatur struktur data yang akan digunakan.. (Artanto, 2022).

1. Entitas

   Entitas adalah sekumpulan objek yang dapat diidentifikasi secara unik.



Gambar 2. 1 Simbol Entitas

2. Atribut

Atribut adalah properti yang menggambarkan karakteristik suatu entitas. Pada diagram, atribut diwakili oleh oval kecil dan diberi label dengan nama atribut. Atribut diwakili oleh oval kecil dan diberi label dengan nama. Contoh atribut dari

entitas Mahasiswa adalah “Nama”, “NIM”, “Alamat”, dll. Di bawah ini adalah beberapa jenis atribut yang umum digunakan dalam database:

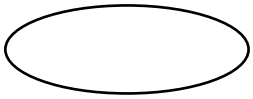
a. Atribut Kunci (*Key Attribute*): Atribut yang digunakan untuk menentukan data yang bersifat unik. Biasanya, data dari atribut kunci berbentuk angka. Contoh: NIM (Nomor Induk Mahasiswa), No. KTP, SIM, NPWP, dan lainnya.

b. Atribut Simpel (*Simple Attribute*): Atribut yang bersifat atomic, tidak dapat dipecah lagi, dan bernilai tunggal. Contoh: alamat rumah, kantor, nama penerbit, tahun terbit jurnal, dll.

c. Atribut Multinilai (*Multivalue Attribute*): Atribut yang memiliki beberapa nilai untuk setiap entitas. Contoh: kumpulan nama pengarang dalam sebuah novel.

d. Atribut Gabungan (*Composite Attribute*): Atribut yang merupakan kombinasi dari beberapa atribut yang lebih kecil. Contoh: nama lengkap yang terdiri dari nama depan, nama tengah, dan nama belakang.

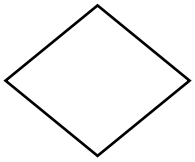
e. Atribut Derivatif (*Derived Attribute*): Atribut yang berasal dari atribut lain dan tidak selalu perlu ditampilkan dalam ERD. Contoh: usia yang dihitung dari tanggal lahir, selisih waktu, dan kelas atau ruang.



Gambar 2. 2 Simbol Atribut

3. Relasi

Relasi adalah hubungan yang terjadi antara satu atau lebih entitas



Gambar 2. 3 Simbol Relasi

2.1.10 *Unified Modeling Language* (UML)

UML (Unified Modeling Language) adalah pengembangan dari teknik pemrograman berorientasi objek yang menghasilkan bahasa pemodelan terstandarisasi untuk pengembangan perangkat lunak. UML digunakan untuk membuat analisis dan perancangan, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Saat ini, UML merupakan bahasa standar yang digunakan untuk visualisasi, spesifikasi, pembentukan, dan pendokumentasian komponen sistem perangkat lunak.

Use Case mendeskripsikan perilaku aplikasi yang akan dibuat. Use case menggambarkan interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibangun. Ini digunakan untuk mengidentifikasi fungsi-fungsi dalam sistem informasi dan pihak yang berhak mengakses fungsi-fungsi tersebut. *Activity Diagram* yang menggambarkan rangkaian aliran dari aktivitas dalam sebuah sistem, proses bisnis, atau menu perangkat lunak. *Class Diagram* menggambarkan struktur sistem dengan mendefinisikan kelas-kelas yang diperlukan untuk membangun sistem. Setiap kelas memiliki atribut, metode, atau operasi yang mendukung fungsi kelas tersebut (A. Rahman, 2023).

2.1.11 Use case Diagram

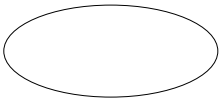
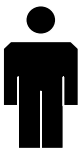
Use case atau diagram *use case* adalah pemodelan yang digunakan untuk mendeskripsikan perilaku (*behavior*) dari sistem informasi yang akan dibangun. Diagram ini menggambarkan interaksi antara satu atau lebih aktor dengan sistem informasi. Secara umum, use case digunakan untuk memahami fungsi-fungsi yang ada dalam sistem dan siapa saja yang berhak menggunakan fungsi tersebut.

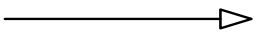
Penamaan dalam *use case* harus sesederhana mungkin dan mudah dipahami. Ada dua elemen utama dalam *use case*, yaitu aktor dan *use case*:

a. Aktor: Aktor adalah entitas yang berinteraksi dengan sistem dari luar sistem itu sendiri. Aktor bisa berupa orang, proses, atau sistem lain. Meskipun simbol aktor digambarkan sebagai orang, aktor tidak selalu merupakan manusia.

b. *Use Case*: *Use case* adalah fungsionalitas yang disediakan oleh sistem, di mana unit-unit dalam sistem berinteraksi atau saling bertukar pesan dengan aktor atau unit lainnya.

Tabel 2. 5 *Use case Diagram*

Simbol	Keterangan
	*Use case* menggambarkan fungsi yang diberikan oleh sistem melalui interaksi dengan aktor. Fungsinya biasanya dijelaskan dengan kata kerja, karena fokusnya adalah pada tindakan yang dilakukan oleh sistem dan aktor. Setiap *use case* mewakili aktivitas yang dapat dilakukan oleh sistem bersama aktor.
	Aktor adalah abstraksi dari orang atau sistem lain yang berinteraksi dengan dan mengaktifkan fungsi dari sistem yang ditargetkan. Untuk mengidentifikasi aktor, perlu ditentukan pembagian peran serta tugas-tugas yang terkait dengan peran tersebut dalam konteks sistem yang sedang dibangun. Satu orang atau sistem dapat muncul dalam beberapa peran, tergantung pada fungsi atau tugas yang dilakukan dalam interaksi dengan sistem.
	Asosiasi dalam *use case* adalah hubungan antara aktor dan *use case*, yang digambarkan dengan garis tanpa panah. Garis ini menunjukkan siapa atau apa yang terlibat dalam interaksi langsung dengan sistem, tetapi tidak menunjukkan aliran data. Asosiasi berfungsi untuk mengindikasikan bahwa aktor tersebut berperan dalam memulai atau

	berpartisipasi dalam fungsionalitas yang diwakili oleh *use case*.
	*Generalization* dalam *use case* menunjukkan hubungan hierarki antara aktor atau *use case* dan digambarkan dengan panah terbuka. Ini berarti bahwa aktor di ujung panah mewarisi peran dari aktor yang lebih umum di awal panah. Aktor tersebut bisa berinteraksi secara pasif dengan sistem melalui aktor yang lebih umum.
	*Includ*e adalah pemanggilan *use case* oleh *use case* lain, yang menggambarkan relasi antara *use case* utama dan *use case* tambahan. *Use case* tambahan memerlukan *use case* utama untuk dijalankan, atau menjadi syarat agar *use case* tambahan dapat berfungsi dengan baik. Dengan kata lain, *use case* tambahan bergantung pada *use case* utama untuk melengkapi atau mendukung fungsionalitasnya.
	**Extend** adalah perluasan dari *use case* lain yang terjadi jika kondisi tertentu terpenuhi. Ini menunjukkan hubungan antara *use case* utama dan *use case* turunan, di mana use case utama dapat berfungsi sendiri tanpa memerlukan use case tambahan. Use case turunan hanya dijalankan jika diperlukan.

2.1.12 Activity Diagram

Diagram aktivitas atau activity diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem, proses bisnis, atau menu dalam perangkat lunak.

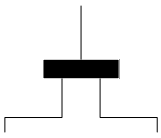
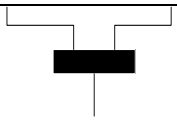
Penting untuk dicatat bahwa diagram ini fokus pada aktivitas yang dilakukan oleh sistem, bukan pada tindakan aktor.

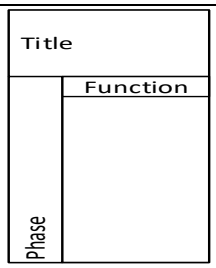
Diagram aktivitas sering digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan Proses Bisnis: Setiap urutan aktivitas menggambarkan proses bisnis yang didefinisikan dalam sistem.
2. Urutan Tampilan Sistem: Menunjukkan pengelompokan atau urutan tampilan antarmuka pengguna, di mana setiap aktivitas dianggap memiliki desain antarmuka.
3. Rancangan Pengujian: Setiap aktivitas dianggap memerlukan pengujian, yang perlu didefinisikan kasus uji-nya.
4. Rancangan Menu: Menampilkan menu yang ada dalam perangkat lunak (Sukamto & Shalahuddin, 2016).

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2. 6 *Activity Diagram*

Simbol	Keterangan
	*Initial Node/Start Point* diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
	*Final Node/End Point* merupakan akhir aktivitas.
	*Action/Activities*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork*/percabangan digunakan untuk menunjukkan kegiatan atau untuk menggabungkan dua kegiatan parallel menjadi satu.
	*Join*/penggabungan atau *rake* digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points* menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*.

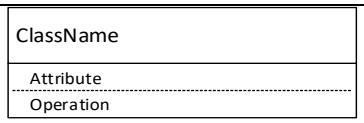
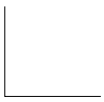
	*Swimlane* merupakan pembagian *activity diagram* untuk menunjukkan siapa yang melakukan apa.

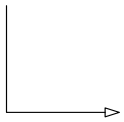
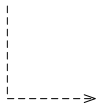
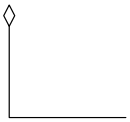
2.1.13 Class Diagram

Class adalah spesifikasi yang, ketika diinstansiasi, akan menghasilkan objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem serta menyediakan layanan untuk memanipulasi keadaan tersebut (metode/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, paket, dan objek, serta hubungan di antara mereka, seperti *containment*, pewarisan, dan asosiasi. *Class* juga merupakan implementasi dari *interface*, yaitu *class* abstrak yang hanya memiliki metode. *Interface* tidak dinyatakan secara langsung tetapi harus diimplementasikan menjadi sebuah *class* terlebih dahulu. Dengan demikian, *interface* mendukung resolusi metode pada saat *run-time* (Subhiyakto & Astuti, 2020).

Tabel 2. 7 *Class Diagram*

Simbol	Keterangan
	*Class* menggambarkan kelas pada struktur sistem.
	*Association* menggambarkan relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*.
	*Directed Association* menggambarka relasi antar kelas dengan makna kelas yang satu digunakan

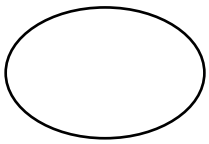
	oleh kelas lain, asosiasi biasanya disertai dengan *multiplicity*.
	*Generalization* menggambarkan relasi antar kelas dengan makna generalisasi spesialisasi (umum-khusus).
	*Dependency* menggambarkan kebergantungan antar kelas.
	*Aggregation* menggambarkan relasi antar kelas dengan makna semua bagian (*whole part*).

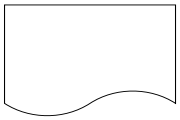
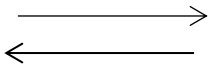
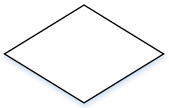
2.1.14 Flowchart

Flowchart merupakan bagian (*chart*) yang menunjukan alir atau arus (*flow*) didalam program atau prosedur sistem secara logika. *Flowchart* (bagan alir) merupakan gambaran – gambaran dalam bentuk diagram alir dari algoritma – algoritma dalam suatu program, yang menyatakan arah alur program tersebut (Solikin, 2018).

Tujuan penggunaan *flowchart* adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai dan rapi dengan menggunakan simbol – simbol yang standar yang dapat dimengerti .(Syamsiah, 2019). Simbol dan fungsi flowchart dapat tabel berikut :

Tabel 2. 8 *Flowchart*

Simbol	**Keterangan**
Simbol Titik Terminal (*Terminal point symbol*) 	Simbol ini digunakan untuk menunjukan awal dan akhir dari suatu proses

Simbol Dokumen 	Simbol ini menunjukan dokument input dan output baik untuk proses manual, mekanik atau komputer.
Simbol Kegiatan Manual 	Simbol ini menunjukan pekerjaan manual
Simbol Garis Alis 	Garis alir menunjukan arus dari proses
Simbol Proses 	Simbol ini menyatakan suatu tindakan yang dilakukan oleh komputer
Simbol Seleksi (*Decision*) 	Simbol ini menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan seperti ya atau tidak.

2.1.15 Flowmap

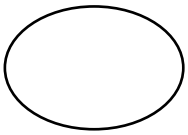
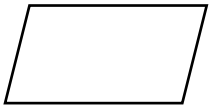
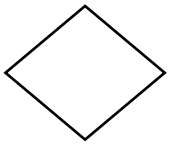
Flowmap adalah gabungan antara peta dan flowchart yang menunjukkan pergerakan benda dari satu lokasi ke lokasi lain, seperti jumlah orang dalam migrasi, jumlah barang yang diperdagangkan, atau jumlah paket dalam jaringan. Flowmap membantu analis dan programmer memecahkan masalah menjadi segmen-segmen yang lebih kecil dan menganalisis alternatif-alternatif dalam pengoperasian (Sandikapura & Sukendar, 2018). Kegunaan flowmap antara lain:

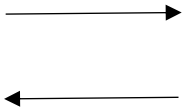
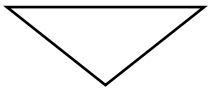
1. Menggambarkan aktivitas yang sedang berlangsung.
2. Menjabarkan aliran dokumen yang terlihat.
3. Menjelaskan hubungan antara data dan informasi dalam aktivitas.
4. Mendefinisikan hubungan antara bagian (pelaku proses) dan proses, baik manual maupun berbasis komputer.
5. Menampilkan aliran data dalam bentuk dokumen masukan dan keluaran.

Aturan membuat *flowmap* untuk membuat analisis menggunakan flowmap Analis dan Programmer memerlukan beberapa tahapan:

1. Peta alur digambar dari atas ke bawah atau dari kiri ke kanan.
2. Kegiatan yang dijelaskan harus didefinisikan dengan jelas dan dapat dimengerti oleh pembaca.
3. Tentukan dengan jelas kapan aktivitas dimulai dan kapan berakhir.
4. Setiap langkah kegiatan harus dijelaskan dengan kata kerja.
5. Setiap langkah harus dilakukan dalam urutan yang benar.
6. Pertimbangkan dengan hati-hati lingkup dan luasnya kegiatan yang dijelaskan. Cabang yang memotong aktivitas tidak perlu digambar dalam diagram alir yang sama.
7. Gunakan simbol flowmap yang standar.

Tabel 2. 9 Tabel Flowmap

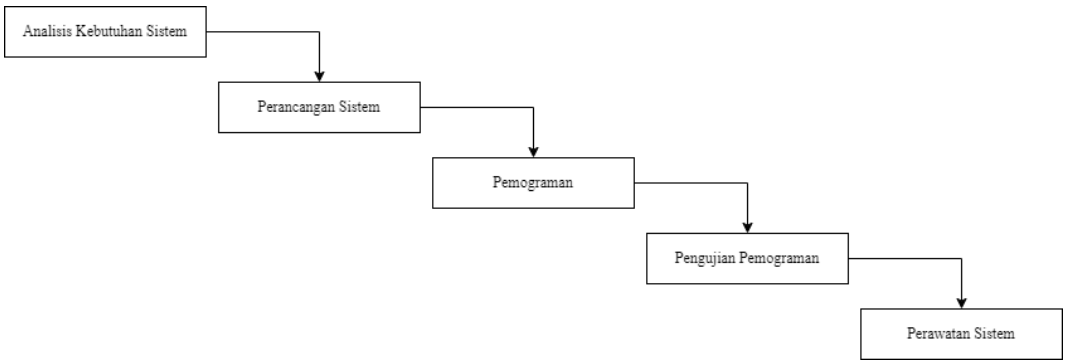
NO	SIMBOL	NAMA	KETERANGAN
1		Terminator	Awal dan Akhir *Flowmap*
2		Proses manual	Kegiatan proses yang dilakukan dengan komputerisasi
3		Proses Komputer	Kegiatan proses yang di lakukan dengan manual
4		*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu

5		Garis alir	Menunjukan alir data dari atau ke proses
6		Arsip	Menunjukan penyimpanan data yang dilakukan secara manual.
7		Dokumen	Menunjukan dokumen *input/output* baik untuk proses manual, mekanik, atau komputer

2.1.16 Metode Pengembangan Sistem

1. Metode *System Development Life Cycle (SDLC)*

Metode *System Development Life Cycle* (SDLC), yang dikenal juga sebagai metode *waterfall*, adalah cara yang digunakan untuk mengembangkan sistem atau proses logika. Metode ini membantu analis sistem merancang dan membuat sistem informasi. SDLC terdiri dari serangkaian langkah berurutan, mulai dari perencanaan, analisis kebutuhan, desain, pengembangan, pengujian, hingga implementasi dan pemeliharaan. Pendekatan ini memastikan setiap tahap diselesaikan sebelum melanjutkan ke tahap berikutnya. (Halimah & Abdullah, 2022).

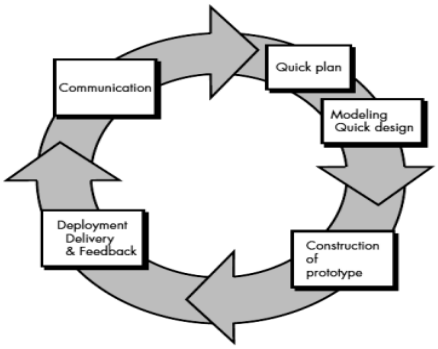


Gambar 2. 4 Model *Waterfall*

2. Metode *Prototyping*

Menurut Raymond McLeod, *prototype* adalah alat yang memberikan gambaran tentang cara sistem berfungsi dalam bentuk lengkap kepada pembuat dan pengguna potensial. Proses untuk menghasilkan sebuah *prototype* disebut *prototyping. Prototyping* adalah pembuatan model sederhana perangkat lunak yang memungkinkan pengguna memiliki pemahaman dasar tentang program dan melakukan pengujian awal. Proses ini memfasilitasi interaksi antara pengembang dan pengguna selama pembuatan, sehingga pengembang dapat dengan mudah memodelkan perangkat lunak yang akan dibuat. Agar model *prototype* berhasil, penting untuk mendefinisikan aturan permainan di awal, di mana pelanggan dan pengembang harus sepakat bahwa *prototype* yang dibangun akan digunakan untuk mendefinisikan kebutuhan. (Widiyanto, 2018).

Berikut merupakan tahapan dalam metode *prototype* (Aditya et al., 2021):



Gambar 2. 5 Model *Prototype*

a. *Communication* (Komunikasi)**:** Menganalisis kebutuhan pengguna.
b. *Quick Plan**:*** Tahap perencanaan kebutuhan.
c. *Modelling Quick Design**:*** Tahap pembuatan desain.
d. *Pembentukan Prototype**:*** Pembuatan perangkat prototype, termasuk pengujian dan penyempurnaan.

Deployment Delivery & Feedback: Tahap ini melibatkan evaluasi *prototype* dan penajaman analisis terhadap kebutuhan pengguna. Perbaikan *Prototype*: Pembuatan tipe akhir berdasarkan hasil evaluasi *prototype*. Produksi Akhir: Proses memproduksi perangkat secara benar agar siap digunakan oleh pengguna.

3. *Rapid Aplication Development (RAD)*

Rapid Application Development (RAD) adalah model proses pengembangan perangkat lunak yang sekuensial dan linier, dengan fokus pada siklus pengembangan yang sangat pendek, biasanya antara 60 hingga 90 hari. Model RAD adalah adaptasi "kecepatan tinggi" dari model sekuensial linier, di mana pengembangan yang cepat dicapai melalui pendekatan konstruksi berbasis komponen.

Menurut Kendal (2010), *Rapid Application Development* (RAD) adalah pendekatan berorientasi objek untuk pengembangan sistem yang mencakup metode pengembangan serta perangkat lunak. RAD bertujuan untuk mempersingkat waktu yang biasanya dibutuhkan dalam siklus hidup pengembangan sistem tradisional, antara perancangan dan penerapan sistem informasi. Akhirnya, RAD berusaha untuk memenuhi syarat bisnis yang berubah dengan cepat.

a. Pemodelan Bisnis (*Business Modeling*)

   Pada tahap pemodelan bisnis, aliran informasi dimodelkan untuk memahami beberapa hal, yaitu informasi yang mengendalikan proses bisnis, informasi yang dihasilkan, pihak yang membuat informasi tersebut, pihak yang mengolahnya, serta arah aliran informasi.

b. Pemodelan Data (*Data Modeling*)

   Aliran informasi yang telah dideskripsikan melalui pemodelan bisnis kemudian disaring untuk menjadi bagian-bagian dari objek data yang diperlukan guna mendukung proses bisnis tersebut.

c. Pemodelan Proses (*Process Modeling*)

   Objek-objek data yang telah didefinisikan sebelumnya diubah agar dapat menghasilkan aliran informasi yang diperlukan untuk fungsi bisnis. Pengolahan deskripsi dilakukan untuk menambah, mengubah, menghapus, atau mengambil kembali objek data tersebut.

d. Pembuatan Aplikasi (*Application Generation*)

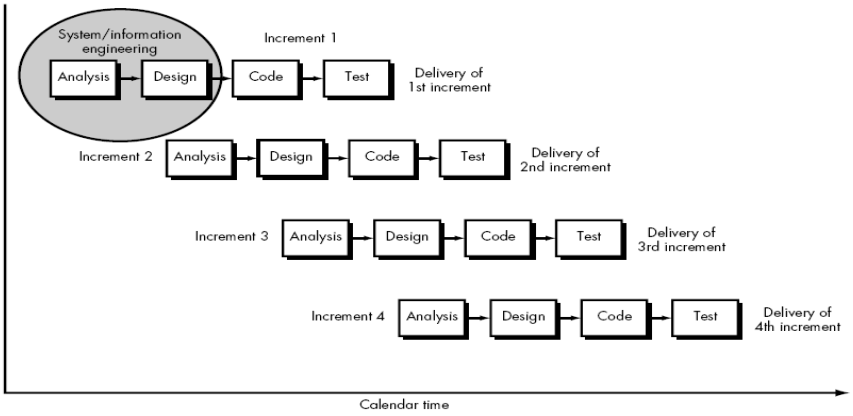
   RAD menggunakan teknik generasi keempat (4GT), sehingga pada

tahap ini jarang menggunakan bahasa pemrograman generasi ketiga (3G). Pembuatan aplikasi lebih ditekankan pada penggunaan kembali (reuse) komponen yang ada atau pembuatan komponen baru jika diperlukan.

e. Pengujian dan Pergantian (*Testing and Turnover*)
   Karena model ini menekankan pada penggunaan kembali komponen yang telah ada (*reuse*), komponen lama tersebut sudah diuji sebelumnya sehingga mengurangi waktu pengujian secara keseluruhan.

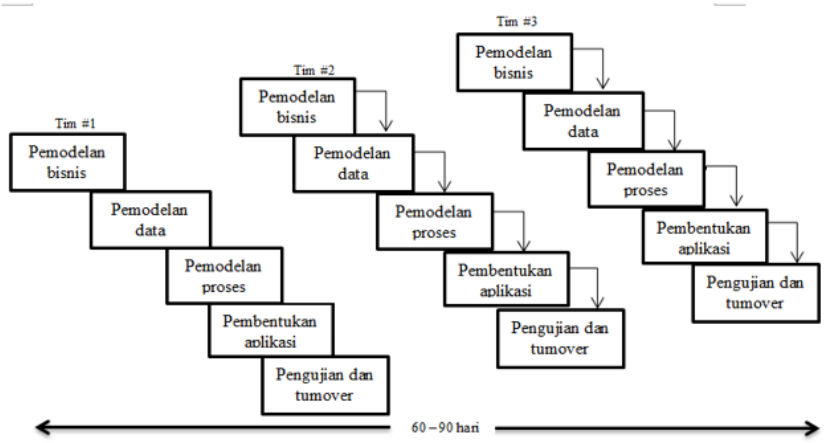
4. Model iteratif *(iterative model)*

Model iteratif (*iterative model*) menggabungkan proses dari model air terjun dengan pendekatan iteratif dari model *prototipe*. Dalam model inkremental, setiap versi perangkat lunak yang dihasilkan mengalami penambahan fungsi seiring dengan setiap inkremen (*increment*) yang dilakukan. Berikut adalah gambar dari model inkremental:



Gambar 2. 6 Model *Iterative*

Model inkremental dirancang untuk mengatasi kelemahan model air terjun yang tidak mendukung iterasi, serta mengatasi masalah pada metode *prototipe* yang memiliki proses terlalu pendek dan tidak selalu menghasilkan produk yang dapat digunakan (seringkali hanya berupa *prototipe*). Dengan model inkremental, produk atau aplikasi dihasilkan untuk setiap tahapan inkremen yang dilakukan.



Gambar 2. 7 Model *Rapid application development* (RAD)

2.2 *Software*

2.2.1 Codeigniter4

CodeIgniter adalah kerangka kerja pengembangan aplikasi PHP yang berbasis arsitektur terstruktur. Tujuannya adalah untuk menyediakan alat bantu seperti *helpers* dan *libraries* yang diperlukan untuk melaksanakan tugas-tugas umum. Hal ini membuat pengembangan proyek menjadi lebih mudah dan cepat, karena pengembang tidak perlu memulai dari nol. *CodeIgniter* merupakan kerangka kerja PHP yang kuat dengan sedikit *bug*, dirancang untuk pengembang yang memerlukan alat yang sederhana dan elegan dalam pembuatan aplikasi web berfitur lengkap. (A. Rahman, 2023)

2.2.2 Visual Code Studio

Untuk pembuatan kode program, dibutuhkan aplikasi yang mumpuni, salah satunya adalah *Visual Studio Code*. *Visual Studio Code* adalah editor kode sumber yang ringan namun kuat, yang berjalan di desktop. Editor ini dilengkapi dengan dukungan bawaan untuk *JavaScript*, *scripting*, dan *Node.js*, serta memiliki beragam ekstensi untuk bahasa pemrograman lain, termasuk C++, C#, Python, dan PHP. (Hartati, 2020)

2.2.3 PHP

PHP adalah aplikasi open source yang memudahkan manajemen *MySQL*. Sebagai bahasa pemrograman berbasis skrip yang berjalan di sisi server, PHP

diproses oleh server dan hasilnya dikirim ke klien, di mana pengguna dapat mengaksesnya melalui *browser* (Sitorus & Sakban, 2021).

2.2.4 XAMPP

XAMPP adalah paket perangkat lunak yang menggabungkan *Apache* HTTP Server, MySQL, dan PHP. Dengan menggunakan XAMPP, instalasi semua komponen yang diperlukan untuk pengembangan web dapat dilakukan dengan mudah, tanpa perlu menginstal setiap perangkat lunak secara terpisah. (Usia & Arfandy, 2020) .

2.2.5 MySQL

Menurut Priyanto dan Hidayatullah, MySQL (*My Structured Query Language*) adalah salah satu aplikasi DBMS yang banyak digunakan oleh para pengembang aplikasi web. Dalam MySQL, sebuah database terdiri dari satu atau lebih tabel. Tabel tersebut terdiri dari beberapa baris, di mana setiap baris memiliki satu atau lebih kolom. (Sitorus & Sakban, 2021)

2.3 Pengujian Sistem

2.3.1. *Black Box testing*

Black Box adalah metode pengujian yang fokus pada apa yang dilakukan oleh sistem, terutama terkait perilaku dan masalah yang muncul. Pengujian ini bertujuan untuk mengidentifikasi *bug* yang ada dalam hasil, pemrosesan, dan perilaku sistem. Biasanya, pengujian *Black Box* dilakukan oleh tester, yang akan mengidentifikasi berbagai kondisi *input* dan melakukan pengujian berdasarkan spesifikasi fungsional program. (Rosmiati, 2021).

2.3.2. White-Box Testing

Dengan memahami cara kerja internal suatu produk, pengujian dilakukan untuk memastikan bahwa semua operasi internal telah dilaksanakan sesuai dengan spesifikasi dan bahwa semua komponen internal berfungsi dengan baik. *White Box* Testing berfokus pada struktur kontrol program. (Cholifah, 2018).