BAB II
TINJAUAN PUSTAKA

2.1 Studi Literature

Pada penelitian terkait ini, penulis mengkaji penelitian terdahulu mengenai sistem aplikasi yang dilakukan oleh penelitian-penelitian sebelumnya yang dapat menjadi dasar dari penelitian yang dilakukan. Beberapa penelitian mengenai sistem pencarian Aplikasi Monitoring Tugas Akhir di antaranya adalah sebagai berikut:

Ali Ibrahim (2011) penelitian dengan judul "Pengembangan Sistem Informasi Monitoring Tugas Akhir *Berbasis Short Message Service (SMS) Gateway* di Fasilkom Unsri" penelitian ini merupakan suatu aplikasi atau perangkat lunak berupa sistem informasi untuk memonitoring tugas akhir yang meliputi pelaksanaan Praktek Kerja Lapangan seminar proposal, seminar hasil, dan ujian tugas akhir. Sehingga dengan sistem ini mahasiswa, Orang tua mahasiswa dan dosen bisa memonitoring terkait masalah tersebut. Sistem informasi ini diharapkan bisa membantu Jurusan Teknik Elektro untuk merekap data mahasiswa terkait dengan tugas akhir yang dikerjakan. Kemudian harapan untuk mahasiswa dengan sistem ini bisa memonitoring perkembangan tugas akhirnya dan mampu menyelesaikan masa studinya selama 4 tahun.

Putra (2013) penelitian dengan judul "Perancangan Sistem Informasi Monitoring Perkembangan Skripsi Dan *Early Warning* Berbasis *SMS Gateway* Pada Prodi Informatika Fak. Teknik Untan" penelitian ini merupakan suatu suatu aplikasi yang mampu mendeteksi perkembangan penulisan skripsi mahasiswa baik oleh dosen pembimbing, maupun dari pihak yang melakukan evaluasi seperti program studi dan jurusan mengenai perkembangan penulisan tugas akhir serta *Early Warning* yang berbasis *SMS gateway* sebagai sarana pemantauan bagi dosen program studi teknik informatika dan memberikan peringatan kepada mahasiswa bimbingannya beserta Orang tua mahasiswa yang bersangkutan dalam mengetahui perkembangan skripsi mahasiswa yang dibimbingnya.

Anwar (2016) penelitian dengan judul "Rancang bangun sistem informasi tugas akhir Berbasis *SMS Gateway* (Studi kasus di jurusan teknik elektro Universitas bangka belitung)". Dengan dirancangnya sistem informasi tugas akhir berbasis *Short Message Service* (*SMS*) *Gateway* yang terintegrasi dengan sistem

web. sistem dapat mengirimkan pesan ke handphone pengguna secara otomatis tanpa mengirimkannya satu per satu dengan cepat. kemudahan dan kecepatan penyampaian informasi mengenai tugas akhir menjadi hal yang penting, sehingga sistem yang terintegrasi yang dapat mempermudah pengelola jurusan, mahasiswa dan dosen.

Berikut ini adalah penjelasan hal-hal yang membedakan setiap kajian terkait, seperti terlihat pada Tabel 2.1 berikut ini.

Tabel 2. 1 Pembanding Kajian Terkait

No	Penulis	Judul	Keterangan
1.	Ali Ibrahim (2011) Jurusan Sistem Informasi, Fakultas Ilmu Komputer, Universitas Sriwijaya Kampus UNSRI Indralaya	Pengembangan Sistem Informasi Monitoring Tugas Akhir Berbasis *Short Message Service* (*SMS*) *Gateway* di Fasilkom Unsri	1. Aplikasi berbasis *website.* 2. Aplikasi menggunakan teknologi *SMS gateway* 3. mengirimkan SMS balasan atau request status proposal mahasiswa, 4. mengirimkan *SMS* jadwal ujian tugas akhir, mengirimkan *SMS early message* kepada mahasiswa, dosen pembimbing, dosen penguji bahwa ujian akan dimulai satu jam lagi, mengirimkan SMS nilai ujian akhir mahasiswa
2.	Randhie Akbar Mula Putra (2013) Program Studi Teknik Informatika Fakultas Teknik Universitas	Perancangan Sistem Informasi Monitoring Perkembangan Skripsi Dan *Early Warning* Berbasis *SMS Gateway* Pada	1. Aplikasi berbasis *website.* 2. Aplikasi menggunakan teknologi *SMS gateway* 3. Aplikasi dapat melakukan mengirimkan *SMS* notifikasi dan *SMS warning* dengan baik kepada Dosen,

No	Penulis	Judul	Keterangan
		Prodi Informatika Fak. Teknik Untan	Mahasiswa, dan Orang tua Mahasiswa.
3.	Khusni Latiful Anwar (2016) Jurusan Teknik Elektro Fakultas Teknik Bangka Belitung	Rancang bangun sistem informasi tugas akhir Berbasis *SMS Gateway* (Studi kasus di jurusan teknik elektro Universitas bangka Belitung)	1. Aplikasi berbasis *website.* 2. Aplikasi menggunakan teknologi *SMS Gateway* 3. Menggunakan *Framework Codeigniter.* 4. Pengelola jurusan menulis pengumuman dengan mengirimkan *SMS* ke mahasiswa-mahasiswa dan dosen tertentu.

Berikut adalah penelitian yang dilakukan oleh penulis, seperti terlihat pada Tabel 2.2 berikut ini.

Tabel 2. 2 Penelitian Yang Dilakukan Penulis

No	Penulis	Judul	Keterangan
1.	Fadil Setiawan (2022), Universitas Tanjungpura Pontianak	Aplikasi Monitoring Tugas Akhir Mahasiswa Informatika UNTAN Dengan *Early Warning* Kepada Orang tua Berbasis *Website* dan *SMS Gateway*	1. Aplikasi Berbasis *website.* 2. Menggunakan *API SPOTA*. 3. Menggunakan teknologi *SMS gateway* berupa *SMS API* sebagai akses mengirim pesan 4. Menggunakan *Framework Codeigniter* 5. Mengirimkan *notifikasi SMS* dengan kepada Dosen dan Mahasiswa yang telah input progres. 6. Mengirimkan *Early Warning* kepada Dosen Pembimbing,

No	Penulis	Judul	Keterangan
			Mahasiswa, dan Orang tua untuk mahasiswa yang lewat 7 hari 7. Mengirimkan *Early Warning* kepada Dosen Pembimbing, Mahasiswa, dan Orang tua untuk mahasiswa yang Tugas Akhir lewat 1 tahun dan 2 tahun. 8. Mengirimkan *Early Warning* Dosen Pembimbing, Mahasiswa, dan Orang tua untuk mahasiswa yang masuk Evaluasi 9. Mengirimkan *Early Warning* Mahasiswa, dan Orang tua untuk masa studi mahasiswa.

2.2 Perangkat Lunak (*Software*)

Perangkat lunak merupakan sebuah perangkat yang tidak berbentuk secara fisik, namun dapat dioperasikan oleh pengguna. Perangkat lunak (*software*) adalah program komputer yang terasosiasi dengna dokumen perangkat lunak seperti dokumentasi kebutuhan, model desain dan cara penggunaan (*user manual*) (Sukamto & Shalahuddin, 2016).

Perangkat lunak sebagai bagian sistem komputer yang sifatnya non riil, merupakan sebuah program sebagai sederetan instruksi yang segaja dibuat atau dibangun untuk mengendalikan komputer sehingga komputer dapat melakukan tindakan sesuai yang dikehendaki pembuatnya. Komputer hanyalah sekedar mesin yang tidak akan dapat melakukan tugas yang dikehendaki pemakai sekiranya tidak didukung oleh perangkat lunak.

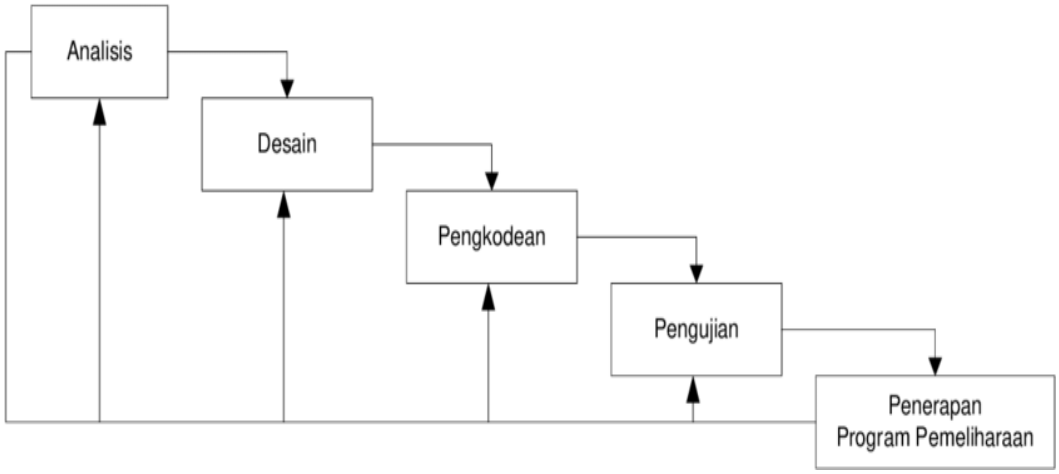
Menurut Kadir (2014), yang menyatakan bahwa perangkat lunak adalah instruksi-instruksi yang ditunjukkan kepada komputer agar dapat melaksanakan tugas sesuai kehendak pengguna. Senada dengan itu, Yurindra (2017)

mengemukakan perangkat lunak merupakan transformasi informasi yang memproduksi, mengatur, memperoleh, memodifikasi, menampilkan atau memancarkan informasi sehingga pekerjaan dapat menjadi lebih sederhana.

Dari beberapa pendapat yang sudah dijelaskan, maka dapat disimpulkan bahwa perangkat lunak adalah instruksi-instruksi atau data yang diformat secara digital yang bisa dibaca dan ditulis oleh komputer sesuai kehendak pengguna.

2.3 Model *Waterfall*

Terdapat beberapa metodologi *Systems Development Life Cycle* (*SDLC*) yang biasa digunakan dalam membangun sebuah sistem, salah satunya adalah model *waterfall*. *Waterfall* merupakan model yang bersifat sistematis dan termasuk dalam model klasik, nama lainnya adalah *Linear Sequential Model* (Pressman, 2001). Tahapan-tahapan model *waterfall* dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Model *Waterfall* (Pressman, 2001)

Penjelasan tahapan-tahapan *waterfall* tersebut yaitu:

1. Analisis

   Fase ini merupakan proses analisis terhadap sistem yang sedang berjalan dengan tujuan untuk mendapatkan jawaban mengenai pengguna sistem, cara kerja sistem dan waktu penggunaan sistem, sehingga kebutuhan yang diperlukan untuk sistem baru akan didapatkan.

2. Desain

   Desain merupakan proses penentuan cara kerja sistem dalam hal perancangan antarmuka, database, dan perancangan alur program. Perancangan diperlukan untuk menggambarkan sistem baru untuk memenuhi kebutuhan *user*.

3. Pengkodean

   Tahapan pengkodean yaitu tahap rancangan sistem yang dibentuk menjadi suatu kode program untuk pembuatan sistem.

4. Pengujian

   Pengujian dilakukan untuk mengetahui kesesuaian sistem berjalan sesuai prosedur atau tidak dan memastikan sistem terhindar dari *error* yang terjadi. Pengujian juga dilakukan untuk memastikan kevalidan dalam proses input sehingga dapat menghasilkan output yang sesuai.

5. Penerapan Program Pemeliharaan

   Fase ini yaitu penerapan program dan pemeliharaan sistem yang berguna untuk melihat kemampuannya, mengecek jika masih ada ditemukan *error* atau ada penambahan fitur-fitur yang belum ada pada sistem tersebut. Pemeliharaan diperlukan ketika adanya perubahan dari pengguna seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.4 *SMS Gateway*

SMS Gateway adalah komunikasi menggunakan *SMS* yang mengandung informasi berupa nomor telepon seluler pengirim, penerima, waktu dan pesan. Informasi tersebut dapat diolah dan bisa melakukan aktivasi transaksi tergantung kode-kode yang sudah disepakati. Aplikasi *SMS Gateway* adalah sebuah perangkat lunak yang menggunakan bantuan komputer dan memanfaatkan teknologi seluler yang diintegrasikan guna mendistribusikan pesan-pesan yang dipadukan lewat sistem informasi melalui media *SMS* yang ditangani oleh jaringan seluler.

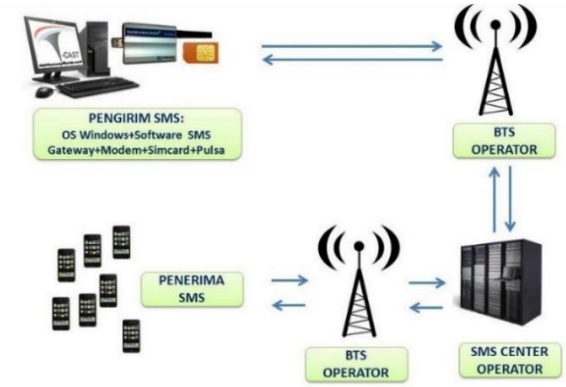
SMS gateway merupakan sistem aplikasi untuk mengirim dan atau menerima *SMS*, terutama digunakan dalam aplikasi bisnis, baik untuk kepentingan promosi, service kepada customer, pengadaan content produk atau jasa, dan seterusnya. Karena merupakan sebuah aplikasi, maka fitur-fitur yang terdapat didalam *SMS gateway* dapat dimodifikasi sesuai dengan kebutuhan, beberapa fitur yang umum dikembangkan dalam aplikasi *SMS gateway* menurut (Mulyani et al., 2012).

Menurut Ali Ibrahim (2011), *SMS Gateway* adalah sebuah perangkat lunak yang menggunakan bantuan komputer dan memanfaatkan teknologi seluler yang diintegrasikan untuk mendistribusikan pesan-pesan yang di generate lewat sistem

informasi melalui media *SMS* yang ditangani oleh jaringan seluler. maka dapat disimpulkan bahwa *SMS gateway* adalah sebuah sistem aplikasi untuk mengirim atau menerima *SMS* dengan menggunakan bantuan komputer untuk mendistribusikan pesan-pesan yang di generate lewat sistem informasi melalui media *SMS* yang ditangani oleh jaringan seluler.

Menurut Ibrahim (2011:86), Mekanisme kerja pengiriman SMS dibagi menjadi 3 bagian yaitu:

1) *Intra-operator SMS*: pengiriman *SMS* dalam satu operator. SMS yang dikirimkan oleh pengirim akan terlebih dahulu masuk ke *SMSC* operator nomor pengirim, kemudian *SMSC* akan mengirimkan ke nomor yang dituju secara langsung. Penerima kemudian akan mengirimkan delivery report yang menyatakan bahwa *SMS* telah diterima ke *SMSC*. *SMSC* kemudian meneruskan report tersebut ke nomor pengirim *SMS*, disertai status proses pengiriman SMS tersebut
2) *Inter-operator SMS*: pengiriman *SMS* antar operator yang berbeda. Yang membedakan adalah mekanisme ini terdapat dua *SMSC* yaitu *SMSC* pengirim dan *SMSC* penerima. *SMS* yang dikirim akan masuk ke *SMSC* pengirim dan diteruskan ke *SMSC* penerima, setelah itu *SMS* dikirimkan ke telepon seluler tujuan. Demikian juga dengan delivery report akan diterima terlebih dahulu oleh *SMSC* penerima, kemudian diteruskan ke *SMSC* pengirim *SMS*. Komunikasi antar *SMSC* dapat berjalan jika telah terdapat kesepakatan kerja sama antar operator tersebut, jika tidak terdapat kesepakatan akan menyebabkan *SMS* yang dikirim dengan nomor tujuan dengan operator berbeda tidak akan sampai pada nomor tujuan yang dituju
3) *SMS Internasional*: pengirim *SMS* dari operator suatu negara ke Negara lain. *SMS* internasional pada hakekatnya sama dengan mekanisme inter- operator, yang membedakan hanya pada *SMSC* nomor penerima adalah *SMSC* operator



Gambar 2. 2 Cara Kerja *SMS Gateway*

Dengan kecepatan teknologi *SMS Gateway* dalam memberikan informasi kepada user, maka dengan adanya usul penelitian dan akan menghasilkan produk perangkat lunak yang dapat membantu meningkatkan komunikasi di fakultas ilmu komputer. Dengan aplikasi yang akan dihasilkan maka proses komunikasi baik dan sangat berguna untuk Orang tua untuk Monitoring kuliah anak dari luar kota.

2.5 MetaKreasi

MetaKreasi merupakan perusahaan yang bergerak di bidang teknologi informasi sebagai penyedia layanan atau jasa *SMS* Broadcast/Massal & WhatsApp Blast dengan cepat dan terpercaya. MetaKreasi memberikan solusi terbaik untuk mempromosikan produk dan jasa perusahaan Anda melalui layanan *SMS Broadcast/Blast* dengan beberapa layanan *SMS* Marketing, *SMS Gateway, SMS Center*, *SMS Masking*, *SMS Masking* berdasarkan lokasi & *SMS Masking* dengan target spesifik. Pengiriman dibagi 5 macam layanan *SMS* yang bisa di gunakan yaitu;

1. *SMS Reguler* adalah Layanan *SMS Reguler* sistem *SMS Gateway Online* dengan menggunakan nomor *GSM* dari *server* Metakreasi untuk kebutuhan kirim SMS Massal berupa pengumuman/informasi. Layanan *SMS Gateway* ini menjadi solusi untuk meningkatkan bisnis Anda dalam melayani pelanggan melalui *SMS*, seperti *SMS* promosi atau notifikasi pembayaran.
2. *SMS Center* adalah Layanan *SMS Gateway* dengan menggunakan nomor *SMS Center* khusus yang bersifat 2 Arah untuk kebutuhan Kirim dan Terima *SMS* pada Perusahaan Anda.

3. *SMS Blast Masking* adalah Layanan *SMS Blast Masking* dengan menampilkan identitas (Sender ID) berupa AlphaNumeric yang dapat di customize menjadi nama Perusahaan Anda.
4. *SMS LBA* (*Location Based Advertising*) adalah layanan *SMS blast* yang menargetkan orang-orang yang berada di lokasi tertentu untuk dikirimkan konten promosi atau penawaran dengan lokasi spesifik yang dikehendaki (bisa menggunakan Titik Koordinat).
5. *SMS Masking Targeted* adalah layanan yang memungkinkan Anda untuk mengirim SMS ke semua orang yang sesuai dengan Target lokasi, Jenis Kelamin, Umur, Pengguna Merk/OS/Type Ponsel, Jenis Paket Data, History Browsing dan lain-lain.

2.6 *Early Warning System*

Early Warning System merupakan Sistem yang digunakan sebisa mungkin untuk mencegah suatu hal buruk yang akan terjadi dengan memberikan peringatan sedini mungkin kepada yang bersangkutan agar bisa menghindari atau meminimalkan akibat yang ditimbulkan (Bouma dan Kaay H.J, 1996).

Sistem yang mudah digunakan dan dirancang untuk membantu mengidentifikasi atau mengenali perkembangan community yang tidak menguntungkan dan mengelola risiko organisasi (Shigang dan Sanjay, 2004).

Sedangkan definisi khusus dari sistem peringatan dini adalah sebagai berikut:

1. Menurut pakar di bidang kesehatan sistem peringatan dini adalah sistem (rangkaian proses) pengumpulan dan analisis data serta desiminasi informasi tentang keadaan darurat yang merupakan fenomena keberadaan bahaya dan mengganggu atau mengancam terhadap manusia (Yurindra, 2017).
2. Menurut pakar di bidang militer sistem peringatan dini adalah suatu jaringan sistem yang memiliki kepekaan lebih, seperti satelit dan radar yang digunakan untuk mendeteksi setiap waktu serangan musuh agar dapat bertahan atau melakukan suatu tindakan pertahanan (Dipenda, 2001).

2.6.1 Komponen *Early Warning System*

Komponen *Early Warning System* berdasarkan tipe informasi adalah sebagai berikut:

1. Petunjuk-petunjuk fungsional (functional indicators)
2. Petunjuk-petunjuk organisasi pada level departemen (organizational indicators)
3. Petunjuk kemajuan dan data promosi (appointment and promotion data)
4. Data pembantu (resource data)
5. Pemeriksaan (audits/review/thematic findings)

2.7 *Application Programming Interface* (*API*)

Application Programming Interface (API) dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Secara umum *API* merupakan dokumentasi dari fungsi, struktur, kelas, dan lain sebagainya yang dapat diakses oleh yang membutuhkan dengan cara tertentu sesuai yang ditentukan layanan. *API* memudahkan pengembang lain mengambil data untuk kemudian diintegrasikan dengan perangkat lunak lain (Santoso & Rais, 2016).

API (Application Programming Interface) adalah sebuah teknologi untuk memfasilitasi pertukaran informasi atau data antara dua atau lebih aplikasi perangkat lunak. *Application Programming Interface* yang berarti Antarmuka Pemrograman Aplikasi adalah sekumpulan perintah, fungsi, dan protocol yang dapat digunakan oleh programmer disaat membangun software buat sistem operasi tertentu saja. Sebuah *API* mendefinisikan bagaimana cara programmer memanfaatkan suatu fitur tertentu dari sebuah komputer. *API* tersedia untuk *sistem windowing*, sistem file, sistem basis data dan sistem jaringan.

API (Application Programming Interface) menyediakan fungsi dan juga perintah dengan bahasa yang lebih terstruktur dan lebih mudah bagi dipahami bagi Programmer, hal ini penting dalam aspek editing dan pengembangan, sehingga Programmer mampu mengembangkan sistem tersebut dengan mudah. *API* juga bisa dipakai pada Sistem Operasi mana saja asalkan telah ada paket-paket *API* nya.

Cara menggunakan *API* karena sebagian besar *API* membutuhkan kunci *API* yakni:

a) Mendapatkan *API Key*

Setelah Anda menemukan *API* yang ingin Anda mainkan, lihat di dokumentasi

untuk persyaratan akses *API* tersebut. Sebagian besar *API* akan meminta Anda untuk menyelesaikan verifikasi identitas, seperti masuk dengan menggunakan akun Google Anda. Anda akan mendapatkan serangkaian huruf dan angka unik untuk digunakan saat mengakses *API*, alih-alih hanya menambahkan email dan kata sandi setiap kali (yang tidak terlalu aman).

b) Menggunakan *HTTP Klien Online*

Cara termudah untuk mulai menggunakan *API* adalah dengan menemukan klien *HTTP online*, seperti *REST-Client*, Postman, atau Paw. Alat yang sudah jadi ini (dan sering gratis) membantu Anda menyusun permintaan Anda untuk mengakses *API* yang ada dengan kunci *API* yang Anda terima. Anda masih perlu mengetahui beberapa sintaks dari dokumentasi, teapi hanya perlu sedikit pengetahuan pengkodean yang diperlukan.

c) Mengambil Data dari *API*

Cara berikutnya yaitu untuk mengambil data dari *API* adalah dengan membangun URL dari dokumentasi *API* yang ada. Contohnya seperti cara menarik data lokasi dari Google Maps melalui *API*, dan kemudian menggunakan koordinat tersebut untuk menemukan foto terdekat di Instagram

2.7.1 Kelebihan dan Kekurangan *API*

Kemunculan teknologi *API* memungkinkan developers untuk berkoordinasi dalam memperbarui sistem dan oleh karena itulah *API* menghasilkan lebih banyak potensi. Terlepas daripada hal tersebut, tentu dalam penggunaannya *API* juga memiliki keuntungan dan kerugian.

1. Kelebihan *API*
   a. Otomasi; Dengan *API*, komputer (bukan manusia) dapat mengelola pekerjaan, melalui *API*, sebuah agensi dapat memperbarui alur kerja mereka agar lebih cepat dan lebih produktif.
   b. Aplikasi; *API* dapat mengakses komponen aplikasi, sehingga pengiriman layanan dan informasi pun menjadi lebih fleksibel.
   c. Lebih banyak cakupan; Dengan *API*, application layer (l*API*san aplikasi) dapat dibuat dan dapat digunakan untuk mendistribusikan informasi dan layanan kepada audiens baru yang dapat dipersonalisasi untuk menciptakan pengalaman pengguna khusus.

d. Tersedianya data baru; *API* memungkinkan semua informasi yang dihasilkan di tingkat (seperti pemerintahan) menjadi tersedia untuk setiap warga negara, bukan hanya segelintir orang.
e. Efisiensi; Ketika akses disediakan ke *API*, konten yang dihasilkan dapat dipublikasikan secara otomatis dan tersedia untuk setiap saluran. Hal ini memungkinkan untuk dibagikan dan didistribusikan dengan lebih mudah.
f. Integrasi; *API* memungkinkan konten tertanam dari situs atau aplikasi apa pun dengan lebih mudah. Hal ini menjamin pengiriman informasi yang lebih lancar dan pengalaman pengguna yang terintegrasi.
g. Personalisasi; Melalui *API*, pengguna atau perusahaan mana pun dapat menyesuaikan konten dan layanan yang paling sering mereka gunakan.
h. Adaptasi; Kebutuhan pengguna sering berubah seiring berjalannya waktu, *API* ada untuk membantu mengantisipasi perubahan. Ketika bekerja dengan teknologi ini, migrasi data didukung lebih baik, dan informasi pun ditinjau lebih cermat. Singkatnya, *API* membuat penyediaan layanan menjadi lebih fleksibel.

2. Kekurangan *API*
   a. Biaya dan waktu yaitu kerugian *API* yang pertama adalah terkait biaya dan waktu. Menyediakan *API* itu membutuhkan biaya yang dapat terbilang mahal dalam hal waktu pengembangan, pemeliharaan berkelanjutan, menyediakan dokumentasi *API* di website (baca pengertian website disini) Anda dan memberikan dukungan kepada pengguna *API* Anda.
   b. Paparan Keamanan adalah paparan keamanan. Menambahkan *API* akan menambahkan permukaan dan potensi serangan lain ke situs web Anda.
   C. Hasilnya Belum Tentu Sesuai adalah terjadinya hasil yang belum tentu sesuai dengan ekspektasi. Anda mungkin tidak menyukai hasilnya atau Anda dapat mengubah arah bisnis Anda. Tidak selalu mudah untuk memprediksi dengan tepat bagaimana *API* Anda akan digunakan. Jika integrasi Anda dengan sistem pesan (misalnya) berarti bahwa pengguna lebih jarang masuk ke layanan Anda, apakah akan ada dampak negatif secara keseluruhan untuk Anda? Ini memang tidak biasa, tetapi hal ini bisa terjadi.

2.8 *Unified Modelling Language* (*UML*)

Unified Modelling Language (*UML*) adalah bahasa pemodelan untuk sistem atau perangkat lunak. Pemodelan (*modeling)* sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami (Nugroho, 2010).

Sedangkan menurut (Sukamto & Shalahuddin, 2016) *UML* (*Unified Modeling Language*) adalah salah satu standar bahasa visual yang banyak digunakan di dunia industri untuk mengidentifikasi *requirement,* membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. *UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. 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

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

Di dalam *UML* ada 13 buah diagram yang dikelompokkan ke dalam tiga kategori yaitu:

1. *Structure Diagrams,* yaitu kumpulan diagram-diagram yang menggambarkan struktur statis dari sistem yang dimodelkan
2. *Behavior Diagrams,* yaitu kumpulan diagram-diagram yang menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi di dalam sistem.
3. *Interaction Diagram,* yaitu kumpulan diagram-diagram yang menggambarkan interaksi sistem dengan sistem lain ataupun interaksi antar subsistem dalam sebuah sistem

Secara garis besar, beberapa diagram utama sudah dapat menggambarkan keseluruhan sistem. Diagram tersebut antara lain *use case diagram, class diagram,* dan *activity diagram.*

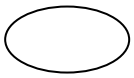
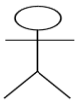
2.8.1 *Use Case Diagram*

Use Case Diagram menurut (Widodo, 2011) Diagram *use case* bersifat

statis, yang memperlihatkan himpunan *Use Case* dan aktor-aktor (suatu jenis khusus dari kelas) dan menggambarkan apa saja aktivitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna yang menjadi persoalan itu apa yang dilakukan bukan bagaimana melakukannya. *Use Case Diagram* menggambarkan fungsionalitas yang diharapkan dari sistem.

Deskripsi simbol-simbol yang digunakan pada *Use Case Diagram* dapat dilihat pada Tabel 2.3 (Nugroho, 2010).

Tabel 2. 3 Deskripsi Notasi pada *Use Case Diagram*

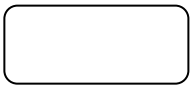
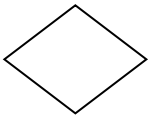
No	Notasi	Nama	Deskripsi
1		*Use Case*	Menggambarkan fungsionalitas yang dimiliki sistem.
2		*Actor*	Menggambarkan semua objek di luar sistem (bukan hanya pengguna sistem/perangkat lunak) yang berinteraksi dengan sistem yang dikembangkan.
3		*Association*	Lintasan komunikasi antara *actor* dengan *use case.*
4	<<extend>> - - - - - - - ->	*Extended* (Ekstensi)	Penambahan perilaku ke suatu *use case* dasar.
5	<<include>> < - - - - - - -	*Include* (Menggunakan)	Penambahan perilaku ke suatu *use case* dasar yang secara *explicit* mendeskripsikan penambahan tersebut.
6		*Generalization* (Generalisasi)	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu merupakan fungsi yang lebih umum dari lainnya.

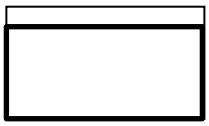
2.8.2 *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) proses bisnis dan urutan aktivitas dalam sebuah proses. *Activity diagram* sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Membuat *activity diagram* pada awal pemodelan proses cukup menguntungkan untuk membantu memahami keseluruhan proses. *Activity diagram* juga bermanfaat untuk menggambarkan *parallel behavior* atau menggambarkan interaksi antara beberapa *use case*.

Berikut merupakan simbol notasi *Activity Diagram* pada Tabel 2.4 (Sukamto & Shalahuddin, 2016).

Tabel 2. 4 Deskripsi Notasi pada *Activity Diagram*

No	Notasi	Nama	Deskripsi
1		Status Awal (*initial node)*	Status awal aktivitas, sebuah diagram aktivitas memiliki status awal.
2		Status Akhir (*final node)*	Status akhir yang dilakukan sistem.
3		Aktifitas (*activity)*	Aktivitas yang dilakukan oleh sistem, biasanya diawali oleh kata kerja.
4		Percabangan (*decision)*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
5		Penggabungan (*join)*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.

No	Notasi	Nama	Deskripsi
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.8.3 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas – kelas yang akan dibuat untuk membangun sistem (Sukamto & Shalahuddin, 2013). *Class diagram* mendeskripsikan jenis – jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka (Fowler, 2005:141). *Class diagram* digunakan untuk menggambarkan hubungan kelas – kelas antara satu dengan yang lain seta memiliki atribut dan operasi yang terdapat dalam sistem yang akan dibuat. Atribut merupakan variabel – variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi – fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private* (-), hanya dapat digunakan oleh *class* yang memilikinya
2. *Public* (+), dapat digunakan oleh *class* lain.
3. *Protected* (#), hanya dapat dipanggil oleh *class* yang bersangkutan jumlah suatu anak yang mewarisinya.

Nilai kardinalitas atau *multiplicity* sebuah *class* menunjukkan jumlah suatu objek yang dapat berhubungan dengan objek lain. Berikut nilai kardinalitas atau *multiplicity* pada Tabel 2.5 dan notasi *class diagram* pada Tabel 2.6 (Sukamto & Shalahuddin, 2013).

Tabel 2. 5 Jenis-jenis *Multiplicity*

No	Indikator	Keterangan
1	0 .. 1	Nol atau satu
2	1	Hanya satu
3	0 .. *	Nol atau lebih
4	1 .. *	Satu atau lebih

Tabel 2. 6 Deskripsi Notasi pada *Class Diagram*

No	Notasi	Nama	Deskripsi
1	**Kelas** +atribut +metode()	*Class*	Menggambarkan konsep dasar pemodelan sistem.
2	————	Asosiasi (*Association*)	Sebuah garis solid antara dua *class*, ditarik dari *class* sumber ke *class* target lebih spesifik, digunakan dalam struktur pewarisan.
3	- - - - - - - - ->	Ketergantungan (*Dependency)*	Relasi antara dua elemen jika perubahan definisi sebuah elemen (*supplier* atau sumber) dapat menyebabkan perubahan pada elemen lainnya (*Client* atau target).

2.9 *Flowchart*

Menurut (Wandah Wibawanto, 2017) *Flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (intruksi) dengan proses lainnya dalam suatu program. Diagram alur dapat menunjukan secara jelas, arus pengendalian suatu algoritma yakni bagaimana melaksanakan suatu rangkaian kegiatan secara logis dan sistematis.

2.9.1 Bentuk – Bentuk *Flowchart*

Menurut (Sulindawati & Muhammad Fathoni, 2010) Sulindawati dan Fathoni (2010:8) flowchart terbagi atas lima jenis, yaitu:

a. Sistem *Flowchart*

   Sistem *Flowchart* merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem.

b. *Flowchart* Dokumen

Flowchart Paperwork menelusuri alur dari data yang ditulis melalui sistem.

c. *Flowchart* Skematik

Flowchart skematik mirip dengan *flowchart* sistem yang menggambarkan suatu sistem atau prosedur. *Flowchart* skematik ini bukan hanya menggunakan simbol-simbol *flowchart* standar, tetapi juga menggunakan gambar-gambar komputer, peripheral, form-form atau peralatan lain yang digunakan dalam sistem.

d. *Flowchart* Program

Flowchart Program dihasilkan dari *flowchart* sistem. *Flowchart* program merupakan keterangan yang lebih rinci tentang bagaimana setiap langkah program atau prosedur sesungguhnya dilaksanakan.

e. *Flowchart* Proses

Flowchart proses merupakan teknik penggambaran rekayasa industrial yang memecah dan menganalisa langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

2.9.2 Teknik Pembuatan

1. *General Way Teknik*

   pembuatan *flowchart* dengan cara ini lazim digunakan dalam menyusun logika suatu program yang menggunakan proses pengulangan secara tidak langsung (Non Direct Loop).

2. *Interaction Way Teknik*

   pembuatan *flowchart* dengan cara ini biasanya dipakai untuk logika program yang cepat serta bentuk permasalahan yang kompleks

2.10 Teknologi Pendukung

2.10.1 *Web*

Menurut Kadir (2014), *World Wide Web (*WWW*)* atau biasa disebut dengan web merupakan salah satu sumber daya Internet yang berkembang pesat. Pertama kali aplikasi web dibangun hanya dengan menggunakan bahasa yang disebut *HTML* (*HyperText Markup Language*) dan protokol yang digunakan dinamakan HTTP (*HyperText Transfer Protocol*). Pada perkembangan berikutnya,

sejumlah skrip dan objek dikembangkan untuk memperluas kemampuan *HTML* yang sekarang ini terdapat banyak skrip seperti: *PHP* dan ASP, sedangkan contoh yang berupa objek antara lain adalah applet *(java*) (Kadir, 2014). Jadi aplikasi web atau aplikasi berbasis web (*Web-based application*) adalah aplikasi untuk menyampaikan informasi kepada pengguna yang menggunakan layanan Internet berbasis web.

Dalam aplikasi tersebut, terjadi pertukaran antara *klien* (komputer yang meminta informasi) dengan *server* (komputer yang memasok atau menanggapi informasi). Web memberikan informasi secara *online* melalui internet langsung. *Klien* melakukan permintaan informasi dengan menggunakan *browser* (contoh *browser: Internet Explorer, Opera, Mozilla*, dan sebagainya). *Server* menerima informasi dan melayani permintaan dari *client*. Hal ini biasa disebut dengan web *server* (contoh *web server: Apache, IIS, Xitami,* dan sebagainya). Setelah itu, web *server* akan berkomunikasi dengan *middleware* (contoh *middleware: ASP, JSP, PHP*, dan sebagainya) untuk bisa berhubungan dengan basis data atau *database* (contoh *database: access, oracle, sql*, dan sebagainya). Setelah berinteraksi dengan *database, server* yang telah mendapatkan informasi akan memberikan tanggapan terhadap *klien* yang meminta informasi tadi.

2.10.2 *HyperText Markup Language* (*HTML*)

Menurut Solihin (2016), *HTML* merupakan singkatan dari *Hyper Text Markup Language*. *HTML* dikembangkan pertama kali oleh Tim Berners-Lee bersamaan dengan protokol HTTP (*Hypertext Transfer Protocol*) pada tahun 1989. Tujuan utama pengembangan *HTML* adalah untuk menghubungkan suatu halaman web dengan halaman web lainnya. Tentunya pada awal pengembangannya.

Menurut Kuswayatno (2006), *HTML* merupakan halaman yang berada pada suatu situs internet atau web. *HTML* merupakan metode yang menautkan (*link*) satu dokumen ke dokumen lain melalui teks. Menurut Deris Setiawan, *HTML* merupakan framework internet, hampir semua situs web yang ada menggunakan *HTML* untuk menampilkan teks, grafik, suara, dan animasinya.

HTML adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas

format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format ASCII normal sehingga menjadi halaman web dengan perintah-perintah *HTML*.

Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (*Standard Generalized Markup Language*), *HTML* adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. *HTML* saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C). *HTML* dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

2.10.3 *Hypertext Preprocessing* (*PHP*)

Menurut Saputra (2011) yang mengemukakan bahwa *PHP* atau yang memiliki kepanjangan *PHP Hypertext Preprocessor* merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu *website* dinamis. *PHP* menyatu dengan kode *HTML*, maksudnya adalah beda kondisi. *HTML* digunakan sebagai pembangun atau pondasi dari kerangka *layout web*, sedangkan *PHP* difungsikan sebagai prosesnya sehingga dengan adanya *PHP* tersebut, *web* akan sangat mudah di-*maintenance*. *PHP* berjalan pada sisi *server* sehingga *PHP* disebut juga sebagai bahasa *Server Side Scripting*. Artinya bahwa dalam setiap/untuk menjalankan *PHP*, wajib adanya *web server*.

PHP ini bersifat *open source* sehingga dapat dipakai secara cuma-cuma dan mampu lintas *platform*, yaitu dapat berjalan pada sistem operasi *Windows* maupun *Linux*. *PHP* juga dibangun sebagai modul pada *web server apache* dan sebagai *binary* yang dapat berjalan sebagai CGI.

2.10.4 *JavaScript*

Javascript menurut Sunyoto (2007) adalah bahasa *scripting* yang populer di internet dan dapat bekerja di sebagian besar *browser* populer seperti *Internet Explorer* (IE), *Mozilla Firefox*, *Netscape* dan *Opera*. Kode *Javascript* dapat disisipkan dalam halaman web menggunakan tag *SCRIPT*.

Menurut Utomo (2007), *JavaScript* merupakan bahasa scripting yang pertama kali dikembangkan oleh Netscape pada tahun 1955. Penulisan *JavaScript* berada di dalam dokumen *HTML* dan pemanggilan program tersebut tergantung pada *browser* (navigator) yang digunakan dalam memanggil halaman yang terdapat pada script tersebut. *JavaScript* juga tidak memerlukan kompilator atau penerjemah khusus untuk menjalankannya.

Selanjutnya menurut Bride (2007), *JavaScript* adalah bahasa pemrograman berbasis *browser*. Kode-kodenya ditulis langsung ke dalam *HTML* dari halaman-halaman web dan diterjemahkan serta dieksekusi sebagai respon terhadap aktivitas-aktivitas pada halaman *web*.

Dengan adanya *JavaScript*, maka teknik penulisan *HTML* dapat dilaksanakan dan dijalankan dengan dua cara, yakni dengan membuat program javascript untuk menghasilkan dokumen *HTML* atau dengan membuat dokumen *HTML* layaknya seperti biasa, sesudah itu jika tersedia program *JavaScript*, maka menambahkan program *JavaScript* tersebut sebagai sisipan saja. Beberapa hal tentang *JavaScript*:

1. *Javascript* didesain untuk menambah interaktif suatu web.
2. *Javascript* merupakan sebuah bahasa *scripting*.
3. Bahasa *scripting* merupakan bahasa pemrograman yang ringan.
4. *Javascript* berisi baris kode yang dijalankan di komputer (*web browser*).
5. *Javascript* biasanya disisipkan (*embedded*) dalam halaman *HTML*.
6. *Javascript* adalah bahasa *interpreter* (yang berarti skrip dieksekusi tanpa proses kompilasi).

2.10.5 *Bootstrap*

Bootstrap merupakan sebuah *framework CSS* yang memudahkan pengembang untuk membangun website yang menarik dan responsif. *Bootstrap* adalah *CSS* tetapi dibentuk dengan *LESS*, sebuah *pre-prosessor* yang memberi fleksibilitas dari penggunaan *CSS* biasa. *Bootstrap* dapat dikembangkan dengan tambahan lainnya karena ini cukup fleksibel terhadap pekerjaan *web* yang mengutamakan desain (Otto, 2011).

Bootstrap adalah sebuah *library framework* CSS yang dibuat khusus untuk bagian pengembangan *front-end website*. bootstrap merupakan salah satu

framework HTML, CSS dan *javascript* yang paling populer di kalangan *web developer*. pada saat ini hampir semua *web developer* telah menggunakan *bootstrap* untuk membuat tampilan *front-end* menjadi lebih mudah dan sangat cepat. karena anda hanya perlu menambahkan *class-class* tertentu untuk misalnya membuat tombol, *grid*, navigasi dan lainnya.

Bootstrap telah menyediakan kumpulan komponen *class interface* dasar yang telah dirancang sedemikian rupa untuk menciptakan tampilan yang menarik, bersih dan ringan. Selain komponen *class interface*, *bootstrap* juga memiliki fitur *grid* yang berfungsi untuk mengatur *layout* pada halaman *website* yang bisa digunakan dengan sangat mudah dan cepat. dengan menggunakan *bootstrap* kita juga diberi keleluasaan dalam mengembangkan tampilan *website* yang menggunakan *bootstrap* yaitu dengan cara mengubah tampilan *bootstrap* dengan menambahkan *class* dan CSS sendiri.

2.10.6 *CodeIgniter (CI)*

Menurut Blanco & Upton (2009:7) *CodeIgniter* adalah *powerful open source PHP framework* yang mudah dikuasai, dibangun untuk *PHP programmers* yang membutuhkan *toolkit* sederhana dan baik untuk membuat *full-featured* web *applications. CodeIgniter* merupakan aplikasi *Open Source* yang berupa *framework PHP* dengan model MVC (*Model, View, Controller*) untuk membangun *Website* dinamis dengan menggunakan *PHP*. *CodeIgniter* memudahkan *developer* untuk membuat aplikasi web dengan cepat mudah dibandingkan dengan membuatnya dari awal.

Model View Controller merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi web, berawal pada bahasa pemrograman *Small Talk*, *MVC* memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi yaitu:

- View, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi web bagian ini biasanya berupa file *template HTML*, yang diatur oleh

controller. *View* berfungsi untuk menerima dan merepresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.

- Model, biasanya berhubungan langsung dengan *database* untuk memanipulasi data (*insert, update, delete, search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
- *Controller*, merupakan bagian yang mengatur hubungan antara bagian model dan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

Dengan menggunakan prinsip MVC suatu aplikasi dapat dikembangkan sesuai dengan kemampuan *developernya*, yaitu *programmer* yang menangani bagian model dan *controller*, sedangkan *designer* yang menangani bagian *view*, sehingga penggunaan arsitektur MVC dapat meningkatkan *maintainability* dan organisasi kode.

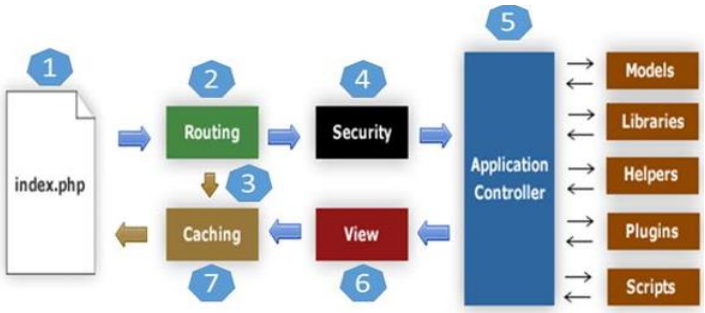
Ada beberapa kelebihan *CodeIgniter* (CI) dibandingkan dengan *Framework PHP* lain,

- Performa sangat cepat: salah satu alasan tidak menggunakan *framework* adalah karena eksekusinya yang lebih lambat daripada *PHP from the scratch*, tapi *Codeigniter* sangat cepat bahkan mungkin bisa dibilang *codeigniter* merupakan *framework* yang paling cepat dibanding *framework* yang lain.
- Konfigurasi yang sangat minim (*nearly zero configuration*) : tentu saja untuk menyesuaikan dengan *database* dan keleluasaan *routing* tetap diizinkan meplakukan konfigurasi dengan mengubah beberapa file konfigurasi seperti *database.PHP* atau *autoload.PHP*, namun untuk menggunakan *codeigniter* dengan setting standard, anda hanya perlu mengubah sedikit saja file pada folder *config*.
- Banyak komunitas: dengan banyaknya komunitas CI ini, memudahkan kita untuk berinteraksi dengan yang lain, baik itu bertanya atau teknologi terbaru.
- Dokumentasi yang sangat lengkap: Setiap paket instalasi *codeigniter* sudah disertai *user guide* yang sangat bagus dan lengkap untuk dijadikan permulaan, bahasanya pun mudah dipahami.

Kekurangan *CodeIgniter* sebagai berikut:

1. *Library* yang sangat terbatas. Hal ini dikarenakan sangat sulit mencari plugin tambahan yang terverifikasi secara resmi, karena pada situsnya *CodeIgniter* tidak menyediakan plugin-plugin tambahan untuk mendukung pengembangan aplikasi dengan CI.
2. Belum adanya editor khusus *CodeIgniter*, sehingga dalam melakukan *create project* dan modul-modulnya harus berpindah-pindah folder.

   Arsitektur *framework CodeIgniter* :



Gambar 2. 3 *Arsitektur framework*

Berikut adalah penjelasan cara kerja CodeIgniter:

1. Index.*PHP* bertindak sebagai controller terdepan, dan menginisialisasi resource yang diperlukan untuk menjalankan CodeIgniter.
2. Router memeriksa HTTP request untuk menentukan apa yang harus dikerjakan.
3. Jika cache file ada, maka akan ditampilkan langsung, dengan melewati eksekusi normal sistem.
4. Sebelum membuat controller, HTTP request akan memeriksa apa yang disubmit user dan memfilternya untuk keamanan.
5. Controller membuat model, core libraries, plugin, helper, dan resource lainnya untuk memproses permintaan tertentu.
6. View ditampilkan di browser sesuai proses yang dikerjakan controller. Jika caching dijalankan, view akan di-cache terlebih dahulu agar dapat ditampilkan di request selanjutnya.

2.10.7 *XAMPP*

XAMPP merupakan paket *PHP* dan *MySQL* berbasis *open source*, yang digunakan sebagai alat pembantu pengembangan aplikasi berbasis *PHP*. *XAMPP*

mengkombinasikan beberapa paket perangkat lunak berbeda ke dalam satu paket (Riyanto, 2010).

Di dalam Paket *XAMPP* terdapat tiga paket penting yaitu *Apache* sebagai *web server, PHP* sebagai bahasa pemrograman dan *MySQL* sebagai *database*. *Apache* adalah *server web* (*web server*) yang dapat dijalankan di banyak sistem operasi. *Apache* merupakan perangkat lunak *open-source* yang dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang dibawah naungan *Apache Software Foundation.*

2.10.8 *MySQL*

Menurut Winarno et al. (2014) yang mengemukakan bahwa *MySQL* adalah sebuah *software database*. *MySQL* merupakan tipe data relasional yang artinya *MySQL* menyimpan datanya dalam bentuk Tabel-tabel yang saling berhubungan. Keuntungan menyimpan data di *database* adalah kemudahannya dalam penyimpanan dan menampilkan data karena dalam bentuk tabel. Sedangkan Menurut Kustiyaningsih (2011), *MySQL* adalah sebuah basis data yang mengandung satu atau jumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau sejumlah tabel. *MySQL* merupakan sistem manajemen hubungan antara basis data yang sangat cepat dan sempurna. *MySQL* berupa alat bantu untuk memanipulasi basis data, sehingga bisa data dapat dengan mudah diisi, diambil, disusun dan diubah datanya. *Server MySQL* pun dapat mengatur kontrol akses dari data sehingga beberapa pengguna dapat sekaligus bekerja pada waktu yang bersamaan.

MySQL adalah *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi yang menggunakan *database* sebagai sumber dan pengolahan datanya, kepopuleran *MySQL* antara lain karena *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan, kinerja *query* sangat cepat dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan yang berskala kecil sampai menengah, *MySQL* juga bersifat tidak berbayar (Arief, 2011). Berikut beberapa kelebihan *MySQL* sebagai berikut yaitu:

1. Kemampuan yang tinggi.
2. Tidak dibutuhkan biaya untuk mendapatkan MySQL

3. Mudah untuk konfigurasi dan dipelajari
4. Dapat dijalankan pada beberapa sistem operasi seperti sistem *Linux* dan *Microsoft Windows*.

2.11 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan suatu teknik yang digunakan menguji apakah sebuah perangkat lunak yang dihasilkan telah sesuai dengan yang diharapkan atau belum. Menurut (Pressman, 2015), pengujian adalah proses eksekusi suatu program untuk menemukan kesalahan sebelum digunakan oleh pengguna akhir (*end-user*).

2.11.1 Pengujian *Scenario Test*

Dalam penelitian ini menggunakan teknik *test case* berdasarkan *skenario test*. *Test Case* adalah serangkaian rancangan tindakan yang ingin dilaksanakan untuk memverifikasi fitur tertentu atau fungsi dari aplikasi perangkat lunak atau *website*. *Test case* adalah pengujian yang dilakukan berdasarkan beberapa masukan seperti kondisi dan hasil yang telah ditentukan sebelumnya (Romeo, 2003).

Sebuah *Scenario Test* biasanya memiliki isi *pre-conditions, expected results* dan *post-conditions*. *Test case* bertindak sebagai titik awal untuk pelaksanaan tes dan setelah mengaplikasikan sekumpulan nilai input, aplikasi memiliki hasil yang definitif dan meninggalkan sistem di beberapa titik akhir atau juga dikenal sebagai *post-condition* eksekusi.

Untuk pengujian perangkat lunak, pembuatan *test case* adalah langkah mendasar. Kemudian prosedur pengujian dirancang untuk *test case* tersebut, dan kemudian skrip dibuat untuk menerapkan prosedur. *Test case* merupakan kunci untuk proses karena mengidentifikasi dan mengkomunikasikan kondisi yang akan diimplementasikan pada *test* dan yang diperlukan untuk memverifikasi keberhasilan.

Tabel 2. 7 Keterangan Kode

Kode	Keterangan Kode
V	*Valid* : Menunjukkan bahwa komponen yang membentuk skenario memiliki nilai yang benar atau *valid*

Kode	Keterangan Kode
I	*Invalid* : Menunjukkan bahwa komponen yang membentuk skenario memiliki nilai yang salah atau *invalid*
NA	*Not Access* : Menunjukkan bahwa komponen yang membentuk skenario tersebut tidak memiliki peranan atau tidak bisa diakses pada saat tertentu

2.11.2 Pengujian *User Acceptance Test* (*UAT*)

User Acceptance Test (UAT) atau Uji Penerimaan Pengguna adalah suatu proses pengujian oleh pengguna yang dimaksudkan untuk menghasilkan dokumen yang dijadikan bukti bahwa perangkat lunak yang telah dikembangkan telah dapat diterima oleh pengguna. Pengujian *UAT* dilakukan oleh pengguna target dari aplikasi yang dibangun untuk mengetahui aplikasi berjalan sesuai dengan proses yang telah ditentukan bertujuan untuk mencatat dan mengoreksi *bug* yang ditemukan sebelum aplikasi di rilis. *UAT* biasanya dinilai oleh pengguna aplikasi tersebut, dengan menggunakan kuesioner sesuai *test case* yang diinginkan.

Proses pengujian UAT diawali dengan menyediakan dokumentasi persyaratan bisnis, kemudian dilanjutkan dengan proses bisnis (alur kerja) atau skenario dan yang terakhir yaitu pengujian menggunakan data (Branch, 2008). Cara untuk mendapatkan suatu data penilaian sebagai bukti yang nyata yang dilakukan end-user salah satunya adalah menggunakan kuesioner. Kuesioner adalah teknik pengumpulan data dengan cara peneliti memberikan daftar pertanyaan atau pernyataan yang tertulis untuk dijawab oleh responden (Sugiyono, 2014). Pengujian *UAT* didasarkan pada dokumen *requirement* yang disepakati bersama. Dokumen *requirement* adalah dokumen yang berisi lingkup pekerjaan perangkat lunak yang harus dikembangkan, dengan demikian maka dokumen ini semestinya menjadi acuan untuk pengujian.

Proses dalam *UAT* meliputi pemeriksaan dan pengujian terhadap hasil pekerjaan. Setelah diperiksa selanjutnya memastikan item-item yang ada dalam dokumen *requirement* sudah ada dalam perangkat lunak yang diuji atau tidak. Selanjutnya menguji semua item yang telah ada telah dapat memenuhi kebutuhan penggunanya. Hasil dari UAT adalah dokumen yang menunjukkan bukti pengujian, berdasarkan bukti pengujian inilah dapat diambil kesimpulan, apakah *software*

yang diuji telah dapat diterima atau tidak. Dokumen dapat berupa kuesioner uji *test case*.

Tabel 2. 8 Pilihan Jawaban *UAT*

Jawaban	**Keterangan**
A	Sangat Buruk
B	Buruk
C	Cukup
D	Baik
E	Sangat Baik

Tabel 2. 9 Bobot Nilai Jawaban

Jawaban	**Bobot**
A. Sangat Buruk	1
B. Buruk	2
C. Cukup	3
D. Baik	4
E. Sangat Baik	5

Tabel 2. 10 Kriteria Interpretasi Skor

Jawaban	**Bobot**
81% - 100%	Sangat Baik
61% - 80%	Baik
41% - 60%	Cukup Baik
21% - 40%	Kurang Baik
0% - 20%	Sangat Kurang Baik