Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB II

LANDASAN TEORI

2.1 Aplikasi Berbasis *Web*

Aplikasi berbasis *web* dapat diartikan sebagai program aplikasi yang berjalan pada Internet, Ekstranet, atau Intranet. Pengguna aplikasi *web* harus menggunakan *web browser* pada komputer untuk menggunakan aplikasi tersebut [5]. Aplikasi berbasis *web* memiliki 3 (tiga) jenis, yaitu:

1. Aplikasi *Web* Statis

   Aplikasi *web* statis tidak memiliki fitur interaksi dengan pengunjung aplikasi. Aplikasi ini hanya bertujuan untuk mendistribusikan informasi kepada publik.

2. Aplikasi *Web* Interaktif Sederhana

   Aplikasi *web* interaktif memiliki fitur yang mengizinkan adanya aktivitas saling bertukar informasi antara pengunjung aplikasi dengan pemilik aplikasi *web* tersebut.

3. Sistem Basis Data Berbasis *Web* Kompleks

   Aplikasi *web* kompleks memiliki fitur yang lebih kompleks sehingga dapat menangani transaksi bisnis *online* modern seperti perbankan *online* dan *query* basis data interaktif.

2.2 PHP

PHP merupakan sebuah bahasa pemrograman berbasis *web* bersumber dari bahasa pemrograman Perl, yang merupakan pengembangan dari bahasa pemrograman C [6].

Saat ini, PHP sudah sering digunakan oleh para pengembang *web* karena PHP memiliki beberapa kelebihan dibandingkan bahasa pemrograman *web* lain. Diantaranya adalah PHP lebih mudah untuk dikembangkan an lebih cepat untuk dieksekusi. Selain itu, kode PHP yang sama dapat berjalan pada *web servers* yang berbeda dan sistem-sistem operasi yang berbeda tanpa harus diubah terlebih dahulu [7].

2.3 MySQL

MySQL merupakan salah satu *Relational Database Management System* (RDBMS) bersifat *open source* yang paling populer dan paling banyak digunakan dalam pengembangan aplikasi *web* [8].

MySQL dapat digunakan untuk banyak jenis keperluan dikarenakan karakteristik arsitekturnya yang unik dan fleksibel. MySQL juga dapat mendukung *data warehouses, embedded applications, online transaction processing* (OLTP), dan berbagai fitur lainnya [9].

MySQL memiliki beberapa kelebihan diantaranya adalah sebagai berikut.

1. Dapat dikonfigurasi sehingga bisa dijalankan di berbagai jenis *hardware*.
2. Mendukung berbagai variasi tipe data.
3. Memiliki desain arsitektur *storage-engine* yang mampu memisahkan aktivitas proses *query* serta aktivitas *server* lainnya dari *data storage*. Hal ini membuat pengembang dapat memilih bagaimana data disimpan serta fitur dan karakteristik lain apa yang ingin digunakan oleh pengembang.

2.4 XAMPP

XAMPP merupakan salah satu *web server* yang bersifat *open source*. XAMPP mendukung Apache HTTP Server, MySQL, serta bahasa pemrograman PHP dan Perl [10]. XAMPP banyak digunakan oleh para pengembang aplikasi *web* karena konfigurasi dan penggunaan fitur yang mudah serta gratis.

2.5 *Decision Support System* (DSS)

Decision Support System (DSS) merupakan suatu sistem yang dapat membantu para pembuat keputusan dalam memecahkan masalah dengan memanfaatkan *database* dan model-model tertentu yang terdapat

di dalam komputer. DSS memiliki teknologi *problem-solving* yang melibatkan orang, *knowledge*, *software*, dan *hardware* untuk memfasilitasi aktivitas pembuatan keputusan [11].

DSS dapat berupa prosesor informasi manusia, prosesor informasi mekanis, atau prosesor informasi *human-machine* yang tergabung di dalam sebuah sistem pengambilan keputusan organisasi [12]. DSS dapat berfokus ke dalam beberapa area permasalahan atau hanya satu area saja serta memiliki beberapa variasi kemampuan *information-processing* yang terlibat di dalam aktivitas pengambilan keputusan.

Secara umum, DSS memiliki 4 komponen utama diantaranya adalah (a) *User Interface* (UI), (b) sistem basis data, (c) *model* dan *analytical tools* dan (d) arsitektur dan jaringan dari DSS itu sendiri [13]. Komponen-komponen tersebut mengidentifikasi persamaan dan perbedaan antara satu tipe DSS dengan tipe lainnya. Perbedaan antara komponen-komponen tersebut pula yang membentuk DSS *framework*.

DSS *framework* memiliki karakteristik utama yaitu komponen teknologi paling dominan yang menjalankan fungsi pendukung keputusan. Berdasarkan hal tersebut, DSS dapat dibagi ke dalam beberapa kategori diantaranya adalah sebagai berikut.

Tabel 2.1 DSS *extended framework*

Komponen Dominan	*User Group*	Tujuan	Teknologi yang Dipakai
Communications: Communications-Driven DSS	Pengguna di dalam dan antara organisasi	Mengadakan *meeting, posting* pengumuman	*Local Area Network* (LAN), *Web-based*
Database: Data-Driven DSS	*Manager*, staf, pengguna antar organisasi	*Ad hoc analysis, query a data warehouse*	*Local Area Network* (LAN), *Web-based*
Document Base: Document-Driven DSS	*Manager*, tenaga spesialis	Menemukan dokumen, *searching* halaman *web*	*Web-based*
Knowledge Base: Knowledge-Driven DSS	Pengguna internal, *customer*	Memilih produk, *management advicce*	*Local Area Network* (LAN), *Web-based*
Model: Model-Driven DSS	*Manager*, staf, *customer*	*Crew scheduling, decision analysis*	*Stand-alone PC, Web based*

2.6 *Neural Network*

Neural network merupakan salah satu bagian dari *soft computing*, dibuat pertama kali pada tahun 1943 oleh Waren McCulloch dan Walter Pits. Metode algoritma ini terinspirasi dari struktur jaringan syaraf serta cara kerja otak manusia.

Secara garis besar, cara kerja *neural network* dimulai dengan membuat *neuron-neuron* buatan yang saling berhubungan. *Neuron-neuron* tersebut akan menerima informasi yang akan diolah di dalamnya dan kemudian hasil olahan informasi tersebut akan diberikan ke *neuron* lainnya untuk diolah kembali hingga mendapatkan *output* yang diinginkan [14].

Neural network sudah sering digunakan sebagai *computational tools* untuk menyelesaikan masalah-masalah kompleks seperti pengenalan

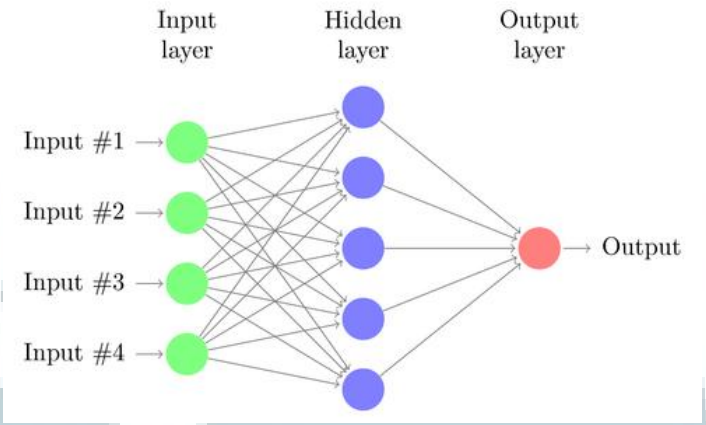
pola, estimasi fungsi, serta masalah klasifikasi yang tidak bisa diselesaikan oleh *analytical tools* lain [15].

Neural network memiliki beberapa kelebihan yang membuat metode ini dapat diaplikasikan di berbagai area diantaranya adalah sebagai berikut.

1. *Adaptive Learning*: Kemampuan untuk belajar bagaimana mengerjakan suatu *task* berdasarkan data yang diberikan.
2. *Real-time Operation*: Komputasi *neural net* dapat dilakukan secara paralel.
3. *Self-organization*: *Neural network* mampu membuat organisasi atau representasi dari informasi yang diterima pada saat proses pembelajaran dan operasi.

2.6.1 Struktur *Neural Network*

Pada dasarnya struktur *neural network* merupakan tiruan dari struktur neuron pada manusia sehingga mampu mengadopsi mekanisme berpikir sebuah sistem atau aplikasi yang menyerupai otak manusia [16].



Gambar 2.1 Struktur *Neural Network*

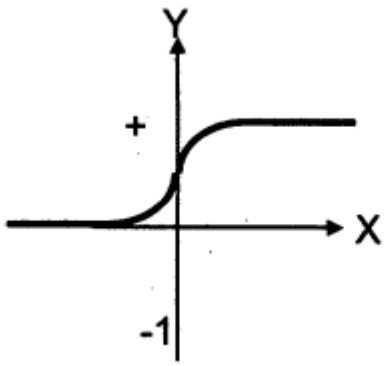
Struktur *neural network* terbentuk atas kumpulan neuron-neuron yang saling berhubungan serta tersusun ke dalam lapisan-lapisan tertentu. Lapisan-lapisan atau *layers* di dalam struktur *neural network* antara lain adalah sebagai berikut.

1. Lapisan *input*: Lapisan dimana data masuk yang nantinya akan diolah di dalam lapisan berikutnya. Tidak ada perubahan yang dilakukan dalam lapisan ini.
2. Lapisan tersembunyi: Lapisan dimana data dari lapisan *input* diolah. *Neural network* dapat memiliki beberapa lapisan tersembunyi atau tidak ada sama sekali. Nilai *input* pada setiap neuron dalam lapisan tersembunyi merupakan hasil perkalian antara bobot (W_i) dengan hasil *output* (X_i) dari lapisan sebelumnya [17], digambarkan dengan rumus berikut.

$$X = \Sigma\, xi\, wi$$

Output dari lapisan ini merupakan hasil olahan *input* yang diolah menggunakan fungsi aktivasi dan akan menjadi *input* untuk lapisan selanjutnya.

Fungsi aktivasi yang paling umum digunakan adalah fungsi Sigmoid, karena dapat digunakan dalam jaringan *back propagation*. Fungsi Sigmoid dapat digambarkan sebagai berikut.



$$Y_{sigmoid} = \frac{1}{1 + e^{-x}}$$

Gambar 2.2 Fungsi Sigmoid

3. Lapisan *output*: Lapisan yang memiliki prinsip kerja yang sama dengan lapisan tersembunyi namun hasil *output* dari lapisan ini sudah bisa dianggap sebagai hasil dari *neural network*.

2.7 Algoritma *Back Propagation*

Back propagation merupakan salah satu jenis pembelajaran *neural network* yang bersifat *supervised learning*, dimana nilai *output* telah ditentukan di dalam *set* data. Algoritma ini dapat digunakan untuk meminimalisir kesalahan atau *error* pada *neural network* [18]. Algoritma

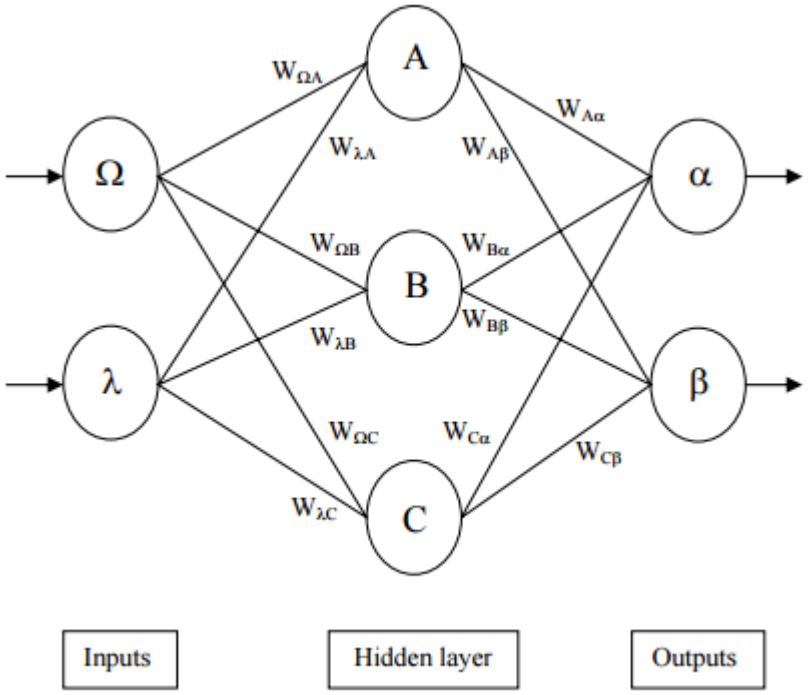
ini digunakan untuk pembelajaran bagi *multi-layer neural network*. Ketika proses pembelajaran selesai, algoritma ini akan menghasilkan *output* yang benar-benar sesuai dengan *input* yang telah ditentukan.

Back propagation dibagi menjadi 2 (dua) fase, diantaranya adalah sebagai berikut.

1. Melakukan proses *forward propagation*, pengolahan data secara searah, melalui jaringan yang telah dibuat hingga menghasilkan nilai *output*.
2. Proses *back propagation* dilakukan bersamaan dengan proses *update* setiap bobot.

Secara garis besar, kalkulasi pada *back propagation* adalah sebagai berikut.

UMN



Gambar 2.3 Ilustrasi *Back Propagation Network*

1. Setelah mendapatkan nilai *output* dari proses *forward propagation*, hitung *error* dari setiap neuron *output*.

$$\delta_\alpha = out_\alpha\,(1 - out_\alpha)\,(Target_\alpha - out_\alpha)$$
$$\delta_\beta = out_\beta\,(1 - out_\beta)\,(Target_\beta - out_\beta)$$

2. Ubah bobot dari setiap neuron pada lapisan *output*.

$$W^+_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\, out_A \qquad W^+_{A\beta} = W_{A\beta} + \eta\delta_\beta\, out_A$$
$$W^+_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\, out_B \qquad W^+_{B\beta} = W_{B\beta} + \eta\delta_\beta\, out_B$$
$$W^+_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\, out_C \qquad W^+_{C\beta} = W_{C\beta} + \eta\delta_\beta\, out_C$$

3. Hitung *error* dari lapisan tersembunyi.

$$\delta_A = out_A\,(1 - out_A)\,(\delta_\alpha W_{A\alpha} + \delta_\beta W_{A\beta})$$
$$\delta_B = out_B\,(1 - out_B)\,(\delta_\alpha W_{B\alpha} + \delta_\beta W_{B\beta})$$
$$\delta_C = out_C\,(1 - out_C)\,(\delta_\alpha W_{C\alpha} + \delta_\beta W_{C\beta})$$

4. Ubah bobot dari setiap neuron pada lapisan tersembunyi.

$$W^+_{\lambda A} = W_{\lambda A} + \eta\delta_A\, in_\lambda \qquad W^+_{\Omega A} = W^+_{\Omega A} + \eta\delta_A\, in_\Omega$$
$$W^+_{\lambda B} = W_{\lambda B} + \eta\delta_B\, in_\lambda \qquad W^+_{\Omega B} = W^+_{\Omega B} + \eta\delta_B\, in_\Omega$$
$$W^+_{\lambda C} = W_{\lambda C} + \eta\delta_C\, in_\lambda \qquad W^+_{\Omega C} = W^+_{\Omega C} + \eta\delta_C\, in_\Omega$$

5. Lakukan kembali proses *forward propagation* dan *back propagation* hingga mencapai batas maksimum pembelajaran atau mencapai batas toleransi *error* yang telah ditentukan sebelumnya.

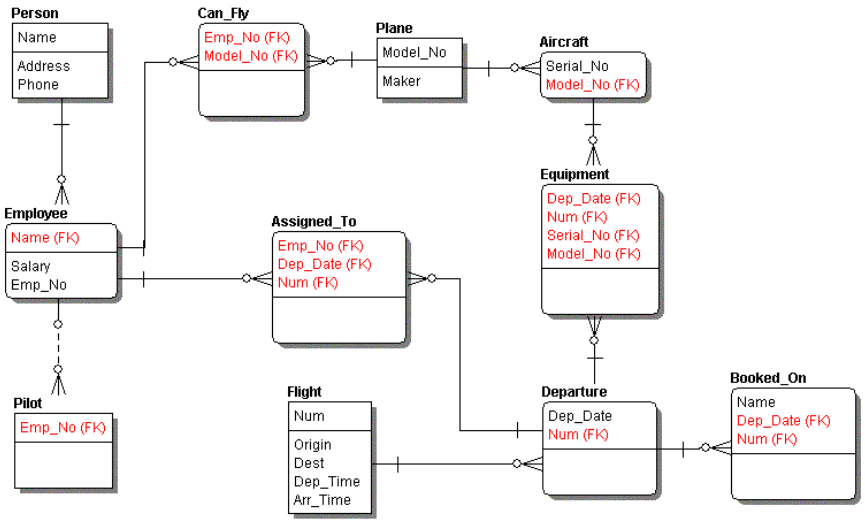
2.8 *K-Fold Method*

K-fold method merupakan salah satu metode *cross validation* yang dipakai dalam proses pengujian sebuah *neural network*. Metode *k-fold* membagi *set* data ke dalam *k* partisi dengan proporsi yang sama untuk digunakan dalam *training* dan *testing*. Salah satu *subset* akan digunakan untuk *testing* sementara sisanya digunakan untuk *training*. Kemudian dilakukan pertukaran fungsi dari *subset* tersebut sehingga *subset* sebelumnya dapat digunakan untuk *training* sementara *subset* baru yang lain digunakan untuk *testing* [19]. Proses tersebut dilakukan sebanyak *k* kali sehingga setiap *subset* dapat digunakan untuk *testing* sebanyak 1 (satu) kali.

Hasil akurasi yang didapatkan dari metode ini berasal dari rata-rata akurasi dari total akurasi pada *testing*. Sementara hasil *error* didapatkan dengan menjumlahkan *error* dari semua *k* proses tersebut.

2.9 *Entity Relationship Diagram* (ERD)

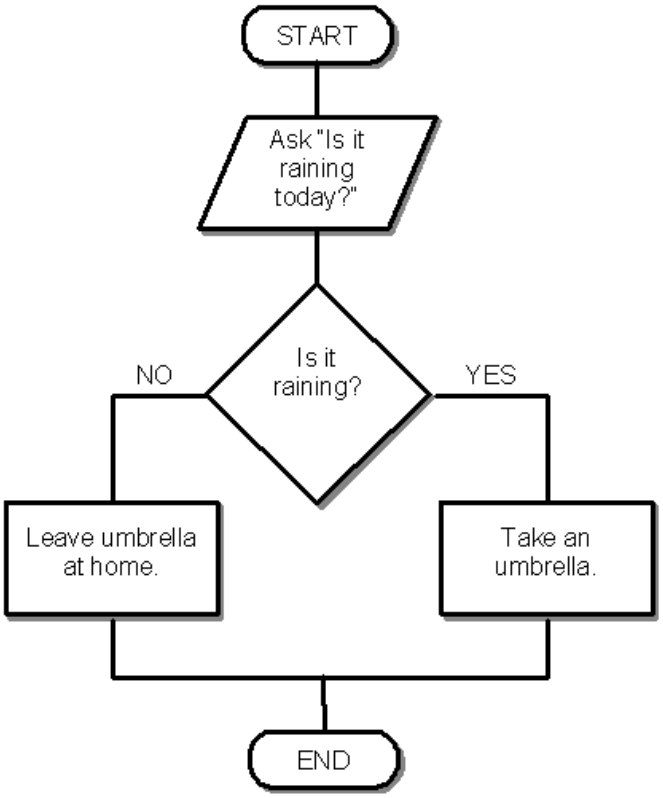
Entity Relationship Diagram digunakan untuk menampilkan relasi antar entitas. Entitas merupakan suatu sumber yang bisa digambarkan sebagai suatu individu ataupun objek. Diagram ini digunakan untuk menggambarkan basis data dari suatu organisasi dan menjadi dasar untuk pembuatan table yang ada pada basis data [20].



Gambar 2.4 Contoh *Entity Relationship Diagram* (ERD)

2.10 Diagram *Flowchart*

Diagram *flowchart* merupakan diagram yang menggambarkan prosedur suatu program secara logis dan sistematis. *Flowchart* digunakan terutama dalam hal dokumentasi dan representasi suatu algoritma [21].



Gambar 2.5 Contoh diagram *flowchart*

Berikut merupakan simbol-simbol yang umumnya dipakai di dalam diagram *flowchart*.

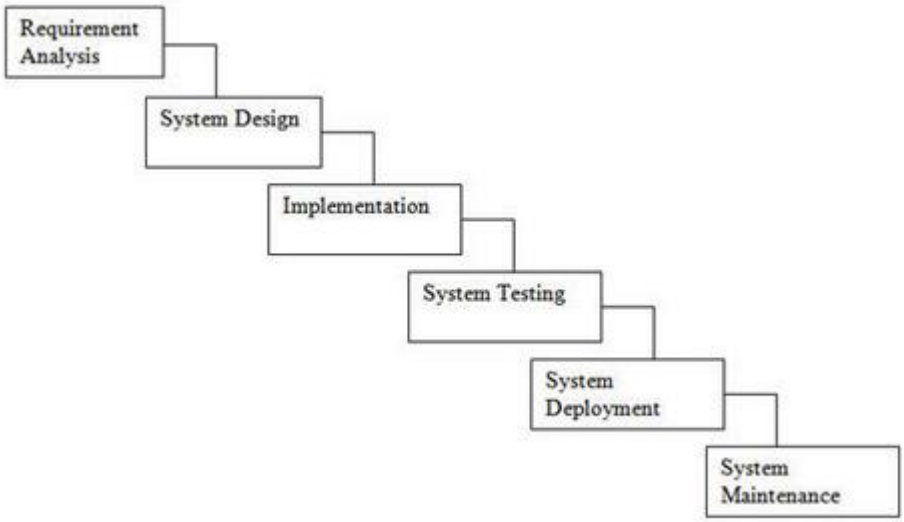
Simbol	Nama Simbol	Keterangan	Simbol	Nama Simbol	Keterangan
	Simbol input/output	Simbol input/output (*input/output symbol*) digunakan untuk mewakili data input/output		Simbol persiapan	Simbol persiapan (*preparation symbol*) digunakan untuk memberi nilai awal suatu besaran
	Simbol Proses	Simbol proses digunakan untuk mewakili suatu proses		Simbol titik terminal	Simbol titik terminal (*terminal point symbol*) digunakan untuk menunjukan awal dan akhir dari suatu proses
	Simbol garis alir	Simbol garis alir (*flow lines symbol*) digunakan untuk menunjukan arus dari proses			
	Simbol penhubung	Simbol penghubung (*connector symbol*) digunakan untuk menunjukan sambungan dari bagan alir yang terputus dihalaman yang sama atau di halaman lainnya			
	Simbol keputusan	Simbol keputusan (*decision symbol*) diginakan untuk suatu penyeleksian kondisi di dalam program			
	Simbol proses terdefinisi	Simbol proses terdefinisi (*predifined proses symbol*) digunakan untuk menunjukan suatu operasi yang rinciannya ditunjukan di tempat lain			

Gambar 2.6 Simbol-simbol diagram *flowchart*

2.11 *System Development Life Cycle* (SDLC)

System Development Life Cycle (SDLC) merupakan suatu siklus pengembangan *software* yang terdiri dari fase-fase yang dilakukan mulai dari perencanaan hingga implementasi *software* [22]. Model dari SDLC dapat berupa model deskriptif atau preskriptif terkait dengan bagaimana *software* dikembangkan.

SDLC *Waterfall* merupakan salah satu model SDLC, dimana proses pengembangan *software* pada model ini bersifat *sequential*. Setiap fase harus diselesaikan terlebih dahulu sebelum melanjutkan ke fase yang berikutnya [23].



Gambar 2.7 Model SDLC *Waterfall*

Fase-fase yang terdapat pada model SDLC *Waterfall* adalah sebagai berikut.

1. Analisi kebutuhan: Fase pengidentifikasian masalah, pengumpulan informasi yang dibutuhkan dalam pengembangan *software*, serta menganalisis kebutuhan-kebutuhan yang diaplikasikan ke dalam *software* tersebut berdasarkan informasi yang telah dikumpulkan.
2. Desain sistem: Fase perancangan arsitektur *software* secara konseptual berdasarkan hasil analisis yang telah dilakukan sebelumnya.
3. Implementasi: Fase implementasi dari rancangan *software* yang telah dibuat serta fase dimana pemrograman *software* dilakukan. Pemeriksaan fungsi-fungsi pada *software* dilakukan untk memastikan *software* berjalan sesuai dengan rancangannya.

4. Uji coba sistem: Fase ujicoba *software* untuk memastikan bahwa *software* dapat menyelesaikan masalah yang didapat pada fase analisis serta berfungsi dengan baik.
5. Penyebaran sistem: Fase dimana *software* lulus ujicoba dan dapat digunakan oleh publik.
6. Perawatan sistem: Fase perawatan *software* secara berkala agar *software* selalu berfungsi dengan baik.