LEMBAR PENGESAHAN

Proposal Tugas Akhir dengan Judul:

MEMBANGUN SISTEM INFORMASI PELATIHAN MAHASISWA BERBASIS *WEB* MENGGUNAKAN *FRAMEWORK CODEIGNITER* PADA *INFOCOM CAREER DEVELOPMENT CENTRE* (i-CDC) UNIVERSITAS TELKOM

Oleh

FERDY F SIBUEA

1106090038

Telah disetujui dan disahkan untuk mengikuti Seminar Proposal Tugas Akhir

Program Studi Strata 1 Sistem Informasi

Fakultas Rekayasa Industri Universitas Telkom

Bandung, 2 Juli 2014

Menyetujui

Pembina

Warih Puspitasari, S.Psi, M.Psi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

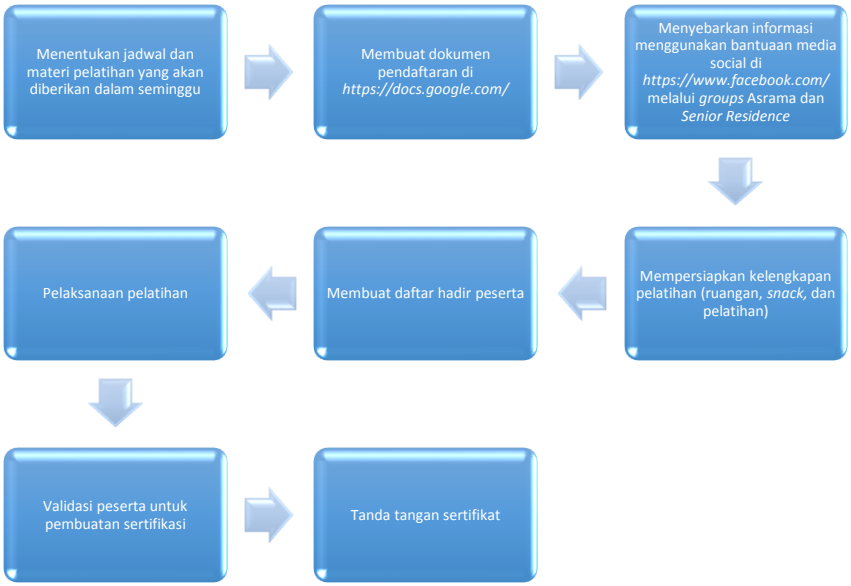
BAB 1
PENDAHULUAN

Bab ini mendeskripsikan tentang latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan penelitian dan sistematika penulisan.

1.1 Latar Belakang

Infocom Career Development Centre (i-CDC) Universitas Telkom merupakan bagian kemahasiswaan yang khusus menangani pengembangan karir mahasiswa Universitas Telkom. Salah satu bentuk pengembangan karir yang dilakukan oleh i-CDC adalah memberikan pelatihan kepada mahasiswa. Jenis-jenis pelatihan yang diberikan kepada mahasiswa antara lain *Personal Goal Setting*, *Self-Awareness*, *Basic Study Skill*, dan *MOCK Interview*, selain itu ada beberapa pelatihan yang dilakukan bersama pihak perusahaan. *Personal Goal Setting*, *Basic Study Skill*, dan *Self-Awareness* dikhususkan untuk mahasiswa baru, sedangkan untuk mahasiswa aktif lainnya hanya pelatihan umum. Untuk alumni sendiri biasanya dilakukan *profiling* alumni dan *MOCK Interview.*

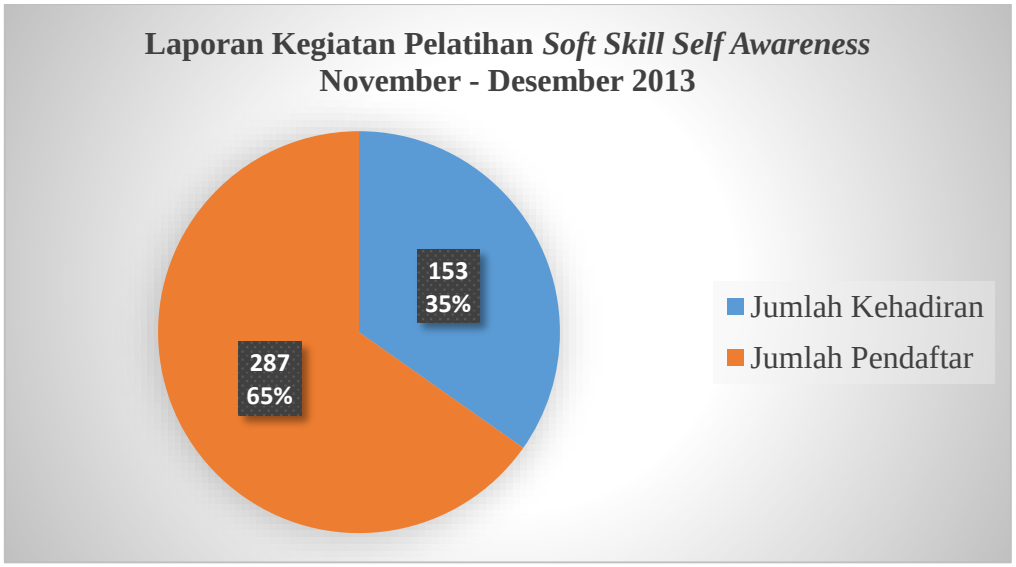
Prosedur yang digunakan i-CDC Universitas Telkom dalam memberikan pelatihan kepada mahasiswa adalah sebagai berikut.



Gambar 1. 1 Prosedur Pelatihan di i-CDC Universitas

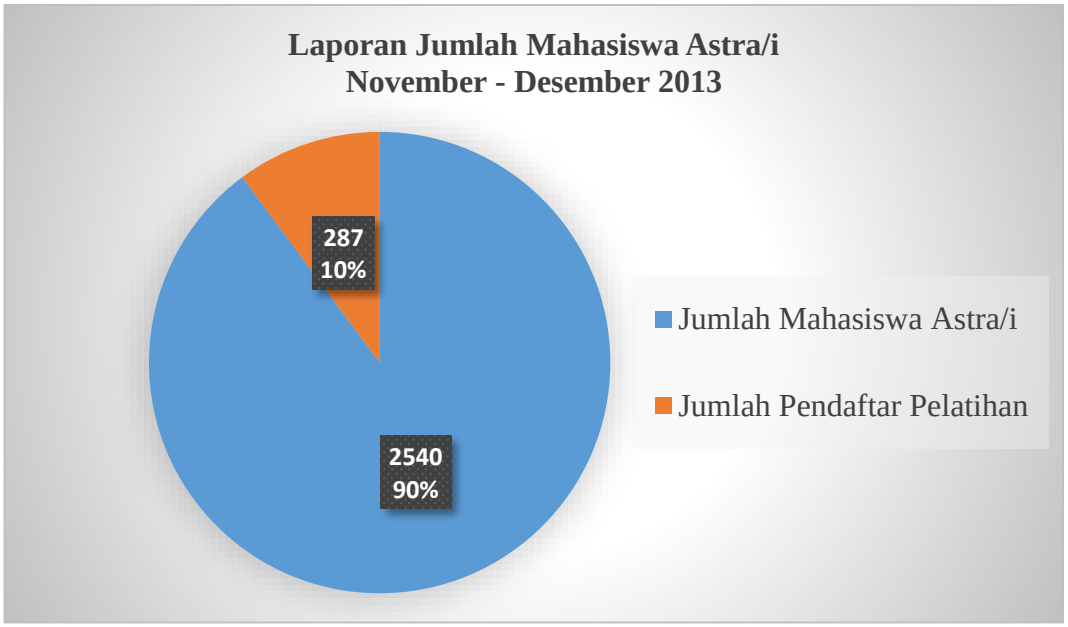
Sebagian prosedur di atas sudah dilakukan secara *online*, hanya saja secara teknis masih tergolong kurang efektif dan efisien. Dapat dilihat pada pembuatan dokumen pendaftaran menggunakan *Google Doc* masih terdapat banyak kelemahan. Kelemahan yang paling utama adalah terjadinya *redundancy* data ketika pendaftar memasukkan data pendaftaran lebih dari satu kali. *Redundancy* data ini sangat berpengaruh terhadap batasan jumlah pendaftar, dalam artian ketika suatu pelatihan akan dilaksanakan maka akan ada batasan jumlah mahasiswa yang diperbolehkan untuk mengikuti pelatihan. Dalam hal ini ketika terjadi *redundancy* data maka secara teknis akan mengurangi kesempatan kepada calon peserta pelatihan lain untuk mengikuti kegiatan pelatihan. Misalkan dalam suatu pelatihan akan dibatasi bahwa jumlah mahasiswa yang diperbolehkan mendaftar maksimal sebanyak 30 peserta maka *google doc* akan ditutup ketika jumlah pendaftar sudah mencapai 30. Penutupan dokumen pendaftaran ini dilakukan secara manual dengan cara memeriksa *google doc* sesering mungkin untuk menghindari terjadinya kelebihan jumlah calon peserta. Apabila *redundancy* yang terjadi ada tiga mahasiswa yang memasukan data pendaftaran sebanyak dua kali maka akan ada tiga *slot* kosong yang seharusnya dapat diisi oleh mahasiswa lain.

Penjelasan diatas masih dalam bagian pendaftaran, sedangkan dalam pelaksanaanya sendiri masih terdapat masalah lain. Berikut merupakan laporan kegiatan pelatihan *soft skill Self Awareness* November - Desember 2013.



Gambar 1. 2 Laporan Kegiatan Pelatihan *Soft Skill Self Awareness*

Sumber: Laporan Triwulan i-CDC September - Desember 2013



Gambar 1. 3 Laporan Jumlah Mahasiswa Astra/i

Sumber: Laporan Triwulan i-CDC September - Desember 2013

Berdasarkan data mahasiswa yang mendaftar dan melaksanakan pelatihan dapat disimpulkan bahwa tidak semua mahasiswa yang melakukan pendaftaran mengikuti kegiatan pelatihan yang diadakan. Alasan utamanya adalah kebanyakan mahasiswa yang mendaftar lupa jadwal pelatihan yang telah ditentukan. Alasan lain adalah mahasiswa berhalangan hadir karena hal tertentu akan tetapi tidak melakukan konfirmasi terhadap pihak pelaksana pelatihan sehingga akibatnya akan menghabiskan *resource*. Oleh karena itu akan dibangun sebuah sistem informasi pelatihan mahasiswa berbasis *web* pada i-CDC Universitas Telkom dengan menggunakan *framework codeigniter*. *Website* ini nantinya akan dilengkapi fitur-fitur untuk memberikan konfirmasi terhadap status kehadiran peserta pelatihan dan *reminder* terhadap peserta dan pelaksana pelatihan. *Reminder* yang diharapkan dapat diterapkan yaitu SMS *gateway* dan notifikasi melalui *e-mail.*

Website yang akan dibangun nantinya akan menggunakan *framework codeigniter* dengan tema *simplicity*. Fitur-fitur utama yang akan disediakan pada *website* yang akan dibangun berupa pembatasan jumlah pendaftaran pelatihan, laporan dan riwayat pelatihan mahasiswa, kritik dan saran terhadap performa mahasiswa dalam menjalani pelatihan, kalender pelatihan, *sharing* foto-foto hasil pelatihan, dan *template* khusus untuk mencetak sertifikat pelatihan.

Metode penelitian yang akan digunakan penulis dalam penelitian ini adalah metode *prototype.* Untuk penjelasan lebih lanjut mengenai metodologi penelitian akan dijelaskan pada bab-bab selanjutnya.

Selain pelatihan mahasiswa, i-CDC Universitas Telkom juga memberikan fasilitas konseling kepada seluruh mahasiswa dengan tujuan untuk membantu mahasiswa dalam menghadapi kendala-kendala perkuliahan dengan solusi-solusi yang dianggap tepat. Prosedur pelaksanaan konseling ini juga masih dilakukan secara manual seperti pada pelatihan mahasiswa. Untuk itu, dilakukan penelitian lain untuk membangun sistem informasi konseling yang dilakukan oleh Christine Purnamasari Sibarani dengan nomor induk mahasiswa (NIM) 1106100043. Kedua sistem informasi ini akan digabung menjadi satu untuk membantu kinerja i-CDC Universitas Telkom.

1.2 Perumusan Masalah

Berdasarkan latar belakang penelitian di atas, maka rumusan masalah dalam penelitian ini adalah sebagai berikut.

1. Bagaimana membangun sistem informasi pelatihan mahasiswa dengan menggunakan *framework codeigniter*?
2. Bagaimana cara pemanfaatan SMS *gateway* dan notifikasi *e-mail* sebagai *reminder* terhadap peserta dan *trainer* pelatihan?
3. Bagaimana keuntungan yang diharapkan dari pemanfaatan *website* pelatihan mahasiswa?

1.3 Tujuan Penelitian

Berdasarkan latar belakang penelitian di atas, maka tujuan penelitian dalam penelitian ini adalah sebagai berikut.

1. Membangun sistem informasi pelatihan mahasiswa dengan menggunakan *framework codeigniter*.
2. Memanfaatkan SMS *gateway* dan notifikasi *e-mail* sebagai *reminder* terhadap peserta dan *trainer* pelatihan.
3. Mengetahui keuntungan dari pemanfaatan *website* pelatihan mahasiswa.

1.4 Manfaat Penelitian

Manfaat penelitian yang diharapkan adalah sebagai berikut.

1. Mempermudah pihak i-CDC Universitas Telkom dalam manajemen data peserta pelatihan mahasiswa.
2. Mengurangi terjadinya *redundancy* data pada saat pendaftaran pelatihan.
3. Mempermudah pihak i-CDC dan mahasiswa Universitas Telkom dalam menyebarkan dan memperoleh informasi berkaitan dengan pelatihan mahasiswa yang akan dilaksanakan.

1.5 Batasan Penelitian

Batasan-batasan penelitian dalam membangun sistem informasi pelatihan mahasiswa ini adalah sebagai berikut.

1. Akses sistem ini diberikan kepada pihak i-CDC Universitas Telkom dan mahasiswa.
2. Aplikasi ini dapat mengirimkan *reminder* kepada *trainer* dan peserta pelatihan melalui bantuan SMS *gateway* dan notifikasi *e-mail.*

1.6 Sistematika Penulisan

Sistematika penulisan dalam penyusunan Tugas Akhir Membangun Sistem Informasi Pelatihan Mahasiswa Berbasis Web Menggunakan *Framework CodeIgniter* pada *Infocom Career Development Centre* (i-CDC) Universitas Telkom adalah sebagai berikut.

BAB 1 PENDAHULUAN

Bab ini berisi tentang latar belakang, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan penulisan, dan sistematika penulisan yang mengacu pada pembangunan *website* pelatihan mahasiswa dengan menggunakan *framework codeigniter.*

BAB 2 LANDASAN TEORI

Bab ini berisi tentang teori dan prinsip-prinsip yang digunakan sebagai bahan referensi penulis dalam membangun *website* pelatihan mahasiswa ini.

BAB 3 METODOLOGI PENELITIAN

Bab ini berisi tentang metode yang digunakan dalam melakukan analisis penelitian terhadap pembangunan *website* pelatihan mahasiswa.

BAB 2

LANDASAN TEORI

Bab ini akan membahas tentang teori-teori yang berkaitan dengan penyusunan tugas akhir ini. Teori-teori ini diambil dari berbagai sumber yang dapat dijadikan referensi dalam penulisan penelitian ini. Beberapa teori yang akan dibahas pada bab ini adalah sebagai berikut.

2.1 Penelitian Terdahulu

Dalam penelitian sebelumnya, Mita Hanifah Rachim sudah membangun sebuah sistem informasi untuk i-CDC Universitas Telkom, hanya saja sistem informasi yang dibangun terfokus pada konseling. Sistem informasi yang dibangun juga menggunakan fitur SMS *Gateway* sebagai pengingat (*reminder*) terhadap konselor dan mahasiswa yang akan melakukan konseling. Sistem informasi ini sudah dapat memenuhi kebutuhan i-CDC dalam bidang konseling. Akan tetapi, sistem informasi ini belum diimplementasikan pada i-CDC Universitas Telkom.

Berdasarkan penelitian terdahulu yang telah dibangun oleh Mita Hanifah Rachim, penulis ingin menambahkan fitur pelatihan mahasiswa ke dalam sistem informasi. Hal ini dilakukan penulis mengingat bahwa pelatihan mahasiswa juga termasuk ke dalam bagian kerja dari i-CDC Universitas Telkom. Untuk lebih meringankan kinerja dari i-CDC Universitas Telkom, penulis berkeinginan untuk menggabungkan semua bagian kerja dari i-CDC Universitas Telkom ke dalam sebuah sistem informasi berbasis *web*, dalam hal ini sistem informasi yang akan dibangun adalah konseling dan pelatihan mahasiswa. Sedangkan untuk fitur-fitur lainnya diharapkan dapat ditambahkan lagi pada penelitian-penelitian selanjutnya.

2.2 Sistem Informasi

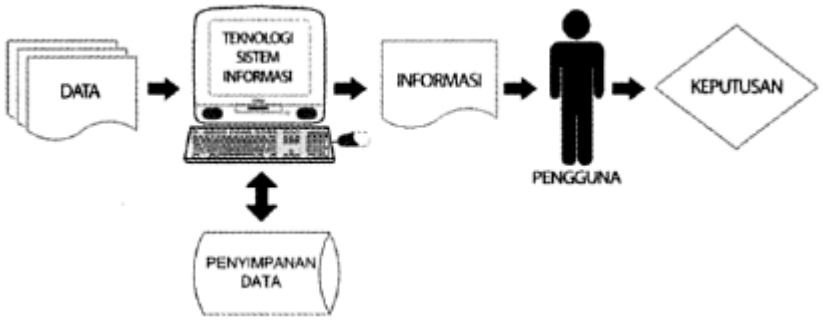
Sistem dapat diartikan sebagai serangkaian komponen-komponen yang saling berinteraksi dan bekerja sama untuk mencapai tujuan tertentu (Bonnie S, 2008). Sistem juga dapat diartikan sebagai sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan (Raymond, 2004). Berdasarkan pengertian tersebut dapat disimpulkan bahwa terdapat tiga elemen penting yang terkandung dalam sistem, yaitu:

- Rangkaian komponen
- Interaksi dan kerja sama
- Tujuan

Sedangkan informasi merupakan hasil pemrosesan data (fakta) menjadi suatu yang bermakna dan bernilai untuk pengambilan keputusan (Bonnie S, 2008). Suatu informasi dapat dikatakan berkualitas apabila memiliki karakteristik-karakeristik sebagai berikut.

- Relevan (sesuai dengan kebutuhan pengguna)
- Akurat (dapat dipercaya)
- Lengkap
- Tepat waktu
- Mudah dipahami (tidak membingungkan)
- Dapat diakses ketika dibutuhkan

Dengan demikian, sistem informasi dapat diartikan sebagai serangkaian komponen berupa manusia, prosedur, data, dan teknologi yang digunakan untuk melakukan sebuah proses untuk menghasilkan informasi yang bernilai dalam pengambilan keputusan (Bonnie S, 2008).



Gambar 2. 1 Interaksi Komponen-komponen Sistem Informasi

Sumber: Inra (2013: 8) Pembangunan Sistem Informasi Perpustakaan dengan Pemanfaatan SMS *Gateway* sebagai Sarana Penunjang Informasi (Studi Kasus SMS BPI 1 Bandung)

2.3 *Framework*

Framework merupakan suatu kerangka kerja yang di dalam bidang pemrograman kumpulan kelas (*class)* dan fungsi (*function, method)* yang disusun secara sistematis berdasarkan kegunaan atau fungsionalitas tertentu untuk mempermudah pembuatan atau pengembangan suatu aplikasi. Secara umum, *framework* yang banyak digunakan saat ini adalah *framework* yang menggunakan konsep *Object-Oriented Programming* (OOP). Keuntungan yang diperoleh dari penggunaan *framework* ini adalah waktu kerja yang menjadi lebih efisien dalam penulisan kode program (*coding*) dan pengaturan berkas-berkas kode (*code*) karena tidak perlu menuliskan kode program untuk fungsionalitas tertentu yang sudah tersedia. Berkas kode dapat disusun secara sistematis sesuai dengan struktur yang ditawarkan oleh *framework* tersebut (Sidik, 2012).

Berikut beberapa perbandingan dari *famework* php yang banyak digunakan.

PHP *Framework*	PHP4	PHP5	MVC	*Multiple* DB's	ORM	DB *Objects*	*Templates*	*Caching*	*Validation*	Ajax	Auth *Module*	*Modules*	EDP
Akelos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ash.MVC	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-
PHPDevShell	-	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	-
PhpOpenbiz	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-
Prado	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP	✓	✓	✓	✓	-	✓	✓	-	✓	✓	✓	✓	✓
Seagull	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Symfony	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
WACT	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	-

WASP	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	-
Yii	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ZooP	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-

Tabel 2. 1 Perbandingan Beberapa *Framework* PHP

www.phpframeworks.com

Keterangan:

1. MVC: Menunjukkan apakah *framework* dilengkapi dengan dukungan *inbuilt* untuk setup *Model-View-Controller*.
2. *Multiple* DB's: Menunjukkan apakah *framework* mendukung beberapa *database* tanpa harus mengubah apa pun.
3. ORM: Menunjukkan apakah *framework* mendukung *mapper object-record*, biasanya sebuah implementasi *ActiveRecord*.
4. DB *Objects*: Menunjukkan apakah *framework* ini meliputi objek *database* lain, seperti *TableGateWay*.
5. *Template*: Menunjukkan apakah *framework* memiliki mesin *template inbuilt*.
6. *Caching*: Menunjukkan apakah *framework* ini meliputi objek *caching* atau cara-cara lain untuk *caching*.
7. Validasi: Menunjukkan apakah *framework* memiliki validasi *inbuilt* atau komponen penyaringan.
8. Ajax: Menunjukkan apakah *framework* dilengkapi dengan dukungan *inbuilt* untuk Ajax.
9. *Auth Module*: Menunjukkan apakah *framework* memiliki modul *built-in* untuk menangani otentikasi pengguna.
10. *Modules*: Menunjukkan apakah *framework* memiliki modul lain, seperti *parser* RSS *feed*, modul PDF maupun modul lainnya yang berguna.
11. EDP: *Event Driven Programming*.

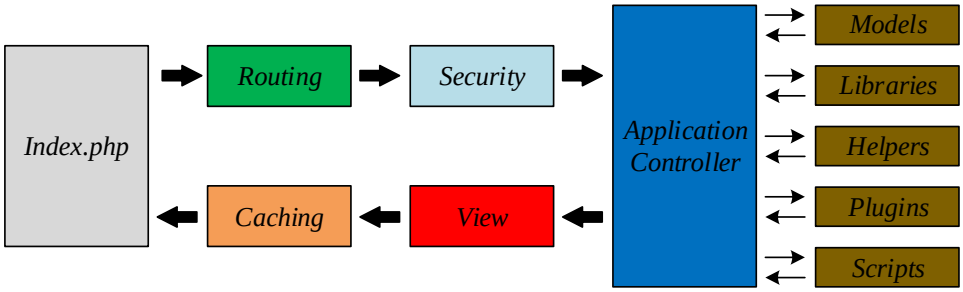
2.3.1 CodeIgniter (CI)

CodeIgniter ditulis oleh Rick Ellis dalam suatu riset kecil-kecilan. Dalam risetnya tersebut Rick Ellis menilai bahwa banyak *framework* php yang memiliki kecenderungan seperti berikut.

- Menggunakan banyak asumsi bahwa pemrograman memiliki keterampilan tinggi dan pengetahuan luas.
- Mempersyaratkan ketergantungan pada PEAR (PHP *Extention and Repository*) dan banyak aplikasi *open source* lain.
- Hanya kompatibel dengan PHP 5.
- Berukuran terlalu besar atau terlalu minimalis untuk digunakan.
- Dokumentasi yang kurang baik.

Rick Ellis membuat CI hanya berukuran kecil, dapat berjalan ringan, tetapi memenuhi fitur umum aplikasi PHP. Dengan demikian, patut disadari bahwa CI sendiri belum tentu dapat memenuhi semua kebutuhan yang diharapkan. Untuk melengkapi kekurangan ini, CI memiliki komunitas yang menyediakan tempat diskusi (*sharing*) aplikasi *third-party*, baik berupa tambahan *library*, *helper*, maupun *plugin* (Pratama, 2010).

Proses aliran data aplikasi pada sistem dapat diilustrasikan seperti berikut.



Gambar 2. 2 *Application Flowchart*

Sumber: Hakim (2010:12) Membangun *Web* Berbasis PHP dengan *Framework CodeIgniter*

2.3.2 MVC CodeIgniter

Gambaran penerapan CI kurang lebih sebagai berikut.

- ***Model***

  Bertanggung jawab untuk melakukan pengelolaan data dalam basis data. Di dalamnya biasa dituliskan perintah untuk mengambil, mengubah, menghapus, dan menambahkan data.

- ***View***

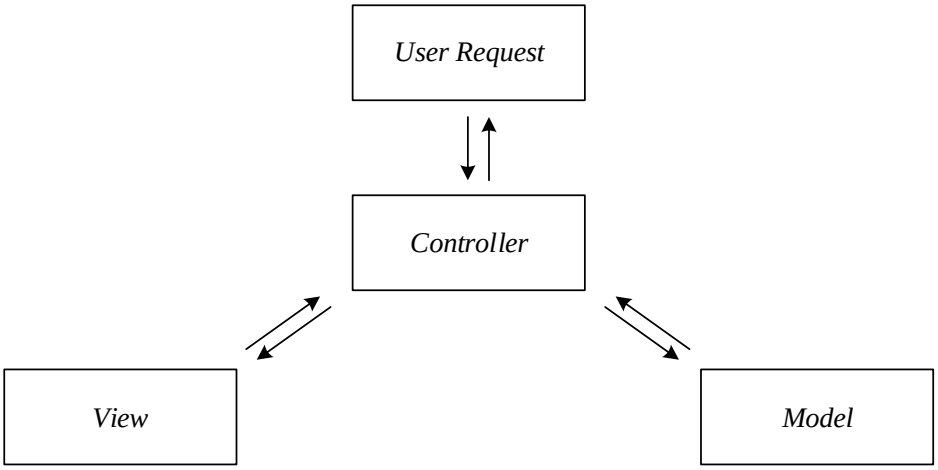
  Merupakan tempat untuk meletakkan apa yang akan ditampilkan di halaman *browser*. Sebuah berkas *view* umumnya berisi kode bahasa pemrograman sisi klien (*client-side scripting*).

- ***Controller***

  Merupakan pengatur utama hubungan antara *model, view,* dan juga sumber daya lain yang tersedia. Sumber daya ini diperoleh dari kelompok atau tipe kelas yang dapat disebut dengan elemen *framework* CI.

Penerapan ini tidak menuntut aturan yang ketat. Aplikasi berbasis CI boleh tidak menggunakan *model*. Logika *model* dapat diletakkan dalam sebuah *controller*. *View* juga boleh tidak digunakan. *View* dapat diletakkan pada *model* atau *controller* (Pratama, 2010).

Adapun alur program aplikasi berbasis *framework CodeIgniter* dapat dilihat pada gambar 2.3



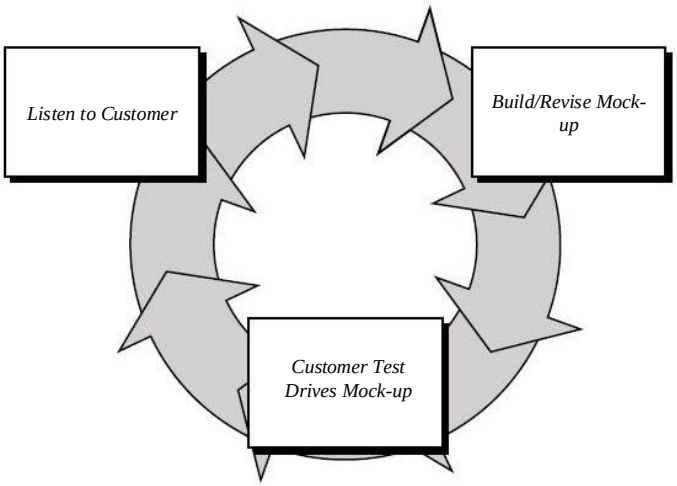
Gambar 2. 3 *Model-View-Controller*

Sumber: Hakim (2010:4) Membangun *Web* Berbasis PHP dengan *Framework CodeIgniter*

Gambar di atas menerangkan bahwa ketika *user request* datang, maka akan ditangani oleh *controller*, kemudian *controller* akan memanggil *model* jika memang diperlukan operasi *database.* Hasil dari *query* oleh *model* kemudian akan dikembalikan ke *controller.* Selanjutnya *controller* akan memanggil *view* yang tepat dan mengkombinasikannya dengan hasil *query model.* Hasil akhir dari operasi ini akan ditampilkan di *browser*.

2.4 Metode *Prototyping-Oriented Software*

Metode *Prototyping-Oriented Software* adalah proses pengembangan sebuah perangkat lunak secara berulang yang memberikan ide bagi pengembang sistem untuk menyajikan gambaran yang lengkap. *Prototype* yang dibangun pada tahap awal merupakan bagian dari produk secara keseluruhan yang mengekspresikan logika maupun fisik pada antarmuka eksternal yang ditampilkan. Konsumen menggunakan *prototype* tersebut untuk memberikan masukan kepada tim pengembang sebelum pengembangan produk dalam skala besar dimulai (Simarmata, 2010). Secara ideal metode pengembangan *prototype* berfungsi sebagai sebuah mekanisme untuk mengidentifikasikan kebutuhan dari sebuah perangkat lunak (Pressman, 2009). Metode cocok digunakan dalam pembangunan sistem informasi yang inovatif berdasarkan perspektif pemakai. Proses metode *Prototyping* ini digambarkan dalam gambar 2.4 berikut ini:



Gambar 2. 4 *Prototyping Model*

Sumber: Khosrow (2005) *Encyclopedia of Information Science and Technology* (5^{th} Volumes)

Gambar di atas menjelaskan bahwa metode *prototyping* dimulai dengan melakukan *meeting* antara pengembang (*developer*) dengan pengguna (*user*). Pada aktivitas *meeting* tersebut *user* menentukan kebutuhan yang akan dibangun pada *website* serta menentukan tujuan keseluruhan untuk perangkat lunak dan mengidentifikasi apapun persyaratan yang diperlukan. Selanjutnya *developer* membuat sebuah *quick design* mengenai aplikasi yang kemudian dapat dipresentasikan kepada *user*. *Quick design* berfokus pada representasi aspek-aspek aplikasi yang akan terlihat oleh *user*.

Keunggulan dalam menggunakan metode ini adalah:

1. Pengembang sistem dapat berkomunikasi aktif dengan pengguna, terutama dalam persamaan persepsi terhadap pemodelan sistem yang akan menjadi dasar pengembangan sistem operasionalnya.
2. Pengguna ikut terlibat aktif dan berpartisipasi dalam menentukan model sistem dan sistem operasionalnya. Dengan kata lain, metode ini akan menghasilkan sistem dengan persepektif pengguna. Diharapkan dengan menggunakan metode ini, sistem dapat diimplementasi dengan baik, sementara biaya pengembangan sistem menjadi lebih murah.
3. Sistem yang akan dibangun memiliki kualitas yang baik karena sesuai dengan kebutuhan yang ada.
4. Sesuai untuk membangun sistem besar yang tidak memiliki panduan untuk memproses kebutuhannya.

Selain keunggulan, metode ini juga memiliki kekurangan:

1. Kurangnya dokumentasi secara rinci pada setiap tahapan akan mengakibatkan deteksi dan *control* setiap langkah menjadi kurang cermat, sehingga bila terjadi kesalahan, akan cukup sulit diperbaiki. Selain itu, jika sistem yang berhasil dibangun akan dikembangkan kembali, ada kemungkinan akan mengalami kesulitan dikarenakan ide-ide yang dihasilkan lebih bersifat *insidensial*.
2. Pengguna dapat mengembangkan ide dan gagasannya ditengah proses pembangunan sistem sehingga *scope* menjadi meluas dan sulit untuk direalisasikan.

Berikut ini adalah tabel perbandingan metode *Prototype, Waterfall* dan *Iterative and Incremental*.

	Metode *Prototype*	**Metode *Waterfall***	**Metode *Iterative and Incremental***
Siklus Pengembangan	*1. Listen to Customer* *2. Build / revise mock-up* 3. *Customer test drive mock-up*	*1. Analysis* *2. Design* *3. Coding and testing* *4. Implementation* 5. *Maintenance*	*1. Inception* *2. Elaboration* *3. Construction* *4. Transition*
Kelebihan	1. Sistem yang dibangun berdasarkan perspektif pengguna. 2. Sesuai dengan sistem yang belum memiliki panduan untuk memproses kebutuhannya. 3. Pengguna ikut terlibat aktif dan berpartisipasi dalam menentukan model sistem dan sistem operasionalnya.	1. Kualitas dari sistem yang dihasilkan akan baik dikarenakan oleh pelaksanaannya secara bertahap. 2. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya.	1. Dapat mengakomodasi jika terjadi perubahan pada tahapan pengembangan yang telah dilaksanakan. 2. Dapat disesuaikan agar sistem bisa dipakai selama hidup *software computer*. 3. Pengembang dan pemakai dapat lebih mudah memahami dan bereaksi terhadap risiko setiap tahapan karena sistem terus bekerja selama proses.

Kekurangan	1. Kurangnya dokumentasi secara rinci pada setiap tahapan akan mengakibatkan deteksi dan *control* setiap langkah menjadi kurang cermat 2. Pengguna dapat mengembangkan ide dan gagasannya ditengah proses pembangunan sistem sehingga *scope* menjadi meluas	1. Diperlukan majemen yang baik, karena proses pengembangan tidak dapat dilakukan secara berulang sebelum menghasilkan suatu perangkat lunak; 2. Kesalahan kecil akan menjadi masalah besar jika tidak diketahui sejak awal pengembangan 3. *User* sulit menyatakan kebutuhan secara eksplisit sehingga tidak dapat mengakomodasi ketidakpastian pada saat awal pengembangan	1. Memerlukan alat ukur *progress* pengembangan *software* secara regular 2. Perubahan yang sering terjadi dapat merubah struktur sistem

Tabel 2. 2 Perbandingan Metode *Prototype, Waterfall* dan *Iterative and Incremental*

Sumber: Ramadhan (2013) Aplikasi *e-Commerce* Berbasis *Web* dengan Metode *Prototyping-Oriented Software* di Perusahaan Fahlevi Jaya Abad

2.5 UML (*Unified Modeling Language*)

Unified Modelling Language (UML) merupakan sebuah bahasa visual untuk menangkap pola dan desain perangkat lunak. UML dapat diterapkan ke berbagai *domain* berbeda dengan dasar pengembangan perangkat lunak (Pitman, 2005).

UML merupakan standar terbuka yang dikontrol oleh *Object Management Group* (OMG), sebuah konsorsium terbuka. Banyak perusahaan OMG dibentuk untuk membuat standar-standar yang mendukung *interoperabilitas*, khususnya *interoperabilitas* system berorientasi objek OMG yang dikenal dengan standar-standar CORBA (*Common Object Request Broker Architecture*).

Pada umumnya metode-metode yang ditujukan untuk pembangunan aplikasi berorientasi objek menggunakan UML untuk pembangunan aplikasi berorientasi objek menggunakan UML untuk memodelkan berbagai artefak dari perangkat lunak. UML adalah sekumpulan simbol dan diagram untuk memodelkan *software* dalam bentuk simbol dan diagram yang kemudian dapat diterjemahkan menjadi kode program.

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case Diagram,* untuk memodelkan proses bisnis.
2. *Activity Diagram,* untuk memodelkan perilaku *Use Case* dan objek di dalam sistem.
3. *Class Diagram,* untuk memodelkan struktur *class.*
4. *Object Diagram,* untuk memodelkan struktur objek.
5. *Conceptual Diagram,* untuk memodelkan aplikasi konsep-konsep yang ada di dalam aplikasi.
6. *Sequence Diagram,* untuk memodelkan pengiriman *message* antar objek.
7. *Collaboration Diagram,* untuk memodelkan interaksi antar objek.
8. *State Diagram,* untuk memodelkan perilaku objek dalam sistem.
9. *Component Diagram,* untuk memodelkan komponen objek.
10. *Deployment Diagram,* untuk memodelkan distribusi aplikasi.

2.5.1 *Use Case Diagram*

Use Cases merupakan cara untuk mengangkat fungsionalitas sistem dan kebutuhan kebutuhannya dalam UML. *Use Case Diagrams* terdiri atas bagian-bagian yang dinamai dari fungsionalitas (*use cases*), orang-orang atau benda-benda yang menyediakan fungsionalitas (*actors*), dan kemungkinan elemen-elemen yang bertanggungjawab untuk penerapan *use cases* tersebut (*subjects*) (Pitman, 2005).

Actor digambarkan dengan sosok manusia. *Use case* mempresentasikan operasi-operasi yang dilakukan oleh actor. *Use case* digambarkan berbentuk elips dengan nama operasi di dalamnya. *Actor* yang melakukan operasi dihubungan garis lurus ke *use case*, atau yang disebut dengan asosiasi.

Dependensi atau hubungan ketergantungan dalam diagram *Use Case* terdapat dua jenis, yaitu *include* dan *extend*. Dependensi *include* adalah hubungan antar dua *usecase* dimana yang satu memanggil yang lain. Jika pada beberapa *use case* terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan *use case* tersendiri dengan relasi dependensi setiap *use case* semula ke *use case* yang baru sehingga memudahkan pemeliharaan.

2.5.2 *Activity Diagram*

Activity diagram adalah versi UML untuk sebuah *flowchart*. *Activity diagram* digunakan untuk menganalisis proses. Sebuah *activity diagram* bukan sebuah *tool* yang sempurna untuk menganalisis masalah dari sistem. Sebagai *tool* untuk menganalisis, pemrogram tidak ingin untuk memulai memecahkan masalah di *level* teknis dengan membuat *class*, tetapi dengan menggunakan *activity diagram* untuk mengerti masalah dan menyaring proses yang terdapat dalam sistem.

Setiap *activity diagram* selalu mempunyai satu *initial state*. *Initial node* yang digambarkan dengan simbol lingkaran padat, merupakan titik yang mengawali *activity diagram*. *Activity diagram* dapat diakhiri dengan memberikan *activity final node* yang digambarkan dengan lingkaran padat dengan mempunyai cincin di bagian luarnya.

Action node adalah sesuatu yang dilakukan atau yang sedang terjadi dalam *activity diagram*. *Action node* mempunyai simbol yang hampir mirip dengan *use case* namun mempunyai bentuk yang lebih ramping dan menyerupai bujur sangkar.

Decision node disebut *decision diamonds* di dalam *flowchart*. Simbol *diamond* adalah satu elemen yang membuat *activity diagram* mengingat akan *flowchart,* yang berguna untuk memberi kondisi percabangan. Keberadaan sebuah *transition fork* untuk menggambarkan tingkah laku yang pararel atau bercabang. Sebuah *transition join* untuk mempertemukan tingkah laku yang pararel.

2.5.3 *Sequence Diagram*

Sequence Diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri antar dimensi vertikal (waktu) dan horizontal (objek-objek yang terkait).

Sequence Diagram biasanya digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal atau *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class. Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

2.5.4 *Class Diagram*

Class Diagrams adalah yang tipe diagram yang paling mendasar dan paling penting dalam UML. Diagram ini digunakan untuk menangkap hubungan statis dari perangkat lunak, dengan kata lain bagaimana benda-benda diletakkan bersama (Pitman, 2005).

Dalam *class diagram* terdapat beberapa relasi atau hubungan antar *class*, yaitu:

1. *Generalization* dan *Inheritence*

   Digunakan untuk memperlihatkan hubungan pewarisan (*inheritance*) antara unsur dalam diagram *class*. Pewarisan memungkinkan satu *class* mewarisi semua atribut, operasi, relasi dari *class* yang berada dalam hirarki pewarisannya.

2. *Association*

   Association merupakan hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui ekstensi *class* lain. Dalam notasi UML terdapat asosiasi dua arah (*bidirectional*) dan satu arah (*undirectional*).

3. *Aggregation*

   Merupakan hubungan antar *class* di mana yang satu adalah bagian dari *class* lainnya.

4. *Composition*

   Composition merupakan *aggregation* dengan hubungan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup sebagian *class* sangat bergantung pada seluruh *class* sehingga bila objek *instance* dari seluruh *class* dihapus maka objek *instance* dari sebagian *class* juga akan terhapus.

5. *Dependency*

   Dependency merupakan hubungan antar-*class* di mana sebuah *class* memiliki ketergantungan pada *class* lainnya tetapi tidak sebaliknya.

6. *Realization*

   Realization merupakan hubungan antar *class*, dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya. Biasanya *realization* digunakan untuk menspesifikasikan hubungan antara sebuah *interface* dengan *class* yang mengimplementasikan *interface* tersebut.

7. *Multiplicity*

   *Multiplicity*adalah jumlah banyaknya objek sebuah *class* yang berelasi dengan sebuah objek lain pada *class* lain yang berasosiasi dengan *class* tersebut.

2.6 *Flowchart*

Flowchart dapat diartikan sebagai representasi grafik dari langkah-langkah yang harus dilakukan dalam menyelesaikan suatu permasalahan yang terdiri atas sekumpulan simbol, dimana masing-masing simbol merepresentasikan suatu kegiatan tertentu.

Keuntungan dalam menggunakan *flowchart* dalam menggambarkan suatu kegiatan adalah:

1. *Relationship*

   Flowchart dapat memberikan gambaran yang efektif, jelas, dan ringkas tentang prosedur logika. Teknik penyajian yang bersifat grafis jelas akan lebih baik dari pada uraian-uraian yang bersifat teks khususnya dalam menyajikan logika-logika yang bersifat kompleks.
2. *Analysis*

   Dengan adanya penganalisisan yang jelas dalam diagram, maka para pembaca dapat dengan mudah melihat permasalahan atau memfokuskan perhatian pada area-area sistem Informasi tertentu.
3. *Communication*

   Karena simbol-simbol yang digunakan mengikuti suatu standar tertentu yang sudah diakui secara umum, maka *flowchart* dapat menjadi alat bantu yang sangat efektif dalam mengkomunikasikan logika suatu masalah atau dalam mendokumentasikan logika tersebut.

Adapun beberapa aturan dalam penggambaran *flowchart*, antara lain:

1. Bagan alir sebaiknya digambar dari atas ke bawah dan mulai dari bagian kiri dari suatu halaman.
2. Kegiatan di dalam bagan alir harus ditunjukkan dengan jelas.
3. Harus ditunjukkan dari mana kegiatan akan dimulai dan dimana akan berakhir.
4. Masing-masing kegiatan di dalam bagan alir sebaiknya digunakan suatu kata yang mewakili suatu pekerjaan.
5. Masing-masing kegiatan di dalam bagan alir harus didalam urutan yang semestinya.
6. Kegiatan yang terpotong dan akan disambung di tempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.
7. Gunakan simbol-simbol standar.

Flowchart terdiri dari beberapa jenis sesuai dengan prosesnya, antara lain:

1. *System Flowchart* (bagan alir sistem)

   Bagan ini menjelaskan urutan dari prosedur-prosedur yang ada dalam sistem. Bagian alir sistem menunjukan apa yang dikerjakan dalam sistem.

2. *Frocument Flowchart* (bagan alir dokumen)

   Merupakan bagan alir yang menunjukan arus dari laporan dan formulir termasuk tembusan-tembusannya.

3. *Schematic Flowchart* (bagan alir skematik)

   Bagan alir skematik hampir sama dengan bagan alir sistem tetapi terdapat perbedaan dalam hal ini menggunakan gambar-gambar untuk memudahkan komunikasi kepada orang-orang yang kurang paham dengan symbol-simbol bagan alir.

4. *Process Flowchart* (bagan alir proses)

   Bagan alir yang digunakan untuk menggambarkan proses dalam suatu prosedur.

5. *Program Flowchart* (bagan alir program)

   Bagan yang menjelaskan secara rinci langkah-langkah dari proses program.

2.7 ERD (*Entity Relationship Diagram*)

Entity Relation Diagram (ERD) merupakan model data berupa notasi grafis dalam pemodelan data konseptual yang menggambarkan hubungan antara penyimpan. Model data sendiri merupakan sekumpulan cara dan peralatan untuk mendeskripsikan data-data yang hubungannya satu sama lain serta batasan konsistensi. Model data terdiri dari model hubungan entitas dan model relasional. Diagram hubungan entitas ditemukan oleh Peter Chen dalam buku *Entity Relational Model-Toward a Unified of Data*. ERD merupakan suatu model yang dibuat berdasarkan anggapan bahwa dunia nyata terdiri dari koleksi objek-objek dasar yang dinamakan Entitas yang serta memiliki Relasi antar entitas-entitas tersebut (Fathansyah, 2007).

1. Entitas (*Entity)*

   Entitas adalah individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain.

   Contoh: Sebuah kursi yang kita duduki, seseorang yang menjadi pegawai di sebuah perusahaan dan sebuah mobil yang melintas di depan kita.

2. Atribut (*Attributes* atau *Properties*)

   Atribut mendeskripsikan karateristik atau properti dari suatu entitas. Atribut terdiri dari beberapa tipe atribut, yaitu:

a. Atribut Sederhana (*Simple Attribute*)

   Atribut sederhana adalah atribut atomik yang tidak dapat dipilah lagi.

b. Atribut Komposit (*Composite Attribute*)

   Atribut komposit adalah atribut yang masih dapat diuraikan lagi menjadi sub-sub atribut yang masing-masing memiliki makna.

   Contoh: Nama bisa dipecah menjadi nama depan, nama tengah, dan nama belakang.

c. Atribut Bernilai Tunggal (*Single-Valued Attribute*)

   Atribut bernilai tunggal adalah atribut yang memiliki paling banyak satu nilai untuk setiap baris data.

   Contoh: untuk setiap baris data mahasiswa hanya boleh berisi 1 nilai karena tidak mungkin ada 2 NIM (Nomor Induk Mahasiswa) untuk 1 mahasiswa yang sama.

d. Atribut Bernilai Banyak (*Multivalued Attribute*)

   Atribut bernilai banyak adalah atribut yang dapat diisi lebih dari 1 nilai, tetapi jenisnya sama.

   Contoh: nomor, gelar, hobi, dan lain-lain.

e. Atribut Turunan

   Atribut turunan adalah atribut yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan oleh atribut atau tabel lain yang berhubungan.

   Contoh: atribut umur yang dapat dikalkulasikan dari atribut tanggal hari ini (tanggal sistem) dikurangi oleh atribut tanggal tahir.

3. Relasi (*Relationship*)

   Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

   Contoh: antara dosen wali dan mahasiswa terdapat hubungan berupa bimbingan, antara nasabah dan pinjaman bank terdapat hubungan peminjam. Relasi terdiri dari beberapa jenis, antara lain:

   a. Relasi Satu ke Satu (*One to One*)

      Setiap entitas pada himpunan entitas E1 berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas E2, dan begitu juga sebaliknya

setiap entitas pada himpunan entitas E2 berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas E1.

b. Relasi Satu ke Banyak (*One to Many*)

Setiap entitas pada himpunan entitas E1 berhubungan dengan banyak entitas pada himpunan entitas E2, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas E2 berhubungan dengan paling banyak satu entitas pada himpunan entitas E1.

c. Relasi Banyak ke Satu (*Many to One*)

Setiap entitas pada himpunan entitas E1 berhubungan dengan paling banyak satu entitas pada himpunan entitas E2, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas E2 berhubungan dengan banyak entitas pada himpunan entitas E1.

d. Relasi Banyak ke Banyak (*Many to Many*)

Setiap entitas pada himpunan entitas E1 berhubungan dengan banyak entitas pada himpunan entitas E2, dan begitu juga sebaliknya setiap entitas pada himpunan entitas E2 berhubungan dengan banyak entitas pada himpunan entitas A1.

4. Kardinalitas (*Cardinality*)

Kardinalitas relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas.

5. Kunci (*Key*)

Kunci adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*row*) dalam tabel secara unik. *Key* terdiri dari 4 macam, yaitu:

a. *Super Key*

Super Key adalah satu atau beberapa atribut (kumpulan atribut) yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.

Contoh: pada entitas karyawan maka atribut yang dapat menjadi *super key* adalah (NIK, nama, alamat).

b. *Candidate Key*

Candidate Key adalah satu atribut atau satu set atribut yang bisa dipilih untuk menjadi *primary key*.

Contoh: NIK dan nama pada entitas karyawan.

c. *Primary Key*

Primary Key adalah satu atribut atau satu set atribut yang tidak hanya mengidentifikasi secara unik suatu kejadian spesifik tetapi juga dapat mewakili suatu kejadian dari suatu entitas.

Contoh: NIK pada sebuah entitas karyawan dipilih menjadi *primary key* karena NIK lebih unik daripada karena tidak ada karyawan yang memiliki NIK yang sama.

d. *Foreign Key*

Foreign Key adalah *primary key* yang berasal dari tabel entitas lain atau diri tabel itu sendiri yang menunjuk ke dirinya.

Contoh: entitas karyawan memiliki ID departemen yang berasal entitas departemen.

2.8 XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), *Apache*, *MySQL*, *PHP* dan *Perl*. Program ini tersedia dalam *GNU General Public License* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis.

2.8.1 *Database*

Database adalah kumpulan dari data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di perangkat keras komputer dan dengan perangkat lunak untuk melakukan manipulasi untuk kegunaan tertentu. *Database* diperlukan karena:

1. Salah satu komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.

2. Menentukan kualitas informasi (akurat, tepat pada waktunya dan relevan). Informasi dapat dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.
3. Mengurangi duplikasi data (*data redudancy*).
4. Hubungan data dapat ditingkatkan (*data relatability*).
5. Mengurangi pemborosan tempat simpanan luar. (Irna, 2009)

2.8.2 MySQL

Mysql cepat dan mudah untuk digunakan (*easy-to-use*) sebagai sistem manajemen *database relational* (RDBMS) yang digunakan untuk *database* pada beberapa *website*. Kecepatan adalah fokus utama pada pengembangan awal Mysql. Salah satu keunggulan MySQL adalah lebih mudah dalam instalasi dan pengunaannya dibanding pesaingnya (Janner, 2006).

MySQL merupakan *database* yang paling digemari dikalangan *web programmer*, dengan alasan bahwa program ini merupakan *database* yang sangat kuat dan cukup stabil untuk digunakan sebagai media penyimpanan data. Sebagai sebuah *database server* yang mampu untuk memanajemen *database* dengan baik, MySQL termasuk *database* yang paling digemari dan paling banyak digunakan dibanding *database* lainnya.

Di dalam dunia internet, MySQL dijadikan sebagai sebuah *database* yang paling banyak digunakan selain *database* yang bersifat *shareware* seperti Ms Acces. Penggunaan MySQL ini biasanya dipadukan dengan menggunakan program aplikasi PHP karena kedua program tersebut telah terbukti kehandalannya dalam menangani permintaan data.

MySQL memiliki *query* yang telah distandarkan oleh ANSI/ISO yaitu menggunakan bahasa SQL sebagai bahasa permintaannya. Hal tersebut juga telah dimiliki oleh *database* server seperti Oracle, PostgreSQL, MSQL, SQL *Server* maupun *database* lainnya yang berjalan pada mode grafis (sifatnya visual) seperti Interbase yang diproduksi oleh Borland. Kemampuan lain yang dimiliki MySQL adalah mampu mendukung *Relational Database Manajemen System* (RDBMS) sehingga dengan kemampuan ini MySQL mampu menangani data-data yang berukuran sangat besar hingga berukuran *Gigabyte* (Nugroho, 2008).

2.9 SMS *Gateway*

SMS *Gateway* adalah sebuah perangkat yang memungkinkan untuk mengirim atau menerima pesan dalam format teks maupun biner dari telepon selular. Gateway sendiri merupakan penggabungan dari dua kata yaitu *gate* yang berarti gerbang, dan *way* yang berarti jalan. SMS *Gateway* merupakan perangkat penghubung antara pengirim sms dengan *database*. Perangkat ini terdiri dari satu set PC, *handphone* dan program aplikasi. Program aplikasi ini yang akan meneruskan setiap *request* dari setiap sms yang masuk dengan melakukan *query* ke dalam *database*, kemudian memberi respon dari hasil *query* ini kepada pengirim sms (Zahra, 2010).

2.10 Metode *Testing*

Ada beberapa metode yang dapat digunakan untuk menguji (*testing)* software. Berikut merupakan penjelasan dari metode *testing.*

1. Black *Box Testing*

   Black box testing merupakan suatu metode pengujian *software* yang dilakukan tanpa perlu mengetahui proses yang berlangsung pada *software. Tester* hanya perlu melakukan *input* dan kemudian memeriksa *output* yang ditampilkan oleh *software* yang diuji. Interaksi yang dilakukan hanya sebatas pada *user interface* dari sistem yang diuji sehingga *tester* tidak perlu mengetahui arsitektur maupu *source code* dari sistem yang di uji.

2. *Grey Box Testing*

   Grey box testing merupakan suatu metode pengujian *software* dimana *tester* perlu mengetahui proses yang terjadi di dalam *software. Tester* perlu mengetahui *source code* dari sistem yang akan diuji, tetapi akses terhadap *source code* dibatasi hanya pada bagian dari suatu sistem yang akan diuji.

3. *White Box Testing*

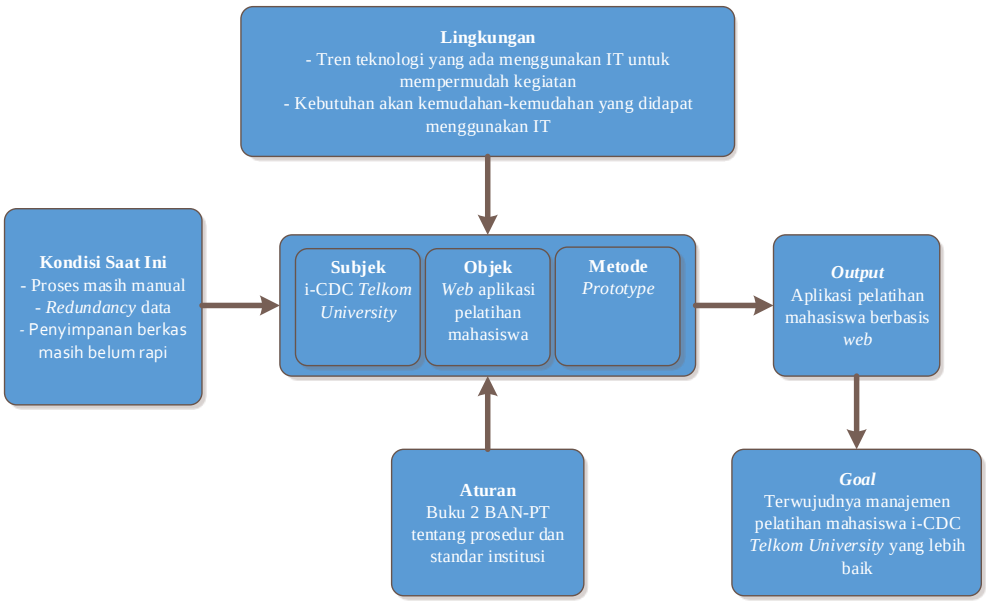
   White box testing merupakan suatu metode pengujian *software* secara menyeluruh. *White box testing* biasa disebut juga sebagai *glass testing* atau *openbox testing. Tester* perlu mengetahui *source code* dari sistem yang akan diuji untuk mengetahui sistem kerja *software.*

BAB 3
METODOLOGI PENELITIAN

Bab ini berisi tentang metode yang digunakan dalam melakukan analisis penelitian terhadap pembangunan *website* pelatihan mahasiswa.

3.1 Model Konseptual

Model konseptual adalah konsep pemikiran yang membantu peneliti untuk merumuskan pemecahan masalah dan membantu dalam merumuskan solusi permasalahan yang ada. Berikut adalah model konseptual penelitian ini.



Gambar 3. 1 Model Konseptual

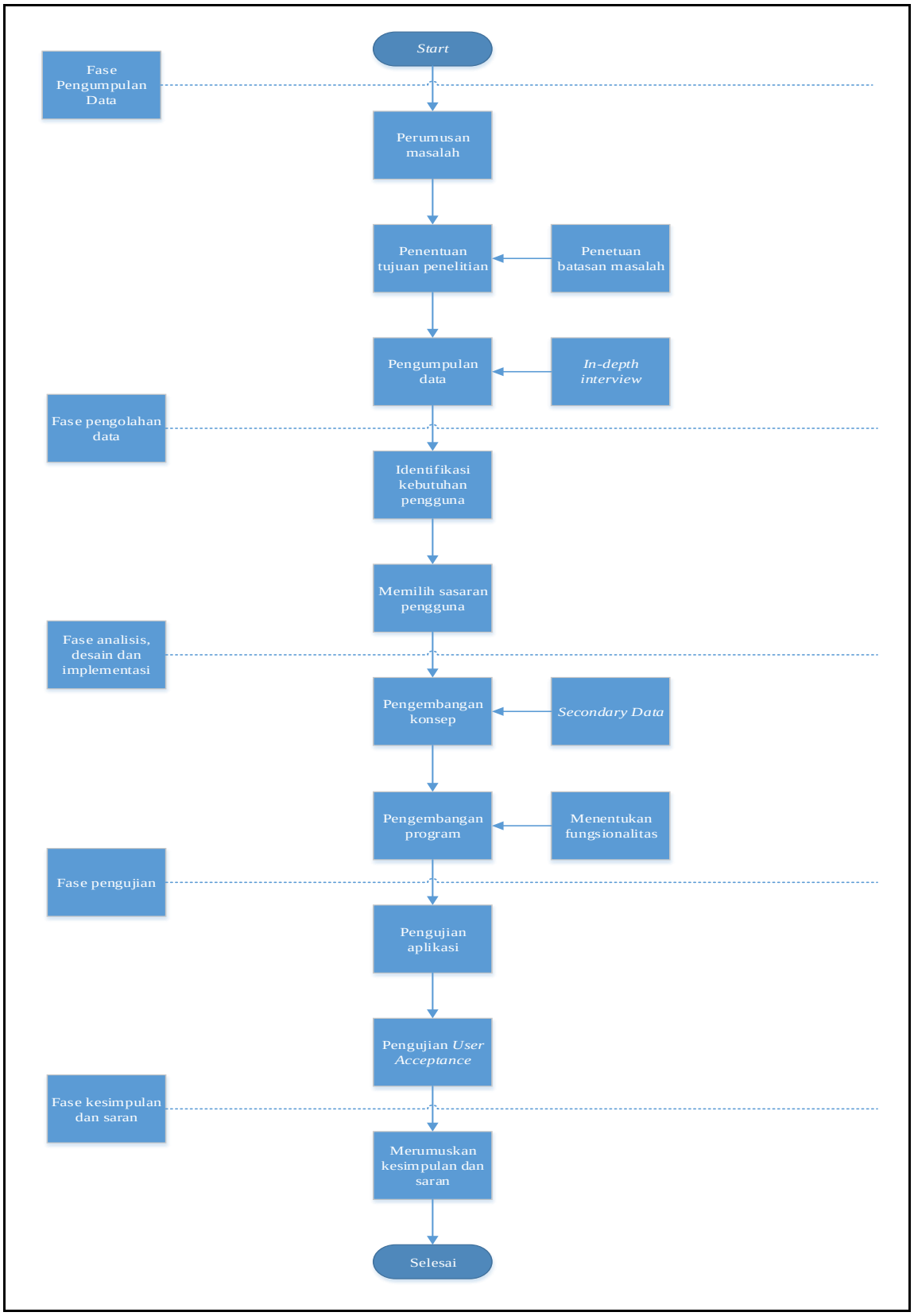
Berdasarkan model konseptual di atas, dapat dilihat bahwa terdapat tiga hal faktor yang mempengaruhi pengembangan sistem, yaitu kondisi saat ini, lingkungan, dan aturan. Pada kondisi saat ini terdapat tiga hal utama yang menjadi bahan pertimbangan dalam membangun sistem. Keadaan pada i-CDC Universitas Telkom saat ini masih menggunakan sistem manual dalam perencanaan pelatihan mahasiswa. Seperti telah dijelaskan pada bab-bab sebelumnya, kondisi ini mengakibatkan terjadinya *redundancy* data dan dokumentasi pelatihan mahasiswa menjadi tidak teratur. Perkembangan teknologi informasi dapat dimanfaatkan untuk

mempermudah kinerja dari i-CDC Universitas Telkom. Sistem pelatihan mahasiswa yang ada akan diperbaiki sesuai dengan tren teknologi yang ada. Tren teknologi yang ada saat ini adalah internet. Dengan adanya internet, sebuah sistem dapat di akses dengan mudah selama jaringan tersedia. Sedangkan untuk aturan-aturan yang digunakan dalam membangun sistem akan mengikuti buku 2 Badan Akreditas Nasional Perguruan Tinggi (BAN-PT) tentang prosedur dan standar institusi.

Setelah mempertimbangkan kondisi saat ini, aturan, dan teknologi yang sedang menjadi tren, maka sistem yang akan dibangun yaitu sistem informasi pelatihan mahasiswa berbasis *web.* Sistem ini akan dibangun dengan menggunakan metode *prototype* untuk mempermudah *developer* mengetahui gambaran sistem secara lebih cepat. *Output* dari penelitian ini adalah sebuah sistem informasi berbasis *web* dengan tujuan untuk mempermudah kinerja i-CDC dalam manajemen pelatihan mahasiswa serta menciptakan sebuah sistem yang lebih baik.

3.2 Sistematika Penelitian

Sistematika penelitian merupakan bagan yang menjelaskan tahapan yang harus dilakukan untuk menyelesaikan penelitian. Berikut adalah sistematika penelitian ini.



Gambar 3. 2 Sistematika Penulisan

Berdasarkan sistematika yang ada, dapat dilihat bahwa pada penelitian ini terdapat lima fase yang akan dilakukan, yaitu fase pengumpulan data; fase pengolahan data; fase analisis, desain dan implementasi; fase pengujian; dan fase kesimpulan dan saran. Penjelasan pada setiap fase penelitian akan dijelaskan sebagai berikut.

3.2.1 Fase Pengumpulan Data

Fase ini dimulai dengan proses studi literatur yang terkait metode pengumpulan data dan metode *prototype* dari sumber-sumber seperti *paper* ataupun buku. Setelah itu akan dilakukan perumusan masalah serta menentukan batasan masalah yang dihadapi agar tercapai tujuan penelitian yang diinginkan. Dengan studi literatur, pertimbangan mengenai batasan-batasan, dan rumusan tujuan penelitian, proses penelitian dapat dilanjutkan ke pengumpulan data terkait dengan hasil penelitian sebelumnya mengenai "Membangun Sistem Informasi Pelatihan Mahasiswa Berbasis *Web* Menggunakan *Framework CodeIgniter* pada *Infocom Career Development Centre* (i-CDC) Universitas Telkom".

Pada proses pengumpulan data akan dilakukan dengan *in-depth interview* dan kuisioner. *In-depth interview* akan dilakukan dengan pihak i-CDC Universitas Telkom yang nantinya akan bertugas sebagai admin sistem yang akan dibangun. Selain admin, *in-depth interview* juga akan dilakukan terhadap *trainer* dan pengawai lainnya yang terkait dengan pelatihan mahasiswa. Data ini nantinya akan digunakan pada fase pengolahan data untuk mengidentifikasikan kebutuhan *user* (admin, *trainer,* dan mahasiswa).

3.2.2 Fase Pengolahan Data

Pada fase ini, data yang telah didapat kemudian diolah dan diidentifikasikan sesuai dengan kebutuhan *user* yang didapat dari *in-depth interview* dan kuisioner. Setelah kebutuhan setiap *user* teridentifikasi, maka hal yang kemudian dilakukan adalah memilih sasaran pengguna. Memilih sasaran pengguna disini maksudnya adalah pengelompokan *user* yang nantinya akan mengakses sistem yang dibangun. Untuk admin sendiri akan diberikan hak penuh dalam melakukan akses ke sistem, sedangkan untuk *trainer* akan diberikan hak untuk mengubah, menambah, maupun menghapus konten yang terkait pelatihan mahasiswa, tetapi akan diberikan batasan

untuk tidak dapat mengubah fungsionalitas dari sistem yang dibangun. Sedangkan untuk mahasiswa sendiri akan dibatasi hanya dapat melakukan setiap proses yang diperlukan dalam pelatihan mahasiswa serta melihat hasil dari pelatihan yang telah dilakukan.

3.2.3 Fase Analisis, Desain, dan Implementasi

Pada fase ini dilakukan analisis terhadap sejumlah data yang telah diolah pada tahap sebelumnya sehingga akan terbentuk sebuah gambaran seperti apa sistem yang diinginkan *user*. Pada pengembangan konsep ini akan dilakukan desain terhadap *interface* dari sistem yang akan dibangun sesuai dengan aturan-aturan perancangan *web*. Konsep dari sistem yang akan dibangun ini adalah *simplicity*, dimana kesederhanaan dari tampilan sistem lebih diutamakan. Hal ini diperlukan untuk mempercepat pengenalan dan pemahaman terhadap *web* yang dibangun untuk menghindari terjadinya kendala baru yang timbul akibat perubahan sistem dari manual ke sistem *online.*

Setelah tahap pengembangan konsep selesai dilakukan, tahap selanjutnya adalah pengembangan program. Pada tahap ini akan didefinisikan setiap fungsionalitas dan hak akses dari setiap *user*. *Coding* dilakukan pada tahap ini untuk mengimplementasikan konsep yang ada ke dalam sebuah sistem informasi berbasis *web*.

3.2.4 Fase Pengujian

Pada fase ini dilakukan pengujian dengan metode *black box testing. Tester* dalam pengujian ini adalah admin. Pengujian dilakukan untuk mengetahui apakah setiap fungsionalitas dari sistem berjalan dengan benar. Jika masih terdapat kesalahan-kesalahan pada sistem, maka akan dilakukan pengkodean (*coding)* ulang. Metode ini dilakukan karena *tester* hanya perlu melakukan pengujian terhadap *user interface* dari sistem dengan melakukan *input* data kemudian memeriksa *output* yang dikeluarkan oleh sistem tanpa perlu mengetahui bagaimana proses yang dilakukan oleh sistem. Untuk itu tidak perlu dilakukan pengujian dengan metoda *white box testing* karena *tester* tidak perlu mengetahui *source code* dari sistem yang dibangun.

Setelah *black box testing* selesai dilakukan dan setiap fungsionalitas berjalan dengan benar, maka pengujian selanjutnya adalah pengujian *user acceptance.* Pengujian ini bertujuan untuk mengetahui apakah setiap *user* bersedia untuk menggunakan sistem yang telah dibangun. *Tester* pada pengujian ini adalah beberapa admin dan beberapa *trainer* serta mahasiswa yang nantinya akan menjadi *user* dari sistem ini.

3.2.5 Fase Kesimpulan dan Saran

Pada fase ini akan diambih kesimpulan terhadap penelitian yang telah dilakukan untuk menjawab rumusan masalah yang ada pada bab 1. Kemudian penulis juga diharapkan memberikan saran berdasarkan penelitian yang telah dilakukan untuk membantu penelitian selanjutnya.

DAFTAR PUSTAKA

Basuki, A.P. 2010. *Membangun Web Berbasis PHP dengan Framework CodeIgniter.* Yogyakarta: Lokomedia.

Bonnie S, M.P. 2008. *Designing Information System.* Jakarta: PT Elex Media Komputindo.

Bonnie S, M.P. 2008. *Designing Information System Concept & Cases with Visio.* Jakarta: PT Elex Media Komputindo.

Fathansyah. 2007. *Basis Data*. Bandung: Informatika.

Hanifah R., Mita. 2012. *Sistem Informasi Manajemen konseling di CDC IT Telkom*. Bandung: Institut Teknologi Telkom.

Inra, S. 2013. Pembangunan Sistem Informasi Perpustakaan dengan Pemanfaatan SMS *Gateway* sebagai Sarana Penunjang Informasi (Studi Kasus SMS BPI 1 Bandung). Bandung: Universitas Kristen Maranatha.

Irna, Y. 2009. *Sistem Manajemen Basis Data*. Bandung: Politeknik Telkom.

Janner, S. 2006. *Menggunakan PHP dan MySql*. Yogyakarta: CV. Andi Offset.

Khosrow-Pour, M. 2005. *Encyclopedia of Information Science and Technology* (5 Volumes). United States: Idea Group Reference.

Nugroho, B. 2008. *Membuat Sistem Informasi Penjualan Berbasis Web dengan PHP dan MySql.* Yogyakarta: Gave Media.

Pitman, D.P. 2005. *UML 2.0 in a Nutshell.* USA: O'Reily Media.

Pratama, A.N. 2010. *CodeIgniter: Cara Mudah Membangun Aplikasi PHP*. Jakarta: Media Kita.

Pressman, R. 2009. *SoftwareEngineering: A Practitioner's Aprroach, 7th International edition.* McGraw-Hill; 7th edition.

Ramadhan, G. 2013. *Aplikasi e-Commerce Berbasis Web dengan Metode Prototyping-Oriented Software di Perusahaan Fahlevi Jaya Abadi*. Bandung.

Raymond, J. 2004. *Sistem Informasi Manajemen.* Jakarta: George Schell.

Sidik, B. 2012. *Framework CodeIgniter.* Bandung: Informatika Bandung.

Simarmata, J. 2010. *Rekayasa Perangkat Lunak.* Yogyakarta: Andi Offset.

Zahra. 2010. *Sistem Pendaftaran Pelatihan di Lembaga Pengembangan Pendidikan (LPP) di Universitas Sebelas Maret Menggunakan SMS Gateway*. Surakarta: Universitas Sebelas Maret.