BAB II

LANDASAN TEORI

BAB II
LANDASAN TEORI

2.1 Landasan Teori

2.1.1 Sistem

Sistem adalah sekumpulan elemen atau komponen yang saling berinteraksi dan bekerja sama untuk mencapai tujuan tertentu. Sistem terdiri dari tiga unsur yaitu: *input* (masukan), proses dan *output* (pengeluaran). *Input* merupakan komponen penggerak atau pemberi tenaga di mana sistem itu dioperasikan, sedangkan *output* adalah hasil operasi. Dalam pengertian sederhana *output* berarti yang menjadi tujuan sasaran atau target pengoperasian suatu sistem [4] . Sistem adalah kumpulan elemen-elemen yang terdiri dari data, jaringan prosedur yang saling berinteraksi, sumber daya manusia, serta teknologi baik perangkat keras maupun perangkat lunak. Semua elemen ini saling berhubungan sebagai satu kesatuan untuk mencapai tujuan atau sasaran tertentu [5].

2.1.2 Informasi

Informasi adalah data yang telah diproses atau diatur sedemikian rupa sehingga memiliki makna dan dapat digunakan untuk pengambilan keputusan atau tujuan tertentu Informasi adalah data yang telah diproses menjadi bentuk yang bermakna bagi penerimanya dan berguna dalam pengambilan keputusan saat ini maupun di masa depan [6].

2.1.3 Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mengintegrasikan kebutuhan pengelolaan transaksi harian, mendukung operasional, bersifat manajerial, dan kegiatan strategi dari suatu organisasi tertentu dengan penyediaan laporan-laporan yang dibutuhkan [7]. Sistem informasi (*Information System*) adalah suatu kerangka atau struktur yang terdiri dari komponen-komponen yang saling terkait, yang terdiri dari orang, perangkat keras (*hardware*), perangkat lunak (*software*), jaringan komunikasi, dan sumber daya data. Tujuan dari sistem informasi adalah untuk mengumpulkan, memproses, dan menyebarkan informasi guna mendukung pengambilan keputusan, manajemen, dan interaksi dalam sebuah organisasi [8].

2.1.4 Sistem Informasi Pembayaran

Pembayaran merujuk pada tindakan atau proses berpindah nilai ekonomi dari satu pihak ke pihak lain atau berpindahnya hak pemilikan atas sejumlah uang dari pembayar kepada penerimanya sebagai imbalan atas barang, jasa, atau kewajiban tertentu, baik dengan uang tunai maupun transfer bank. Sistem informasi pembayaran adalah suatu sistem informasi yang dirancang

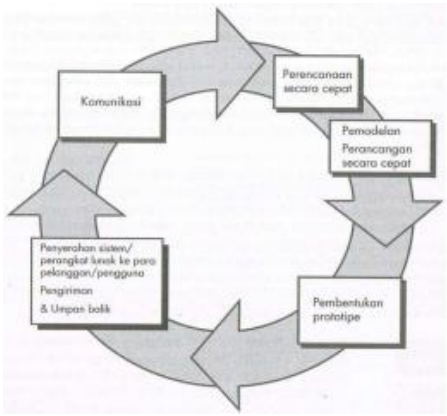
untuk mengelola dan memfasilitasi proses pembayaran dalam suatu organisasi atau lingkungan bisnis. Tujuan dari sistem ini adalah untuk mempercepat proses pembayaran, memastikan keakuratan data pembayaran, dan meningkatkan efisiensi dalam pengelolaan transaksi pembayaran [9]. Sistem informasi pembayaran adalah suatu sistem yang dirancang untuk mengelola dan memfasilitasi proses pembayaran dalam suatu organisasi. Sistem ini mengintegrasikan berbagai komponen teknologi informasi untuk mendukung aktivitas terkait pembayaran, seperti pencatatan transaksi, verifikasi pembayaran, pengelolaan data pelanggan, dan pembuatan laporan keuangan [10].

2.1.5 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang berkaitan dengan pengembangan, perancangan, pembuatan, dan pemeliharaan perangkat lunak yang berkualitas tinggi [11]. Rekayasa perangkat lunak merupakan bidang ilmu yang mencakup seluruh tahapan produksi perangkat lunak, dimulai dari analisis kebutuhan pengguna, penentuan detail kebutuhan pengguna, desain, pengkodean, pengujian, hingga pemeliharaan sistem setelah digunakan. Tujuannya adalah untuk menghasilkan perangkat lunak yang berkualitas tinggi yang memenuhi kebutuhan pengguna dan dapat beroperasi secara efektif. Rekayasa perangkat lunak melibatkan pendekatan terstruktur dan metodologi pengembangan yang dirancang untuk memastikan kualitas dan keberlanjutan perangkat lunak [12].

A. Metode Pengembangan Sistem

Metode yang digunakan untuk pengembangan sistem adalah *prototype. Prototype* adalah representasi awal atau contoh dari sebuah produk, sistem, atau perangkat lunak yang dibuat dengan tujuan untuk menguji konsep, desain, fungsi, atau fitur sebelum tahap produksi atau konstruksi sebenarnya dimulai [13]. Metode *prototype* yang digunakan di dalam penelitian ini bertujuan untuk mendapatkan gambaran sistem yang akan dibangun melalui rancangan metode *prototype* yang kemudian akan dievaluasi oleh *user*. Setelah melalui evaluasi oleh *user*, rancangan tersebut akan menjadi pedoman untuk pembuatan sistem yang akan sebagai hasil akhir dari penelitian ini. Berikut penjelasan dari tahapan metode *prototype*, yaitu [14]:



Gambar 2.1 Tahapan dalam Metode *Prototype* [15]

1. Komunikasi

   Proses komunikasi yang dilakukan adalah dengan wawancara untuk mengumpulkan dan mengetahui kebutuhan *user* terhadap produk akhir. Komunikasi membantu pengembang untuk memahami kebutuhan dan garis besar sistem sesuai dengan keinginan pengguna terkait dengan sistem yang akan dikembangkan.

2. Perencanaan Cepat

   Tahapan ini dikerjakan dengan kegiatan penentuan spesifikasi untuk pengembangan berdasarkan kebutuhan sistem yang didasarkan pada hasil komunikasi yang dilakukan agar pengembangan dapat sesuai dengan yang diharapkan.

3. Pemodelan secara Cepat

   Menggambarkan model sistem yang akan dikembangkan melibatkan proses perancangan dengan menggunakan *Unified Modeling Language* (UML). Pada tahap ini, *prototype* dibangun dengan perancangan sistem sementara, kemudian dievaluasi dengan *user* untuk memastikan kesesuaian dengan yang diinginkan atau masih perlu dilakukan evaluasi kembali.

4. Pembentukan *Prototype*

   Setelah pemodelan *prototype* sesuai dengan keinginan pengguna, langkah berikutnya yaitu pembuatan sistem (pengkodingan) dari rancangan sistem yang dibuat diterjemahkan ke dalam bahasa pemrograman.

5. Penyerahan sistem/perangkat lunak ke pengguna, pengiriman, dan umpan balik

   Pada tahap ini dilakukan evaluasi guna mengidentifikasi kelemahan dan kekurangan sistem. Keseluruhan sistem diuji dengan detail untuk pengecekan kinerjanya. Setelah itu, pengguna memberikan umpan balik yang akan digunakan untuk memperbaiki kebutuhan sistem.

B. Alat Bantu atau *Tools*

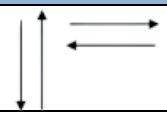
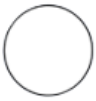
1. *Flowchart*

Flowchart adalah gambaran langkah-langkah dan urutan prosedur dari suatu aplikasi, yang dikembangkan dalam bentuk *diagram* untuk memudahkan pemahaman [16]. *Flowchart* membantu dalam memahami, merancang, dan mengkomunikasikan aliran kerja atau prosedur dalam suatu sistem atau proses. *Flowchart* berisi simbol-simbol yang menunjukkan alur instruksi sistem yang berjalan berurutan [17]. Berikut ini adalah notasi atau simbol –simbol dari 3 (tiga) kelompok *flowchart* sebagai berikut [18].

a) *Flow Direction Symbols* (Simbol Penghubung/alur)

Simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan yang lainnya. Simbol ini juga disebut *connecting line*, simbol tersebut adalah sebagai berikut.

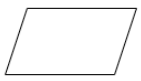
Tabel 2.1 Simbol Penghubung

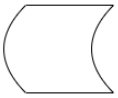
No	Simbol	Nama	Keterangan
1		Arus/ *Flow*	Untuk menyatakan jalannya arus suatu proses.
2		*Communication Link*	Untuk menyatakan bahwa adanya transisi suatu data atau informasi dari suatu lokasi ke lokasi lainnya.
3		*Connector*	Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman/lembaran sama.
4		*Offline Connector*	Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman atau lembaran yang berbeda.

b) *Input/Output Symbols* (Simbol *Input-Output*)

Simbol yang menunjukkan jenis peralatan yang digunakan sebagai media *input* atau *output*. Berikut beberapa simbol yang termasuk dapat dilihat pada Tabel 2.2

Tabel 2.2 Simbol *Input* dan *Output*

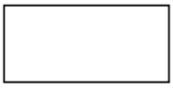
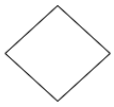
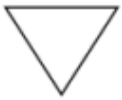
No	Simbol	Nama	Keterangan
1		*Input / Output*	Untuk menyatakan proses *input* dan *output* tanpa tergantung dengan jenis

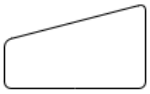
			peralatannya.
2		*Punched Card*	Untuk menyatakan *input* berasal dari kartu atau *output* ditulis ke kartu.
3		*Disk Storage*	Untuk menyatakan *input* berasal dari *disk* atau *output* disimpan ke *disk.*
4		*Magnetic Tape*	Untuk menyatakan *input* berasal dari pita magnetis atau *output* disimpan ke pita magnetis.
5		*Document*	Untuk menyatakan dokumen.
6		*Display*	Untuk menyatakan peralatan *output* yang digunakan yaitu layar, printer, dan sebagainya.

c) *Processing Symbols* (Simbol Proses)

Simbol yang menunjukkan langkah-langkah atau tindakan yang diambil dalam suatu proses. Berikut beberapa simbol proses dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol Proses

No	Simbol	Nama	Keterangan
1		Proses	Sebuah fungsi pemrosesan yang dilaksanakan oleh komputer biasanya menghasilkan perubahan terhadap data atau informasi.
2		*Manual Operation*	Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual).
3		*Decision*/Logika	Untuk menunjukkan suatu kondisi tertentu dengan dua kemungkinan, Ya/Tidak.
4		*Predefined Process*	Untuk menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.
5		Terminal	Untuk meyatakan permulaan atau akhir suatu program.
6		*Offline Storage*	Untuk menunjukkan bahwa data dalam simbol ini akan disimpan ke suatu media tertentu.

7		*Manual Input*	Simbol untuk pemasukan data secara manual *online keyboard.*

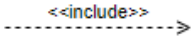
2. *Unified Modeling Language* (UML)

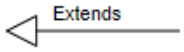
Unified Modeling Language (UML*)* adalah salah satu bahasa standar yang sering diterapkan di dunia industri untuk mendefinisikan kebutuhan, menganalisis dan merancang, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. *Unified Modeling Language* (UML*)* merupakan metodologi dalam mengembangkan sistem *Unified Modeling Language* (UML) adalah bahasa standar yang digunakan untuk menggambarkan, merancang, dan mendokumentasikan sistem perangkat lunak. Secara khusus *Unified Modeling Language* (UML) mendeskripsikan langkah-langkah penting dalam pengembangan keputusan analisa, perancangan, serta implementasi dalam sistem perangkat lunak [19]. Berikut penjelasan mengenai beberapa diagram yang ada di UML [20].

a) *Use Case Diagram*

Use case diagram merupakan metode pemodelan yang menggambarkan bagaimana satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengidentifikasi fungsi-fungsi yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut [21]. Berikut adalah simbol-simbol yang ada pada *use case diagram* dapat dilihat pada Tabel 2.4.

Tabel 2.4 *Use Case Diagram*

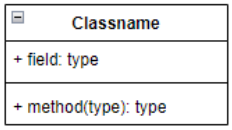
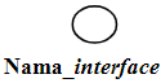
No	Simbol	Nama	Keterangan
1		*Actor*	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan *use case.*
2		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor.*
3		*Association* / Asosiasi	Menghubungkan *actor* untuk berinteraksi dengan *use case.*
4		*Include*	Menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya.

5		*Extends*	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi.

b) *Class Diagram*

Class diagram adalah visualisasi dari struktur tabel yang akan dibuat dalam suatu sistem. Diagram ini digunakan untuk menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang diperlukan untuk membangun sistem tersebut. Salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk memodelkan struktur statis dari sistem perangkat lunak. Diagram ini menunjukkan kelas-kelas yang ada dalam sistem beserta atribut, metode (operasi), dan hubungan antar kelas tersebut. *Class diagram* membantu dalam memahami dan mendesain struktur internal dari sistem yang sedang dikembangkan [22]. Berikut adalah simbol-simbol yang ada pada *class diagram* dapat dilihat pada Tabel 2.5.

Tabel 2.5 *Class Diagram*

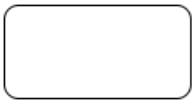
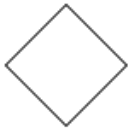
No	Simbol	Nama	Keterangan
1		*Class*	Kelas pada struktur sistem.
2		*Generalization*	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
3		*Interface*/antarmuka	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
4		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
5		*Aggregation*/agregasi	Relasi antar kelas dengan makna semua bagian.
6		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*.

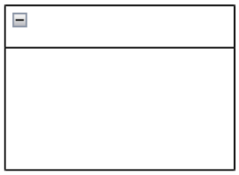
7		*Directed association*/asosiasi berarah	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.

c) *Activity Diagram*

Activity diagram adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk memodelkan alur kerja atau aktivitas dalam sistem. Diagram ini menggambarkan urutan aktivitas, keputusan, dan aliran kontrol dalam sebuah proses atau sistem [23]. *Activity diagram* adalah pemodelan yang menggambarkan cara kerja suatu sistem, sebuah *activity diagram* digambarkan dengan sebuah alur secara terstruktur proses kerja dari *use case* yang sedang diproses dari titik awal sampai titik akhir. Setiap aktivitas direpresentasikan dengan notasi-notasi yang sesuai fungsinya [24]. Berikut adalah simbol-simbol yang ada pada *activity diagram* dapat dilihat pada Tabel 2.6.

Tabel 2.6 *Simbol Activity Diagram*

No	Simbol	Nama	Keterangan
1		*Initial Node*	Digunakan untuk mewakili titik awal atau keadaan awal suatu aktivitas.
2		*Activity*	Digunakan untuk mewakili kegiatan proses.
3		*Control Flow*	Digunakan untuk merepresentasikan aliran kontrol dari satu aksi ke aksi lainnya.
4		*Activity Final Node*	Digunakan untuk menandai akhir dari semua aliran kontrol dalam aktivitas.
5		*Decision Node*	Menunjukkan percabangan aliran berdasarkan kondisi tertentu.

6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.1.6 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek (OOP) adalah pendekatan baru dalam pemikiran untuk menangani masalah-masalah yang akan dicoba atasi dengan bantuan komputer, dimana setiap objek sebagai entitas tunggal yang memiliki kombinasi struktur data dan fungsi tertentu [25]. OOP (*Object Oriented Programming*) atau pemrograman berorientasi objek adalah paradigma pemrograman yang menggunakan konsep-konsep pemrograman berbasis objek untuk merancang dan mengembangkan perangkat lunak. Paradigma ini berfokus pada representasi objek-objek dalam sistem perangkat lunak dan interaksi antara objek-objek tersebut. OOP sering digunakan dalam pengembangan berbagai jenis perangkat lunak, mulai dari aplikasi *desktop* hingga pengembangan web dan pemrograman perangkat seluler [26]. Contoh bahasa pemrograman yang mendukung pemrograman berorientasi objek termasuk *Java, Python, C++,* dan banyak lagi. Dalam pemrograman berorientasi objek memiliki beberapa konsep dasar antara lain [27]:

a. Kelas *(Class)*

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Sebuah kelas mempunyai sifat (atribut), kelakuan (operasi/*method*), hubungan (*relationship*), dan arti. Suatu kelas dapat diturunkan kekelas yang lain, dimana atribut dan kelas semula dapat diwariskan kekelas yang baru.

b. Objek (*Object*)

Merupakan perwujudan dari kelas, setiap objek akan mempunyai atribut dan *method* yang dimiliki oleh classnya, contohnya: amir, ahmad, yani merupakan object dari class manusia. Setiap objek dapat berinteraksi dengan objek lainnya meskipun berasal dari kelas yang berbeda. Secara sederhananya, dapat dikatakan terdiri dari properti (atribut) dan *method*.

c. Metode (*Method*)

Method adalah fungsi atau prosedur yang dibuat oleh seseorang *programmer* didalam suatu *class*. Pada sebuah *method* di dalam sebuah kelas juga memiliki izin akses seperti atribut, yaitu *private, public,* dan *protected*. Sebuah kelas boleh memiliki lebih dari satu *method* dengan nama yang sama asalkan memiliki parameter masukan yang berbeda. Hal ini memungkinkan compiler atau interpreter dapat mengenali *method* mana yang dipanggil.

d. Atribut (*Attribute*)

Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya. Atribut dapat memiliki hak akses *private*, *public* maupun *protected*. Sebuah atribut yang dinyatakan sebagai private hanya dapat diakses secara langsung oleh kelas yang membungkusnya, sedangkan kelas lainnya tidak dapat mengakses atribut ini secara langsung.

e. Abstraksi (*Abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang komplek menjadi suatu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapsulasi (*Encapsulation*)

Pembungkusan atribut data dan layanan (operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan (*Inheritance*)

Merupakan konsep mewariskan *attribute* dan *method* yang dimiliki oleh sebuah *class* kepada *class* turunannya. Dengan konsep ini *class* yang dibuat cukup mendefinisikan *attribute* dan *method* yang spesifik didalamnya, sedangkan *attribute* dan *method* yang lebih umum akan didapatkan dari *class* yang menjadi induknya.

h. Polimorpisme (*Polymorphism*)

Merupakan konsep yang memungkinkan digunakannya suatu *interface* yang sama untuk memerintah suatu objek agar melakukan suatu tindakan yang mungkin secara prinsip sama tetapi secara proses berbeda.

2.1.7 *Website*

Website adalah kumpulan halaman web yang terdiri dari komponen atau sekumpulan komponen dari teks, gambar, suara, dan atau gabungan dari semuanya yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terhubung dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*) sehingga menjadi media informasi yang lebih menarik untuk dikunjungi [28].

2.1.8 *PHP*

PHP merupakan singkatan dari *Hypertext Preprocessor* yang digunakan sebagai bahasa *script server-side* dalam pengembangan web yang disisipkan dalam file HTML. Pemakaian bahasa *PHP* memungkinkan situs *website* menjadi dinamis sehingga menjadi lebih mudah dan efisien [29].

2.1.9 *Framework*

Framework adalah kerangka kerja perangkat lunak yang menyediakan struktur dan pondasi untuk pengembangan aplikasi. Berisi kumpulan instruksi-instruksi yang dikumpulkan dalam *class* dan *function-function* dengan fungsi masing-masing untuk memudahkan *developer* dalam memanggilnya tanpa harus menuliskan *syntax* program yang sama berulang-ulang serta dapat menghemat waktu [30].

a. *Model-View-Controller* (MVC)

Model-View-Controller (MVC) adalah arsitektur perangkat lunak atau pola desain yang membagi menjadi tiga komponen utama model, *view*, dan *controller*. Tujuannya adalah untuk mengenkapsulasi data serta pemrosesannya (*Model*), memisahkannya dari proses manipulasi (*Controller*), dan tampilan (*View*) untuk direpresentasikan dalam antarmuka pengguna. MVC adalah pendekatan yang populer dalam pembangunan aplikasi web karena membantu dalam mengorganisasi kode, memudahkan pemeliharaan, dan memungkinkan pengembangan yang lebih terstruktur [31].

b. *Laravel*

Laravel adalah *framework* PHP dengan kode terbuka (*open source*) dengan desain MVC (*Model-View-Controller*) yang digunakan untuk membangun aplikasi *website*. *Framework* ini pertama kali dibangun oleh Taylor Otwell pada tanggal 22 Februari 2012 [32].

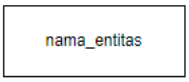
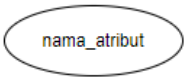
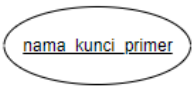
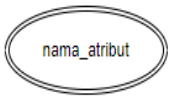
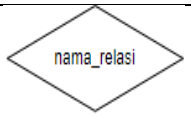
2.1.10 *Database*

Database adalah sekumpulan tabel yang berhubungan dan terorganisasi atau sekumpulan *record-record* yang dapat menyimpan data dan hubungan diantaranya. *Database* menyediakan cara efisien untuk menyimpan, mengelola, dan mengakses data. Dengan menggunakan *database*, informasi dapat diorganisir secara terstruktur dan diakses dengan cepat dan akurat [33].

A. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) adalah diagram berbentuk notasi grafis yang digunakan dalam pemodelan data untuk menggambarkan hubungan antara entitas dalam suatu sistem. ERD membantu pengembang sistem untuk memahami struktur dan hubungan dalam suatu sistem informasi, yang kemudian dapat digunakan sebagai dasar untuk merancang dan mengimplementasikan basis data [34]. Simbol ERD dapat dilihat pada Tabel 2.7.

Tabel 2.7 Simbol *Entity Relationship Diagram*

No	Simbol	Nama	Keterangan
1		*Entity*/entitas	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan merupakan nama tabel.
2		Atribut	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
3		Atribut kunci primer	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa *id*; kunci primer dapat lebih dari satu kolom; asalkan dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).
4		Atribut multi nilai/*multivalue*	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih.
5		Relasi	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja.
6		Asosiasi/*Association*	Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan *one to many* yang menghubungkan entitas A dan entitas B.

B. *Database Management System* (DBSM)

Database management system atau DBMS merupakan perangkat lunak yang dirancang untuk menyediakan cara efisien untuk membuat, mengelola, dan mengakses basis data. DBMS berfungsi sebagai perantara antara pengguna dan basis data, menyediakan antarmuka untuk memasukkan, memperbarui, dan mengambil data, serta memungkinkan pengguna untuk mengatur dan mengorganisir data dengan efisien [35].

1. *Data Definition Language* (DDL)

Data Definition Language (DDL) adalah kumpulan beberapa perintah *SQL* yang digunakan untuk mendefinisikan dan mengelola struktur objek dalam *database*. Dengan DDL, pengguna dapat membuat, mengubah, dan menghapus objek database seperti tabel dan isi tabel [36]. Perintah-perintah DDL memungkinkan pengguna untuk merancang dan mengelola struktur *database* agar sesuai dengan kebutuhan kebutuhan aplikasi atau sistem yang akan diimplementasikan. Berikut contoh *script* DDL.

a. *Create,* digunakan untuk membuat *database* dan tabel baru dalam *database*. Dibawah ini contoh *query* membuat *database* dan tabel : *CREATE TABLE* siswa*;*
b. *Alter,* digunakan untuk mengubah struktur tabel yang ada. Contoh *query* untuk menambah *field* pada tabel di *database* : *ALTER TABLE siswa ADD* hobi *VARCHAR (20);*
c. *Drop,* digunakan untuk menghapus *database* dan tabel, .Contoh *query* untuk menghapus *table*: *DROP TABLE pembayaran;*

2. *Data Manipulation Language* (DML)

Data Manipulation Language (DML) merupakan *database* yang digunakan untuk melakukan modifikasi dan pengambilan data pada suatu *database*. Pengolahan atau modifikasi ini meliputi:

a. *Insert* : Digunakan untuk menyisipkan atau memasukan data dalam tabel.
b. *Select* : Digunakan untuk mengambil data atau menampilkan data dari suatu tabel atau beberapa tabel
c. *Update* : Digunakan untuk memperbarui isi data dalam tabel.
d. *Delete* : Digunakan untuk menghapus data dari tabel

2.1.11 Uang Sekolah

Uang sekolah adalah sejumlah uang yang dibayarkan oleh siswa atau orang tua/wali siswa sebagai biaya untuk membiayai pendidikan di suatu institusi pendidikan. Uang sekolah ini dapat mencakup berbagai komponen, termasuk biaya pendidikan, biaya administrasi, pembelian buku atau perlengkapan pendidikan dan biaya lainnya. Besar uang sekolah biasanya ditentukan oleh sekolah itu sendiri berdasarkan program dan fasilitas yang disediakan, serta kebijakan pemerintah terkait biaya pendidikan . Uang sekolah dapat berbeda-beda tergantung pada jenis sekolah (negeri atau swasta), jenjang pendidikan, dan berbagai faktor lainnya.

2.1.12 *Midtrans*

Midtrans merupakan sebuah perusahaan layanan pembayaran *online* yang ada di Indonesia. Didirikan pada tahun 2012, *Midtrans* menyediakan solusi pembayaran *online* untuk bisnis *e-commerce*, aplikasi *mobile*, dan usaha kecil-menengah [37]. Layanan *midtrans* mencakup berbagai metode pembayaran seperti kartu *kredit*, *transfer bank*, pembayaran melalui *minimarket* atau *retail partner*, *e-wallet*, dan metode pembayaran lainnya. *Midtrans* juga menyediakan fitur keamanan untuk melindungi transaksi pembayaran, termasuk enkripsi data, verifikasi dua faktor, dan pemantauan transaksi secara *real-time.*

2.1.13 Notifikasi *WhatsApp*

Whatsapp adalah aplikasi pesan instan yang memungkinkan pengguna untuk mengirim pesan teks, berbagai gambar, dokumen, audio, melakukan panggilan suara, dan lain sebagainya dengan mudah melalui koneksi internet. Notifikasi *Whatsapp* merujuk pada pemberitahuan yang dikirimkan melalui aplikasi *Whatsapp* untuk memberitahu orang lain tentang suatu informasi [38].