BAB II
LANDASAN TEORI

1.1 Konsep Dasar Sistem Informasi

Konsep dasar sistem informasi membahas tentang pengertian dasar sistem infromasi yaitu pengertian sistem , pengertian informasi, pengertian sistem informasi , dan komponen-komponen sistem informasi. Penjelasannya adalah sebagai berikut :

1.1.1 Pengertian Sistem

Sistem adalah kumpulan elemen yang saling berhubungan dan berinteraksi dalam satu kesatuan untuk menjalankan suatu proses pencapaian suatu tujuan utama (Khasanah, 2018).

1.1.2 Pengertian Informasi

Informasi adalah data yang diolah menjadi yang lebih berguna dan lebih berarti bagi yang menerimanya (Jogianto, 2009).

1.1.3 Pengertian Sistem Informasi

Sistem informasi (*information system*)merupakan kombinasi teratur dari orang-orang, perangkat keras (*hardware*), perangkat lunak (*software*), jaringan komunikasi dan sumber daya data yang mengumpulkan, mengubah dan menyebarkan informasi dalam sebuah organisasi (Uus Rusmawan, 2019).

1.1.4 Komponen-Komopnen Sistem Informasi

Sistem Informasi memiliki fungsi untuk mengolah data dan mencapai tujuannya yaitu menghasilkan informasi yang relevan, tepat waktu dan akurat. Oleh karena itu sistem informasi memiliki enam buah komponen. Komponen-komponen sistem informasi adalah sebagai berikut :

1. *Input*

Input merupakan data yang masuk ke dalam sistem informasi. Komponen ini perlu ada karena merupakan bahan dasar pengolahan informasi. Sistem informasi tidak akan dapat menghasilkan jika tidak mempunyai komponen input.

2. Model

Informasi yang dihasilkan oleh sistem informasi berasal dari data yang diambil dari basis data yang diolah lewat suatu model-model tertentu. Komponen Output Output merupakan komponen yang harus ada di sistem informasi yang berguna bagi para pemakainya. Output dari sistem informasi dibuat dengan menggunakan data yang ada dibasis data dan diproses menggunakan model tertentu.

3. Teknologi

Teknologi merupakan komponen yang penting di sistem informasi. Tanpa adanya teknologi yang mendukung, maka sistem informasi tidak akan dapat menghasilkan informasi dengan tepat waktu.

4. Basis Data (*Database*)

Basis data adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya, yang tersimpan diperangkat keras komputer dan digunakan diperangkat lunak untuk memanipulasinya.

5. Komponen Pengendalian

Komponen pengendalian digunakan untuk menjamin bahwa informasi yang dihasilkan oleh sistem informasi merupakan informasi yang akurat (Sakti, 2016).

1.2 Konsep Basis Data

Konsep basis data membahas tentang pengertian basis data, MySQL, normalisasi , ERD (*Entity Relationship Diagram*) , relasi , *primary key* , dan *foreign key* . Berikut ini adalah penjelasan dari konsep basis data tersebut :

1.2.1 Database

Basis Data (*database*) adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (*query*) basis data disebut sistem manajemen basis data (*database management system*, DBMS). Sistem basis data dipelajari dalam ilmu informasi (Nahlah dan Amirudin, 2015).

Database merupakan kumpulan dari beberapa tabel dan di dalam tabel tersebut terdapat baris dan kolom. Baris dalam *database* sering disebut dengan istilah *record* atau *row*, sedangkan kolom sering disebut juga dengan istilah *field* atau *coloumn*. Dalam database terdapat sebutan-sebutan untuk satuan data yaitu :

1. Karakter, ini adalah satuan kerja data terkecil. Data terkecil atas susunan karakter yang pada akhirnya mewakili data yang memiliki arti dari sebuah fakta.
2. *Field*, adalah kumpulan dari karakter yang mewakili fakta tertentu.
3. *Record*, adalah kumpulan dari *field*.
4. Tabel, adalah sekumpulan dari *record-record* yang mewakili kesamaan *entity* dalam dunia nyata. Kumpulan dari tabel adalah *database*, wujud fisik sebuah *database* dalam komputer adalah *file* yang di dalamnya terdapat berbagai tingkatan data.
5. *File*, adalah bentuk fisik dari penyimpanan data. *File database* berisi semua data yang telah disusun dan diorganisasikan sedemikian rupa sehingga memudahkan pemberian informasi (Rosa dan Shalahuddin, 2015).

   Berikut ini merupakan beberapa tipe-tipe data *database.*

Tabel 2.1 Tipe-tipe data (Robi'in, 2002)

Nama tipe data	**Ukuran**	**Range**	**Deskripsi**
BLOB		-	Tipe data yang menyimpan bilangan berbentuk binary. Gambar yang biasa tersimpan di komputer sebenarnya memiliki nilai-nilai binary kemudian dapat ditampung pada tipe data ini. Tidak hanya gambar, *file* music, video, document dan lainnya juga dapat disimpan.
CHAR(n)	*n* karakter	1 – 32.767 byte	Digunakan untuk teks bertipe karakter atau string.
DATE	64 bits	1 Jan 100 – 29 Feb 32768	Juga menyediakan informasi waktu.

DECIMAL	Variabel	*Precision* = 1 - 15 yang menentukan besar minimal digit dari bilangan yang akan disimpan. Sedangkan scale = 1 - 15 yang menentukan besar digit untuk desimalnya harus kurang dari atau sama dengan *precision*.	Nomor yang mempunyai nilai desimalnya yang terdiri dari precision dan scale, sebagai contoh DECIMAL (10,3).
INTEGER	32 bit	-2.147.483.648 sampai 2.147.483.647	Biasanya digunakan untuk bilangan bulat yang panjang. (signed long = longword)

1.2.1.1 Operasi Dasar Basis Data

Dalam sebuah disk, sebuah basis data dapat dimanipulasi. Adapaun oeprasi-operasi dasar yang dapat dilakukan terhadaa sebuah basis data, meliputi ;

1. Pembuatan basis data baru (*create basis data*)
2. Penghapusan basis data (*drop basis data*)
3. Pembuatan file/tabel baru kesuatu basis data (create basis data)
4. Penghapusan file/tabel baru ke suatu basis data (drop tabel)
5. Penambahan/pengisian data baru ke sebuah file/tabel disebuah basis data (*insert)*
6. Pengambilan data dari sebuah file/tabel (*retrieve/search*)
7. Pengubahan data dari sebuah file/tabel (*update*)
8. Penghapuskan data dari sebuah file/tabel (*delete*) (Dantes, 2019).

1.2.1.2 Bahasa Dalam Basis Data

Bahasa basis data merupakan perantara bagi pemakai dengan basis data dalam berinterksi, yang telah ditetapkan oleh pembuat DBMS. Ada dua macam bahasa yang digunakan untuk mengelola dan mengorganisasikan data, yaitu :

1. Bahasa Defini Data (DDL/*Data deffinition leanguage*)
   Struktur/skema basis data yang menggambarkan/mewakili desain basis data secara keseluruhan dispesifikasikan dengan bahasa khusus yang disebut Data Definition Language (DDL). Contohnya, perintah CLEAR , ALTER , DROP , RENAME.
2. Bahasa Manipulasi Data (DML/*Data Manipulation Language*)
   DML merupakan bentuk bahasa basis data yang berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data. Contohnya adalah perintah SELECT, INSERT, UPDATE, dan DELETE (Dantes, 2019).

1.2.2 MySQL

MySQL adalah database yang cepat dan tangguh, sangaat cocok ika digabungkan dengan php, dengan database kita dapat menyimpan, mencari dan mengklasifikasikan data dengan lebih akurat dan profesional. MySQL menggunakan SQL language (Struktur *Query Language*) artinya MySQL menggunakan query atau bahasa pemrograman yang sudah standar didalam dunia database. MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.
(Fauzi, 2015)p

1.2.2.1 Tipe Data MySQL

Tipe-tipe data MySQL sebagai berikut (Hidayahtullah, 2017) .

1. Tipe Data Numerik

Tabel 2.2 Tipe Data Numerik

Tipe Data	Deskripsi
INT	Nilai Integer yang bias bertanda atau tidak. Jika Bertanda, maka rentang yang diperbolehkan adalah -2147483648 sampai 2147483647, sedangkan jika tidak bertanda maka rentangnya dari 0 sampai 4294967295
TINY INT	Nilai integer yang sangat kecil. Rentangnya 128-127 untuk yang bertanda dan 0-255 untuk yang tidak bertanda.
SMALLINT	Nilai integer yang sangat kecil dengan rentang 31768 sampai 32767 untuk yang bertanda sedangkan untuk yang tidak bertanda dari 0-65535.
MEDIUMINT	Integer dengan ukuran sedang dengan rentang -8388608 sampai 8388607a atau 0 sampai 16777215.
BIGINT	Integer dengan ukuran besar dengan rentang 9223372036854775808 sampai 92233722036854775807 atau 0 sampai 18446744073709551615
FLOAT(M,D)	Bilangan pecahan dengan panjang (termasuk jumlah decimal) M dan jumlah decimal D. Presisi desimalnya bisa sampai 24 digit. Defaultnya float (10,2). Bilangan float selalu bisa bertanda
DOUBLE(M,D)	Adalah bilangan pecahan dengan presisi dua kali lipat. Panjang (termasuk jumlah decimal M dan jumlah decimal D. Presisi desimalnya bisa sampai 53 digit. Defaultnya Double(16,4). Bilangan float selalu bisa bertanda, Sinonim dari DOUBLE and REAL
DECIMAL(M,D)	Adalah bilangan pecahan dan harus didefinisikan M dan D-nya. Setiap decimal membutuhkan tempat 1 byte. Sinonim dari DECIMAL dan NUMERIC.

2. Tipe Data String

Berikut ini adalah tipe data String yang paling umum di dalam MySQL.

Tabel 2.3 Tipe Data String

Tipe Data	Deskripsi
CHAR(M)	String dengan ukuran tetap. Ukurannya antara 1 sampai 255 karakter. Ukuran ditentukan dengan nilai M. Contoh: CHAR (6)
VARCHAR(M)	String dengan ukuran bervariasi anatar 1 sampai dengan 255 karakter. Contoh: VARCHAR(25)
TEXT	String dengan ukuran maksimum 65535 karakter. String yang tersimpan di dalam TEXT dianggap tidak case sensitive. Untuk kapasitas yang lebih kecil bisa menggunakan TINYTEXT dengan kapasitas maksimal 255 karakter. Sedangkan untuk kapasitas yang lebih besar bisa menggunakan MEDIUMTEXT (maksimal 16777215 karakter) dan LONGTEXT (maksimal 4294967295 karakter).
BLOB	Binary Large Object(BLOB) adalah tipe data untuk menyimpan data binary dalam jumlah besar. Biasanya digunakan untuk menyimpan citra. Untuk menyimpan data yang lebih kecil bisa menggunakan TINYBLOB (maksimal 255 karakter) sedangkan untuk kapasitas yang lebih besar bisa menggunakan MEDIUMBLOB (maksimal 16777215 karakter) dan LONGBLOB (maksimal 4294967295 karakter).
ENUM	Enumerasi atau sebuah list (daftar). Jadi misalnya anda ingin bahwa sebuah nilai terbatas hanya boleh dengan nilai tertentu saja maka anda bisa embuat sebuah daftar. Misalnya saja nilai hanya bisa menjadi ENUM (‘A’,’B’,’C’,’D’,’E’).

3. Tipe Data Tanggal dan Waktu

Berikut ini adalah tipe data tanggal dan waktu dalam MySQL :

Tabel 2. 4 Tipe Data Tanggal dan Waktu

Tipe Data	Deskripsi
DATE	Adalah tipe data tanggal dengan format YYYY-MM-DD antara 1000-01-01 and 9999-12-31 Contoh: 17 Agustus 1945 akan di simpan sebagai 1945-08-17
DATETIME	Adalah kombinasi tanggal dan wajtu dengan format YYYY-MM-DD HH:SS dan rentang data antara 1000- 01-01 00-00-00 sampai dengan 9999-12-31 23-59-59. Contoh: Pukul 10:00 pagi pada tanggal 17 Agustus 1945 akan disimpan sebagai 1945-08-17 10:00:00
TIMESTAMP	Sebuah penanda waktu antara 1 Januari 1970 tengah malam sampai dengan tahun 2037. Formatnya mirip dengan DATETIME tetapi tanpa pembatas di antara angkanya. Contoh: pukul 10:00 pagi pada tanggal 17 Agustus 1945 akan disimpan sebagai 19450817000000
TIME	Menyimpan waktu dalam format HH:MM:SS contoh pukul 10:00 akan disimpan menjadi 10:00:00
YEAR(M)	Menyimpan data tahun dalam format 2 dan 4 digit. Jika M diisi dengan nilai 2, maka rentang tahunnya dari 1970 – 2069 sedangkan jika M diisi dengan nilai 4 maka YEAR bisa bernilai 1901 sampai dengan 2155. Default nilai M adalah 4

1.2.3 Normalisasi

Normalisasi pada basis data merupakan proses pengelompokan data elemen menjadi tabel-tabel yang menunjukan entity dan relasinya. (Syukuro dan Hasan, 2015) Bentuk-bentuk normalisasi adalah sebagai berikut:

1. Bentuk tidak normal (*Unnormalized Form*)

   Bentuk ini merupakan kumpulan data yang akan disimpan, tidak ada keharusan mengikuti suatu format tertentu, dapat saja data tidak lengkap atau terduplikasi dan data dikumpulkan apa adanya. Berikut contohnya:

Tabel 2.5 Contoh Unnormalized Form

No. Faktur	Tgl	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	Mega	B-001	Gula	12.000	1
				B-002	Kopi	15.000	1
F-002	13/12/2016	P-002	Cindy	B-002	Kopi	15.000	1
				B-003	Teh	7.000	2

2. Bentuk normal pertama (1NF atau *First Normal Form*)

   Suatu tabel dikatakan dalam bentuk normal pertama (1NF) bila setiap kolom bernilai tunggal dalam satu baris (*record*). Tiap *field* hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti ganda. Berikut contohnya:

Tabel 2.6 Contoh First Normal Form

No. Faktur	Tgl	Kode Pelanggan	Nama	Kode Barang	Nama Barang	Harga	Qty
F-001	12/12/2016	P-001	Mega	B-001	Gula	12.000	1
F-001	12/12/2016	P-001	Mega	B-002	Kopi	15.000	1
F-002	13/12/2016	P-002	Cindy	B-002	Kopi	15.000	1
F-002	13/12/2016	P-002	Cindy	B-003	Teh	7.000	2

Bentuk normalisasi pertama (1 NF) ini mempunyai ciri yaitu setiap data dibentuk *file* datar atau rata (*flat file*), data dibentuk dalam satu record dan nilai-nilai dari *field-field* berupa nilai yang tidak dapat dibagi-bagi lagi

3. Bentuk normal kedua (2NF atau *Second Normal Form*).

Suatu tabel berada dalam bentuk normal kedua (2NF) jika tabel berada dalam bentuk normal pertama dan setiap atribut yang bukan kunci (non key) bergantung secara fungsional terhadap kunci utama (*Primary key*). Berikut contohnya:

Tabel 2.7 Transaksi (2NF)

No. Faktur	Tgl	Kode Pelanggan	Kode Barang	Qty
F-001	12/12/2019	P-001	B-001	1
F-001	12/12/2019	P-001	B-002	1
F-002	13/12/2019	P-002	B-002	1
F-002	13/12/2019	P-002	B-003	2

Tabel 2.9 Kode Pelanggan(2NF)

Kode Pelanggan	Nama
P-001	Mega
P-002	Cindy

Tabel 2.8 Kode Barang (2NF)

Kode Barang	Nama Barang	Harga
B-001	Gula	12.000
B-002	Kopi	15.000
B-003	Teh	7.000

4. Normal Ketiga (3NF atau *Third Normal Form*)

   Suatu tabel berada dalam bentuk normal ketiga (3NF) jika tabel berada dalam bentuk normal kedua, setiap atribut bukan kunci (*non key*) tidak memiliki ketergantungan secara transitif terhadap kunci utama (*primary key*), artinya setiap atribut bukan kunci harus bergantung hanya pada *primary key* secara keseluruhan. Berikut contohnya:

Tabel 2.11 Tabel Transaksi (3NF)

No. Faktur	Tgl	Kode Pelanggang
F-001	12/12/2016	P-001
F-002	12/12/2016	P-001
F-003	13/12/2016	P-002
F-004	13/12/2016	P-002

Tabel 2.10 Tabel Pelanggan (3NF)

Kode Pelanggan	Nama
P-001	Mega
P-002	Cindy

Tabel 2.13 Tabel Detail (3NF)

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 2.12 Tabel Barang (3NF)

Kode Barang	Nama Barang	Harga
B-001	Gula	12.000
B-002	Kopi	15.000
B-003	Teh	7.000

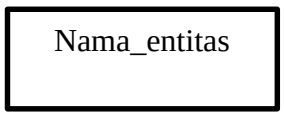
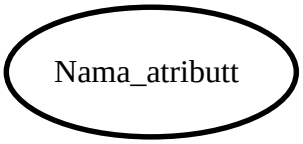
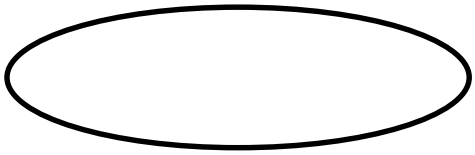
1.2.4 ERD (*Entity Relationship Diagram*)

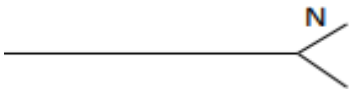
ERD (*Entity Relationship Diagram*) adalah sutau model untuk mejelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. ERD untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol.

ERD (*Entity Relationship Diagram)* dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data reasional.(Rosa dan Sahalahuddin, 2013 : 50).

Berikut ini adalah simbol dari ERD (*Entity Relationship Diagram)* (Rosa dan Sahalahuddin, 2013 : 50).

Tabel 2.14 Simbol ERD (*Entity Relationship Diagram*)

Simbol	Deskripsi
Entitas/*Entity* 	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 belum merupakan merupakan nama table
Atribut 	Field atau kolom data yang butuh disimpan dalam suatu entitas.
Atribut kunci primer 	field atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses recordyang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).
Atribut multinilai/*multivalue* 	field atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.

Relasi  	Relasi yang menghubungkan antar entitas;biasanya diawali dengan kata kerja.
Asosiasi/*association*  	penghubung antara relasi dan entitas di mana di kedua ujungnya memiliki multiplicity kemungkinan jumlah pemakaian kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan one to many menghubungkan entitas A dan B.

1.2.5 Relasi

Relasi antar tabel yaitu menghubungkan antar entitas–entitas yang ada dalam perancangan sistem. (Abdullah, 2015)

1.2.6 *Primary Key*

Primary key merupakan satu atau lebih nilai pada kolom untuk membuat sebuah record unik. *Primary key* adalah satu atribut yang bukan hanya mengidentifikasi secara unik suatu kejadian spesifik, tetapi juga dapat mewakili setiap kejadian dari suatu entitas(Nahrun & Rusmala, 2016).

1.2.7 *Foreign Key*

Foreign key adalah satu atribut atau satu set minimal atribut yang melengkapi satu hubungan yang menunjukkan ke induknya. *Foreign key* berguna untuk menjamin integritas referensial dan untuk memastikan nilai pada suatu kolom atau kumpulan kolom sesuai dengan nilai yang ada pada tabel lain (Nahrun & Rusmala,2016).

1.3 PHP *Framework*

PHP adalah bahasa scripting yang menyatu dengan HTML dan dijalankan pada serverside.Artinya semua sintaks yang diberikan akan sepenuhnya dijalankan pada server sedang yang dikirim ke browser hanya hasilnya saja. Ketika seorang pengguna internet membuka situs yang menggunkan fasilitas serverside scripting PHP, maka terlebihh dahulu server yang bersangkutan akan memproses semua perintah PHP diserver lalu mengirimkan hasilnya dalam format TML ke web server pengguna internet tadi. Sehingga kode asli yang ditulis dengan PHP tidak terlihat di browser pengguna. PHP berfungsi mengambil informasi dari form berbasis web dan menggunakannya untuk berbagai macam fungsi, sebagai bahasa untuk mengidentifikasi seberapa banyak pengunjung menggunakan bahasa PHP, Pengaturan layout dalam berbagai macam browser seperti Firefox, Bahasa Pemrogaman Php dalam web sangat luas, jadi semua tergantung pada anda, kalau anda sangat teliti anda akan tau fungsinya (Fauzi dkk , 2015).

1.3.1 *Codeigniter*

Codeigniter adalah sebah framework yang dibuat berdasarkan design pattern model view controller atau biasa disingkat MVC. Design Pattern adalah kumpulan penjelasan mengenai metode-metode bagaimana cara menyelesaikan suatu masalah yang umum ditemui dalam proses perancangan perangkat lunak (Software Design). Design pattern merupakan petunjuk bagaimana cara menyelesaikan suatu masalah di seputar dunia software design. (Suhartini, 2020).

1.3.2 *Laravel*

Laravel merupakan sebuah *framework* yang relatif baru, namun begitu cepat naik popularitasnya diantara komunitas PHP karena *framework* ini bersih, cepat dan mudah untuk digunakan. Laravel adalah *framework* PHP terbuka yang dirancang di atas arsitektur MVC. Laravel memiliki *syntax* yang ekspresif dan menyediakan modularisasi tingkat tinggi. *Framework* ini juga memiliki banyak fasilitas yang bertujuan untuk mengeliminasi prosedur rutin yang harus dikerjakan oleh pengembang, seperti *routing*, otentikasi, sesi dan *queries*. Laravel juga mengintegrasikan peralatan-peralatan dan pustaka lain untuk memudahkan pengembang.

Laravel yang diciptakan oleh Taylor Otwell merupakan *framework*, yang menggabungkan fitur-fitur terbaik dari *framework* yang ada seperti CodeIgniter, Symphony, Zend, dan lain-lain. Fitur-fitur yang ada di dalam Laravel seperti *routing, built-in authenthication, template engine* bernama Blade dan *fluent query builder*, memungkinkan pengembang untuk dengan mudah membangun sebuah *website* yang kokoh dan aman. Hal-hal tersebut sangat penting, karena keamanan merupakan hal yang penting dalam membangun sebuah *website* transaksional.

Back-end website yang dibangun akan sepenuhnya menggunakan Laravel, mulai dari *routing*, *processing*, transaksi hingga penampilan data. Untuk tampilan sebagian akan menggunakan *template engine* Laravel yaitu Blade (Hazmi, 2018).

1.3.3 *Yii*

Yii *Framework* atau lebih dikenal dengan sebutan Yii, merupakan kerangka kerja *open source* berbasis PHP. Pola desain yang dijalankan oleh Yii *Framework* mengadopsi konsep MVC (*Model – View – Controller*).

Yii merupakan salah satu dari sekian banyak *framework* PHP yang cukup populer dikalangan PHP *Developer*, Yii adalah salah satu dari sederetan PHP *Framework* yang bersifat *open source*. Berdasarkan situs resminya, Yii adalah *Framework* (kerangka kerja) PHP berbasis komponen, berkinerja tinggi untuk pengembangan aplikasi *web* berskala besar. Yii juga meyediakan *reusability* maksimum dalam pemrograman *web* dan mampu meningkatkan kecepatan pengembangan secara signifikan.

Dengan diterapkannya teknologi *framework* Yii pada aplikasi berbasis *web* dapat membantu didalam mengembangkan semua jenis aplikasi *web*. Dikarenakan *framework* Yii sangat ringan dan dilengkapi dengan mekanisme *caching* yang canggih, Yii sangat cocok untuk pengembangan aplikasi dengan lalu lintas-tinggi, seperti portal, forum, sistem manajemen konten (CMS), sistem *ecommerce*, dan lain-lain (Zakir, 2017).

1.3.4 Symfony

Symfony adalah sebuah framework lengkap yang didesain untuk mengoptimalkan pengembangan aplikasi berbasis web dengan memberikan beberapa fitur andalan.

Symponi mengelompokkan aturan-aturan bisnis aplikasi (*business rules*), logika server dan tampilan presentasi. Sympony menyediakan bermacam-macam alat dan kelas-kelas yang ditujukkan untuk memeprsek waku pengembangan sebuah aplikasi *web* yang komplek (Naista, 2017).

1.3.5 CakePHP

CakePHP merupakan sebuah *rapid development framework* yang gratis dan *open source* untuk PHP. CakePHP adalah sebuah *framework* atau kerangka kerja untuk membuat aplikasi CRUD (*Create, Read, Update, Delete*) berbasis bahasa pemrograman PHP. CakePHP juga menjadi salah satu *framework* pilihan yang memungkinkan seorang pengembang *web* untuk membuat sebuah aplikasi dengan karakter pengembangan RAD (*Rapid Application Development*) yang memungkinkan untuk digunakan dan dikembangkan menjadi aplikasi lain (Naista, 2017).

1.4 Konsep MVC (Model-*View-Controller*)

Model-*View-Controller* (MVC) adalah sebuah konsep yang diperkenalkan oleh penemu Smalltalk (Trygve Reenskaug) untuk meng-enkapsulasi data bersama dengan pemrosesan (model), mengisolasi dari proses manipulasi (*controller*) dan tampilan (*view*) untuk direpresentasikan pada sebuah user interface. MVC mengikuti pendekatan yang paling umum dari Layering. Layering hanyalah sebuah logika yang membagi kode kita ke dalam fungsi di kelas yang berbeda. Pendekatan ini mudah dikenal dan yang paling banyak diterima. Keuntungan utama dalam pendekatan ini adalah penggunaan ulang (*reusability*) kode.

1.4.1 Model

Model digunakan untuk mengelola informasi dan memberitahu pengamat ketika ada perubahan informasi. Hanya model yang mengandung data dan fungsi yang berhubungan dengan pemrosesan data. Sebuah model meringkas lebih dari sekedar data dan fungsi yang beroperasi di dalamnya. Pendekatan model yang digunakan untuk komputer model atau abstraksi dari beberapa proses dunia nyata. Hal ini tidak hanya menangkap keadaan proses atau sistem, tetapi bagaimana sistem bekerja. Sebagai contoh, programmer dapat menentukan model yang menjembatani komputasi back-end dengan front-end GUI (*graphical user interface*).

1.4.2 *View*

View bertanggung jawab untuk pemetaan grafis ke sebuah perangkat. View biasanya memiliki hubungan 1-1 dengan sebuah permukaan layar dan tahu bagaimana untuk membuatnya. View melekat pada model dan me-render isinya ke permukaan layar. Selain itu, ketika model berubah, view secara otomatis menggambar ulang bagian layar yang terkena perubahan untuk menunjukkan perubahan tersebut. Terdapat kemungkinan beberapa view pada model yang sama dan masing-masing view tersebut dapat merender isi model untuk permukaan tampilan yang berbeda

1.4.3 *Controller*

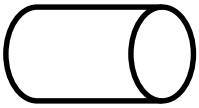
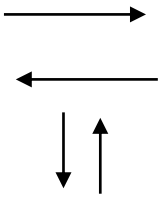
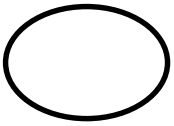
Controller berguna untuk menerima input dari pengguna dan menginstruksikan model dan view untuk melakukan aksi berdasarkan masukan tersebut. Sehingga, controller bertanggung jawab untuk pemetaan aksi pengguna akhir terhadap respon aplikasi. Sebagai contoh, ketika pengguna mengklik tombol atau memilih item menu, controller bertanggung jawab untuk menentukan bagaimana aplikasi seharusnya merespon (Abd. Rachman Dayat, 2017).

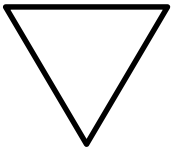
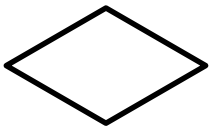
1.5 *Flowmap*

Flowmap adalah campuran peta dan *flowchart*, yang menunjukan pergerakan benda dari satu lokasi ke lokasi lain, seperti jumlah orang dalam migrasi, jumlah barang yang diperdagangkan, atau jumlah paket dalam jaringan. *Flowmap* menolong analisis dan *programmer* untuk memecahkan masalah ke dalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. Berikut simbol-simbol yang digunakan dalam membuat *flowmap* menurut (Purwanto, 2017)**.**

Tabel 2. 15 Simbol Flowmap

Simbol	Keterangan
 Dokumen	Menunjukkan dokumen *input* dan *output.*

 Kegiatan Manual	Menunjukan pekerjaan atau kegiatan manual.
 Proses	Menunjukan kegiatan proses dari operasi program komputer.
 *Keyboard*	Menunjukan *input* yang menggunakan *online keyboard.*
 *Storage Data*	Menunjukan tempat penyimpanan data pada operasi program komputer seperti : MySQL, Access, Oracle, dll.
 *Storage Data*	Menunjukan tempat penyimpanan data pada *hard disk* (*secondary storage*)
 Garis Alir	Menunjukkan arus dari setiap proses
 Penghubung	Menunjukkan penghubung ke halaman yang masih sama atau ke halaman lain.

 Arsip	Menunjukkan pengarispan *file* tanpa menggunakan komputer.
 Keputusan	Untuk menunjukan suatu kondisi tertentu, yang akan menghasilkan dua kemungkinan jawaban, ya atau tidak.
 Terminal	Untuk menyatakan permulaan atau akhir suatu program.

1.6 *Website*

Website merupakan sebuah kumpulan halaman-halaman web beserta file-file pendukungnya, seperti file gambar, video, dan file digital lainnya yang disimpan pada sebuah web server yang umumnya dapat diakses melalui internet. Atau dengan kata lain, website adalah sekumpulan folder dan file yang mengandung banyak perintah dan fungsi fungsi tertentu, seperti fungsi tampilan, fungsi menangani penyimpanan data, dan sebagainya (Suhartini dkk , 2020).

1.7 Metode Pengembangan Sistem

Metode adalah kesatuan metode-metode, prosedur-prosedur, konsep-konsep pekerjaan dan aturan-aturan yang digunakan oleh suatu ilmu pengetahuan, seni atau disiplin yang lainnya. Sedang metode adalah suatu cara, teknik yang sistematik untuk mengerjakan sesuatu. Metodologi pengembangan sistem berarti adalah metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan dan prosulat-prosulat yang akan digunakan untuk mengembangkan suatu sistem informasi.

Pengembangan sistem (*System Development)* dapat berarti menyusun suatu sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang telah ada. Ada beberapa hal yang menyebabkan perlunya perbaikan terhadap sistem lama, yaitu sebagai berikut:

1. Adanya permasalahan-permasalahan (*problems*) yang timbul di sistem yang lama, misalnya ketidakberesan sistem yang lama menyebabkan sistem lama tidak dapat beroperasi sesuai yang diharapkan, adanya pertumbuhan organisasi yang menyebabkan harus disusunnya sistem yang baru.
2. Untuk meraih kesempatan-kesempatan (*oppoetunities*) seperti peluang-peluang pasar, pelayanan yang mengikat kepada pelanggan.
3. Adanya instruksi-instruksi (*directives*) yang dimaksud adalah penyusunan sistem yang baru dapat juga terjadi karena adanya instruksi-instruksi dari pimpinan atas ataupun dari luar organisasi, misalnya peraturan pemerintah.

Dalam pengembangan sistem informasi, perlu digunakan suatu metodologi yang dapat digunakan sebagai pedoman bagaimana dan apa yang harus di kerjakan selama pengembangan sistem. Dengan mengikuti metode atau prosedur-prosedur yang diberikan oleh suatu metodologi, maka pengembangan sistem diharapkan dapat diselesaikan dengan berhasil. Urutan-urutan prosedur untuk pemecahan masalah ini dikenal dengan istilah algoritma (*algorithm*).

Dalam pengembangan sistem terdapat prinsip-prinsip pengembangan sistem, yaitu:

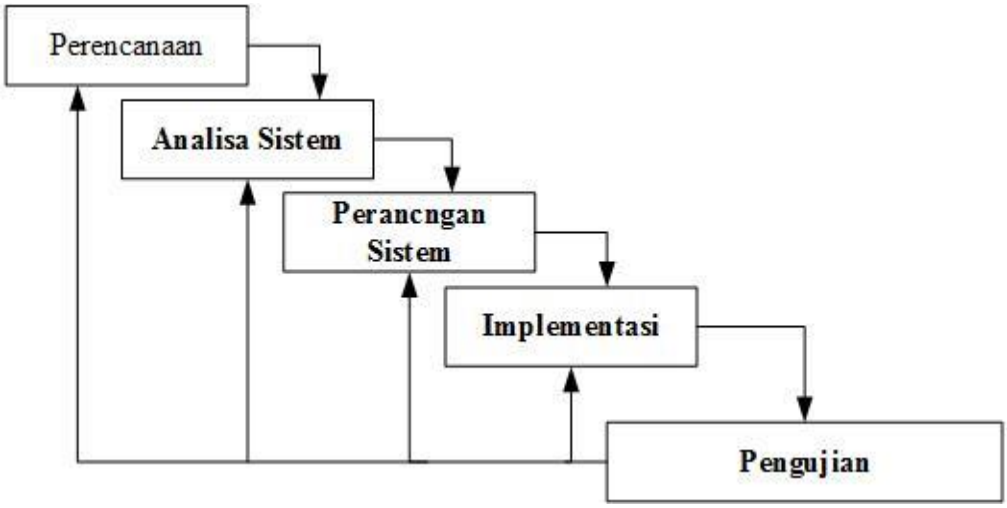
1. Sistem yang dikembangkan adalah untuk manajemen, maksudnya adalah setelah sistem selesai dikembangkan, maka yang akan menggunakan informasi dari sistem ini adalah manajemen, sehingga sistem harus dapat mendukung kebutuhan yang diperlukan oleh manajemen.
2. Sistem yang dikembangkan adala investasi modal yang besar.
3. Sistem yang dikembangkan memerlukan orang yang terdidik, manusia merupakan faktor utama yang menentukan berhasil tidaknya suatu sistem, baik dalam proses pengembangannya, penerapannya, maupun dalam proses operasinya.
4. Tahapan kerja dan tugas-tugas harus dilakukan dalam proses pengembangan sistem.
5. Proses pengembangan sistem tidak harus urut, misalnya tahapan dapat dilakukan secara bersama-sama.

6. Jangan takut membatalkan proyek, hal ini merupakan pantangan untuk membatalkan suatu proyek yang sedang berjalan, namun jika sudah tiak layak lagi maka proyek dapat dibatalkan tetapi harus dipertimbangkan dengan cermat.
7. Dokumentasi harus ada untuk pedoman dalam pengembangan sistem (Hartono, 2005)

1.7.1 Metode *Waterfall*

Metode *waterfall* sebagai metode pengembangan sistem. Model ini mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dalam tingkat kemajuan *system* pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model *waterfall* merupakan metode yang paling banyak digunakan dalam *software engineering*, karena pemodelan sistem terbagi menjadi tahapan-tahapan yang mengikuti pola teratur, seperti layaknya air terjun.

Tahapan-tahapan pada model *waterfall* dapat dilihat pada gambar berikut:



Gambar 2.1 Metode *Waterfall*

Berdasarkan model *waterfall*, garis besar penyelesaian masalah dalam pembuatan aplikasi ini terdapat 5 tahapan yang meliputi:

1. Tahap Perencanaan Sistem

   Pada tahap ini akan dilakukan pendefinisian seluruh kebutuhan perangkat lunak yang nantinya akan dijadikan sebagai SRS (*software Requirements Spesifications*).

2. Tahap Analisa Sistem

   Tahapan analisis terdiri atas analisis kebutuhan dan analisis pemodelan. Analisis kebutuhan merupakan pengidentifikasian kebutuhan yang diperlukan oleh sistem.

3. Tahap Perancangan

   Proses perancangan sistem membagi persyaratan dalam *system* perangkat keras atau perangkat lunak.

4. Tahap Implementasi

   Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program.

5. Tahap Pengujian Sistem.

   Tahap pengujian adalah proses eksekusi suatu program, bila pengujian dilakukan secara sukses (sesuai dengan sasaran tersebut) maka tidak akan ditemukan kesalahan di dalam perangkat lunak (Muslim, 2016).

1.7.2 Metode *Propotipe*

Motode *prototipe* merupakan suatu teknik untuk mengumpulkan informasi tertentu mengenai kebutuhan-kebutuhan informasi pengguna secara cepat. Berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan atau pemakai. Prototipe tersebut akan dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak (Susanto & Andriana, 2016).

Berikut ini penjelasan dari tahapan motode prototipe:

1. Mendengarkan pelanggan

   Pada tahap ini dilakukan pengumpulan kebutuhan dari sistem dengan cara mendengar keluhan dari pelanggan. Untuk membuat sistem yang sesuai kebutuhan dan harus diketahui terlebih dahulu bagaimana sistem yang sedang berjalan untuk kemudian mengetahui masalah yang terjadi.

2. Membangun/memperbaiki *Prototipe*

   Pada tahap ini, dilakukan perancangan dan membuat prototipe sistem. Prototipe yang dibuat disesuaikan dengan kebutuhan sistem yang telah didefinisikan sebelumnya dari keluhan pelanggan atau pengguna

3. Uji Coba

   Pada tahap ini, prototipe dari sistem di uji coba oleh pelanggan atau pengguna. Kemudian dilakukan evaluasi kekurangan-kekurangan dari kebutuhan pelanggan. Pengembangan kemudian kembali mendengarkan keluhan dari pelanggan untuk memperbaiki *prototipe* yang ada.

1.8 Metode Pemodelan Sistem

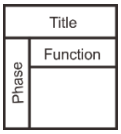
Unified Modeling Language (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1.8.1 *Activity Diagram*

Activity diagram adalah sesuatu yang menjelaskan tentang alir kegiatan dalam proses yang sedang dirancang, bagaimana proses awal berawal, keputusan yang mungkin terjadi, dan bagaiman sistem akan berakhir. *Activity* diagram juga dapat menjelaskan metode pararel yang mungkin terjadi pada beberapa eksekusi. Simbil-simbol *activity* diagram yaitu :

Tabel 2.16 Simbol *activity diagram*

Simbol	Deskripsi
	Status awal/*Start Point*, merupakan status awal aktivitas sistem
	Status Akhir/*End Point*, merupakan status akhir yang dilakukan sistem
	Aktivitas/*Activites*, menggambarkan suatu proses aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja

	Penggabungan/*Join* merupakan asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
	Percabangan/*Decision*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.* Jika ada pilihan aktivitas lebih dari satu
	*Swimlane*, memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

1.8.2 *Class Diagram*

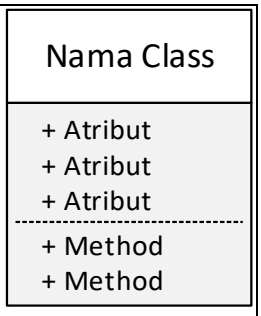
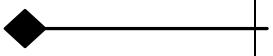
Class diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem/perangkat lunak yang digunakan. *Class diagram* memberi gambaran (diagram statis) tentang sistem/perangkat lunak dan relasi-relasi yang ada didalamnya. Sebuah *Class* memiliki tiga area pokok:

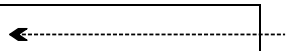
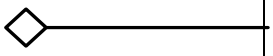
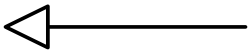
a. Nama, merupakan nama dari sebuah kelas
b. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari *class*
c. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*.

Berikut adalah notasi–notasi umum yang terdapat pada *class diagram* menurut (Kusdiwiardi, 2015) :

Tabel 2.17 Simbol *Class Diagram*

Nama	Deskripsi	Simbol
Class	Blok-blok pembangun pada pemrograman berorientasi obyek. Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah	

	mendefinisikan *property*/atribut *class*. Bagian akhir mendefinisikan method-method dari sebuah *class*.	
Association	Asosiasi merupakan sebuah relationship paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*. (Contoh: *One-to-one, one-to-many, many-to-many*).	
Composition	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.	
Dependency	*Class* menggunakan *class* yang lain. Penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency*	

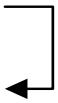
	dilambangkan sebagai sebuah panah bertitik-titik.	
Aggregation	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi.	
Generalization	Relasi *generalization* sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi obyek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "*parent*"-nya/induknya.	

1.8.3 *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirim dan diterima oleh objek. Simbol-simbol dan fungsi *Sequence Diagram* dapat dilihat pada tabel 2.3 menurut (Rosa dan Shalahuddin, 2015) berikut :

Tabel 2.18 Simbol *Sequence Diagram*

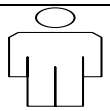
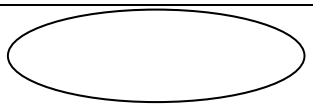
Simbol	Keterangan
 Aktor	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari *system*

 Garis Hidup	Menyatakan kehidupan suatu objek
nama objek : nama kelas Objek	Menyatakan objek yang berinteraksi pesan
 Waktu Aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi
<<create>>  Pesan tipe create	Menyatakan suatu objek membuat objek yang lain, arah panah mengarahkan pada objek yang dibuat
 Pesan tipe call	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
1 : masukan Pesan tipe send	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim
1 : keluaran Pesan tipe retrun	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu

1.8.4 *Usecase Diagram*

Use case Diagram adalah suatu urutan interaksi yang saling berkaitan antara sistem dan aktor. *Use case* dijalankan melalui cara menggambarkan tipe interkasi antara user suatu program (sistem) dengan sistemnya sendiri. *Use case* melalui sebuah cerita yang mana sebuah sistem itu dipakai. *Use case* juga dipakai untuk membentuk perilaku sistem yang akan dibuat. Sebuah *use case* menggambarkan sebuah interaksi antara pengguna (aktor) dengan sistem yang sudah ada.

Tabel 2.19 Simbol *Use Case* Diagram

Simbol	Keterangan
	*Actor* : Mewakili peran orang, sistem yang lain, atau alat ketika berkomuikasi dengan usecase
	*Use Case* : Abstraksi dan interaksi antara sistem dan aktor
	Association : Abstraksi dari penghubung antara aktor dengan usecase
	Generaliazi : Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*
	Extend : Menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika sutau kondisi terpenuhi
	Include : Menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalisasi dari use case lainnya

1.9 Metode Pengujian Sistem

Pengujian *software* sangat diperlukan untuk memastikan *software*/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji *software* harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau testing sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan *software* seperti halnya analisis, desain, dan pengkodean. Pengujian software haruslah dilakukan dalam proses rekayasa perangkat lunak atau *software engineering*. Sejumlah strategi pengujian *software* telah diusulkan dalam literatur. Semuanya menyediakan template untuk pengujian bagi pembuat *software*. Dalam hal ini, semuanya harus memiliki karakteristik umum berupa:

1. Testing dimulai pada level modul dan bekerja keluar ke arah integrasi pada sistem berbasiskan komputer .
2. Teknik testing yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya.
3. *Testing* diadakan oleh pembuat/pengembang *software* dan untuk proyek yang besar oleh group *testing* yang independen.
4. *Testing* dan *Debugging* adalah aktivitas yang berbeda tetapi *debugging* harus diakomodasikan pada setiap strategi *testing*

Ada beberapa jenis pengujian perangkat lunak, antara lain:

1. Pengujian *white box* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan petunjuk untuk mendapatkan program yang benar secara 100%.
2. *Black-Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program (Mustaqbal, Firdaus dan Rahmadi, 2015).

1.9.1 *White-Box Testing*

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau *software* dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Kalau modul yang telah dan sudah di hasilkan berupa *output* yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan di cek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan. Kasus yang sering menggunakan *white box testing* akan di uji dengan beberapa tahapan yaitu:

1. Pengujian seluruh keputusan yang menggunakan logikal.
2. Pengujian keseluruh *loop* yang ada sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

Kelebihan *White Box Test*ing antara lain:

1. Kesalahan Logika

   Menggunakan sintax '*if*' dan *sintax* pengulangan. Langkah selanjutnya metode *white box testing* ini akan mencari dan mendeteksi segala kondisi yang di percaya tidak sesuai dan mencari kapan suatu proses perulangan di akhiri.
2. Ketidaksesuaian Asumsi

   Menampilkan dan memonitor beberapa asumsi yang diyakini tidak sesuai dengan yang diharapkan atau yang akan diwujudkan, untuk selanjutnya akan dianalisa kembali dan kemudian diperbaiki.
3. Kesalahan Pengetikan

   Mendeteksi dan mencaribahasa-bahasa pemograman yang di anggap bersifat *case sensitif*.

Kelemahan *White Box Testing* adalah pada perangkat lunak yang jenisnya besar, metode *white box testing* ini dianggap boros karena melibatkan banyak sumber daya untuk melakukannya (Mustaqbal, Firdaus dan Rahmadi, 2015).

1.9.2 *Black-Box Testing*

Black Box Testing berfokus pada spesifikasi fungsional dari perangkat lunak. *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box Testing* bukanlah solusi alternatif

dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing. Black Box* Testing cenderung untuk menemukan hal-hal berikut:

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

Tidak seperti metode *whitebox* yang dilaksanakan diawal proses, ujicoba blackbox diaplikasikan dibeberapa tahapan berikutnya. Karena ujicoba blackbox dengan sengaja mengabaikan struktur kontrol, sehingga perhatiannya difokuskan pada informasi *domain*. Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada *input-input* tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Dengan mengaplikasikan ujicoba *blackbox*, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut :

1. Kasus uji yang berkurang, jika jumlahnya lebih dari 1, maka jumlah dari uji kasus tambahan harus di desain untuk mencapai ujicoba yang cukup beralasan.
2. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, daripada kesalahan yang terhubung hanya dengan suatu ujicoba yang spesifik (Mustaqbal, 2015).

1.10 Cuti

Cuti adalah hak setiap pekerja dalam setiap tahun kerja, biasanya hak cuti itu adalah selama dua belas hari kerja dan dalam kurun waktu tersebut pekerja yang

bersangkutan tetap mendapat gaji penuh dan waktu cuti itu diperhitungkan sebagai bagian masa aktif untuk perhitungan pensiun kelak.

Berdasarkan Undang-undang No.13 tahun 2003 Pasal 79 ayat (2) dan Pasal 84, hanya karyawan yang sudah bekerja minimal 12 bulan yang berhak mendapat cuti tahunan 12 hari. Karena itu, perusahaan berwenang untuk menolak permintaan cuti dan karyawan yang belum genap 1 tahun bekerja. Apabila perusahaan bersedia memberikan ijin, maka disebut sebagai : Cuti di Luar Tanggungan" dan perusahaan dapat memotong gaji pekerja tersebut secara prorata sesuai dengan jumlah ketidakhadirannya.

Peraturan Pemerintah Republik Indonesia Nomor 24 Tahun 1976 Peratuan Pemerintah Cuti Pegawai Negeri Sipil Bab 2 Bagian Pertama Tentang Jenis Cuti Pasal 3. Adapun jenis Cuti Pegawai Negeri Sipil sebagai berikut : Cuti Tahunan, Cuti Besar, Cuti Sakit, Cuti bersalin, Cuti Karena Alasan Penting, dan Cuti Diluar Tanggungan Negara. Adapun penjelasan dari jenis cuti tersebut .

1. Cuti Tahunan Setiap Pegawai Negeri Sipil

   Cuti tahunan pegawai negeri sipil yang telah bekerja sekurang kurangnya satu tahun secara terus menerus berhak atas cuti tahunan. Lamanya cuti tahunan adalah 12 (dua belas) hari kerja. Cuti tahunan tersebut dapat diambil secara terpecahpecah, dengan ketentuan setiap bagian tidak boleh kurang dari 3 (tiga) hari kerja. Cuti tahunan yang tidak diambil dalam tahun yang bersangkutan dapat diambil dalam tahun berikutnya untuk paling lama 18 (delapan belas) hari kerja termasuk cuti tahunan dalam tahun yang sedang berjalan. Cuti tahunan yang tidak diambil dalam kurun waktu 2 (dua) tahun berturut-turut atau lebih, dapat diambil dalam tahun berikutnya untuk paling lama 24 (dua puluh empat) hari kerja, termasuk cuti tahunan dalam tahun yang sedang berjalan.

2. Cuti Besar

   Cuti besar adalah cuti yang dapat diambil oleh seorang PNS setelah bekerja 6 tahun berturut-turut. Lama cuti besar adalah maksimal 90 hari kalender.

3. Cuti Sakit

   Cuti sakit adalah cuti yang bisa diambil oleh seorang PNS ketika sakit dan membutuhkan waktu istirahat untuk pemulihan kondisinya.

4. Cuti Bersalin

   Cuti bersalin adalah cuti yang dapat diambil oleh PNS wanita untuk melahirkan anak pertama, kedua, dan ketiga. Lama cuti bersalin adalah maksimal 90 hari kalender.

5. Cuti Karena Alasan Penting PNS

   PNS dapat mengambil cuti karena alasan penting untuk paling lama 60 hari kalender. Lamanya cuti karena alasan penting hendaknya ditetapkan sedemikian rupa, sehingga benar-benar hanya untuk waktu yang diperlukan saja. Dalam kondisi tertentu, cuti alasan penting hanya bisa diambil setelah hak cuti pegawai yang bersangkutan tidak mencukupi lagi/habis.

6. Cuti di Luar Tanggungan Negara Cuti

   Cuti diluar tanggungan negara dapat diberikan kepada PNS yang telah bekerja sekurang-kurangnya 5 tahun secara terus-menerus dan adanya alasan-alasan pribadi yang penting dan mendesak (Setiyanto, Samopa, & Alwi, 2013).