BAB II
TINJAUAN PUSTAKA

II.1 Profil Instansi

Travel MGo Shuttle merupakan anak perusahan dari PT Citra Maharlika Nusantara Corpora Tbk merupakan nama baru dari PT Cipaganti Citra Graha Tbk, yang bergerak di bidang jasa travel. PT Cipaganti Citra Graha Tbk berpusat di GRAHA WIJAYA PURI BLOK A/3-4. Jl. Wijaya 2, Kebayoran Baru, Jakarta Selatan 12160 – Indonesia.

II.1.1 Sejarah Instansi

Secara historis PT Citra Maharlika Nusantara Corpora Tbk merupakan nama baru dari PT Cipaganti Citra Graha Tbk, sebuah perusahaan terbuka yang berawal dari usaha kecil jual beli mobil bekas di Bandung pada sekitar tahun 1985. Citra Maharlika Nusantara Corpora, secara harafiah mempunyai arti "citra kebangsawanan Nusantara", gambaran tentang nilai-nilai kewiraan, cerdik, bijaksana dan berbudi luhur. Kemudian perusahaan terus berkembang dan berhasil membuktikan eksistensinya sebagai salah satu perusahaan transportasi yang dikenal oleh masyarakat khususnya para pengguna jasanya. Pada perkembangan selanjutnya, diakhir tahun 2014 terjadi perubahan pemegang saham mayoritas yang menjadi pemegang kendali perusahaan. Dibentuklah manajemen baru, yang terus berupaya memperbaiki kinerja usaha dan pembenahan permasalahan-permasalahan perusahaan. Termasuk dengan memberi nuansa baru melalui langkah penggantian nama PT. Cipaganti Citra Graha Tbk. menjadi PT. Citra Maharlika Nusantara Corpora Tbk. disusul dengan re-branding atas sejumlah layanan bisnis di CMNC.

II.1.2 Logo Instansi

Logo bagi perusahaan merupakan identitas dari suatu instansi sebagai jati diri perusahaan. Logo PT. Citra Maharlika Nusantara Corpora Tbk. dapat dilihat pada Gambar II.1.



Gambar II.1 Logo PT. Citra Maharlika Nusantara Corpora Tbk.

II.1.3 Keterangan Logo

Dari logo tersebut di atas, terdapat makna tersendiri bagi PT. Citra Maharlika Nusantara Corpora Tbk., baik dari segi warna dan nama CMNC itu sendiri. makna yang terkandung dalam logo tersebut yaitu gambar mahkota menegaskan visinya menjadi perusahaan kelas dunia dan juga dominasi warna biru bermakna harapan.

II.1.4 Visi dan Misi

Adapun visi dan misi PT. Citra Maharlika Nusantara Corpora Tbk. adalah sebagai berikut:

a. Visi CMNC

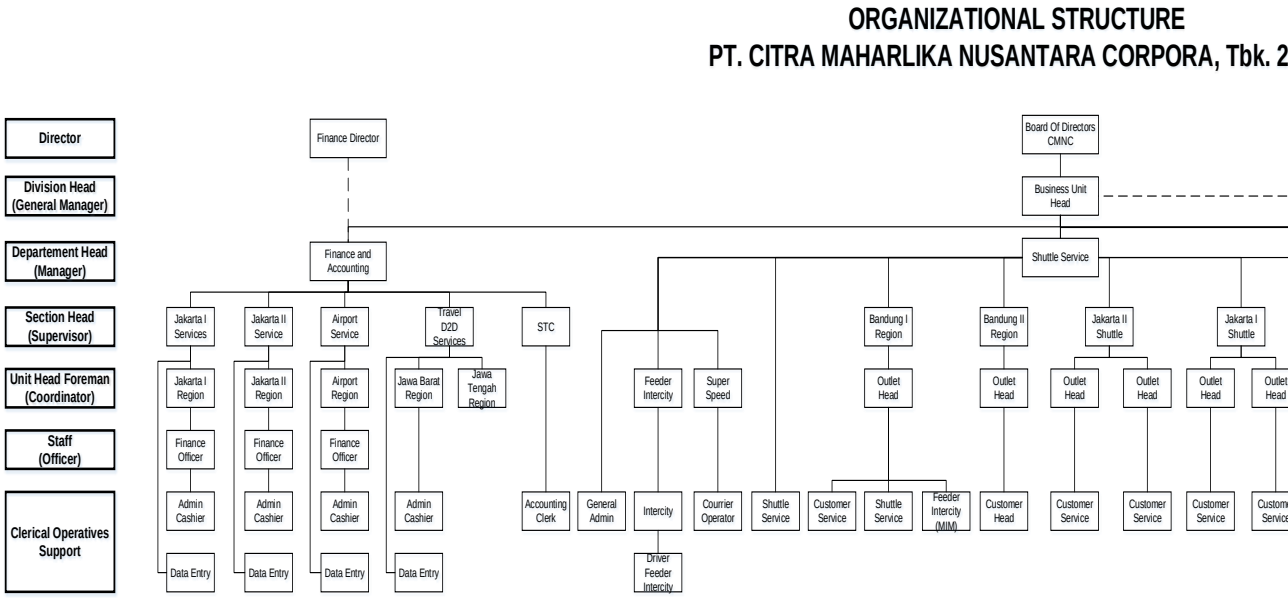
   Menjadi perusahaan kelas dunia yang membantu masyarakat Indonesia saling bertatap muka dan bekerjasama untuk mencapai kesejahteraan yang lebih baik.

b. Misi CMNC

   Mengembangkan sistim transportasi berskala nasional, membangun infrastruktur transportasi, dan menyediakan sarana transportasi berbasis teknologi mutakhir dan sumber daya manusia yang berkualitas.

II.1.5 Struktur Organisasi

Struktur organisasi ini merupakan penggambaran secara grafik yang menggambarkan struktur kerja dari setiap departemen yang mempunyai wewenang dan tanggung jawab masing – masing yang ada di lingkungan perusahaan. Struktur organisasi yang ada di MGo Huttle dapat dilihat pada gambar di bawah ini.



Gambar II.2 Struktur Organisasi MGo Shuttle

II.1.6 Deskripsi dan Tanggung Jawab

Dalam tataran praktisnya, masing – masing bagian memiliki *job description* yang berbeda-beda, yaitu sebagai berikut:

a. Director

Director atau direktur merupakan fungsi jabatan tertinggi dalam sebuah perusahaan yang secara garis besar bertanggung jawab mengatur perusahaan secara keseluruhan. Tugas dan tanggung jawab direktur dapat dilihat sebagai berikut:

1. Memutuskan dan menentukan peraturan dan kebijakan tertinggi perusahaan
2. Bertanggung jawab dalam memimpin dan menjalankan perusahaan
3. Bertanggung jawab atas kerugian yang dihadapi perusahaan termasuk juga keuntungan perusahaan
4. Merencanakan serta mengembangkan sumber – sumber pendapatan dan pembelanjaan kekayaan perusahaan

5. Bertindak sebagai perwakilan perusahaan dalam hubungannya dengan dunia luar perusahaan
6. Menetapkan strategi – strategi strategis untuk mencapai visi dan misi perushaan
7. Mengkoordinasikan dan mengawasi semua kegiatan di perusahaan, mulai bidang administrasi, kepegawaian hingga pengadaan barang
8. Mengangkat dan meberhentikan karyawan perusahaan

b. Division Head (General Manager)

Division Head (General Manager) merupakan jabatan kerja tinggi di sebuah perusahaan setelah direktur. Tugas dan tanggung jawab *division head* dapat dilihat sebagai berikut ini:

1. Memimpin perusahaan dan menjadi motivator bagi karyawannya
2. Mengelola operasional harian perusahaan
3. Merencanakan, melaksanakan, mengkoordinasi, mengawasi dan menganalisis semua aktivitas bisnis perusahaan
4. Mengelola perusahaan sesuai dengan visi dan misi perusahaan
5. Merencanakan, mengelola dan mengawasi peroses penganggaran di perusahaan
6. Merencanakan dan mengontrol kebijakan perusahaan agar dapa berjalan dengan maksimal
7. Memastikan setiap departemen melakukan strategi perusahaan dengan efektif dan optimal
8. Mengelola anggaran keuangan perusahaan
9. Memutuskan dan membuat kebijakan untuk kemajuan perusahaan
10. Membuat prosedur dan standar perusahaan
11. Membuat keputusan penting dalam hal investasi, integrase, aliansi dan divestasi
12. Merencanakan dan mengeksekusi rencana strategis perusahaan jangka menengah dan jangka panjang untuk kemajuan perusahaan
13. Menghadiri pertemuan, seminar, konferensi maupun pelatihan

c. Department Head (Manager)

Department Head (Manager) pada MGo Shuttle dibagi menjadi beberapa departemen yang diantaranya adalah *finance & accounting, shuttle service, travel service*, dan *pool operation.* Tugas dan tanggung jawab manajer secara umum dapat dilihat sebagai berikut ini:

1. Mempertahankan staf dengan merekrut, memilih, mengorientasi, dan melakukan pelatihan karyawab, menjaga lingkungan kerja yang nyaman, aman, dan tertib
2. Menyelesaikan dan mengevaluasi kinerja staf dengan berkomunikasi perncanaan, *monitoring*, dan menilai hasil pekerjaan

3. Melakukan coaching, konseling, dan mendisiplinkan karyawan, mengembangkan, mengkoordinasikan sistem, kebijakan, prosedur, dan standar produktivitas
4. Menetapkan tujuan strategis dengan mengumpulkan bidang bisnis yang bersangkutan, keuangan, layanan, dan informasi
5. Mengidentifikasi dan mengevaluasi tren, memilih tindakan, mendefinisikan tujuan dan mengevaluasikan hasil
6. Menyelesaikan tujuan keuangan dengan perencanaan kebutuhan, mempersiapkan anggaran tahunan, pengeluaram, menganalisis varians dan memulai tindakan korektif
7. Mempertahankan kualitas layanan dengan menegakan standar kualitas dan layanan pelanggan, menganalisis dan menyelesaikan kualitas dan layanan masalah pelanggan, merekomendasikan perbaikan sistem
8. Mempertahankan pengetahuan professional dan teknis dengan menghadiri workshop pelatihan, meninjau publikasi professional, membangun jaringan pribadi, benchmarking, berpartisipasi dalam sisial bermasyarakat secara professional
9. Pengambilan keputusan, membuat rencana, menyusun organisas, pengarahan organisasi, pengendalian, penilaian dan pelaporan.

d. Section Head (Supervisor)

Section Head (Supervisor) merupakan seseorang yang memonitor jalannya pekerjaan, kondisi dalam suatu grup maupun organisasi pekerjaan. Tugas dan tanggung jawab supervisor dapat dilihat sebagai berikut ini:

1. Menyampaikan kebijakan yang disampaikan oleh jabatan di atasnya kepada seluruh bawahan dan grupnya
2. Mengatur kelompok kerja pada grup yang dipegangnya
3. Memberikan tugas pada bawahannya
4. Melaksanakan tugas, proyek, dan pekerjaan secara langsung
5. Memberikan training pada bawahan
6. Memimpin dan memotivasi bawahan
7. Menegakan aturan yang telah ditentukan oleh perusahaan
8. Mendisiplinkan bawahan
9. Memecahkan masalah sehari hari yang rutin
10. Membuat rencana jangka pendek untuk tugas yang telah ditetapkan oleh atasannya
11. Mengontrol dan mengevaluasi kinerja bawahan
12. Memberikan info pada manajemen mengenasi kondisi bawahan, atau menjadi perantara pekerja dengan manajemen

e. Unit Head Foreman (Coordinator)

Unit Head Foreman (Coordinator) merupakan seorang yang diberi tanggung jawab untuk suatu skope permasalahan dan mengkoordinasikan orang, seperti coordinator tim. Tugas dan tanggung jawab coordinator dapat dilihat sebagai berikut ini:

1. Mengetahui beban bawahan, apakah overload atau normal
2. Mengetahui apa yang dikerjakan bawahan
3. Mengetahui target yang dikerjakan bawahan
4. Memastikan bahwa bawahannya selalu memiliki pekerjaan sesuai dengan beban kerja mereka masing – masing
5. Mencarikan pekerjaan agar orang yang dibawahnya memenuhi pekerjaan mereka
6. Memberikan arahan / solusi jika bawahannya mengalami kesulitan
7. Bertanggung jawab atas pekerjaan yang dibawahnya
8. Bertanggung jawab atas penyelesaian pekerjaan bawahhnya dan pekerjaannya sendiri

f. Staff (Officer)

Staff (Officer) merupakan seseorang yang melakukan tugas – tugas pokok dari suatu perusahaan. Tugas dan tanggung jawab staff dapat dilihat sebagai berikut ini:

1. Mengumpulkan data
2. Mengintepretasikan data
3. Mengusulkan alternative tinfakan
4. Mendiskukusikan rencana – rencana yang sedang dipikirkan dengan berbagai hak dan memperoleh kesepakatan mereka atau memperoleh alasan mengapa rencana tersebut ditolak
5. Mempersiakan instruksi – instruksi tertlis dan dokumen – dokumen lainnya yang diperlukan untuk melaksanakan kegiatan – kegiatan yang merupakan realisasi dari rencana yang telah ditetapkan
6. Mengamati kegiatan – kegiatan operasional dan kondisi – kondisi yang dihadapi untuk mengadakan apakah instruksi telah dijalankan dengan baik dan apakah instruksi tersebut menghambat atau memperlancar proses pencapaian tujuan
7. Mengusahakan pertukaran informasi antara para petugas operasional mengenai pelaksanaan untuk meningkatkan kegiatan koordinasi
8. Memberikan informasi dan nasihat kepada petugas operasional mengenai pelaksanaan tugas – tugas yang telah didelegasikan kepada mereka

g. Clerical Operatives Support

Clerical Operatives Support pada perusahaan dibagi menjadi beberapa bagian yaitu *data entry, admin, customer service,* dan *pengemudi*. *Clerical operatives support* bertugas untuk mengurus kegiatan yang berhubungan langsung dengan penumpang, memasukan data penumpang, dan menedengarkan keluhan penumpang.

II.2 Landasan Teori

Sub-bab ini menjelaskan tentang apa saja teori yang relevan yang digunakan untuk menjelaskan variabel yang diteliti dan sebagai dasar untuk memberi jawaban sementara pada rumusan masalah yang diajukan. Teori yang dijelaskan pada sub-bab ini meliputi penjelasan kelelahan, kantuk, denyut jantung, Mi Band, bluetooth, dan android.

II.2.1 Kelelahan

Kelelahan adalah suatu mekanisme perlindungan tubuh agar tubuh terhindar dari kerusakan lebih lanjut sehingga terjadi pemulihan setelah istirahat. Kelelahan sering dihubungkan dengan kondisi kurang tidur, kondisi akubat tidur yang terganggu, atau kebutuhan kuat untuk tidur yang berhubungan dengan panjangnya waktu kerja, dan stres kerja yang bervariasi.

Kelelahan kerja akan menurunkan kinerja dan menambah tingkat kesalahan kerja. Meningkatnya kesalahan kerja akan memberikan peluang terjadinya kecelakaan kerja dalam industri. Pembebanan otot secara statispun (static muscular loading) jika dipertahankan dalam waktu yang cukup lama akan mengakibatkan RSI (Repetition Strain Injuries), yaitu nyeri otot, tulang, tendon, dan lain-lain yang diakibatkan oleh jenis pekerjaan yang bersifat berulang (repetitive) [5].

Ada beberapa pendapat mengenai tipe kelelahan akibat kerja, yang diantaranya adalah sebagai berikut [6] :

1. Kelelahan otot (muscular fatigue), disebabkan oleh aktivitas yang membutuhkan tenaga fisik yang banyak dan berlangsung lama. Tipe ini berhubungan dengan perubahan biokimia tubuh dan dirasakan individu dalam bentuk sakit yang akut pada otot. Kelelahan ini dapat dikurangi dengan mendesain prosedur kerja baru yang melindungi individu dari pekerjaan yang terlalu berat, misalnya dengan mendesain ulang peralatan atau penemuan alatalat baru serta melakukan sikap kerja yang lebih efisien.
2. Kelelahan mental (mental fatigue), berhubungan dengan aktivitas kerja yang monoton. Kelelahan ini dapat membuat individu kehilangan kendali akan pikiran dan perasaan, individu menjadi kurang ramah dalam berinteraksi dengan orang lain, pikiran dan perasaan yang seharusnya ditekan karena dapat menimbulkan konflik dengan individu lain menjadi

lebih mudah diungkapkan. Kelelahan ini diatasi dengan mendesain ulang pekerjaan sehingga membuat karyawan lebih bersemangat dan tertantang untuk menyelesaikan pekerjaan.

3. Kelelahan emosional (emotional fatigue), dihasilkan dari stres yang hebat dan umumnya ditandai dengan kebosanan. Kelelahan ini berasal dari faktor-faktor luar di tempat kerja, perusahaan dapat mengatasi kelelahan ini dengan memberikan pelayanan konseling bagi karyawan agar kelelahan emosional yang dirasakan karyawan dapat teratasi dan performansi kerja karyawan meningkat.
4. Kelelahan ketrampilan (skills fatigue), berhubungan dengan menurunnya perhatian pada tugas-tugas tertentu seperti tugas pilot atau pengontrol lalu lintas udara. Pada kelelahan tipe ini standar akurasi dan penampilan kerja menurun secara progresif. Penurunan ini diperkirakan menjadi penyebab utama terjadinya kecelakaan mobil dan pesawat terbang, sehingga karyawan harus selalu diawasi dan diupayakan agar terhindar dari kelelahan ini dengan pemberian waktu istirahat yang cukup.

II.2.2 Kantuk

Kantuk adalah kecenderungan untuk jatuh tertidur. Keadaan mengantuk dapat menurunkan performansi manusia, seperti berkurangnya kecepatan reaksi, penurunan kewaspadaan, dan menurunkan kemampuan memproses informasi. Oleh karena itu, keadaan mengantuk saat mengemudi tergolong dalam satu tindakan yang berbahaya [2]. Kantuk dapat mempengaruhi proses pada saat mengemudi dan performa dari pengemudi dapat turun hingga dibawah standar yang dibutuhkan oleh situasi lalu lintas.

Secara garis besar faktor yang mempengaruhi tingkat kantuk terutama pada saat mengemudi antara lain adalah [7]:

1. Kuantitas Tidur

   Tidur sebelum bekerja adalah faktor yang paling mempengaruhi keadaan sadar dan tingkat kewaspadaan pengemudi. Kurang tidur yang bersifat kronis merupakan akibat dari kurangnya waktu tidur dalam jangka waktu yang lama, sedangkan kurang tidur yang bersifat akut merupakan akibat dari kurangnya waktu tidur dalam waktu satu malam yang pendek. Kebutuhan jam tidur biasanya dipengaruhi oleh usia, semakin bertambah usia seseorang maka semakin berkurang jam tidurnya.
2. Kualitas Tidur

Kualitas tidur yang baik akan ditandai dengan tidur yang tenang, merasa sangat segar saat bangun tidur di pagi hari dan individu merasa penuh semangat untuk melakukan aktivitas hidup lainnya.

3. Status Gizi

   Hasil penelitian menyatakan bahwa Indeks Massa Tubuh (IMT) tertinggi dimiliki oleh responden yang memiliki durasi tidur yang kurang sering mengantuk. Penelitian lain menyatakan adanya peningkatan IMT seiring dengan berkurangnya durasi tidur.

4. Faktor Kesehatan

   Faktor kesehatan menurut National Transport Commission (2007) yang terkait dengan mengantuk saat berkendara antara lain adalah:

   a. Sleep apnea terjadi bila tenggorokan runtuh saat tidur sehingga terlalu sedikit mencapai paru-paru, sehingga sering bangun karena kekurangan oksigen.
   b. Narkolepsi terjadi bila seseorang tidur di siang hari tanpa jelas dan mengalami masalah tidur di malam hari.
   c. Beberapa orang juga menderita kegelisahan di malam hari yang membuat mereka terjaga.
   d. Beberapa penyakit seperti diabetes, jika tidak dikontrok, bisa juga mengakibatkan kelelahan.
   e. Kelebihan berat badan atau obesitas biasanya tidak dianggap sebagai penyakit. Namun, obesitas dapat menyebabkan masalah tidur dan sangat memberikan kontribusi untuk terjadinya sleep apnea.
   f. Kelelahan dan kantuk setelah tidur yang cukup dapat menunjukkan masalah medis.

5. Usia

   Penelitian membuktikan bahwa walaupun shift berhubungan dengan tidur, mengantuk subyektif, kinerja dan kehidupan social, tetapi umur hanya berpengaruh pada perubahan banyaknya tidur, rasa mengantuk subyektif dan kewaspadaan psikomotorik.

6. Masa Kerja

   Pengemudi yang professional atau dengan masa kerja yang lebih lama, karena sifat dari pekerjaannya telah menghabiskan sebagian besar jam kerja mereka untuk mengendarai kendaraan. Selain jangka waktu mengemudi, pengemudi yang professional banyak bekerja dalam shift, memiliki jam kerja tidak teratur, dan bekerja pada malam hari.

7. Shift Kerja

   Shift kerja adalah jadwal kerja dengan jam kerja pekerja yang berbeda dengan jam kerja standar (jam 8 pagi hingga 5 sore) atau jadwal kerja yang berbeda dari jadwal kerja

mingguan standar hari senin hingga hari jumat di Amerika Serikat. Shift kerja merupakan sebuah istilah yang menjelaskan jadwal kerja yang tidak standar, berhubungan dengan gangguan fisiologi yang mendasari individu. Shift kerja memaksa pekerja untuk mengubah jam tidurnya berdasarkan waktu kerja dan waktu istirahat, sehingga hal ini dapat mengganggu siklus normal dari aktivitas fisik.

8. Pola Kerja / Rotasi Shift

   Hal yang haru diperhatikan perusahaan dalam memilih rotasi shift baik pendek maupun panjang adalah sedikitnya terdapat satu minggu untuk irama sirkadian beradaptasi dengan perubahan jadwal. Rotasi shift yang panjang terdiri dari rentang bekerja shift malam selama 4-6 minggu, maka seseorang pekerja mungkin akan perlu bekerja malam selama 2 kali dalam setahun. Sedangkan, rotasi shift yang pendek akan melibatkan tidak lebih dari 3 shift malam dalam waktu yang bersamaan.

9. Durasi Mengemudi

   Durasi mengemudi telah diketahui sejak lama sebagai salah satu faktor resiko terpenting pada kejadian jatuh tertidur saat mengemudi. Meskipun durasi mengemudi bukan merupakan faktor yang krusial, sebagai contoh 60% dari kecelakaan yang berakibat fatal pada pengemudi terjadi dalam waktu 1 jam mengemudi.

II.2.3 Denyut Nadi

Denyut nadi adalah frekuensi irama denyut/detak jantung yang dapat dipalpasi (diraba) dipermukaan kulit pada tempat-tempat tertentu. Siklus jantung terdiri dari periode relaksasi yang dinamakan diastole dan diikuti oleh periode kontraksi yang dinamakan systole. Kekuatan darah masuk ke dalam aorta selama sistolik tidak hanya menggerakkan darah dalam pembuluh ke depan tetapi juga menyusun suatu gelombang tekanan sepanjang arteri. Gelombang tekanan mendorong dinding arteri seperti berjalan dan pendorongnya teraba sebagai nadi. Urutan normal bagian-bagian jantung yang berdenyut yaitu kontraksi atrium (sistolik atrium) diikuti oleh kontraksi vertikel (sistolik vertikel) dan selama diastolik keempat ruangan relaksasi.

Kecepatan denyut jantung adalah besaran yang berupa frekuensi denyut jantung dan memiliki satuan *beats per minute* (bpm). Besaran dengan satuan bpm menyatakan jumlah denyut jantung setiap menitnya. Kecepatan denyut jantung normal manusia adalah 60-100 bpm. [2]

II.2.4 Mi Band 2

Mi Band 2 merupakan merek *smartwatch* yang dikeluarkan oleh Xiaomi. Fitur yang digunakan pada *smartwatch* ini ada beberapa yaitu [3]:

1. *Water Resistant* (anti air) dengan spesifikasi IP-67 yang juga tahan dengan debu.
2. Memiliki sensor detak jantung
3. *Sport Tracker* yang dapat memperlihatkan aktivitas pengguna
4. *Sleep tracking* yang dapat mendeteksi seberapa besar kualitas tidur pengguna.

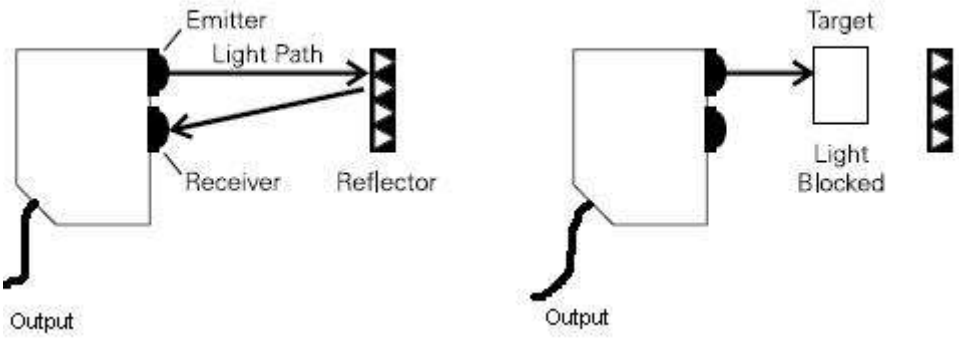
II.2.5 Bluetooth

Bluetooth adalah sebuah teknologi komunikasi *wireless* (tanpa kabel) yang beroperasi dalam pita frekuensi 2.4 GHz unlisenced ISM (Industrial, Scientific and Medical) dengan menggunakan sebuah *frequency hopping transceiver* yang mampu menyediakan layanan komunikasi data dan suara secara real-time antara host-host bluetooth dengan jarak jangkauan layanan terbatas. Bluetooth 4.0 adalah teknologi Bluetooth versi terbaru dengan keunggulan utama hemat energy yang resmi diadopsi oleh Bluetooth *Special Interest Group* (SIG) pada tahun 2010 [8]. Bluetooth 4.0 memiliki beberapa keunggulan yaitu:

1. Konsumsi daya yang lebih kecil sehingga hemat baterai atau listrik
2. Waktu pemakaian yang lebih lama sampai tahunan dengan hanya menggunakan baterai kapasitas kecil (*coin cell batteries*).
3. Biaya produksi yang rendah.
4. Jarak yang lebih panjang. Mampu menjangkau lebih dari 100 meter.
5. Kecepatan transfer yang lebih baik dengan kemampuan sekitar 1 Mbps.
6. Sistem keamanan yang lebih terkontrok dibandingkan dengan sistem keamanan pada Bluetooth generasi-generasi sebelumnya.

II.2.6 Sensor Photoelectric

Sensor Photoelectric Hokuyo PE-H2TY adalah sensor yang bekerja dengan prinsip seperti transistor sebagai saklar. Energi cahaya akan diubah menjadi suatu sinyal listrik. Adanya suatu reflector yang berfungsi untuk memantulkan cahaya yang dipancarkan oleh Photoelectric Hokuyo PE-H2TY. Karakteristik yang dimiliki adalah sebagai berikut:



Gambar II.3 Prinsip Kerja Sensor Photoelectric

1. Beroperasi pada catu tegangan: 12 Volt – 24 Volt DC.
2. Arus yang dikonsumsi maksimal 20 miliampere
3. Sumber cahaya yang digunakan adalah LED merah
4. Memiliki penguat sendiri (diatur dengan potensiometer)
5. Jarak pendeteksian 10 – 100 mm
6. Waktu respon yang dimiliki 1 milidetik On dan 1 milidetik Off
7. Tegangan ripple harus kurang 10% dari tegangan sumber.

II.2.7 Photoplethysmograph (PPG)

Photoplethysmograph (PPG) adalah suatu perangkat yang digunakan untuk mengukur kondisi peredaran darah yang dipompa oleh jantung pada organ tertentu dalam tubuh. Hasil pengukuran tersebut dapat dimanfaatan untuk mengetahui kondisi jantung dalam periode tertentu. Selain untuk mengamati kinerja jantung, PPG juga dapat dimanfaatkan untuk memonitor pernafasan dan mengatur saturasi oksigen dalam darah.

II.2.8 *Volley Library*

Volley Library merupakan produk yang diperkenalkan oleh Google untuk mempermudah pertukaran data tanpa harus membuat deretan kode yang sangat panjang. Secara default volley menggunakan metode singkronisasi yang menyebabkan tidak perlunya membuat sebuah method atau fungsi yang menggunakan class asyntask. Fitur-fitur yang disediakan oleh volley library adalah sebagai berikut [9]:

a. Melakukan sebuah request dalam pengutamaan prioritas dalam sebuah antrian.
b. Sangat efektif untuk melakukan chace efisiensi penyimpanan (memory)
c. Dapat melakukan perubahan class sesuai dengan kebutuhan
d. Dapat melakukan pembatalan dalam sebuah request API

e. Secara otomatis menjadwal permintaan jaringan

f. Menyeduakan debugging dan tracing tools.

II.2.9 Google Map API

Untuk memuat Google Maps JavaScript API, gunakan tag `script` seperti yang ada dalam contoh berikut [10]:

```
    <script async defer
      src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&
callback=initMap">
    </script>
```

URL yang terdapat dalam tag `script` adalah lokasi file JavaScript yang memuat semua simbol dan definisi yang dibutuhkan untuk menggunakan Maps JavaScript API. Atribut `async` memungkinkan browser merender bagian selebihnya dari situs web Anda saat Maps JavaScript API dimuat. Bila API sudah siap, ia akan memanggil fungsi yang ditetapkan menggunakan parameter `callback`.

Agar peta ditampilkan pada laman web, hal pertama yang harus dilakukan adalah memesan tempat. Hal ini dengan cara membuat elemen bernama `div` dan memperoleh referensi ke elemen ini dalam model objek dokumen (DOM) browser. Ada dua opsi yang diperlukan untuk setiap peta: `center` dan `zoom`.

```
map = new google.maps.Map(document.getElementById('map'), {
  center: {lat: -34.397, lng: 150.644},
  zoom: 8
});
```

Resolusi awal untuk menampilkan peta disetel oleh properti `zoom`, dalam hal ini zoom `0` menyatakan peta bumi yang diperkecil maksimum, dan tingkat zoom yang lebih besar akan memperbesar dengan resolusi lebih tinggi. Menetapkan tingkat zoom sebagai integer.

```
zoom: 8
```

Menampilkan peta seluruh bumi sebagai gambar tunggal akan memerlukan peta yang teramat besar, atau peta kecil dengan resolusi sangat rendah. Akibatnya, gambar peta dalam Google Maps dan Maps JavaScript API akan dipecah menjadi "petak" peta dan "tingkat zoom." Pada tingkat zoom rendah, satu rangkaian kecil petak peta mencakup area yang luas; di tingkat zoom yang lebih tinggi, petak memiliki resolusi lebih tinggi dan mencakup area yang lebih

kecil. Daftar berikut menunjukkan perkiraan tingkat detail yang diharapkan untuk terlihat di setiap tingkat zoom:

1. 1: Dunia
2. 5: Daratan luas/benua
3. 10: Kota
4. 15: Jalan
5. 20: Bangunan

II.2.10 GPS (*Global position system*)

Global position system adalah sistem navigasi berbasis satelit terdiri dari jaringan 24 satelit ditempatkan ke orbit oleh Departemen Pertahanan Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978. Layanan GPS dahulu hanya dipergunakan untuk keperluan militer namun mulai terbuka untuk publik. 24 satelit GPS tersebut berada sekitar 12.000 mil di atas bumi bergerak mengelilingi bumi 12 jam dengan kecepatan 7.000 mil per jam. Satelit GPS berkekuatan energi sinar matahari, memiliki baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak ada energi matahari dan memiliki roket penguat kecil pada masing-masing satelit agar dapat mengorbit tepat pada tempatnya.

Satelit-satelit GPS harus selalu berada pada posisi orbit yang tepat untuk menjaga akurasi data yang dikirim ke GPS *receiver*, sehingga harus selalu dipelihara agar posisinya tepat. Posisi satelit-satelit tersebut selalu dipantau oleh stasiun pengendali. Stasiun-stasiun pengendali di bumi ada di Hawaii, Ascension Islan, Diego Garcia, Kwajalein dan Colorado Spring. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS *receiver* yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. GPS *receiver* mengambil informasi tersebut dan melakukan perhitungan triangulation untuk menentukan lokasi pengguna dengan tepat. GPS receiver membandingkan waktu sinyal dikirim dengan waktu sinyal tersebut diterima untuk mengetahui jarak satelit. Dengan mengetahui jarak tersebut, GPS receiver dapat melakukan perhitungan dan menentukan posisi pengguna dan menampilkan dalam peta elektronik. Setelah menentukan posisi pengguna, 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 sangat presisi dalam mengirim informasi karena satelit tersebut menggunakan jam atom. Jam atom yang ada pada satelit merupakan partikel atom yang diisolasi, sehingga dapat menghasilkan jam yang akurat dibandingkan dengan jam biasa. Keistimewaan GPS adalah mampu bekerja dalam berbagai kondisi cuaca, siang atau malam.

Keakuratan sebuah perangkat GPS bisa mencapai 15 meter, bahkan model terbaru yang dilengkapi teknologi Wide Area Augmentation System (WAAS) keakuratannya sampai 3 meter. Karena GPS bekerja mengandalkan satelit, maka penggunaannya disarankan di tempat terbuka. Penggunaan di dalam ruangan, atau di tempat yang menghalangi arah satelit (di angkasa), maka GPS tidak akan bekerja secara akurat dan maksimal. Perhitungan waktu yang akurat sangat menentukan akurasi perhitungan untuk menentukan informasi lokasi. Semakin banyak sinyal satelit yang dapat diterima maka akan semakin presisi data posisi yang dihasilkan. Selain itu, ketinggian juga mempengaruhi proses kerja GPS, karena semakin tinggi maka semakin bersih atmosfer, sehingga gangguan semakin sedikit [11].

Sinyal yang diterima oleh alat penerima (receiver) di permukaan, dimana GPS receiver ini akan mengumpulkan informasi dari satelit GPS [11], 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.

II.2.11 Android

Android adalah sebuah sistem operasi pada handphone yang bersifat terbuka dan berbasis pada sistem operasi Linux [12] Android memiliki beberapa versi yang salah satunya adalah Android 5.0 yang pertama kali diperkenalkan di bawah codename “Android L” pada 25 Juni 2014 selama presentasi keynote pada konferensi pengembang Google I/O. Di samping Lollipop, presentasi difokuskan pada sejumlah platform Android yang berorientasi dan teknologi baru, termasuk Android TV, pada platform Android Auto, dapat dipakai pada platform komputasi Android Wear, dan platform pelacakan kesehatan Google Fit.

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada. Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

1. Activities

   Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

   Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack*. Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* Android menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

   Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi:

   a. *Active*

      Setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala cara, bahkan akan

menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.

b. *Paused*

Dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused*. Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, Android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.

c. *Stopped*

Ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped*. *Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antarmuka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive*.

d. *Inactive*

Kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali. Kondisi transisi ini sepenuhnya ditangani oleh manajer memori Android. Android akan memulai menutup aplikasi yang mengandung *activity inactive*, kemudian *stopped activity*, dan dalam kass luar biasa *paused activity* juga akan ditutup.

2. Services

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

3. Intents

Intents merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman *web*, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit,

contohnya jika suatu aplikasi ingin menampilkan *URL*, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

4. Broadcast Receiver

   Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service*, *Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

5. Content Providers

   Content Providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam file sistem, dalam database *SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.

II.2.12 Java

Java adalah dimana sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada computer standalone ataupun pada lingkungan jaringan [13]. Java Virtual Machine (JVM) adalah sebuah mesin interpreter yang akan membaca bytecode dalam file .class dari suatu program sebagai representasi langsung dari program yang berisi bahasa mesin. Oleh karena itu, bahasa java disebut sebagai bahasa pemrograman yang portable karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM. Adapun beberapa kelebihan dari bahasa pemrograman Java adalah sebagai berikut:

1. Sederhana dan Ampuh

   Java dirancang untuk mudah dipelajari, terutama bagi programmer yang telah mengenal C/C++ akan mudah sekali untuk berpindah ke Java. Pemakai dapat belajar membuat program dengan Java secara cepat jika telah memahami konsep dasar pemrograman berorientasi objek. Java tidak memiliki hal-hal yang mengejutkan dan aneh. Java memberi anda kemampuan untuk menuangkan semua ide, karena bahasa pemrograman ini bukan merupakan scripting language (bahasa naskah) yang menghilangkan kemampuan kita untuk berinovasi, tetapi dengan cara berorientasi objek yang mudah dan jelas.

2. Aman

Java dirancang sebagai bahasa pemrograman yang handal dan aman. Aplikasi-aplikasi yang dibangun dengan bahasa Java sangat handal dengan manajemen memori yang bagus. Aplikasi Java juga dikenal sangat secure, yaitu kasus-kasus seperti buffer everflow yang umumnya menjadi lubang keamanan aplikasi-aplikasi berbasis C/C++ tidak terjadi di Java, karena pengaturan securitynya yang bagus.

3. Berorientasi Objek

   Paradigma pemrograman berorientasi objek merupakan paradigma pemrograman masa depan. Java merupakan bahasa pemrograman berorientasi objek. Java bukan turunan langsung dari bahasa pemrograman manapun, juga sama sekali tidak kompetibel dengan semuanya. Java memiliki keseimbangan, Java secara cepat jika telah memahami konsep dasar pemrograman berorientasi objek. Java tidak memiliki hal-hal yang mengejutkan dan aneh. Java memberi anda kemampuan untuk menuangkan semua ide, karena bahasa pemrograman menyediakan mekanisme peng-class-an sederhana, dengan model antar muka dinamik yang intuitif hanya jika diperlukan.

4. Kokoh

   Java membatasi anda dari beberapa hal kunci supaya anda dapat menemukan kesalahan lebih cepat saat mengembangkan program. Java langsung memeriksa program saat anda menuliskannya, dan sekali lagi ketika program di jalankan. Karena Java adalah bahasa yang sangat ketat dalam hal tipe data dan deklarasi, banyak kesalahan umum terjadi saat kompilasi. Hal ini akan lebih menghemat waktu jika dibandingkan dengan keharusan menjalankan program terlebih dahulu dan memeriksa semua bagian program untuk melihat ketidakcocokan dinamis selama program berjalan. Ini adalah contoh di mana Java lebih luwes dan kokoh dari beberapa bahasa lain, tetapi dengan imbalan yang layak untuk kelebihan itu.

5. Interaktif

   Java memiliki beberapa kemampuan yang memungkinkan program melakukan beberapa hal pada saat bersamaan, tanpa harus kesulitan menangani proses yang akan terjadi selanjutnya. Jalinan program-program Java yang mudah digunakan memungkinkan kita untuk memikirkan pembuatan perilaku khusus, tanpa harus mengintegrasikan perilaku tersebut dengan model pemrograman global yang mengatur perulangan kejadian.

6. Netral terhadap Berbagai Arsitektur

   Java telah mengambil beberapa keputusan yang sulit dalam pembuatan bahasa Java dan bagaimana program dijalankan, jadi anda dapat sepenuhnya percaya “tulis sekali, jalan di mana saja, kapan saja, selamanya”.

7. Terinterpretasi dan Berkinerja Tinggi

   Java dilengkapi keajaiban lintas platform yang luar biasa dengan kompilasi ke dalam representasi langsung yang disebut kode-byte Java (Java byte-code), yang dapat diterjemahkan oleh sistem manapun yang memilki program Java didalamnya. Java, bagaimanapun dirancang untuk tetap berkinerja baik pada CPU yang tidak terlalu kuat. Walaupun Java merupakan bahasa terinterpretasi, kodekode Java telah dirancang dengan hati-hati sehingga mudah diterjemahkan ke dalam bahasa asli suatu mesin untuk menghasilkan kinerja yang tinggi.

II.2.13 PHP

PHP adalah suatu bahasa *server-side* yang didesain khusus untuk aplikasi web. PHP dapat disisipkan diantara HTML, dan karena bahasa *server-side*, maka bahasa PHP akan dieksekusi di server, yang dikirimkan ke browser adalah "hasil jadi" bentuk HTML, dan kode PHP tidak akan terlihat. PHP dahulu merupakan proyek pribadi dari Rasmus Lerdorf tahun 1994 (dengan dikeluarkannya PHP versi 1 yang digunakan untuk membuat home page pribadi yang dipublikasikan 8 juni 1995). Versi pertama ini berupa kumpulan script PERL. Untuk versi kedua Ramus menulis ulang script-script PERL tersebut menggunakan bahasa C, kemudian menambahkan fasilitas untuk form HTML dan koneksi MYSQL. Adapun PHP didapat dari singkatan *PHP Hypertext Preprocessor*.

Setelah mengalami perkembangan, Zeev Suraski dan Andi Gutmans (termasuk Rasmus) yang tergabung dalam suatu kelompok open source, maka mereka memulai versi 3 PHP nya, PHP telah menampakkan keunggulan sebagai salah satu bahasa *server scripting* yang handal. Melalui perkembangan yang pesat ini banyak fasilitas yang ditambahkan oleh kelompok ini PHP disebut sebagai "PHP: Hypertext Preprocessor". Pada bulan Mei 2000 dirilis PHP versi 4 yang dikeluarkan oleh Zend Engine yaitu sebuah produk yang diciptakan oleh Suraski dan Gutmans. Kehandalan PHP diperlengkap dengan dirilisnya PHP versi 5 pada 13 Juli 2004 yang mendukung OOP serta berbagai manajemen database yang luas dan penanganan error yang mudah. Sintak yang digunakan berasal dari bahasa C, Java maupun PERL. Sampai tulisan ini dibuat versi PHP yang terbaru adalah versi 5.2.0 yang dirilis pada November 2006.

PHP merupakan bahasa script yang digunakan untuk membuat halaman web yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu yang terbaru. Semua script PHP dieksekusi pada server dimana script tersebut dijalankan. Oleh karena itu, spesifikasi server lebih berpengaruh pada eksekusi dari script php daripada spesifikasi client.

Namun tetap diperhatikan bahwa web yang dihasilkan tentunya harus dapat dibuka oleh browser pada client. Dalam hal ini versi dari html yang digunakan harus didukung oleh browser client. PHP termasuk dalam *Open Source Product*. Jadi dapat menambah *source code* dan dapat mendistribusikannya secara bebas. PHP juga diedarkan secara gratis. Kita bisa mendapatkannya secara gratis. PHP juga dapat berjalan diberbagai web server semisal IIS, Apache, PWS.

Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh *user* sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membuat halaman web yang dinamis, yaitu halaman web yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman web [14].

II.2.14 Pemrograman Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem. Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan ibjek yang berkorespondensi dengan objek-objek dunia nyata.

Ada banyak cara untuk mengabstarksikan dan memodelkan objek-objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek yang akan dienkapsulasikan menjadi suatu kesatuan. Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu [15]. Berikut ini adalah beberapan konsep dasat yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*Class*)

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama, kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (model/operasi), hubungan (relationship) dan arti. Suatu kelas dapat diturunkan dari kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara teknis kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.
2. Objek (*Object*)

Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata, benda, manusia, satuan, organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi dan mempunyai operasi yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan. Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dilihat dari segi teknis adalah elemen yang pada saat runtime yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehingga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk kode, disebut sebagai kelas.

3. Metode (*Method*)

   Operasi atau metode pada sebuah kelas hampis sama dengan dungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode berfungsi untuk memanipulasi objek itu sendiri. Metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Metode dapat berasal dari event, aktifitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata.

4. Atribut (*Attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*Abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*Encapsulation*)

   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya.

7. Pewarisan (*Inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*Interface*)

Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antarobjek

    Komunikasi antarobjek dilakukan lewat pesan (message) yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*Polymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

13. *Package*

    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam package yang berbeda.

II.2.15 UML (*Unified modeling language*)

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual [16]. Sejarah UML sendiri terbagi dalam dua fase; sebelum dan sesudah munculnya UML. Dalam fase sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan desain berbeda-beda, sehingga dapat dikatakan belum memiliki standarisasi. mempersatukan metode tersebut dan dimotori oleh Object Management Group (OMG) maka pengembangan UML dimulai pada akhir tahun 1994 ketika Grady Booch dengan metode OOD (Object-Oriented Design), Jim Rumbaugh dengan metode OMT (Object Modelling Technique) mereka ini bekerja pada Rasional *Software* Corporation dan Ivar Jacobson dengan metode OOSE (Object-Oriented *Software* Engineering) yang bekerja pada perusahaan Objectory Rasional.

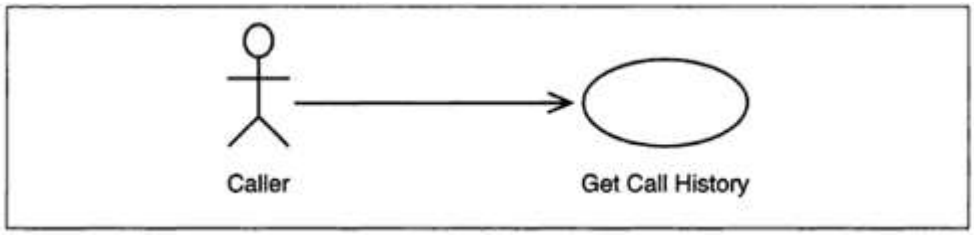
Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan Object Oriented karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik.

II.2.15.1.1 *Use case* Diagram

Diagram yang menggambarkan actor, *use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur untuk aktor. Sebuah *use case* digambarkan sebagai elips horizontal dalam suatu diagram UML *use case*. *Use case* memiliki dua istilah yaitu:

1. System *use case*; interaksi dengan sistem.
2. Business *use case*; interaksi bisnis dengan konsumen atau kejadian nyata.

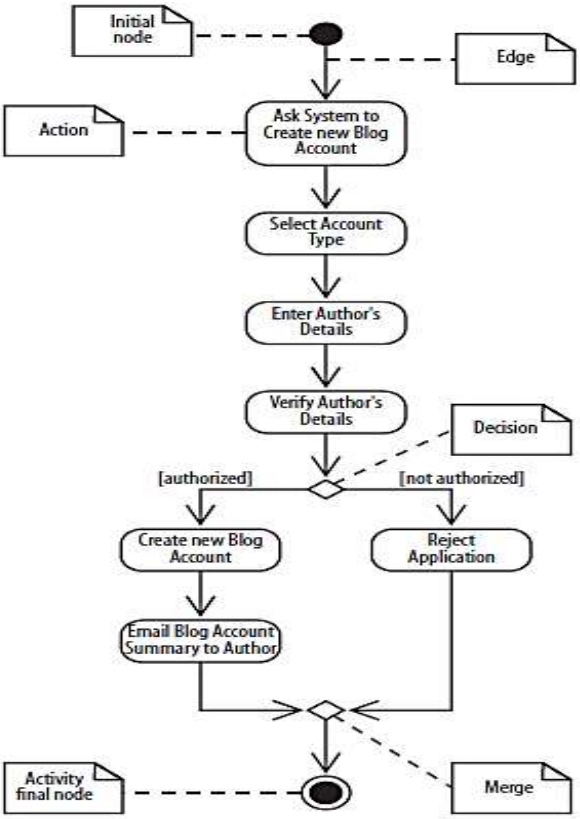
Berikut ini merupakan contoh dari *use case* diagram yang dapat dilihat pada Gambar II.4 Contoh Usecase diagram sederhana.



Gambar II.4 Contoh Usecase diagram sederhana

II.2.15.1.2 *Activity diagram*

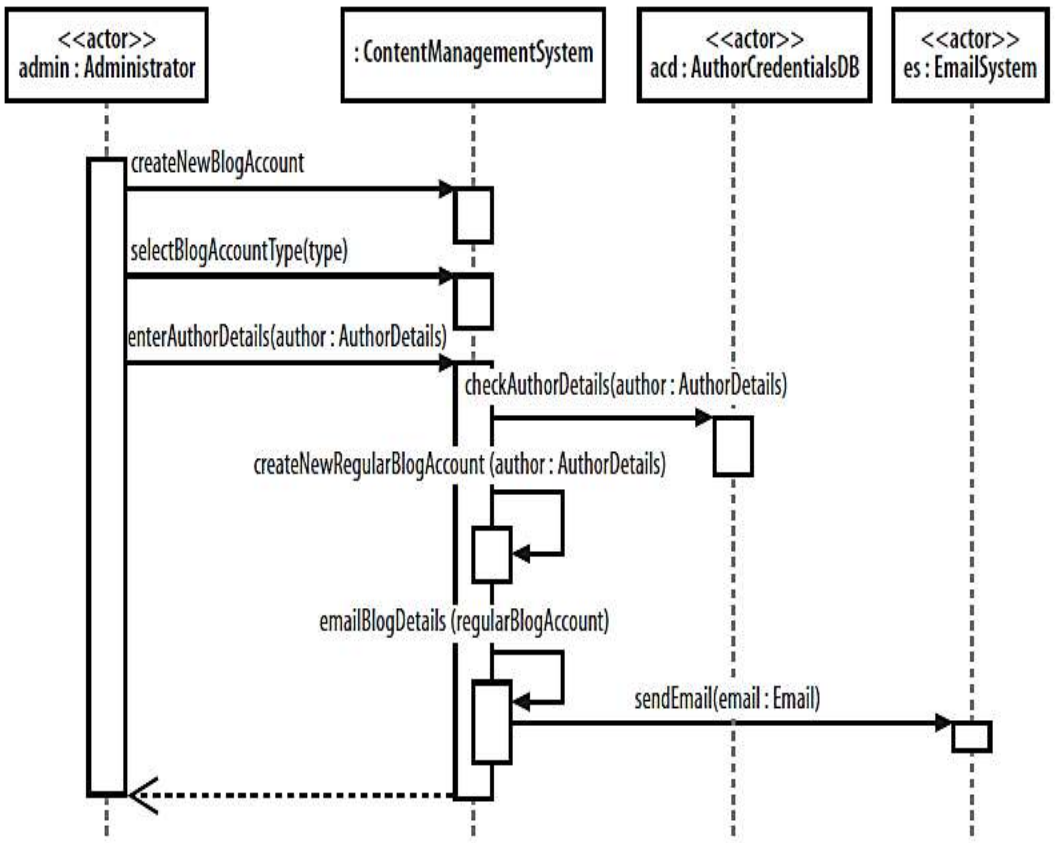
Activity diagram menggambarkan aktifitas-aktifitas, objek, state, transisi state dan event. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas. Gambar II.5 Contoh Bentuk *Activity diagram* merupakan contoh gambar *activity diagram*:



Gambar II.5 Contoh Bentuk *Activity diagram*

II.2.15.1.3 *Sequence diagram*

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram. Contoh dari *sequence diagram* dapat dilihat pada Gambar II.6 Contoh Bentuk *Sequence diagram*.



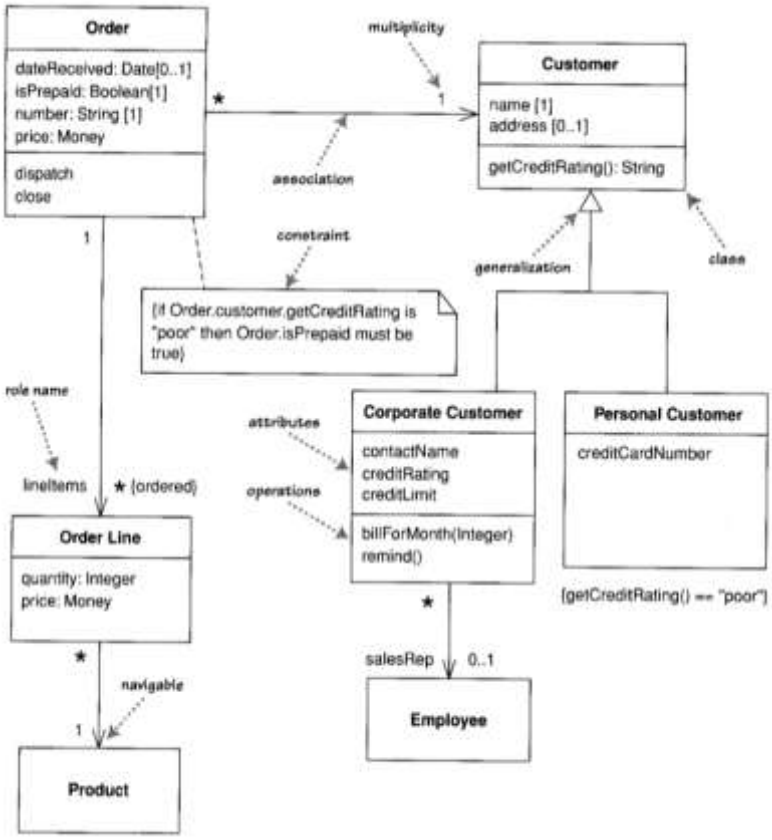
Gambar II.6 Contoh Bentuk *Sequence diagram*

II.2.15.1.4 *Class diagram*

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti *public*, *private* dan *protected.* Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. Class memiliki tiga area pokok:

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Contoh dari *class diagram* dapat dilihat pada Gambar II.7 Contoh Bentuk *Class diagram.*



Gambar II.7 Contoh Bentuk *Class diagram*

II.2.16 Geocoding API

Geocoding adalah salah satu fitur dari Google Maps API yang merupakan proses untuk mencari koordinat geografis (latitude dan longitude atau garis lintang dan bujur) dari alamat tertentu (nama jalan, nama kota, kode pos, nama negara, dll) [17]. Kebalikan dari Geocoding adalah reverse Geocoding, yaitu mencari data geografis berdasarkan koordinat geografis. Jika ada suatu koordinat yang dimasukan lalu akan muncul data geografis yang ada pada koordinat itu seperti nama jalan, kode pos, negara, kota, provinsi, dsb. Google Geocoding API merupakan suatu web service yang bisa diakses menggunakan HTTP Request biasa, hanya melalui URL.

Parameter yang dibutuhkan ketika memanggil Geocoding adalah sebagai berikut:

1. Address: alamat yang ingin digunakan, format alamat yang digunakan harus berdasarkan yang telah ditentukan oleh masing – masing negara. Tambahan alamat seperti nama unit, suite atau lantai lebih baik dihindari.

2. Components: sebuah filter komponen yang ingin kamu dapatkan dengan menggunakan Geocode.
3. Key: Key aplikasi. Key ini mengidentifikasi aplikasi untuk tujuan manajemen kuota.

II.2.17 Distance Matrix API

Google Maps Distance Matrix API adalah layanan yang menyediakan waktu dan jarak perjalanan untuk matriks tempat asal dan tujuan, berdasarkan rute yang direkomendasikan di antara titik awal dan akhir [18]. Dengan menggunakan Distance Matrix API dapat diperoleh informasi sebagai berikut:

1. Durasi dan Jarak: Perkiraan waktu dan jarak yang disarankan
2. Moda Transportasi: Tetapkan moda transportasi yang akan digunakan dalam perhitungan jarak dan waktu tempuh.

Lalu Lintas: Gunakan lalu lintas yang lalu dan saat ini untuk memperkirakan waktu tempuh.