

BAB II KAJIAN LITERATUR

2.1. Sistem Informasi Manajemen

SIM dapat dijelaskan menjadi beberapa bagian antara lain konsep dasar sistem, konsep dasar informasi dan sistem informasi dan konsep sistem informasi manajemen [4]. Sistem adalah merupakan satu kesatuan data yang terhubung dan terorganisir secara prosedural. Informasi adalah data yang telah dikelola dan diproses untuk memberikan arti dan memperbaiki proses pengambilan keputusan. Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi organisasi yang bersifat manajerial dalam kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan oleh pihak luar tertentu [5]. Sistem informasi manajemen (SIM), juga disebut sistem manajemen informasi (MIS), adalah sistem yang dirancang untuk mengumpulkan, menyimpan, dan menyebarkan data, yang merupakan informasi yang diperlukan untuk menjalankan berbagai fungsi manajemen [6].

2.1.1. Konsep Dasar Sistem

Sistem berasal dari bahasa Latin (*systema*) dan bahasa Yunani (*systema*) adalah suatu kesatuan yang terdiri atas elemen atau komponen yang dihubungkan bersama untuk memudahkan aliran informasi, materi, atau energi untuk mencapai suatu tujuan. Sistem adalah suatu peng-organisasian yang saling berinteraksi, saling tergantung dan terintegrasi dalam kesatuan variabel atau komponen [4].

2.1.2. Konsep Sistem Informasi

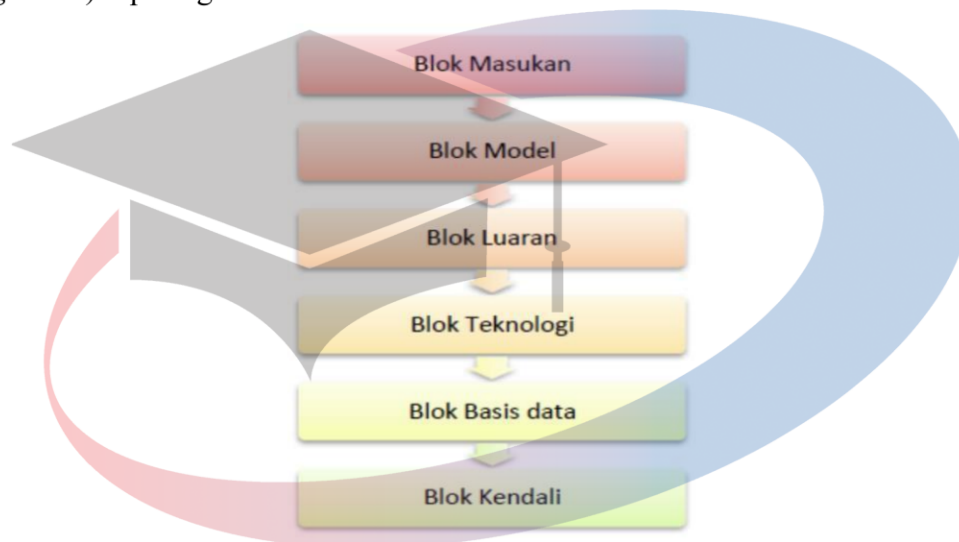
Informasi adalah pesan atau kumpulan pesan yang terdiri dari order sekuens dari simbol, atau makna yang dapat ditafsirkan dari pesan atau kumpulan pesan.

Sistem informasi adalah suatu sistem di dalam organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan.

Sistem informasi pada dasarnya mengandung 3 kegiatan inti, yakni *input* (masukan), pemrosesan, dan *output* (keluaran). Ketiga kegiatan tersebut dapat menghasilkan informasi yang diperlukan untuk pengambilan keputusan, pengendalian operasional, analisis pemecahan

masalah, dan menciptakan produk baru. Kegiatan input menerima dan mendeteksi bahan-bahan atau serangkaian data-data yang diperlukan baik dari internal maupun eksternal. Kegiatan pemrosesan mengolah dan menganalisis data yang diperoleh dari input menjadi suatu bentuk yang memiliki arti atau format yang dapat dipahami manusia. Kegiatan *output* menyalurkan informasi yang telah diolah kepada para pengguna. Setelah itu sistem informasi memerlukan umpan balik yang digunakan untuk evaluasi dan perbaikan dalam hal pengambilan keputusan berikutnya [4].

Sistem informasi terdiri dari komponen-komponen yang disebut dengan blok bangunan (*building block*) seperti gambar dibawah ini.



Gambar 2.1 Blok Bangunan

1. Blok Masukan, merupakan data yang terdiri dari metode dan media yang masuk ke dalam sistem informasi untuk menangkap data yang akan dimasukkan seperti dokumen-dokumen dasar.
2. Blok Model, merupakan model yang terdiri dari kombinasi logika, prosedur, dan model matematika yang kemudian akan memanipulasi data *input* dan data yang tersimpan di basis data dengan menggunakan cara yang telah ditentukan agar dapat menghasilkan *output* yang diinginkan.
3. Blok Luaran, merupakan luaran berupa informasi yang berkualitas dan dokumentasi yang bermanfaat bagi semua tingkatan manajemen serta semua pengguna sistem.
4. Blok Teknologi, merupakan *tool box* atau alat dari suatu pekerjaan sistem informasi. Teknologi digunakan untuk berbagai hal seperti, data masukan, menjalankan model, sebagai penyimpanan dalam mengakses data, menghasilkan dan mengirimkan *output* ke

pengguna, serta mengontrol sistem secara keseluruhan. Teknologi terdiri dari 3 jenis, yakni *hardware*, *software*, dan *brainware*.

5. Blok Basis Data, merupakan kumpulan dari berbagai macam data yang saling terhubung satu dengan yang lainnya, tersimpan pada *hardware* dan dimanipulasi oleh *software*. Data yang disimpan pada *database* harus diorganisasikan sedemikian rupa agar informasi yang dihasilkan dapat berkualitas. *Database* yang diorganisasikan dengan baik juga berguna agar kapasitas penyimpanan menjadi lebih efisien. *Database* diakses menggunakan paket *software* yang biasa disebut dengan *Database Management System (DBMS)*.
6. Blok Kendali, dirancang dan diterapkan untuk mencegah kerusakan sistem informasi dari hal-hal yang tidak diinginkan yang bersumber dari bencana, temperatur, air, debu kecurangan, kegagalan dari sistem itu sendiri. Sehingga apabila terlanjur terjadi kesalahan-kesalahan dapat dengan cepat diatasi [4].

2.1.3. Konsep Sistem Informasi Manajemen

Sistem Informasi Manajemen (SIM) adalah sistem perencanaan dalam pengendalian *internal* suatu bisnis yang mencakup penggunaan akuntansi manajemen atas manusia, dokumen, teknologi, dan prosedur untuk memecahkan masalah bisnis seperti biaya produk, layanan, atau suatu strategi bisnis [4].

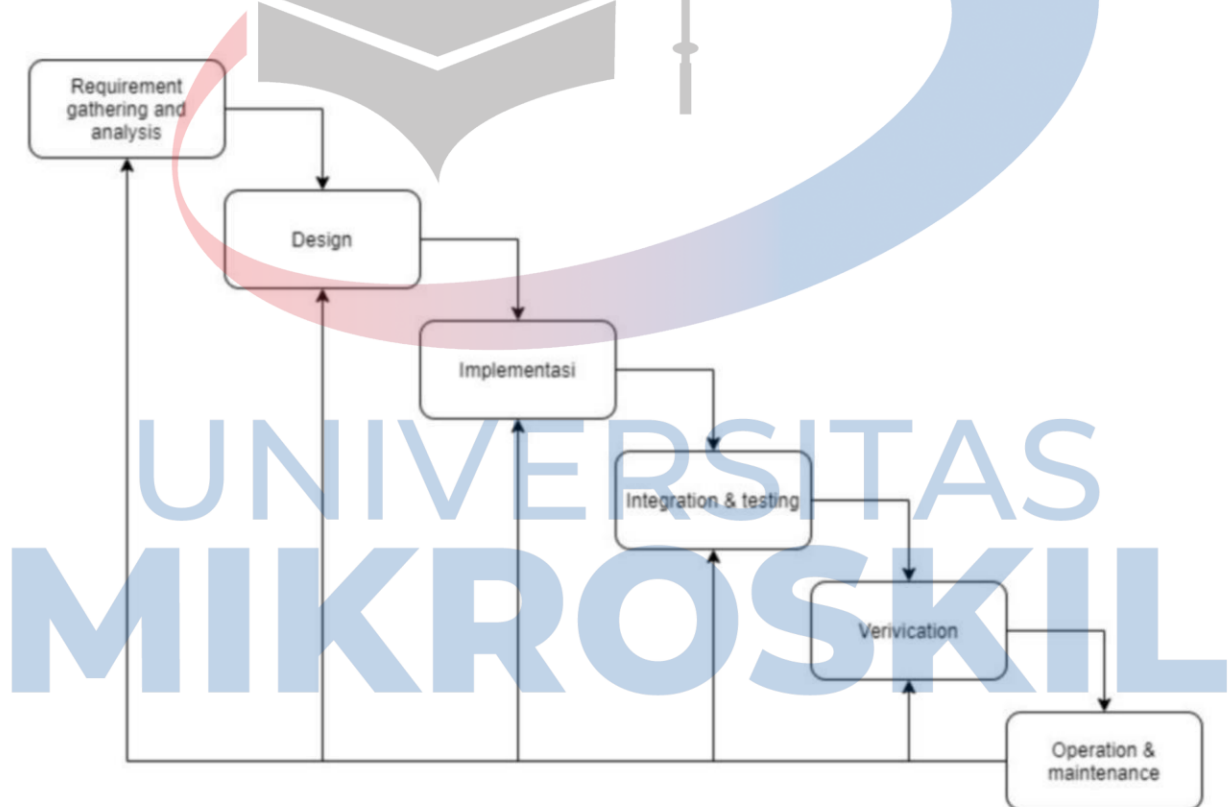
Sistem informasi manajemen terdiri dari lima komponen utama yaitu:

1. Orang adalah pengguna yang menggunakan sistem informasi untuk mencatat transaksi bisnis sehari-hari. Pengguna biasanya adalah profesional yang berkualifikasi seperti akuntan, manajer sumber daya manusia, dll. Departemen TIK biasanya memiliki staf pendukung yang memastikan bahwa sistem berjalan dengan baik.
2. Prosedur Bisnis, ini merupakan praktik terbaik yang disepakati yang memandu pengguna dan semua komponen lainnya tentang cara bekerja secara efisien. Prosedur bisnis dikembangkan oleh orang-orang yaitu pengguna, konsultan, dll.
3. Data, catatan transaksi bisnis sehari-hari. Untuk bank, data dikumpulkan dari aktivitas seperti penyetoran, penarikan, dll.
4. Perangkat Keras, perangkat keras terdiri dari komputer, printer, perangkat jaringan, dll. Perangkat keras menyediakan daya komputasi untuk memproses data. Ini juga menyediakan kemampuan jaringan dan pencetakan. Perangkat keras mempercepat pemrosesan data menjadi informasi.

5. *Software* adalah program yang berjalan di *hardware*. Perangkat lunak dipecah menjadi dua kategori utama yaitu perangkat lunak sistem dan perangkat lunak aplikasi. Perangkat lunak sistem mengacu pada sistem operasi yaitu Windows, Mac OS, dan Ubuntu, dll. Perangkat lunak aplikasi mengacu pada perangkat lunak khusus untuk menyelesaikan tugas-tugas bisnis seperti program Peng-gajian, sistem perbankan, sistem tempat penjualan, dll [4].

2.2. *Software Development Life Cycle (SDLC) dan Waterfall Model*

Metode SDLC adalah proses pembuatan dan perubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem rekayasa perangkat lunak. Metode *waterfall* adalah metode kerja yang menekankan fase-fase yang berurutan dan sistematis. Disebut *waterfall* karena proses mengalir satu arah ke bawah seperti air terjun. Metode *waterfall* ini harus dilakukan secara berurutan sesuai dengan tahap yang ada.



Gambar 2.2 Metode *Waterfall*

Berikut adalah tahap-tahap pengembangan dalam metode *waterfall* [4].

1. *Requirement gathering and analysis*

Mengumpulkan kebutuhan secara lengkap untuk dianalisis dan mendefinisikan kebutuhan apa saja yang harus dicapai oleh program. Informasi dapat diperoleh melalui wawancara, diskusi, atau *survey*.

2. *Design*

Melakukan perancangan desain perangkat lunak sebagai perkiraan sebelum dibuatnya kode. Desain sistem dapat dibuat menggunakan *Flowchart*, *Mind Map*, atau *Entity Relationship Diagram (ERD)*.

3. Implementasi

Tahap dimana seluruh desain yang sebelumnya sudah dibuat diubah menjadi kode-kode program. Kode yang dihasilkan masih berbentuk modul-modul yang harus digabungkan di tahap selanjutnya.

4. *Integration and testing*

Di tahap ini dilakukan penggabungan modul-modul yang sudah dibuat sebelumnya dan melakukan pengujian untuk mengetahui apakah perangkat lunak yang dibuat telah sesuai dengan desain dan fungsinya atau tidak.

5. *Verification*

Di tahap ini, pengguna atau klien yang langsung melakukan pengujian pada sistem, apakah sistem telah sesuai dengan yang disetujui atau belum sesuai.

6. *Operation & maintenance*

Tahap ini merupakan tahap terakhir dari *model waterfall*. Sistem yang sudah selesai dijalankan serta dilakukan pemeliharaan. Pemeliharaan berupa memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya.

2.3. *Website*

Website adalah kumpulan dari halaman-halaman situs yang biasanya terangkum dalam sebuah domain atau subdomain yang tempatnya berada didalam *World Wide Web (WWW)* di *Internet*. Sebuah *web page* adalah dokumen yang ditulis dalam format *Hyper Text Markup Language (HTML)* yang hampir selalu diakses melalui HTTP, yaitu *protocol* yang menyampaikan informasi dari server *website* untuk ditampilkan kepada para pemakai melalui *web browser*. Semua publikasi dari *website-website* tersebut dapat membentuk sebuah jaringan [7].

2.4. **Teknologi Yang Digunakan**

Berikut merupakan teknologi yang akan digunakan dalam pengembangan *website* dari Brayan Fitness Centre:

2.4.1. Frontend

Frontend bukan lagi sekedar membuat *website* statis. *Frontend* membuat *website* SPA *Single Page Application*, dimana *user* tidak perlu terus berpindah halaman saat bermain dengan *websitenya*, membuat penggunaannya semakin nyaman. Pada dasarnya, *frontend* adalah salah satu bagian dari *website* yang menampilkan tampilan pada para pengguna. Bagian ini dibuat dengan menggunakan *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS), dan juga JavaScript. Sehingga, suatu URL bisa bekerja dan menampilkan situs *website* dengan baik.

HTML adalah singkatan dari *Hyper Text Markup Language* yang merupakan bahasa pemrograman dasar dalam pembuatan *website*, HTML terdiri dari *Head*, *Body* dan di dalamnya terdapat *Tag* dan *Attribute*, walaupun dikatakan sebagai bahasa pemrograman, tetapi HTML belum dapat dikatakan sebagai bahasa pemrograman karena HTML tidak memiliki hal-hal yang di butuhkan oleh Bahasa pemrograman yaitu logika.

CSS atau singkatan dari *Cascading Style Sheet* adalah suatu aturan untuk mengatur tampilan dari *website* sehingga tampilan dalam *web* lebih terstruktur. CSS sendiri bukanlah bahasa pemrograman, CSS lebih seperti konfigurasi tampilan dari suatu *tag* pada *website*. CSS dapat merubah *text*, warna, *background* dan posisi dari suatu *tag* [8].

Pada penelitian ini penulis akan menggunakan ReactJs sebagai fondasi utama untuk tampilan dan menggunakan *Node Package Manager* (NPM) untuk menambahkan *package*.

2.4.1.1. ReactJs

React adalah *open-source library* JavaScript deklaratif, efisien dan fleksibel untuk membangun antarmuka pengguna. React memungkinkan untuk membuat *user interface* yang kompleks dengan set kode kecil yang terisolasi yang disebut "komponen". React JS ini digunakan untuk menangani lapisan tampilan dalam aplikasi satu halaman dan pengembangan *mobile application*. React JS dikelola oleh Facebook, Instagram, komunitas pengembang dan korporasi. React berusaha untuk memberikan kecepatan, kesederhanaan, dan skalabilitas. Beberapa fitur yang paling mencolok adalah JSX, Komponen *Stateful*, Model Objek Dokumen Virtual [9].

JSX adalah sintaks seperti XML atau HTML yang digunakan oleh React untuk memperluas ECMAScript sehingga teks seperti XML atau HTML dapat digunakan berdampingan dengan kode JavaScript atau React. React memiliki logika rendering secara *inheren* digabungkan dengan logika UI lainnya, seperti bagaimana *event* ditangani, bagaimana

status dapat berubah dari waktu ke waktu, dan bagaimana data dapat disiapkan untuk ditampilkan. JSX direkomendasikan untuk digunakan bersama dengan react dengan tujuan untuk menjelaskan bagaimana tampilan UI sehingga memungkinkan react untuk menampilkan pesan dan peringatan yang lebih berguna bagi pengembang. JSX memudahkan pengembang karena kode antara markup (HTML) dapat digabungkan langsung dengan logika (Javascript) sehingga penulisan sintaksnya lebih sederhana dan mudah daripada penulisan keduanya secara terpisah [9].

2.4.1.2. Node Package Manager (NPM)

NPM merupakan *Node Package Manager* yang populer digunakan secara luas oleh pengembang JavaScript untuk berbagi *tool*, menginstal modul, dan mengelola dependensi. NPM dapat dilihat sebagai repositori untuk proyek *open source* bagi siapa saja yang ingin menggunakan dan mempublikasikan kode-kode tertentu. Hal yang dapat dilakukan NPM adalah dapat *menginstal* dan *meng-uninstal package*, serta mengelola versi dan dependensi yang diperlukan untuk menjalankan proyek [9].

2.4.2. Backend

Backend adalah ‘tulang belakang’ dari sebuah *website*. *Backend engineer* berfungsi untuk menyiapkan data yang akan muncul pada sebuah *website*. *Backend* juga bertanggung jawab atas fungsionalitas dari *website* tersebut. Meskipun perannya tidak diketahui banyak orang dan tidak berhubungan langsung dengan *user*, tapi kelancaran sebuah *website* tergantung dari bagaimana *backend* membangun pondasi data di belakangnya [8]. *Backend* merupakan halaman *web* yang diakses oleh pihak pengelola *website* itu sendiri [10]. Pada penelitian ini penulis akan menggunakan bahasa pemrograman *backend* menggunakan Go dan menghasilkan API untuk mengirimkan data ke *frontend*.

2.4.2.1. Golang

Golang (*Go Language*) merupakan sebuah bahasa pemrograman yang mengkombinasikan keamanan dan performa untuk pengembangan sistem yang bersifat *open source* dan dikembangkan di Google oleh Rob Pike, Robert Griesemer, dan Ken Thompson beserta kontributor lainnya dalam komunitas pengembang *open source*. Tujuan dari pengembangannya adalah untuk membangun bahasa yang mempunyai keunggulan dari sisi kecepatan, keandalan, skalabilitas, dan kesederhanaan. Golang juga termasuk dalam bahasa

yang dapat diketik secara statis serta menghasilkan kode *biner* pada mesin yang dapat dikompilasi. Hingga saat ini, Golang mulai banyak digunakan pada perusahaan-perusahaan besar maupun *startup* yang bergerak di bidang teknologi [8], [11]. Hal ini dikarenakan pemrograman dengan Golang ini memiliki beberapa kelebihan, antara lain [11]:

1. Mendukung konkurensi dalam sistem pemrograman dengan sangat baik dengan peng-aplikasiannya sendiri yang cukup mudah.
2. Merupakan bahasa pemrograman yang bersifat *open source*.
3. Memiliki sistem *garbage collection* yang baik dengan memanfaatkan bantuan *built-in garbage collector process* (Goroutines).
4. Memiliki sintaks yang bersifat bersih, tidak membebani sistem terlalu berlebihan.

2.4.2.2. REST API

Representational State Transfer (REST) adalah arsitektur perangkat lunak yang digunakan untuk sistem layanan *web* terdistribusi. Sedangkan REST API adalah aplikasi yang menyediakan data *GET, PUT, POST* dan *DELETE* ke layanan web. Layanan web (*web service*) sendiri adalah *web* yang dibuat untuk mendukung kebutuhan situs *web* atau aplikasi lain untuk akses langsung dan respons terhadap permintaan klien melalui antarmuka pemrograman aplikasi sehingga memungkinkan klien dapat melihat data dan memfasilitasi pertukaran antar program komputer [12].

API atau *web service* merupakan salah satu faktor yang mempengaruhi performansi suatu sistem. API (*Application Programming Interface*) merupakan sekumpulan prosedur yang dapat digunakan oleh aplikasi lain untuk memenuhi kebutuhannya. API sendiri dapat diartikan sebagai antarmuka yang dibangun oleh pengembang sistem agar sebagian atau keseluruhan fungsi sistem dapat diakses secara terprogram. API dibangun dengan tujuan mampu mempersingkat proses pengembangan sebuah perangkat lunak agar fitur-fitur yang sama bisa digunakan oleh perangkat lunak lain dan pengembang tidak perlu membangun fitur yang sama lagi [13].

2.5. Database

Database merupakan suatu koleksi terstruktur dari data yang saling terkait, disimpan dalam media penyimpanan komputer, dan dapat diakses serta dikelola menggunakan perangkat lunak khusus. *Database* digunakan untuk menyimpan, mengelola, dan mengorganisir data dengan tujuan memberikan akses yang efisien, aman, dan terstruktur terhadap informasi [14].

Berikut adalah beberapa peran utama yang dimainkan oleh sistem *database* [14]:

1. Penyimpanan Data: Peran utama sistem *database* adalah menyimpan data secara terstruktur dan terorganisir. Data dapat disimpan dalam tabel relasional, dokumen, grafik, atau format lainnya tergantung pada jenis sistem *database* yang digunakan. Sistem *database* memungkinkan penyimpanan data yang efisien, aman, dan dapat diakses dengan cepat.
2. Pengelolaan Data: Sistem *database* memungkinkan pengelolaan data secara efektif. Ini termasuk pembuatan, pengeditan, dan penghapusan data. Dengan menggunakan bahasa *query* seperti SQL (*Structured Query Language*), pengguna dapat dengan mudah melakukan manipulasi data, seperti menyaring data, mengurutkan data, menggabungkan data dari beberapa tabel, dan lain sebagainya.
3. Keamanan Data: Sistem *database* menyediakan fitur keamanan untuk melindungi data dari akses yang tidak sah. Ini mencakup autentikasi pengguna, kontrol akses berbasis peran, enkripsi data, dan *audit trail*. Dengan menggunakan sistem *database* yang tepat, organisasi dapat menjaga kerahasiaan, integritas, dan ketersediaan data mereka.
4. Konsistensi Data: Sistem *database* memastikan konsistensi data dengan menerapkan aturan integritas data. Aturan ini memastikan bahwa data yang dimasukkan ke dalam *database* memenuhi batasan dan hubungan yang telah ditentukan sebelumnya. Misalnya, dengan menggunakan kunci asing, sistem *database* dapat memastikan bahwa tidak ada data yang terhapus secara tidak sengaja yang dapat menyebabkan inkonsistensi.
5. Pemulihan Data: Sistem *database* menyediakan fitur pemulihan data dalam kasus kegagalan sistem, seperti kerusakan *hardware* atau kesalahan manusia. Dengan menggunakan teknik seperti *backup* dan *restore*, *log* transaksi, dan replikasi data, sistem *database* dapat memulihkan data ke keadaan yang konsisten dan dapat dipulihkan setelah kegagalan.
6. Skalabilitas dan Kinerja: Sistem *database* dirancang untuk mengelola jumlah data yang besar dan memungkinkan skalabilitas vertikal (menambah kapasitas perangkat keras) dan skalabilitas *horizontal* (menambah jumlah server). Sistem *database* juga berupaya untuk memberikan kinerja yang tinggi dengan menggunakan indeks, optimasi *query*, dan teknik lainnya.
7. Integrasi Aplikasi: Sistem *database* memungkinkan integrasi data antara aplikasi yang berbeda. Dengan menggunakan antarmuka atau API yang tepat, aplikasi dapat berkomunikasi dengan sistem *database* untuk mengambil atau memperbarui data. Hal ini memungkinkan berbagi data yang konsisten dan terintegrasi antara aplikasi yang berbeda.

Database Management System (DBMS) merupakan perangkat lunak untuk mengendalikan pembuatan, pemeliharaan, pengolahan, dan penggunaan data yang berskala besar. Penggunaan DBMS saat ini merupakan hal yang sangat penting dalam segala aspek, baik itu dalam skala yang besar atau kecil. Sebagai contoh media sosial Facebook menggunakan DBMS untuk menyimpan data-data pengguna Facebook yang sangat banyak kedalam DBMS MySQL. MySQL merupakan *software database open source* yang paling populer di dunia. MySQL menjadi pilihan utama bagi banyak pengembang *software* dan aplikasi hal ini dikarenakan kelebihan MySQL diantaranya sintaksnya yang mudah dipahami, didukung program-program umum seperti C, C++, Java, PHP, Python. Pengguna MySQL tidak hanya sebatas pengguna perseorangan maupun perusahaan kecil, namun perusahaan seperti Yahoo!, Google, Nokia, Youtube, Wordpress juga menggunakan DBMS MySQL [15]. Dalam penelitian ini penulis memilih menggunakan MySQL sebagai tempat penyimpanan data dari anggota dan data transaksi keanggotaan.

2.5.1. Normalisasi Tabel

Normalisasi adalah suatu teknik untuk menghasilkan sekumpulan relasi/tabel yang memiliki karakteristik tertentu, untuk memenuhi kebutuhan organisasi [16]. Proses normalisasi diperlukan dalam membentuk tabel-tabel yang normal [16], [17] :

1. Bentuk Tidak Normal (UNF)

Data awal yang masih belum memiliki format, masih terdapat data yang berulang dan data yang tidak lengkap. Data yang dikumpulkan murni berdasarkan input yang telah diperoleh.

2. Bentuk Normal Tahap 1 (1NF)

Bentuk normal yang pertama atau 1NF mensyaratkan beberapa kondisi dalam sebuah *database*, berikut adalah fungsi dari bentuk normal pertama ini.

- a. Menghilangkan duplikasi kolom dari tabel yang sama.
- b. Buat tabel terpisah untuk masing-masing kelompok data terkait dan mengidentifikasi setiap baris dengan kolom yang unik (*primary key*).

3. Bentuk Normal Tahap 2 (2NF)

Syarat untuk menerapkan normalisasi bentuk kedua ini adalah data telah dibentuk dalam 1NF, berikut adalah beberapa fungsi normalisasi 2NF.

- a. Menghapus beberapa subset data yang ada pada tabel dan menempatkan mereka pada tabel terpisah.
- b. Menciptakan hubungan antara tabel baru dan tabel lama menciptakan *foreign key*.

c. Tidak ada atribut dalam tabel yang secara fungsional bergantung pada *candidate key* tabel tersebut.

4. Bentuk Normal Tahap 3 (3NF)

Normalisasi *database* dalam bentuk 3NF bertujuan untuk menghilangkan seluruh atribut atau *field* yang tidak berhubungan dengan *primary key*. Dengan demikian tidak ada ketergantungan transitif pada setiap *candidate key*. Syarat dari bentuk normal ketiga atau 3NF adalah:

- Memenuhi semua persyaratan dari bentuk normal kedua.
- Menghapus kolom yang tidak tergantung pada *primary key*.

Contoh normalisasi tabel sebagai berikut:

UNF	1NF	2NF	3NF
Kode Produk	TABEL PRODUK	TABEL PRODUK	TABEL PRODUK
Nama Produk	Kode Produk	Kode Produk	Kode Produk
No Mesin	Nama Produk	Nama Produk	Nama Produk
Remarks	Remarks	Remarks	Remarks
Harga Jual	Harga Jual	Harga Jual	Harga Jual
Stok	Stok	Stok	Stok
Nama Pelanggan	TABEL PRODUKSI	TABEL PRODUKSI	TABEL PRODUKSI
Alamat Pelanggan	Kode Produk	Kode Produksi	Kode Produksi
Email Pelanggan	Kode Karyawan	Kode Produk	Kode Produk
Telepon Pelanggan	Nama Karyawan	Kode Karyawan	Kode Karyawan
Tanggal Produksi	Jabatan	No Mesin	No Mesin
Jumlah Produksi	No Mesin	Tanggal Produksi	Tanggal Produksi
Lokasi	Tanggal Produksi	Jumlah Produksi	Jumlah Produksi
Tanggal Penerimaan	Jumlah Produksi	TABEL WAREHOUSE	TABEL WAREHOUSE
Tanggal Pengiriman	TABEL WAREHOUSE	Kode Produksi	Kode Warehouse
Tanggal Penjualan	Kode Produk	Kode Produk	Kode Produksi
Jumlah Penjualan	Kode Karyawan	Kode Karyawan	Kode Produk
Kode Karyawan	Tanggal Penerimaan	Lokasi	Kode Karyawan
Nama Karyawan	Tanggal Pengiriman	Tanggal Pengiriman	Lokasi
Jabatan	Lokasi	Tanggal Penerimaan	Tanggal Penerimaan
	Nama Karyawan	Jabatan	Tanggal Pengiriman
	Jabatan	Status	Status
	Status	TABEL PENJUALAN	TABEL PENJUALAN
	TABEL PENJUALAN	Kode Produk	Kode Penjualan
	Kode Pelanggan	Jumlah Penjualan	Kode Pelanggan
	Kode Produk	Tanggal Penjualan	Kode Produk
	Jumlah Penjualan	Nama Pelanggan	Jumlah Penjualan
	Tanggal Penjualan	Alamat Pelanggan	Tanggal Penjualan
	Nama Pelanggan	Email Pelanggan	TABEL KARYAWAN
	Alamat Pelanggan	Telepon Pelanggan	Kode Karyawan
	Email Pelanggan	TABEL KARYAWAN	Nama Karyawan
	Telepon Pelanggan	Kode Karyawan	Jabatan
		Nama Karyawan	TABEL PELANGGAN
		Jabatan	Kode Pelanggan
			Nama Pelanggan
			Alamat Pelanggan
			Email Pelanggan
			Telepon Pelanggan

Gambar 2.3 Normalisasi *Database*

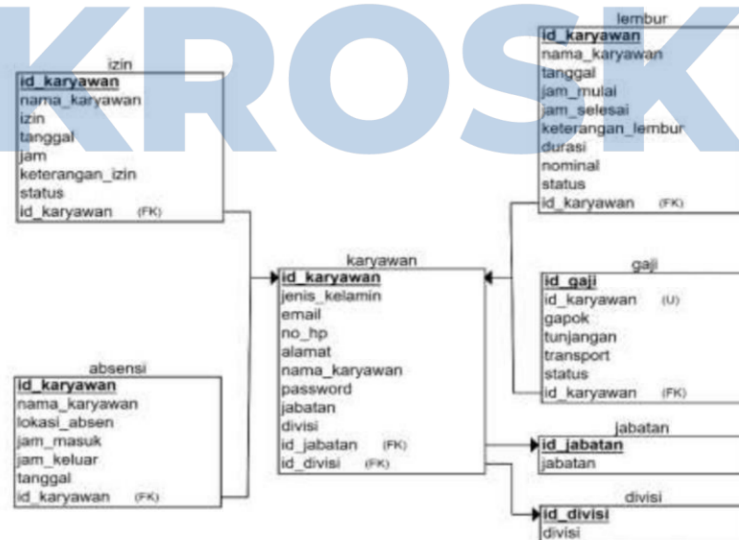
2.6. Logical Record Structure (LRS)

Entity Relationship Diagram (ERD) adalah suatu representasi visual yang menggambarkan koneksi antara elemen-elemen (kondisi relasional), yang kemudian dapat diaplikasikan dalam bentuk tabel relasional pada tahap berikutnya [18].

Logical Record Structure (LRS) merupakan struktur data yang digunakan dalam bidang komputasi dan pemrosesan data. Ini merujuk pada cara bagaimana data dalam suatu *file* atau dataset diorganisir, direpresentasikan, dan disusun secara logis tanpa memperhatikan bagaimana data tersebut disimpan secara fisik dalam media penyimpanan [18].

logical record structure (LRS) adalah representasi dari struktur *record-record* pada tabel-tabel yang terbentuk dari hasil antar himpunan entitas. Menentukan kardinalitas, jumlah tabel dan *foreign key*. Model sistem yang digambarkan dengan sebuah *entity relationship diagram* akan mengikuti pola atau aturan pemodelan tertentu dalam kaitannya dengan konversi ke LRS, maka perubahan yang terjadi adalah mengikuti aturan sebagai berikut. Setiap entitas akan dirubah menjadi bentuk kotak. Sebuah atribut relasi disatukan dalam sebuah kotak bersama entitas jika hubungan yang terjadi pada *entity relationship diagram* 1:M (relasi bersatu dengan kardinalitas M) atau tingkat hubungan 1:1 (relasi bersatu dengan kardinalitas yang paling membutuhkan referensi), sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubungannya M:M (*many to many*) dan memiliki *foreign key* sebagai *primary key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan [19].

Bentuk diagram *Entity Relationship Diagram* (ERD) yang sudah menjadi *Logical Record Structure* (LRS) sebagai berikut [18]:




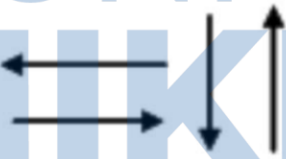


Gambar 2.3 Logical Record Structure (LRS)

2.7. Data Flow Diagram (DFD)

Data Flow Diagram (DFD) disebut juga dengan Diagram Arus Data (DAD). DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data, dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut, dan interaksi antara data yang tersimpan, dan proses yang dikenakan pada data tersebut [20].

Tabel 2.1 Simbol DFD

No	Gambar	Keterangan
1		Kesatuan Luar (<i>External Entity</i>) Merupakan kesatuan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada diluar lingkungan luarnya yang akan memberikan <i>input</i> atau menerima <i>output</i> sistem.
2		Proses, simbol ini digunakan untuk melakukan proses pengolahan data, yang menunjukkan suatu kegiatan yang mengubah aliran data yang masuk menjadi keluaran.
3		Penyimpanan Data/ <i>Data Store</i> merupakan tempat penyimpanan dokumen-dokumen atau <i>file-file</i> yang dibutuhkan.
4		Aliran Data, menunjukkan arus data dalam proses.

Syarat pembuatan DFD ini akan menolong profesional sistem untuk menghindari pembentukan DFD yang salah atau DFD yang tidak lengkap atau tidak konsisten secara logika. Beberapa syarat pembuatan DFD dapat menolong profesional sistem untuk membentuk DFD yang benar, menyenangkan untuk dilihat dan mudah dibaca oleh pemakai [21]. Syarat-syarat pembuatan DFD ini adalah [21]:

1. Pemberian nama untuk tiap komponen DFD
2. Pemberian nomor pada komponen proses
3. Penggambaran DFD sesering mungkin agar enak dilihat

4. Penghindaran penggambaran DFD yang rumit
5. Memastikan DFD yang dibentuk itu konsisten secara logika

Umumnya kesalahan dalam pembuatan DFD adalah [22]:

1. Proses mempunyai *input* tetapi tidak menghasilkan *output*. Kesalahan ini disebut dengan *black hole* (lubang hitam), karena data masuk ke dalam proses dan lenyap tidak berbekas seperti dimasukkan ke dalam lubang hitam.
2. Proses menghasilkan *output* tetapi tidak pernah menerima *input*. Kesalahan ini disebut dengan *miracle* (ajaib), karena ajaib dihasilkan *output* tanpa pernah menerima *input*.
3. *Input* yang masuk tidak sesuai dengan kebutuhan proses.
4. Data *Store* tidak memiliki keluaran.
5. Data *Store* tidak memiliki masukan.
6. Hubungan langsung antar entitas luar.
7. Masukan langsung entitas data *store*.
8. Keluaran langsung dari data *store* ke Entitas luar.
9. Hubungan langsung antar data *store*.
10. Ketidaksesuaian antara data masukan dan keluaran dalam penyimpanan data.



UNIVERSITAS
MIKROSKIL