BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Komputer

Menurut Kadir (2013:2), “Komputer merupakan peralatan elektronik yang biasa dipakai orang untuk membantu pelaksanaan pekerjaan”. Sedangkan menurut Puspitosari (2013:1), “Komputer adalah sebuah alat yang digunakan untuk mengolah data menurut perintah yang telah dirumuskan. Komputer dapat didefinisikan sebagai sekumpulan alat elektronik yang saling terkoordinasi satu sama lain sehingga dapat menerima data, kemudian mengolah data, dan pada akhirnya akan menghasilkan suatu keluaran yang berupa informasi (Input > Proses > Ouput)”.

Berdasarkan pengertian diatas maka dapat simpulkan bahwa Komputer adalah sebuah peralatan elektronik yang digunakan untuk mengolah data dan memudahkan pekerjaan.

2.1.2 Website

Menurut Abdulloh (2018:1), “Website atau disingkat web dapat diartikan sekumpulan halaman yang terdiri atas beberapa laman yang berisi informasi dalam bentuk data digital, baik berupa teks, gambar, video, audio, dan animasi lainnya yang disediakan melalui jalur koneksi internet”.

Menurut (Juansya 2015), “Aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus computer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan”.

Berdasarkan pengertian diatas aplikasi adalah perangkat lunak yang melaksanakan suatu fungsi bagi pengguna aplikasi yang berisi instruksi yang dapat diubah dengan mudah.

2.2 Teori Khusus

2.2.1 Unified Modeling Language (UML)

2.2.1.1 Pengertian Unified Modeling Language (UML)

Menurut Shalahuddin (2013:133), "UML*(Unified Modeling Language)* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

Pada perkembangan teknik pemprograman berorentasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemprograman berorentasi objek, yaitu *Unified Modelling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasi, menggambarkan, membangun, dan mendokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan *UML* tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya *UML* paling banyak digunakan pada metodologi berorientasi objek. (Rosa A.S dan M. Shalahudin, 2018:133).

2.2.1.2. Sejarah *UML*

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS. Secara aktual, penggunaan bahasa pemprograman berorentasi objek pada saat itu nasih terbatas, namun telah banyak menarik perhatian di saat itu. Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek.

Karena banyaknya metodologi- metodologi yang berkembang pesat saat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep, seperti

konsep *Object Modeling Technique (OMT)* dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities*, *Collaborators (CRC)* dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama Rational Software Corporation menghasilkan bahasa yang disebut dengan *Unified Modeling Language (UML).*

Pada tahun 1996, *Object Management Group (OMG)* mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 *UML* diakomodasi oleh *OMG* sehingga sampai saat ini *UML* telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya.

Secara fisik UML adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML terbaru adalah UML 2.3 yang terdiri dari 4macam spesifikasi yaitu *Diagram Interchange Specification,* UML Infrastructure, UML Superstruktur, dan Object Constraint Language (OCL). (Rosa A.S dan M. Shalahudin 2014:138).

2.2.1.3 Diagram *UML*

Rosa A.S dan M. Shalahudin (2018:140), pada *UML* terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori.

Berikut ini penjelasan singkat dari pembagian kategori tersebut:

1. *Structure diagram*
   Adalah kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. *Structure diagram* terdiri dari *class diagram, object diagram, component diagram, composite structure diagram, package diagram dan deployment diagram.*
2. *Behavior diagram*
   Adalah kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Behavior diagram terdiri dari *Use case diagram, Aktivity diagram, State Machine System.*
3. *Interaction diagram*

Adalah kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. *Interaction diagram* terdiri dari *Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram.*

2.2.1.4. *Use Case Diagram*

Menurut Rosa dan M. Shalahudin (2018:155), *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 itu.

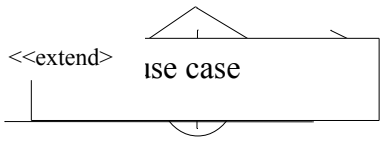
Berikut adalah simbol-simbol yang ada pada diagram *Use case:*

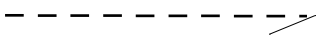
Tabel 2.1. Simbol-simbol diagram *Use case*

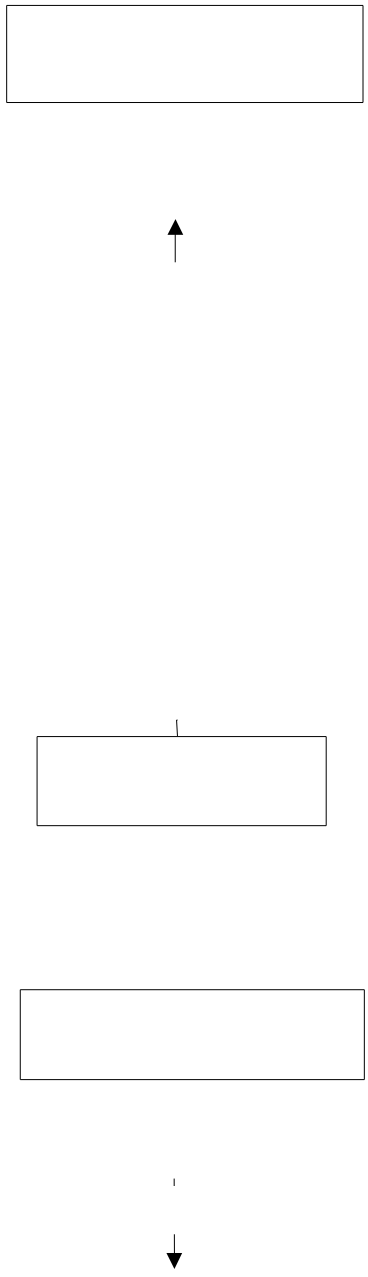
No.	Simbol	Deskripsi
1.	*Use case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan Dengan menggunakan kata kerja diawal frase nama *use case*.

2.	Aktor/*actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari jadi walaupun simbol dari aktor
3.	Assosiasi/*association*	Komunikasi antara aktor dan *use case* yang berpartisilpasi pada *use case* atau *use case* memiliki interaksi dengan aktor.

Lanjutan Tabel 2.1. Simbol-simbol diagram *Use case*

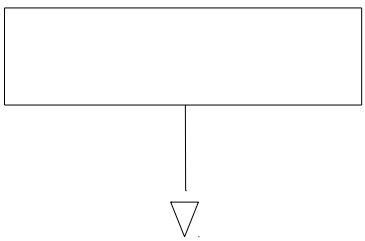
No.	Simbol	Deskripsi
4.		Ekstensi / *extend* Relasi *use case* tambahkan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahanmemiliki nama depan yang sama dengan *use case* yang ditambahkan, misal





arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend* -nya merupakan jenis yang sama dengan *use case* yang menjadi induknya.

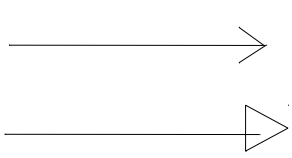
Lanjutan Tabel 2.1. Simbol-simbol diagram *Use case*

No.	Simbol	Deskripsi
5.		Generalisasi / *generalization* Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya: 

arah panah mengarah pada *use case* yang menjadi generalisasinya (umum.

6.

Menggunakan / include / *uses*



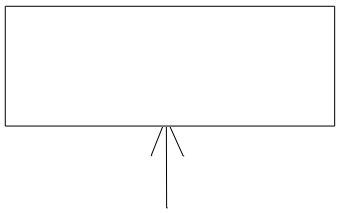
Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

Lanjutan Tabel 2.1. Simbol-simbol diagram *Use case*

No.	Simbol	Deskripsi

ada dua sudut pandang yang cukup besar mengenai include di *use case*:

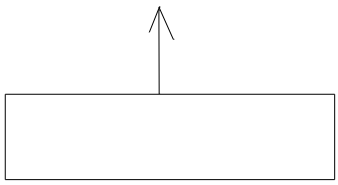
- *include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahkan dijalankan nasal pada kasus berikut



- include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:

Lanjutan Tabel 2.1. Simbol-simbol diagram *Use case*

No.	Simbol	Deskripsi



Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

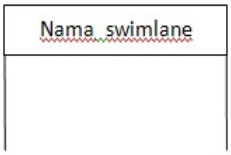
Sumber : Rosa A.S dan M. Shalahudin (2018:156)

2.3.1.5 *Activity Diagram*

Rosa dan M. Shalahudin (2018:161), diagram aktivitas atau *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu di perhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas.

Tabel 2.2. Simbol-simbol *Activity diagram*

No.	Simbol	Deskripsi
1.	Status awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas	Aktivitas yang Dilakukan sistem,
		aktivitas biasanya diawali dengan Kata kerja.

3.	Percabangan/*decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan/join	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status akhir	Status akhir yang dilakukan oleh sistem, sebuah diagram Aktivitas memiliki sebuah status akhir.
6.	Swimlane  Atau  	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber : Rosa A.S dan M. Shalahudin (2018:162)

2.3.1.6 *Class Diagram*

Rosa dan M. Shalahudin (2014:141), 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 *method* atau operasi. Berikut penjelasan atribut dan *method* :

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau *method* adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Kelas – kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau programmer dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas Main
2. Kelas yang menangani tampilan sistem
3. Kelas yang diambil dari pendefinisian Use case
4. Kelas yang di ambil dari pendefinisian data.

Berikut adalah simbol-simbol yang ada pada diagram kelas :

Tabel 2.3. Simbol-simbol *Class diagram*

No.	Simbol	Deskripsi
1.	Kelas **nama_kelas** +atribut +operasi()	Kelas pada struktur system.
2.	Antarmuka / *interface*	Sama dengan konsep *interface* dalam perograman berorientasi objek.
3.	Asosiasi / *association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4.	Asosiasi berarah / *directed association*	Relasi antarkelas dengan makna yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.

5.	Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus).
6.	Kebergantungan / *dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.
7.	Agregasi / *aggregation*	Relasi antarkelas dengan makna semua-bagian (*whole-part*).

Sumber : Rosa A.S dan M. Shalahudin (2018:146)

2.3.1.7 *Sequence Diagram*

Rosa dan M. Shalahudin (2018:165), diagram *Sequence* menggambarkan kelakuan objek pada *Use case* dengan mendeskripsikan waktu hidup objek dengan *massage* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan

maka diagram sekuen yang harus dibuat juga semakin banayak. Berikut adalah simbol-simbol yang ada pada diagram sekuen :

Tabel 2.5. Simbol-simbol *Sequence diagram*

No.	Simbol	Deskripsi
1.	Aktor Atau tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar 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 diawal frase nama aktor.
2.	Garis hidup/*lifeline*	Menyatakan kehidupan suatu objek.

Tabel 2.3. Lanjutan Simbol-simbol *Sequence diagram*

No.	Simbol	Deskripsi
3.	Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan.

4.	Waktu Aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan, yang dilakukan di dalamnya, misalnya 2: cekStatusLogin() 1: login() 3: open()
5.	Pesan tipe *create* <<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,

Tabel 2.3. Lanjutan Simbol-simbol *Sequence diagram*

No.	Simbol	Deskripsi

		Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi /metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
7.	Pesan tipe *send* 1 : masukan	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8.	Pesan tipe *retur*n 1 : keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
9.	Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri , sebaiknya jika ada *create* maka ada *destroy*.

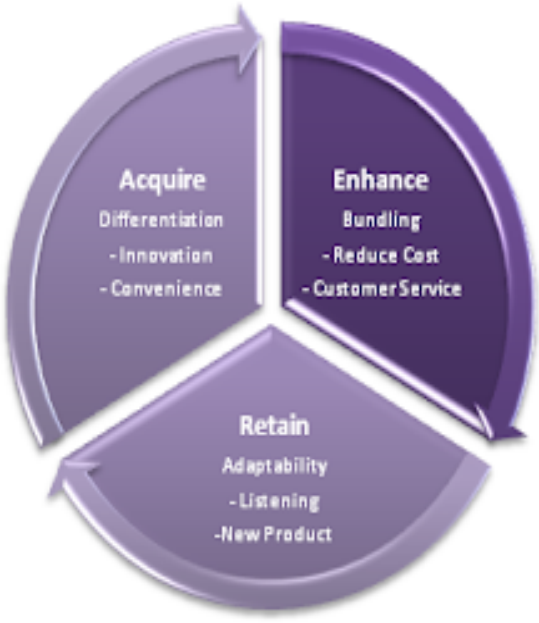
Sumber : Rosa A.S dan M. Shalahudin (2018:165)

2.3 Teori Judul

2.3.1 *Customer Relationship Management* (CRM)

Kotler dan Amstrong (2013:36) menyatakan bahwa Customer Relationship Management adalah konsep yang paling penting dalam pemasaran modern. Yang dalam arti lebih luas, CRM adalah keseluruhan proses membangun dan memelihara hubungan pelanggan yang menguntungkan dengan mengantarkan nilai dan kepuasan pelanggan yang unggul.

Customer adalah Pelanggan, Relationship adalah Hubungan, dan Management adalah Manajemen. Jadi, yang bisa saya tangkap selama 1 semester ini mengenai penjelasan CRM adalah bahwa CRM merupakan suatu bentuk usaha ataupun kegiatan yang dilakukan oleh perusahaan atau seseorang untuk menjalin hubungan baik dengan pelanggan agar pelanggan bisa tetap loyal kepada perusahaan. Banyak cara yang bisa digunakan untuk untuk mempertahankan pelanggan, menarik perhatian pelanggan kembali, atau bahkan meningkatkan keinginan pelanggan.



Sumber: Kalakota, Ravi., and Robinson, Marcia. 2000. Ebusiness. Roadmap for Success. USA: Addison Wesley.

Ada 3 fase dari CRM yang perlu diketahui, yaitu:

1. Acquire

Di fase ini lebih ke arah bagaimana cara perusahaan untuk menarik ataupun mendapatkan pelanggan.

2. Enhance

Pada fase ini, perusahaan mencoba melakukan usaha agar pelanggan yang sudah ada dapat naik ke level selanjunya. Usaha yang biasa dilakukan dalam fase ini adalah dilakukannya bundling suatu produk maupun jasa. Pada fase ini seakan-akan pelanggan dianggap lebih istimewa dengan adanya ini.

3. Retain

Pada fase ini lebih dikenal juga dengan fase mempertahankan pelanggan. Disini juga perusahaan mencoba untuk menawarkan produk yang lebih baru dimana produk baru ini pasti lebih baik dari pada produk yang lama kepada pelanggan.

2.3.2 Aplikasi

Menurut (Juansya 2015), "Aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus computer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan".

Berdasarkan pengertian diatas Aplikasi adalah perangkat lunak yang melaksanakan suatu fungsi bagi pengguna aplikasi yang berisi instruksi yang dapat diubah dengan mudah.

2.3.3 Pengertian Iklan

Iklan adalah segala bentuk penyajian dan promosi ide, barang atau jasa secara nonpersonal oleh suatu sponsor tertentu yang memerlukan pembayaran. (Kotler dan Amstrong, 2012 : 454).

Berdasarkan pengertian diatas iklan adalah semua informasi yang disampaikan kepada masyarakat berdasarkan UU Pers yang telah ditetapkan.

2.3.4 Pengertian Penerapan

Wahab dalam Van Meter dan Van Horn (2008:65) "Penerapan merupakan tindakan-tindakan yang dilakukan baik oleh individu-individu atau kelompok-kelompok yang diarahkan pada tercapainya tujuan yang telah digariskan dalam keputusan". Dalam hal ini, penerapan adalah pelaksanaan sebuah hasil kerja yang diperoleh melalui sebuah cara agar dapat dipraktekkan kedalam masyarakat.

Berdasarkan pengertian diatas maka dapat disimpulkan bahwa penerapan adalah tindakan yang dilakukan oleh seseorang yang telah diarahkan untuk tujuan yang ditetapkan dalam keputusan.

2.4 Teori Program

2.4.1 Bootstrap

Bootstrap merupakan salah satu framework CSS paling populer dari sekian banyak framework CSS yang ada. Bootstrap memungkinkan desain sebuah web menjadi responsif sehingga dapat dilihat dari berbagai macam ukuran *device* dengan tampilan tetap menarik (Abdulloh,2018:261).

2.4.2 CSS (*Cascading Style Sheet*)

CSS adalah singkatan dari *Cascading Style Sheet* yaitu dokumen web yang berfungsi mengatur elemen HTML dengan berbagai property yang tersedia sehingga dapat tampil dengan berbagai gaya yang diinginkan (Abdulloh,2018:45).

2.4.3 PHP

Menurut Arief, (2011:43) PHP adalah Bahasa server-side –scripting yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan server-side-scripting maka sintaks dan perintah-perintah PHP akan diesksekusi diserver kemudian hasilnya akan dikirimkan ke browser dengan format HTML.

Hidayatullah (2017:223) menyatakan, "PHP Hypertext Preprocessor atau disingkat dengan PHP ini adalah suatu bahasa scripting khususnya digunakan untuk web delevopment. Karena sifatnya yang server side scripting, maka untuk menjalankan PHPharus menggunakan web server.

2.4.4 MySQL

Nugroho (2004:29), “MySQL (My Structure Query Language) atau yang biasa dibaca “mai-se-kuel’’adalah sebuah program pembuat database yang bersifat open source, artinya siapa saja boleh menggunakannya.”

2.4.5 *Database*

Basis data (*database*) dapat didefinisikan sebagai himpunan kelompok data saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah (Hidayatullah dan Kawistara, 2014:147).

2.4.6 HTML (*Hypertext Markup Language*)

Hypertxt Markup Language (HTML) adalah bahasa standard yang digunakan untuk menampilkan halaman *web*. Yang bisa dilakukan HTML yaitu (Hidayatullah dan Kawistara, 2014:13).

1. Mengatur tampilan dari halaman *web* dan isinya.
2. Membuat tabel dalam halamn *web*.
3. Mempublikasikan halaman *web* secara *online*.
4. Membuat form yang bisa digunakan untuk menangani registrasi dan transaksi via *web*.
5. Menambahkan objek-objek seperti citra, audio, video, animasi, *java applet* dalam halaman *web*.
6. Menampilkan area gambar (canvas) di *browser*.

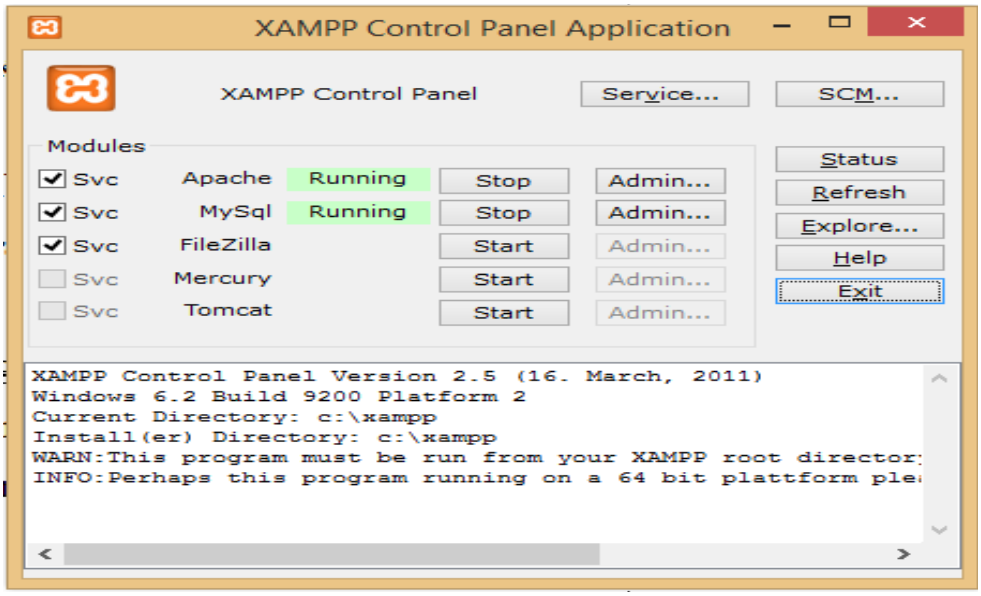
2.5.7 XAMPP

Nugroho (2013 : 1) menjelaskan, XAMPP adalah paket program web lengkap yang dapat Anda pakai untuk belajar pemrograman web, khususnya PHP dan MySQL, paket ini dapat didownload secara gratis dan legas.”

Dibawah folder utama xampp, terdapat beberapa folder penting yang perlu diketahui. Penjelasan fungsinya sebagai berikut:

Tabel 2.4 Folder-Folder Penting Dalam XAMPP

No	Nama Folder	Fungsi
1.	Apache	Folder utama dari *Apache Webserver*.
2.	Htdocs	Folder utama untuk menyimpan data-data latihan *web*, baik PHP maupun HTML biasa. Pada folder ini, anda dapat membuat subfolder sendiri untuk mengelompokkan file latihannya. Semua folder dan file program di htdocs bisa diakses dengan mengetikkan alamat http://localhost/ di browser.
3.	Manual	Berisi subfolder yang di dalam terdapat manual program dan *database*, termasuk manual PHP dan MYSQL.
4.	Mysql	Folder utama untuk *database* MYSQL server. di dalamnya terdapat subfolder data (lengkapnya: C:\xampp\mysql\data) untuk merekam semua nama *database*, serta subfolder bin yang berisi *tools* klien dan server MYSQL.
5.	Php	Folder utama untuk program PHP.



Gambar 2.2 Tampilan XAMPP

2.5 Metode *Extreme Programming* (XP)

Metode *Extreme Programming* (XP) merupakan salah satu metode *Agile* yang cukup populer. XP dikatakan sesuai untuk pengembangan sistem informasi atau perangkat lunak berskala kecil sampai dengan menengah. Satu team XP idealnya terdiri dari tiga sampai sepuluh orang pemrograman (D.E. Avison & Fitzgerald, 2006). Jumlah pemrogram tersebut ideal untuk pengembangan sistem yang bersifat dinamis dan yang membutuhkan maupun fitur sistem informasinya berubah dengan cepat.

Extreme Programming (XP) didefinisikan sebagai sub disiplin pengembangan perangkat lunak yang mengutamakan niali-nilai kesederhanaan, komunikasi, umpan balik, dan keberanian. (Jeffrias, 2001).

Jika dilihat secara seksama, maka dalam XP terdapat tahapan-tahapan sebagai berikut (D.E. Avison & Fitzgerald, 2006):

1. *Perencanaan.*
   Dalam perencanaan disusun ruang lingkup aplikasi, prioritas fungsi dan fitur yang harus dikembangkan, anggota team yang terlibat, apa yang harus dikerjakan untuk tiap tahapan secara terperinci, estimasi kebutuhan biaya, jadwal pengerjaan, dan standar kualitas yang diharapkan (termasuk rencana pengujian). Aspek bisnis dan

teknis coba diseimbangkan. Perencanaan didasarkan pada *User Stories* dan dilengkapi dengan penetapan skala prioritas kebutuhan pengguna.

2. *Perancangan*.
   Prinsip perancangan yang digunakan dalam XP adalah kesederhanaan, umpan balik dan keberanian, dan memungkinkan adanya perubahan secara bertahap. Kesederhanaan berarti pengembang diharapkan menggunakan langkah yang paling mudah dalam mewujudkan suatu fungsionalitas aplikasi. Proses perancangan wajib melibatkan para pemangku kepentingan. Keterlibatan para pemangku kepentingan dianggap dapat membantu menemukan solusi terbaik dan menumbuhka rasa memiliki aplikasi pada para pemangku kepentingan tersebut.
3. *Penulisan Kode Sumber.*
   Di sinilah konsep pemrograman berpasangan diterapkan dan digunakan secara intensif. Pengujian aplikasi dilakukan menggunakan data dari pemrogram maupun dari pengguna. Pengguna melakukan uji coba dan memberikan umpan balik secara cepat. Umpan balik tersebut menjadi masukan bagi team pengembang untuk menyempurnakan aplikasi. Pengguna harus dapat memberikan kontribusi langsung selama proses penulisan kode sumber aplikasi berlangsung. Komitmen waktu pengguna diharapkan dapat diberikan secara penuh.
4. *Produksi.*
   Keseluruhan modul yang dikembangkan diintegrasikan dan diuji secara komperehensif dalam tahapan ini. Pengujian tersebut bermaksud memastikan bahwa aplikasi yang dikembangkan berjalan dengan optimal. Tidak seperti metodologi lain, dokumentasi bukan prioritas utama.

2.6 Metode Pengujian *Black Box*

Metode pengujian digunakan untuk mengetahui fungsi yang telah ditentukan bahwa suatu sistem telah dirancang dapat menunjukan bahwa masing-masing fungsi sepenuhnya beroperasi. Pengujian kotak hitam *(black box),* juga disebut pengujian prilaku, berfokus pada persyaratan fungsional perangkat lunak.

Pengujian kotak hitam *(black box)* berupaya untuk menemukan kesalahan dalam kategori berikut: (1) fungsi yang salah atau hilang, (2) kesalahan dalam struktur data atau akses basis eksternal, (4) kesalahan perilaku atau kinerja, dan (5) kesalahan inisialisasi dan penghentian (Pressman, 2012:597).