

**PANDUAN LENGKAP MEMBANGUN SISTEM
MONITORING KINERJA MAHASISWA INTERNSHIP
BERBASIS GLOBAL POSITIONING SYSTEM**

**PANDUAN LENGKAP MEMBANGUN SISTEM
MONITORING KINERJA MAHASISWA INTERNSHIP
BERBASIS GLOBAL POSITIONING SYSTEM**

Aip Suprpto Munari

Student

Penulis

ISBN

Editor

Penyunting Dan

lain-lain

Quotes

CONTRIBUTORS

CONTENTS IN BRIEF

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

LISTINGS

KATA PENGANTAR

Buku ini merupakan panduan lengkap membangun sistem monitoring kinerja mahasiswa *internship* berbasis *global positioning system*.

ACKNOWLEDGMENTS

ACRONYMS

SYMBOL

INTRODUCTION

BAB I

INTERNSHIP

1.1 Latar Belakang

Program *internship* merupakan program yang dikhususkan bagi mahasiswa yang telah memiliki pengetahuan (*knowledge*) Program Studi D4 Teknik Informatika minimal 5 Semester. Kebutuhan untuk pemenuhan kemampuan (*skill*) yang dimiliki

menjadi dasar dalam program *internship* ini. Upaya mendekatkan kurikulum yang berbasis kompetensi dan kebutuhan industri mendorong program ini dilakukan dengan pola yang disesuaikan dengan kondisi pegawai/pekerja. Hal ini dimaksudkan untuk interpretasi *knowledge* yang dimiliki serta meningkatkan (*enrichment*) pengetahuan dalam bidang Teknik Informatika melalui praktek langsung. Harapan besar agar pengetahuan menjadi kompetensi yang unggul melalui pengenalan dan pengayaan terhadap kemampuan (*skill*) dalam *internship* ini.

1.2 Tujuan

1. Untuk memberikan kesempatan mahasiswa Politeknik Pos Indonesia dengan kesempatan sebelum terjun ke dunia kerja dalam upaya memperoleh pengalaman kerja.
2. Memfasilitasi perusahaan.
3. Mengembangkan pertukaran pengetahuan dan pengalaman diantara Politeknik Pos Indonesia dengan beberapa perusahaan yang tertarik untuk melakukan kerjasama.

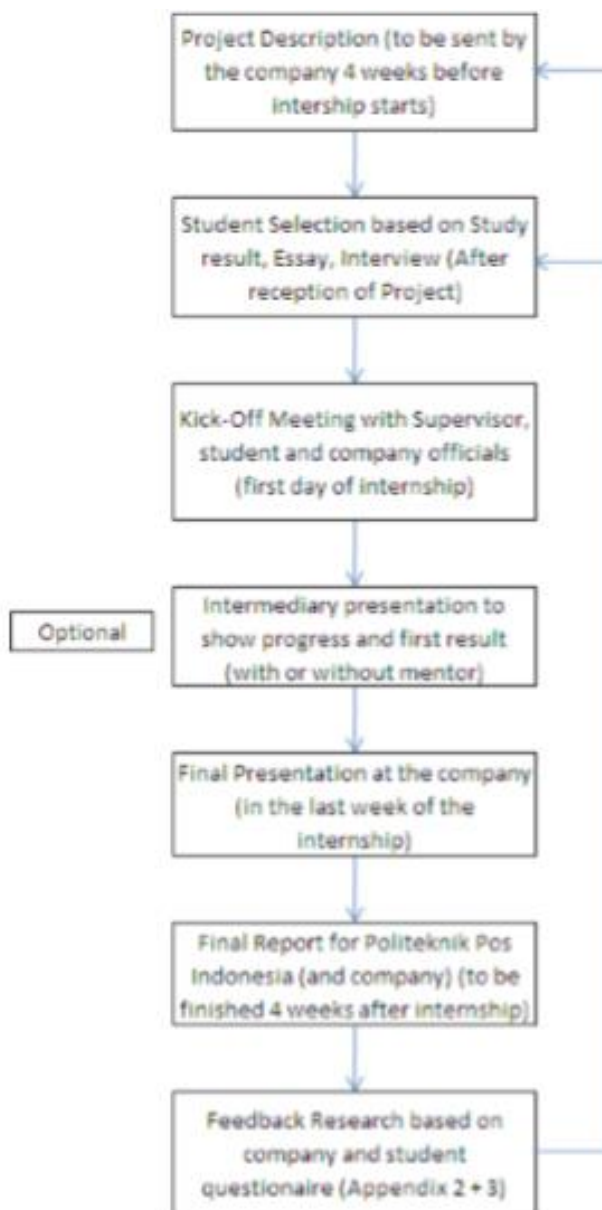
1.3 Waktu Pelaksanaan

Batas waktu yang ideal dalam *internship* adalah antara 3-4 bulan, dimana satu bulan terakhir digunakan oleh mahasiswa peserta *internship* untuk menyelesaikan laporan. Namun demikian peserta *internship* dan perusahaan dapat melakukan negosiasi tentang batas waktu tersebut dan memberitahukan kepada Politeknik Pos Indonesia.

1.4 Prosedure Program *Internship*

Secara umum prosedur *internship* dilakukan melalui beberapa tahap. Tahap pertama mahasiswa mempersiapkan aplikasi yang ditujukan ke Kepala Program Studi atau Pembantu Direktur III (Bidang Kemahasiswaan), Tahap kedua dilakukan seleksi administrasi meliputi : IPK, Konsentrasi Subjek *Internship* dan penilaian aspek individu. Tahap ketiga pengiriman aplikasi ke perusahaan-perusahaan rekanan. Tahap keempat penempatan sesuai dengan kriteria dan kebutuhan perusahaan (dilakukan melalui test, wawancara atau secara langsung). Dalam pembuatan *internship* 1 ini, isi laporan berupa analisis , perancangan dan pembuatan aplikasi/alat. Untuk itu

dalam pembuatan laporan *internship* 1 ditekankan pada permasalahan yang mahasiswa/I dapatkan di perusahaan.



Gambar 1.1 Prosedur Program Internship

1. Mentoring

Agar tercapai tujuan serta manfaat bagi kedua belah pihak, maka diperlukan adanya sistem bimbingan atau mentor yang dilakukan oleh karyawan yang ditunjuk oleh pihak manajemen perusahaan, sehingga terjadi sinergitas antara mahasiswa dan perusahaan.

1. Definisi

Mentoring adalah suatu proses hubungan yang dibangun antara mentor dengan peserta *internship* (mahasiswa) yang di fokuskan pada upaya pengembangan *soft skill* mahasiswa, baik dari segi *attitude* maupun *skill* demikian juga dengan dengan *competency* dalam rangka meningkatkan produktivitas kerja peserta pada saat ini maupun masa yang akan datang.

2. Tujuan

Mentoring bertujuan untuk memberikan bantuan bagi para mahasiswa yang *internship* agar mereka lebih cepat beradaptasi dengan lingkungan kerja, karyawan lain serta dengan tugas-tugas yang diberikan kepadanya. Sedangkan target mentoring adalah memberikan dorongan, motivasi serta transfer *of knowledge* yang dibutuhkan bagi para mahasiswa yang *internship*, agar mereka lebih peduli dan lebih cepat beradaptasi sehingga mereka dapat memberikan kontribusi yang diharapkan oleh perusahaan.

3. Sasaran Pengembangan

Sasaran yang ingin dicapai dalam mentoring ini adalah pengembangan sikap kerja para mahasiswa yang ikut *internship*, pengembangan kompetensi, baik yang bersifat *generic* maupun yang bersifat *technical*, sehingga terjadi transfer *of knowledge* yang didasarkan pada proses pembelajaran ditempat kerja (*learning by doing*).

4. Personel Mentoring

Terdiri dari dua mentor : 1). Mentor dari program studi sebanyak maksimum 2 orang dengan penempatan dari program studi atau jurusan, 2). Mentor dari perusahaan maksimum 2 orang dengan penetapan dari perusahaan yang bertalian.

5. Peran dan Tanggung Jawab

Agar program dan internship ini dapat berjalan dengan baik sesuai dengan pola kemitraan maka harus ada komitmen yang harus ditaati dan dihormati oleh para peserta internship. Adapun peran dan tanggung jawab baik mentor maupun peserta internship dapat diuraikan pada table ini.

No.	Mentor	Peserta <i>Internship</i>
1.	Memberikan tugas-tugas selama masa <i>internship</i> yang dituangkan dalam <i>Activity Plan</i> peserta. Dalam menyusun <i>Activity Plan</i> ini mentor diharapkan dapat memberikan informasi dan ide-ide untuk menarik inisiatif dan kreativitas peserta <i>internship</i> .	Melakukan tugas-tugas yang menjadi tanggung jawabnya.
2.	Memberikan bimbingan dalam bentuk arahan maupun <i>feed back</i> sesuai dengan target kompetensi yang harus dikuasai peserta <i>internship</i> .	Mempelajari hal-hal yang bersifat teknis di Unit kerjanya dalam rangka menguasai kompetensi teknik sesuai dengan standar yang diharapkan.
3.	Membantu apabila peserta mengalami kesulitan, baik dalam masalah teknis pekerjaan maupun koordinasi dengan pihak-pihak terkait. Apabila ada	Mengkomunikasikan setiap ada masalah kerja yang dihadapinya secara jelas kepada mentor.

	masalah dengan sikap peserta, mentor diharapkan mengkoordinasikan masalah tersebut dengan <i>Human Resources Dept</i> yang bertalian.	
4.	Melakukan evaluasi terhadap kinerja dan kompetensi peserta selama internship sesuai dengan kriteria evaluasi yang disediakan oleh pihak perusahaan.	Menerima <i>feed back</i> secara konstruktif.
5.	Pemberian arahan masukan tentang pembuatan pelaporan <i>internship</i> sesuai dengan <i>project paper</i> yang akan diminati peserta <i>internship</i> .	Mengumpulkan data, temuan penyimpangan dan masalah yang kemudian dipresentasikan serta usulan perbaikannya (bila ada)

1.5 Sistem Evaluasi

Program *Internship* diharapkan mampu memberikan kontribusi positif bagi kedua belah pihak. Hal yang memerlukan proses evaluasi berkesinambungan yang memungkinkan perbaikan-perbaikan yang relevan bagi program ini. Harapan lebih besar program ini dapat diposisikan sebagai bagian dari suatu strategi *recruitment* jangka panjang (*long term recruitment strategy*) bagi perusahaan yang bertalian, sehingga selama masa internship itu berjalan, maka kesempatan dimaksud dapat digunakan atau *internship* dalam situasi kerja sebenarnya. Secara umum evaluasi ini dilakukan secara bertahap sebagaimana pada gambar 1, berupa presentasi awal dari peserta setelah 2 bulan melakukan kegiatan *internship/internship*, dan presentasi akhir setelah peserta melakukan kegiatan *internship/internship* selama 3-4 bulan.

1. Tujuan Evaluasi

Evaluasi bertujuan antara lain :

- a. Untuk memberikan masukan serta sebagai dasar penilaian kemampuan dan hasil yang diperoleh selama melakukan *internship/internship*.
- b. Melakukan penilaian atas capaian kesempatan kontribusi positif keilmuan dalam kegiatan operasional perusahaan.
- c. Lebih jauh nasa rekomendasi kepada pihak manajemen perusahaan dalam pengambilan keputusan khususnya dalam recruitment pegawai baru.

2. Kriteria

Evaluasi terdiri dari 2 (dua) bagian yaitu yang menyangkut dengan :

- a. *Generic Competency*, yaitu kompetensi umum yang dimiliki oleh setiap peserta untuk tingkat/level pegawai yang dipersyaratkan oleh perusahaan yang bertalian.
- b. *Technical Competency*, yaitu pengetahuan dan *technical skill* yang dibutuhkan untuk menjalani suatu tugas, jabatan karyawan pada tingkat/level tertentu di perusahaan yang bertalian.

3. Proses Evaluasi

Agar evaluasi ini berjalan dengan baik, disamping dapat digunakan oleh manajemen Politeknik Pos Indonesia maupun perusahaan, maka evaluasi *internship* dapat dilakukan melalui 2 (dua) tahap yaitu :

1. *Progress review generic competency*, targetnya adalah untuk mengetahui perkembangan proyek peserta dan hambatan yang dihadapi dalam penyelesaiannya, serta untuk mengevaluasi sikap dan kompetensi peserta *internship*.
2. *Final review generic and technical review*, targetnya adalah evaluasi akhir yang berkaitan dengan penyelesaian proyek sebagai *final report* atau sebagai sebagai *internship* yang merupakan persyaratan bagi kelulusan mereka yang menyangkut dengan kemampuan daya serap peserta selama mereka *internship*.

1.6 Hak Dan Kewajiban

Selama masa internship peserta mempunyai hak dan kewajiban yang harus dipatuhi, antara lain :

A. Hak Peserta Internship Selama Internship :

1. Peserta dapat diberikan uang saku dan tunjangan lain yang berlaku pada perusahaan yang bertalian. (Sesuai dengan kemampuan Perusahaan yang bertalian).
2. Memperoleh keterangan yang berkaitan dengan :
 - a. Ruang lingkup program *internship*.
 - b. Profil perusahaan.
 - c. Peraturan dan tata tertib perusahaan yang berlaku bagi mahasiswa *internship*.
 - d. Perlengkapan kerja (bila ada dan diharuskan) sesuai dengan peraturan perusahaan.
3. Monitor dan bimbingan selama masa *internship* oleh seorang mentor yang ditunjuk oleh manajemen perusahaan dimana peserta *internship*.

B. Kewajiban Peserta Selama *Internship* :

1. Mentaati semua peraturan yang berlaku di perusahaan tersebut.
2. Mengerjakan tugas-tugas yang telah disampaikan oleh mentor.
3. Mengikuti *review* yang diselenggarakan oleh manajemen perusahaan penerima *internship*, berupa :
 - a. *review activity plan*, peserta mempresentasikan *activity plan* yang telah disepakati bersama mentor.
 - b. *Progress Review*, peserta mempresentasikan laporan aktivitas yang telah dijalankan kepada pihak manajemen perusahaan dan

manajemen Politeknik Pos Indonesia di tempat yang disepakati oleh perusahaan tempat dimana peserta *internship* berada.

- c. *Final report*, peserta membuat laporan tugas selama periode tertentu sesuai dengan yang diminta oleh perusahaan, misalnya disesuaikan dengan *activity plan* yang ditandatangani oleh mentor.

C. Peran dan Tanggung Jawab Peserta

Peran dan Tanggung jawab peserta anatara lain :

1. Melakukan tugas-tugas yang menjadi tanggung jawabnya.
2. Mempelajari hal-hal teknis di unit kerjanya dalam rangka memahami kompetensi teknis sesuai dengan ruang lingkup kerjanya.
3. Menerima *feed back* secara konstruktif.
4. Membuat laporan dan mempersentasikan temuan masalah yang dihadapi berikut dengan pemecahan masalahnya atau perbaikan (bila ada).

1.7 Lampiran-Lampiran

1. Panduan Penyusunan Laporan
2. Progrees Report Harian
3. Format Evaluasi Peserta Internship (diisi oleh Mentor)
4. Format Evaluasi Mentor (diisi oleh Mahasiswa)

BAB II

BAHASA PEMROGRAMAN

2.1 Codeigniter

2.1.1 Sejarah Codeigniter

Kelahiran codeignier adalah bermula dari kegalauan Rick Ellis atas banyaknya kode PHP yang harus ditulis ketika membangun salah satu CMS kesayangannya, *expression engine*. Rick Ellis berusaha mempermudah penulisan kode- kode program PHP tersebut menjadi lebih singkat. Akhirnya Rick Ellis mengambil inisiatif dengan membuat sendiri kode singkat / *shortcode* dari fungsi-fungsi yang ada di PHP.

Pembuatan *shortcode* tersebut membuatnya mampu membangun *expression engine* dengan sangat bagus, efisien dan cepat. Selain itu, performanya juga sangat bagus.

Setelah tidak berapa lama, Rick Ellis melalui situsnya ellislab(dot)com membagikan *shortcode* yang dibuat sendiri itu untuk digunakan oleh *developer* lainnya. Tujuannya yaitu membantu *developer* lain dalam menangani masalah dalam *framework* yang terkenal. Banyak *developer* yang membantu perkembangan *Codeigniter* sehingga menjadi *framework* terpopuler di tahun 2006.

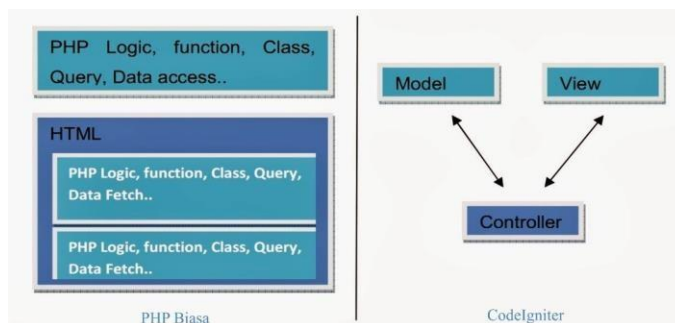


Gambar 1. Logo Codeigniter

2.1.2 Pengertian CodeIgniter

Codeigniter (CI) merupakan sebuah *web application framework* yang bersifat *open source* dimana digunakan untuk membangun aplikasi php dinamis. *Framework* itu sendiri merupakan abstraksi di dalam sebuah perangkat lunak yang menyediakan fungsi yang *generic* sehingga dapat dirubah oleh kode yang dibuat *user*, sehingga dapat menyediakan perangkat lunak untuk aplikasi tertentu.

Untuk membuat website dinamis bisa menggunakan codeigniter karena menjadi salah satu framework php dengan konsep MVC (*Model, View, Controller*) yang dapat mempermudah developer dalam pembuatan aplikasi web. Selain ringan dan cepat, codeigniter juga memiliki dokumentasi yang lengkap disertai dengan contoh implementasi kodenya. Dengan dokumentasi yang lengkap akan menjadi salah satu alasan kuat mengapa banyak orang memilih *codeigniter* sebagai *framework* pilihannya. Berikut merupakan gambar perbandingan PHP biasa dengan Codeigniter.

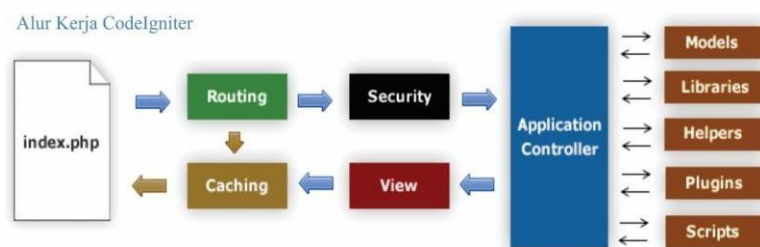


Gambar 2. Perbandingan PHP dengan Codeigniter
Konsep MVC memisahkan sistem berdasarkan komponen utama yang membangun sistem seperti manipulasi data, user interface, dan bagian pengontrol sistem. Terdapat 3 jenis

komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi, diantaranya :

- a. Model, dapat berhubungan dengan *database* seperti *insert*, *update*, *delete*. Menangani validasi dari bagian controller, namun model tidak berhubungan langsung dengan *view*.
- b. Controller, berfungsi mengatur hubungan antara model dan *view* supaya bisa saling berkomunikasi, controller tersebut yang mengatur segala proses dalam aplikasi.
- c. View, merupakan bagian yang menangani *presentation*. Pada bagian *view* biasanya berupa file dengan template HTML, yang diatur oleh *controller*. View akan menerima kemudian mempresentasikan data kepada *user*. Pada bagian ini tidak memiliki akses langsung terhadap bagian model, untuk bisa mengakses ke bagian model maka harus *request* terlebih dahulu ke *controller*.

Alur kerja *framework* Codeigniter dapat digambarkan sebagai berikut:



Gambar 3. Alur Kerja *Framework* Codeigniter

- a. *Index.php*: berfungsi sebagai file pertama dalam program yang akan dibaca oleh program.
- b. *Router*: *router* akan memeriksa *HTTP request* untuk menentukan hal apa yang harus dilakukan oleh program.
- c. *Cache File*: apabila dalam program sudah terdapat “*cache file*” maka file tersebut akan langsung dikirim ke browser. File *cache* inilah yang dapat membuat sebuah *website* dapat dibuka dengan lebih cepat. *Cache file* dapat melewati proses yang sebenarnya harus dilakukan oleh program *codeigniter*.
- d. *Security* : Sebelum *controller* di load secara keseluruhan, maka data yang disubmit oleh *user* dalam bentuk request *HTTP* akan di periksa terlebih dahulu melalui *security* yang dimiliki oleh *Codeigniter*.
- e. *Controller*: *controller* akan membuka file bagian model, *core libraries*, *helper* dan semua *resources* yang dibutuhkan dalam program tersebut.
- f. *View*: bagian terakhir akan dilakukan pengecekan semua program yang ada dalam *view* kemudian file akan mengirimkannya ke browser supaya dapat dilihat. Apabila file *view* sudah ada yang di “*cache*” maka file *view* baru yang belum ter-*cache* akan *update* file *view* yang sudah ada.

2.1.3 Fungsi Codeigniter

- a. Mempercepat dan mempermudah kita dalam pembuatan *website*.
- b. Menghasilkan struktur pemrograman yang sangat rapi, baik dari segi kode maupun struktur file phpnya.
- c. Memberikan standar *coding* sehingga memudahkan kita atau orang lain untuk mempelajari kembali sistem aplikasi yang dibangun.

2.1.4 Keunggulan dan kekurangan Codeigniter

2.1.4.1 Keunggulan Codeigniter

1. Performa yang sangat cepat dibandingkan dengan framework yang lainnya.
2. Konfigurasi yang sangat minim, untuk menyesuaikan dengan database terlebih dahulu merubah beberapa konfigurasi pada bagian file *database.php* dan *autoload.php*

2.1.4.2 Kekurangan Codeigniter

1. Codeigniter dikembangkan oleh Ellis lab dan bukan oleh suatu komunitas, yang menyebabkan *update code engine*-nya tidak secepat *framework* lain.
2. Tidak direkomendasikan untuk pembuatan *web* dengan skala besar (*enterprise*) walaupun tersedia banyak *library*.
3. Masih menemukan banyak kekurangan dalam menambahkan file, sehingga banyak file yang tidak penting dengan mudah bisa tersimpan.

2.2 Framework

2.2.1 Pengertian Framework

Framework adalah kerangka kerja. *Framework* merupakan sekumpulan *script class* dan *function* yang dapat memudahkan *developer/programmer* dalam menangani suatu permasalahan dalam pembuatan aplikasi seperti pemanggilan *variable*, koneksi ke *database*, dll. sehingga *developer* bisa lebih fokus dan lebih cepat dalam membangun sebuah aplikasi.

Framework bisa disebut komponen pemrograman yang siap digunakan kembali kapan saja, sehingga *programmer* tidak harus membuat *script* yang sama secara berulang-ulang.

Secara sederhana dapat dijelaskan bahwa *framework* merupakan sekumpulan fungsi (*libraries*), maka seorang *programmer* tidak perlu lagi membuat fungsi-fungsi (biasanya disebut kumpulan *Library*) dari awal, *programmer* tinggal memanggil kumpulan *library* atau fungsi yang sudah ada didalam *framework*, tentunya cara menggunakan fungsi-fungsi itu sudah ditentukan oleh *framework*. Berikut merupakan beberapa contoh fungsi standar framework diantaranya fungsi.

grafik, tabel bergaya zebra, validasi, *upload*, *captcha*, proteksi terhadap XSS (XSS filtering), *template*, kompresi, XML, *paging*, enkripsi, email, SEO, *session*, *security*, kalender, bahasa, manipulasi gambar, dan lain-lain.

2.2.2 Fungsi Framework

1. Dapat membantu kerja *developer* dalam membangun aplikasi sehingga aplikasi bisa diselesaikan dalam waktu yang singkat.
2. Penerapan *design patterns* memudahkan dalam rancangan, pengembangan dan pemeliharaan sistem.

3. *Stability* dan *Reability* yang dibangun didalam sistem lebih stabil dan handal karena berbasis pada *framework* yang sudah teruji stabilitas dan keandalannya.
4. *Coding Style* konsisten, memudahkan dalam membaca kode dalam menemukan *bugs*.
5. *Security Concern* dan *framework* akan mengantisipasi dengan memasang perisai terhadap adanya berbagai masalah keamanan yang mungkin terjadi.
6. Dokumentasi, *framework* dapat mendisiplinkan kita untuk menulis dokumen apa yang kita tulis.

2.2.3 Kelebihan dan Kelemahan *Framework*

2.2.3.1 Kelebihan *Framework*

1. Lebih cepat dan efisien

Jika mengerjakan proyek besar, maka penggunaan *framework* dapat membantu mempercepat proses pengembangan. Pada umumnya, *framework* memiliki beragam fungsi dan plugin yang bisa dimanfaatkan. Dalam kerangka kerja ini, maka proses pengembangan proyek jauh lebih cepat daripada harus menulis kode dari awal. Selain itu, tidak perlu menulis berulang- ulang untuk kode yang bersifat repetitif.

2. Menghemat biaya

Sebagian besar bersifat open source dan gratis untuk bisa digunakan. Biaya yang harus dikeluarkan oleh client juga akan menjadi lebih kecil karena proses pengerjaan yang lebih simple dan lebih cepat.

3. Memperhatikan faktor keamanan

Framework telah banyak digunakan oleh *developer*, dan kemungkinan adanya masalah keamanan maupun bug yang telah diperbaiki. Selain itu, *framework* biasanya memiliki komunitas dalam jumlah cukup banyak yang dapat berperan dalam jangka panjang. Setiap kali pengguna menemukan celah keamanan, maka mereka dapat memberi tahu tim untuk segera memperbaikinya.

2.2.3.2 Kelemahan *Framework*

1. Kurangnya pemahaman bahasa pemrograman

Jika bekerja menggunakan kerangka kerja kemudian hanya mengetahui sedikit tentang bahasa pemrograman yang digunakan, maka *developer* hanya mempelajari mengenai kerangka tersebut. Sehingga pemahaman mengenai bahasa pemrograman menjadi tidak berkembang.

2. Memiliki batasan

Dalam penggunaannya, kerangka ini juga memiliki beberapa batasan yang tidak dapat Anda modifikasi. Sehingga Anda harus bekerja sesuai dengan standar yang digunakan di dalamnya. Oleh karena itu, ketika mengembangkan sebuah aplikasi, maka harus menggunakan kerangka kerja yang sesuai dengan kebutuhan *developer*.

3. Kode Publik

Karena bersifat publik, maka kode dapat digunakan siapa saja termasuk pihak-pihak yang mempunyai niat buruk.

2.3 *Object Oriented Programming* (OOP)

2.3.1 Sejarah OOP

Konsep *Object Oriented Programming* (OOP) pertama kali muncul di MIT (*Massachusetts Institute of Technology*) pada era 1960-an. Sekitar beberapa tahun kemudian antara 1962-1965, sebuah bahasa pemrograman yang mendasari konsep OOP diperkenalkan dengan nama bahasa pemrograman SIMULA 1, dikembangkan oleh Kristen Nygaard dan Ole-Johan yang merupakan warga negara Norwegia. Setelah itu pada tahun 1967 keluarlah SIMULA 67.

Namun pada tahun 1980-an bahasa pemrograman C++ mematahkan kepercayaan tersebut. Bahasa pemrograman C++ menjadi bahasa pemrograman yang populer dan mendominasi hingga sekarang. Bahasa pemrograman C++ yang merupakan gabungan dari 2 konsep bahasa pemrograman, yakni C dan SIMULA.

Semenjak C++ terkenal, banyak sekali pengembang yang terinspirasi oleh C++ dan pada tahun 1990-an, bahasa pemrograman Java diperkenalkan yang mengaku terinspirasi oleh C++, dan tahun 2002 perusahaan Microsoft juga mengeluarkan bahasa bawaan dari C++ yaitu C# (C-Sharp), kemudian disusul dengan VB.Net dengan fitur OOP yang merupakan penyempurnaan dari bahasa VB 0.6 yang tidak mendukung fitur OOP.

2.3.2 Pengertian OOP

Object Oriented Programming (OOP) merupakan paradigma dalam melakukan pemrograman yang berorientasi objek, sehingga pengolahan data disatukan dalam *class* dan *object*. Masing-masing objek dapat memiliki sifat dan tugasnya. Pada paradigma ini, objek-objek tersebut dapat bekerja sendiri dan juga dapat saling bekerja sama dengan kemungkinan untuk

saling berhubungan, seperti menerima, mengirim data kepada objek lainnya dan memproses data. Paradigma OOP dapat dilihat sebagai interaksi dari objek yang saling berhubungan satu sama lain untuk melakukan tugasnya.

OOP bertujuan untuk memberikan pola pikir dalam mengembangkan sebuah program, pola pikir tersebut dipercaya dapat memberikan kemudahan dalam pembuatan program, pengembangan program, perawatan program, dan fleksibilitas.

2.3.3 Jenis-jenis OOP pada bahasa pemrograman

Konsep OOP dibagi menjadi 3 jenis, yaitu :

- a. Bahasa OOP murni, merupakan sebuah bahasa yang mengharuskan program ditulis hanya berupa objek saja. Contoh – Eifel, Smaltalk, Ruby, Jade, dan lain- lain.
- b. Bahasa OOP *hybrid*, merupakan bahasa yang dirancang untuk pemrograman objek dengan beberapa elemen prosedural.
- c. Bahasa OOP *hybrid* dalam *web*, salah seperti bahasa OOP *hybrid*, yang membedakan hanya konsep yang digunakan dalam pemrograman *web*.

2.3.4 Konsep OOP

- a. *Class*, adalah sebuah rancangan untuk mendefinisikan karakter dan perilaku dari objek, yang merupakan kumpulan atas definisi dan fungsi-fungsi dalam suatu unit, untuk suatu tujuan tertentu.
- b. *Object*, adalah dasar dari modularitas dan struktur pada OOP, dan merupakan representasi dari *class*, objek akan memiliki sifat dan perilaku dari *class* yang digunakan.
- c. *Encapsulation*, adalah konsep dalam implementasi untuk membungkus data dan fungsi menjadi satu entitas, dan membatasi akses dari luar *class*.
- d. *Inheritance*, adalah konsep pewarisan *class*. *Class* juga dapat menurun dan memiliki apa yang dimiliki oleh *class* lainnya.
- e. *Abstraction*, adalah konsep untuk mendesain sebuah objek, teknik dalam menyembunyikan detail suatu proses dalam objek tersebut, dengan tujuan untuk memfokuskan pengguna pada fungsi inti objek.
- f. *Polymorphism*, adalah kemampuan dari *class* yang berbeda untuk menyediakan penggunaan yang berbedabagi interface (publik) yang sama.

2.3.5 Keunggulan dan kekurangan OOP

1.3.5.1 Keunggulan OOP

1. OOP menyediakan struktur yang sangat jelas untuk program, sehingga OOP sangat bagus digunakan untuk mendefinisikan tipe data.
2. OOP akan mempermudah dalam melakukan *maintenance* dan memodifikasi kode yang sudah ada. Objek yang baru dapat dibuat tanpa harus mengubah kode yang sudah ada.

3. OOP menyediakan sebuah *framework* dimana komponen *software* yang tersedia dalam *framework tersebut* dapat dengan mudah diadaptasi dan dimodifikasi oleh programmer. Hal ini sering digunakan dalam pengembangan *Graphical User Interfaces* (GUI).

1.3.5.2 Kekurangan OOP

1. Tidak memperbolehkan implementasi yang kuat pada *reuse*.
2. *Property software* tidak terikat dalam satu unit fungsional sehingga harus *crosscut* di antara komponennya.
3. *Crosscut* tersebut mengakibatkan sulitnya pengembangan dan pemeliharaan.

2.4 *Hypertext Preprocessor* (PHP)

2.4.1 Sejarah PHP

PHP pertama kali dibuat oleh Rasmus Lerdorf, seorang programmer C. Pada saat itu PHP bernama *Form Interpreted* (FI), yang tampilannya berupa sekumpulan *script* yang digunakan untuk mengolah data dalam berbentuk *form* dari *web*. Jadi, awal mula PHP digunakannya untuk menghitung jumlah pengunjung didalam *web*-nya.

Dengan alasan untuk meningkatkan performa, Rasmus Lerdorf membuat ulang kode program tersebut dalam bahasa C. Lerdorf menyebut kode program ini sebagai *Personal Home Page*. Versi ini pertama kali keluar pada tahun 1995. Isinya adalah sekumpulan skrip PERL yang dibuatnya untuk membuat halaman *web*-nya menjadi dinamis. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI, kedependekan dari *Hypertext Prerocessing / Form Interpreter*.

Dalam perilsan kode sumber ini menjadi *open source*, maka dari itu banyak *programmer* yang tertarik untuk mengembangkan PHP. Kemudian pada tahun 1996 ia mengeluarkan PHP versi 2.0 yang kemampuannya telah dapat mengakses *database* dan dapat terintegrasi dengan *Hypertext Markup Language* (HTML). Selanjutnya *interpreter* PHP sudah diimplementasikan dalam program C. Dalam perilsan ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP / FI secara signifikan. PHP versi 2.0 ini telah menarik banyak perahitian *programmer*, namun bahasa ini memiliki masalah dengan kestabilan yang kurang bisa diandalkan. Hal ini dikarenakan Lerdorf hanya bekerja sendiri untuk mengembangkan PHP.

Pada saat itu Andi Gutmans dan Zeev Suraski, mengambil bagian dan membuat ulang *parsing engine* yang menjadi dasar dari PHP supaya lebih stabil. Dengan dukungan dari banyak *programmer* lainnya, proyek PHP secara perlahan beralih dari proyek satu orang menjadi proyek masal yang lebih akrab kita kenal sebagai *open-source project*. Selanjutnya The PHP Group yang merupakan kumpulan programmer dari seluruh dunia bersatu untuk mengembangkan PHP.

Pada tahun 1998 tepatnya pada tanggal 6 Juni 1998 telah rilis PHP versi 3.0 yang dirilis oleh Rasmus sendiri bersama kelompok pengembang *software*-nya.

PHP 4.0 adalah versi PHP yang paling banyak digunakan pada awal abad ke-21. Versi ini banyak diadopsi dikarenakan kemampuannya untuk membangun aplikasi *web* sangat kompleks, akan tetapi tetap memiliki kecepatan dan stabilitas yang tinggi. Pada Juni 2004, *Zend* kembali merilis PHP 5.0. Dalam versi 5.0 ini, mengalami perubahan yang cukup besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk mengetahui perkembangannya. Beberapa penambahan fitur meliputi *PHP Data Objects* (PDO) untuk mengakses *database*, *closures*, *trait*, dan *namespaces*.

Namun dikarenakan beberapa alasan seperti kurangnya *programmer*, dan performa yang tidak memuaskan, pengembangan PHP 6 dihentikan dan fitur yang ada dimasukkan ke dalam PHP 5.

Pada tahun 2014, sebuah proyek lanjutan PHP mulai mengemuka, yakni PHP 7 yang berkembang dari banyak eksperimen yang dinamakan *PHP Next Generation* (PHPNG), yang dikembangkan Dmitry Stogov, Xinchun Hui, dan Nikita Popov.

2.4.2 Pengertian PHP

PHP adalah bahasa pemrograman *script* sisi server yang didesain untuk pengembangan *web*. Dimana PHP ini merupakan singkatan dari *Hypertext Preprocessor* yang digunakan sebagai bahasa pemrograman umum. Selain itu PHP juga digunakan secara bersamaan dengan bahasa pemrograman lainnya seperti bahasa pemrograman HTML, dan Javascript.

Bahasa pemrograman PHP sering disebut sebagai bahasa pemrograman *server-side*, karena berbeda dengan bahasa pemrograman *client-side* seperti Javascript yang diproses di *web browser (client)*. PHP juga menjadi dasar dari aplikasi *Content Management System* (CMS) yang populer seperti Joomla, Drupal, dan Wordpress.

2.4.3 Fungsi PHP

Salah satu fungsi dari PHP ini dapat disisipkan pada dokumen HTML. Karena kemampuan inilah PHP juga sering disebut sebagai bahasa pemrograman *script* atau *scripting language*. Berikut ini merupakan sintaksis dasar pada PHP.

a. Pembatas

PHP hanya mengeksekusi kode yang tertulis dalam pembatas yang telah ditentukan oleh sintaks PHP. Apapun di luar pembatas tidak diproses oleh PHP. Pembatas paling umum adalah “<?php” untuk membuka dan “?” untuk menutup kode PHP. Tujuan dari pembatas ini yaitu untuk memisahkan kode PHP dari kode PHP lainnya, seperti HTML, dan Javascript.

b. Variabel

Variabel dalam PHP diawali dengan simbol dolar “\$”. Pada PHP versi 5.0 diperkenalkan jenis isyarat yang memungkinkan fungsi untuk menjadi parameter objek dari *class* tertentu, *array*, atau fungsi. Namun, jenis petunjuk tidak dapat digunakan dengan jenis skalar seperti angka atau *string*. Contoh variabel dapat ditulis sebagai *\$nama_variabel*.

c. Komentar

PHP memiliki 3 jenis sintaks sebagai komentar pada kode yaitu blok “/* */”, komentar 2 baris “//”, serta tanda “#” digunakan untuk komentar 1 baris. Komentar bertujuan untuk meninggalkan catatan pada kode PHP dan tidak akan diterjemahkan ke program.

2.4.4 Keunggulan dan kekurangan PHP

2.4.4.1 Keunggulan PHP

1. Bahasa pemrograman PHP merupakan *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. *Web server* yang mendukung PHP dapat ditemukan di mana-mana dari mulai *apache*, *IIS*, *Lightpad*, hingga *xitami* dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya *developer* yang siap membantu dalam pengembangan.
4. Dari sisi pemahamannya, PHP merupakan bahasa *scripting* yang sangat mudah digunakan karena memiliki referensi yang cukup banyak.
5. PHP merupakan bahasa pemrograman yang bersifat *open source* (Gratis) yang bisa digunakan di berbagai sistem operasi seperti Windows, Linux, Unix, Macintosh dan dapat dijalankan secara *runtime* melalui *console* serta dapat menjalankan perintah-perintah sistem.

2.4.4.2 Kekurangan PHP

1. Tidak ideal jika untuk pengembangan skala besar.
2. Tidak memiliki sistem pemrograman berorientasi objek yang sesungguhnya.
3. Tidak dapat memisahkan antara tampilan dengan logik dengan baik.
4. PHP memiliki kelemahan dalam hal keamanan tertentu, dimana *programmer* sering tidak teliti dalam melakukan pemrograman dan kurang memperhatikan isu dan konfigurasi PHP.

2.5 Google Maps AP

2.5.1 Pengertian Google Maps API

Google Maps API merupakan pengembangan teknologi dari google yang digunakan untuk menanamkan Google Map di suatu aplikasi yang tidak dibuat oleh Google. Google Maps API merupakan suatu library dengan bentuk *javascript* yang berguna untuk memodifikasi peta yang ada di Google Maps sesuai dengan kebutuhan. Dalam perkembangannya Google Maps API diberikan kemampuan untuk mengambil gambar peta statis. Melakukan geocoding, dan memberikan penuntun arah. Google Maps API bersifat gratis untuk publik.

Penggunaan Google Maps API pada pengembangan aplikasi android dengan menggunakan Eclipse dan komputer menggunakan sistem operasi windows. Google Maps API terbagi menjadi 4 platform yaitu *web*, *Webservice*, *Android*, dan *iOS*.

2.5.2 Platform Google Maps API

1. Google Maps API for Android

- Google Maps Android API

<https://developers.google.com/maps/documentation/android-api/>

API ini digunakan untuk aplikasi yang menampilkan peta seperti GIS di Android. Fitur yang disediakan yaitu 3D Building, Custom Map, Custom Marker, Integrasi dengan webservice pihak ketiga dan yang lainnya. Intinya jika dalam pembuatan aplikasi GIS seperti peta masjid atau sebaran penduduk di Suatu daerah, maka inilah APInya. Bahasa yang dipakai adalah JAVA.

- Google Place API for Android

<https://developers.google.com/places/android-api/>

Digunakan untuk aplikasi yang membutuhkan fitur mendeteksi lokasi disekitar user di Android dengan bahas JAVA. Misalkan daftar restoran terdekat atau pom bensin terdekat.

2. Google Maps API for iOS

API disini digunakan untuk membuat Aplikasi iPad atau iPhone

- Google Maps SDK for iOS

<https://developers.google.com/maps/documentation/ios-sdk/>

Sama dengan Google Maps Android API yang membedakan yaitu API ini untuk Platform Apple. Bahasa yang digunakan adalah Objective-C.

- Google Place API for iOS

<https://developers.google.com/places/ios-api/> Sama dengan Place API for Android yang membedakan API ini untuk Platform Apple. Bahasa yang digunakan adalah Objective-C.

3. Google Maps API for Web

Kumpulan API ini digunakan untuk membuat aplikasi WEB seperti GIS dengan bahasa pemrograman *Javascript* dan HTML. API untuk web ini dapat dilihat dokumentasinya di

<https://developers.google.com/maps/web/>.

API untuk web ini terbagi menjadi :

- Google Map Javascript API, digunakan untuk menampilkan Peta di webApp atau website yang dibuat dengan custom UI, Marker, Infowindows dan integrasi.
 - Google Maps Embeded API, digunakan untuk menampilkan peta suatu lokasi tanpa menggunakan bahasa pemrograman, peta ini hampir sama dengan menampilkan peta Google di website yang dibuat secara interaktif.
 - Google Street View Image API biasa digunakan untuk menampilkan lokasi jalan dari StreetView.
API ini sifatnya interaktif, sangat cocok untuk menampilkan lokasi dengan sudut pandang 360 derajat.
 - Google Static Maps API ini biasa digunakan untuk menampilkan Peta Static sebuah lokasi. API ini hampir mirip dengan Embeded API, hanya yang membedakan peta nya tidak interaktif. (sudah dalam format Gambar).
 - Google Place Javascript API, digunakan untuk melisting lokasi point of interest seperti hotel, restoran atau lokasi lokasi lain yang terdata di Google map, API ini fungsinya hampir sama dengan Google Place API for Android dan Google Place API for IOS.
4. Google Maps Webservice API
API ini merupakan layanan yang outputnya berupa JSON. Artinya sembarang bahasa pemrograman atau platform bisa mengaksesnya. Webservice lebih fokus dalam memberikan data dengan format outputnya diserahkan ke programmer. Jika akan membuat aplikasi Canggih berbasis Google Maps maka wajib di pelajari di

<https://developers.google.com/maps/web-services/>.

Berikut ini adalah komponennya :

- Google Maps GeoCoding API ini berfungsi untuk mengkonversi koordinat menjadi alamat jalan atau lokasi menjadi koordinat (reverse GeoCoding).
- Google Place Webservice API ini memiliki fungsi yang sama dengan Google Place API, yang membedakan yaitu webservice ini lebih luas pemakaiannya dan tidak dibatasi oleh bahasa pemrograman java, Objective-C atau javascript
- Google Maps Elevation API ini memiliki fungsi menampilkan ketinggian suatu lokasi dari atas permukaan laut. Misalkan kota Yogyakarta ketinggian dari permukaan laut 10 Meter, sedangkan jika lokasi koordinatnya adalah puncak gunung merapi, ketinggiannya 3000 Meter.
- Google Maps Road API ini memiliki fungsi sangat spesifik, yaitu memastikan Programer GPS tracker menampilkan log perjalanan persis di jalan yang dilalui. Biasanya dipakai perusahaan GPS tracking atau asset Tracking.
- Google Map GeoLocation API ini memiliki fungsi yang sangat canggih, dikarenakan bisa mendeteksi lokasi user walaupun GPS di smartphone dimatikan. GeoLocation API memanfaatkan sinyal wifi untuk mendeteksi/melacak keberadaan lokasi user. Artinya perangkat yang tidak memiliki GPS masih bisa dideteksi lokasinya walaupun tidak seakurat sensor GPS.

- Google Maps Direction API. Dipakai untuk menunjukkan jalur perjalanan dari lokasi A ke lokasi B. Biasanya lokasi A adalah lokasi User saat ini dan lokasi adalah Point of interest yang dituju.
- Google Maps Timezone API digunakan untuk mendeteksi Timezone suatu koordinat/lokasi. Misalkan input Jakarta, maka timezone GMT+7.
- Google Maps Distance Matrix API ini sangat cocok untuk mengkalkulasi jarak dan waktu tempuh ke suatu lokasi. API ini masih berhubungan dengan Direction API.

2.5.3 Kelebihan dan kekurangan Google Maps API

2.5.3.1 Kelebihan Google Maps API

1. Dukungan penuh yang dilakukan google sehingga terjamin dan bervariasi fitur yang ada pada Google Maps API
2. Banyak pengembang yang menggunakan Google Maps API sehingga mudah dalam mencari referensi dalam pengembangan aplikasi.

2.5.3.2 Kekurangan Google Maps API

1. Jika ingin mengakses maka harus terkoneksi ke internet pada perangkat yang digunakan.

2.6 XAMPP

2.6.1 Sejarah XAMPP

XAMPP adalah pengembangan dari LAMP (*Linux Apache, MySQL, PHP and PERL*). XAMPP didirikan oleh Kai 'Oswald' Seidler dan Kay Vogelgesang pada tahun 2002. Xampp tersebut merupakan salah satu project non-profit, project ini bertujuan untuk mempromosikan Apache web server kepada publik.

2.6.2 Pengertian XAMPP

XAMPP yaitu perangkat lunak yang mendukung banyak system operasi, merupakan kompilasi dari beberapa program. XAMPP merupakan tools yang menyediakan paket perangkat lunak. Dengan menginstall XAMPP maka tidak perlu melakukan kembali instalasi, konfigurasi web server Apache seperti PHP dan MySQL secara manual. XAMPP akan menginstalasi dan mengkonfigurasikannya secara otomatis.

XAMPP merupakan salah satu paket instalasi Apache, PHP dan MySQL yang dapat digunakan untuk membantu proses instalasi ketiga aplikasi tersebut. Selain paket instalasi instant XAMPP versi 1.6.4 juga memberikan fasilitas pilihan penggunaan PHP v4.0 atau PHP v5.0.

2.6.3 Penjelasan singkatan XAMPP

- **X** : Program ini bisa dijalankan di banyak sistem operasi seperti Windows, Linux, Mac OS, dan Solaris.
- **A** : Apache, server aplikasi Web.
- **M** : MySQL, merupakan aplikasi database yang berfungsi untuk mengolah/menyimpan data.
- **P** : PHP, merupakan bahasa pemrograman web.
- **P** : Perl, merupakan bahasa pemrograman untuk semua tujuan.

2.6.4 Fitur-fitur XAMPP

Berikut ini merupakan beberapa fitur-fitur dari xampp :

1. *Apache*

Apache dapat diartikan sebagai perangkat lunak sumber terbuka yang menjadi alternatif dari server web Netscape.

2. *MySQL*

MySQL atau sering disebut “My Structured Query Language”. MySQL ini berjalan sebagai server yang menyediakan multi-user untuk mengakses ke sejumlah database.

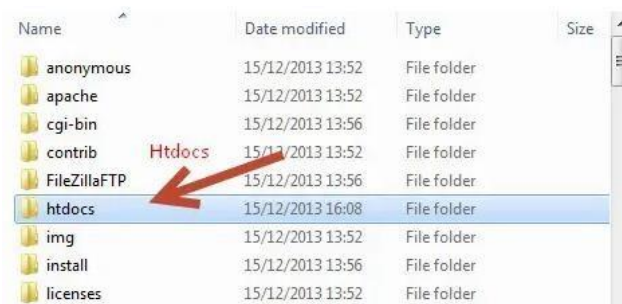
3. *PHP*

Basaha pemrograman *script* sisi server yang didesain untuk pengembangan *web*.

2.6.5 Bagian-bagian Xampp

Berikut ini adalah merupakan bagian-bagian dari XAMPP :

1. Htdocs

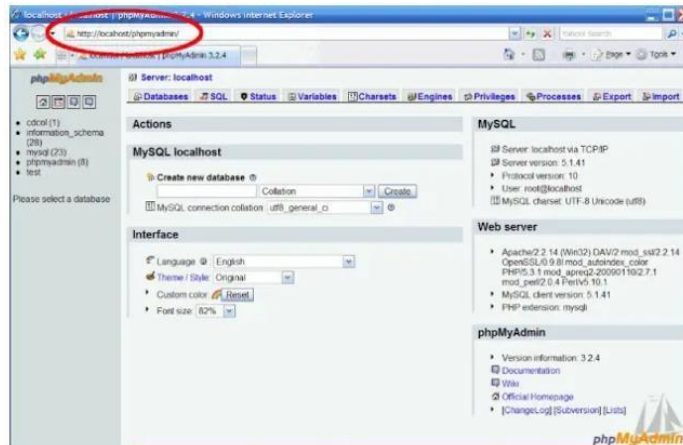


Name	Date modified	Type	Size
anonymous	15/12/2013 13:52	File folder	
apache	15/12/2013 13:52	File folder	
cgi-bin	15/12/2013 13:56	File folder	
contrib	15/12/2013 13:52	File folder	
FileZillaFTP	15/12/2013 13:56	File folder	
htdocs	15/12/2013 16:08	File folder	
img	15/12/2013 13:52	File folder	
install	15/12/2013 13:56	File folder	
licenses	15/12/2013 13:52	File folder	

Gambar x. Htdocs

Htdocs merupakan sebuah folder yang digunakan untuk menyimpan aplikasi yang akan dibuat.

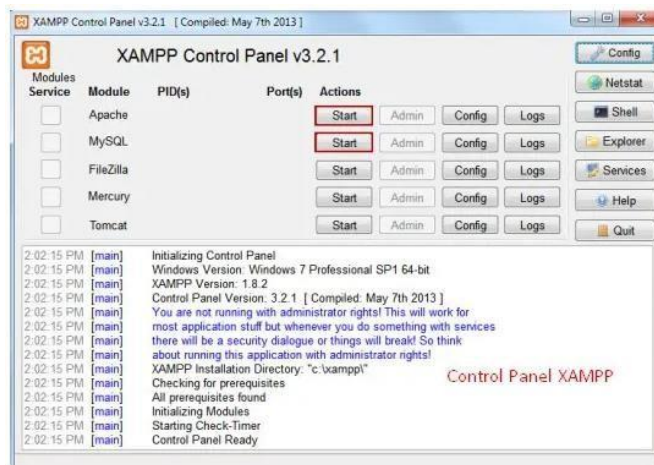
2. PhpMyadmin



Gambar x. phpMyadmin

phpMyadmin merupakan sebuah tempat yang digunakan untuk mengelola database MySQL yang telah di install pada komputer atau laptop. Untuk mengakses phpMyadmin dengan mengetikan <http://localhost/phpMyadmin> pada browser maka akan muncul tampilannya seperti gambar x.

3. Control Panel



Gambar x. Control Panel

Control Panel merupakan sebuah layanan untuk mengelola XAMPP baik itu mengontrol (start atau stop XAMPP) serta layanan service lainnya.

2.6.6 Kelebihan dan Kekurangan XAMPP

Berikut ini terdapat beberapa kelebihan dan kekurangan pada XAMPP, yaitu :

2.6.7.1 Kelebihan Xampp

1. Database Storage Engine ini banyak digunakan oleh programmer karena sifatnya free (gratis).
2. Kemampuannya mempunyai kapasitas yang cukup mumpuni yaitu sekitar 60.000 tabel dengan jumlah record mencapai 5.000.000.000 bahkan untuk yang terbaru sudah lebih dari itu.
3. Keamanan penyimpanan data pada xampp sudah terbilang cukup aman.
4. Kelebihan paling utama dari xampp ini yaitu kecepatannya.

2.6.7.2 Kekurangan Xampp

1. Tidak cocok untuk menangani data dalam jumlah yang besar, baik untuk menyimpan data maupun untuk memproses data.
2. Memiliki keterbatasan dalam kemampuan kinerja pada server ketika data yang disimpan telah melebihi batas maksimal kemampuan.

2.7 Instalasi Aplikasi Yang Akan Digunakan

Alat yang dibutuhkan untuk membangun sistem pemetaan ini diantaranya :

- *Xampp*
- *Sublime text*
- *Codeigniter*

1. Xampp

Xampp merupakan sebuah paket perangkat lunak (*software*) komputer yang sistem penamaannya diambil dari akronim kata Apache, MySQL / MariaDB, PHP, dan PERL. Sementara huruf “X” yang terdapat pada awal kata berasal dari istilah *cross platform* sebagai simbol bahwa aplikasi ini bisa dijalankan di empat sistem operasi yang berbeda.

Jika dijabarkan secara gamblang, masing-masing huruf yang adad di dalam nama XAMPP memiliki arti sebagai berikut.

X = *Cross Platform*, merupakan kode penanda untuk *software cross platform* atau yang bisa berjalan di banyak sistem operasi.

A = *Apache*, apache adalah aplikasi *web server* yang bersifat gratis dan bisa dikembangkan oleh banyak orang (*open source*).

M = *MySQL / MariaDB*, MySQL atau MariaDB merupakan aplikasi *database server* yang dikembangkan oleh orang yang sama. MySQL berperan dalam mengolah, mengedit, dan menghapus daftar melalui *database*.

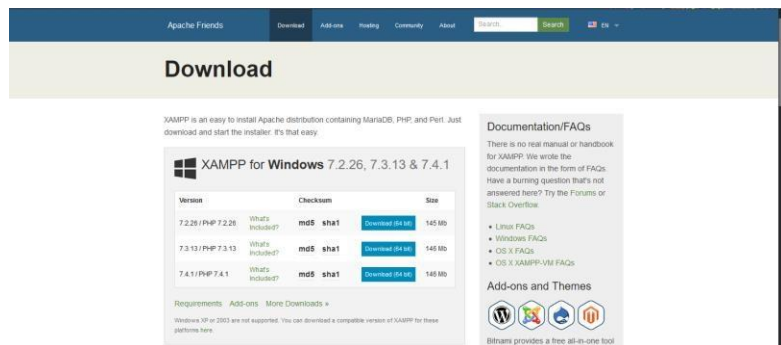
P = *PHP*, huruf “P” yang pertama dari akronim kata XAMPP adalah inisial untuk menunjukkan eksistensi bahasa

pemrograman PHP. Bahasa pemrograman ini biasanya digunakan untuk membuat *website dinamis*.

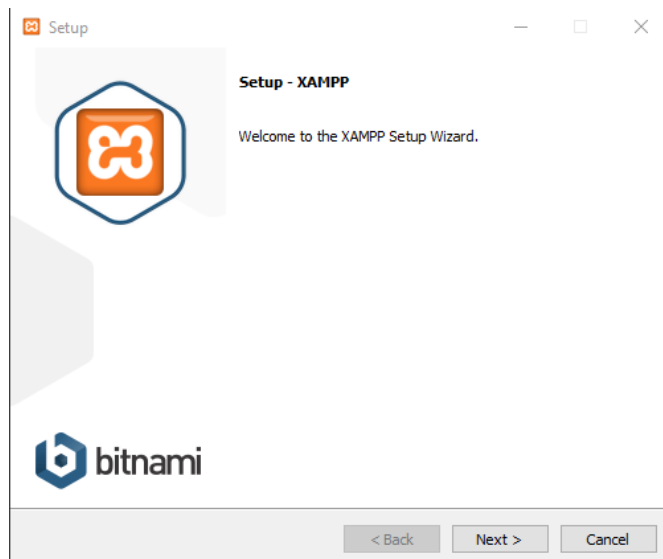
P = Perl, untuk huruf “P” selanjutnya merupakan singkatan dari bahasa pemrograman Perl yang kerap digunakan untuk memenuhi berbagai macam kebutuhan. Perl ini bisa berjalan di dalam banyak sistem operasi, sehingga sangat fleksibel dan banyak digunakan.

Berikut merupakan langkah-langkah untuk melakukan instalasi xampp pada sistem operasi windows.

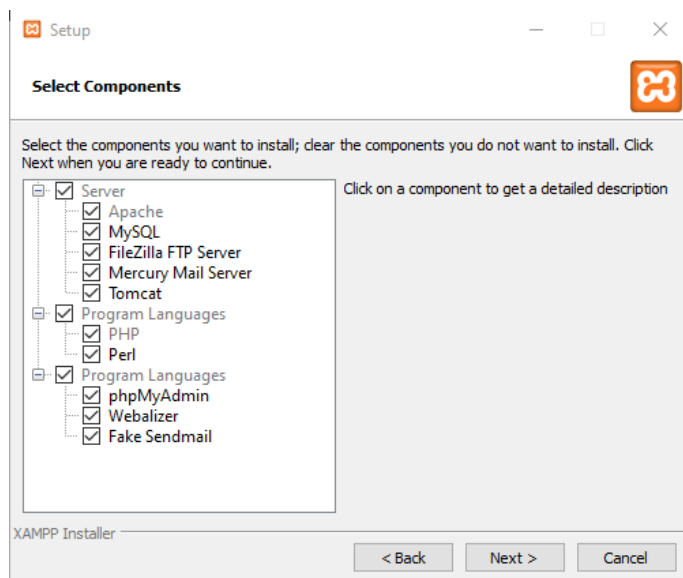
- a. Pertama, *download* terlebih dahulu file installer xampp pada link berikut ini. [https://apachefriends\(dot\)org](https://apachefriends(dot)org)



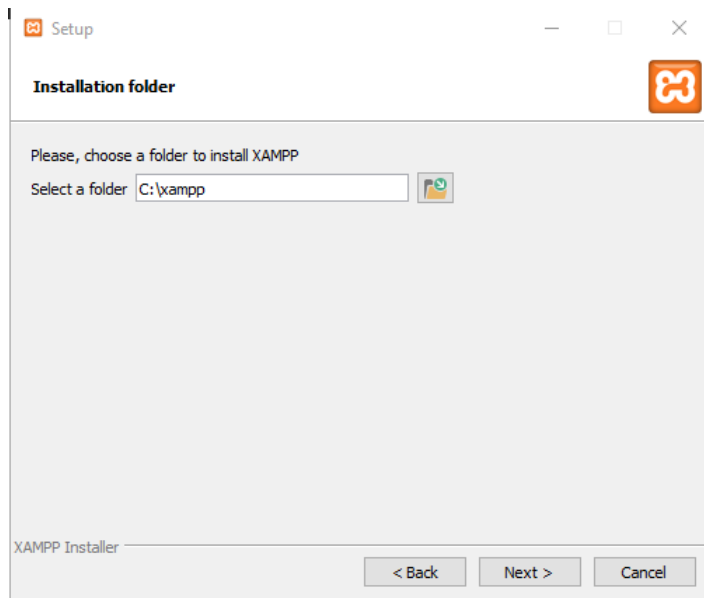
- b. Pilih file installer sesuai dengan spesifikasi laptop yang digunakan.
- c. Klik file xampp yang sudah berhasil di *download*, maka akan muncul tampilan seperti ini.



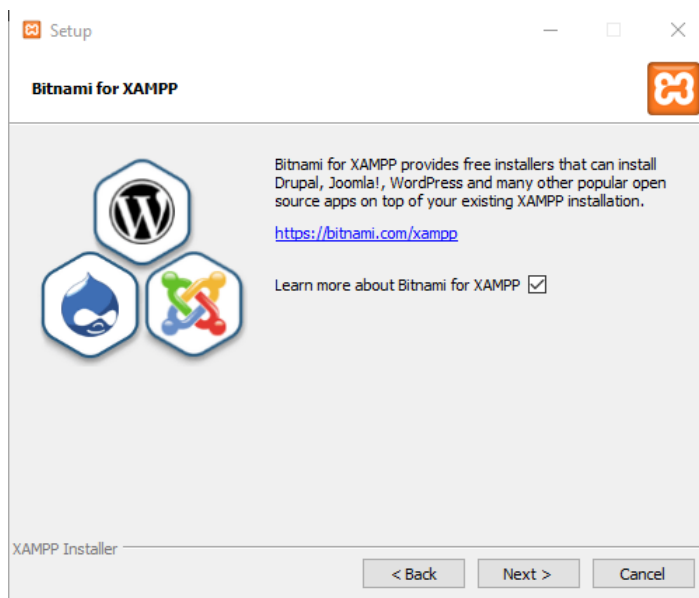
- d. Kemudian, pilih next untuk melanjutkan pada proses instalasi.



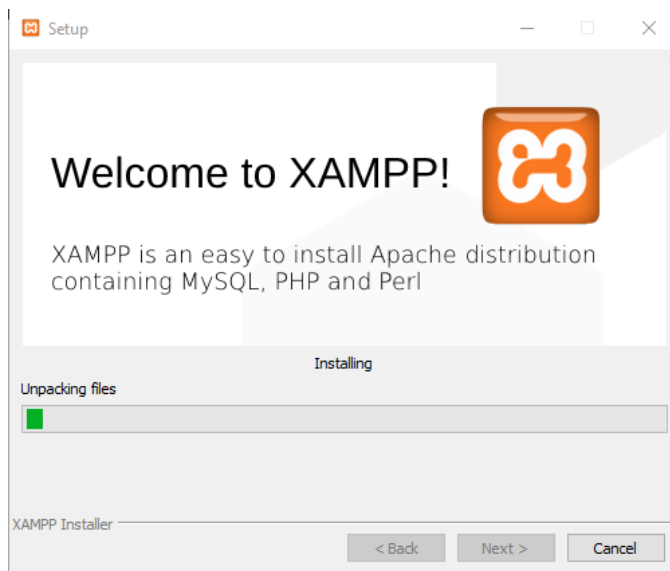
- e. Pada tampilan *select component*, centang semua pilihannya agar dapat menjalankan semua program yang dipilih. Kemudian pilih *next*.



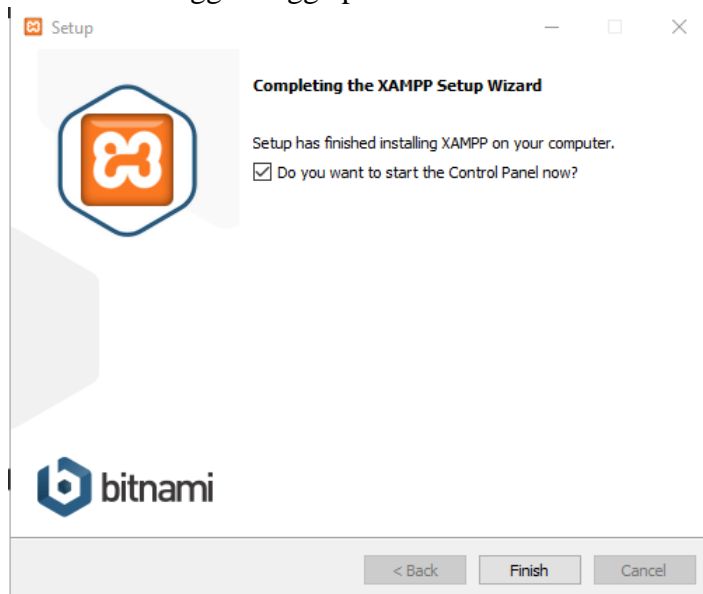
- f. Pada, *installation folder* secara *default* sistem akan menyimpannya pada direktori C. Jika sudah klik *next* untuk melanjutkan proses instalasi.



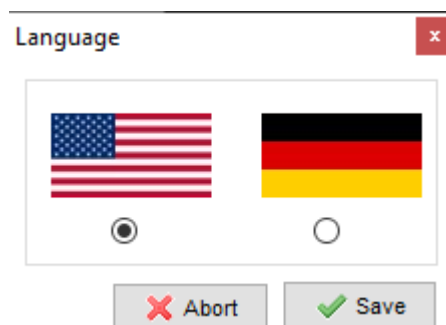
- g. Pada tampilan ini langsung saja klik *next* untuk melanjutkan.



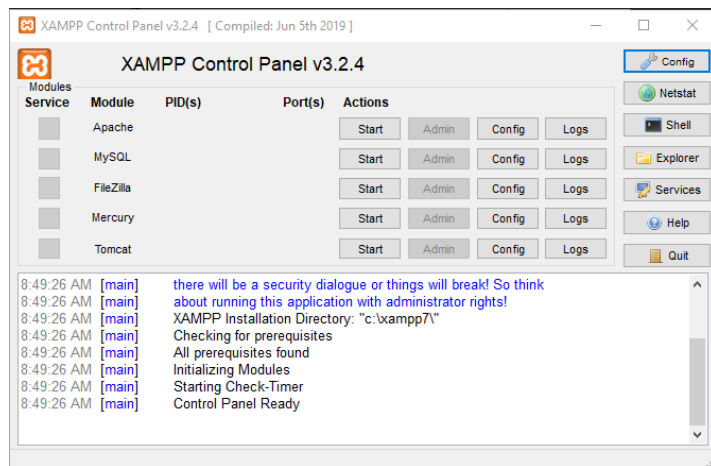
- h. Kemudian tunggu hingga proses instalasi selesai.



- i. Jika telah selesai, selanjutnya klik *Finish* untuk membuka *control panel*.



- j. Pilih bahasa yang akan digunakan pada xampp. Lalu klik *save*.



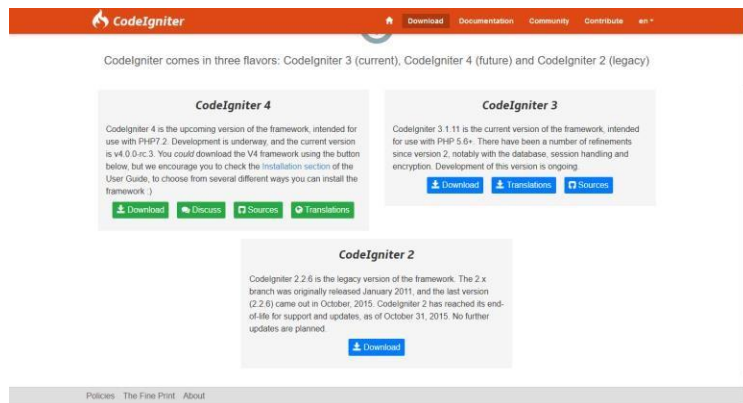
k. Jika berhasil, maka tampilan utama *control panel* dari xampp seperti berikut.

2. Codeigniter

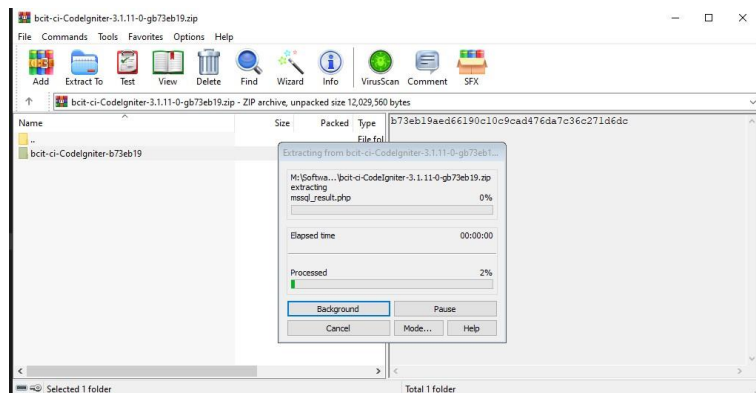
Codeigniter (CI) merupakan sebuah *web application framework* yang bersifat *open source* dimana digunakan untuk membangun aplikasi php dinamis. *Framework* itu sendiri merupakan abstraksi di dalam sebuah perangkat lunak yang menyediakan fungsi yang *generic* sehingga dapat dirubah oleh kode yang dibuat *user*, sehingga dapat menyediakan perangkat lunak untuk aplikasi tertentu.

Berikut ini merupakan tahapan-tahapan untuk instalasi codeigniter pada sistem operasi windows.

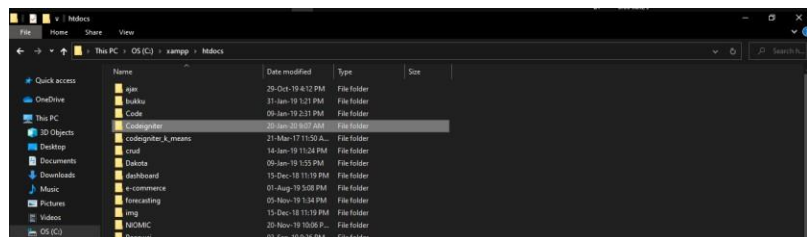
- a. Pertama, *download* terlebih dahulu codeigniter pada *website* resminya pada link berikut.
<https://codeigniter.com/download>.



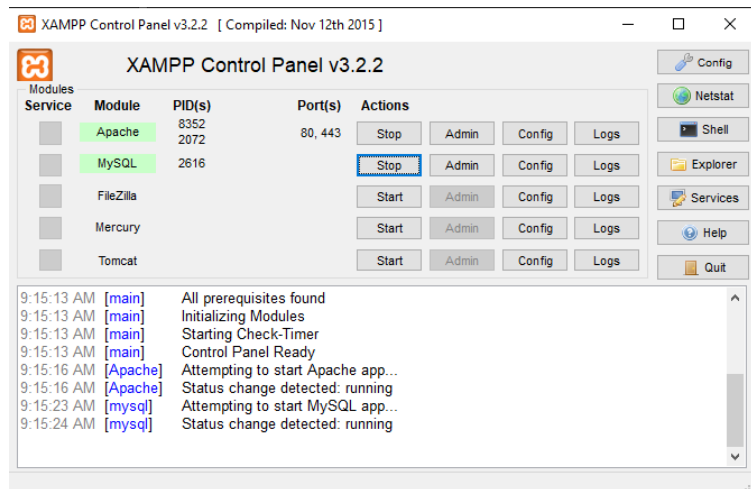
- b. Setelah berhasil di *download*, selanjutnya ekstrak file tersebut.



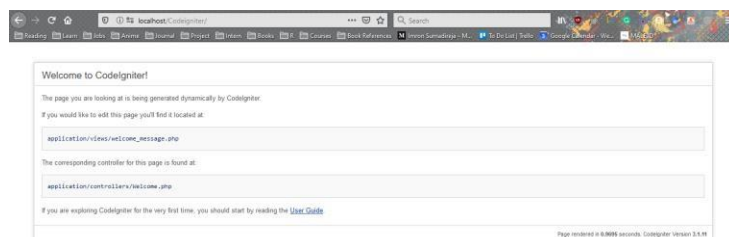
- c. Kemudian, *copy* file yang telah di ekstrak ke dalam direktori htdocs, seperti gambar berikut.



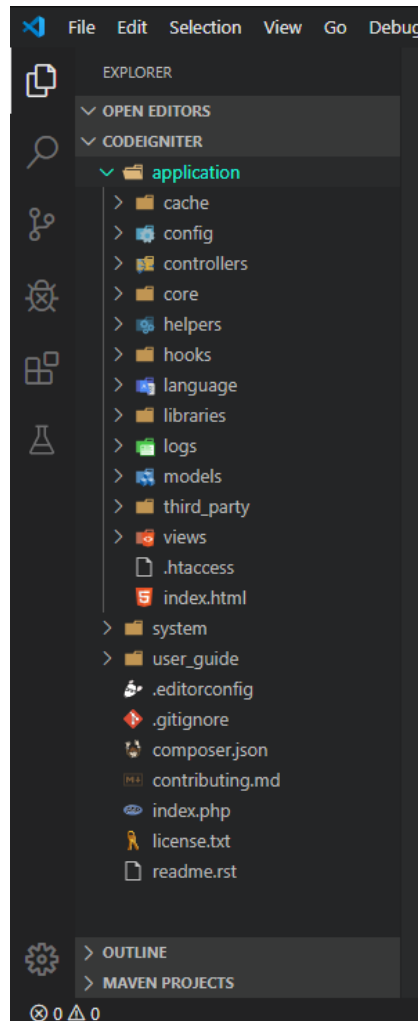
- d. Untuk menjalankannya, kita menggunakan xampp, untuk itu silakan jalankan terlebih dahulu aplikasi xampp-nya seperti berikut.
- e. Buka aplikasi xamppnya, kemudian klik tombol start pada kolom actions baris pertama untuk menjalankan apache, dan klik tombol start pada kolom actions baris kedua untuk menjalankan MySQL.



- f. Setelah xamppnya berhasil dijalankan, proses selanjutnya tinggal memanggil folder yang telah disimpan pada direktori htdocs di browser. Maka tampilan utamanya seperti gambar berikut.



- g. Berikut ini merupakan struktur folder yang terdapat pada codeigniter, diantaranya sebagai berikut.

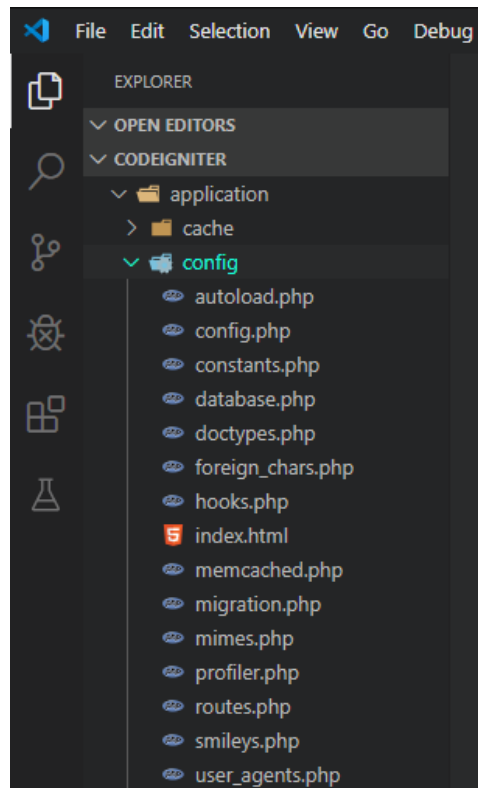


- a) *Application*, merupakan folder yang pada dasarnya menyimpan aplikasi yang sedang kita buat.
- b) *Cache*, merupakan folder yang menyimpan semua cache yang dibuat oleh cache library.

- c) *Config*, merupakan folder yang menyimpan informasi mengenai konfigurasi aplikasi seperti autoload, database, routes, dan lainnya.
- d) *Controller*, merupakan folder yang menyimpan controller-controller aplikasi yang dapat digunakan untuk menyusun aktivitas program.
- e) *Core*, merupakan folder untuk memperluas *class* inti codeigniter.
- f) *Helpers*, merupakan folder untuk menyimpan helpers.
- g) *Hooks*, merupakan folder untuk menyimpan hooks untuk mengubah alur fungsi dari core codeigniter.
- h) *Language*, merupakan folder untuk menyimpan bahasa-bahasa yang akan digunakan.
- i) *Libraries*, merupakan folder untuk menyimpan library.
- j) *Logs*, merupakan folder untuk menyimpan semua error log apabila error log diaktifkan.
- k) *Models*, merupakan salah satu dari konsep MVC yang dapat mengakses database.
- l) *Third_party*, merupakan folder untuk menyimpan fungsi-fungsi tambahan dalam cara kerja codeigniter.
- m) *Views*, merupakan folder untuk menyimpan tampilan dari aplikasi yang kita buat.
- n) *System*, merupakan folder untuk menyimpan sistem inti dari codeigniter.

3. Konfigurasi dasar pada codeigniter

Dalam memulai codeigniter, ada beberapa konfigurasi dasar yang perlu kita ketahui, diantaranya autoload.php, config.php dan database.php. Semua konfigurasi pada codeigniter, terletak pada satu tempat yakni di dalam folder *application/config*



- a. Autoload.php, file ini digunakan untuk mengatur fungsi-fungsi yang akan dimuat otomatis di awal ketika program dijalankan. Untuk melakukan konfigurasi pada file autoload.php, silakan buka file- nya seperti gambar berikut.

```
autoload.php X
application > config > @ autoload.php
1 <?php
2 defined('BASEPATH') OR exit('No direct script access allowed');
3
4 /*
5  * AUTO-LOADER
6  * -----
7  * This file specifies which systems should be loaded by default.
8  *
9  * In order to keep the framework as light-weight as possible only the
10  * absolute minimal resources are loaded by default. For example,
11  * the database is not connected to automatically since no assumption
12  * is made regarding whether you intend to use it. This file lets
13  * you globally define which systems you would like loaded with every
14  * request.
15  *
16  * -----
17  * Instructions
18  * -----
19  * These are the things you can load automatically:
20  *
21  * 1. Packages
22  * 2. Libraries
23  * 3. Drivers
24  * 4. Helper files
25  * 5. Custom config files
26  *
27  */
```

- b. Kemudian temukan kode berikut.

```
61 $autoload['libraries'] = array();
```

- c. Ubah kode tersebut menjadi seperti berikut.

```
61 $autoload['libraries'] = array('database');
```

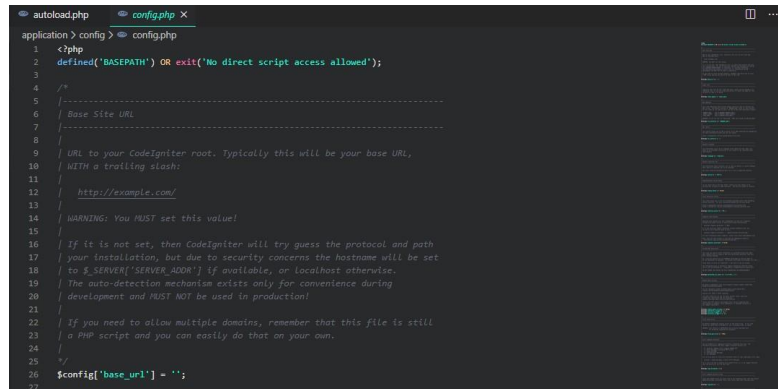
Kode tersebut dapat diartikan kita dapat meload library 'database' secara otomatis.

- d. Selanjutnya, temukan kode berikut, dan tambahkan 'url' didalamnya.

```
92 $autoload['helper'] = array('url');
93
```

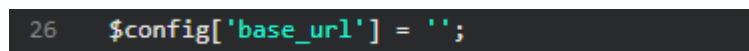
Kode tersebut dapat diartikan kita dapat meload helper "url" secara otomatis.

- e. Config.php, pada file ini terdapat beberapa konfigurasi yang secara standar sudah terkonfigurasi, namun terdapat beberapa konfigurasi yang perlu diperhatikan, untuk konfigurasi dasar, cukup mengetahui konfigurasi base_url.



```
application > config > config.php
1 <?php
2 defined('BASEPATH') OR exit('No direct script access allowed');
3
4 /*
5 |-----
6 | Base Site URL
7 |-----
8 |
9 | URL to your CodeIgniter root. Typically this will be your base URL,
10 | WITH a trailing slash:
11 |
12 | http://example.com/
13 |
14 | WARNING: You MUST set this value!
15 |
16 | If it is not set, then CodeIgniter will try guess the protocol and path
17 | your installation, but due to security concerns the hostname will be set
18 | to $_SERVER['SERVER_ADDR'] if available, or localhost otherwise.
19 | The auto-detection mechanism exists only for convenience during
20 | development and MUST NOT be used in production!
21 |
22 | If you need to allow multiple domains, remember that this file is still
23 | a PHP script and you can easily do that on your own.
24 |
25 |*/
26 $config['base_url'] = '';
```

f. Kemudian temukan kode berikut.



```
26 $config['base_url'] = '';
```

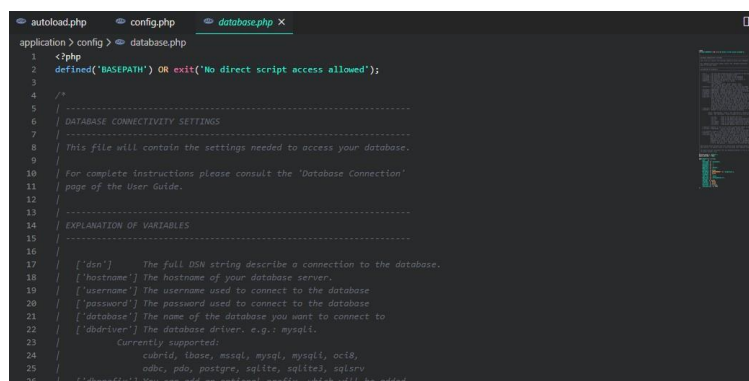
g. Kemudian tambahkan kode tersebut, menjadi nama folder



```
26 $config['base_url'] = 'http://localhost/Codeigniter/';
```

yang disimpan pada htdocs.

h. Database.php, merupakan salah satu file yang berkaitan dengan koneksi ke database. Adapun konfigurasi yang perlu diperhatikan, diantaranya: hostname, username, password dan database. Buka file database.php pada teks editor seperti gambar berikut.



```
application > config > database.php
1 <?php
2 defined('BASEPATH') OR exit('No direct script access allowed');
3
4 /*
5 |-----
6 | DATABASE CONNECTIVITY SETTINGS
7 |-----
8 | This file will contain the settings needed to access your database.
9 |
10 | For complete instructions please consult the 'Database Connection'
11 | page of the User Guide.
12 |
13 |-----
14 | EXPLANATION OF VARIABLES
15 |-----
16 |
17 | ['dsn'] The full DSN string describe a connection to the database.
18 | ['hostname'] The hostname of your database server.
19 | ['username'] The username used to connect to the database
20 | ['password'] The password used to connect to the database
21 | ['database'] The name of the database you want to connect to
22 | ['driver'] The database driver. e.g.: mysql.
23 |
24 | Currently supported:
25 | cubrid,ibase,mysql,mysqli,oci8,
26 | pdo,pgsql,sqlite,sqlite3,sqldr
```

- i. Kemudian temukan kode seperti berikut pada file database.php.

```
73 $active_group = 'default';
74 $query_builder = TRUE;
75
76 $db['default'] = array(
77     'dsn' => '',
78     'hostname' => 'localhost',
79     'username' => '',
80     'password' => '',
81     'database' => '',
82     'dbdriver' => 'mysqli',
83     'dbprefix' => '',
84     'pconnect' => FALSE,
85     'db_debug' => (ENVIRONMENT !== 'production'),
86     'cache_on' => FALSE,
87     'cachedir' => '',
88     'char_set' => 'utf8',
89     'dbcollat' => 'utf8_general_ci',
90     'swap_pre' => '',
91     'encrypt' => FALSE,
92     'compress' => FALSE,
93     'stricton' => FALSE,
94     'failover' => array(),
95     'save_queries' => TRUE
96 );
97
```

- j. Kemudian tambahkan kode tersebut seperti gambar berikut.


```

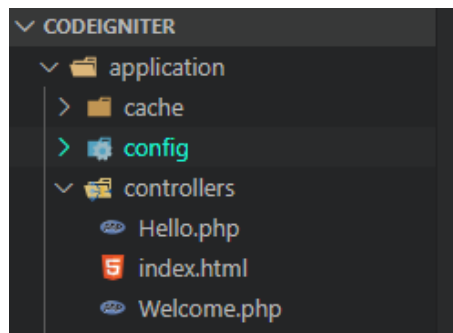
73 $active_group = 'default';
74 $query_builder = TRUE;
75
76 $db['default'] = array(
77     'dsn' => '',
78     'hostname' => 'localhost', //hostname
79     'username' => 'root', //username
80     'password' => '', //password
81     'database' => 'database_name', //nama database
82     'dbdriver' => 'mysqli',
83     'dbprefix' => '',
84     'pconnect' => FALSE,
85     'db_debug' => (ENVIRONMENT !== 'production'),
86     'cache_on' => FALSE,
87     'cachedir' => '',
88     'char_set' => 'utf8',
89     'dbcollat' => 'utf8_general_ci',
90     'swap_pre' => '',
91     'encrypt' => FALSE,
92     'compress' => FALSE,
93     'stricton' => FALSE,
94     'failover' => array(),
95     'save_queries' => TRUE
96 );
97

```

A. Hello World Codeigniter

Untuk menguji pemahaman pada codeigniter, kita akan mencoba dengan menampilkan text “Hello World” pada browser menggunakan controller.

Buat sebuah controller dengan nama Hello.php seperti gambar berikut.



Setelah itu, ketikkan kode seperti gambar berikut.

```
application > controllers > Hello.php
1  <img alt="PHP icon" data-bbox="354 354 368 368"/>?php
2  class Hello extends CI_Controller{
3      function index(){
4          echo "Hello World!";
5      }
6  }
7  <img alt="PHP icon" data-bbox="354 431 368 445"/>
```

Jika berhasil maka hasilnya akan seperti gambar berikut.



BAB III

GPS

3.1 Global Positioning System (GPS)

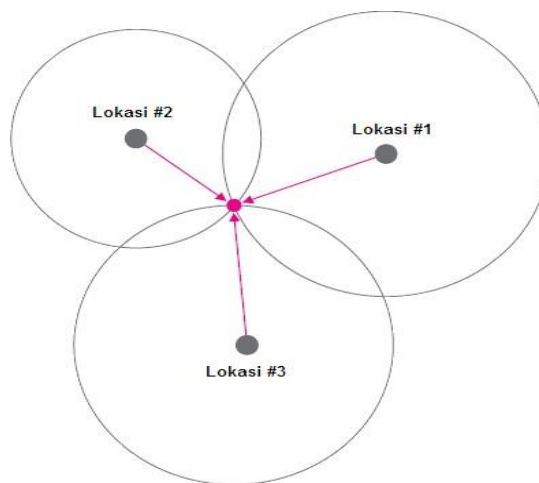
Global Positioning System (GPS) merupakan sebuah alat atau sistem yang dapat digunakan untuk menginformasikan penggunanya dimana dia berada (secara global) dipermukaan bumi yang berbasis satelit. Data dikirim dari satelit berupa sinyal radio dengan data digital.

3.1.1 Definisi Global Positioning System (GPS)

GPS (*Global Positioning System*) adalah sistem navigasi yang berbasiskan satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan (*Departemen of Defense*) Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS *reciever* yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi diubah menjadi titik yang dikenal dengan nama *Way-point* nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian di layar pada peta elektronik.

GPS adalah satu-satunya sistem satelit navigasi global untuk penentuan lokasi, kecepatan, arah, dan waktu yang telah beroperasi secara penuh di dunia saat ini (*undergraduate thesis* Wildan Habibi, ITS, Surabaya Januari : 2011). GPS menggunakan konstelasi 27 buah satelit yang mengorbit bumi, dimana sebuah GPS *receiver* menerima informasi dari tiga atau lebih satelit tersebut seperti terlihat dalam Gambar 2.1 dibawah, untuk menentukan posisi.

GPS *receiver* harus berada dalam *line-of sight* (LoS) terhadap ketiga satelit tersebut untuk menentukan posisi, sehingga GPS hanya ideal untuk digunakan dalam *outdoor positioning*.



Gambar 3.1 Trilaterasi Dalam Global Positioning System (GPS) (Sumber : Ary Mazharuddin S, S.Kom., M.Kom.Sc., Surabaya, Januari:2011)

Aplikasi yang berada disisi target (*client*) setelah mendapatkan *request* dari pelacak (*server*) maka *client* akan meminta koordinat posisinya pada GPS (*Global Positioning System*), yang kemudian akan dikirimkan ke pelacak (*server*).

Sejak tahun 1980, layanan GPS yang dulunya hanya untuk keperluan militer mulai terbuka untuk publik. Meskipun satelit-satelit tersebut berharga ratusan juta dolar, namun setiap orang dapat menggunakannya dengan gratis. Satelit-satelit ini mengorbit pada ketinggian sekitar 12.000 mil dari permukaan bumi. Posisi ini sangat ideal karena satelit dapat menjangkau *area coverage* yang lebih luas. Satelit-satelit ini akan selalu berada posisi yang bisa menjangkau semua area di atas permukaan bumi sehingga dapat meminimalkan terjadinya blank spot (area yang tidak terjangkau oleh satelit).

Setiap satelit mampu mengelilingi bumi hanya dalam waktu 12 jam. Sangat cepat, sehingga mereka selalu bisa menjangkau dimana pun posisi Anda di atas permukaan bumi. GPS *reciever* sendiri berisi beberapa *integrated circuit* (IC) sehingga murah dan teknologinya mudah untuk di gunakan oleh semua orang. GPS dapat digunakan untuk berbagai kepentingan, misalnya mobil, kapal, pesawat terbang, pertanian dan di integrasikan dengan komputer maupun laptop. (Jurnal Andi Sunyoto, STMIK AMIKOM Jogjakarta, 2013:1)

Berikut beberapa contoh perangkat GPS *reciever*:



Gambar 3.2 Macam-Macam Perangkat GPS

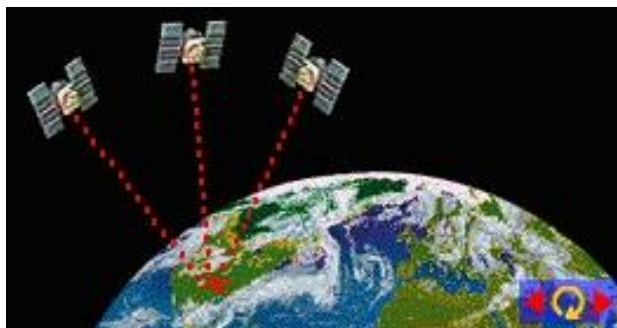
**(Sumber : Jurnal Andi Sunyoto, STMIK AMIKOM
Jogjakarta, 2013:1)**

3.1.2 Cara Kerja *Global Positioning System* (GPS)

Setiap daerah di atas permukaan bumi ini minimal terjangkau oleh 3-4 satelit. Pada prakteknya, setiap GPS terbaru bisa menerima sampai dengan 12 channel satelit sekaligus. Kondisi langit yang cerah dan bebas dari halangan membuat GPS dapat dengan mudah menangkap sinyal yang dikirimkan oleh satelit. Semakin banyak satelit yang diterima oleh GPS, maka akurasi yang diberikan juga akan semakin tinggi.

Cara kerja GPS secara sederhana ada 5 langkah, yaitu :

1. Memakai perhitungan “*triangulation*” dari satelit.
2. Untuk perhitungan “*triangulation*”, GPS mengukur jarak menggunakan travel time sinyal radio.
3. Untuk mengukur *travel time*, GPS memerlukan memerlukan akurasi waktu yang tinggi.
4. Untuk perhitungan jarak, kita harus tahu dengan pasti posisi satelit dan ketinggian pada orbitnya.
5. Terakhir harus mengoreksi *delay* sinyal waktu perjalanan di atmosfer sampai diterima receiver.



Gambar 3.3 Cara Satelit menentukan Posisi

**(Sumber : Jurnal Andi Sunyoto, STMIK AMIKOM
Jogjakarta, 2013:1)**

Satelit GPS berputar mengelilingi bumi selama 12 jam di dalam orbit yang akurat dia dan mengirimkan sinyal informasi ke bumi. GPS *reciever* mengambil informasi itu dan dengan menggunakan perhitungan “*triangulation*” menghitung lokasi *user* dengan tepat. GPS *reciever* membandingkan waktu sinyal di kirim dengan waktu sinyal tersebut di terima. Dari informasi itu didapat diketahui berapa jarak satelit. Dengan perhitungan jarak GPS *reciever* dapat melakukan perhitungan dan menentukan posisi user dan menampilkan dalam peta elektronik.



Gambar 3.4 Tampilan GPS Reciever

**(Sumber : Jurnal Andi Sunyoto, STMIK AMIKOM
Jogjakarta, 2013:1)**

Sebuah GPS *reciever* harus mengunci sinyal minimal tiga satelit untuk menghitung posisi 2D (*latitude* dan *longitude*) dan *track* pergerakan. Jika GPS *receiver* dapat menerima empat atau lebih satelit, maka dapat menghitung posisi 3D (*latitude*, *longitude* dan *altitude*). Jika sudah dapat menentukan posisi *user*, selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah yang dituju, jalur, tujuan perjalanan, jarak tujuan, matahari terbit dan matahari terbenam dan masih banyak lagi.

Satelit GPS dalam mengirim informasi waktu sangat presisi karena Satelit tersebut memakai jam atom. Jam atom yang ada pada satelit ialah dengan partikel atom yang di isolasi, sehingga dapat menghasilkan jam yang akurat dibandingkan dengan jam biasa. Perhitungan waktu yang akurat sangat menentukan akurasi perhitungan untuk menentukan informasi lokasi kita. Selain itu semakin banyak sinyal satelit yang dapat diterima maka akan semakin presisi data yang diterima karena ketiga satelit mengirim *pseudo-random code* dan waktu yang sama. Ketinggian itu menimbulkan keuntungan dalam mendukung proses kerja GPS, bagi kita karena semakin tinggi maka semakin bersih atmosfer, sehingga gangguan semakin sedikit dan orbit yang cocok dan perhitungan matematika yang cocok. Satelit harus tetap pada posisi yang tepat sehingga stasiun di bumi harus terus memonitor setiap pergerakan satelit, dengan bantuan radar yang presisi selalu di cek tentang altitude, position dan kecepatannya.

3.1.3 Cara Satelit Menentukan Posisi Lokasi

Sinyal yang dikirimkan oleh satelit ke GPS akan digunakan untuk menghitung waktu perjalanan (*travel time*). Waktu perjalanan ini sering juga disebut sebagai *Time of Arrival* (TOA). Sesuai dengan prinsip fisika, bahwa untuk mengukur jarak dapat diperoleh dari waktu dikalikan dengan cepat rambat sinyal. Maka, jarak antara satelit dengan GPS juga dapat diperoleh dari prinsip fisika tersebut. Setiap sinyal yang dikirimkan oleh satelit akan juga berisi informasi yang sangat detail, seperti orbit satelit, waktu, dan hambatan di atmosfer. Satelit menggunakan jam atom yang merupakan satuan waktu paling presisi. Untuk dapat menentukan posisi dari sebuah GPS secara dua dimensi (jarak),

Dibutuhkan minimal tiga buah satelit. Empat buah satelit akan dibutuhkan agar didapatkan lokasi ketinggian (secara tiga dimensi). Setiap satelit akan memancarkan sinyal yang akan diterima oleh GPS *receiver*. Sinyal ini akan dibutuhkan untuk menghitung jarak dari masing-masing satelit ke GPS. Dari jarak tersebut, akan diperoleh jari-jari lingkaran jangkauan setiap satelit. Lewat perhitungan matematika yang cukup rumit, interseksi (perpotongan) setiap lingkaran jangkauan satelit tadi akan dapat digunakan untuk menentukan lokasi dari GPS di permukaan bumi. (Jurnal Andi Sunyoto, STMIK AMIKOM Jogjakarta, 2013:1)

3.1.4 Manfaat Penggunaan *Global Positioning System* (GPS)

Dengan menggunakan GPS, Anda dapat menandai semua lokasi yang pernah Anda kunjungi. Misalnya, Lokasi Politeknik Negeri Sriwijaya kita beri *waypoint* dan tempat-tempat lainnya. Sebenarnya, ada banyak manfaat yang bisa diambil jika Anda mengetahui *waypoint* dari suatu tempat. Pertama, Anda dapat memperkirakan jarak lokasi yang Anda tuju dengan lokasi asal Anda. GPS keluaran terakhir dapat memperkirakan jarak Anda ke tujuan, sampai estimasi lamanya perjalanan dengan kecepatan aktual yang sedang Anda tempuh. Kedua, lokasi di daratan memang cukup mudah untuk dikenali dan diidentifikasi. Namun, jika Anda kebetulan berada ditempat memancing yang terletak di tengah lautan ataupun tempat melihat matahari terbenam yang berada di puncak gunung. Di saat seperti inilah sebuah GPS akan menunjukkan manfaatnya.

Dengan teknologi GPS dapat digunakan untuk beberapa keperluan sesuai dengan tujuannya. GPS dapat digunakan oleh peneliti, olahragawan, petani, tentara, pilot, petualang, pendaki, pengantar barang, pelaut, kurir, penebang pohon, pemadam kebakaran dan orang dengan berbagai kepentingan untuk meningkatkan produktivitas, keamanan, dan untuk kemudahan. Dari beberapa pemakai di atas dikategorikan menjadi:

1. Lokasi

Digunakan untuk menentukan dimana lokasi suatu titik dipermukaan bumi berada.

2. Navigasi

Membantu mencari lokasi suatu titik di bumi

3. *Tracking*

Membantu untuk memonitoring pergerakan obyek dan membantu memetakan posisi tertentu, dan perhitungan jaringan terdekat

4. *Timing*

Dapat dijadikan dasar penentuan jam seluruh dunia, karena memakai jam atom yang jauh lebih presisi dibanding dengan jam biasa.

Tidak peduli posisi Anda, di tengah laut, di tengah hutan, di atas gunung, ataupun di pusat kota. Selama GPS dapat menerima sinyal dari satelit secara langsung tanpa halangan, maka GPS akan selalu memberikan informasi koordinat posisi Anda. GPS membutuhkan area pandang yang bebas langsung ke langit. Halangan-halangan seperti pohon, gedung, bahkan kaca film kelas *V-Kool*, bisa mengurangi akurasi sinyal yang diterima oleh GPS. Bahkan bukan tidak mungkin GPS tidak bisa menerima sinyal sama sekali dari satelit.

GPS juga memiliki *feature* tambahan yang mampu memberikan informasi selama anda di perjalanan, seperti kecepatan, lama perjalanan, jarak yang telah ditempuh, waktu, dan masih banyak. (Jurnal Andi Sunyoto, STMIK AMIKOM Jogjakarta, 2013:1)

3.1.5 Model Dan Interkoneksi *Global Positioning System* (GPS)

Sebuah GPS juga memiliki *firmware* yang bisa di-upgrade. *Upgrade firmware* ini bisaanya disediakan pada *site* produsen GPS tersebut. *Upgrade firmware* bisaanya menggunakan kabel yang dibundel atau-pun tersedia sebagai aksesoris. Kabel ini juga ternyata bisa digunakan untuk menghubungkan GPS ke komputer (baik itu notebook, PC, maupun PDA dengan sedikit bantuan konverter). *Software* GPS yang tersedia untuk berbagai *platform* tersebut juga cukup banyak. Dengan software tersebut, Anda dapat dengan mudah *mendownload* informasi dari GPS. Memori sebuah GPS memang relatif terbatas, sehingga kemampuan ekstra untuk menyimpan informasi yang pernah Anda tempuh ke PC/PDA (yang bisaanya memiliki memori lebih besar) tentu akan sangat menyenangkan. Untuk media komunikasi GPS dengan *hardware* lain.

selain kabel, model GPS sekarang juga ada yang dilengkapi dengan *Bluetooth*, *Infrared*.

Berdasarkan fisik, model GPS dibagi menjadi beberapa tipe antara lain model *portable/handheld* (ukurannya menyerupai ponsel), ada yang lebih besar (bisaanya di mobil/kapal), ada pula yang menggunakan *interface* khusus untuk dikoneksikan ke *notebook* maupun PDA (*Palm*, *Pocket PC* maupun *Nokia Communicator*). GPS untuk keperluan *outdoor* bisaanya juga dilengkapi dengan perlindungan anti air dan tahan benturan.

Beberapa GPS keluaran terakhir bahkan sudah menyediakan layar warna dan kemampuan komunikasi radio jarak pendek (*FRS/Family Radio Service*). Tentu saja, semakin banyak *feature* yang ditawarkan pada sebuah GPS maka semakin tinggi pula harganya.

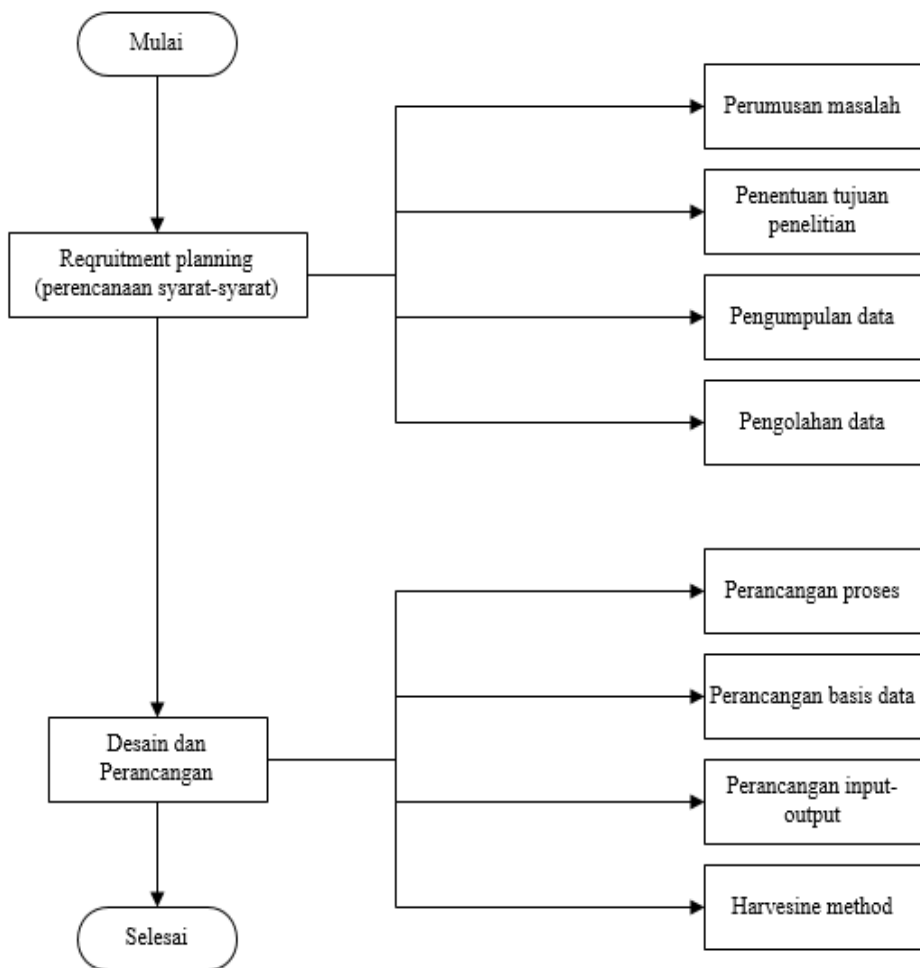
Jika suatu saat Anda ingin pergi ke lokasi yang pernah Anda kunjungi dengan menggunakan GPS. Maka, Anda tinggal mengupload data yang pernah Anda simpan di komputer kembali ke GPS. Selanjutnya, Anda akan mendapatkan rekaman perjalanan Anda terdahulu. Lokasi dan *track* yang pernah Anda kunjungi akan dapat Anda temui kembali dengan cepat, dan tentu saja meminimalkan resiko tersesat. (Jurnal Wildan Habibi, ITS : 2010)

BAB IV

IMPLEMENTASI SISTEM

4.1 Diagram Alur Metodologi Penelitian

Metode penelitian yang di gunakan dalam penelitian ini di bagi kedalam beberapa tahap agar peroses yang dilakukan lebih terarah. Secara umum langkah-langkah penelitian yang di lakukan untuk memonitoring kinerja mahasiswa *internship* di Politeknik Pos Indonesia adalah sebagai berikut :



Gambar 4.1 Diagram Alur Metodologi Penelitian

4.2 Tahapan-Tahapan Diagram Alur Metodeologi Penelitian

Tahapan -tahapan diagram alur metodeologi penelitian yang digunakan peneliti adalah sebagai berikut :

4.2.1 Perumusan Masalah

Pada langkah ini penulis akan mencari permasalahan apa yang terjadi di perguruan tinggi, yang selanjutnya akan diteliti sehingga masalah yang akan dibahas menjadi lebih mudah dalam penentuan metode yang digunakan.

Dari hasil peninjauan masalah penulis menemukan bahwa sistem kegiatan mahasiswa *internship* di Politeknik Pos Indonesia ditemukan permasalahan yang terjadi yaitu :

1. Bagaimana memodelkan aplikasi monitoring kinerja mahasiswa *internship*.
2. Bagaimana membuat aplikasi monitoring kinerja mahasiswa *internship* berdasarkan proses bisnis yang ada.
3. Bagaimana menggunakan metode harvesine untuk menghitung jarak kedua titik koordinat.

4.2.2 Penentuan Tujuan Penelitian

Pada langkah ini penulis akan menentukan tujuan dari dilakukannya penelitian ini. Tujuan dilakukannya penelitian ini yaitu untuk membangun sebuah sistem sehingga mempermudah bagian admin atau prodi dalam memonitoring kinerja mahasiswa *internship*.

1. Membuat pemodelan sistem sehingga dapat menghasilkan *warning* yang dibutuhkan.
2. Untuk lebih memaksimalkan desain *user interface* monitoring kinerja mahasiswa *internship* dengan aplikasi berbasis *website*.

3. Untuk menghitung jarak kedua titik koordinat pada monitoring kinerja mahasiswa *internship* dengan menggunakan metode harvesine.

4.2.3 Pengumpulan Data

Pengumpulan data yaitu pengumpulan data yang diperlukan untuk menyelesaikan laporan ini, data-data yang akan dikumpulkan oleh penulis yaitu merupakan data yang dibutuhkan dalam membangun sistem monitoring kinerja mahasiswa internship.

Dalam pengumpulan data ini penulis memperoleh dua data, yaitu :

1. Data primer merupakan data yang diperoleh secara langsung dari sumbernya diamati dan dicatat untuk pertama kalinya, dan mempunyai hubungan erat dengan permasalahan yang dihadapi perusahaan tersebut. Metode wawancara atau interview dipergunakan untuk memperoleh data dengan metode wawancara dengan narasumber yang akan diwawancarai.
2. Data sekunder adalah data yang tidak langsung memberikan data kepada peneliti, misalnya penelitian harus melalui orang lain atau mencari melalui dokumen. Data ini diperoleh dengan menggunakan studi literatur yang dilakukan terhadap banyak buku dan diperoleh berdasarkan catatan-catatan yang berhubungan dengan penelitian, selain itu peneliti mempergunakan data yang diperoleh dari internet.

4.2.4 Pengolahan Data

Setelah data-data yang diperlukan dalam penelitian ini telah terkumpul, maka langkah selanjutnya pengolahan data. Pengolahan data adalah manipulasi data agar menjadi bentuk yang lebih berguna. Data tersebut yaitu:

1. Data laporan kinerja

Dari hasil laporan kinerja yang dilakukan sesuai jadwal masuk kegiatan mahasiswa *internship* dilakukan rekap data yang bisa dijadikan *logbook* dan menghitung data laporan yang masuk serta jarak koordinat yang dihitung.

4.2.5 Analisis dan Perancangan Sistem

Dari data yang telah dikumpulkan selanjutnya dilakukan analisis sistem yang akan dibangun berdasarkan sistem yang sedang berjalan saat ini. Kemudian selanjutnya mulai dilakukan perancangan sistem berdasarkan hasil analisis kebutuhan sistem.

4.2.5.1 Perancangan Proses

Merancang proses sistem ini dengan menggunakan *tool* yang sama dengan tahap analisis sistem yaitu UML (*Unified Modelling Language*) agar lebih memahami langkah awal membangun sistem secara fisik.

4.2.5.2 Perancangan Basisdata

Merancang basis data (*database*) yang dilakukan dengan class diagram yang menggambarkan hubungan antar *entity* yang ada pada *use case* diagram dan spesifikasi tabel.

4.2.5.3 Perancangan Input-Output

Merancang Input-Output dengan membuat rancangan layar tampilan. Setelah rancangan layar tampilan terbentuk maka dilakukan tahap implementasi.

4.2.5.4 Method Harvesine

Metode Haversine Formula dapat digunakan untuk menghitung jarak antara dua titik, berdasarkan posisi garis lintang latitude dan posisi garis bujur longitude sebagai variabel inputan. Haversine Formula adalah persamaan penting pada navigasi, memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang. Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R 6.367, 45 km, dan lokasi dari 2 titik di koordinat bola (lintang dan bujur) masing-masing adalah lon1, lat1, dan lon2, lat2 [25]. Metode Haversine Formula tersebut kini sudah mengalami pengembangan, yaitu dengan menggunakan rumus *spherical law of cosine* sederhana, dimana dengan penghitungan komputer dapat memberikan tingkat presisi yang sangat akurat antar dua titik. Pertama ditentukan terlebih dahulu titik awal dan titik tuju, titik awal berupa latitude1(lat1) dan longitude1(long1), titik tuju berupa latitude2(lat2) dan longitude2(long2). Titik awal dan titik tuju tersebut berbentuk desimal derajat yang kemudian dirubah menjadi nilai sudut radian, kemudian lakukan perhitungan dengan rumus Haversine Formula, yaitu:

Rumus Harvesine

$$x = (\text{lng2}-\text{lng1}) * \cos ((\text{lat1}+\text{lat2})/2);$$

$$y = (\text{lat2}-\text{lat1});$$

$$d = \text{sqrt}(x*x+y*y)*R$$

Keterangan :

x = longitude (lintang)
y = latitude (bujur)
d = jarak (km)
R = Radius Bumi = 6371 km
1 derajat = 0.0174532925 radian

Apabila metode ini di implemetasikan pada sistem yang akan dibangun maka proses yang dilakukan menyesuaikan dengan kebutuhan pada aplikasi tersebut yang dimana titik koordinat pertama *latitude*, *longitude* adalah sebagai titik acuan yang diambil dari data registrasi mahasiswa/i, sedangkan untuk titik koordinat kedua mengambil dari data *report activity* harian yang mana hasil akhir dari perhitungan kedua jarak digunakan sebagai data kehadiran mahasiswa/i dalam bentuk penilaian oleh pembimbing.

5.1. Analisis dan Perancangan Sistem

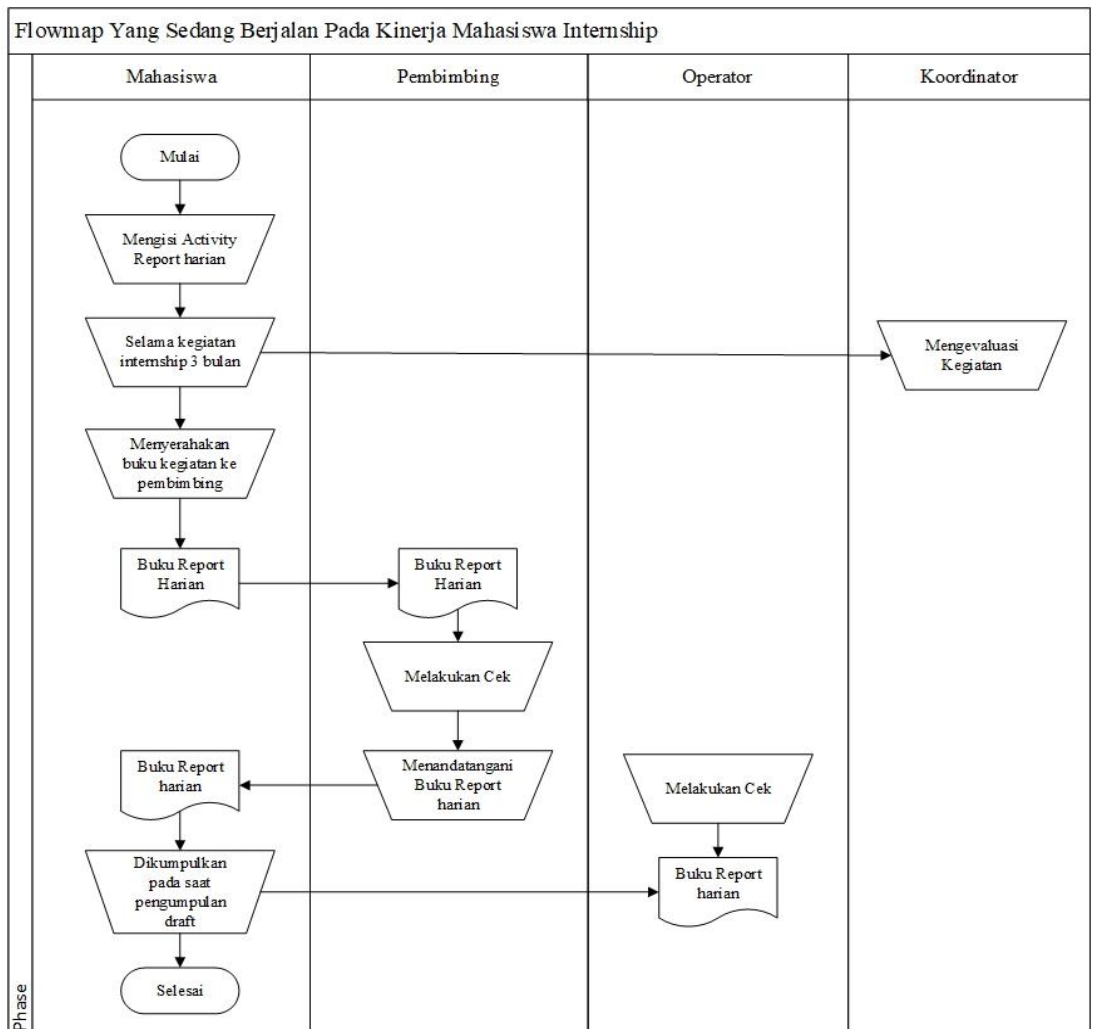
Perencanaan sistem menyangkut estimasi dari kebutuhan-kebutuhan fisik, tenaga kerja dan dana yang dibutuhkan untuk mendukung pengembangan sistem ini serta untuk mendukung operasinya setelah diterapkan sedangkan analisis merupakan proses untuk menentukan bentuk dari kebutuhan sistem yang menunjang kebutuhan pada saat membangun dan implementasi. Secara garis besar disebut juga sebagai proses mempelajari aktifitas system untuk memahami gambaran menyeluruh tentang sehingga perancang telah mengetahui apa saja kebutuhan dari sistem tersebut.

5.1.1. Analisis Sistem Berjalan (*Current System*)

Sistem kegiatan mahasiswa *internship* di Politeknik Pos Indonesia saat ini sedang berjalan masih menggunakan cara manual lebih khususnya dibagian pengawasan kinerja atau laporan kegiatan harian yang dilakukan mahasiswa tersebut. Kendala yang terjadi adalah kurang efektif karena data dapat dimanipulasi apabila dilakukan dengan cara manual sehingga data yang dibutuhkan kurang valid.

5.1.1.1. Analisis Prosedur yang berjalan (*Flowmap*)

Flowmap untuk analisis prosedur yang sedang berjalan dibuat agar alur sistem monitoring kinerja mahasiswa *internship* yang sedang berjalan dapat lebih mudah dipahami. Berikut adalah analisis sistem yang sedang berjalan dalam bentuk *flowmap* atau prosedur :



Gambar 5.1 Flowmap kegiatan harian internship yang sedang berjalan

Keterangan :

Pada *flowmap* tersebut dapat dijelaskan bahwa mahasiswa mengisi formulir kehadiran kegiatan *internship*. Di dalam buku tersebut tercantum form catatan yang harus diisi oleh mahasiswa/i selama kegiatan *internship* berlangsung yaitu 3 (tiga) bulan untuk sebagai *report* harian, *activity* apa saja yang dilakukan oleh mahasiswa/i pada tempat *internship*nya. Kemudian mahasiswa/i menyerahkan *buku/logbook* tersebut kepada pembimbing untuk

dilakukannya pengecekan serta menandatangani *report* harian dan operator melakukan pengecekan pada saat pengumpulan *draft*.

5.1.1.2. Analisis Dokumen Yang Digunakan

Analisis dokumen yang digunakan merupakan tahap analisis terhadap beberapa dokumen yang digunakan, yaitu :

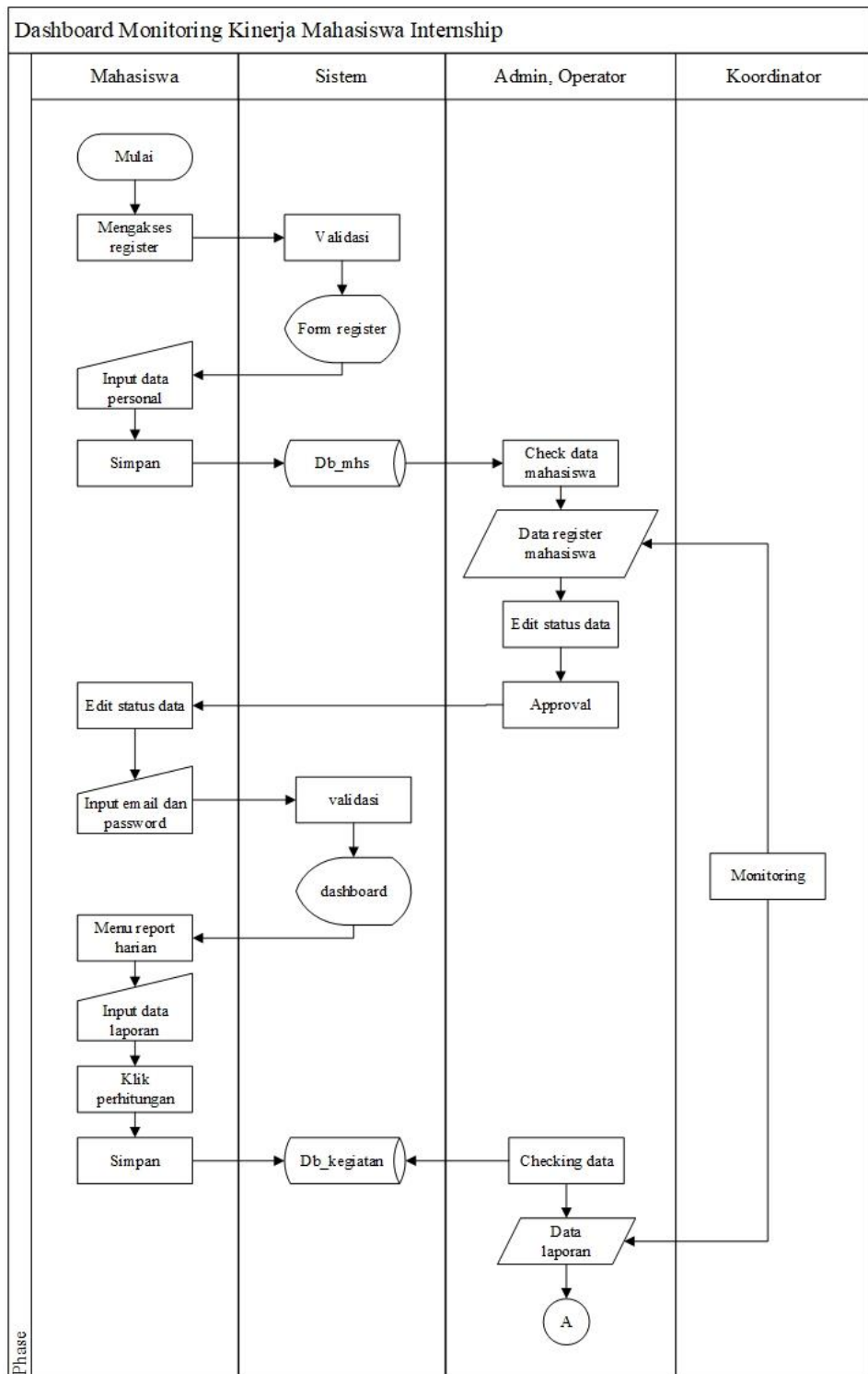
1. Formulir *report* harian , berupa form catatan aktivitas yang dilakukan dan menjadi sebagai buku *logbook*.
2. Dokumen data laporan, menjelaskan mengenai data-data laporan yang dikirimkan oleh mahasiswa berupa laporan atau materi ke pembimbing eksternal/mentor.

5.1.2. Analisis Sistem yang akan dibangun

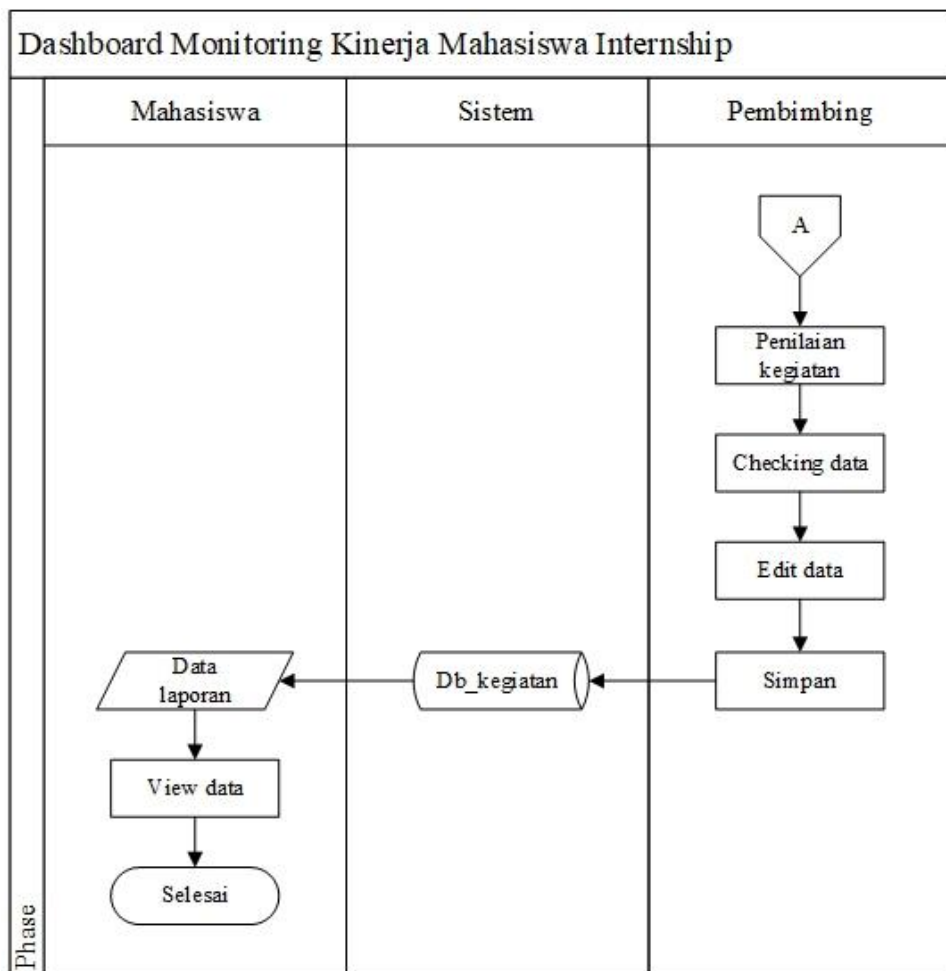
Dari analisis yang telah dilakukan maka akan dibangun sistem yang dapat memenuhi kebutuhan perusahaan/ perguruan tinggi dalam pengolahan data kinerja mahasiswa *internship*. Sistem ini sebelumnya dilakukan secara manual dan akan dibuat terkomputerisasi sehingga proses bisnis yang terjadi dapat lebih baik lagi dan meng-efisien-kan kinerja mahasiswa yang *internship* dan memperkecil kesalahan dalam data laporan.

Proses-proses yang sebelumnya dilakukan secara manual akan dijadikan sistem yang mempermudah user. Sistem yang akan dibangun memfokuskan pada pengolahan data laporan kinerja sehingga proses lebih efektif dapat berjalan sesuai kebutuhan user.

5.1.2.1 Analisis Prosedur yang akan dibangun (*flowmap*)



Gambar 5.2 Flowmap monitoring kinerja mahasiswa internship yang akan dibangun



Gambar 5.3 Flowmap monitoring kinerja mahasiswa internship yang akan dibangun

Keterangan :

Pada gambar *flowmap* diatas dapat dijelaskan aktor pertama dimulai dari mahasiswa melakukan tahap registrasi lalu tahap kedua aktor yang akan menerima data registrasi adalah operator untuk mengubah status data. Setelah data di *approval* maka operator memberikan hak akses untuk login kepada mahasiswa lalu sistem akan memvalidasi akun tersebut. Selanjutnya sistem akan menampilkan halaman *dashboard* dan mahasiswa memilih menu *report* harian kemudian meng-input data kegiatan yang dilakukan *user* tersebut lalu menyimpannya.

Data kegiatan yang sudah dilakukan aksi maka akan menjadi data laporan yang tersimpan pada *database*, kemudian aktor pembimbing prodi DIV Teknik Informatika akan melakukan tahap *check-ing* menyeluruh data laporan untuk dilakukannya penilaian sebagaimana proses tersebut sama dengan koordinator hanya perbedaan dari segi aksi yaitu hanya me-monitoing ke seluruhan data.

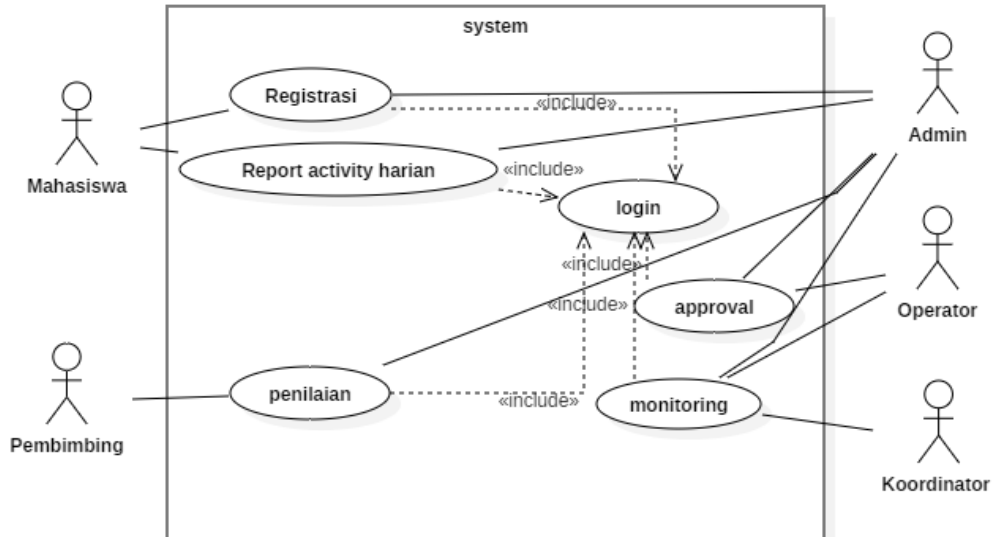
5.1.2.2. Analisis Dokumen yang dibangun

Dalam sistem monitoring kinerja mahasiswa *internship* ini terdapat beberapa dokumen yang digunakan, yaitu :

1. Dokumen data laporan, menjelaskan mengenai data-data laporan yang dikirimkan oleh mahasiswa berupa laporan atau materi ke pembimbing untuk dijadikan sebagai penilaian.

5.2. UML (*Unified Modelling Language*)

5.2.1. Usecase Diagram



Gambar 5.4 Usecase Diagram Dashboard Monitoring Kinerja Mahasiswa Internship

5.2.1.1. Definsi Aktor

Pada bagian ini akan dijelaskan aktor-aktor yang terlibatkan didalam sistem:

Tabel 5.1 Definisi Aktor

No	Aktor	Deskripsi
1.	Admin	Admin adalah pihak yang bertanggung jawab atas segala pengolahan data yang ada didalam sistem.
2.	Koordinator	Koordinator adalah pihak yang bertanggung jawab serta melakukan koordinasi kegiatan yang dilakukan.
3.	Mahasiswa	Mahasiswa adalah pihak yang memiliki hak untuk melakukan pengisian <i>report activity</i> harian.
4.	Operator	Operator adalah pihak yang bertugas menjaga, melayani, menjalankan suatu peralatan, mesin, telepon dan sebagainya
5.	Pembimbing	Pembimbing adalah pihak yang bertugas untuk melakukan penilaian tugas yang dilakukan oleh mahasiswa.

5.2.1.2. Definisi Usecase

Pada bagian ini akan dijelaskan usecase yang ada didalam sistem :

Tabel 5.2 Definisi Usecase

No	Usecase	Deskripsi
1.	Registrasi	Merupakan aktivitas untuk melakukan pendaftaran data diri sehingga terhubung dengan sistem.
2.	<i>Report activity</i> harian	Merupakan aktivitas untuk mencatat kegiatan yang dilakukan dalam bentuk sebagai laporan.

3.	Approval	Merupakan aktivitas untuk melakukan penerimaan data apabila sesuai dengan kebutuhan.
4.	Penilaian	Merupakan aktivitas untuk melakukan pengolahan informasi dan mengukur terkait pencapaian hasil kinerja yang telah dilakukan.
5.	Monitoring	Merupakan suatu kegiatan mengamati secara seksama suatu keadaan atau kondisi, termasuk juga perilaku atau kegiatan tertentu.
6.	Login	Merupakan pemberian hak akses kepada pihak yang terkait dengan sistem (aktor).

5.2.1.3. Skenario Usecase

Skenario untuk tiap masing-masing usecase diatas adalah :

Tabel 5.3 Skenario Diagram Usecase Registrasi

Identifikasi	
Nomor	UC1
Nama	<i>Registrasi</i>
Tujuan	Merupakan aktivitas untuk melakukan pendaftaran data diri sehingga terhubung dengan sistem.
Deskripsi	
Aktor	Mahasiswa/i
Skenario Utama	
Kondisi Awal	Form sudah tersedia
Kondisi Akhir	Data registrasi mahasiswa telah berhasil ditambahkan.
Main Flow Event	
Aksi aktor	Reaksi sistem
1. Mahasiswa memilih <i>link register a new membership</i>	2. Sistem akan menampilkan form registrasi mahasiswa

3. Mahasiswa akan melakukan <i>insert</i> data diri pada form yang tersedia.	4. Mengambil data dari <i>database</i> kemudian terjadi eksekusi terhadap perintah yang diberikan dan nantinya akan disimpan di <i>database</i>
<i>Exceptional Flow of Event</i>	Jika aktor salah dalam memasukkan data maka sistem akan menampilkan pesan <i>error</i> . Sedangkan jika data valid maka data akan disimpan di <i>database</i> .

Tabel 5.4 Skenario Diagram Usecase Report activity harian

Identifikasi	
Nomor	UC2
Nama	<i>Report activity harian</i>
Tujuan	Merupakan aktivitas untuk mencatat kegiatan yang dilakukan dalam bentuk sebagai laporan.
Deskripsi	
Aktor	Mahasiswa/i
Skenario Utama	
Kondisi Awal	Form sudah tersedia
Kondisi Akhir	Data <i>report</i> harian telah berhasil ditambahkan.
<i>Main Flow Event</i>	
Aksi aktor	Reaksi sistem
1. Mahasiswa memilih menu <i>report</i> harian	2. Sistem akan menampilkan form <i>report activity harian</i> .
3. Mahasiswa akan melakukan <i>insert</i> data laporan form yang tersedia dan melakukan perhitungan.	4. Mengambil data dari <i>database</i> kemudian terjadi eksekusi terhadap perintah yang diberikan dan nantinya akan disimpan di <i>database</i>
<i>Exceptional Flow of Event</i>	Jika aktor salah dalam memasukkan data maka sistem akan menampilkan pesan <i>error</i> .

	Sedangkan jika data valid maka data akan disimpan di <i>database</i> .
--	--

Tabel 5.5 Skenario Diagram Usecase approval

Identifikasi	
Nomor	UC3
Nama	<i>Approval</i>
Tujuan	Merupakan aktivitas untuk melakukan penerimaan data apabila sesuai dengan kebutuhan.
Deskripsi	
Aktor	Admin, Operator
Skenario Utama	
Kondisi Awal	Form sudah tersedia
Kondisi Akhir	Data mahasiswa telah berhasil dikelola (<i>insert, update, delete, view</i>).
Main Flow Event	
Aksi aktor	Reaksi sistem
1. Aktor memilih menu data mahasiswa	2. Sistem akan menampilkan form isian data mahasiswa
3. Aktor akan melakukan <i>insert, update, view</i> atau <i>delete</i> data.	4. Mengambil data dari <i>database</i> kemudian terjadi eksekusi terhadap perintah yang diberikan dan nantinya akan disimpan di <i>database</i>
Exceptional Flow of Event	Jika aktor salah dalam memasukkan data maka sistem akan menampilkan pesan <i>error</i> . Sedangkan jika data valid maka data akan disimpan di <i>database</i> .

Tabel 5.6 Skenario Diagram Usecase penilaian

Identifikasi

Nomor	UC4
Nama	<i>Penilaian</i>
Tujuan	Merupakan aktivitas untuk melakukan pengolahan informasi dan mengukur terkait pencapaian hasil kinerja yang telah dilakukan.
Deskripsi	
Aktor	Pembimbing
Skenario Utama	
Kondisi Awal	Form sudah tersedia
Kondisi Akhir	Data laporan telah berhasil dikelola (<i>insert, update, delete, view</i>).
Main Flow Event	
Aksi aktor	Reaksi sistem
1. Aktor memilih menu penilaian, selanjutnya mengklik <i>detail</i> data.	2. Sistem akan menampilkan form isian data laporan mahasiswa per- <i>progress</i>
3. Aktor akan melakukan <i>insert, update, view</i> atau <i>delete</i> data.	4. Mengambil data dari <i>database</i> kemudian terjadi eksekusi terhadap perintah yang diberikan dan nantinya akan disimpan di <i>database</i>
Exceptional Flow of Event	Jika aktor salah dalam memasukkan data maka sistem akan menampilkan pesan <i>error</i> . Sedangkan jika data valid maka data akan disimpan di <i>database</i> .

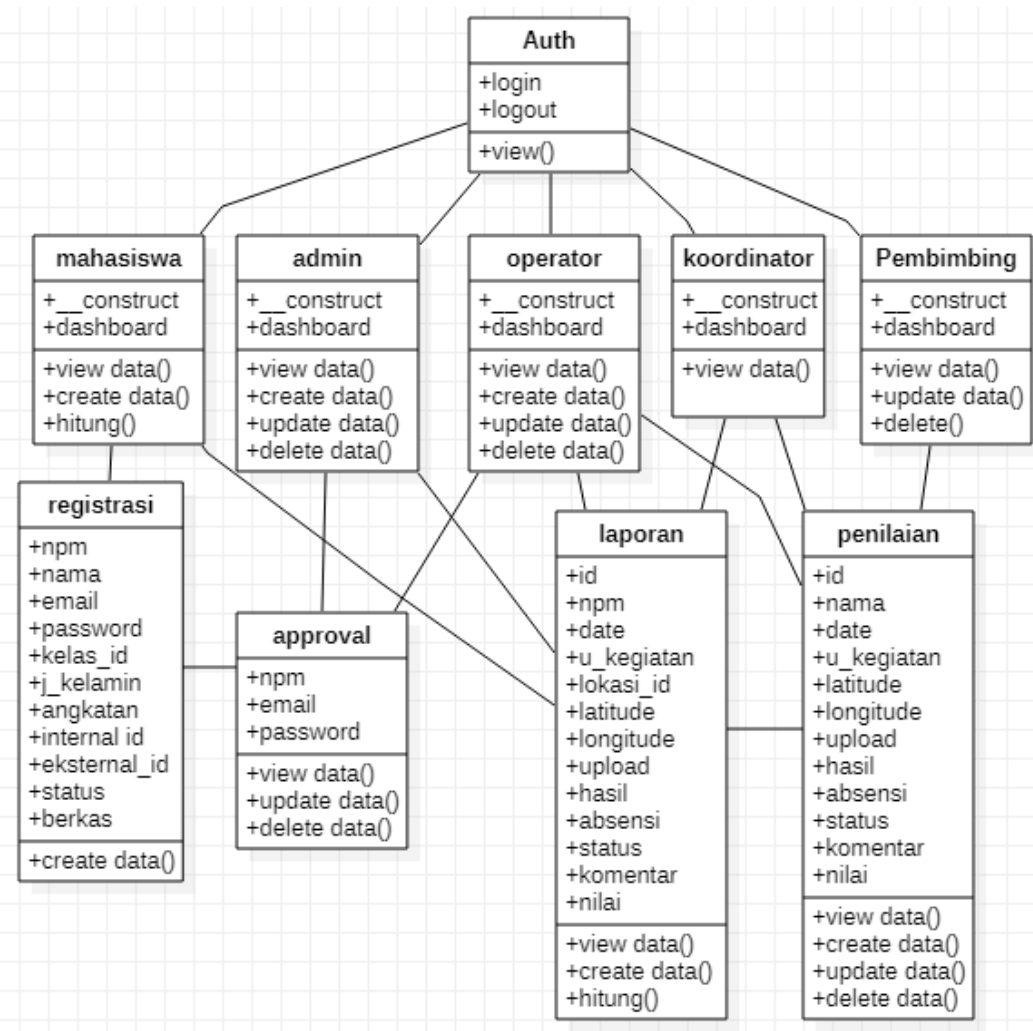
Tabel 5.7 Skenario Diagram Monitoring

Identifikasi	
Nomor	UC5
Nama	<i>Monitoring</i>

Tujuan	Merupakan aktivitas untuk melakukan pengolahan informasi dan mengukur terkait pencapaian hasil kinerja yang telah dilakukan.
Deskripsi	
Aktor	Koordinator
Skenario Utama	
Kondisi Awal	Form sudah tersedia
Kondisi Akhir	Semua data yang dikelola. (<i>view</i>).
<i>Main Flow Event</i>	
Aksi aktor	Reaksi sistem
1. koordinator semua menu yang ada pada aplikasi.	2. Sistem akan menampilkan form isian data dari setiap menu yang dibuka
3. Koordinator akan <i>view</i> semua data.	4. Mengambil data dari <i>database</i> kemudian terjadi eksekusi terhadap perintah yang diberikan dan nantinya akan disimpan di <i>database</i>
<i>Exceptional Flow of Event</i>	Jika aktor salah dalam memasukkan data maka sistem akan menampilkan pesan <i>error</i> . Sedangkan jika data valid maka data akan disimpan di <i>database</i> .

5.3 Class Diagram

5.3.1 Class Diagram

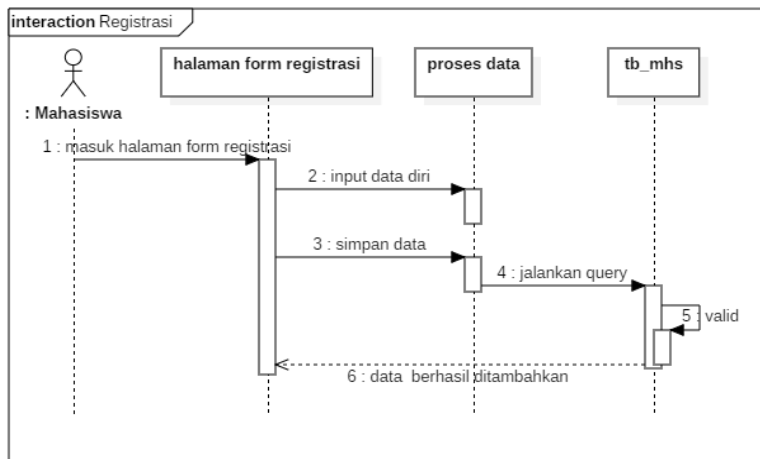


Gambar 5.5 Class Diagram

5.3.2 Sequence Diagram

Sequence diagram digunakan untuk memodelkan pengiriman pesan (*message*) antar *objects*. *Sequence* diagram menjelaskan secara detail urutan proses yang dilakukan dalam system untuk mencapai tujuan dari *use case*, intraksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi dan informasi yang diperlukan oleh masing-masing operasi.

1. Sequence Diagram Registrasi

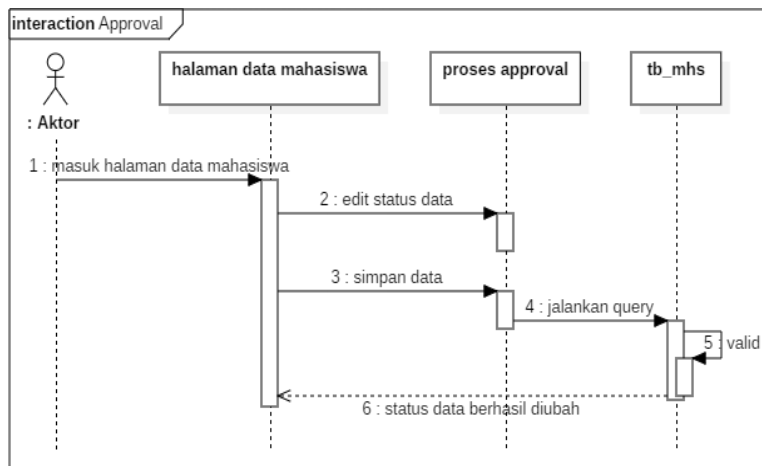


Gambar 5.6 Sequence Diagram Registrasi

Keterangan :

Pada *sequence* diagram ini menjelaskan proses registrasi. Aktor yang melakukan proses tersebut adalah mahasiswa. Pertama mahasiswa masuk pada halaman *form* registrasi lalu meng-input-kan data dirinya. Apabila data diri mahasiswa sudah dilengkapi maka proses selanjutnya adalah menyimpan data dan dilanjutkan dengan proses *OpenTable* ke tabel yang dituju yaitu *tb_mahasiswa*. Selanjutnya sistem akan memvalidasi hasil dari operasi, apakah berhasil atau gagal yang dimunculkan dengan *reply message(alert)*.

2. Sequence Diagram Approval

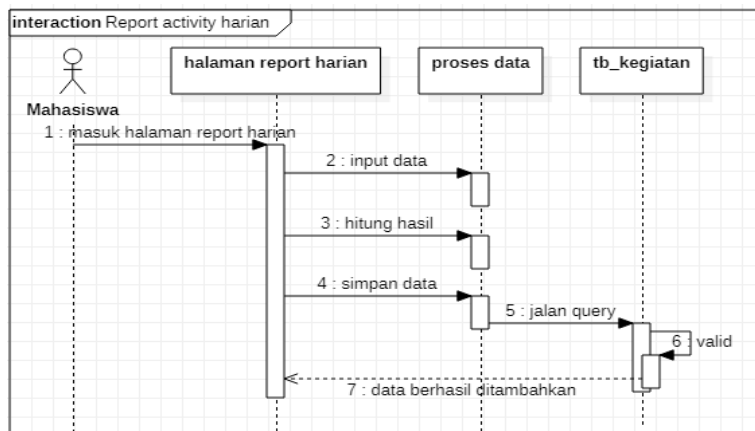


Gambar 5.7 Sequence Diagram Approval

Keterangan :

Pada *sequence* diagram ini menjelaskan proses *approval* yang dilakukan oleh admin dan operator. Proses yang dilakukan adalah menerima data registrasi mahasiswa/i untuk memberi hak akses masuk ke aplikasi. Pertama aktor masuk ke halaman data mahasiswa lalu meng-*edit* status datanya, kemudian melakukan perintah simpan data dan dilanjutkan dengan proses *OpenTable* ke tabel yang dituju yaitu tb_mahasiswa. Selanjutnya sistem akan memvalidasi hasil dari operasi, apakah berhasil atau gagal yang dimunculkan dengan *reply message(alert)*.

3. Sequence Diagram Report Activity Harian

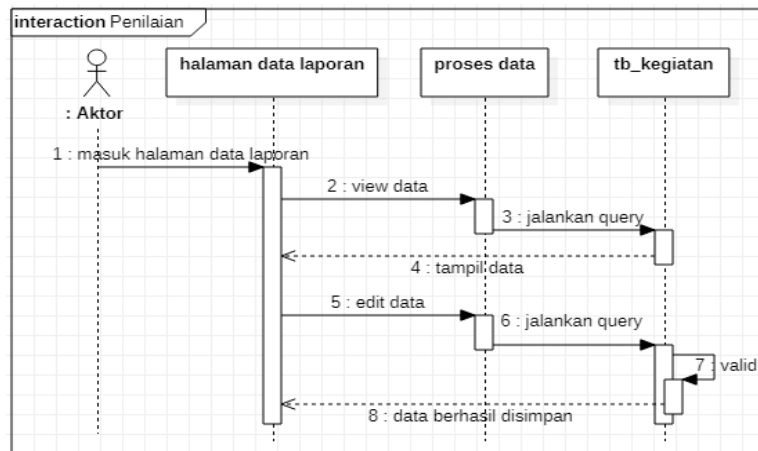


Gambar 5.8 Sequence Diagram Report Activity Harian

Keterangan :

Pada *sequence* diagram ini menjelaskan proses untuk meng-*input* data kegiatan laporan harian yang dilakukan ditempat *internship* dan aktor yang melakukannya adalah mahasiswa. Untuk melakukan proses tersebut mahasiswa masuk ke halaman *report* harian yang sudah tersedia *form input*-an, lalu mengisi data sesuai dengan kegiatan yang dilakukan serta melakukan perhitungan jarak dan proses selanjutnya menyimpan data, kemudian dilanjutkan dengan proses *OpenTable* ke tabel yang dituju yaitu *tb_kegiatan*. Selanjutnya sistem akan memvalidasi hasil dari operasi, apakah berhasil atau gagal yang dimunculkan dengan *reply message(alert)*.

4. Sequence Diagram Penilaian

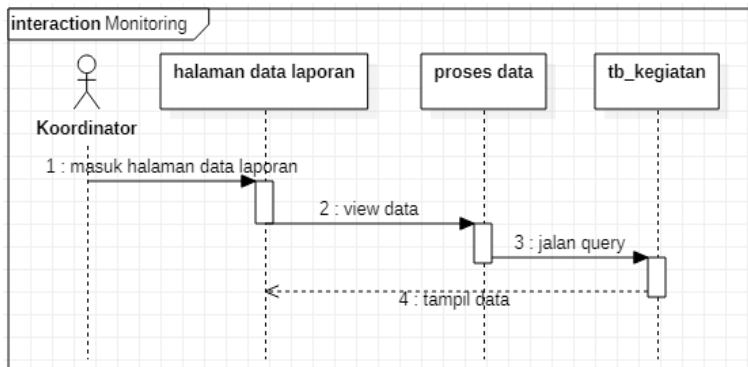


Gambar 5.9 Sequence Diagram Penilaian

Keterangan :

Pada *sequence* diagram ini menjelaskan proses penilaian terhadap kinerja mahasiswa dalam konteks me-monitoring dari data laporan masuk atau kegiatan yang dilakukan yang dituangkan dalam laporan. Dibalik proses tersebut aktor yang melakukannya adalah pembimbing, pertama aktor masuk ke halaman data laporan. Kemudian sistem akan menampilkan semua data laporan masuk, lalu aktor akan melakukan *edit* data untuk memberikan keterangan absensi, status, nilai dan dilanjutkan dengan proses *OpenTable* ke tabel yang dituju yaitu *tb_penilaian*. Selanjutnya sistem akan memvalidasi hasil dari operasi, apakah berhasil atau gagal yang dimunculkan dengan *reply message(alert)*.

5. Sequence Diagram Monitoring



Gambar 5.10 Sequence Diagram Monitoring

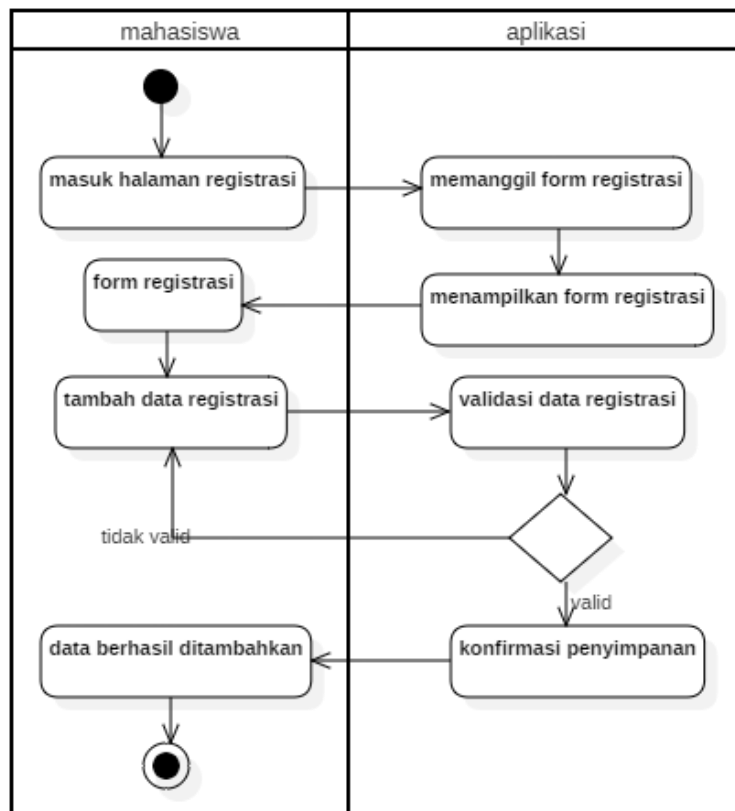
Keterangan :

Pada *sequence* diagram ini menjelaskan proses kelola laporan yang dilakukan oleh aktor koordinator. Pertama aktor masuk ke halaman data laporan dan sistem akan menampilkan semua data laporan serta menampilkan *button* aksi yaitu *view data*. Selanjutnya aktor mengklik *button* tersebut dan dilanjutkan dengan proses *OpenTable* ke tabel yang dituju yaitu *tb_kegiatan*. Selanjutnya sistem akan memvalidasi hasil dari operasi, apakah berhasil atau gagal yang dimunculkan dengan *reply message(alert)*.

5.3.3 Activity Diagram

Activity diagram menggambarkan berbagi aliran aktivitas dalam system yang sudah dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berhasil. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

1. Activity Diagram Registrasi



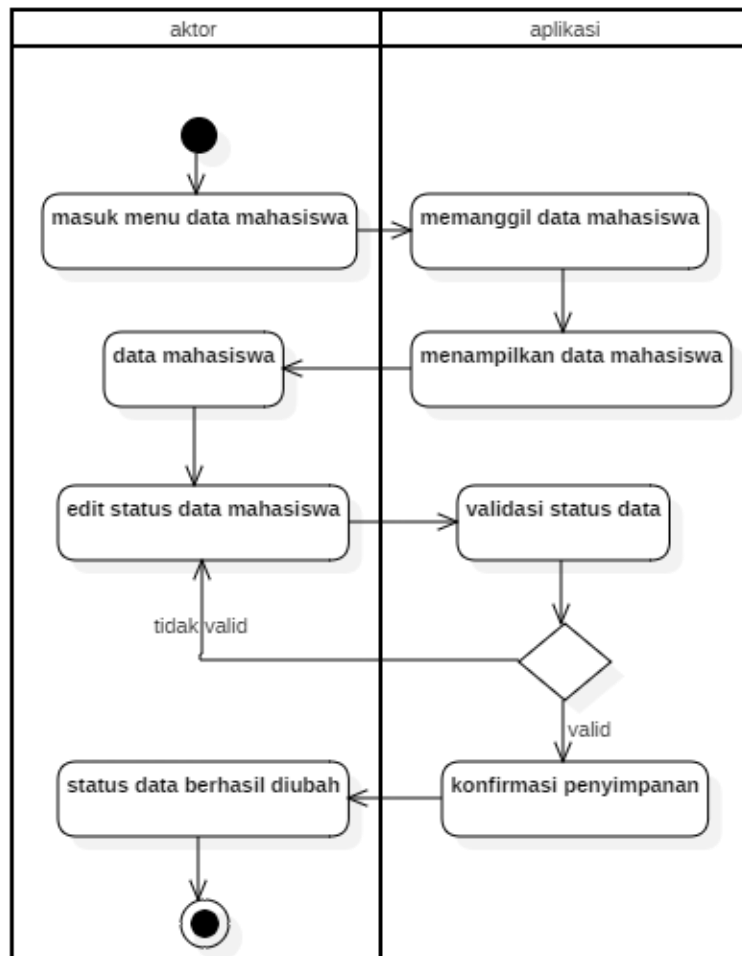
Gambar 5.11 Activity Diagram Registrasi

Keterangan :

Pada *activity* diagram ini menjelaskan proses registrasi yang dilakukan oleh aktor mahasiswa. Mahasiswa masuk ke halaman registrasi lalu sistem menampilkan halaman *form* registrasi ke mahasiswa. Setelah itu mahasiswa dapat melakukan pengolahan data registrasi.

Pengolahan yang pertama yaitu tambah data registrasi yang dimana sistem akan melakukan pengecekan apakah data sudah ditambahkan atau belum ditambahkan, apabila telah ditambahkan maka sistem dapat mengkonfirmasi penyimpanan dan menampilkan *alert*.

2. Activity Diagram Approval



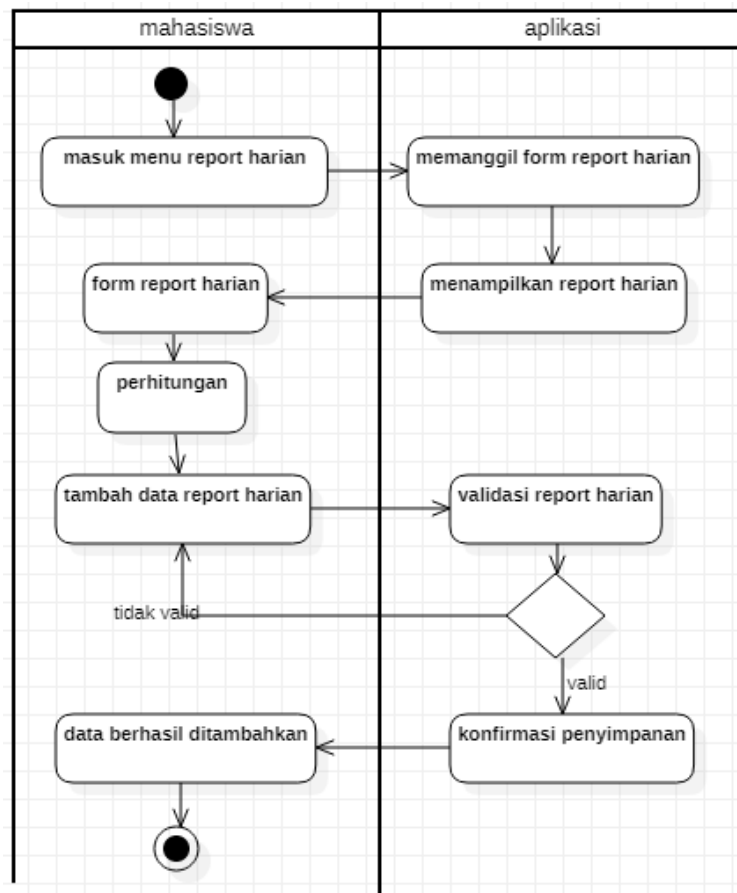
Gambar 5.12 Activity Diagram Approval

Keterangan :

Pada activity diagram ini menjelaskan proses approval yang dilakukan oleh aktor admin dan operator. Aktor masuk menu data mahasiswa lalu sistem menampilkan halaman data mahasiswa ke aktor. Setelah itu aktor dapat pengolahan data mahasiswa.

Pengolahan yang pertama yaitu edit status data mahasiswa yang dimana sistem akan melakukan pengecekan apakah data sudah diubah atau sebaliknya, apabila telah diubah sistem dapat mengkonfirmasi penyimpanan dan menampilkan *alert*.

3. Activity Diagram Report Activity Harian



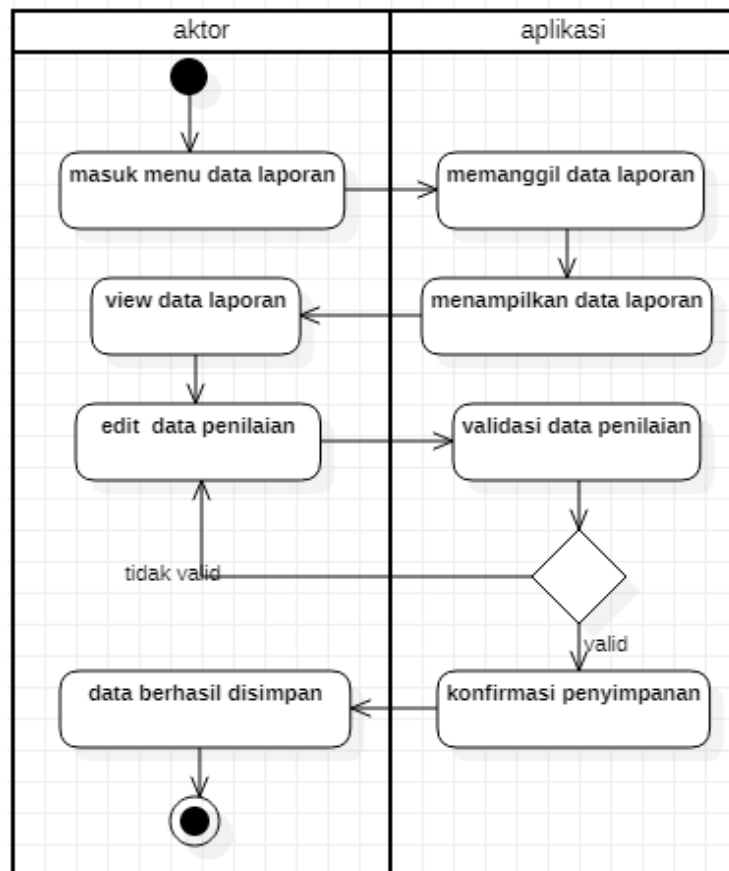
Gambar 5.13 Activity Diagram Report Activity Harian

Keterangan :

Pada *activity* diagram ini menjelaskan proses *report* harian yang dilakukan oleh aktor mahasiswa. Mahasiswa masuk menu *report* harian kemudian sistem menampilkan halaman data laporan ke mahasiswa. Setelah itu mahasiswa dapat melakukan pengolahan data laporan.

Pengolahan yang pertama yaitu tambah data laporan dan perhitungan yang dimana sistem akan melakukan pengecekan apakah data sudah ditambahkan atau belum ditambahkan, apabila telah ditambahkan maka sistem dapat mengkonfirmasi penyimpanan dan menampilkan *alert*.

4. Activity Diagram Penilaian



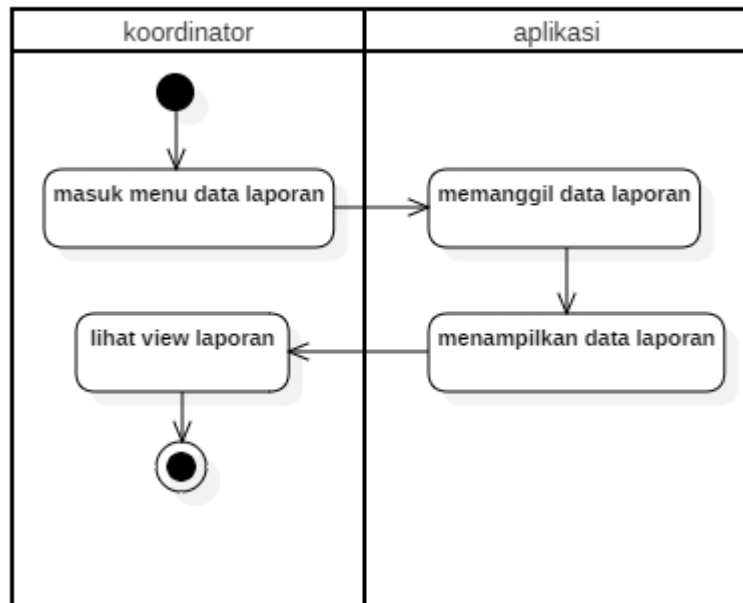
Gambar 5.14 Activity Diagram Penilaian

Keterangan :

Pada *activity* diagram ini menjelaskan proses penilaian yang dilakukan oleh aktor pembimbing. Admin masuk menu data laporan lalu sistem menampilkan halaman data laporan ke admin. Setelah itu admin dapat melakukan pengolahan data laporan. Pengolahan yang pertama yaitu detail data laporan yang dimana sistem akan menampilkan data-data keterangan dari laporan itu sendiri dan pengolahan yang kedua yaitu tambah data penilaian yang dimana sistem akan melakukan pengecekan

apakah data sudah ditambahkan atau belum ditambahkan, apabila telah ditambahkan maka sistem dapat mengkonfirmasi penyimpanan dan menampilkan *alert*.

5. Activity Diagram Monitoring



Gambar 5.15 Activity Diagram Monitoring

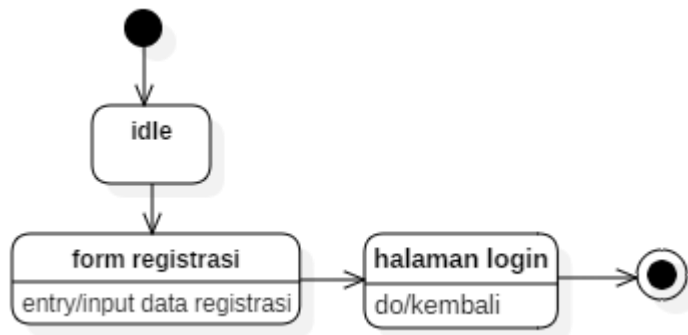
Keterangan :

Pada *activity* diagram ini menjelaskan proses kelola laporan yang dilakukan oleh aktor koordinator. Aktor masuk menu data laporan dan sistem menampilkan halaman data laporan ke aktor. setelah itu aktor dapat melakukan *check*-ing data laporan dengan memberikan aksi pada *button view*, lalu sistem menampilkan keterangan data laporan yang lengkap.

5.3.4 Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan dari suatu *state* ke *state* lainnya. Suatu objek pada *system* sebagai akibat dari stimulasi yang diterima *statechart* diagram mendeskripsikan bagaimana suatu objek mengalami perubahan status adanya *trigger* dan *event-event*. Menunjukkan kondisi yang dapat dialami atau terjadi pada sebuah objek.

1. *Statechart* Diagram Registrasi

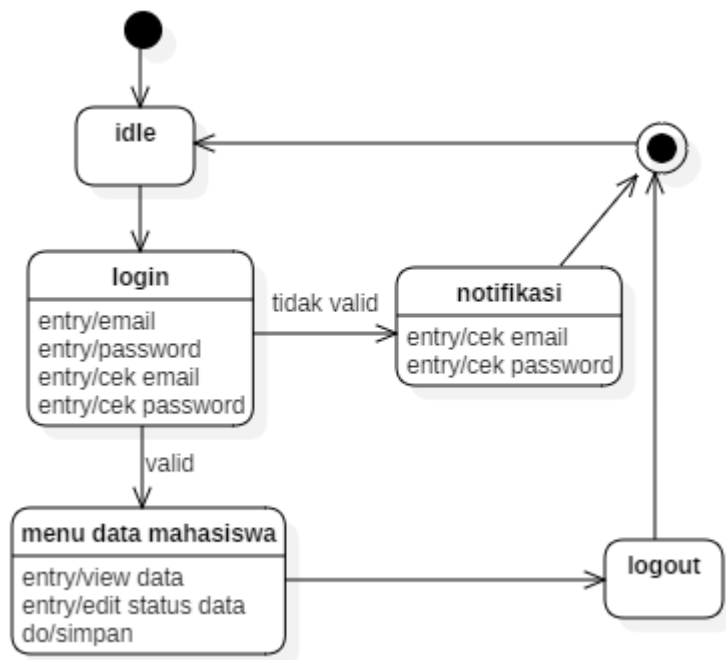


Gambar 5.16 Statechart Diagram Registrasi

Keterangan :

Pada *statechart* diagram ini menjelaskan proses program registrasi yang dilakukan oleh aktor mahasiswa. Aplikasi dalam keadaan *idle* dan status aplikasi berubah saat mahasiswa melakukan aksi yaitu meng-*input* data pada form registrasi. Setelah melakukan operasi tersebut, aktor kembali pada halaman *login*.

2. Statechart Diagram Approval

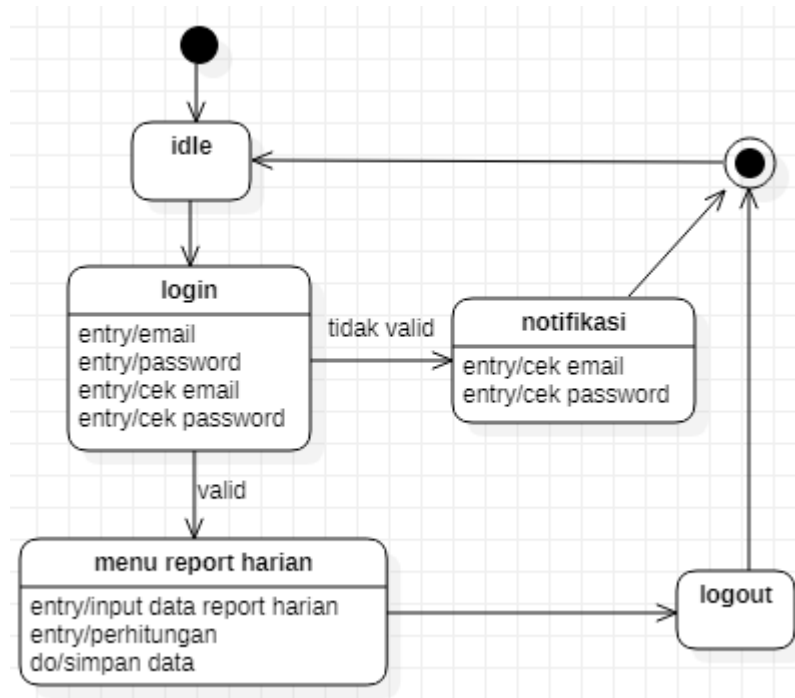


Gambar 5.17 Statechart Diagram Approval

Keterangan :

Pada *statechart* diagram ini menjelaskan proses *approval* data mahasiswa yang dilakukan oleh aktor admin dan operator. Aplikasi dalam keadaan *idle*, kemudian status aplikasi berubah saat aktor melakukan *login*. Apabila login valid maka aktor di alihkan ke halaman utama dan jika *login* tidak valid maka aktor kembali ke halaman *login*. Setelah aktor berhasil melakukan *login* dan masuk ke menu data mahasiswa maka aktor dapat melakukan pengolahan berupa *view*, *edit* dan hapus data mahasiswa. Setelah melakukan pengolahan data mahasiswa maka aktor dapat melakukan *logout*.

3. Statechart Diagram Report Activity Harian

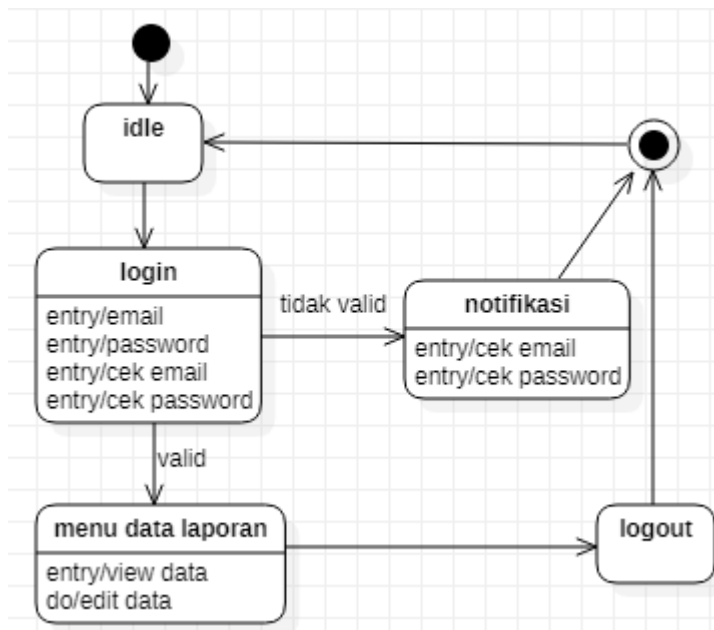


Gambar 5.18 Statechart Diagram Report Activity Harian

Keterangan :

Pada *statechart* diagram ini menjelaskan proses meng-*input* data laporan kegiatan harian yang dilakukan oleh aktor mahasiswa. Aplikasi dalam keadaan *idle* lalu status aplikasi berubah saat mahasiswa melakukan *login*. Apabila *login* valid maka mahasiswa dialihkan ke halaman utama dan apabila *login* tidak valid maka mahasiswa kembali ke halaman login. Setelah mahasiswa berhasil melakukan *login* dan masuk ke halaman utama maka mahasiswa memilih menu *report* harian dan melakukan pengelolaan berupa input, perhitungan dan simpan data *report* harian. Setelah melakukan pengelolaan data *report* harian maka mahasiswa dapat melakukan *logout*.

4. Statechart Diagram Penilaian

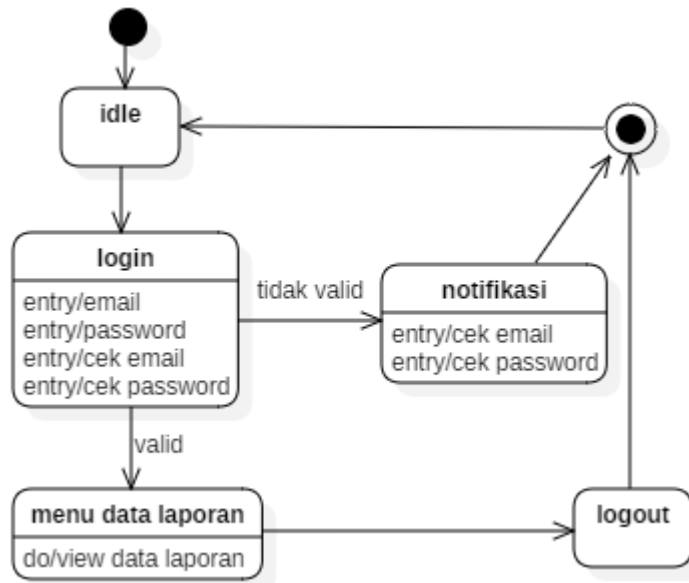


Gambar 5.19 Statechart Diagram Penilaian

Keterangan :

Pada *statechart* diagram ini menjelaskan proses penilaian yang dilakukan oleh aktor pembimbing. Aplikasi dalam keadaan *idle* lalu status aplikasi berubah saat aktor melakukan *login*. Apabila *login* valid maka aktor di alihkan ke halaman utama dan apabila *login* tidak valid maka aktor kembali ke halaman *login*. Setelah aktor berhasil melakukan *login* dan masuk ke halaman utama maka aktor masuk ke menu data laporan serta melakukan pengelolaan berupa tambah, *view* dan *edit* data laporan. Setelah melakukan pengelolaan data laporan maka aktor dapat melakukan *logout*.

5. Statechart Diagram Monitoring

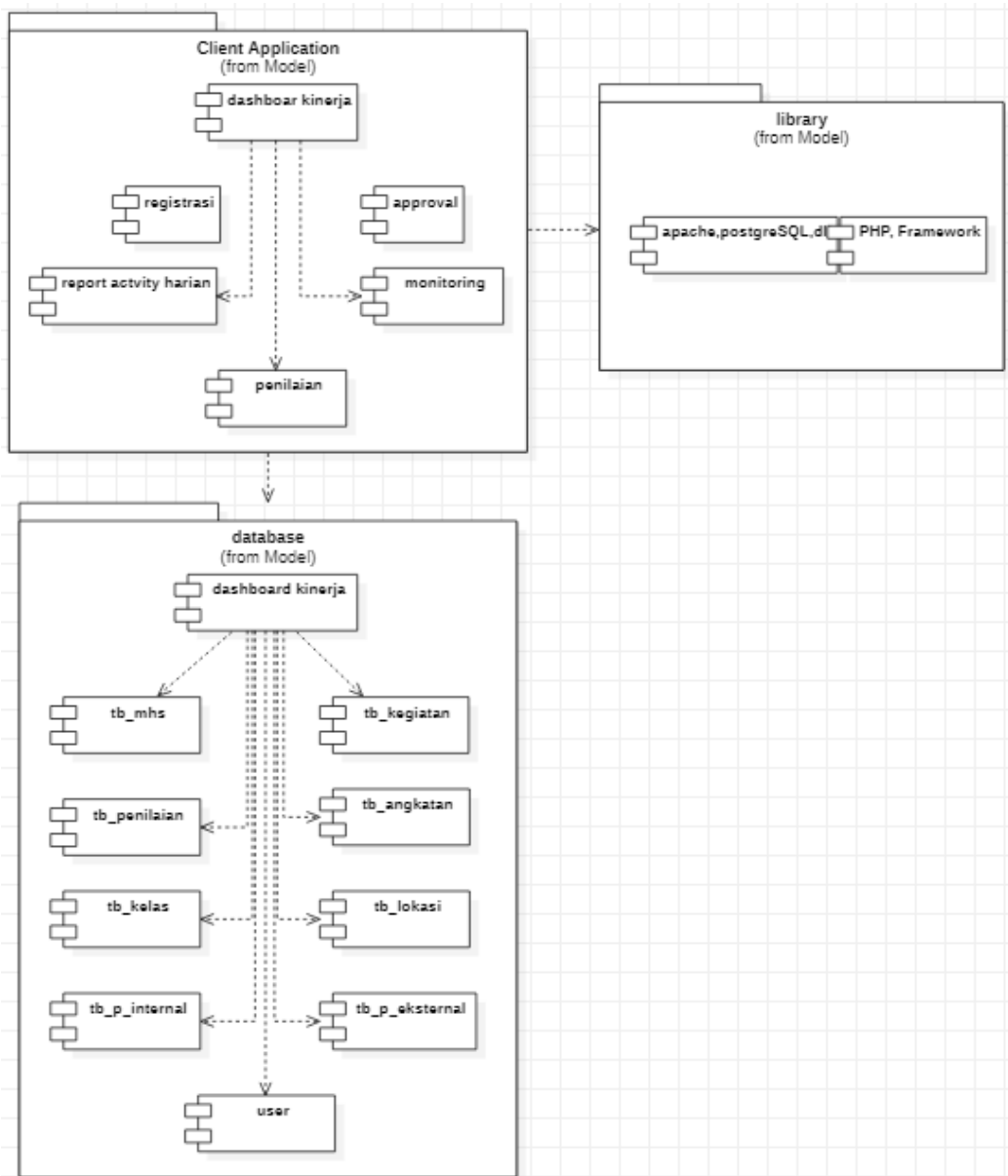


Gambar 5.20 Statechart Diagram Monitoring

Keterangan :

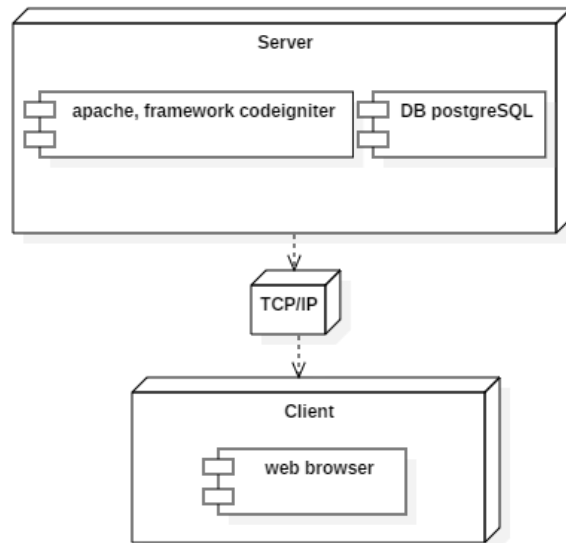
Pada *statechart* diagram ini menjelaskan proses kelola laporan yang dilakukan oleh aktor koordinator. Aplikasi dalam keadaan *idle* lalu status aplikasi berubah saat aktor melakukan *login*. Apabila login valid maka aktor di alihkan ke halaman utama dan apabila *login* tidak valid maka aktor kembali ke halaman *login*. Setelah admin berhasil melakukan *login* dan masuk ke menu data laporan maka aktor dapat melakukan aksi seperti *check-ing* data dengan menekan pada *button* detail dan sistem akan menampilkan data yang lengkap. Setelah melakukan *check-ing* data laporan maka aktor dapat melakukan *logout*.

5.3.5 Component Diagram



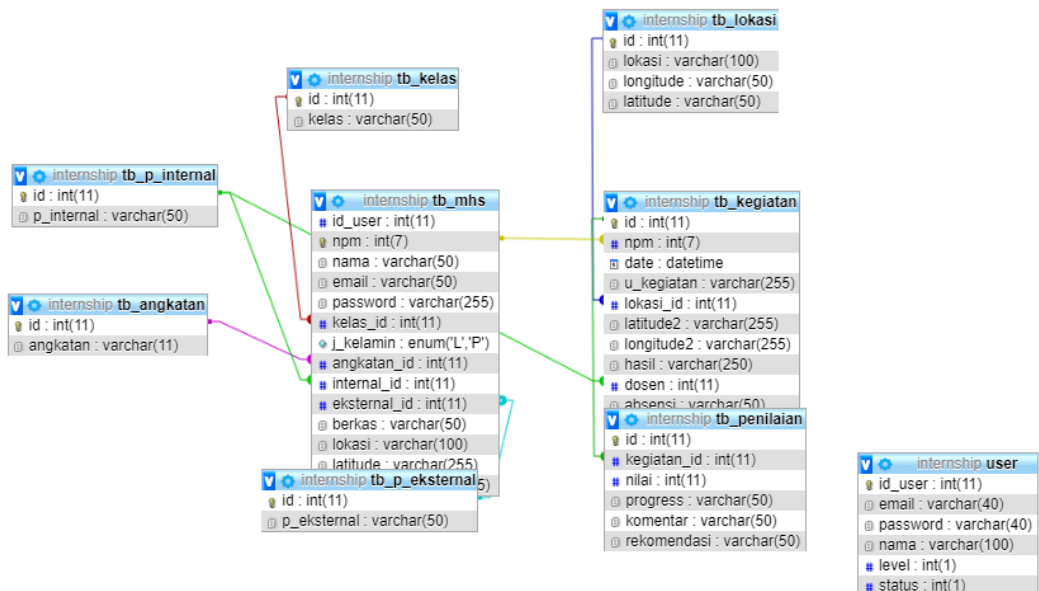
Gambar 5.21 Component Diagram

5.3.6 Deployment Diagram



Gambar 5.22 Deployment Diagram

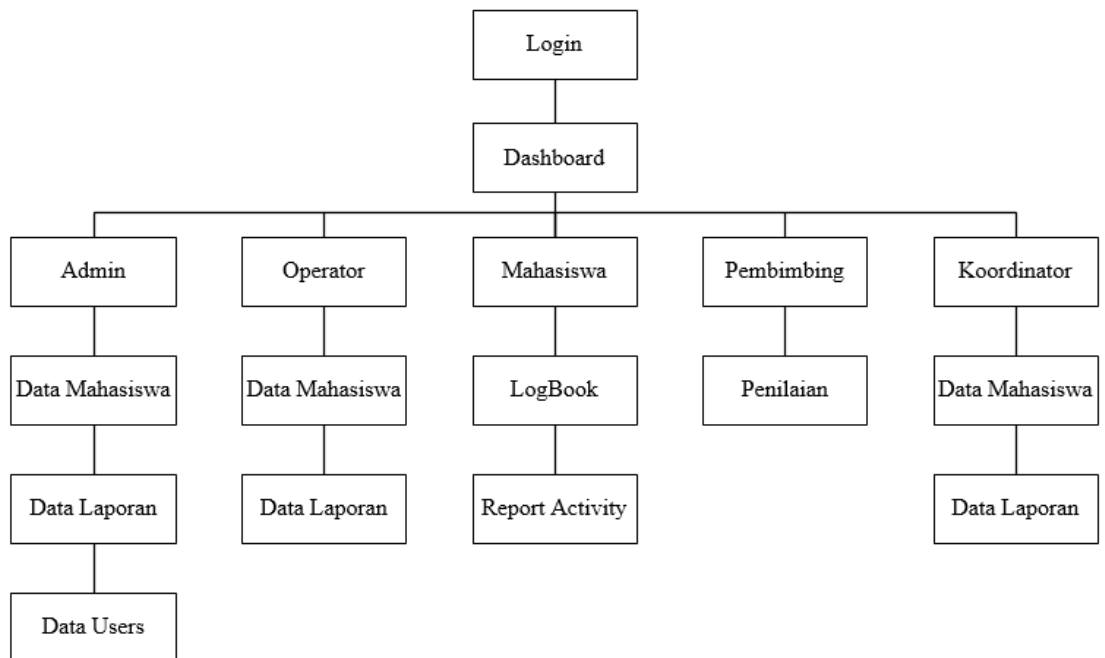
5.3.7 Perancangan Database



Gambar 5.23 Perancangan Database

5.3.8 Struktur Menu

Fungsi-fungsi yang dirancang pada tahap perancangan ini dibagi ke dalam beberapa menu yang bertujuan untuk memudahkan pengoperasian program. Menu yang digunakan pada program saat ini dapat dilihat pada struktur berikut :



Gambar 5.24 Struktur Menu

5.3.9 Perancangan *Interface*

1. Form *Login*




Sign in to start your session

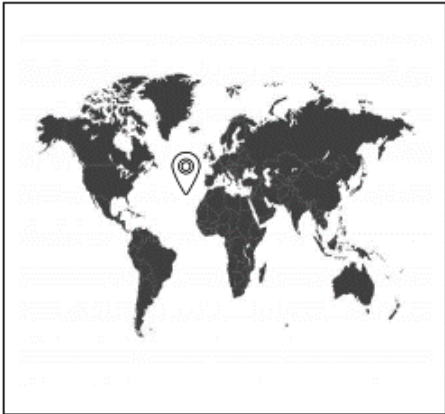
[Register new a membership](#)

Gambar 5.25 Login

2. Form *Registrasi*



REGISTRASI




☐ Laki-laki ☐ perempuan

Gambar 5.26 Form Registrasi

3. Halaman *Approval Data Mahasiswa*

myINTERNSHIP

admin@gmail.com


Admin

Dashboard

Data Mahasiswa

Data Register Internship


No.	Nama Mahasiswa	NPM	Email	Program Studi	Status	Aksi
1.	Aip Suprpto M	1164063	fulan01@gmail.com	D4 Teknik Informatika 4C	1 - Aktif	<div>DetailEditDelete</div>
2.	Fathi Rabbani	1164074	fulan02@gmail.com	D4 Teknik Informatika 4C	1 - aktif	<div>DetailEditDelete</div>

Gambar 5.27 Halaman Approval Data Mahasiswa

4. *Form Report Activity Harian*

myINTERNSHIP

mahasiswa@gmail.com


Fulan 01


Dashboard

Report Harian

Report Harian kegiatan

Uraian Kegiatan

NPM



Posisi kamu disini!

Latitude

Longitude

hasil

Klik hasil

Pilih File

Upload File


Simpan


Reset

Gambar 5.28 Form Report Activity Harian


5. Form Penilaian


my!INTERNSHIP


 pembimbing@gmail.com




Dosen A



 Dashboard

 Data Mahasiswa

 Data Laporan

Data Laporan internship


 > Mahasiswa


No.	Nama Mahasiswa	Kegiatan	Email	Program Studi	File	Aksi
1.	Alip Suprpto M	Internship 1	fulan01@gmail.com	D4 Teknik Informatika 4C	 Kegiatan.docx	<div>EditHapus</div>
2.	Alip Suprpto M	Internship 1	fulan02@gmail.com	D4 Teknik Informatika 4C	 Kegiatan.docx	<div>EditHapus</div>

Gambar 5.29 Form Penilaian


6. Form Monitoring


my!INTERNSHIP


 koordinator@gmail.com




Koordinator



 Dashboard

 Data Mahasiswa

 Data Laporan

Data Laporan internship

 > Mahasiswa

No.	Nama Mahasiswa	Kegiatan	Pembimbing	Absensi	File	Aksi
1.	Alip Suprpto M	Internship 1	fulan01@gmail.com	Hadir	 Kegiatan.docx	<div>Detail</div>
2.	Alip Suprpto M	Internship 1	fulan02@gmail.com	Hadir	 Kegiatan.docx	<div>Detail</div>

Gambar 5.30 Form Monitoring

5.4 Perancangan Arsitektur Perangkat Lunak dan Perangkat Keras Sistem

Dalam perancangan dashboard monitoring kinerja mahasiswa internship membutuhkan beberapa perangkat lunak yaitu sebagai berikut.

5.4.1 Perangkat Lunak

Perangkat lunak pendukung yang digunakan adalah sebagai berikut :

Tabel 5.8 Perangkat Lunak

No.	Jenis		Keterangan
1.	Sistem Operasi	:	<i>Microsoft Windows 10 Profesional 64-bit</i>
2.	Bahasa Pemrograman	:	<i>PHP dengan Framework CI (Codeigniter)</i>
3.	Database	:	<i>PostgreSQL</i>
4.	Perangkat Lunak	:	<i>Visual Studio Code</i> <i>Microsoft Visio 2013</i> <i>StarUML 3.1.0</i>

Tabel 5.8 Perangkat Lunak

5.4.2 Perangkat Keras

Perangkat keras pendukung yang digunakan adalah sebagai berikut :

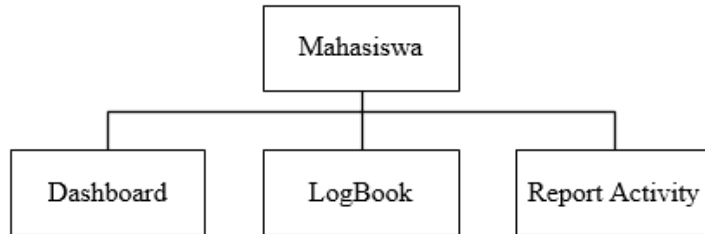
Tabel 5.8 Perangkat Keras

No.	Jenis		Keterangan
1.	<i>Processor</i>	:	Intel® core™i3
2.	<i>Memory</i>	:	4 GB
3.	<i>Monitor</i>	:	LCD 14,1 Inchi
4.	<i>Mouse dan keyboard</i>	:	<i>Standard</i>

Tabel 5.9 Perangkat Keras

5.5 Pemetaan Struktur Diagram User / Aktor Sistem

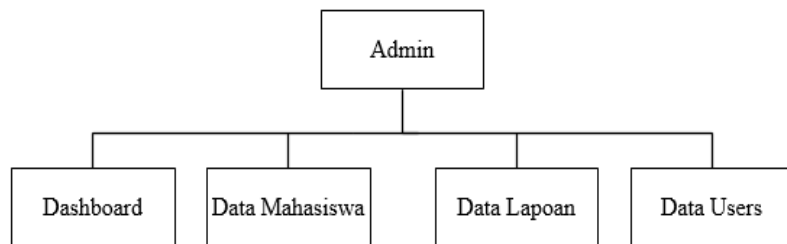
Dibawah ini merupakan pemetaan struktur diagram menu untuk



mahasiswa.

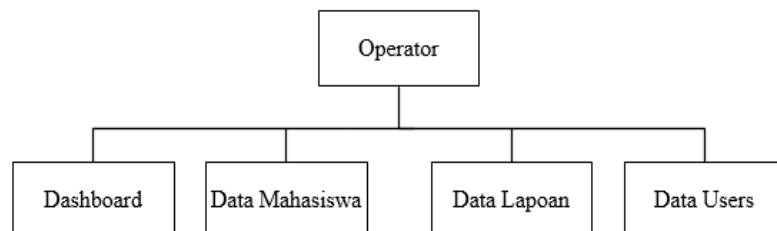
Gambar 5.31 Pemetaan Struktur Diagram Mahasiswa

Dibawah ini merupakan pemetaan struktur diagram menu untuk admin.



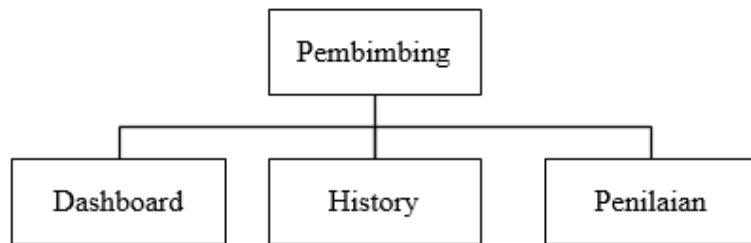
Gambar 5.32 Pemetaan Struktur Diagram Admin

Dibawah ini merupakan pemetaan struktur diagram menu untuk operator.



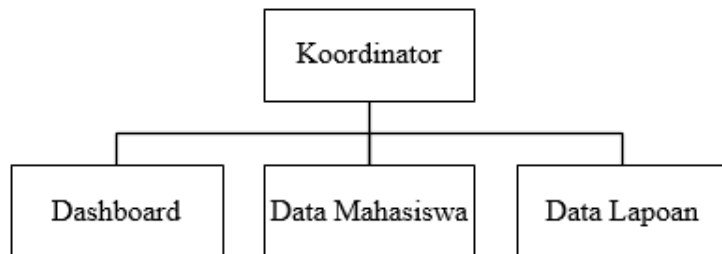
Gambar 5.33 Pemetaan Struktur Diagram Operator

Dibawah ini merupakan pemetaan struktur diagram menu untuk pembimbing.



Gambar 5.34 Pemetaan Struktur Diagram Pembimbing

Dibawah ini merupakan pemetaan struktur diagram menu untuk koordinator



Gambar 5.35 Pemetaan Struktur Diagram Koordinator

6.1 Implementasi Dengan Perhitungan Harvesine Formula

Metode Haversine Formula dapat digunakan untuk menghitung jarak antara dua titik, berdasarkan posisi garis lintang latitude dan posisi garis bujur longitude sebagai variabel inputan. Haversine Formula adalah persamaan penting pada navigasi, memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang.

Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R 6.367, 45 km, dan lokasi dari 2 titik di koordinat bola (lintang dan bujur) masing-masing adalah lon1, lat1, dan lon2, lat2 [25]. Metode Haversine Formula tersebut kini sudah mengalami pengembangan, yaitu dengan menggunakan rumus *spherical law of cosine* sederhana, dimana dengan penghitungan komputer dapat memberikan tingkat presisi yang sangat akurat antar dua titik. Pertama ditentukan terlebih dahulu titik awal dan titik tuju, titik awal berupa latitude1(lat1) dan longitude1(long1), titik tuju berupa latitude2(lat2) dan longitude2(long2). Titik awal dan titik tuju tersebut berbentuk desimal derajat yang kemudian dirubah menjadi nilai sudut radian, kemudian lakukan perhitungan dengan rumus Haversine Formula, yaitu:

Rumus Harvesine

$$x = (\text{lng2}-\text{lng1}) * \cos ((\text{lat1}+\text{lat2})/2);$$

$$y = (\text{lat2}-\text{lat1});$$

$$d = \text{sqrt}(x*x+y*y)*R$$

Keterangan :

x	= longitude (lintang)
y	= latitude (bujur)
d	= jarak (km)
R	= Radius Bumi = 6371 km
1 derajat	= 0.0174532925 radian

6.2 Hasil dan Pembahasan

Sistem informasi monitoring kinerja mahasiswa *internship* menerapkan metode formula harvesine dan dikembangkan berbasis *website* agar memudahkan pengguna mengakses sistem. Sistem ini menggunakan 8 data jumlah mahasiswa/i yang *internship* di prodi DIV Teknik Informatika Politeknik Pos Indonesia.

Pencarian lokasi pengguna yang memanfaatkan *Global Positioning System Geolocation* dari *GoogleMaps*. Langkah selanjutnya adalah mencari dari titik

koordinat mahasiswa registrasi sebagai titik koordinat acuan dan titik koordinat setiap melakukan *report* harian yang telah diinput menggunakan formula haversine, sebagaimana jarak yang didapatkan dari hasil perhitungan kedua titik koordinat tersebut yang ditampilkan oleh sistem dengan visualisasi *alert*.

Google Maps API dimanfaatkan untuk menampilkan peta digital beserta markers dari titik koordinat acuan ke titik koordinat mahasiswa. Dalam perhitungan jarak, sistem menggunakan koordinat *default* yaitu prodi DIV Teknik Informatika, Politeknik Pos Indonesia dan dapat menggunakan koordinat pengguna dengan memanfaatkan fungsi Geolocation. Setelah mendapatkan koordinat pengguna, kemudian sistem mulai menghitung jarak menggunakan formula haversine. Berikut contoh analisis cara kerja metode haversine formula dalam perhitungan jarak antara dua titik :

1. Koordinat Patokan (acuan)

lat1, lng1
-6.873776, 107.575639

Gambar 6.1 Koordinat Acuan

Keterangan :

Data diatas diambil dari koordinat data registrasi untuk sebagai titik acuan pada saat menggunakan perhitungan formula harvesine. Penulis mengambil satu sample data lat1, lng1 yaitu -6.873776, 107.575639.

2. Koordinat *Report* Harian

lat2, lng2
-6.873529, 107.576098
-6.873770, 107.576701
-6.873529, 107.576098
-6.873333, 107.576141
-6.873399, 107.575530
-6.873399, 107.575530
-6.873399, 107.575530
-6.873383, 107.575380

Gambar 6.2 Report Harian

Keterangan :

Data diatas mengambil dari data koordinat kegiatan pada saat mahasiswa melakauk *report* harian. Selanjutnya penulis akan mempraktekan salah satu data diatas, contoh data lat2, lng2 yang digaris merahkan (-6.873529, 107.576098).

Percobaan 1

1. Titik koordinat pertama

(Prodi DIV Teknik Informatika, Politeknik Pos Indonesia)

$$\text{Lat 1} = -6.873776 \times 0.0174532925$$

$$= -0.1199700231 \text{ Radian}$$

$$\text{Lng 1} = 107.575639 \times 0.0174532925$$

$$= 1.87754909334 \text{ Radian}$$

2. Titik koordinat kedua

(mahasiswa *report* harian.Auditorium)

$$\text{Lat 2} = -6.873529 \times 0.0174532925$$

$$= -0.11996571214 \text{ Radian}$$

$$\text{Lng 2} = 107.576098 \times 0.0174532925$$

$$= 1.8775571044 \text{ Radian}$$

$$\begin{aligned} 3. \quad x &= (\text{lng2}-\text{lng1}) * \cos ((\text{lat1}+\text{lat2})/2) \\ &= (1.8775571044-1.87754909334) * \cos ((-0.1199700231 + \\ &\quad 0.11996571214)/2) \\ &= 0.00000795348 \end{aligned}$$

$$\begin{aligned} 4. \quad y &= (\text{lat2}-\text{lat1}) \\ &= (-0.11996571214-(-0.1199700231)) \\ &= 0.00000431096 \text{ d} \\ &= \text{sqrt} (x*x+y*y)*R \end{aligned}$$

$$= \sqrt{((0.00000795348 \times 0.00000795348) + (0.00000431096 \times 0.00000431096)) \times 6371}$$

$$= \sqrt{(0.00000000008184222) \times 6371}$$

$$= 0.00000904666900024534 \times 6371$$

$$= 0.0576363282 \text{ KM to Meter}$$

$$= 0.0576363282 \times 1000$$

$$= 57.6363282006 \text{ Meter.}$$

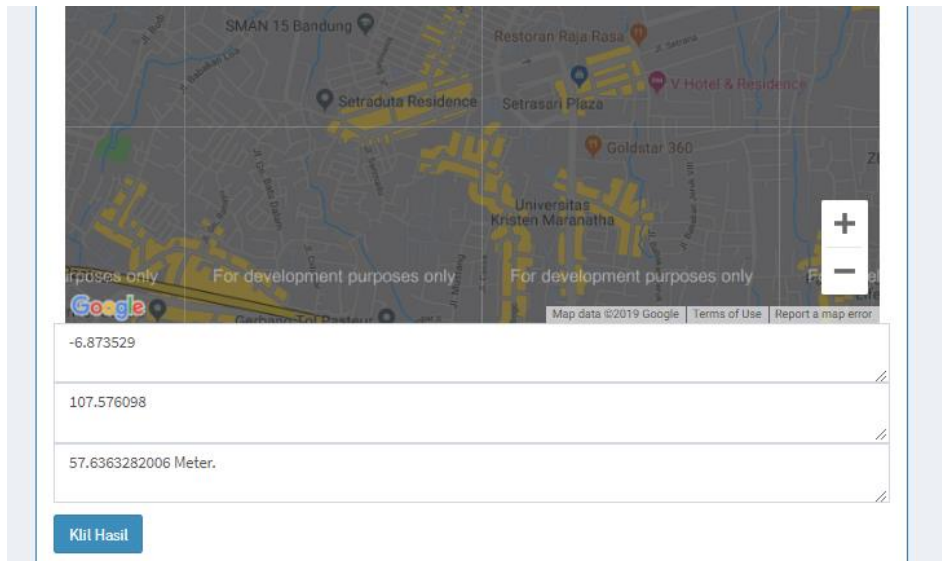
5. Implementasi *source code* dan di jalankan di *browser*

```
function haversine($id)
{
    $explodeVal = explode('~', $id);
    // dari prodi
    $lat1 = -6.873776;
    $lon1 = 107.575639;
    // target
    $lat2 = $explodeVal[0];
    $lon2 = $explodeVal[1];
    // distance between latitudes
    // and longitudes
    $dLat = ($lat2 - $lat1) *
        M_PI / 180.0;
    $dLon = ($lon2 - $lon1) *
        M_PI / 180.0;

    // convert to radians
    $lat1 = ($lat1) * M_PI / 180.0;
    $lat2 = ($lat2) * M_PI / 180.0;

    // apply formulae
    $a = pow(sin($dLat / 2), 2) +
        pow(sin($dLon / 2), 2) *
        cos($lat1) * cos($lat2);
    $rad = 6371;
    $c = 2 * asin(sqrt($a));
    $SimpanRad = $rad * $c;
    // Driver code
    $penjumlahan = $SimpanRad*1000;
    $data['jsonarray'] = $penjumlahan.' METER';
    echo json_encode($data);
}
```

Gambar 6.3 Implementasi Source Code



Gambar 6.4 implementansi browser

Percobaan 2

6. Titik koordinat pertama

(kamar 1)

$$\begin{aligned}\text{Lat } 1 &= -6.8741201 \times 0.0174532925 \\ &= -0.11997602878 \text{ Radian}\end{aligned}$$

$$\begin{aligned}\text{Lng } 1 &= 107.57703269999999 \times 0.0174532925 \\ &= 1.877573418 \text{ Radian}\end{aligned}$$

7. Titik koordinat kedua

(kamar 5)

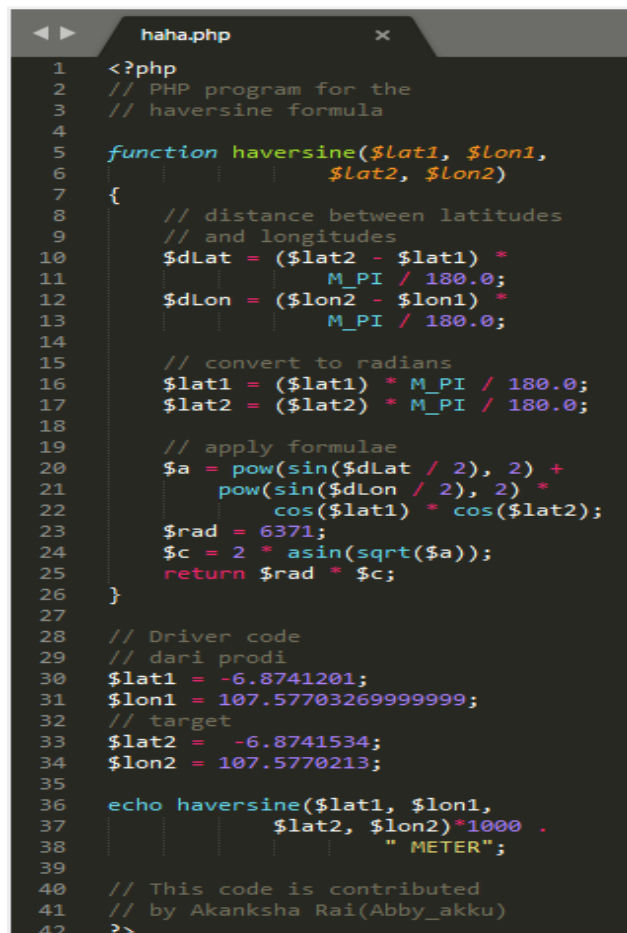
$$\begin{aligned}\text{Lat } 2 &= -6.8741534 \times 0.0174532925 \\ &= -0.11997660998 \text{ Radian}\end{aligned}$$

$$\begin{aligned}\text{Lng } 2 &= 107.5770213 \times 0.0174532925 \\ &= 1.87757321903 \text{ Radian}\end{aligned}$$

$$\begin{aligned}8. \quad x &= (\text{lng2} - \text{lng1}) * \cos ((\text{lat1} + \text{lat2})/2) \\ &= (1.87757321903 - 1.877573418) * \cos ((-0.11997602878 + -0.11997660998)/2) \\ &= 0.00000795348\end{aligned}$$

$$\begin{aligned}
9. \quad y &= (\text{lat2} - \text{lat1}) \\
&= (-0.11997660998 - (-0.11997602878)) \\
&= 0.0000006242096 \text{ d} \\
&= \sqrt{x^2 + y^2} * R \\
&= \sqrt{((0.000002395348 \times 0.00000435348) + (0.000003131096 \times 0.00000431096))} \times 6371 \\
&= \sqrt{(0.000000000081876422)} \times 6371 \\
&= 0.00006897975669024534 \times 6371 \\
&= 0.039108034268386 \text{ KM to Meter} \\
&= 0.03910803488 \times 1000 \\
&= 3.91080348 \text{ Meter.}
\end{aligned}$$

10. Implementasi *source code* dan dijalankan dibrowser

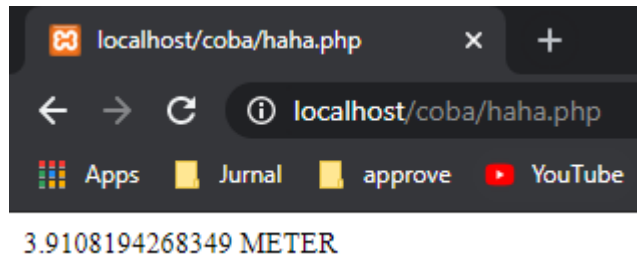


```

1  <?php
2  // PHP program for the
3  // haversine formula
4
5  function haversine($lat1, $lon1,
6                    $lat2, $lon2)
7  {
8      // distance between latitudes
9      // and longitudes
10     $dLat = ($lat2 - $lat1) *
11             M_PI / 180.0;
12     $dLon = ($lon2 - $lon1) *
13             M_PI / 180.0;
14
15     // convert to radians
16     $lat1 = ($lat1) * M_PI / 180.0;
17     $lat2 = ($lat2) * M_PI / 180.0;
18
19     // apply formulae
20     $a = pow(sin($dLat / 2), 2) +
21          pow(sin($dLon / 2), 2) *
22          cos($lat1) * cos($lat2);
23     $rad = 6371;
24     $c = 2 * asin(sqrt($a));
25     return $rad * $c;
26 }
27
28 // Driver code
29 // dari prodi
30 $lat1 = -6.8741201;
31 $lon1 = 107.57703269999999;
32 // target
33 $lat2 = -6.8741534;
34 $lon2 = 107.5770213;
35
36 echo haversine($lat1, $lon1,
37               $lat2, $lon2)*1000 .
38               " METER";
39
40 // This code is contributed
41 // by Akanksha Rai(Abby_akku)
42 ?>

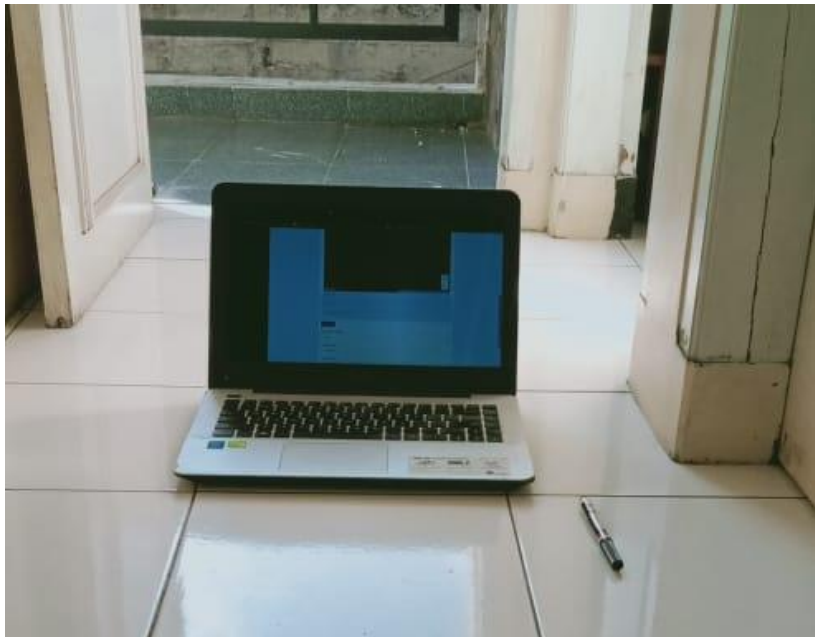
```

Gambar 6.5 implementasi source code 2



Gambar 6.6 Hasil Pada Browser

11. Perhitungan manual dengan alat meter gulung



Gambar 6.7 Koordinat latitude, longitude 1



Gambar 6.8 Koordinat latitude, longitude 1



Gambar 6.9 Koordinat latitude, longitude 2



Gambar 6.10 Koordinat latitude, longitude 2



Gambar 6.11 Hasil Jarak

Hasil akhir dari koordinat latitude, longitude 1 dan latitude, longitude 2 adalah sebesar 391 cm, apabila di rubah ke meter menjadi 3,91 Meter.

12. Data koordinat hasil dari pengujian sistem

Data-data koordinat hasil perhitungan metode harvesine formula dengan sistem yang dibangun dan nama lokasi yang diambil oleh peneliti adalah parkir belakang gedung pendidikan :

	Perpindahan	Arah Mata Angin				
No.	Jarak	Utara	Selatan	Barat	Timur	Range
	0					
1.	2 Meter	0	13 Meter	14 Meter	14 Meter	14 Meter
2.	4 Meter	3 Meter	13 Meter	6 Meter	6 Meter	10 Meter
3.	6 Meter	13 Meter	15 Meter	6 Meter	114 Meter	108 Meter
4.	8 Meter	13 Meter	8 Meter	8 Meter	6 Meter	7 Meter
5.	10 Meter	15 Meter	11 Meter	12 Meter	12 Meter	4 Meter
6.	12 Meter	66 Meter	8 Meter	14 Meter	14 Meter	58 Meter
7.	14 Meter	11 Meter	14 Meter	13 Meter	13 Meter	3 Meter
8.	16 Meter	14 Meter	10 Meter	19 Meter	18 Meter	9 Meter
9.	18 Meter	14 Meter	71 Meter	8 Meter	7 Meter	64 Meter
10.	20 Meter	14 Meter	57 Meter	59 Meter	3 Meter	56 Meter
	Rata-rata					33,3

Gambar 6.12 Hasil Pengujian Sistem

Keterangan :

Pada gambar 6.12 adalah hasil pengujian dari sistem yang dilakukan oleh peneliti dengan mengambil lokasi di parkir belakang gedung pendidikan serta melakukan pergerakan dari 0 sampai 20 meter yang menghasilkan rata-rata 33,3 dari penjumlahan *range* setiap mata arah angin. Hasil dari data diatas maka peneliti menambahkan rumus angka pada sistem (33,3) agar perhitungan jarak lebih mendekati akurat.

```
// Driver code
// dari acuan
$lat1 = -6.8734373;
$lon1 = 107.57479509999999;
// target
$lat2 = -6.8734351;
$lon2 = 107.5754055;

echo haversine($lat1, $lon1,
               $lat2, $lon2)*33,3 .
               "METER";

?>
```

Gambar 6.13 Source Code Sistem

Apabila dilakukan perhitungan maka hasil dari sistem yang dibangun adalah sebagai berikut :

	Perpindahan	Arah Mata Angin			
No.	Jarak	Utara	Selatan	Barat	Timur
	0				
1.	2 Meter	0	4 Meter	4 Meter	4 Meter
2.	4 Meter	1 Meter	4 Meter	2 Meter	2 Meter
3.	6 Meter	4 Meter	5 Meter	2 Meter	38 Meter
4.	8 Meter	5 Meter	2 Meter	2 Meter	2 Meter
5.	10 Meter	5 Meter	3 Meter	4 Meter	4 Meter
6.	12 Meter	2 Meter	3 Meter	4 Meter	4 Meter
7.	14 Meter	3 Meter	4 Meter	4 Meter	4 Meter
8.	16 Meter	3 Meter	4 Meter	6 Meter	6 Meter
9.	18 Meter	4 Meter	23 Meter	2 Meter	2 Meter
10.	20 Meter	4 Meter	19 Meter	19 Meter	1 Meter

Gambar 6.14 Hasil Pengujian Sistem

6.3 Evaluasi Metode

Hasil dari perhitungan yang dilakukan oleh peneliti secara manual serta diperbandingkan dengan perhitungan sistem hasil yang diperoleh sama, jadi untuk diterapkan pada sistem yang dibangun sangat membantu untuk menghitung jarak mahasiswa/i pada pergerakan saat melakukan laporan kegiatan

