# BAB 2
# LANDASAN TEORI

## 2.1. Profil CV Rumah Robot Indonesia (Robonesia)

CV Rumah Robot Indonesia (Robonesia) adalah sebuah *startup* yang menyediakan media pembelajaran robotika berupa ekstrakulikuler, privat, dan bimbingan belajar robotika yang tersebar di lebih dari 40 sekolah di Kota Bandung dan lebih dari 1200 siswa di Jawa Barat pada tingkat sekolah dasar hingga sekolah menengah atas. Robonesia merupakan startup dibawah naungan PT. Kolaborasi Kapital Indonesia yang berkantor di Jl. Kampung Padi No. B11, Kota Bandung, Jawa Barat, Indonesia.

### 2.1.1. Visi dan Misi

Berikut adalah penjelasan visi dan misi dari CV Rumah Robot Indonesia:

**Visi**

Menjadi pelopor pendidikan Robotika dan teknologi muslim untuk peradaban islam masa depan.

**Misi**

1. Menyiapkan kurikulum robotika professional bernafaskan peradaban islam.
2. Menyelenggarakan pendidikan robotika berbasis project Real sosial kemasyarakatan.
3. Mengembangkan sarana dan prasarana robotika untuk penguatan generasi muslim.

### 2.1.2. Struktur Organisasi

Berikut gambar  adalah struktur organisasi CV Rumah Robot Indonesia (Robonesia) dapat dilihat pada Gambar 2.1 Struktur Organisasi CV Rumah Robot Indonesia:


**Gambar 0.1 Struktur Organisasi CV Rumah Robot Indonesia.**

Struktur dari CV Rumah Robot Indonesia terdiri atas beberapa divisi diantaranya adalah Divisi *Product Development*, Divisi *Finance* & *Administration*, Divisi *Human Resource Development*, Divisi *Accounting*, Divisi *Public Relation*, Divisi *Marketing*, Divisi *Manager Technic Officer*, dan Divisi *Supply Chain*.

Adapun struktur organisasinya berbentuk lingkaran, hal tersebut mengambarkan jalannya koordinasi yang sejajar dan saling berkaitan antar satu divisi dan divisi lainnya. Sementara untuk pengambilan keputusan setiap divisi wajib disampaikan dan harus melalui persetujuan *General Manager*. Untuk lebih lengkapnya dapat dilihat pada Gambar 2.1 Struktur Organisasi CV Rumah Robot Indonesia.

### 2.2.Landasan Teori

Landasan teori menjelaskan beberapa teori yang berkaitan dengan pembangunan aplikasi sebagai dasar pemahaman yang dipakai terhadap aplikasi tersebut yang telah disusun secara sistematis dengan dasar yang kuat. Dalam sebuah penelitian, landasan teori bertujuan untuk memberikan gambaran dari teori yang terkait dalam penelitian tersebut.

### 2.2.1. Reminder

*Reminder* atau dalam bahasa indonesia adalah pengingat menurut kamus besar bahasa Indonesia pengingat merupakan orang, alat, atau benda yang digunakan untuk mengingatkan hal-hal penting. Menurut Valerie Tripp, seorang penulis buku anak-anak yang memiliki gelar Magister Pendidikan dari Universitas Harvard menyebutkan bahwa Reminder adalah sesuatu yang memanggil ingatan atau pikiran [8]. Pengingat juga berarti sesuatu yang mengingatkan ataupun alat elektronik yang berbunyi atau memberi tanda secara otomatis untuk memberi peringatan tentang waktu, acara dan sebagainya.

Tanpa pengingat atau *reminder*, manusia sering mengalami lupa akan suatu kegiatan dan apa saja yang telah dilakukan maupun kegiatan yang akan dilakukan di waktu yang akan datang. Pada kenyataannya, tidak sedikit dari masyarakat yang sudah merencanakan ekgiatan untuk dilakukan baik sendiri maupun pihak tertentu pada akhirnya lupa untuk melakukan kegiatan tersebut, ataupun ingat saat waktu yang sudah mendesak dan mengalami keterlambatan. Untuk mengatasi itu, tiap individu memiliki agenda dimana agenda merupakan daftar catatan yang berisi kegiatan dari seseorang. Rata-rata agenda yang dimiliki masyarakat adalah agenda berupa buku. Tidak sedikit orang yang pada dasarnya membutuhkan agenda, tetapi malas untuk membuatnya karena orang tersebut harus direpotkan untuk membawa buku dan alat tulis untuk mencatatnya. Namun, seiring berkembangnya teknologi, agenda biasanya dimuat pada media kertas dapat dibuat kedalam media elektronik seperti *smarthphone.* Di *smartphone,* biasanya seseorang akan menuliskan kegiatan pada agendanya lalu mengatur waktu untuk diingatkan kembali saat waktu kegiatan tersebut akan dilakukan menggunakan *alarm* ataupun *reminder.*

Reminder atau pengingat biasanya berkaitan erat dengan *alarm* dan janji. *Alarm* pada umumnya untuk memberi peringatan kepada pengguna bahwa ada suatu kegiatan atau ada sesuatu yang harus dilakukan pada waktu yang telah ditentukan sebelum *alarm* berbunyi. Biasanya sebelum mengatur pengingat menggunakan *reminder* yang direncanakan dan dirancang dulu sebuah jadwal. Pengertian jadwal menurut kamus besar bahasa Indonesia adalah pembagian waktu berdasarkan rencana pengaturan urutan kerja, daftar atau sistem kegiatan atau rencana kegiatan dengan pembagian waktu pelaksanaan yang terperinci. Sedangkan

pengertian penjadwalan adalah proses, cara, perbuatan menjadwalkan atau memasukkan ke dalam jadwal. Cara kerja sistem reminder adalah server selalu mencocokkan jam yang sudah disimpan di database dengan jam pada sistem operasi server.

Perbedaan antara reminder dengan alarm adalah terletak pada *memo*. *Reminder* bisa mencantumkan *memo* atau catatan sekaligus pengingat. Sedangkan untuk alarm hanya digunakan sebagai pengingat waktu. *Reminder* biasanya digunakan sebagai pencatat janji, jadwal keseharian, tugas, maupun pengajaran. Untuk *alarm* pada umumnya digunakan sebagai alat bantu untuk membangunkan seseorang jika ingin melakukan kegiatan pada waktu yang telah ditentukan. Seiring berkembanganya zaman, *reminder* ataupun alarm dapat ditemui pada aplikasi *smartphone*.

Dalam sebuah aplikasi, aplikasi *reminder* atau pengingat adalah sebuah aplikasi yang dirancang untuk pengguna agar membantu pengguna dalam mengingatkan suatu kegiatan ataupun rencana yang telah dijadwalkan sebelumnya. *Reminder* dapat lebih bermanfaat ketika informasi kontektual digunakan untuk menyajikan informasi pada waktu yang tepat dan tempat yang tepat. Selain itu, reminder dapat digunakan dalam manajemen waktu yang berfungsi untuk memberi *alarm* peringatan berupa pemberitahuan berbasis lokasi, berupa pemberitahuan berbasis lokasi, waktu maupun catatan yang bersifat kontekstual [9].

*Reminder* yang dimaksud dalam penelitian ini adalah *reminder* atau pengingat yang menginformasikan kepada pengajar di CV Rumah Robot Indonesia pada waktu yang tepat dan tempat yang tepat agar segera bergegas dan pergi ke tempat pengajaran yang telah ditentukan. Dengan menggunakan pendeteksi kemacetan dan merekomendasikan rute tercepat serta menginformasikan cuaca kepada pengajar. *Reminder* atau pengingat dapat membuat para pengajar untuk datang tepat waktu ke lokasi-lokasi yang telah ditentukan.

### 2.2.2. Aplikasi

Aplikasi, program aplikasi atau perangkat lunak aplikasi adalah program komputer yang dirancang untuk membantu manusia melakukan suatu kegiatan. Bergantung pada aktivitas yang dirancang, aplikasi dapat memanipulasi teks,

angka, audio, grafik, dan kombinasi elemen-elemen ini. Beberapa paket aplikasi fokus pada satu tugas, seperti pengolah kata [10].

Aplikasi dapat di *bundle* dengan komputer dan sistem perangkat lunaknya atau diterbitkan secara terpisah. Aplikasi dapat dibangun sebagai proyek pribadi ataupun *open source*. Aplikasi yang dibangun untuk platform seluler disebut sebagai aplikasi *mobile*.

### 2.2.3. Android

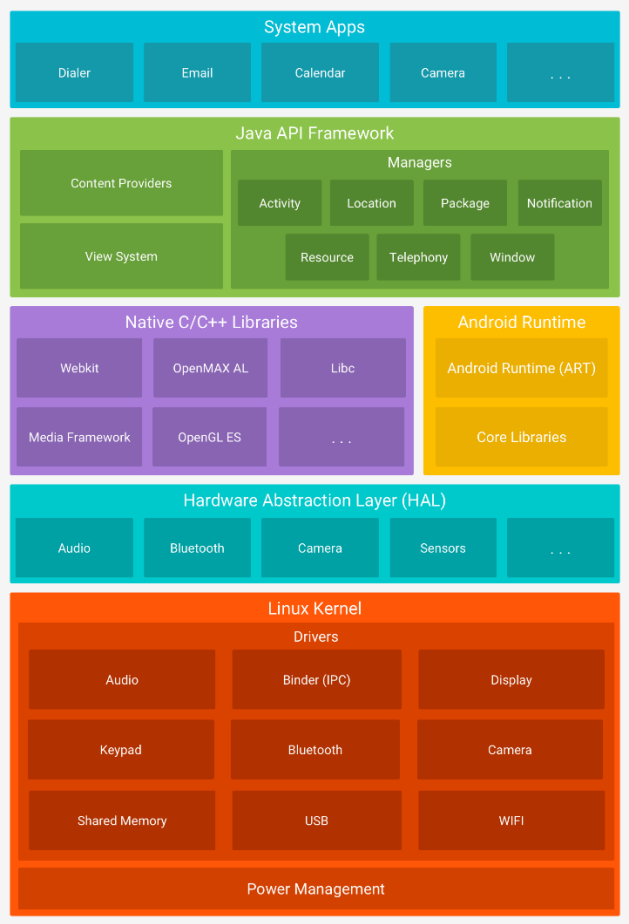
Android adalah sistem operasi untuk perangkat mobile berbasis *Linux* yang awalnya dikembangkan oleh *Android Inc*. *Android* terdiri dari system operasi, *middlewar* dan aplikasi [11]. Android adalah istilah dalam bahasa Inggris yang berarti robot yang menyerupai manusia. Pada tahun 2005, *Google* secara resmi telah membeli android sehingga pengembangan *Android* sepenuhnya berada di tangan *Google*. Dalam proses pengembangan sistem operasi *Android*, dibentuklah organisasi *Open Handset Alliance*. *Google* merilis software open source untuk *Android*, sehingga dapat berkontribusi untuk mengembangkan *Android* [12].

Berikut beberapa kelebihan Android:

1. Sangat mudah mengoperasikan *Smartphone Android*.
2. Banyak *fitur*.
3. Dari segi tampilan, *Android* tidak kalah bagusnya dengan *OS* lain.
4. Sistem operasi dibuat *opensource* untuk pengembangan.
5. Aplikasi yang gratis hingga berbayar, dan bisa di *download* di *Google Play* [13].

#### 2.2.3.1.Arsitektur *Android*

Arsitektur Android dapat digambarkan seperti pada **Gambar 3 Arsitektur *Android*.** Pada gambar tersebut diperlihatkan bahwa terdapat empat tingkatan atau lapisan dalam arsitektur Android yang dimulai dari ***Linux Kernel*** sebagai lapisan terbawah, diikuti lapisan ***Libraries*** dan ***Android Runtime***, lapisan ***Application Framework***, hingga lapisan ***Applications***. Berikut adalah Gambar 2.2 Arsitektur Android.




*Sumber Gambar :* https://developer.android.com/guide/platform?hl=id

**Gambar 0.2 Arsitektur *Android***

**a. Linux Kernel**

Linux Kernel berada pada lapisan terbawah dari susunan *Android* yang menyediakan suatu perantara bagi perangkat keras (*hardware*) dengan lapisan diatasnya. Berdasarkan pada Linux versi 2.6, Linux Kernel menyediakan *preemptive multitasking* dan layanan level dasar inti sistem, seperti *memory*, *process*, dan *power management*. Selain itu, Kernel tersebut juga menyediakan suatu tingkatan jaringan dan *device drivers* untuk perangkat keras seperti *monitor, Wi-Fi,* dan *audio* [14].

**b. *Libraries* dan *Android Runtime***

***Libraries***

*Libraries* adalah layer dimana fitur-fitur *Android* berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya.

Berjalan di atas Kernel, layer ini meliputi berbagai *library* C/C++ inti seperti *Libc SSL*, serta:

a. *Libraries* media untuk pemutaran media audio dan video.
b. *Libraries* untuk manajemen tampilan.
c. *Libraries Graphics* mencakup *SGL* dan *OpenGL* untuk grafis 2D dan 3D.
d. *Libraries* SQLite untuk dukungan *database.*
e. *Libraries SSL* dan *WebKit* terintegrasi dengan *web browser* dan *security*.
f. *Libraries LiveWebcore* mencakup *modern web browser* dengan *engine embedded webview.*
g. *Libraries 3D* yang mencakup implementasi *OpenGL ES1.0 API's* [11].

***Android Runtime***

*Layer* yang membuat aplikasi *Android* dapat dijalankan dimana dalam prosesnya menggunakan Implementasi *Linux. Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi *Android.* Di dalam *Android Run Time* dibagi menjadi dua bagian yaitu:

a. *Core Libraries*: Aplikasi *Android* dibangun dalam bahasa *Java*, sementara *Dalvik* sebagai virtual mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh *Core Libraries.*
b. *Dalvik Virtual Machine*: Virtual mesin berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux Kernel untuk melakukan threading dan manajemen tingkat rendah [11].

**c. Application Framework**

*Application Framework* merupakan sekumpulan layanan yang bersama-sama membentuk suatu lingkungan di mana aplikasi-aplikasi Android berjalan dan diatur. *Framework* ini menerapkan konsep saling berbagi dan saling melengkapi yang mana suatu aplikasi mampu membagikan kemampuan beserta dengan data yang terkait di dalamnya untuk dapat digunakan oleh aplikasi lainnya [14]. Komponen-komponen yang termasuk di dalam *Application Framework* adalah sebagai berikut:

1. *Views.*
2. *Content Provider.*
3. *Resource Manager*.
4. *Notification Manager*.
5. *Activity Manager* [11].

**d. Application**

Lapisan *Application* ini meliputi seluruh aplikasi bawaan (*native application*) yang telah dilengkapi dengan fitur khusus (seperti aplikasi *web browser* dan aplikasi *email*), maupun aplikasi-aplikasi pihak ketiga (*third party applications*) yang diinstal sendiri oleh pengguna setelah memberi perangkat tersebut [14].

### 2.2.4. Android Studio IDE

Pada 16 Mei 2013, *Google* meluncurkan Android Studio sebagai lingkungan pengembangan terintegrasi (*Integrated Development Environment/IDE)* baru untuk Android. Didasarkan pada *IntelliJ IDEA*, Android Studio menyediakan beberapa *fitur* tambahan dan pengembangan dari *Eclipse ADT* yang sebelumnya digunakan dalam pengembangan aplikasi Android. Versi stabil pertama Android Studio dirilis pada Desember 2014 dan menjadi IDE resmi yang digunakan dala pengembangan Android [14].

### 2.2.5. Bootstrap

*Bootstrap* adalah paket aplikasi siap pakai untuk membuat *front-end* sebuah *website*. Bisa dikatakan, *bootstrap* adalah *template* desain *web* dengan fitur plus. *Bootstrap* diciptakan untuk mempermudah proses desain *web* bagi berbagai tingkat pengguna, mulai dari *level* pemula hingga yang sudah berpengalaman. Cukup bermodalkan pengetahuan dasar mengenai *HTML* dan *CSS*. [15]

### 2.2.6. Kemacetan Lalu Lintas

Kemacetan lalu lintas adalah suatu kondisi pada transportasi yang ditandai oleh kecepatan yang lebih lambat, waktu perjalanan yang lebih lama, dan peningkatan antrian kendaraan. Kemacetan lalu lintas di jaringan jalan perkotaan menjadi semakin bermasalah sejak tahun 1950-an [16]. Ketika permintaan lalu

lintas cukup besar sehingga interaksi antara kendaraan memperlambat kecepatan arus lalu lintas, ini mengakibatkan kemacetan.

### 2.2.7. Pendeteksi Kemacetan

Alat pendeteksi kemacetan sudah banyak digunakan oleh semua orang, berbagai fitur dan berbagai jenis telah tersedia saat ini, seperti pendeteksi kemacetan menggunakan *cctv* lalu lintas, kamera ataupun menggunakan aplikasi.

Penggunaan teknologi smartphone android yang ada sekarang ini yang memiliki beberapa fasilitas seperti *Global Positioning System* (GPS), maka dapat dimanfaatkan untuk mengetahui posisi titik koordinat pengguna jalan. Nantinya titik-titik koordinat yang terdeteksi berada dalam radius yang berdekatan dan pergerakannya lambat, maka dikategorikan daerah tersebut lambat (macet), sehingga pengguna aplikasi dapat melihat melalui *google maps* dimana titik-titik koordinat yang bertumpuk pada lokasi tertentu, sehingga dapat menghindari dan memilih rute yang lainnya [17].

### 2.2.8. Global Positioning System (GPS)

GPS adalah singkatan dari *Global Positioning System*, yang merupakan sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh alat penerima (*receiver*) di permukaan, dimana *GPS receiver* ini akan mengumpulkan informasi dari satelit *GPS*, seperti:

a. Waktu.

   *GPS receiver* menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.

b. Lokasi.

   *GPS* memberikan informasi lokasi dalam tiga dimensi:

   1. *Latitude*
   2. *Longitude*
   3. *Elevasi*

c. Kecepatan.

   Ketika berpindah tempat, *GPS* dapat menunjukkan informasi kecepatan berpindah tersebut.

d. Arah perjalanan.

*GPS* dapat menunjukkan arah tujuan.

e. Simpan lokasi.

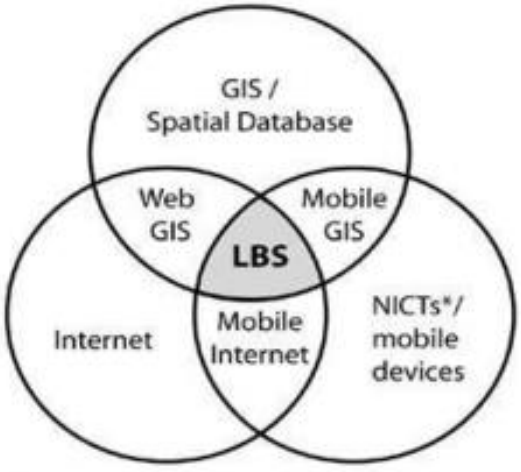
Tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh *GPS receiver*.

f. Komulasi data.

*GPS receiver* dapat menyimpan informasi *track*, seperti total perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/jam sampai tujuan, dan sebagainya [18].

### 2.2.9. Location Based Services (LBS)

Location Based Service atau Layanan Berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network* yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut [19]. LBS menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang digunakan dan memungkinkan memberikan interaksi dua arah. Pengguna memberitahu penyedia layanan untuk mendapatkan informasi yang dibutuhkan, dengan menggunakan referensi lokasi posisi pengguna tersebut. Layanan LBS dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu Geographic Information System, Internet Service, dan Mobile Devices, hal ini dapat dilihat pada Gambar 2.3 LBS Pertemuan Dari Tiga Teknologi




**Gambar 0.3 LBS Pertemuan Dari Tiga Teknologi**

Komponen LBS Dalam Layanan Berbasis Lokasi terdapat Lima komponen penting yaitu meliputi:

a. *Mobile Devices*: Suatu alat yang digunakan oleh pengguna untuk meminta informasi yang dibutuhkan. Informasi dapat diberikan dalam bentuk suara, gambar, dan text.
b. *Comunication Network*: Komponen kedua adalah jaringan komunikasi yang mengirim data pengguna dan informasi yang diminta dari mobile terminal ke *Service Provider* kemudian mengirimkan kembali informasi yang diminta ke pengguna. *Communication network* dapat berupa jaringan seluler (GSM, CDMA), *Wireless Local Area Network* (WLAN), atau *Wireless Wide Area Network* (WWAN)
c. *Positioning Component*: Untuk memproses suatu layanan maka posisi pengguna harus diketahui.
d. *Service and Application Provider*: Penyedia layanan menawarkan berbagai macam layanan kepada pengguna dan bertanggung jawab untuk memproses informasi yang diminta oleh pengguna.
e. *Data and Content Provider*: Penyedia layanan tidak selalu menyimpan semua data yang dibutuhkan yang bisa di akses *pengguna*. Untuk itu, data dapat diminta dari *data and content provider*.

### 2.2.10. Application Programming Interface (API)

*Application Programming Interface* (API) adalah antarmuka komputasi ke komponen perangkat lunak atau sistem, yang menentukan bagaimana komponen atau sistem lain dapat menggunakannya. Ini mendefinisikan jenis panggilan atau permintaan yang dapat dibuat, bagaimana membuatnya, format data yang harus digunakan, konvensi yang harus diikuti, dan lain lain. API juga dapat menyediakan mekanisme ekstensi sehingga pengguna dapat memperluas fungsionalitas yang ada dengan berbagai cara dan untuk berbagai derajat [20]. API dapat sepenuhnya khusus, khusus untuk komponen, atau dapat dirancang berdasarkan standar industri untuk memastikan interoperabilitas. Beberapa API harus didokumentasikan, yang lain dirancang agar dapat "diinterogasi" untuk menentukan fungsionalitas yang didukung. Karena komponen / sistem lain hanya mengandalkan API, sistem yang

menyediakan API dapat (idealnya) mengubah detail internal "di belakang" API itu tanpa mempengaruhi penggunanya.

Saat ini, dengan munculnya *REST* dan layanan *web* melalui *HTTP*, istilah ini sering dianggap merujuk ke API layanan tersebut ketika tidak diberikan konteks lain (lihat bagian *API Web)*. Kadang-kadang istilah API, dengan ekstensi, digunakan untuk merujuk pada subset entitas perangkat lunak (kode, subkomponen, modul, dll.) Yang berfungsi untuk benar-benar mengimplementasikan API dari beberapa komponen atau sistem yang mencakup.

### 2.2.11. GoogleMaps API

*Google Maps* adalah layanan pemetaan web yang dikembangkan oleh Google. Menawarkan citra satelit, foto udara, peta jalan, pemandangan panorama 360 ° interaktif di jalan (Street View), kondisi lalu lintas on-time, dan perencanaan rute untuk bepergian dengan berjalan kaki, mobil, sepeda dan udara (dalam versi beta), atau transportasi umum [21].

Pengembang menggunakan layanan ini untuk menambahkan peta berdasarkan data dari Google Maps API dan *server google maps* secara otomatis.

### 2.2.12. Google Direction API

*Google Direction API* adalah layanan yang menghitung arah antar lokasi menggunakan permintaan HTTP. *Google Direction API* memiliki beberapa parameter yang di butuhkan untuk meminta lama perjalanan. Daftar dari parameter yang ada dapat dilihat pada Tabel 2.1 Daftar parameter yang dibutuhkan *Google Direction API* [22].

**Tabel 0.1 Daftar parameter yang dibutuhkan *Google Direction API***

| Nama Parameter | Masukan |
|---|---|
| *Origin* | Koordinat atau Nama lokasi |
| *Destination* | Koordinat atau Nama lokasi |
| *key* | Kunci API |

Parameter yang Terdapat pada Tabel 2.1 merupakan parameter yang dibutuhkan untuk Google Maps Google Maps Direction API untuk dapat digunakan parameter *Origin* merupakan parameter untuk Titik Awal perhitungan, parameter

*Destination* Merupakan parameter untuk titik akhir perhitungan, parameter Key merupakan kunci API yang digunakan pada aplikasi, *Google Maps Direction API* sendiri memiliki beberapa parameter Opsional yang diberikan daftar dari parameter opsional yang ada dapat dilihat pada Tabel 2.2 Daftar parameter opsional *Google Maps Direction API*

**Tabel 0.2 Daftar parameter opsional *Google Maps Direction API***

| Nama Parameter | Masukan |
|---|---|
| *mode* | *Driving,walking,bicycling,transit* |
| *waypoints* | Koordinat Lokasi atau Nama Lokasi |
| *alternatives* | *True* atau *flase* |
| *language* | Bahasa yang digunakan |
| *region* | *country code top-level domain* |
| *Avoid* | *Tolls,highway,ferries,indoor* |
| *Units* | Satuan yang digunakan |
| *Arrival_time* | *Unix Time* |
| *Departure_time* | *Unix Time,now* |
| *Traffic_model* | *Best_guess,Pessimistic,optimistic* |
| *Transit_mode* | *Bus,subway,train,tram,rail* |
| *Transit_routing_preference* | *Less_walking,fewer_transfer* |

Penjelasan mengenai parameter parameter pada Tabel 2.2 adalah sebagai berikut:

1. *Mode*

   Merupakan parameter untuk menentukan mode transportasi yang akan digunakan saat menghitung arah.

2. *Waypoints*

   Merupakan parameter untuk menentukan susunan lokasi perantara untuk disertakan di sepanjang rute antara titik asal dan tujuan sebagai melewati atau menghentikan lokasi. Titik jalan mengubah rute dengan mengarahkannya melalui lokasi yang ditentukan.

3. *Alternatives*

   menetapkan bahwa layanan arah dapat memberikan lebih dari satu alternatif rute dalam respons.

4. *Language*

   Merupakan parameter bahasa yang akan digunakan untuk hasil dari *Google Maps Direction API.*

5. *Region*

   Merupakan Parameter area untuk membatasi hasil dari *geocoder.*

6. *Avoid*

   Merupakan Parameter yang membatasi jalur yang digunakan seperti tidak melewati toll .

7. *Units*

   merupakan Parameter yang menentukan satuan jarak yang akan di gunakan pada hasil .

8. *Arrival Time*

   Merupakan parameter yang digunakan untuk menspesifikasi waktu Sampai yang di inginkan namun parameter ini hanya dapat digunakan bila berkendara menggunakan transit.

9. *Departure Time*

   Merupakan parameter yang digunakan untuk menspesifikasi waktu keberangkatan yang di inginkan parameter ini membutuhkan masukan Unix Time yaitu detik semenjak 1970 januari 1 *UTC* (*Coordinated universal Time).*

10. *Traffic model*

    Merupakan Parameter yang dipakai untuk asumsi perhitungan prediksi lama Perjalanan.

11. *Transit model*

    Merupakan Parameter yang dipakai untuk asumsi perhitungan prediksi lama Perjalanan transit.

12. *Transit routing preference*

Merupakan parameter untuk memilih jalur transit apakah lebih banyak jalan atau berkendara.

### 2.2.13. Google direction API Travel time prediction

Prediksi pada Gooogle Direction API sendiri menggunakan data kondisi lalu lintas terdahulu [22] dimana data lalu lintas yang digunakan merupakan data lalu lintas waktu pada satu hari dan hari dalam satu minggu, dimana pessimistic dan optimistic hanya menggunakan data best_guess dapat pula mempertimbangkan kondisi lalu lintas yang sedang terjadi [23].

### 2.2.14. UNIX time

UNIX time adalah adalah sistem untuk menggambarkan suatu titik waktu berbasis integer dari jumlah detik tanpa lompatan yang berlalu sejak 00:00 1 Januari 1970 UTC. Meningkatnya penggunaan sistem mirip UNIX dalam aplikasi Internet dan *smartphone* berarti bahwa metode ketepatan waktu komputasi ini hampir memonopoli representasi waktu dalam sistem digital. Perangkat iOS, Android, Linux, dan MacOS semuanya menggunakan standar ini dalam perhitungan waktu [24].

### 2.2.15. Cuaca

Cuaca adalah keadaan atmosfer, yang menggambarkan misalnya tingkat panas atau dingin, basah atau kering, tenang atau badai, cerah atau berawan. Sebagian besar fenomena cuaca terjadi di tingkat terendah atmosfer, troposfer, tepat di bawah stratosfer. Cuaca mengacu pada suhu sehari-hari dan aktivitas presipitasi, sedangkan iklim adalah istilah untuk rata-rata kondisi atmosfer dalam periode waktu yang lebih lama [25]. Ketika digunakan tanpa kualifikasi, "cuaca" secara umum dipahami sebagai cuaca di bumi.

Cuaca didorong oleh perbedaan tekanan udara, suhu, dan kelembaban antara satu tempat dan tempat lainnya. Perbedaan-perbedaan ini dapat terjadi karena sudut matahari pada titik tertentu, yang bervariasi dengan garis lintang. Kontras suhu yang kuat antara udara kutub dan tropis menimbulkan sirkulasi atmosfer skala terbesar: Sel Hadley, Sel Ferrel, Sel Polar, dan aliran jet. Sistem cuaca di garis lintang pertengahan, seperti siklon ekstratropis, disebabkan oleh ketidakstabilan aliran aliran jet. Karena sumbu bumi dimiringkan relatif terhadap bidang orbitnya,

sinar matahari terjadi pada sudut yang berbeda pada waktu yang berbeda dalam setahun. Di permukaan bumi, suhu biasanya berkisar ± 40 ° C (−40 ° F hingga 100 ° F) setiap tahun. Selama ribuan tahun, perubahan orbit Bumi dapat memengaruhi jumlah dan distribusi energi surya yang diterima oleh Bumi, sehingga memengaruhi iklim jangka panjang dan perubahan iklim global.

### 2.2.16. OpenWeatherMaps API

*OpenWeatherMap* adalah layanan *online* yang menyediakan data cuaca. Dimiliki oleh OpenWeather Ltd berkantor pusat di London, Inggris. *Openweathermap* memberikan data cuaca saat ini, perkiraan dan data historis (mulai dari 1979) ke lebih dari dua juta pelanggan [26].

Lebih dari dua puluh API cuaca telah dikembangkan untuk mendapatkan berbagai jenis data cuaca. Mereka mendukung berbagai bahasa, satuan ukuran dan format data. Selain itu, layanan OpenWeatherMap memungkinkan setiap pengguna untuk mendapatkan data cuaca dasar di situs web perusahaan.

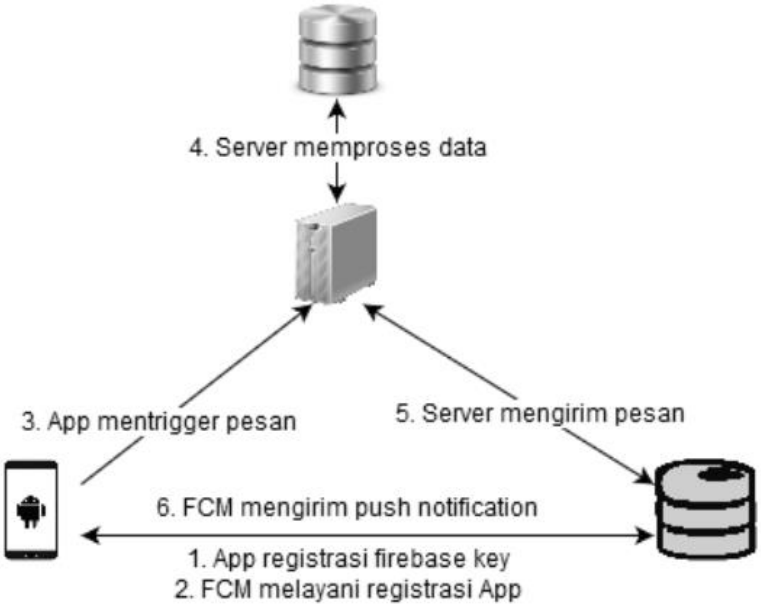
### 2.2.17. Monitoring

*Monitoring* atau dalam bahasa Indonesia Pemantauan dapat dijelaskan sebagai kesadaran (*awareness*) tentang apa yang ingin diketahui. Monitoring akan memberikan informasi bahwa pengukuran dan evaluasi yang diselesaikan berulang dari waktu ke waktu, monitoring umumnya dilakukan untuk tujuan tertentu.

Dalam sebuah aplikasi, biasanya monitoring berupa pemantauan suatu data ataupun aktifitas yang dilakukan untuk tujuan tujuan tertentu.

### 2.2.18. Firebase Cloud Messaging

Firebase merupakan teknologi back-end as a service (BaaS) berbasis cloud dari Google yang menyediakan berbagai layanan, salah satunya adalah fitur push notification bernama firebase cloud messaging. Firebase cloud messaging merupakan perkembangan dari Google cloud messaging yang menyediakan layanan agar developer dapat mengirimkan pesan notifikasi ke perangkat android dari server [27] . Pada Gambar 2.4 Alur Kerja Firebase Cloud Messaging adalah penjelasan bagaimana alur kerja firebase cloud messaging dapat mengirimkan pesan ke perangkat android pengguna:




**Gambar 0.4 Alur Kerja Firebase Cloud Messaging [28]**

a. Ketika aplikasi pada perangkat baru pengguna dijalankan, secara otomatis akan melakukan registrasi FCM token.
b. Firebase Cloud Messaging melakukan generate FCM token dan menyimpannya pada akun pengguna. Aplikasi mengaktifkan *trigger* untuk meminta Firebase Cloud Messaging mengirimkan push notification.
c. Server memproses data yang ada pada basis data dan menyeleksi sesuai dengan FCM token yang diminta.
d. Server mengirimkan daftar FCM token yang menjadi tujuan dari pesan push notification.
e. Firebase Cloud Messaging mengirimkan pesan push notification ke perangkat android yang telah diseleksi.

### 2.2.19. Java

*Java* adalah suatu teknologi di dunia *software* komputer, yang merupakan suatu bahasa pemrograman, dan sekaligus suatu *platform.* Sebagai bahasa pemrograman, *Java* dikenal sebagai bahasa pemrograman tingkat tinggi. Java mudah dipelajari, terutama bagi *programmer* yang telah mengenal C/C++. Java merupakan bahasa pemrograman berorientasi objek yang merupakan paradigma pemrograman masa depan. Sebagai bahasa pemrograman Java dirancang menjadi handal dan aman. Java juga dirancang agar dapat dijalankan di semua platform. Dan

juga dirancang untuk menghasilkan aplikasi – aplikasi dengan performansi yang terbaik, seperti aplikasi *database* Oracle 8i/9i yang core-nya dibangun menggunakan bahasa pemrograman Java. Sedangkan Java bersifat neutral architecture, karena *Java Compiler* yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitekture perangkat keras yang disebut sebagai *Java Bytecode* [29].

### 2.2.20. Javascript Object Notation (JSON)

*Java Script Object Notation* (JSON) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. JSON merupakan format teks yang tidak bergantung pada bahasa pemograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga *C* termasuk *C, C++, C#, Java, Java Script, Python* dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record)*, struktur (*struct*), kamus (*dictionary)*, tabel hash *(hash table*), daftar berkunci (*keyed list*) atau *associative array*. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list)* atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut.

1. Objek

   Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai

   Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

4. *String*

   *String* adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam *string* dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan string C atau Java.

5. Angka

   Fungsi Angka pada JSON sangat mirip dengan fungsi angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan [29].

### 2.2.21. Hypertext Processor (PHP)

PHP (*Hypertext Processor*) yaitu salah satu bahasa pemrograman *web server-side* yang bersifat *open source* yang terintegrasi dengan *HTML* dan berada pada *server* (*server side HTML embedded scripting*).

Pada awalnya PHP merupakan kependekan dari *Personal Home Page* (Situs personal). PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form Interpreted* (FI), yang wujudnya berupa sekumpulan *script* yang digunakan untuk mengolah data formulir di *web*. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FFI. Perilisan kode sumber ini menjadi sumber terbuka, maka banyak programmer yang tertarik untuk ikut mengembangkan PHP [13].

Berikut ini beberapa kelebihan bahasa pemrograman PHP:

1. Banyaknya *web server* yang mendukung bahasa pemrograman PHP sehingga konfigurasinya semakin mudah.

2. Pengembangan bahasa pemrograman PHP tergolong lebih mudah, karena banyak *programmer* yang membantu dalam mengembangkan maupun menggunakannya.
3. Relatif mudah dan dipahami, karena materi-materi/referensi untuk mempelajari PHP sudah banyak.
4. Bahasa pemrograman PHP juga dapat disisipkan ke dalam HTML.
5. Cocok digunakan untuk pemrograman *web* dinamis, walau bisa juga untuk membuat program komputer lainnya.
6. PHP merupakan bahasa pemrograman bersifat *opensource*, sehingga dapat digunakan di banyak sistem operasi komputer dan tentunya gratis [13].

### 2.2.22. MySQL

MySQL merupakan *database engine* atau *server database* yang mendukung bahasa *database* pencarian SQL. MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang *multithread* dan *multiuser*. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License*(GPL). Mereka juga menjual bawah lisensi komersil untuk kasus-kasus di mana penggunanya tidak cocok dengan pengguna GPL.

MySQL merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah dan cepat secara otomatis. Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh user maupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan database server lainnya dalam hal *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query* MySQL bisa sepuluh kali lebih cepat dari *PostgreSQL* dan lima kali lebih cepat dibandingkan Interbase [13].

Berikut beberapa kelebihan MySQL:

1. MySQL dapat digunakan berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.

2. MySQL merupakan *open source*.
3. MySQL dapat digunakan oleh beberapa *user* dalam waktu bersamaan tanpa mengalami masalah atau konflik.
4. MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak lagi SQL per satuan waktu.
5. MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.
6. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. MySQL mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris.
8. MySQL memiliki *interface* (antarmuka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*) [13].

**2.2.23. Web Service**

*Web Service* adalah perangkat lunak sistem yang di desain untuk mendukung interaksi antar-mesin melalui jaringan komputer. *Web service* memiliki antarmuka yang dideskripsikan ke format yang dapat di proses oleh mesin seperti *Web Service Description Language*(WSDL).

Berikut beberapa kelebihan *Web Service*:

1. *Web service* dapat digunakan untuk mentransformasikan satu atau beberapa bisnis logic atau class dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. *Web service* memiliki kemudahan dalam proses *deployment*-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. *Web service* cukup di *upload* ke *web server* dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.
3. *Web service* berjalan di *port* 8o yang merupakan protokol standar HTTP, dengan demikian *web service* tidak memerlukan konfigurasi khusus di sisi *firewall* [13]

### *2.2.24.* Konsep Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau biasa disebut OOP (*Object Oriented Programming)* adalah sebuah konsep dalam *programming* yang lebih memandang dari segi bagian perbagian atau dilihat berdasarkan objek sehingga mempermudah *programmer* men*design* perangkat lunak karena hanya harus mengimplementasikan objek-objek tersebut dan menghubungkan antar objek tersebut [30].

#### 2.2.24.1. Unified *Modelling Language*

UML (*Unified Modeling Language*) merupakan pengganti dari metode analisis berorientasi object dan design berorientasi object (O OA&D) yang dimunculkansekitar akhir tahun 80-an dan awal tahun 90-an.

UML merupakan gabungan dari metode *Booch*, *Rumbaugh* (OMT) dan *Jacobson*. Tetapi UML ini akan mencakup lebih luas daripada OOA&D. Pada pertengahanpengembangan UML dilakukan standarisasi proses dengan OMG (*ObjectManagement Group*) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

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

Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika ingin berdiskusi tentang desain dengan seseorang, maka hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain. UML merupakan bahasa standar untuk penulisan blueprint software yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak [30].

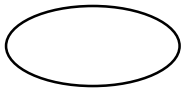
##### 2.2.24.1.1. Diagram dalam UML

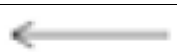
Dalam UML terdapat diagram-diagram yang bisa menggambarkan bagian atau aspek tertentu dari sebuah sistem. Sehingga dapat terlihat jelas alur dan gambaran umum dari perangkat lunak yang dibangun. Ada beberapa jenis diagram dalam UML yaitu:

#### 2.2.24.1.1.1. *Use Case* Diagram

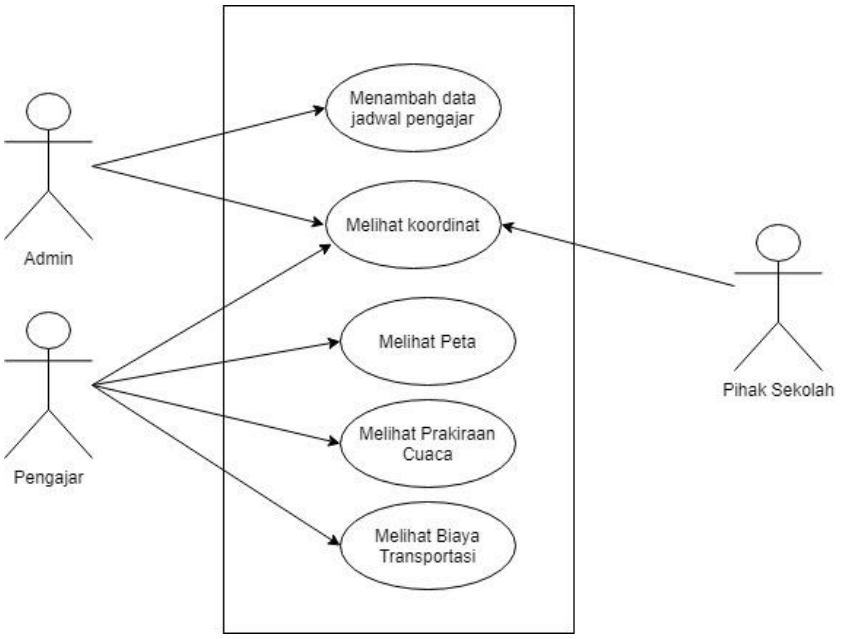
Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh system dalam bentuk teks sebagai dokumentasi dari use case symbol namun dapat juga dilakukan dalam activity diagrams. Use case digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada di dalam sistem [30]. Tabel simbol *use case* diagram dapat dilihat pada Tabel 2.4 Simbol *Use Case* Diagram.

**Tabel 0.3 Simbol *Use Case* Diagram**

| No. | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1 |  | *Actor* | Merupakan orang, proses atau *system* lain yang berinteraksi dengan *system* yang akan dibuat. Jadi walaupun simbol aktor dalam diagram *usecase* berbentuk orang, namun aktor belum tentu orang. |
| 2 |  | *Use case* | Merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling berinteraksi atau bertukar pesan antar unit maupun aktor. |
| 3 |  | *Association* | Merupakan relasi yang digunakan untuk menggambarkan interaksi antara *usecase* dan aktor. Asosiasi juga menggambarkan berapa banyak objek lain yang bisa berinteraksi dengan suatu objek atau disebut *multiplicity*. |
| 4 |  | *Extend* | Menghubungkan antara satu objek dengan objek lain. |

| | | | |
|---|---|---|---|
| 5 |  | *Generalization* | Hubungan dimana objek berbagai perilaku dan struktur data dari objek yang ada di atasnya objek induk. |
| 6 |  | *Include* | Menspesifikasi bahwa *usecase* sumber secara eksplisit. |
| 7 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri. |
| 8 |  | *System Boundary* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |

Berikut adalah contoh penggunaan dari *use case* diagram, dapat dilihat di Gambar 2.5 Contoh *Use Case* Diagram.

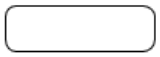
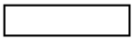
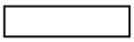
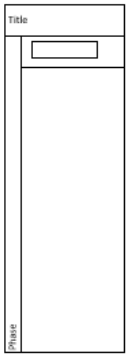



**Gambar 0.5 Contoh *Use Case* Diagram**

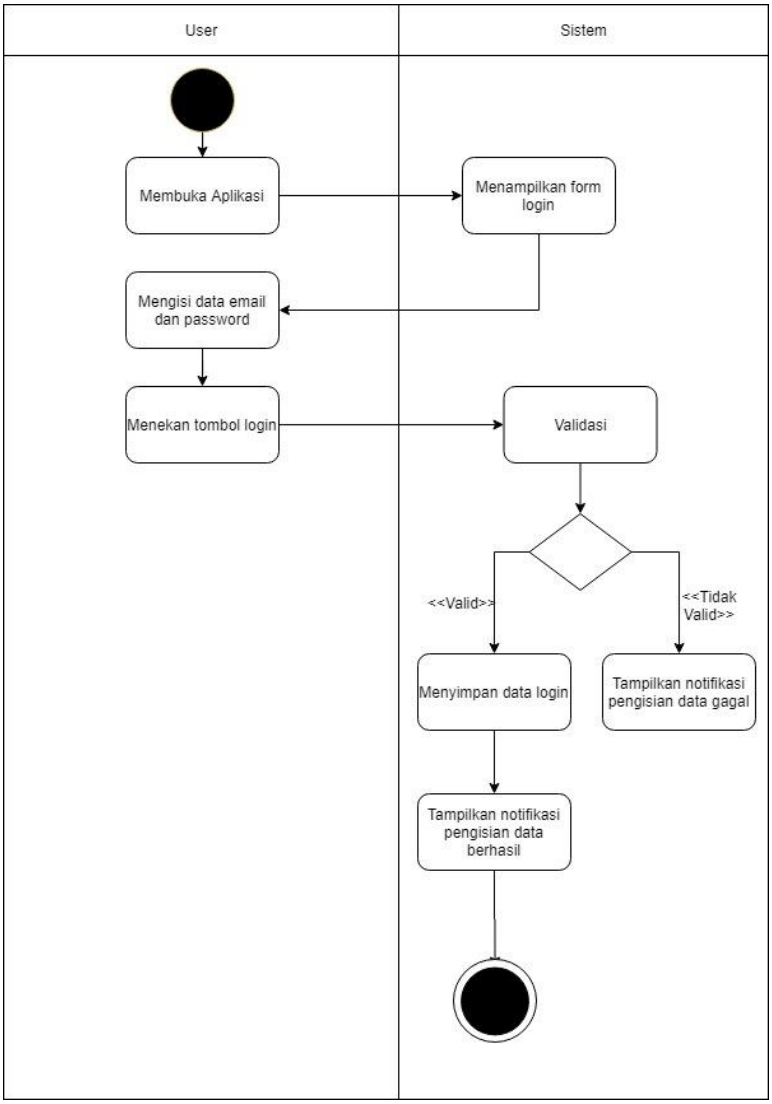
#### 2.2.24.1.1.2. *Activity* Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi. [30]. Tabel simbol *activity* diagram dapat dilihat pada Tabel 2.5 *Activity* Diagram.

**Tabel 0.4 Simbol *Activity* Diagram**

| No. | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | *Action* | *State* dari sistem yang mencerminkan suatu aksi. |
| 2. |  | *Decision* | Digunakan untuk menggambarkan suatu pengambilan keputusan / tindakan pada kondisi tertentu. |
| 3. |  | *Initial Node* | Menunjukkan dimulainya suatu aktifitas. |
| 4. |  | *Final Node* | Menunjukkan akhir dari aktifitas. |
| 5. |  | *Fork Node* | Digunakan untuk menunjukkan aktifitas yang dilakukan secara paralel. |
| 6. |  | *Join Node* | Digunakan untuk menunjukkan aktifitas yang digabungkan. |
| 7. |  | *Swimlane* | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi. |

Berikut adalah contoh dari *activity* diagram, dapat dilihat di Gambar 2.6 Contoh *Activity* Diagram.

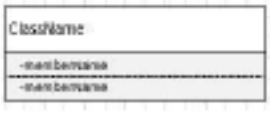
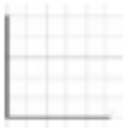


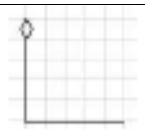

**Gambar 0.6 Contoh *Activity* Diagram**

### 2.2.24.1.1.3. *Class* Diagram

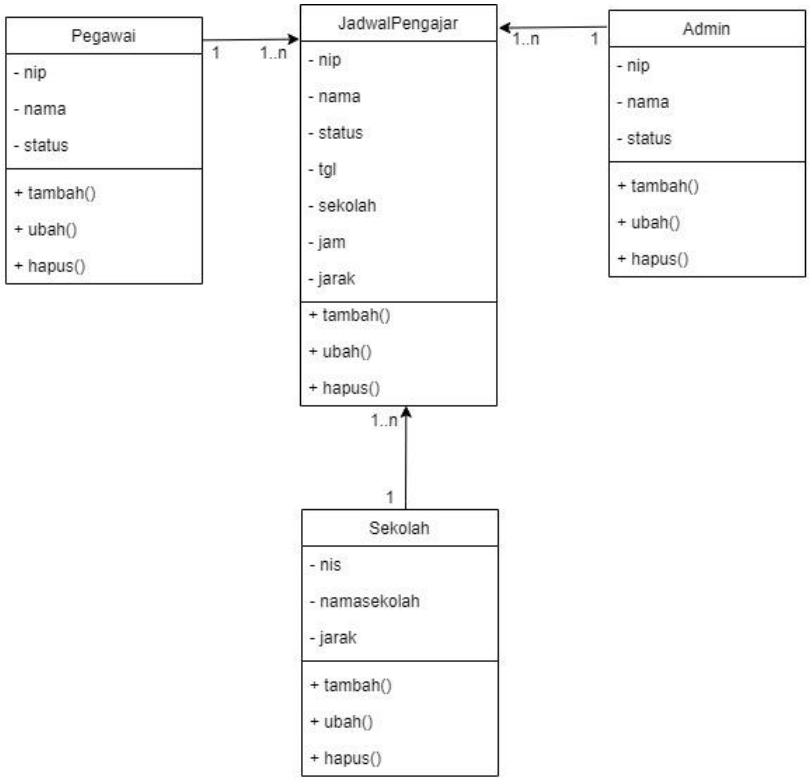
Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu class merupakan spesialisasi dari *class* lainnya), atau package (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class* diagram [30]. Tabel simbol *Class* Diagram dapat dilihat pada Tabel 2.6 Simbol *Class* Diagram

**Tabel 0.5 Simbol *Class* Diagram**

| No | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | *Class* | Blok pembangunan pada pemrograman berorientasi objek. Bagian atas adalah bagian dari class. Bagian tengah mendefinisikan *property*/*atribut class*. Bagian akhir mendefinisikan *method -method* dari sebuah class. |
| 2. |  | *Association* | *Relationship* paling umum antara 2 *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 class. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapatmenampilkan hukum-hukum multiplisitas pada sebuah relasi. |
| 3. |  | *Compisition* | Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempat bergantung tersebut. |
| 4. |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri (*independent*). |

| | | | |
|---|---|---|---|
| 5. |  | *Aggregation* | Mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi. |
| 6. |  | *Directed Association* | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai *multiplicity* |

Berikut adalah contoh penggunaan class diagram, dapat dilihat pada Gambar 2.7 Contoh *Class* Diagram.

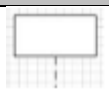
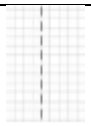
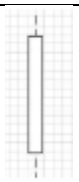
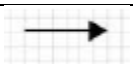
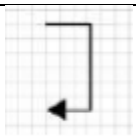
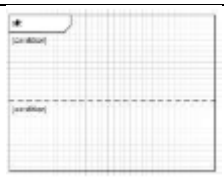



**Gambar 0.7 Contoh *Class* Diagram**

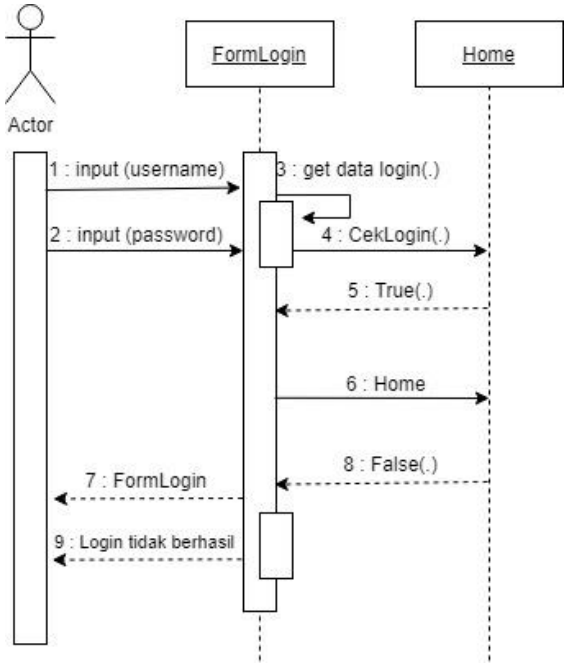
### 2.2.24.1.1.4. Sequence Diagram

Menggambarkan kolaborasi dinamis antara sejumlah objek. Kegunaanya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem [30]. Tabel simbol *sequence* diagram dapat dilihat pada Tabel 2.7 Simbol *Sequence* Diagram.

**Tabel 0.6 Simbol *Sequence* Diagram**

| No. | Simbol | Nama | Keterangan |
|---|---|---|---|
| 1. |  | partisipan | Merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. |
| 2. |  | *Lifeline* | Mengindikasikan keberadaan sebuah objek dalam basis waktu. |
| 3. |  | *Activation* | Mengindikasikan sebuah objek yang akan melakukan sebuah aksi. |
| 4. |  | *Message* | Mengindikasikan komunikasi antar *obejct*. |
| 5. |  | *Self-Message* | Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri. |
| 6. |  | *Sequence fragment* | Membungkus sebagian interaksi dengan diagram urutan yang memiliki beberapa tipe seperti *ref, assert, loop, break, alt, opt, neg, par, region* |

Berikut adalah contoh dari sequence diagram, dapat dilihat di Gambar 2.8 Contoh *Sequence* Diagram.




**Gambar 0.8 Contoh *Sequence* Diagram**