

BAB II

LANDASAN TEORI

2.1 Tinjauan Jurnal

Setelah peneliti melakukan telaah terhadap beberapa penelitian, ada beberapa yang memiliki keterkaitan dengan pembangunan aplikasi absensi menggunakan *iBeacon* sebagai berikut:

Penelitian Xiangjie Li (2016) yang berjudul “*Design and Implementation of Indoor Positioning System Based on iBeacon*” menjelaskan bahwa meningkatnya permintaan penentuan informasi lokasi terutama di dalam ruangan dengan menggunakan perangkat *mobile*. Tujuan dari penelitian ini adalah untuk merancang dan menerapkan sistem informasi lokasi dalam ruangan berbasis *mobile* dengan teknologi *BluetoothLow Energy* berdasarkan *iBeacon*. Penelitian ini merancang dan menerapkan sistem penentuan posisi dalam ruangan dengan mengadopsi metode filter Gaussian dan metode filter Kalman yang tidak beralasan untuk mengekstrak sinyal dengan kuat dari perangkat *iBeacon*. Hasil penelitian ini sistem dapat menunjukkan posisi dengan menggunakan *mobile* terminal dan sistem ini bisa berjalan baik di sistem android maupun iOS. Metode ini memiliki kinerja yang lebih baik dibandingkan dengan metode WiFi di karenakan hasil percobaan menunjukkan bahwa kesalahan hanya 4 meter dan sistem ini dapat mencapai posisi yang akurat dan kuat.

Penelitian Maria Varsamou and Theodore Antonakopoulos (2014) yang

berjudul “*A Bluetooth Smart Analyzer in iBeacon Networks*” mengatakan bahwa baru-baru ini telah diusulkan agar posisi di dalam ruangan dapat dicapai perangkat sensor BLE yang di sebut *iBeacons* di berbagai lokasi di tempat tertentu. Tujuan dari penelitian ini adalah menyajikan aplikasi berbasis android untuk menganalisis jaringan *iBeacon* dan menentukan lokasi sinyal terbaik. Penelitian ini juga menunjukkan analisa yang di sajikan untuk mengukur pola radiasi perangkat *iBeacon*. Metode penelitian ini dengan menggunakan perangkat lunak berbasis Android untuk mengumpulkan statistik mengenai variasi waktu temporal dan variasi spasial RSSI yang diamati di jaringan *iBeacon*. Penelitian ini juga menunjukkan penerapan analisis ini untuk mengukur pola radiasi perangkat *iBeacon*. Hasil percobaan menunjukkan bahwa metode pengukuran yang diusulkan memberikan alat yang andal dan berguna untuk menganalisis radiasi perangkat *iBeacon*.

Penelitian Yang Yang, Zhouchi Li dan Kaveh Pahlavan (2016) yang berjudul “*Using iBeacon for Intelligent In-Room Presence Detection*” mengatakan bahwa mengetahui jumlah pengunjung yang ada di dalam ruangan bisa dilakukan dengan teknologi *iBeacon*. Tujuan dari penelitian ini adalah menutup pintu masuk secara otomatis setelah seseorang masuk atau keluar dari ruangan dan mengirimkan data ke server. Metode yang di gunakan dalam penelitian ini adalah dengan menggunakan *single iBeacon approach* yang hanya menggunakan satu buah *iBeacon*. Hasil dari penelitian ini adalah menyelidiki dan mengembangkan sistem deteksi keberadaan di dalam ruangan berbasis *iBeacon* untuk mencatat pengguna di sebuah ruangan. Kinerja optimal dari penelitian yang di lakukan ini bisa mencapai 100%.

Penelitian Jingjing Yang (2015) yang berjudul “*An iBeacon based Indoor Positioning Systems for Hospitals*” dalam penelitiannya menjelaskan bahwa membantu pasien menemukan departemen atau tempat mereka di rumah sakit adalah masalah yang perlu dipecahkan oleh rumah sakit dengan segera. Tujuan dari penelitian ini adalah menentukan posisi di dalam ruangan berbasis *iBeacon* untuk rumah sakit. Penelitian ini pertama-tama adalah menganalisis keuntungan dari *iBeacon* dibandingkan dengan teknologi penentuan posisi dalam ruangan biasa. Metode yang digunakan dalam penelitian ini adalah mendeteksi keberadaan pasien dan memberikan lokasi dimana pasien itu berada. untuk merekomendasikan departemen atau unit terdekat kepada pasien. Hasil dari penelitian ini adalah, sistem mampu memberitahukan keberadaan departemen di dalam ruangan rumah sakit dan juga keberadaan kamar yang di rekomendasikan kepada pasien.

Tabel II.1 Perbedaan Tinjauan Jurnal dengan Penelitian yang diusulkan

Nama peneliti	Judul	Metode	Hasil penelitian	Perbedaan
Xiangjie Li (2016)	<i>Design and Implementat ion of Indoor Positioning System Based on iBeacon</i>	Penelitian ini merancang dan menerapkan sistem penentuan posisi dalam ruangan dengan mengadopsi metode filter Gaussian dan	Sistem dapat menunjukkan posisi dengan menggunakan mobile terminal dan sistem ini bias berjalan baik di sistem android maupun iOS.	Metode yang akan digunakan oleh penulis adalah <i>Double iBeacons Approach</i> sedangkan metode yang digunakan

		metode filter Kalman yang tidak beralasan untuk mengekstrak sinyal dengan kuat dari perangkat <i>iBeacon</i> .		penulis sebelumnya adalah metode filter Gaussian dan metode filter Kalman.
Maria Varsamou and Theodore Antonakopoulos (2014)	<i>ABluetooth Smart Analyzer in iBeacon Networks</i>	Metode penelitian ini dengan menggunakan perangkat lunak berbasis Android untuk mengumpulkan statistik mengenai variasi waktu temporal dan variasi spasial RSSI yang diamati di jaringan <i>iBeacon</i> .	Hasil percobaan menunjukkan bahwa metode pengukuran yang diusulkan memberikan alat yang andal dan berguna untuk menganalisis radiasi perangkat <i>iBeacon</i> .	Metode yang akan digunakan oleh penulis adalah dengan perangkat lunak berbasis iOS dan menyimpan data di <i>local storage</i> perangkat iOS. Sementara metode penelitian yang digunakan oleh penulis sebelumnya adalah dengan menggunakan perangkat lunak

				berbasis android.
Yang Yang, Zhouchi Li dan Kaveh Pahlavan (2016)	<i>Using iBeacon for Intelligent In-Room Presence Detection</i>	Metode yang di gunakan dalam penelitian ini adalah dengan menggunakan <i>single iBeacon approach</i> yang hanya menggunakan satu buah <i>iBeacon</i> .	Hasil dari penelitian ini adalah menyelidiki dan mengembangkan sistem deteksi keberadaan di dalam ruangan berbasis <i>iBeacon</i> untuk mencatat pengguna di sebuah ruangan	Metode yang digunakan oleh penulis adalah <i>double iBeacon approach</i> . Sementara metode pada penulis sebelumnya dengan menggunakan <i>single iBeacon approach</i> .
Jingjing Yang (2015)	<i>An iBeacon-based Indoor Positioning Systems for Hospitals</i>	Metode yang digunakan dalam penelitian ini adalah mendeteksi keberadaan pasien dan memberikan lokasi dimana pasien itu	Hasil dari penelitian ini adalah, sistem mampu memberitahukan keberadaan departemen di dalam ruangan rumah sakit dan juga keberadaan kamar yang di	Metode yang digunakan oleh penulis adalah mendeteksi kedatangan dan kepergian pengguna sementara penelitian sebelumnya adalah

		berada.	rekomendasikan kepada pasien.	mendeteksi dan memberikan lokasi kepada pengguna.
--	--	---------	----------------------------------	--

2.2 Konsep Dasar Program

Dalam penulisan skripsi ini, penulis menggunakan konsep pemrograman terstruktur dikarenakan mudah untuk dimengerti dan memiliki algoritma pemecahan masalah yang sederhana, standar dan efektif menurut (Rosa dan Shalahudin, 2011).

2.2.1 Pemrograman Terstruktur

Pemrograman terstruktur adalah konsep atau paradigma atau sudut pandang pemrograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer. Modul-modul (pembagian program) biasanya dibuat dengan mengelompokkan fungsi-fungsi dan prosedur-prosedur yang diperlukan sebuah proses tertentu. Fungsi-fungsi dan prosedur-prosedur ditulis secara sekunsial atau terurut dari atas ke bawah sesuai dengan kebergantungan antarfungsi atau prosedur (fungsi atau prosedur yang dapat dipakai oleh fungsi atau prosedur dibawahnya harus yang sudah ditulis atau dideklarasikan di atasnya). Pemodelan pada pemrograman terstruktur dibagi berdasarkan fungsi-fungsi dan prosedur-prosedur. Oleh karena itu, pemodelan pada pemrograman terstruktur lebih fokus kepada cara bagaimana memodelkan data dan fungsi-fungsi atau prosedur-prosedur yang harus dibuat. Jenis paradigm pemrograman yang dapat digunakan untuk

membuat program, baru setelah itu ditentukan paradig pemrograman apa yang akan digunakan (Rosa dan Shalahudin, 2011).

2.2.2 Konsep Dasar Terstruktur

Pemrograman terstruktur adalah proses yang berorientasi kepada teknik yang digunakan untuk merancang dan menulis program secara jelas dan konsisten. Pemodelan Data merupakan suatu teknik yang berorientasi kepada data dengan menunjukkan sistem hanya datanya saja terlepas dari bagaimana data tersebut akan diproses atau digunakan untuk menghasilkan informasi. Rekayasa Informasi merupakan perpaduan dari pemodelan data dan proses, juga memberikan penekanan baru terhadap pentingnya perencanaan sistem informasi (Rosa dan Shalahudin, 2011).

2.2.3 Aplikasi Mobile

Menurut (Lim, 2015) Selama lima tahun terakhir, perangkat mobile, seperti *smartphone*, dan tablet telah jauh lebih populer dari pada perangkat *desk-based* tradisional seperti komputer pribadi dan laptop. Sejak iOS telah dirilis iPhone pada 2007, dan Android telah dirilis pada berbagai *smartphone* dan tablet pada tahun 2008, Sistem Operasi paling populer yang berjalan pada perangkat komputasi telah menjadi sistem operasi *mobile*. Dengan demikian, semakin banyak aplikasi yang berjalan pada Sistem Operasi *mobile* daripada Sistem Operasi *desktop*.

2.2.4 Pengertian Aplikasi

Menurut Nazrudin Safaat H (2012) aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas. Misalnya termasuk perangkat lunak perusahaan, *software* akuntansi, perkantoran, grafis perangkat lunak dan pemutar media. Dapat disimpulkan bahwa aplikasi merupakan *software* yang

berfungsi untuk melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data. Program aplikasi merupakan program siap pakai. Program yang dirancang untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasi ialah program pemroses kata dan Web Browser. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung.

Menurut Nazrudin Safaat H (2012) klasifikasi aplikasi dapat dibagi menjadi 2 (dua) yaitu:

1. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
2. Aplikasi paket, dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

2.2.5 Pengertian Aplikasi *Mobile*

Menurut Nazrudin Safaat H (2012) aplikasi *mobile* berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan *mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Maka aplikasi *mobile* dapat di artikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah–pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil. Aplikasi *mobile* ini dapat diakses melalui perangkat nirkabel, pager, PDA, telepon seluler, *smartphone*, dan

perangkat sejenisnya.

2.2.6 Karakteristik Perangkat *Mobile*

Menurut Nazrudin Safaat H (2012) Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain dan layout, tetapi mereka memiliki karakteristik yang sangat berbeda dari sistem desktop. Berikut karakteristik perangkat *mobile* menurut Nazrudin Safaat H (2012), diantaranya adalah:

1. Ukuran yang kecil

Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

2. Memori yang terbatas

Perangkat *mobile* juga memiliki memory yang kecil, yaitu *primary* (RAM) dan *secondary* (disk). Pembatasan ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis dari perangkat ini. Dengan pembatasan jumlah dari memori, pertimbangan-pertimbangan khusus harus diambil untuk memelihara pemakaian dari sumber daya yang mahal ini.

3. Daya proses yang terbatas

Sistem *mobile* tidaklah setangguh rekan mereka yaitu desktop. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi status dari sumber daya ini. Seperti harddisk dan RAM, Anda dapat menemukan mereka dalam ukuran yang pas dengan sebuah kemasan kecil.

4. Mengonsumsi daya yang rendah

Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin *desktop*. Perangkat ini harus menghemat daya karena mereka berjalan pada

keadaan dimana daya yang disediakan dibatasi oleh baterai-baterai.

5. Kuat dan dapat diandalkan

Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

6. Konektivitas yang terbatas

Perangkat *mobile* memiliki *bandwith* rendah, beberapa dari mereka bahkan tidak tersambung. Kebanyakan dari mereka menggunakan koneksi *wireless*.

7. Masa hidup yang pendek

Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala. Coba ambil kasus sebuah *handphone*, mereka *booting* dalam hitungan detik dan kebanyakan orang tidak mematikan *handphone* mereka bahkan ketika malam hari. PDA akan menyala jika anda menekan tombol *power* mereka.

2.2.7 Bluetooth

Menurut (www.bluetooth.com/Pages/Basic.aspx, 31-07-2017) *Bluetooth* merupakan teknolog komunikasi nirkabel jarak pendek yang menggunakan gelombang radio dalam melakukan pertukaran data. *Bluetooth* dikenal dengan karakteristiknya yang simple, hemat energi dan tingkat keamanannya yang tinggi. Sampai dengan hari ini, walaupun sudah termasuk teknologi kuno, *Bluetooth* masih ada hampir disemua alat komunikasi modern. Ketika pertama kali diciptakan oleh Ericsson pada tahun 1994 *bluetooth* hanyalah sebagai alternatif wireless dari kabel RS 232. Kini *Bluetooth* dapat kita gunakan untuk bertukar data yang lebih kompleks seperti musik, video dan gambar. Untuk dapat saling berkomunikasi pertama-tama 2

device yang ingin berkomunikasi harus menyalakan *Bluetooth* lalu melakukan *pairing* satu sama lain. Ketika sudah terhubung (melakukan *pairing*) maka sudah dapat terjadi komunikasi/pertukaran data, 2 *device* yang saling terhubung ini juga akan menciptakan suatu jaringan komputer yang disebut dengan istilah PAN (*Personal Area Network*). Untuk jarak yang dapat dijangkau oleh *bluetooth* sendiri cukup bervariasi, mulai dari 1 meter sampai dengan 100 meter, tergantung dari jenis gelombang radio yang digunakan.

2.2.8 Bluetooth Low Energy

Bluetooth Low Energy sering disebut dengan *Bluetooth Smart*, *bluetooth* yang membutuhkan daya rendah dalam kerjanya (A VeriFone White Paper, 2013). Dikatakan dalam (<http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons>, 31-07-2017) bahwa hanya dengan sumber daya dari baterai berbentuk koin (seperti baterai bios komputer) sebuah modul BLE dapat bertahan hingga 3 tahun. BLE adalah sebuah *Personal Area Network* (PAN) yang memancarkan sinyal data dengan jarak tertentu. Syarat yang harus dimiliki oleh sebuah perangkat sehingga dia bisa bekerja dalam kategori BLE adalah bahwa perangkat tersebut harus sudah tertanamkan *bluetooth* versi 4.0. Beberapa perangkat dengan *bluetooth* versi ini seperti iPhone 4s atau seri di atasnya. Tidak hanya versi *bluetooth* saja yang disyaratkan, untuk iPhone atau iPad juga harus telah terinstal versi iOS 7. Sedangkan untuk *smartphone* Android versi sistem operasi yang terinstal yaitu minimal *Jelly Bean* versi 4.3.

2.2.9 iBeacon

iBeacon adalah implementasi dari teknologi BLE yang memungkinkan perangkat iOS atau Android untuk bisa mendeteksi sinyal yang dipancarkan oleh *iBeacon* tersebut dan memberikan informasi dimanakah posisi perangkat tersebut terhadap *iBeacon* (<http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons>, 31-07-2017). Tetapi posisi yang dimaksud bukanlah posisi secara fisik seperti lintang dan bujur, melainkan posisi relatif *receiver* terhadap *iBeacon* apakah dia dalam *rangeimmediate*, *near* atau *far*. Untuk bisa mengetahui estimasi posisi *receiver* terhadap *iBeacon* secara pasti maka perlu adanya peta secara lokal, yaitu menentukan sebuah lokasi atau ruang atau tempat tertentu, kemudian dari lokasi tersebut dipetakan dalam bentuk koordinat. Dengan meletakkan beberapa *iBeacon* pada koordinat tertentu dalam lokasi tersebut akan dapat digunakan untuk mengetahui estimasi posisi *receiver* terhadap *iBeacon*. *Positioning* dengan cara seperti ini bisa dikatakan berbasis *context aware*.

2.2.10 Sistem Operasi iOS pada Smartphone

Adelphia (2015) mendefinisikan iOS adalah sistem operasi yang dikembangkan oleh perusahaan Apple untuk ponsel iPhone, tetapi kemudian berkembang dan dapat digunakan ke dalam perangkat Apple yang lainnya seperti iPod Touch, Apple TV dan iPad. Menurut Adelphia (2015) sistem operasi ini bersifat tertutup dan hanya bisa digunakan oleh perangkat Apple, jadi anda tidak akan menemukan sistem operasi iOS pada perangkat serupa dengan merek lain. Didalam iOS juga terdapat komponen *abstraction layers*, yaitu lapisan sistem iOS yang terbagi menjadi empat bagian,

seperti *framework* yang berfungsi untuk membangunkan *user* ke *hardware* (Adelphia, 2015). Menurut Adelphia (2015) dalam iOS juga terdapat komponen Darwin yang masih satu keluarga dengan *UNIX*, beberapa lapisan dalam iOS adalah:

- a) Lapisan yang berhubungan langsung ke hardware atau disebut juga *Core OS Layer*.
- b) Bagian yang berisi layanan yang membentuk sistem dasar semua aplikasi, yaitu *Core Service Layer*.
- c) Sementara itu, bagian untuk grafis disebut Media Layer. Lapisan ini terbentuk untuk mengarahkan *audio*, *video*, dan teknologi grafis lainnya menjadikan iOS kaya akan multimedia.
- d) Yang terakhir adalah *Cocoa Touch layer*, bagian ini berfungsi untuk interaksi antara user dan optimasi *focus*.

Keempat bagian tersebut bersatu dalam iOS. Sistem operasi ini juga dikonsep untuk dapat bekerja dengan baik dilayar sentuh (Adelphia, 2015). Menurut Adelphia (2015) ada beberapa fitur menarik yang bisa anda temukan di sistem operasi iOS dalam iPhone, diantaranya:

- a) *User friendly*.
- b) Kemampuan untuk bekerja secara *multi tasking*.
- c) Desain yang elegan.
- d) Banyak pengembang yang memberikan aplikasi untuk diunduh melalui App Store.
- e) Ukuran memori yang cukup besar. Sampai saat buku ini ditulis masih tersedia ukuran 8-128 GB dan juga Sulit terserang oleh virus.

- f) *Upgrade* sistem operasi dapat dengan mudah dilakukan melalui *smartphone* ataupun PC.

Namun menurut Adelphia (2015) banyak hal yang mungkin tidak disukai oleh beberapa individu ketika hendak memakai sistem operasi ini, diantaranya adalah:

- a) Harga yang relatif mahal. Belum diketahui sebabnya kenapa merek Apple selalu mempunyai harga yang diatas rata-rata.
- b) Belum tersedia pemasangan memori tambahan (*eksternal*).
- c) Tidak dapat melakukan modifikasi terhadap sistem operasi.
- d) *Output* suara masih terlalu kecil.

Berikut ini daftar versi iPhone beserta iOS yang digunakan dari awal kemunculannya hingga sekarang (Adelphia, 2015)

Tabel II.2 Perkembangan iPhone

Sumber: Adelphia (2015)

Versi	iOS versi	Prosesor	Kamera
iPhone	iOS 3.1.3	620 Mhz	2 MP
iPhone 3G	iOS 4.2.1	412 Mhz arm 11	2 MP
iPhone 3Gs	iOS 6.0	600 Mhz Cortex A8	3 MP
iPhone 4	iOS 6.0	1 Ghz Cortex A8	5 MP
iPhone 4s	iOS 6.0	Dual Core 1 Ghz A9	8 MP
iPhone 5	iOS 6.0	Dual Core 1,2 Ghz	8 MP
iPhone 5s	iOS 7.0	Dual Core 1,3 Ghz	8 MP
iPhone 6	iOS 8.0	Dual Core 1,4 Ghz	8 MP

2.2.11 Database

Menurut Connolly dan Begg (2010:65), *database* adalah sekumpulan data tersebar yang berhubungan secara logis, dan penjelasan dari data ini dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi.

Menurut Inmon (2011:493), *database* adalah sekumpulan data yang saling berhubungan yang disimpan (biasanya dengan redundansi yang terkontrol dan terbatas) berdasarkan skema. Sebuah *database* dapat melayani *single* atau *multiple applications*.

Menurut Gottschalk dan Saether dalam jurnal (2010:41), *database* adalah sekumpulan data yang terorganisir untuk mendukung banyak aplikasi secara efisien dengan memusatkan data dan mengontrol *data redundant*.

Berdasarkan definisi-definisi yang dijabarkan oleh para ahli di atas, maka dapat disimpulkan *database* adalah sekumpulan data yang saling berhubungan dan terorganisir yang disimpan berdasarkan skema dengan memusatkan data dan mengontrol *data redundant* untuk memenuhi kebutuhan informasi dari suatu organisasi.

2.2.12 Database Management System (DBMS)

Menurut (Hoffer, Prescott, & Topi, 2011: 11) *Database Management System* (DBMS) merupakan sebuah sistem perangkat lunak yang dapat membuat,

menciptakan dan memelihara akses ke pengguna *database*. Menurut Kroenke and Auer (2012:11) DBMS adalah suatu program komputer yang digunakan untuk membuat, memproses, dan mengelola *database*.

Menurut Connolly & Begg (2010:66) *Database Management System* (DMS) adalah sebuah *software system* yang memungkinkan *user* mendefinisi, membentuk dan mengatur *database* dan mengendalikan akses ke *database* DBMS berinteraksi dengan pengguna aplikasi program dan *database*.

DBMS menurut Connolly & Begg (2010:66) menyediakan fasilitas:

1. *Data Definition Language* (DDL), yang berguna untuk menspesifikasikan tipe data, struktur dan *constraint* data.
2. *Data Manipulation Language* (DML), yang berguna untuk memberikan fasilitas *query* data.
3. Pengendalian akses *database*, antara lain mengontrol:
 - a. Keamanan sistem: Mencegah *user* yang tidak memiliki hak untuk mengakses *database*.
 - b. Integrasi Sistem: Menjaga konsistensi data.

Jadi, dari beberapa pengertian diatas dapat disimpulkan bahwa DBMS merupakan sebuah sistem perangkat lunak yang memungkinkan pengguna untuk dapat membuat, menciptakan, memelihara dan mengontrol akses dari *database* ke pengguna *database*.

2.2.13 MySQL

Menurut Sidik (2012, p.333), *MySQL* merupakan *software database* yang termasuk paling populer di lingkungan *Linux*, kepopuleran ini karena ditunjang karena performansi *query* dari *databasenya* yang saat itu bisa dikatakan paling cepat, dan jarang bermasalah. *MySQL* telah tersedia juga di lingkungan *Windows*. *PHP* secara *default* telah mendukung *MySQL* karena *MySQL* telah dimiliki oleh Oracle, dimana mengembangkan *database* yang murni *open source* dan *freeware* dengan nama *MariaDB*.

MySQL diciptakan di negara Swedia oleh perusahaan *MySQL AB*. Adapun masing-masing nama yang berjasa dalam menciptakan *MySQL* adalah David Axmark, Alan Larsson, dan Michael “Monty” Widenius. Perangkat lunak ini tersebar luas secara gratis karena memiliki lisensi *GNU General Public License*. Sampai sekarang, tercatat ada beberapa bersinkronisasi dengan *MySQL*, seperti C, C++, C#, bahasa pemrograman yang cukup populer *Eiffel*, bahasa pemrograman *Smalltalk*, bahasa pemrograman *Java*, bahasa pemrograman *Lisp*, *Perl*, *PHP*, bahasa pemrograman *Python*, *Ruby*, *REALbasic*, dan *Tcl* Sidik (2012, p.333).

Menurut Saputra, Subagio, dan Saluky (2012, p.7), *MySQL* bekerja menggunakan *SQL Language (Structure Query Language)*, yang diartikan bahwa *MySQL* merupakan standard penggunaan *database* di dunia untuk pengolahan data. Pada umumnya, perintah yang paling sering digunakan dalam *MySQL* adalah *SELECT*, *INSERT*, *UPDATE*, dan *DELETE*. *SQL* juga menyediakan perintah untuk membuat *database*, *field*, ataupun *index* untuk menambah atau menghapus data. *MySQL* juga merupakan *database* yang mampu berjalan di semua sistem operasi,

powerful, dan sangat mudah untuk dipelajari, dan hosting servernya juga banyak mengadopsi *MySQL* sebagai *standard database*. Ada juga beberapa kelebihan *MySQL* menurut Saputra, Subagio, dan Saluky (2012, p.7) yaitu:

1. Bersifat *open source*, yang memiliki kemampuan untuk dapat dikembangkan lagi.
2. Menggunakan bahasa *SQL*, yang merupakan standar bahasa dunia dalam pengolahan data.
3. *Super performance* dan *reliable*, pemrosesan *database*-nya sangat cepat dan stabil.
4. Sangat mudah dipelajari
5. Memiliki dukungan *support* pengguna *MySQL*.
6. Mampu lintas *platform*, dapat berjalan di berbagai sistem operasi.
7. *Multiuser*, di mana *SQL* dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami konflik.

2.2.14 PHP My Admin

Bunafit (2013:15), *PHPMyAdmin* adalah aplikasi manajemen *database server MySQL* berbasis *web*. Dengan aplikasi *phpMyAdmin* kita bias mengelola *database* sebagai *root* atau juga sebagai *user* biasa, kita bisa membuat *database* baru, mengelola *database* dan melakukan operasi perintah-perintah *database* secara lengkap seperti saat kita di *MySQLPromp*.

2.2.15 PHP

Menurut Mehdi (2007) *PHP* merupakan singkatan dari *PHP Hypertext Preprocessor*, ia merupakan bahasa berbentuk skrip yang ditempatkan dalam *server* dan diproses di *server*. Secara khusus *PHP* dirancang untuk membentuk aplikasi *web* dinamis artinya ia dapat membentuk suatu tampilan berdasarkan permintaan terkini. Kelahiran *PHP* bermula ketika Rasmus Lerdorf membuat sejumlah skrip *perl* yang dapat mengamati siapa saja yang melihat-lihat daftar riwayat hidupnya., yakni pada tahun 1994. Skrip ini selanjutnya dikemas menjadi *tool* yang disebut dengan “*Personal Home Page*”. Pada tahun 1995, Rasmus menciptakan *PHP/FI* Versi 2. Pada versi inilah pemrograman dapat menempelkan kode terstruktur di dalam tag *HTML*. *PHP* juga dapat berkomunikasi dengan *database* dan melakukan banyak perhitungan yang kompleks. Pada saat ini *PHP* cukup populer sebagai piranti pemrograman *Web*, terutama dilingkungan *Linux*, Walaupun demikian, *PHP* sebenarnya juga dapat berfungsi pada *server-server* yang berbasis *UNIX*, *Windows*, dan *Macintosh*. Pada awalnya *PHP* dirancang untuk diintegrasikan dengan *web server Apache*. Namun belakangan ini *PHP* juga dapat bekerja dengan *Web Server* seperti *PWS (Personal Web Server)*, *IIS (Internet Information Server)* dan *Xitami*.

2.2.16 CodeIgniter

Menurut Hakim (2010:8) CodeIgniter adalah sebuah *framework* *PHP* yang dapat membantu mempercepat developer dalam pengembangan aplikasi *web* berbasis *PHP* dibanding jika menulis semua kode program dari awal. CodeIgniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc. (<http://ellislab.com>), sebuah perusahaan yang memproduksi *CMS (Content Management System)* yang cukup handal, yaitu

Expression Engine (<http://www.expressionengine.com>). Saat ini, CodeIgniter dikembangkan dan dimaintain oleh *Expression Engine Development Team*. Adapun beberapa keuntungan menggunakan CodeIgniter menurut Hakim (2010:8), diantaranya:

1. Gratis

CodeIgniter berlisensi dibawah Apache/BSD *open source*.

2. Ditulis Menggunakan PHP 4

Meskipun CodeIgniter dapat berjalan di PHP 5, namun sampai saat ini kode program CodeIgniter masih dibuat dengan menggunakan PHP 4.

3. Berukuran Kecil Ukuran

CodeIgniter yang kecil merupakan keunggulan tersendiri. Dibanding dengan *framework* lain yang berukuran besar.

4. Menggunakan Konsep MVC

CodeIgniter menggunakan konsep MVC yang memungkinkan pemisahan *layerapplication-logic* dan *presentation*.

5. URL yang Sederhana

Secara default, URL yang dihasilkan CodeIgniter sangat bersih dan *Serach Engine Friendly* (SEF).

6. Memiliki Paket *Library* yang Lengkap

CodeIgniter mempunyai *library* yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oleh sebuah aplikasi berbasis web, misalnya mengakses *database*, mengirim email, memvalidasi form, menangani *session* dan sebagainya.

7. *Extensible*

Sistem dapat dikembangkan dengan mudah menggunakan *plugin* dan *helper*, atau dengan menggunakan *hooks*.

8. Tidak Memerlukan *Template Engine*

Meskipun CodeIgniter dilengkapi dengan *template parser* sederhana yang dapat digunakan, tetapi hal ini tidak mengharuskan kita untuk menggunakannya.

9. Dokumentasi Lengkap dan Jelas

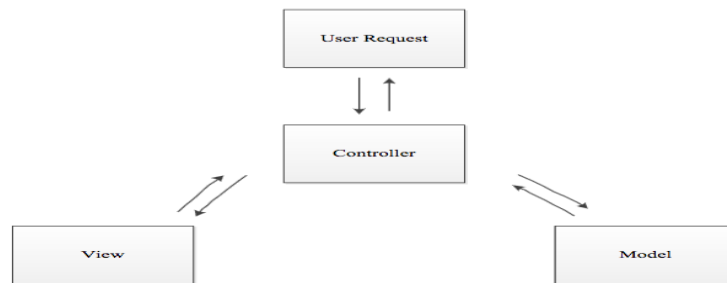
Dari sekian banyak *framework*, CodeIgniter adalah satu-satunya *framework* dengan dokumentasi yang lengkap dan jelas.

10. Komunitas

Komunitas CodeIgniter saat ini berkembang pesat. Salah satu komunitasnya bisa dilihat di (<http://codeigniter.com/forum/>).

2.2.17 MVC (*Model-View-Controller*)

Menurut Hakim (2010: 4) CodeIgniter adalah *frameworkPHP* yang dibuat berdasarkan kaidah *Model-View-Controller*. Dengan MVC, maka memungkinkan pemisahan antara *layerapplication-logic* dan *presentation*. Sehingga, dalam sebuah pengembangan web, seorang *programmer* bisa berkonsentrasi pada *core-system*, sedangkan web *designer* bisa berkonsentrasi pada tampilan web. Menariknya, skrip *PHP*, *queryMySQL*, *Javascript* dan *CSS* bisa saling terpisah, tidak dibuat dalam satu skrip berukuran besar yang membutuhkan *resource* besar pula untuk mengesekusinya. Adapun alur program aplikasi berbasis *framework* Codeigniter dapat dilihat pada gambar II.1.



Gambar II.1 *Model-View-Controller*

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

CodeIgniter

Gambar diatas menurut Hakim (2010: 4) menerangkan bahwa ketika datang sebuah *user request*, 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 dibrowser.

Menurut Hakim (2010: 4) dalam konteks CodeIgniter dan aplikasi berbasis web, maka penerapan konsep MVC mengakibatkan kode program dapat dibagi menjadi tiga kategori, yaitu:

1. *Model*

Kode program (berupa OOP *class*) yang digunakan untuk memanipulasi *database*.

2. *View*

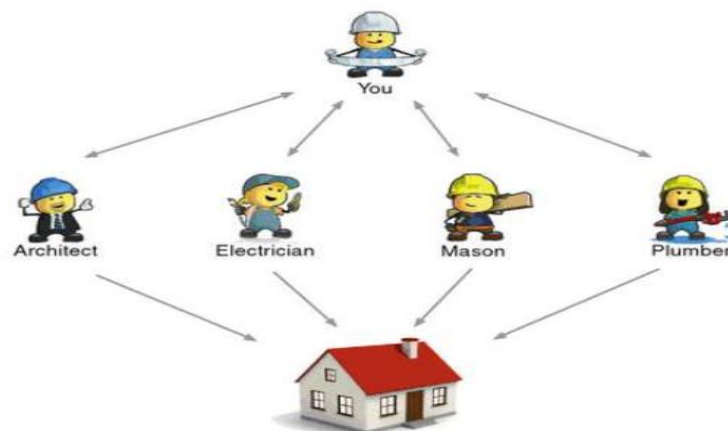
Berupa *templatehtml/xml* atau *php* untuk menampilkan data pada *browser*

3. Controller

Kode program (berupa OOP *class*) yang digunakan untuk mengontrol aliran aplikasi (sebagai pengontrol *model* dan *View*)

2.2.18 API

API merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain. Penjelasan ini dapat dicontohkan dengan analogi apabila akan dibangun suatu rumah. Dengan menyewa kontraktor yang dapat menangani bagian yang berbeda, pemilik rumah dapat memberikan tugas yang perlu dilakukan oleh kontraktor tanpa harus mengetahui bagaimana cara kontraktor menyelesaikan pekerjaan tersebut. Dari analogi tersebut, rumah merupakan *software* yang akan dibuat, dan kontraktor merupakan API yang mengerjakan bagian tertentu dari *software* tersebut tanpa harus diketahui bagaimana prosedur dalam melakukan pekerjaan tersebut (Reddy, 2011).



Gambar II.2 Analogi API pada Pembangunan Rumah

(Sumber: *API Design for C*, Reddy, 2011)

Interface pada *software* merupakan suatu *entry points* yang digunakan untuk mengakses seluruh *resources* yang terdapat di dalam *software* tersebut. Dengan adanya API, maka terdapat aturan bagaimana *software* dapat berinteraksi dengan *software* lain untuk mengakses *resources* melalui *interface* yang telah tersedia.

Menurut (3Scale Networks, 2011) secara struktural, API merupakan spesifikasi dari suatu *data structure*, *objects*, *functions*, beserta parameter-parameter yang diperlukan untuk mengakses *resource* dari aplikasi tersebut. Seluruh spesifikasi tersebut membentuk suatu *interface* yang dimiliki oleh aplikasi untuk berkomunikasi dengan aplikasi lain, dan API dapat digunakan dengan berbagai bahasa *programming*, ataupun hanya dengan menggunakan URL (*Uniform Resource Locator*) yang telah disediakan oleh suatu *website*. API dapat diklasifikasikan menjadi beberapa kategori, hal ini dilihat dari abstraksi apa yang dideskripsikan di dalam sistem. Kategori-kategori ini diantaranya:

Tabel II.3 Kategori API

Sumber: 3Scale Networks (2011)

Kategori API	Deskripsi	Contoh
<i>Operating System</i>	API yang digunakan untuk fungsi dasar yang dapat dilakukan oleh komputer. Seperti proses I/O, eksekusi program.	API for MS Windows
<i>Programming Languages</i>	API yang digunakan untuk	Java API

	memperluas kapabilitas dalam melakukan eksekusi terhadap suatu bahasa pemrograman.	
<i>Application Services</i>	API yang digunakan untuk mengakses data dan layanan yang disediakan dari suatu aplikasi.	API for mySAP (BAPI/ <i>Business Application Programming Interface</i>)
<i>Infrastructure Services</i>	Digunakan untuk mengakses infrastruktur dari suatu komputer. Infrastruktur disini adalah komputer beserta <i>peripheral</i> seperti <i>storage</i> , aplikasi, dan lain-lain.	Amazon EC2 (<i>Elastic Compute Cloud</i>) untuk akses untuk <i>virtual computing</i> dan Amazon S3 (<i>Simple Storage Service</i>) untuk menyimpan data dalam jumlah besar.
<i>Web Services</i>	API yang digunakan untuk mengakses <i>content</i> dan layanan yang disediakan oleh suatu <i>webapplication</i> .	<i>Facebook Graph</i> API yang digunakan untuk mengakses informasi yang dapat dibagikan.

2.2.19 REST (*Representational State Transfer*)

REST (*Representational State Transfer*) merupakan jenis arsitektur yang terdapat pada *web* untuk melayani suatu *service*. REST merepresentasikan cara interaksi antara *server* dan *client* untuk melakukan proses pertukaran informasi melalui media yang sama. Dalam suatu jaringan, agar suatu *resource* dapat diakses, maka diperlukan identifikasi dan suatu bentuk manipulasi. Dapat digunakan URI (*Uniform Resource Identifier*) yang digunakan untuk mengidentifikasi *resource* yang ada pada suatu jaringan, dan dapat membuat *resource* menjadi *addressable*, yang berarti *resource* dapat diketahui lokasinya dan dapat dimanipulasi dengan menggunakan suatu aplikasi (Webber, Parastatidis, & Robinson, 2010: 5).

REST dapat digunakan sebagai *interface* dari API untuk mengakses suatu *resource*. API yang mengikuti prinsip dari REST *architecture* memberikan kemudahan bagi *developer* untuk tidak perlu mengetahui bagaimana struktur dari API di dalam *server*. Dalam hal ini, *server* akan memberikan informasi bagaimana agar *client* dapat mengakses *service* melalui API yang telah disediakan.

Penggunaan protokol HTTP pada REST *architecture* untuk komunikasi antara *client* dan *server* terletak pada HTTP *method*, yaitu *GET*, *POST*, *PUT*, dan *DELETE*. *Method* ini dapat digunakan untuk mengakses *resources* yang ada pada *server*, bergantung dari instruksi yang diberikan oleh *server*.

Dengan menggunakan protokol HTTP, URI dapat dijadikan sebagai media yang digunakan untuk mengakses *resource* dari *server*. Hal ini disebut dengan URI *tunneling*.

URI *tunneling* mempergunakan URI untuk mentransfer informasi pada antar sistem yang dalam jaringan dengan melakukan *encode* pada URI itu sendiri. Dengan mengirim HTTP *method* yang telah disebutkan sebelumnya, *server* dapat melakukan eksekusi terhadap suatu program yang menghasilkan atau mengambil suatu *resource* dan mengirimkannya kembali ke *client*. Dalam proses ini terjadi proses *mapping* dari URI menjadi *method call* pada *server* yang dituju (Webber, Parastatidis, & Robinson, 2010: 37).

2.2.20 JSON

Menurut Deitel (2012, p1303) JSON (*JavaScript Object Notation*) adalah suatu format pertukaran data komputer. Format dari JSON adalah berbasis teks, dapat terbaca oleh manusia, digunakan untuk mempresentasikan struktur data sederhana, dan tidak bergantung dengan bahasa apapun. Biasanya digunakan pada aplikasi Ajax. Format JSON sering digunakan untuk mentransmisikan data terstruktur melalui koneksi jaringan. Secara umum, JSON digunakan untuk mentransmisikan data antara *server* dan aplikasi *web*. Jenis media internet yang resmi untuk JSON adalah aplikasi/json. Format JSON sering digunakan untuk serialisasi dan mengirimkan data terstruktur melalui koneksi jaringan, terutama untuk pengiriman data antara server dan aplikasi *web* melayani sebagai alternatif ke XML.

2.2.21 Macintosh

Macintosh atau yang biasa disebut Mac adalah salah satu jenis komputer yang diproduksi oleh Apple *Computer* Inc. Macintosh diperkenalkan pertama kali pada Januari 1984. Macintosh adalah komputer pertama yang memperkenalkan sistem antar muka grafis (GUI) (Setiawan, 2008).

2.2.22 Mac OS

Mac OS adalah singkatan dari *Macintosh Operating System*, Mac OS adalah sistem operasi yang dibuat oleh Apple *ComputerInc.* yang dikhususkan untuk komputer Macintosh dan tidak kompatibel dengan PC berbasis IBM, Namun mulai tahun 2006 Mac OS sudah bisa digunakan pada komputer dengan arsitektur x86. Sistem operasi ini pertama kali diperkenalkan bersamaan dengan komputer Macintosh (Setiawan, 2008).

2.2.23 iOS SDK dan Xcode

iOS SDK (*Software Development Kit*) yang bekerja pada komputer Macintosh menyediakan antarmuka, *tools* dan semua sumber yang digunakan untuk membangun aplikasi iOS. (Apple, *iOS Technology Overview*, 2010). Apple memberikan sebagian besar sistem untuk antarmuka mereka ke dalam paket yang disebut dengan *framework* atau kerangka kerja. *Framework* adalah sebuah direktori yang berisi *library* dan sumber-sumber seperti *header files*, gambar, aplikasi bantuan, dan lain-lain yang digunakan untuk mendukung *library* yang tersedia. *Framework* digunakan dengan cara *me-link framework* yang diinginkan ke dalam proyek aplikasi. Menghubungkan *framework* dengan proyek yang dibuat memberi akses ke semua fitur dari *framework* tersebut dan juga memungkinkan untuk *developmenttools* mengetahui dimana menemukan suatu *file* tertentu (Apple, *iOS Technology Overview*, 2010, pp. 14-15).

Komponen yang ada pada SDK menurut Apple, *iOS Technology Overview* (2010, p. 15) yaitu:

Xcode *Tools* – alat yang mendukung pengembangan aplikasi iOS, termasuk:

1. Xcode – sebuah alat yang mendukung pengembangan untuk mengelola proyek aplikasi dan memungkinkan untuk mengedit, mengkompilasi, menjalankan dan memperbaiki *error* pada kode. Xcode terintegrasi dengan alat pendukung lainnya dan merupakan aplikasi utama yang digunakan selama pengembangan.
2. *Interface Builder* – alat yang digunakan untuk merancang antarmuka secara visual. Objek antarmuka yang dibuat disimpan pada suatu *file* dan dimuat ke aplikasi saat *runtime*.
3. *Instruments* – alat yang digunakan untuk menganalisa performa dan mendeteksi kesalahan. *Instruments* bisa digunakan untuk mendapatkan informasi mengenai perilaku *runtime* dari aplikasi dan mengidentifikasi kemungkinan kesalahan.
4. *iOS Simulator* – aplikasi Mac OS X yang mensimulasikan teknologi iOS, mengizinkan pengembang untuk menguji aplikasi iOS secara lokal pada komputer Macintosh yang digunakan.
5. *iOS Developer Library* – referensi dan dokumentasi konsep yang menjelaskan semua tentang teknologi iOS dan proses pengembangan aplikasi.

Walaupun aplikasi bisa dicoba pada *iOS Simulator*, pengembang juga dimungkinkan untuk menguji aplikasinya ke piranti yang sesungguhnya menggunakan Xcode dan *Instruments*. *iOS Simulator* ideal untuk membangun dan menguji aplikasi secara cepat namun tidak bisa menggantikan pengujian pada piranti sesungguhnya. Untuk mencoba aplikasi ke salah satu piranti iOS, pengembang harus

terdaftar pada iOS *Developer Program* (Apple, *iOS Technology Overview*, 2010, p. 15).

2.2.24 Cocoa

Cocoa dan Cocoa Touch *framework* yang bekerja untuk Mac OS X dan iOS adalah sebuah kerangka kerja yang terintegrasi ke dalam Xcode. Cocoa itu sendiri merupakan salah satu API (*Application Programming Interface*) berbasis objek milik Apple Inc. API Cocoa yang bertingkat tinggi membuat pengembang lebih mudah untuk menambahkan animasi, jaringan, tampilan *platform* asli, dan karakter untuk aplikasi dengan hanya beberapa baris kode. Cocoa *framework* terdiri dari *libraries*, API dan *runtimes*. Dengan menggunakan Cocoa, pengembang dapat membangun aplikasi dengan cara yang sama dengan Mac OS X. Aplikasi otomatis akan mempunyai karakter dan tampilan yang sama dengan Mac OS X. Cocoa dengan menggunakan Xcode adalah cara yang paling baik untuk membuat aplikasi untuk Mac (Apple, Cocoa: Mac OS X *Technology Overview*, 2011).

2.2.25 Cocoa Touch

Cocoa Touch, berbeda dengan Cocoa, merupakan ekstensi dari Cocoa untuk aplikasi pada sistem operasi iOS pada iPhone, iPad dan iPod Touch. Cocoa Touch digunakan untuk menambahkan fungsi sensor gerak dan animasi.

Cocoa Touch mengatur dan mengelola interaksi pengguna pada iOS. Karena Cocoa Touch merupakan ekstensi dari Cocoa, maka semua teknologi Cocoa Touch berasal dari Cocoa. Cocoa Touch dan antarmuka iOS benar-benar dirancang ulang untuk menangani *multi-touch*. Semua fitur antarmuka yang biasa digunakan pada

platform iOS seperti iPhone, iPad dan iPod Touch telah disediakan pada Cocoa Touch *framework* (Apple, Cocoa: Mac OS X *Technology Overview*, 2011).

Dibangun di atas paradigma *Model-View-Controller*, Cocoa Touch menyediakan dasar yang kuat untuk menciptakan sebuah aplikasi menarik. Ketika dikombinasikan dengan alat pengembang pada Xcode seperti *Interface Builder*, keduanya akan menjadi mudah dan menyenangkan ketika menggunakan sistem *drag-and-drop* pada saat pembuatan aplikasi.

2.2.26 UIKit

UIKit adalah sebuah *framework* yang menyediakan kelas-kelas untuk membangun dan mengatur perancangan antarmuka aplikasi iOS. UIKit menyediakan objek, *event handling*, penggambaran model, *windows*, *views*, dan kontrol yang didesain secara spesifik untuk antarmuka layar sentuh (Apple, *UIKit FrameworkReferences*, 2011, p. 26).

2.2.27 Swift

Menurut Neuburg (2014) swift adalah bahasa pemrograman baru yang dikembangkan oleh Apple untuk membangun aplikasi berbasis iOS dan OS X. Swift dibangun menggunakan bahasa pemrograman C dan Objective C. Swift mengadopsi *safe programming pattern* dan menambahkan fitur-fitur modern yang membuat *programming* lebih mudah, lebih fleksibel dan menyenangkan. Swift didukung oleh Cocoa dan *CocoaTouch framework* yang telah matang dan banyak disukai. Swift sudah dikembangkan selama bertahun-tahun. Apple meletakkan dasar untuk Swift dengan meningkatkan *compiler* yang telah ada. *Debugger*, dan infrastruktur, menyederhanakan manajemen memori dengan *Automatic Reference Counting* (ARC).

Swift adalah bahasa pemrograman yang mudah dipelajari untuk pemula, *syntax* yang *expressive* membuat belajar Swift lebih menyenangkan. Swift menyediakan fitur *playgrounds* yang memungkinkan programmer untuk bereksperimen dan langsung melihat hasilnya tanpa perlu *build* dan menjalankan sebuah aplikasi. Swift menggabungkan bahasa pemikiran pemrograman modern dengan budaya *Engineering* Apple, Apple mengoptimalkan kinerja *compiler* dan proses *development* yang lebih mudah tanpa mengorbankan performa untuk sekedar membuat “Hello world” ataupun aplikasi kompleks. Ini yang membuat swift sebagai investasi untuk *developer* dan Apple di masa depan.

2.2.28 Core Data

Core Data adalah sebuah framework yang menyediakan solusi untuk tugas-tugas umum yang berhubungan dengan siklus objek dan pengelolaan objek (Apple, Core Data, 2010, pp. 17-19).

Fitur dari Core Data antara lain:

1. Menyediakan bantuan untuk melakukan undo dan redo.
2. Mampu mengatur konsistensi dari relasi antar objek.
3. Dapat mengurangi memori yang berlebihan dari suatu program dengan memuat objek-objek.
4. Melakukan validasi terhadap nilai properti dari sebuah objek secara otomatis.
5. Mengizinkan pengguna mengubah skema yang telah ada dengan lebih mudah.
6. Menggunakan *NSFetchedResultsController* untuk terintegrasi dengan Cocoa pada Mac OS X.

7. Mengumpulkan accessor (getter –metoda yang mengembalikan nilai dari private class) yang sesuai untuk relasi to-many.
8. Mengelompokkan dan mengorganisasi data pada memori dan pada antarmuka.
9. Mendukung penyimpanan objek pada tempat penyimpanan data eksternal.
10. *Compile query* dengan baik.
11. Menyediakan fitur *locking* untuk mengatasi permasalahan pada *multi-writer*.

2.2.29 Alamofire

Menurut (<http://nshipster.com/alamofire>, 01-08-2017) Alamofire adalah *package* yang digunakan untuk mempermudah pemanggilan API. Di bahasa pemrograman swift sudah tersedia fungsi untuk pemanggilan API, namun penggunaannya sangat tidak *developer friendly*. Dengan menggunakan Alamofire penulisan kode untuk pemanggilan API jadi lebih singkat dan tentunya lebih mudah untuk dibaca oleh *developer*. Berikut adalah contoh pemanggilan API menggunakan Alamofire.

Contoh *codehttp request* dengan menggunakan Alamofire menurut (<http://nshipster.com/alamofire>, 01-08-2017)

```
Alamofire.request(.GET, "http://httpbin.org/get")
    .response{ (request, response, data, error) in
println(request)
println(response)
println(error)
}
```

2.2.30 Metode Waterfall

Menurut Pressman (2015:42), model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Nama model ini

sebenarnya adalah “*Linear Sequential Model*”. Model ini sering disebut juga dengan “*classic life cycle*” atau metode *waterfall*. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.

Fase-fase dalam *Waterfall Model* menurut referensi Pressman (2015:17):

1. *Communication (Project Initiation & Requirements Gathering)*

Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan *customer* demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi *software*. Pengumpulan data-data tambahan bisa juga diambil dari jurnal, artikel, dan internet.

2. *Planning (Estimating, Scheduling, Tracking)*

Tahap berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan *tracking* proses pengerjaan sistem.

3. *Modeling (Analysis & Design)*

Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur *software*, tampilan *interface*, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

4. *Construction (Code&Test)*

Tahapan Construction ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

5. *Deployment (Delivery, Support, Feedback)*

Tahapan *Deployment* merupakan tahapan implementasi *software* ke *customer*, pemeliharaan *software* secara berkala, perbaikan *software*, evaluasi *software*, dan pengembangan *software* berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya.

2.3 Metode Algorithma

Algoritma yang digunakan pada pembuatan skripsi ini adalah model Algoritma *Floyd Warshall* untuk menentukan jalur terdekat.

2.3.1 Algoritma *Floyd Warshall*

Algoritma *Floyd-Warshall* adalah sebuah algoritma analisis graf untuk mencari bobot minimum dari graf berarah. Dalam satu kali eksekusi algoritma, akan didapatkan jarak sebagai jumlah bobot dari lintasan terpendek antar setiap pasang

simpul tanpa memperhitungkan informasi mengenai simpul-simpul yang dilaluinya. Algoritma ini yang juga dikenal dengan nama Roy-Floyd. Dalam pengertian lain Algoritma *Floyd-Warshall* adalah suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu. (Novandi.R.A.D, 2007)

2.3.2 Karakteristik Algoritma *Floyd Warshall*

Beberapa karakteristik yang dimiliki oleh algoritma *Floyd-Warshall* menurut Novandi.R.A.D (2007) antara lain:

1. Persoalan dibagi atas beberapa tahap, yang setiap tahapnya hanya akan diambil satu keputusan.
2. Masing-masing tahap terdiri atas sejumlah status yang saling berhubungan dengan status tersebut. Status yang dimaksud di sini adalah berbagai kemungkinan masukan yang ada pada tahap tersebut.
3. Ketika masuk ke suatu tahap, hasil keputusan akan transformasi.
4. Bobot pada suatu tahap akan meningkat secara teratur seiring bertambahnya jumlah tahapan.
5. Bobot yang ada pada suatu tahap tergantung dari bobot tahapan yang telah berjalan dan bobot pada tahap itu sendiri.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan pada tahap sebelumnya.

7. Terdapat hubungan rekursif yang menyatakan bahwa keputusan terbaik dalam setiap status pada tahap k akan memberikan keputusan terbaik untuk setiap status pada tahap $k + 1$.
8. Prinsip optimalitas berlaku pada persoalan yang dimaksud.

2.3.3 Pseudocode Algoritma Floyd Warshall

Menurut Novandi.R.A.D (2007) *pseudo-code* algoritma *Floyd Warshall* adalah sebagai berikut:

```
function fw(int[1..n,1..n] graph) {
    // Inisialisasi
    var int[1..n,1..n] jarak := graph
    var int[1..n,1..n] sebelum
    for i from 1 to n
    for j from 1 to n
        if jarak[i,j] < Tak-hingga
            sebelum[i,j] := i
    // Perulangan utama pada algoritma
    for k from 1 to n
    for i from 1 to n
    for j from 1 to n
        if jarak[i,j] > jarak[i,k] +
            jarak[k,j]
            jarak[i,j] = jarak[i,k] + jarak[k,j]
            sebelum[i,j] = sebelum[k,j]
    return jarak
}
```

2.4 Pengujian Aplikasi

Pengujian merupakan bagian yang penting dalam siklus pengembangan perangkat lunak. Tujuan dari pengujian ini adalah untuk menjamin bahwa perangkat lunak yang dibangun memiliki kualitas yang handal Pressman (2012). Pengujian terhadap program itu sendiri bertujuan agar program dapat berjalan dengan baik tanpa mengalami gangguan atau *error*, dan memungkinkan untuk dilakukannya

pengembangan sistem lebih lanjut. Pengujian aplikasi ini menggunakan metode pengujian *black box*. Pengujian *black box* ini tidak perlu tahu apa yang sesungguhnya terjadi dalam sistem atau perangkat lunak, yang diuji adalah masukan serta keluarannya.

2.4.1 Black Box Testing

Seringkali perangkat lunak mengandung kesalahan pada proses-proses tertentu pada saat perangkat lunak sudah berada di tangan *user*. Kesalahan-kesalahan (*error*) pada perangkat lunak ini sering disebut dengan *bug* maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke pelanggan atau selama perangkat lunak masih terus dikembangkan (Rossa dan Shalahuddin, 2013:271). Pengujian tidak hanya untuk meminimalisasai kesalahan secara teknis tapi juga kesalahan non teknis, misalnya pengujian pesan kesalahan sehingga *user* tidak bingung atau tidak mengerti dengan pesan kesalahan yang muncul, atau juga jika masukan dan keluaran yang diperlukan berkapasitas sangat besar (Rossa dan Shalahuddin, 2013:272).

2.5 Peralatan Pendukung

Dalam kegiatan pembangunan aplikasi absensi otomatis dengan menggunakan *iBeacon* ini diperlukan peralatan pendukung untuk memperlancar dan mengidentifikasi *database* yang ada. Adapun peralatan pendukung yang berperan dalam proses pembuatan aplikasi ini antara lain:

2.5.1 Data Flow Diagram (DFD)

Kristanto (2008:61) *Data Flow Diagram* merupakan suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan

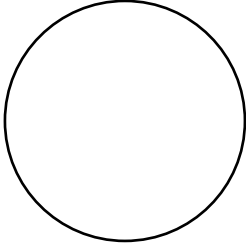
data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut.



Sukamto dan Shalahuddin (2014:288) *Data Flow Diagram* atau dalam Bahasa Indonesia menjadi Diagram Alir Data (DAD) adalah representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengatur dari masukan (*input*) dan keluaran (*output*). DFD tidak sesuai untuk memodelkan sistem yang menggunakan pemrograman berorientasi objek.

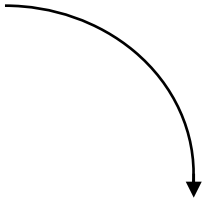
Sukamto dan Shalahuddin (2014:71) notasi-notasi pada DFD (Edward Yourdon dan Tom DeMarco) adalah sebagai berikut:

Tabel II.4 Simbol-simbol *Data Flow Diagram*

Sumber: Sukamto dan Shalahuddin (2014:71)

NOTASI	KETERANGAN
	<p>Proses atau fungsi atau prosedur pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang harusnya menjadi fungsi atau prosedur di dalam kode program Catatan:</p> <p>Nama yang diberikan pada sebuah proses biasanya berupa kata kerja.</p>

	<p><i>File</i> atau basis data atau penyimpanan (<i>storage</i>) pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemograman terstruktur, maka pemodelan notasi inilah yang harusnya dibuat menjadi tabel-tabel basis data yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel basis data yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel basis data (<i>Entity Relationship Diagram</i> (ERD), <i>Conceptual Data Model</i> (CMD), <i>Physical Data Model</i> (PDM))</p> <p>Catatan:</p> <p>Nama yang diberikan pada sebuah penyimpanan biasanya kata benda.</p>
	<p>Entitas luar (<i>external entity</i>) atau masukan (<i>input</i>) atau keluaran (<i>output</i>) atau orang yang memakai atau berinteraksi dengan perangkat lunak yang dimodelkan atau sistem lain yang terkait dengan aliran data dari sistem yang dimodelkan.</p> <p>Catatan:</p>

	Nama yang digunakan pada masukan (<i>input</i>) atau keluaran (<i>output</i>) biasanya berupa kata benda.
	<p>Aliran data merupakan data yang dikirim antar proses, dari penyimpanan ke proses, atau dari proses ke masukan (<i>input</i>) atau keluaran (<i>output</i>).</p> <p>Catatan:</p> <p>Nama yang digunakan pada aliran data biasanya berupa kata benda, dapat diawali dengan kata data misalnya “data siswa” atau tanpa kata data misalnya “siswa”.</p>

Menurut Sukanto dan Shalahuddin (2014:72), berikut ini adalah tahapan-tahapan perancangan dengan menggunakan *DFD*:

1. Membuat *DFD Level 0* atau sering disebut juga *Context Diagram DFD Level 0* menggambarkan sistem yang akan dibuat sebagai suatu entitas tunggal yang berinteraksi dengan orang maupun sistem lain. *DFD Level 0* digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar.
2. Membuat *DFD Level 1* *DFD Level 1* digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan. *DFD*

Level1 merupakan hasil *breakdown DFD Level 0* yang sebelumnya sudah dibuat.

3. Membuat *DFD Level 2* Modul-modul pada *DFD Level 1* dapat di *breakdown* menjadi *DFD Level 2*. Modul mana saja yang harus di *breakdown* lebih detail tergantung pada tingkat kedetilan modul tersebut. Apabila modul tersebut sudah cukup *detail* dan *rinci* maka modul tersebut sudah tidakperlu untuk di *breakdown* lagi. Untuk sebuah sistem, jumlah *DFD Level 2* sama dengan jumlah modul pada *DFD Level 1* yang di *breakdown*.
4. Membuat *DFD Level 3* dan seterusnya. *DFD Level 3, 4, 5* dan seterusnya merupakan *breakdown* dari modul pada *DFD Level* di atasnya. *Breakdown* pada *level 3, 4* dan *5* dan seterusnya aturannya sama persis dengan *DFD Level 1* atau *Level 2*.

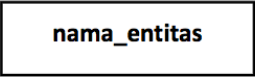
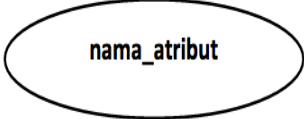
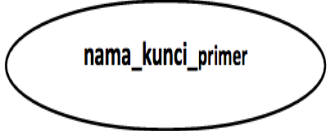
2.5.2 Entity Relationship Diagram (ERD)

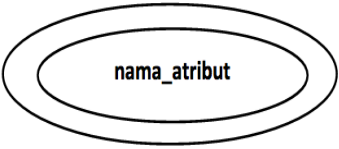

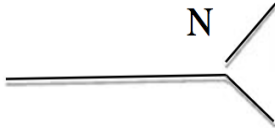
Sukamto dan Shalahuddin (2014:289), *Entitiy Relationship Diagram (ERD)* adalah pemodelan awal basis data yang akan dikembangkan berdasarkan teori himpunan dalam bidang matematika untuk pemodelan basis data relasional.

Sukamto dan Shalahuddin (2014:50), ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen). Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), notasi Crow's Foot, dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi dari Chen. Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen:

Tabel II.4 Simbol-simbol *Entity Relationship Diagram* (ERD)

Sumber : Sukamto dan Shalahuddin (2014:50)

SIMBOL	DESKRIPSI
<p>Entitas / <i>Entity</i></p> 	<p>Entitas merupakan data inti yang akan disimpan bakal tabel pada basis data benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer penamaan entitas biasanya lebih kekata bendadan belum merupakan nama tabel.</p>
<p>Atribut</p> 	<p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas.</p>
<p>Atribut Kunci Primer</p> 	<p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagaikunci akses <i>record</i> yang diinginkan biasanya berupa id kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).</p>

	<p>Atribut multinilai/ <i>multivalued</i></p> <p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.</p>
<p>Relasi</p> 	<p>Relasi yang menghubungkan antar entitas biasanya diawali dengan kata kerja.</p>
<p>Asosiasi/<i>association</i></p> 	<p>Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki <i>multiplicity</i> kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan <i>one to many</i> menghubungkan entitas A dan entitas B.</p>




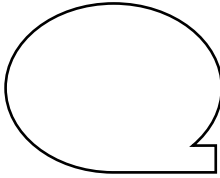
2.5.3 Flow Chart

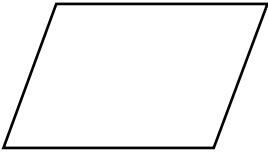
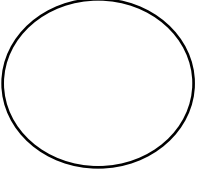
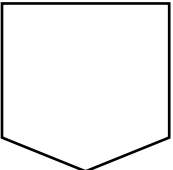

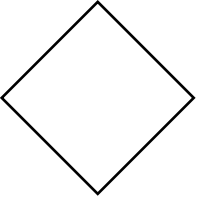
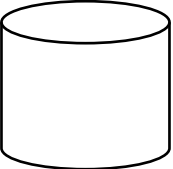

Indrajani (2015:36), *Flow chart* adalah penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program.


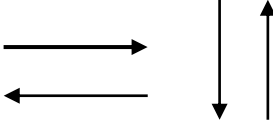
Indrajani (2015:38), menjelaskan simbol-simbol dalam *Flow Chart* adalah sebagai berikut:

Tabel II.5 Simbol-simbol *Flow Chart*

Sumber: Indrajani (2015:38)

SIMBOL	KETERANGAN
	Simbol <i>Start</i> atau <i>End</i> yang mendefinisikan awal atau akhir dari sebuah <i>flowchart</i> .
	Simbol pemrosesan yang terjadi pada sebuah alur kerja.
	Simbol yang menyatakan bagian dari program (sub program).
	Simbol masukan atau keluaran dari atau ke sebuah pita <i>magnetic</i> .

	<p>Simbol <i>Input/Output</i> yang mendefinisikan masukan dan keluaran proses.</p>
	<p>Simbol konektor untuk menyambung proses pada lembar kerja yang sama.</p>
	<p>Simbol konektor untuk menyambung proses pada lembar kerja yang berbeda.</p>
	<p>Simbol masukan atau keluaran dari atau ke sebuah dokumen.</p>
	<p>Simbol untuk memutuskan proses lanjutan dari kondisi tertentu.</p>
	<p>Simbol <i>database</i> atau basis data.</p>
	<p>Simbol yang menyatakan piranti keluaran, seperti layar monitor, <i>printer</i>.</p>

	Simbol yang mendefinisikan proses yang dilakukan secara manual.
	Simbol untuk menghubungkan antar proses atau antar simbol.

2.5.4 Kamus Data

Sukamto dan Shalahuddin (2014:73), Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*output*) dapat dipahami secara umum (memiliki standar cara penulisan).

Sukamto dan Shalahuddin (2014:73), menjelaskan simbol-simbol yang digunakan dalam kamus data, yaitu:

Tabel II.6 Simbol-simbol dalam kamus data

Sumber: Sukamto dan Shalahuddin (2014:73)

SIMBOL	ARTI
=	disusun atau terdiri atas
+	Dan
[]	baik ...atau...
{ } ⁿ	n kali diulang/ bernilai banyak
()	data operasional
...	batas komentar

Kamus data pada *DFD* nanti harus dapat dipetakan dengan hasil perancangan basis data yang dilakukan sebelumnya. Jika ada kamus data yang tidak dapat dipetakan pada tabel hasil perancangan basis data berarti hasil perancangan basis data dengan perancangan dengan *DFD* masih belum sesuai, sehingga harus ada yang diperbaiki baik perancangan basis datanya, perancangan *DFD*-nya, atau keduanya Sukamto dan Shalahuddin (2014:73).