BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian tentang deteksi keaslian sepatu tidak banyak ditemukan sehingga dalam tinjauan pustaka akan lebih membahas tentang implementasi sistem pakar.

Penelitian dari Turang (2018) dengan judul Aplikasi Sistem Pakar Berbasis *Web* Untuk Mendiagnosa Penyakit Syaraf Pusat Dengan Metode *Forward Chaining* membahas tentang diagnosis dini penyakit syaraf pusat yang memberikan informasi kepada pengguna mengenai syaraf yang diderita berdasarkan keluhan atau gejala yang timbul. Sistem pakar ini bertujuan membantu pengguna untuk mengetahui penyakit syaraf pusat yang diderita dan terapi penyembuhannya serta memberikan informasi yang perlu digunakan oleh pengguna yang mengalami gangguan penyakit syaraf. Di aplikasi ini memiliki data rekomendasi yang dihasilkan dari seorang pakar yaitu dokter syaraf yang dilengkapi dengan jenis penyakit, gejala penyakit, cara pengobatan, dan terapi penyembuhannya. Pada penelitian ini, terdapat 12 penyakit dan 44 gejala. Sistem pakar ini menggunakan pohon pelacakan untuk menentukkan penyakit syaraf dengan mengunakan proses telusuri *Depth-First Search* yang akan menelusuri kaidah secara mendalam. Penelitian ini menggunakan metode *Forward Chaining* dimana pencocokan data atau pernyataan dimula dari bagian sebelah kiri (*IF* terlebih dahulu) dengan kata lain penalaran dimulai dari fakta terlebih dahulu untuk menguji kebenaran hipotesa.

Penelitian dari Wijaya dan Raharja (2015) dengan judul Implementasi Metode *Forward Chaining* Pada Sistem Pakar Penentuan Karakter Diri Berbasis *Website* Menggunakan *Framework Codeigniter* membahas tentang sebuah sistem pakar yang digunakan untuk membantu para psikolog melakukan penentuan karakter guna untuk melakukan tes karakter diri yang akan menghasilkan pengenalan diri serta dapat mengetahuai kelebihan dan kekurangan yang ada pada diri sendiri. Pada penelitian ini terdapat 4 tipe karakter dan 102 gejala. Sistem pakar ini berbasis *website* yang

menggunakan metode *Forward Chaining* dan menggunakan bahasa pemrograman PHP dengan *Framework CodeIgniter*.

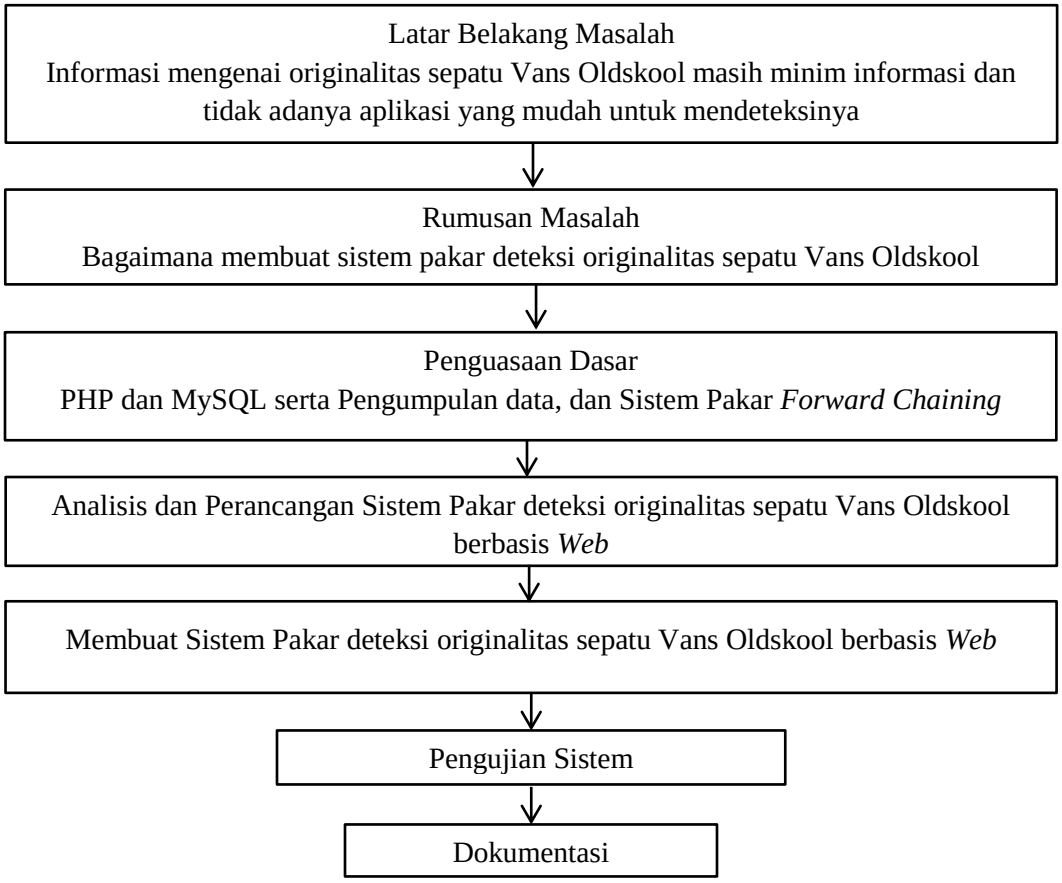
Penelitian Saputra, dkk (2015) dengan judul Sistem Pakar Untuk Diagnosa Penyakit Kucing Berbasis *Web* Menggunakan *Framework Codeigniter* membahas tentang membantu para pemelihara dan pecinta kucing agar dapat mengetahui penyakit yang menyerang kucing dan sekaligus dapat pula mengetahui solusi yang tepat untuk menangani penyakit tersebut yang dapat diakses dengan melalui internet. Pada penelitian ini terdapat 11 penyakit dan 42 gejala. Sistem pakar ini dikembangkan dengan metode *Forward Chaining*, bahasa pemrograman PHP, *Framework CodeIgniter*, dan *database* MySql.

Penelitian dari Dharmaningrum (2018) dengan judul Implementasi Metode Forward Chaining Untuk Deteksi Dini Gangguan Bipolar Pada Remaja Berbasis Web. Penelitian tersebut membahas tentang sebuah sistem pakar yang digunakan untuk mengetahui deteksi dini gangguan bipolar pada remaja. Dalam penelitian ini, terdapat 2 gangguan dan 19 gejala. Penelitian ini menggunakan bahasa pemrograman PHP. Sistem pakar ini menggunakan metode *Forward Chaining* dan berbasis *website*.

Penelitian dari Yasmiyati (2017) dengan judul Sistem Pakar *Diagnosis* Penyakit Pada Perokok Dengan Metode *Forward Chaining* Berbasis *Web*. Penelitian tersebut membahas tentang *diagnosis* penyakit pada perokok yang memberikan informasi kepada pengguna agar mengetahui penyakit-penyakit yang ditimbulkan karena merokok berdasarkan gejala yang timbul. Sistem pakar diharapkan dapat mengurangi jumlah perokok. Aplikasi ini dapat memberikan informasi tentang penyakit yang ada dengan presentase sebesar 89,2%. Informasi yang diberikan aplikasi ini memuat tentang informasi penyakit, gejala awal penyakit, dan pengobatan penyakit. Dalam penelitian ini, terdapat 8 penyakit dan 22 gejala Penelitian ini menggunakan bahasa pemrograman PHP. Metode yang digunakan dalam penelitian ini menggunakan *Forward Chaining* dan berbasis *website*.

2.2 Kerangka Pemikiran

Tahapan kerangka pemikiran dalam membuat sistem pakar deteksi originalitas sepatu Vans Oldskool pada Gambar 2.1.



Gambar 2.1. Kerangka Pemikiran

Penjelasan dari kerangka pemikiran tersebut adalah :

1. Latar Belakang Masalah

Informasi yang kurang tentang ciri deteksi originalitas sepatu Vans Oldskool mengakibatkan banyak dari masyarakat yang tidak mengetahui cara membedakan Vans Oldskool asli atau palsu, sehingga masyarakat masih awam dan kurangnya informasi untuk membedakan asli atau palsu yang berdampak pada kebingungan masyarakat dalam mengidentifikasi sepatu Vans Oldskool yang mereka miliki apakah asli atau palsu.

2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan maka rumusan masalahnya adalah bagaimana membuat sistem pakar deteksi originalitas sepatu Vans Oldskool berbasis *Web*?

3. Penguasaan dasar PHP dan MySQL

Penguasaan dasar PHP dan MySQL merupakan suatu kunci terciptanya sistem pakar deteksi originalitas sepatu Vans Oldskool tersebut, karena pembuatan sistem ini menggunakan bahasa pemrograman PHP dan menggunakan database MySQL sebagai media penyimpan datanya.

4. Pengumpulan data

Tahap pengumpulan data pada penelitian ini melalui observasi, wawancara dan studi literatur. Pengumpulan data bertujuan untuk mengetahui permasalahan dan memenuhi kebutuhan informasi tentang sepatu Vans Oldskool.

5. Analisis dan perancangan sistem

Analisis dan perancangan sistem pakar deteksi originalitas sepatu Vans Oldskool bertujuan untuk mengetahui kesiapan data yang terkumpul dan rancangan dari aplikasi sistem pakar deteksi originalitas sepatu Vans Oldskool yang diharapkan dapat memberikan suatu informasi edukatif dengan efektif dan mudah dipahami oleh pengguna. Perancangan sistem pada penelitian ini menggunakan UML (*Unified Modeling Language*).

6. Rancang Bangun Sistem

Membuat sistem pakar deteksi originalitas sepatu Vans Oldskool menggunakan bahasa pemrograman PHP dan *database* MySQL sebagai media penyimpanan datanya.

7. Pengujian Sistem

Pengujian sistem merupakan tahap setelah sistem berhasil dibuat. Pengujian sistem bertujuan untuk mengetahui kelebihan, kelemahan serta mengetahui kelayakan suatu sistem untuk digunakan oleh masyarakat. Pengujian sistem pada penelitian ini menggunakan metode *BlackBox* dan pengujian dengan kuisoner.

8. Dokumentasi

Pada tahap dokumentasi ini menjelaskan dan memberikan tampilan hasil dari pembuatan aplikasi sistem pakar deteksi originalitas sepatu Vans Oldskool yang telah dirancang sebelumnya.

2.3 Teori – teori Pendukung

2.3.1 Sistem

Menurut Kadir (2014), sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Tujuan utama sistem yang umum ada tiga macam yaitu:

a. Untuk mendukung fungsi kepengurusan manajemen.
b. Untuk mendukung pengambilan keputusan manajemen.
c. Untuk mendukung kegiatan operasi perusahaan.

2.3.2 Sistem Pakar

Menurut Budiharto dan Suhartono (2014), sistem pakar adalah program komputer yang mensimulasi penilaian dan perilaku manusia atau organisasi yang memiliki pengetahuan dan pengalaman ahli dalam bidang tertentu.

Kelebihan dan karakteristik sistem pakar menurut Budiharto dan Suhartono (2014), banyak digunakan pada aplikasi terkini dan kompleks karena :

a. Sistem pakar dapat bertindak sebagai konsultan, instruktur, atau pasangan/rekan.
b. Meningkatkan *availability* atau kepakaran tersedia pada semua perangkat komputer.
c. Mengurangi bahaya.
d. Pengetahuan dapat tidak lengkap, namun keahlian dapat diperluas sesuai kebutuhan. Program konvensial harus lengkap sesuai kebutuhan sebelum mereka dapat digunakan.
e. *Database* yang cerdas, sistem pakar dapat digunakan untuk mengakses *database* secara cerdas.

Sistem pakar biasanya didesain untuk memiliki karakteristik sebagai berikut :

a. *High performance*.
b. *Adequate respone time*.
c. *Good reliability*.
d. *Understandable*.

Manfaat dan kekurangan sistem pakar menurut Haryadi (2016) mengandalkan kecerdasan dari pakar yang direkam menjadi mesin inferensi dan *rule* inferensi sehingga memiliki memiliki manfaat dan kekurangan tertentu. Adapun manfaat dari sistem pakar yaitu:

a. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
b. Membuat seseorang yang awam bekerja seperti layaknya seorang pakar.
c. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
d. Mampu menangkap pengetahuan dan kepakaran seseorang.
e. Memudahkan akses pengetahuan pakar.
f. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula yang bekerja dalam sistem pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
g. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

Setiap sistem yang memiliki manfaat pasti memiliki kekurangan tertentu, begitu juga dengan sistem pakar menurut Haryadi (2016). Adapun kekurangan dari sistem pakar yaitu :

a. Biaya yang sangat mahal untuk membuat dan memeliharanya.
b. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
c. Sistem pakar tidak 100% bernilai benar.

Konsep dasar sistem pakar dapat berjalan atau berfungsi dengan baik apabila dasar-dasar dari sistem pakar terpenuhi. Konsep dasar sistem pakar terdiri dari

beberapa bagian. Adapun konsep dasar dari sistem pakar menurut Haryadi (2016) terdiri dari :

a. Keahlian adalah suatu pengetahuan khusus yang diperoleh dari latihan, belajar dan pengetahuan dapat berupa fakta, teori, aturan untuk memecahkan masalah.
b. Ahli (*expert*) adalah melibatkan kegiatan mengenali dan memformulasikan permasalahan, memecahkan masalah secara tepat dan tepat, menerangkan pemecahannya, belajar dari pengalaman, merestrukturarisasi pengetahuan, memecahkan aturan serta menentukan relevansi.
c. Mentransfer keahlian (*transferring expertise*) adalah proses pentransferan keahlian dari seorang pakar ke dalam komputer agar dapat digunakan oleh orang lain yang bukan pakar. Pengetahuan tersebut ditempatkan ke dalam sebuah komponen yang dinamakan dengan basis pengetahuan.
d. Menyimpulkan aturan (*inferencing rule*) adalah merupakan kemampuan komputer yang telah diprogram. Penyimpulan ini dilakukan oleh mesin inferensi yang meliputi prosedur tentang penyelesaian masalah.
e. Peraturan (*rule*) adalah diperlukan karena mayoritas dari sistem pakar bersifat *rule-based systems*, yang berarti pengetahuan disimpan dalam bentuk peraturan.
f. Kemampuan menjelaskan (*explanation capability*) adalah karakteristik dari sistem pakar yang memiliki kemampuan dalam menjelaskan atau memberi saran mengapa tindakan tertentu dianjurkan atau tidak dianjurkan.

Setiap sistem memiliki komponen-komponen yang bentuknya, begitu juga dengan sistem pakar. Komponen-komponen ini berinteraksi dan bekerja sama untuk mencapai tujuan dari sistem pakar. Adapun komponen sistem pakar menurut Haryadi (2016) yaitu:

a. Basis pengetahuan (*knowledge base*) Inti dari program sistem pakar karena basis pengetahuan ini merupakan representasi pengetahuan dari seorang pakar.
b. Basis data adalah bagian yang mengandung semua fakta yang didapatkan pada saat pengambilan kesimpulan sedang dilaksanakan.

c. Mesin inferensi adalah bagian yang mengandung mekanisme fungsi berfikir dan pola penalaran sistem yang digunakan oleh seorang pakar.
d. Antar muka pemakai (*user interface*) adalah bagian penghubung antara program sistem pakar dengan pemakainya. Pada bagian ini terjadi dialog antara program dan pemakai dimana program mengajukan pertanyaan berbentuk ya atau tidak (*yes or no question*) atau berbentuk menu pilihan. Melalui jawaban dari pemakai maka sistem pakar akan mengambil kesimpulan berupa informasi atau anjuran sesuai dengan sifat dari sistem pakar.

Metode inferensi sistem pakar merupakan proses untuk menghasilkan informasi dari fakta yang diketahui atau diasumsikan. Inferensi adalah konklusi logis (*logical conclusion*) atau implikasi berdasarkan informasi yang tersedia. Dalam sistem pakar proses inferensi dilakukan dalam suatu modul yang disebut *inference engine* (mesin inferensi). Ada dua metode inferensi yang penting dalam sistem pakar yaitu :

a. Algoritma *forward-chaining* adalah satu dari dua metode utama *reasoning* (pemikiran) ketika menggunakan *inference engine* (mesin pengambil keputusan) dan bisa secara logis dideskripsikan sebagai aplikasi pengulangan dari modus ponens (satu set aturan inferensi dan argumen yang valid). *Forward-chaining* mulai bekerja dengan data yang tersedia dan menggunakan aturan - aturan inferensi untuk mendapatkan data yang lain sampai sasaran atau kesimpulan didapatkan. Mesin inferensi yang menggunakan *Forward Chaining* mencari aturan-aturan inferensi sampai menemukan satu dari *antecedent* (dalil hipotesa atau klausa *IF - THEN*) yang benar. Ketika aturan tersebut ditemukan maka mesin pengambil keputusan dapat membuat kesimpulan, atau konsekuensi (klausa *THEN*), yang menghasilkan informasi tambahan yang baru dari data yang disediakan.
b. Algoritma *backward-chaining*, sesuai namanya bekerja mundur dari *query*-nya. Jika *query* q diketahui adalah benar, maka tak ada yang perlu dikerjakan selanjutnya. Selain itu, algoritmanya akan mencari implikasi-implikasi di dalam

basis data pengetahuan atau *knowledge base* yang kesimpulannya adalah q. Jika semua premis - premis dari salah satu implikasi-implikasi tersebut bisa dibuktikan benar, maka q adalah benar. *Backward-chaining* adalah sebuah bentuk pemikiran yang dikendalikan oleh tujuan atau *goal*.

2.3.3 *Website*

Menurut Humairah (2015), *website* merupakan kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing masing dihubungkan dengan jaringan-jaringan halaman.

Unsur - Unsur *website* untuk menyediakan keberadaan sebuah *website* yang dikutip Wiratama (2014), maka harus tersedia unsur - unsur penunjangnya sebagai berikut :

a. Nama domain (*Domain name/URL – Uniform Resource Locator*) Pengertian Nama domain atau biasa disebut dengan *domain name* atau URL adalah alamat unik di dunia internet yang digunakan untuk mengidentifikasi sebuah *website*, atau dengan kata lain *domain name* adalah alamat yang digunakan untuk menemukan sebuah *website* pada dunia *internet*.
b. *Web Hosting* dapat diartikan sebagai ruangan yang terdapat dalam *harddisk* tempat menyimpan berbagai data, file-file, gambar dan lain sebagainya yang akan ditampilkan di *website*. Besarnya data yang bisa dimasukkan tergantung dari besarnya *web hosting* yang disewaatau dipunyai, semakin besar *web hosting* semakin besar pula data yang dapat dimasukkan dan ditampilkan dalam *website*. *Web hosting* juga diperoleh dengan menyewa. Besarnya *hosting* ditentukan ruangan *harddisk* dengan ukuran MB (*Mega Byte*) atau GB (*Giga Byte*).
c. Bahasa Program (*Scripts Program*) adalah bahasa yang digunakan untuk menerjemahkan setiap perintah dalam *website* yang pada saat diakses. Jenis bahasa program sangat menentukan statis, dinamis atau interaktifnya sebuah

website. Semakin banyak ragam bahasa program yang digunakan maka akan terlihat *website* semakin dinamis, dan interaktif serta terlihat bagus. Jenis jenis bahasa program yang banyak dipakai para desainer *website* antara lain HTML, ASP, PHP, JSP, Java Scripts, Java applets dan lain - lainnya. Bahasa dasar yang dipakai setiap *website* adalah HTML sedangkan PHP, ASP, JSP dan lain – lainnya merupakan bahasa pendukung yang bertindak sebagai pengatur dinamis, dan interaktifnya situs. Bahasa program ASP, PHP, JSP atau lainnya bisa dibuat sendiri.

d. Desain *website* menentukan kualitas dan keindahan sebuah *website* supaya informasi yang ada di dalam *website* dapat tersampaikan dengan baik. Desain sangat berpengaruh kepada penilaian pengunjung akan bagus tidaknya sebuah *website*.

2.3.4 *Unified Modeling Language* (UML)

Menurut Sukamto dan Shalahuddin (2016) dijelaskan pada perkembangan teknologi perangkat lunak diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisari agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Pada perkembangan teknik pemrograman berosientasi objek, muncul sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (UML).

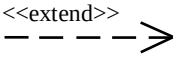
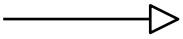
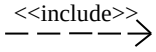
Ada 6 (enam) macam diagram dalam *Unified Modeling Language* (UML), yaitu :

2.3.4.1 *Use Case Diagram*

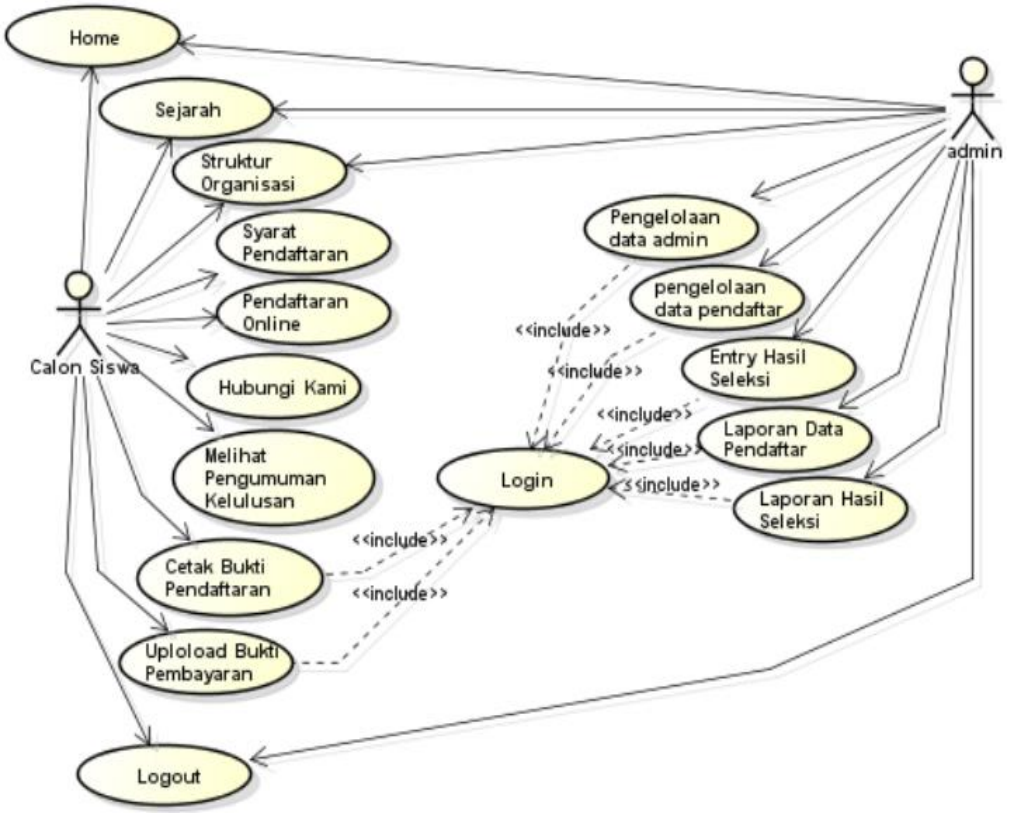
Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di

dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol *use case diagram* dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Use Cass Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1	Nama *Use Case* 	*Use case*	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan kata kerja di awal di awal frase nama *use case*.
2		*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3		*Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4	<<extend>> 	*Extend*	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5		*Generalization*	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6	<<include>> 	*Include*	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

Berikut adalah contoh *use case* diagram yang terlihat pada Gambar 2.2.



Gambar 2.2 Contoh *Use Case* Diagram.

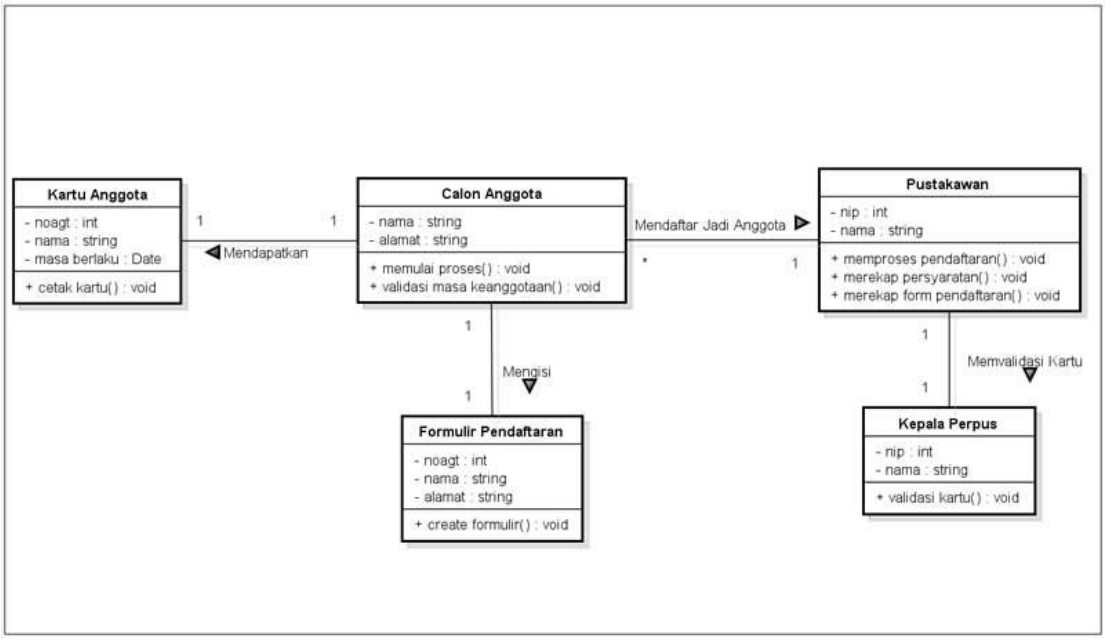
2.3.4.2 *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode operasi. Kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas (Sukamto dan Shalahuddin, 2016). Simbol *class diagram* dapat dilihat pada Tabel 2.2.

Tabel 2.2 *Class Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1	Nama kelas + Atribut + Operasi()	*Class*	Kelas pada struktur sistem
2	◯	*Interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3	———	*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity.*
4	——→	*Directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain biasanya juga disertai dengan *multiplicity*.
5	——▷	*Generalization*	Relasi antar kelas dengan makna generalisasi - spesialisasi (umum – khusus).
6	——→	*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
7	——◇	*Aggregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*).

Berikut adalah contoh *class* diagram yang terlihat pada Gambar 2.3.

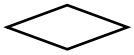
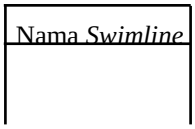


Gambar 2.3 Contoh *Class* Diagram.

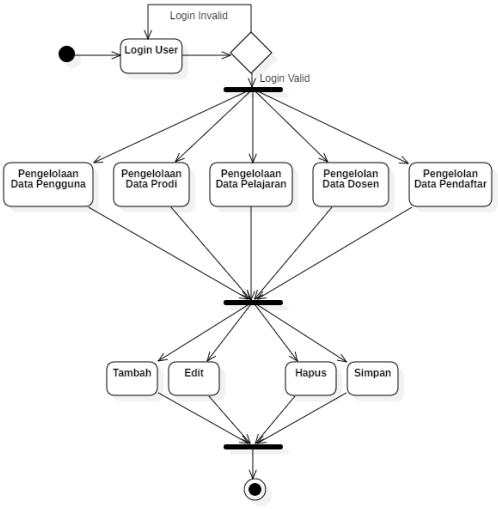
2.3.4.3 *Activity Diagram*

Activity diagram menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Sukamto dan Shalahuddin, 2016). Simbol *Activity Diagram* ditujukan pada Tabel 2.3.

Tabel 2.3 *Activity Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		Status awal	Status awal aktivitas sistem.
2		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3		*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5		Status akhir	Status akhir yang dilakukan sistem.
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Berikut adalah contoh *activity* diagram yang terlihat pada Gambar 2.4.

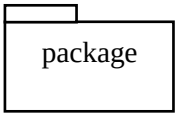
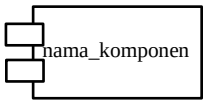


Gambar 2.4 Contoh *Activity* Diagram.

2.3.4.4 *Component Diagram*

Diagram komponen *atau component* diagram dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem (Sukamto dan Shalahuddin, 2016). Simbol *component* diagram dilihat pada Tabel 2.4.

Tabel 2.4 *Component Diagram*

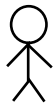
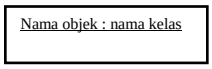
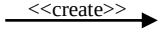
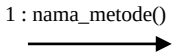
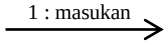
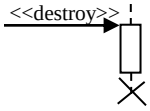
NO	GAMBAR	NAMA	KETERANGAN
1		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
2		*Component*	Komponen sistem
3		*Dependency*	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
4		*Interface*	Sama dengan konsep *interface* pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen
5		*Link*	Relasi antar komponen

Berikut adalah contoh *component* diagram yang terlihat pada Gambar 2.5.

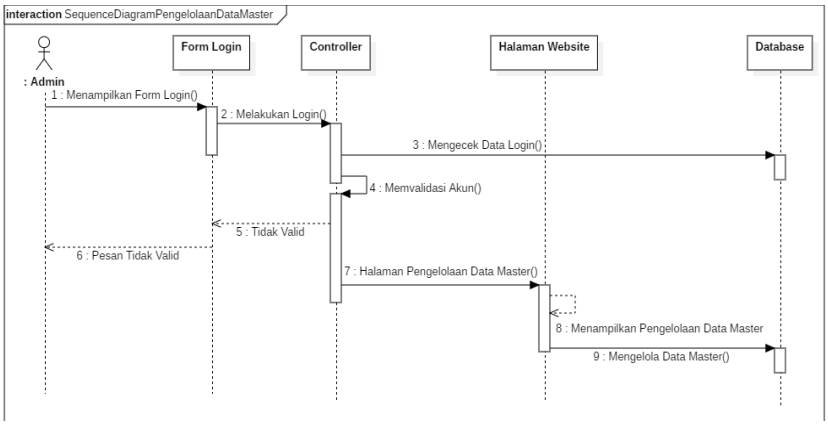
Gambar 2.5 Contoh *Component* Diagram.

2.3.4.5 *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (Sukamto dan Shalahuddin, 2016). Simbol *sequence* diagram dapat dilihat pada Tabel 2.5 *Sequence Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanyanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2		*Lifeline*	Menyatakan kehidupan suatu objek.
3		Objek	Menyatakan objek yang berinteraksi pesan.
4		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi.
5		Pesan tipe create	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6		Pesan tipe call	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.
7		Pesan tipe send	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8		Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy.*

Berikut adalah contoh *sequence* diagram yang terlihat pada Gambar 2.6.

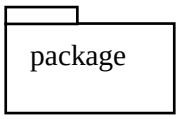
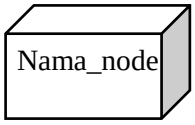
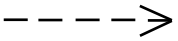


Gambar 2.6 Contoh *Sequence* Diagram.

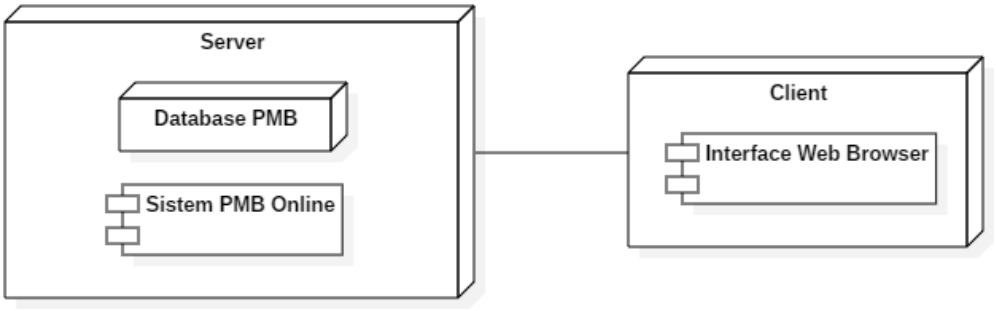
2.3.4.6 *Deployment Diagram*

Deployment diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan sistem tambahan yang menggambarkan rancangan *device*, *node*, dan *hardware*. Sistem *client*, sistem terdistribusi murni, rekayasa ulang aplikasi (Sukamto dan Shalahuddin, 2016). Simbol *deployment* diagram ditunjukkan pada Tabel 2.6.

Tabel 2.6 *Deployment Diagram*

NO	SIMBOL	NAMA	KETERANGAN
1.		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih *node.*
2.		*Node*	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika didalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelum pada diagram komponen.
3.		*Dependency*	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai.
4.		*Link*	Relasi antar *node.*

Berikut adalah contoh *deplyoment* diagram yang terlihat pada Gambar 2.7.



Gambar 2.7 Contoh *Deployment* Diagram.

2.3.5 Sepatu Vans

Pada jurnal Press (2002) dan artikel tentang Vans di forum Kaskus yang menerangkan tentang sejarah berdirinya Vans yaitu pada 16 maret 1966, Paul Van Doren dan tiga orang temannya membuat perusahaan baru bernama Van Doren Rubber Co (sekarang dikenal Vans). Vans adalah produsen sepatu, pakaian, dan aksesoris untuk olahraga *skateboard*, BMX , dan selancar. Pada awalnya, Paul Van Doren bekerja di pabrik sepatu bermerek *Randy's* sebagai buruh pembuat sepatu dan penyapu lantai. Setelah 20 tahun bekerja disana, ia sudah beberapa kali naik jabatan karena ketekunannya. Sampai akhirnya, Paul menjabat sebagai *Vice President* di *Randy's*. Akhirnya, ia memutuskan untuk keluar dari perusahaan sepatu tersebut dan pindah ke Southern California. Disana Paul dan temannya mendirikan perusahaan sepatu baru pada 1966 yang merupakan cikal bakal Vans. Vans memiliki beberapa jenis yang beredar di dunia, berikut adalah beberapa jenis dari sepatu Vans :

a. Vans Oldskool. Walaupun bukan yang pertama dibuat oleh Vans, berawal pada tahun 1977 yang disebut Vans *style* 36 atau yang sekarang kita kenal Vans Old Skool. Sebelum Old Skool awalnya dirilis, pendiri Vans Paul Van Doren sedang mencari cara untuk membantu membedakan sepatu Vans dari merek lain seperti Adidas maupun Nike yang sudah sangat dikenali. Van Doren dikenal sebagai *doodler* dan selalu membawa pensil dan kertas, ia menyusun gelombang sederhana sisi garis dan menyebutnya *jazz stripe* (garis putih gelombang) untuk

ikonik pengenal identitas sepatu Vans OldSkool. Sepatu Vans Oldskool terlihat pada Gambar 2.8.

Gambar 2.8 Sepatu Vans Oldskool.

b. Vans Authentic didesain di awal mula Vans berdiri pada tahun 1966, di Anaheim, California. Vans Authentic bisa dikatakan *premier* dari awal berdirinya Vans, yang pertama kali disebut Vans 44 *deck shoes*, yang sekarang dikenal sebagai Vans Authentic. Sepatu Vans Authentic terlihat pada Gambar 2.9.

Gambar 2.9 Sepatu Vans Authentic.

c. Vans Sk8 Hi merupakan sepatu yang pada tahun 1978 Vans memperkenalkan model terbaru mereka yaitu *Style* 38 yang sekarang dikenal sebagai Vans Sk8-Hi. Meskipun terlihat cukup sederhana, namun pada saat itu Vans Sk8 Hi cukup revolusioner karena memperkenalkan sepatu *skate* dengan model *Hi-Top* untuk pasar *skate shoes*. Sepatu *skate Hi-Top* sangat diminati karena memberikan perlindungan maksimal pada pergelangan kaki untuk bermain *skateboard*. Sepatu Vans Sk8 Hi dapat dilihat pada Gambar 2.10.

Gambar 2.10 Sepatu Vans Sk8 Hi.

d. Vans Era adalah salah satu *type* Vans *classics* yang pertama dirancang oleh Vans *shoes* setelah tipe *Authentic*. Rilis pada pertengahan tahun 1970-an yang fokus pada kultur *skateboard* meningkatkan popularitasnya. Pada waktu itu, merek *skateboard* sepatu Vans melayani sepatu slim mereka untuk orang-orang yang aktif dalam olahraga khususnya *skateboard*. Salah satu model khusus yang asalnya disebut Vans 95 ini menjadi solusi bagi para *skateborder* yang ingin bermaan *skate* dengan sepatu jenis slim. Sepatu Vans Era dapat dilihat pada Gambar 2.11.

Gambar 2.11 Sepatu Vans Era.

e. Vans Slip-On sepatu yang bermula pada tahun 1979 Vans memperkenalkan sepatu *type* 44 yang sekarang disebut Vans Slip-On dan dengan bantuan pemain *skateboard* dan BMX, Vans Slip-On menjadi salah satu yang populer di Southern California. Sepatu Vans Slip-On dapat dilihat pada Gambar 2.12.

Gambar 2.12 Sepatu Vans Slip-On.

2.3.6 PHP

Menurut Hidayatullah dan Kawistara (2017) mengemukakan bahwa PHP (*Hypertext Preprocessor*) adalah suatu bahasa *scripting* khususnya digunakan untuk *web development*. PHP memiliki sifat *server side scripting* sehingga untuk menjalankan PHP harus menggunakan web server.

Menurut Simarmata (2006), PHP begitu cepat populer dan berkembang begitu cepat karena PHP mempunyai beberapa keunggulan, yaitu:

a. Cepat, karena ditempelkan (*embedded*) didalam kode HTML, sehingga waktu tanggap menjadi pendek.
b. Tidak mahal – gratis. Pada kenyataannya PHP adalah gratis dan anda bisa mendapatkannya tanpa harus membayarnya.
c. Mudah untuk digunakan. PHP berisi beberapa fitur khusus dan fungsi yang dibutuhkan untuk membuat halaman web dinamis. Bahasa PHP dirancang untuk dimasukkan dengan mudah didalam file HTML.
d. Berjalan pada beberapa sistem operasi. Dia berjalan pada sistem operasi yang beragam, Windows, Linux, Mac OS dan kebanyakan variasi dari Unix.
e. Dukungan teknis tersedia secara luas, karena PHP menyediakan dukungan gratis via daftar diskusi e-mail.
f. Aman. Pengguna tidak melihat kode PHP, karena kode yang ditampilkan pada browser adalah kode HTML.
g. Dirancang untuk mendukung database. PHP meliputi kemampuan yang dirancang untuk berinteraksi dengan database tertentu. *Costumizable* Lisensi *open source* sehingga mengizinkan para pemprogram untuk memodifikasi *software* PHP, Menambahkan atau memodifikasi fitur-fitur yang dibutuhkan untuk lingkungan mereka sendiri.

Berikut adalah contoh *script* dari PHP (*Hypertext Preprocessor*) :

```
<html><head><title> Latihan menulis PHP </title></head>
<body>Belajar PHP <?php echo “Ini adalah bahasa PHP”;?></body></html>
```

2.3.7 MySQL

Menurut Hidayatullah dan Kawistara (2017), MySQL adalah salah satu aplikasi DBMS yang sudah sangat banyak digunakan oleh para pemrogram aplikasi web.

Sedangkan menurut Sibero (2013), MySQL adalah aplikasi perangkat lunak yang bertugas untuk menjalankan fungsi pengolahan data. Pertama MySQL dikembangkan oleh MySQL AB yang kemudian diakuisisi Sun *Microsystem* dan terakhir MySQL dikelola oleh Oracle *Coorporation*.

Menurut Saputra, dkk (2012), beberapa kelebihan yang dimiliki MySQL adalah sebagai berikut:

a. Bersifat *open source*, yang memiliki kemampuan untuk dapat dikembangkan lagi.
b. Menggunakan bahasa SQL (*Structure Query Language*), yang merupakan standar bahasa dunia dalam pengolahan data.
c. *Super performance* dan *realible*, tidak bisa diragukan, pemrosesan *database*-nya cepat dan stabil.
d. Sangat mudah dipelajari (*easy of use*)
e. Memiliki dukungan *support* (*group*) pengguna MySQL.
f. Mampu lintas *platform*, dapat berjalan di berbagai sistem operasi.
g. *Multiuser*, dimana MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami konflik.

Berikut adalah contoh *script* sederhana MySQL yang dapat dilihat pada Gambar 2.13.

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'password';
$koneksi = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $koneksi )
{
  die('Gagal Koneksi: ' . mysql_error());
}
$sql = 'INSERT INTO karyawan '.
       '(nama_karyawan,alm_karyawan, gaji_karyawan, tgl_gabung) '.
       'VALUES ( "nyekrip", "alamat nyekrip", 34000, NOW() )';

mysql_select_db('test_db');
$tambahdata = mysql_query( $sql, $koneksi );
if(! $tambahdata )
{
  die('Gagal tambah data: ' . mysql_error());
}
echo "Berhasil tambah data\n";
mysql_close($koneksi);
?>
```

Gambar 2.13. Contoh *Script* MySQL.

2.3.8 Basis Data

Hidayatullah dan Kawistara (2017), basis data adalah himpunan kelompok data yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Secara lebih lengkap pemanfaatan basis data dilakukan untuk memenuhi tujuan kecepatan, kemudahan, efisiensi ruang penyimpanan, keakuratan, ketersediaan, kelengkapan, keamanan dan pemakaian bersama. Contoh basis data yang dapat dilihat pada Gambar 2.14.

MHS

NPM	Nama	Alamat
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor
50095487	Pipit	Bekasi
21196353	Qurasih	Bogor

MKUL

KDMK	NTKULIAH	SKS
KK021	P. Basis Data	2
KD132	SIM	3
KU122	Pancasila	2

NILAI

NPM	KDMK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0
10296832	KD132	40	30

Gambar 2.14 Contoh Basis Data.

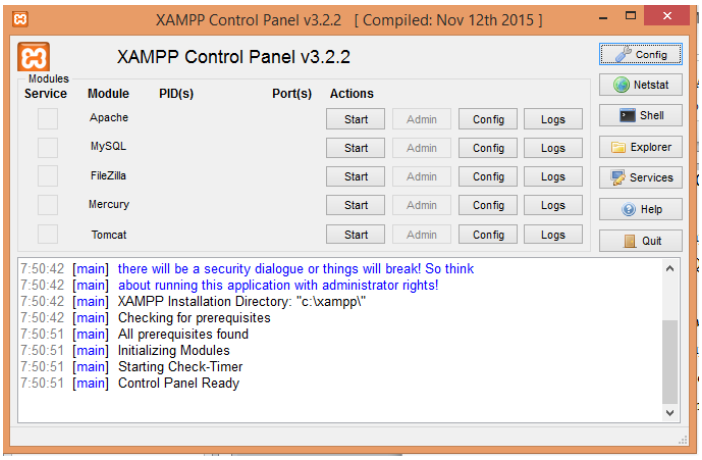
2.3.9 Xampp

Menurut Purbadian (2016), XAMPP merupakan suatu *software* yang bersifat *open source* yang merupakan pengembangan dari LAMP (Linux, Apache, MySQL, PHP and Perl).

Ada beberapa keunggulan dari XAMPP yaitu sebagai berikut:

a. *Database storage engine* ini banyak digunakan oleh *programmer* apalagi oleh *web developer* karena sifatnya yang gratis. Untuk yang *expert* sudah ada yang bayar.
b. Kemampuannya sudah bisa diandalkan, mempunyai kapasitas yang cukup mumpuni sekitar 60.000 tabel dengan jumlah *record* mencapai 5.000.000.000 bahkan untuk yang terbaru sudah lebih.
c. Keamanan datanya cukup aman walaupun tidak sehebat Postgre apalagi Oracle.
d. *Engine* ini *multiplatform* sehingga mampu diaplikasikan diberbagai sistem operasi. MySQL cocok diaplikasikan diaplikasi kelas kecil dan menengah.
e. Kelebihan paling utama engine ini adalah kecepatannya.

Berikut adalah tampilan panel XAMPP yang dapat dilihat pada Gambar 2.15



Gambar 2.15 Tampilan XAMPP.

2.3.10 Boostrap

Menurut Husein (2013), Boostrap merupakan *framework* ataupun *tools* untuk membuat aplikasi *web* ataupun *web responsive* secara tepat, mudah, dan gratis. Boostrap terdiri dari CSS (*Cascending Style Sheet*) dan HTML (*Hypertext Markup Language*) untuk menghasilkan *Grid, Layout, Typhography, Table, Form, Navigation,* dan lain-lain. Di dalam Boostrap juga sudah terdapat *JQuery plugins* untuk menghasilkan komponen UI yang menarik seperti *Transitions, Modal, Dropdown, Scrollspy, Tooltip, Tab, Popover, Alert, Button, Carousel*, dan lain-lain.

2.3.11 CSS

Menurut Hartono (2013), CSS (*Cascading Style Sheet*) adalah kumpulan aturan-aturan pemformatan yang mengontrol tampilan konten dalam sebuah halaman *web*. Terdapat tiga jenis CSS, yaitu:

a. *Inline style sheet* : cukup menambahkan atribut *style* di *tag* yang ingin kita berikan pemformatan.
b. *Internal style sheet* : meletakkan aturan pemformatan dengan CSS dibagi <head> dari *html* dengan tambahan *tag* <style>.
c. *External style sheet* : memisahkan antara file CSS dengan file HTML-nya.

Berikut adalah contoh *script* yang menggunakan CSS dapat dilihat pada Gambar 2.16.

```
<html>
<head>
<meta charset="utf-8">
<title>Layout Website Sederhana</title>
<link href="style.css" type="text/css" rel="stylesheet">
</head>
<body>
<div class="wrap">
    <div class="header">
            <h1>HEADER</h1>
    </div>
        <div class="nav">
            MENU
    </div>
    <div class="main">
        <div class="content">
            <h2>Content</h2>
            <p>Content Web</p>
        </div>

        <div class="sidebar">
            <h2>Righr Sidebar</h2>
        </div>

        <div class="clear"></div>

    </div>

    <div class="footer">
        <center><p>Copyright &copy; 2017
www.modulkomputer.com</p></center>
    </div>

</div>
</body>
</html>
```

Gambar 2.16 Contoh *Script* Dengan CSS.

2.3.12 Pengujian Sistem Dengan *BlackBox*

Pengujian menggunakan sekumpulan aktifitas validasi, dengna pendekatan *blackbox testing*. Menurut Sukamto dan Shalahuddin (2011), *blackbox testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian *blackbox* dilakukan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai spesifikasi yang dibutuhkan.

Menurut Pressman (2010) tujuan dari pengujian adalah untuk menemukan dan memperbaiki sebanyak mungkin kesalahan dalam program sebelum menyerahkan program kepada pengguna. Salah satu pengujian yang baik adalah pengujian yang memiliki probabilitas tinggi dalam menemukan kesalahan.

2.3.13 Skala Likert

Menurut Sugiyono (2010) skala likert digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang fenomena sosial. Untuk perhitungan interval penilaian dalam Likert dapat dilihat pada tabel 2.7.

Tabel 2.7 Interval Penilaian

Jawaban	Skor	Interval	Interprestasi
STS	1	$1.00 \leq x < 1.75$	Sangat Tidak Setuju
TS	2	$1.75 \leq x < 2.50$	Tidak Setuju
S	3	$2.50 \leq x < 3.25$	Setuju
SS	4	$3.25 \leq x < 4.00$	Sangat Setuju

Cara perhitungan untuk menyatakan interpretasi dihitung dengan menentukan nilai rata-rata dari setiap pertanyaan. Berikut adalah penulis berikan contoh hasil perhitungan yang didapat dari 30 responden;

STS = 2 responden dengan skor 1

TS = 3 responden dengan skor 2

S = 15 responden dengan skor 3

SS = 10 responden dengan skor 4

Jumlah responden = 30 responden

Nilai rata-rata = {(2x1) + (3x2) + (15x3) + (10+4) }/40

= { 2 + 6 + 45 + 40 }

= 93/30

= 3.1

Dari hasil perhitungan diatas menjukkan hasil dengan nilai 3.1 dan dapat diinterpestasikan bahwa responden setuju.