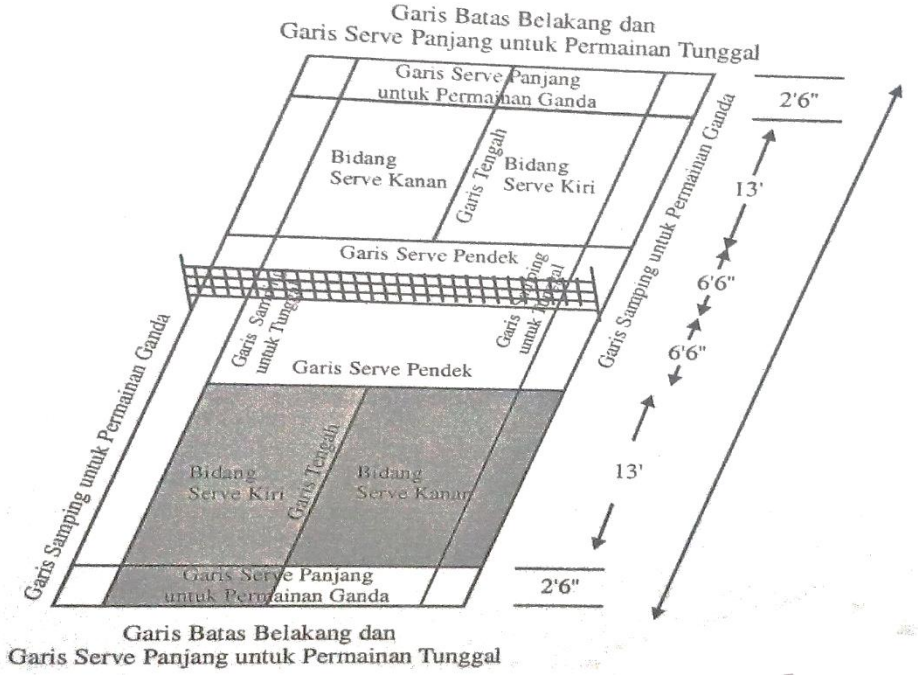
BAB 2
TINJAUAN PUSTAKA

2.1 *Badminton*

Bulu tangkis atau *badminton* pertama dimainkan di tempat kediaman *Duke of Beaufort* yang berada di kota badminton di Gloucestershire. Peralatan utama dalam bermain bulu tangkis adalah raket dan *shuttlecock*. Dalam permainannya, pukulan *service* merupakan pukulan pertama yang mengawali suatu permainan bulu tangkis. Baik pelaku *service* atau penerima *service* harus berdiri di bidang *service* yang ditentukan. Pada prinsipnya, permainan bulu tangkis dapat dilakukan baik di dalam maupun di luar ruangan. Namun semua turnamen resmi sampai saat ini dilakukan di dalam ruangan. Semua garis batas lapangan bulu tangkis, dibuat dengan ketebalan/lebar 3,8 cm. Jaring yang melintang di tengah lapangan, yang membatasi kedua sisi lapangan, terbuat dari bahan katun atau nilon. Tinggi jaring yaitu 155 cm di tiang dan 152 cm di tengah lapangan [1].



Gambar 2.1 Lapangan Bulu tangkis [1]

2.2 Aplikasi

Aplikasi adalah penggunaan dalam suatu komputer,instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output*. Menurut Kamus Kamus Besar Bahasa Indonesia, aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer. Program merupakan kumpulan *instruction set* yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini.

Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada instructionset. Program aplikasi merupakan program siap pakai yang dirancang untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh aplikasi ialah program pemproses kata dan Web Browser. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan app.

Secara historis, aplikasi adalah *software* yang dikembangkan oleh sebuah perusahaan. App adalah *sofware* yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang parallel dengan yang terjadi antar sistem operasi yang dimunculkan [6].

2.3 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/smartphone. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Aliance yang merupakan konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan open source pada perangkat mobile .

Di dunia terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Service (GMS) dan kedua adalah yang benar – benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD). Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis smartphone yang menggunakan Android sebagai sistem operasinya. Telepon selular ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam progam kerja Android ARM Holdings, Atheros Communication, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka Android, perangkat mobile yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan bug dan penambahan fitur baru [7].

2.3.1 Android SDK (*Software Development Kit*)

Android SDK adalah tools API (Application Programming Interface) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan aplikasi kunci yang di-release oleh Google. Saat ini disediakan Android SDK (Software Development Kit) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan [7].

2.3.2 Arsitektur Android

Secara Garis besar arsitektur android dapat dijalankan dan digambarkan sebagai berikut [7] :

1. Application dan Widgets

Application dan Widgets adalah layer dimana user berhubungan dengan aplikasi saja, dimana biasanya user men-download aplikasi, melakukan instalasi dan menjalankan aplikasi.

2. Application Frameworks

Android adalah “Open Development Platform” yaitu Android menawarkan kepada pengembang atau member kemampuan untuk membangun aplikasi yang inovatif. Pegembang bebas untu mengakses perangkat keras, akses informasi *resource,* menjalankan *service background*, mengatur alarm, dan menambahkan status notifikasi. Komponen-komponen yang termasuk di dalam *applications frameorks* adalah sebagai berikut :

a. *Views*
b. *Content provider*
c. *Resource manager*
d. *Notification manager*

e. *Activity manager*

3. Libraries

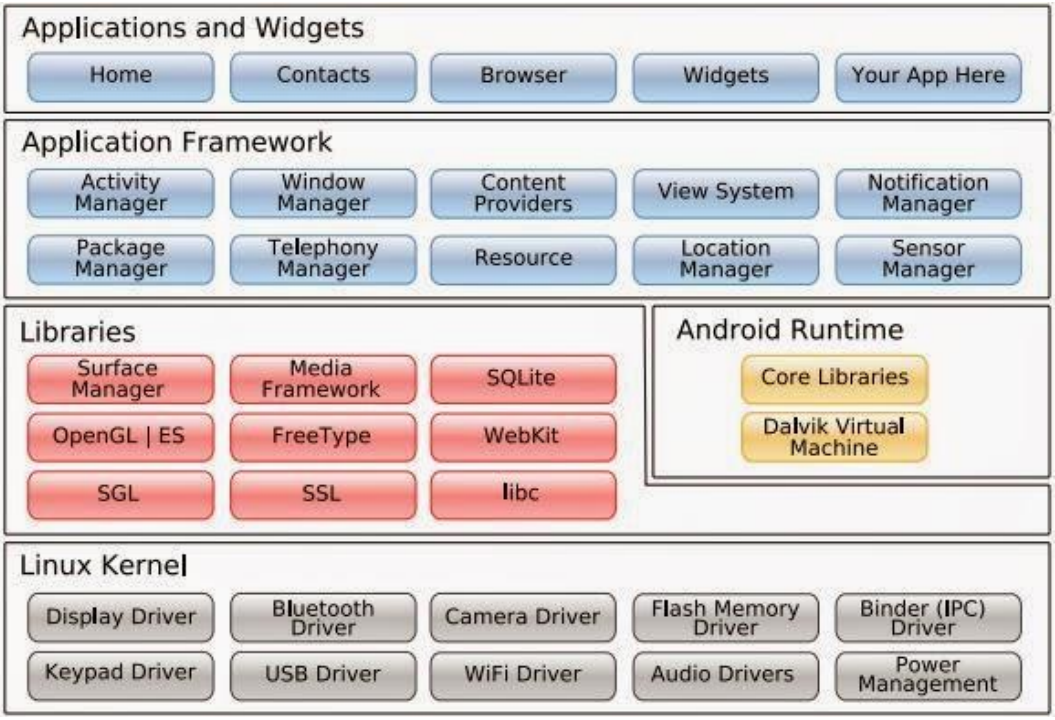
Libraries adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya.

4. Android Runtime

Layer yang membuat aplikasi Android dapat djalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik virtual machine (DVM)* merupakan mesin yang membentuk dasar kerangka aplikasi Android.

5. Linux Kernel

Linux Kernel adalah layer dimana inti dari operating system dari Android itu berada. Berisi *file-file* system yang mengatur sistem *processing, memory, recource, drivers,* dan sistem-sistem operasi android lainnya.



Gambar 2.2 Arsitektur Android [7]

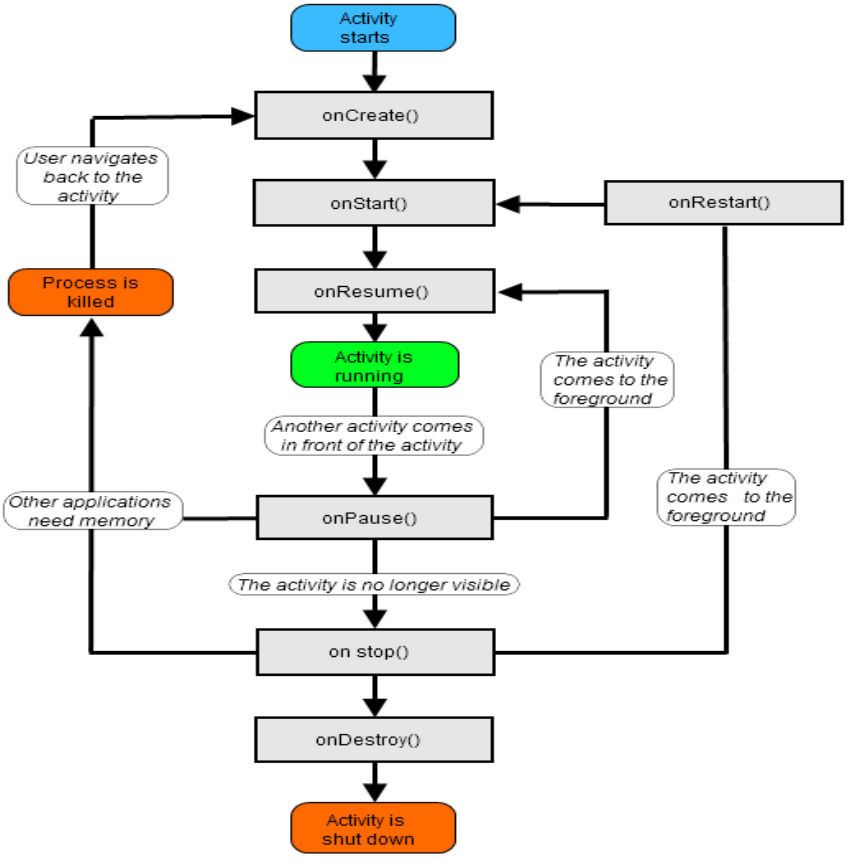
2.3.3 Komponen Android

Sebuah aplikasi android akan terdiri atas beberapa komponen penting sebagai berikut [8]:

1. *Activity*

   Activity merupakan layar tampilan pada sebuah aplikasi android. Secara sederhananya bahwa *activity* adalah layar yang pengguna dapat melihatnya. Sebuah aplikasi android dapat terdiri atas beberapa *activity* yang dapat dipindah dalam rentang waktu tertentu ketika menjalankan aplikasi. *Activity* ini merupakan komponen aplikasi yang umum dan sering berhubungan dengan tampilan, dimana user akan berinteraksi dengan aplikasi yang dibuat.

   Skema *lifecycle* dari sebuah *activity* adalah sebagai berikut :



Gambar 2.3 *Lifecycle Activity* [8]

a. Ketika sebuah *activity* dipanggil, maka kode yang ada pada method onCreate() akan dijalankan. Dan method ini juga akan memanggil satu parameter informasi status terakhir yang telah disimpan oleh method onSaveInstanceState().
b. Method onStart() digunakan untuk mengeksekusi *activity* ketika sebuah *user interface* ditampilkan.
c. Method onPause() akan dipanggil ketika ada *activity* lain yang akan mneggantikan *activity* yang ada.
d. Method onStop() digunakan ketika aplikasi sudah tidak dijalankan atau tidak dibutuhkan untuk sementara waktu.
e. Method onRestart() digunakan ketika *activity* di-restart dari posisi semula onStop().
f. Method onDestroy() akan dijalankan setelah *activity* aplikasi di-destroy atau ketika habis memori sehingga aplikasi akan di *terminate.*

2. *Views*

   Views merupakan *widget interface* atau antarmuka pada tampilan aplikasi android, misalnya seperti *tombol*, *input text*. *Views* akan mempunyai atribut yang dapat diubah (diklik, *double* klik) atau diubah tampilannya (warna, ukuran).

3. *Services*

   Services adalah layanan yang dilakukan dari perintah yang diberikan. Dan dijalankan di belakang layar atau tanpa menggunakan *user interface* pengguna. Misalnya seperti MP3 player, yang menjalankan musik selagi pengguna melakukan berbagai *activity*.

4. *Content Provider*

   Content provider merupakan *interface* terstruktur untuk data.

5. *Intents*

   Intents merupakan pesan yang bersifat asinkon yang akan memerintahkan aplikasi meminta sesuatu dari komponen yang ada pada sistem android. Secara

sederhananya, intents merupakan aksi yang akan dilakukan setelah mendapatkan perintah dari *services* atau *activity*.

6. *Broadcast Receiver*

   Broadcast Receiver digunakan untuk menerima pesan dari sistem atau intents.

7. *Widgets*

   Widgets meruapakan komponen interaktif yang ada pada layar utama android. Beberapa ikon *widget* akan berada interaktif pada layar yang memperbolehkan pengguna melakukan aksi tertentu.

2.3.4 Fundamental Android

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data file resource yang dibutuhkan oleh aplikasi, dimana prosesnya di-package oleh tools yang dinamakan “apt tools” ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang disebut dengan apikasi. Ada empat jenis komponen pada aplikasi Android yaitu [7]:

1. *Activities*

   Suatu aktivity akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interasksi. Satu *activity* biasanya akan dipakai untuk meampilkan aplikasi atau yang bertindak sebagai user interface saat aplikasi diperlihatkan kepada pengguna.

2. *Service*

   Service tidak memiliki *Graphic User interface* (GUI), tetapi *service* berjalan secara bacground.

3. *Broadcast receiver*

   Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Broadcast receiver tidak memiliki *user infterface* (UI), tetapi memiliki sebuah *activity* untuk merespon informasi yang diterima.

4. *Content provider*

   Content provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistemseperti SQLite. *Content provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu activity.

2.3.5 Versi Android

Telepon pertama yang menggunakan sistem operasi android adalah HTC Dream yang dirilis pada 22 oktober 2008. Adapun versi-versi android yang pernah dirilis adalah sebagai berikut :

1. Android versi 1.1
2. Android versi 1.5 (*Cupcake*)
3. Android 1.6 (*Donut*)
4. Android versi 2.0/2.1 (*Eclair*)
5. Android Versi 2.2 (*Froyo*)
6. Android Versi 2.3 (*Gingerbread*)
7. Android Versi 3.0 (*Honeycomb*)
8. Android Versi 4.0 (*Ice Cream Sandwich*)
9. Android Versi 4.1 (*Jelly Bean*)
10. Android Versi 4.4 (*KitKat*)
11. Android Versi 5.0 (*Lollipop*)
12. Android Versi 6.0 (*Marshmallow*)
13. Android Versi 7.0 (*Nougat*)
14. Android Versi 8.0 (*Oreo*).

2.4 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - Integrated Development Environment (IDE) untuk pengembangan aplikasi Android,

berdasarkan IntelliJ IDEA . Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi Android, misalnya [9]:

1. Sistem versi berbasis Gradle yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat pengujian dan kerangka kerja yang ekstensif
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk Google Cloud Platform, mempermudah pengintegrasian Google Cloud Messaging dan App Engine.

2.4.1 Struktur Proyek

Setiap proyek di Android Studio berisi satu atau lebih modul dengan file kode sumber dan file sumber daya. Berbagai jenis modul meliputi [9] :

1. Android app modules
2. Library modules
3. App Engine modules

Secara default, Android Studio menampilkan file proyek Anda dalam tampilan proyek Android. Pandangan ini diselenggarakan oleh modul untuk menyediakan akses cepat ke file kunci sumber proyek Anda. Semua file membangun terlihat di

tingkat atas di bawah Script Gradle dan setiap modul aplikasi mengandung tiga unsur berikut [9] :

1. Manifests : file Manifest.
2. Java : Sumber file kode.
3. Res : file Sumber Daya.

2.5 *User Generated Content* (UGC)

User Generated Content (UGC) yang juga dikenal sebagai *consumer-generated media* (CGM), mengacu kepada materi atau konten apa pun yang dibuat dan diunggah ke internet untuk umum oleh non-media profesionals. Contohnya adalah video-video yang diunggah ke youtube dan profil-profil seseorang di facebook.

User Generated Content saat ini merupakan salah satu bentuk konten yang paling cepat berkembang di internet. Pada dasarnya bekerja untuk bagaimana mengubah user dapat berinteraksi dengan internet, dan bagaimana suatu pengiklanan dapat sampai kepada penerima iklan. Dengan semakin tersediannya akses internet berkecepatan tinggi dan adanya teknologi pencarian, *user generated cotent* menjadi salah satu bentuk dominan dari media global [10].

2.6 Google Maps

Google Maps adalah layanan aplikasi peta online yang disediakan oleh Google secara gratis. Layanan peta Google Maps ini dapat diakses melalui situs resminya yaitu http://maps.Google.com. Layanan yang dibuat secara interaktif oleh Google Maps menawarkan peta dan gambar satelit untuk seluruh dunia dimana di dalamnya peta dapat digeser sesuai keinginan penggunam, mengubah level *zoom*, serta mengubah tampilan jenis peta. Fasilitas lain yang disediakan oleh Google Maps antara lain adalah pencarian lokasi dengan memasukkan kata kunci seperti nama, tempat, kota atau jalan, dan fasilitas lain seperti perhitungan rute perjalanan dari satu tempat ke tempat lain.

Google Maps dibuat dengan menggunakan kombinasi dari gambar peta, database, serta obyek-obyek interaktif yang dibuat dengan bahasa pemrograman HTML, Javascript, dan AJAX, serta beberapa bahasa pemrograman lainnya. Gambaran yang muncul pada peta merupakan hasil komunikasi dengan database pada web server Google untuk menampilkan gabungan dari potongan-potongan gambar yang diminta. Dimana, keseluruhan citra yang ada di integrasikan ke dalam database pada Google server yang dapat dipanggil sesuai kebutuhan [11].

2.7 *Global Positioning System* (GPS)

Global Positioning system (GPS) adalah sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Sistem dalam GPS menggunakan 24 satelit yang akan mengirimkan sinyal gelombang mikro ke bumi dimana sinyal ini akan diterima oleh alat penerima (*receiver)* di permukaan. GPS *receiver* ini akan mengumpulkan informasi dari satelit GPS, seperti [12]:

a. Waktu, GPS *receiver* menerima informasi waktu dari jam atom yang mempunyai keakuratan sangat tinggi.
b. Lokasi, GPS memberikan informasi lokasi dalam tiga dimensi yaitu latitude, longitude, dan elevasi.
c. Kecepatan, GPS dapat membeikan informasi kecepatan ketika berpindah tempat.
d. Arah Perjalanan, GPS dapat menunjukan arah tujuan.
e. Simpan lokasi, GPS *receiver* dapat menyimpan tempat-tempat yang sudah pernah atau ingin di kunjungi.
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.

2.8 *Location Based Services* (LBS)

Location Based Services (LBS) adalah service yang berfunsi untuk mencari dengan teknologi GPS atau *global positioning service* dan Google's *cell-based location.* Map dan layanan berbasis lokasi menggunakan lintang dan bujur untuk menentukan lokasi geografis. Namun, sebagai pengguna yang dibutuhkan adalah alamat dan posisi *realtime* pengguna bukan nilai lintang dan bujur. Pada andoid, telah disediakan *geocoder* yang mendukung *forward* dan *reverse geocoding.* Dengan menggunakan *geocoder,* nilai lintang dan bujur dapat di konversi menjadi alamat dunia nyata atau sebaliknya.

Location Based Services atau layanan berbasis lokasi adalah istilah umum yang dignkan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang digunakan. Terdapat dua unsur utama LBS yaitu [12]:

a. *Location manager* (API Maps)

   Berfungsi untuk menyediakan tools atau *resource* untuk LBS. *Application Programming Interface* (API) Maps menyediakan fasilitas untuk menampilkan, memanipulasi maps/peta beserta *feature-feature* lainnya seperti tampilan satelit, jalan, maupun gabungannya. Dimana paket ini tersedia pada com.google.android.maps.

b. *Location provider* (API Location)

   Berfungsi untuk menyediakan teknologi pencarian lokasi yang digunakan oleh *device* atau perangkat. API *location* berhubungan dengan data GPS dan data lokasi *realtime.* API *location* berada pada paket android yaitu dalam paket android.location. dengan *location manager,* pengguna dapat menentukan lokasi pengguna saat ini, *track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan.

2.9 Java

Sejarah pemrograman java dimulai ketika pada tahun 1990, Jamis Gosling, pemrogram yang bekerja pada sun Microsystems mendapat tugas menciptakan program untuk mengendalikan perangkat elektronika rumah tangga. Java adalah penyederhanaan dari C++. Sun sendiri mendeskripsikan java sebagai bahasa pemrograman yang sederhana, berorientasi objek, bertipe statik, tekompilasi, *multi-threaded, gerbage collected*, netral arsitektur, robust, aman, dan dapat diperluas [13].

1. Sederhana

   Java tidak mendukung pointer matematis, casting tipe implisit, struktur atau unions, operator overloading, template, header filer, maupun pewarisan majemuk.

2. Berorientasi objek

   Seperti halnya C++, java menggunakan kelas-kelas untuk mengelola kode ke dalam modul-modul logika. Pada saat runtime, program menciptakan objek dari kelas-keals yang ada.

3. Bertipe statik

   Semua objek yang digunkan dalam peogram harus dideklarasikan sebelum digunkan.

4. Tekompilasi

   Sebelum dapat menjalankan program yang ditulis dalam bahasa pemrograman java, program harus dikompilasi oleh kompilator java. Hasil kompilasinya berupa file kode byte, yang mirip dengan file kode mesin yang dapat dieksekusi di bawah sistem operasi apa pun asal memiliki interpreter java.

5. *Multi-threaded*

   Program java dapat berisi eksekusi *multi threaded* yang memungkinkan program mengulas citra pada layar pada satu threas, sementara tetap menerima masukan dari keyboard pada thread utamnya.

6. *Gerbage collected (*Pembuang sampah*)*

   Program java nenyingkirkan sendiri “sampah-sampah” yang tak berguna, artinya program tidak perlu menghapus objek-objek yang dialokasikannya di memori.

7. Bebas arsitektur

   Pada dasarnya, java tidak dirancang untuk prosesor ataupun sistem operasi tertentu. Java dirancang untuk bekerja pada berbagai arsitektur prosesor dan berbagai sistem operasi.

8. Tangguh (*robust)*

   Karena interpreter java memeriksa seluruh akses sistem yang diakukan program, maka program java tidak akan membuat sistem menjadi crash. Bila terjadi masalah serius, program java membuat perkecualian (*exception). Exception* ini dapat ditangani dan dikelola oleh program tanpa beresiko memacetkan sistem.

9. Aman

   Sistem java tidak hanya memverifikasi seluruh akses ke memori tetapi juga menjamin bahwa tidak ada virus yang membonceng pada applet yang sedang berjalan.

10. Dapat diperluas

    Program java mendukung metode native yakni fungsi-fungsi yangditulis dalam bahasa lain, biasanya C++. Dukungan terhadap metode native ini memungkinkan pemrogram menulis fungsi yang bisa dieksekusi secara lebih cepat daripada fungsi yang sama yang ditulis dalam java.

2.10 *Processor Hypertext Protocol* (PHP)

PHP *Hypertext Preprocessor* atau disingkat dengan PHP adalah suatu bahasa *scripting* khususnya digunakan untuk *web Development.* Karena sifatnya yang *server side scripting,* maka untuk menjalankan PHP harus menggunakan web server. PHP

dapat diintegrasikan dengan HTML, JavaScript, Jquery, Ajax. Namun pada umumnya PHP lebih banyak digunakan bersamaan dengan file bertipe HTML.

Pada awalnya PHP merupakan kependekan dari *Personal Home Page*. Pertama kali dibuat oleh Rasmus lerdorf pada tahun 1995. Pada waktu itu PHP masih bernama *Form interpreted (FI)*, yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari web.

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang *Interpreter* PHP menjadi lebih bersih, lebih baik, dan lebih cepat. Pada tahun 1998 perusahaan tersebut merilis *Intepreter* baru untuk PHP dan meresmikan rilis tersebut dan singkatan PHP diubah menjadi akronim berulang PHP : *Hypertext processing.*

PHP sendiri dapat melakukan tugas-tugas yang dilakukan dengan mekanisme CGI seperti mengambil, mengumpulkan data dari database, meng-generate halaman dinamis, atau bahkan menerima atau mengirim cookie. Yang menjadi keutamaan PHP itu sendiri adalah PHP bisa digunakan di berbagai *operating system.*

Berikut adalah contoh kode program PHP [21] :

```
<html>
<title> contoh program PHP </title>
<head> </head>
<body>
<h4> Contoh 1 Program PHP </h4>
<?php
 Echo “hello, ini program PHP pertamaku”;
?>
</body>
</html>
```

2.11 MySQL

MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi *user* serta menggunakan perintah standar SQL (*structured Query Language*). Selain sebagai database server, MySQL juga merupakan program yang dapat mengakses suatu database MySQL yang

berposisi sebagai server. Jadi MySQL adalah sebuah database yang dapat digunakan sebagai Client maupun Server.

Database MySQL merupakan suatu perangkat lunak database yang berbentuk database relasional atau dalam bahasa basisdata sering disebut dengan *Relation Database Management System* (RDBMS) yang menggunakan suatu bahasa permintaan bernama SQL. Istilah MySQL dan SQL adalah kedua istilah yang berbeda artinya. MySQL adalah sebuah program database, sedangkan SQL adalah bahasa perintah (*Query*) dalam program MySQL [22].

2.12 *JavaScript Object Notation* (JSON)

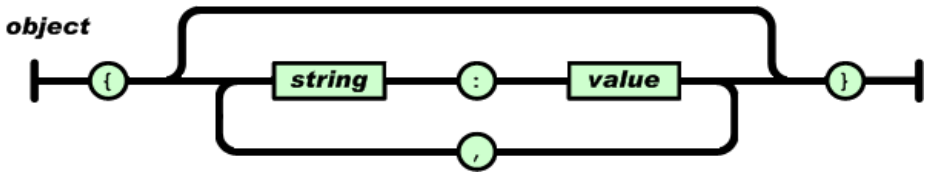
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. JSON terbuat dari dua struktur [14]:

1. 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*.
2. 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 pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data

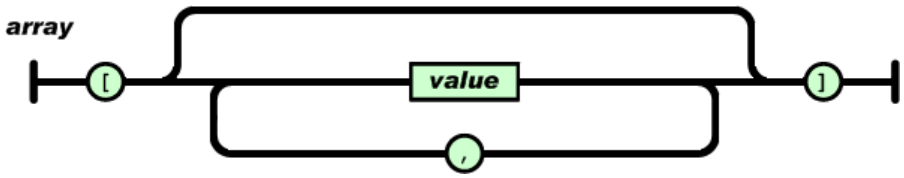
mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut [14]:

1. 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).



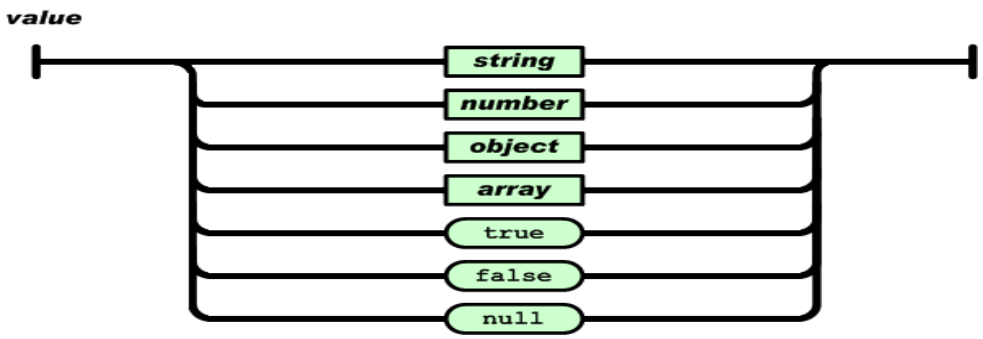
Gambar 2.4 Objek JSON [14]

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



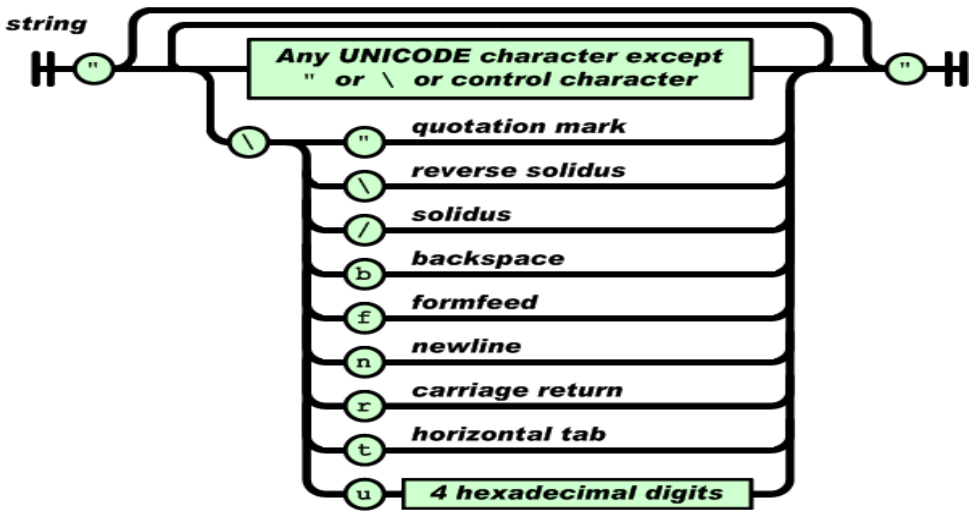
Gambar 2.5 Larik JSON [14]

3. 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.



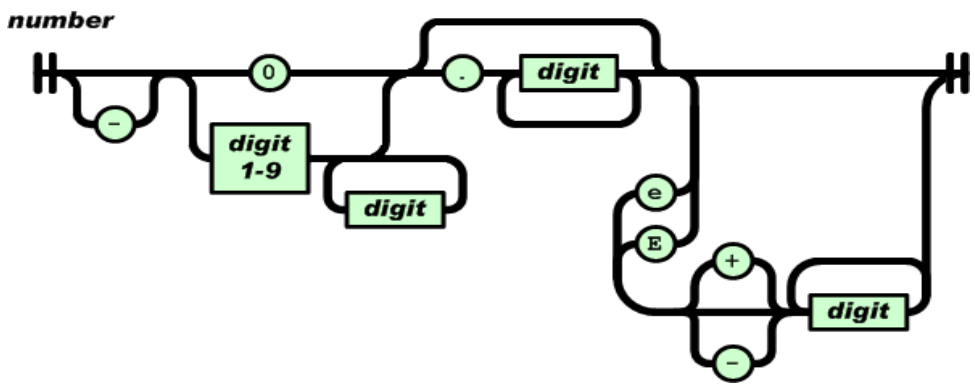
Gambar 2.6 Value JSON [14]

4. 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[11] sangat mirip dengan string C atau Java.



Gambar 2.7 String JSON [14]

5. Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2.8 Number JSON [14]

Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detail encoding yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan [14].

2.13 *Object Oriented Programming* (OOP)

Booch menyatakan "pemrograman berorientasi objek adalah metode implementasi dimana program diorganisasikan sebagai kumpulan objek yang bekerja sama. Masing-masing objek merepresentasikan instan dari kelas, dan kelas-keals itu anggota suatu hirarki kelas-kelas yang disatukan lewat keterhubungan pewarisan".

Stroustrup mendeskripsikan pemrograman berorientasi objek sebagai penggabungan abstraksi data dengan pewarisan. Fitur-fitur penting untuk mendukung abstraksi data adalah *constructor* serta *destructor* untuk penciptaan dan pemusnahan objek. Tipe berparameter, mekanisme *exception, object coercions* dan *iterator* [15].

2.13.1 Keunggulan Pendekatan berorientasi objek

Keunggulan pendekatan berorientasi objek adalah sebagai berikut [15]:

1. Bekerja yang mendekati kognisi manusia.
2. Menghasilkan sistem yang dibangun di atas bentuk-bentuk antara yang stabil. Dan dengan demikian lebih mampu untuk mengikuti perubahan.

3. Dapat digunakan tidka hanya pada perancangan perangkat lunak tapi juga seluruh proses pengembangan perangkat lunak.
4. Mereduksi resiko pengembangan sistem-sistem kompleks terutama karena pemaduan terjadi menyebar pada siklus kehidupan tidak terjadi dalam sekejab sebagaimana kejadian ledakan besar (*big bang).*
5. Pendekatan ini mampu mngeksploitasi keampuhan bahasa pemrograman berbasis objek.

2.13.2 Kriteria – Kriteria Pendekatan Objek

Berikut ini adalah beberapa kriteria-kriteria penting pendekatan objek yang harus dipahami tentang metodologi berorientasi objek [15]:

1. *Seamlessness*

   Konsep berorientasi objek dapat diterapkan di seluruh siklus hidup pengembang sistem (*System Developmnent life cycle*) dari analisis, perancangan, sampai implementasi.

2. Kelas

   Pendekatan berorientasi objek berdasar pada pentingnya kelas. Kelas adalah elemen perangkat lunak yang mendeskripsikan tipe data abstrak (ADT – *Abstract data type*) dimana implementasinya dapat secara parsial atau total. Implementasi parsial bila di kelas terdapat operas/metode abstrak.

3. *Assertion*

   Fitur tipe data abstrak harus secara formal menspesifikasikan properti kelas Assertion berperan pada prakondisi (*precondition*), poskondisi (*postcondition*), invarian kelas. Assertion mendeskripsikan efek fitur-fitur di objek independen terhadap implementasi fitur-fitur itu.

4. Kelas Merupakan Cara Pemodulan

   Pendekatan berorientasi objek merupakan teknik pembentukan arsitektur perangkat lunak. Dampak utama penerapannya adalah pada struktur modul dari

sistem perangkat lunak. Peran utamanya di sistem berorientasi objek dimainkan oleh kelas. Pada pendekatan murni berorientasi objek maka kelas merupakan satu-satunya pemodulan.

5. Kelas merupakan *type*

   Kelas cukup ampuh untuk menghilangkan kebutuhan mekanisme typing yang lain. Setiap *type* harus didasarkan pada kelas.

6. *Exception handling*

   Sistem perangkat lunak seharusnya mampu menangani kondisi abnormal secara terstruktur dengan baik.

7. *Static typing*

   Sistem *typing* yang terdefinisi baik memaksakan sejumlah deklarasi tipe dan aturan-aturan kompatibilitas tipe serta menjamin keselamatan tipe saat berjalan.

8. *Genericity*

   Agar penerapan typing menjadi praktis adalah kemampuan untuk mendefinisikan type-parameterized classes atau disebut generik.

9. Pewarisan

   Pendekatan berorientasi objek harus memungkinkan mendefinisikan kelas yang merupakan turunan kelas yang lain. Pewarisan dapat berupa pewarisan tunggal (*single inheritance*) dan pewarisan jamak (*multiple inheritance*).

10. *Redefinition*

    Mengubah implementasi atau properti lain dari fitur-fitur yang diwariskan. Redefinisi dapat mempengaruhi fitur, penanda (tipe argumen dan hasil) dan spesifikasi.

11. *Polymorphism*

    Dimungkinkan untuk entitas ditautkan ke objek-objek beragam tipe. Polymorphism dikendalikan oleh pewarisan.

12. *Dynamic binding*

    Dynamic binding mempunyai pengaruh utama pada aplikasi berorientasi objek. Mekanisme ini memungkinkan pengembang menulis panggilan secara sederhana untuk beberapa panggilan yang dapat dimungkinkan, bergantung pada situasi saat sebenarnnya panggilan itu dieksekusi.

13. *Run-time type interogation*

    Perangkat lunak memerlukan mekanisme untuk mengakses objek dengan cara yang aman tanpa melanggar konstrain static typing. Mekanisme ini harus dirancang agar tidak emniadakan manfaat-manfaat dari dynamic binding dan polymorphism.

14. *Defender feature and classes*

    Dalam banyak kondisi diperlukan deklarasi kelas sebagai *deferred class deffered feature* dan *deferred classes* masih memiliki *assertion* yang mendeskripsika properti-properti abstrak tapi implementasi ditunda di kelas-kelas kongkrek turunannya. Kelas *non-deffered* disebut kelas efektif, sementara kelas *deffered* sering disebut kelas abstrak.

15. *Memory management* dan *gerbage collection*

    Sistem berorientasi objek cenderung untuk menciptakan banyak objek dengan kebergantungan yang sering kompleks. Kebijkan mengharuskan pengembang mengelola memori akan mengurangi efisiensi pengembang. Di lingkungan bahasa pemrograman berorientasi objek yang bagus adalah pengelola memori dilakukan secara otomatis dibawah kendali *gerbage collector*.

2.14 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh model-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO). UML merupakan standar yang relatif terbuka yang dikontrol

oleh Objeck Management Group (OMG), sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan awal 1990-an [16].

2.14.1 Diagram – Diagram UML

UML memiliki 13 jenis diagram resmi yaitu diagram *Activity*, *Class*, *Comunnication, Component, Composite structure*, *Deployment, Interaction overview, Object, Package, Sequence, State mechine, Timing,* dan diagram *Use case*. Meskipun jenis jenis diagram ini merupakan cara orang-orang memperlakukan UML. Para perancang UML tidak memandang diagram sebagai bagian yang sentral. Dan hasilnya, jenis-jenis diagram bukanlah hal yang mutlak. secara legal dapat menggunakan elemen-elemen satu jenis diagram untuk diagram yang lain [16].

Pada penelitian ini diagram yang digunakan adalah diagram *Use case, Class, activity,* dan *Sequence.*

1. Diagram *Use Case*

   Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. Sebuah *use case* adalah serangkaian skenarion yang dikemas menjadi satu oleh tujuan pengguna umum [16].

   Dalam bahasa *use case*, para pengguna disebut sebagai aktor. Aktor merupakan sebuah peran yang dimainkan oleh seorang pengguna dalam kaitannya dengan sistem. aktor tidak harus manusia. Jika sebuah sistem melakukan layanan untuk sebuah sistem komputer lain, sistem lain tersebut merupakan aktor [16].

   Cockburn menjelaskan sebuah skema tingkatan *use case*. Terdapat tingkat *sea level, fish level dan kite level.*

a. *Sea level*

Inti *use case* berada pada tingkat *sea level. Use case sea level* khususnya mewakili sebuah interaksi diskrit antara aktor utama dan sistem. *use case* semacam ini akan memberikan suatu nilai pada aktor utama dan biasanya memberi waktu beberapa menit sampai setengah jam bagi aktor utama untuk menyelesaikan.

b. *Fish level*

Use case yang ada pada tinkat *fish level* adalah *use case* yang ada hanya karena mereka dimasukkan oleh *use case sea level.*

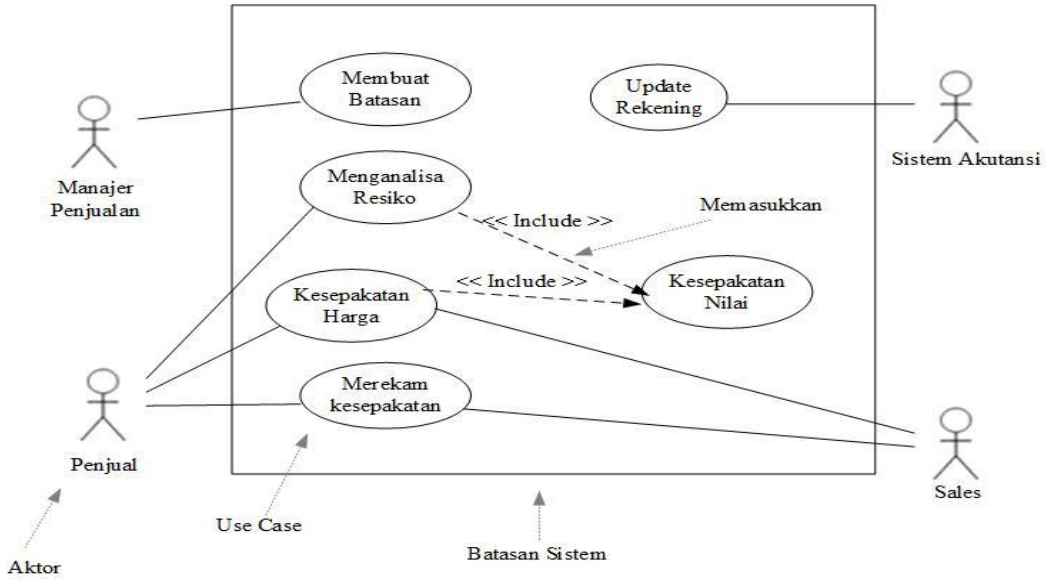
c. *Kite level*

Use case kite level menampilkan bagaimana *use case sea level* sesuai dengan interaksi bisnis yang lebih luas. *Use case kite level* biasanya merupakan *use case* bisnis.

Use case diagram menampilkan aktor, *use case,* dan hubungan antara mereka :

a. Aktor mana yang menggunakan *use case* mana.
b. *Use case* mana yang memasukkan *use case* lain.

Berikut ini adalah contoh dari diagram *use case* :

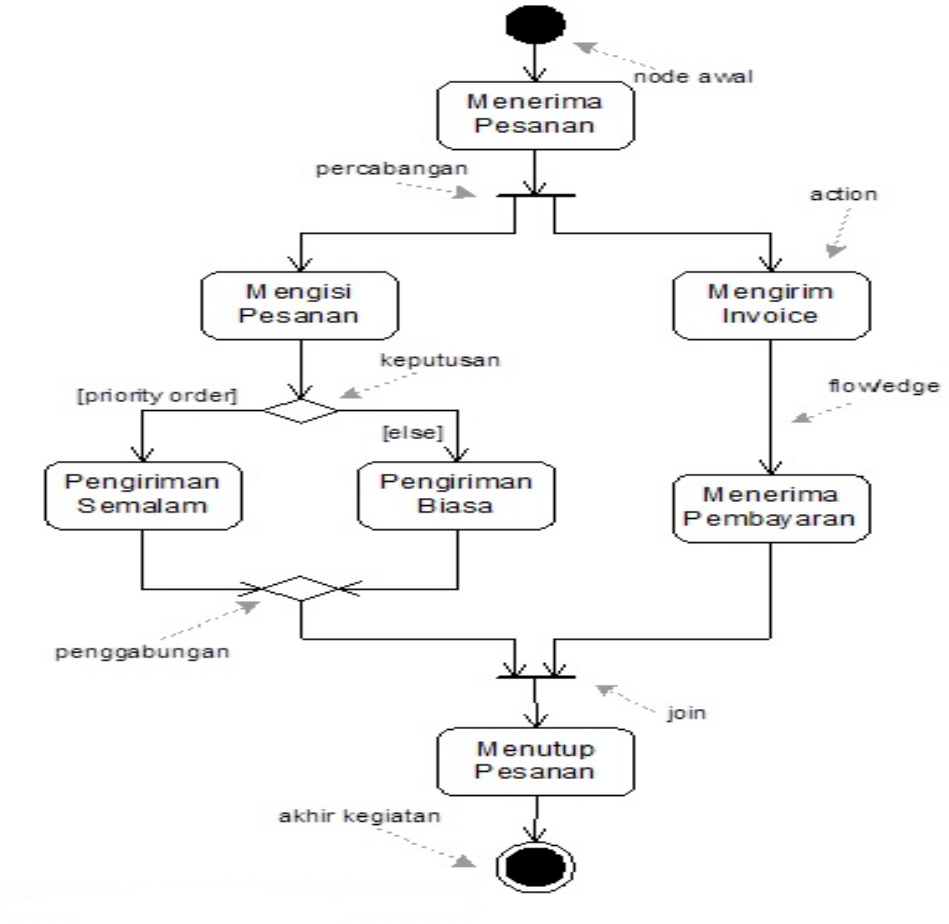


Gambar 2.9 Contoh Diagram *Use Case* [16]

2. Diagram *Activity*

Diagram *activity* adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram *activity* memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram *activity* dan diagram alir adalah diagram alir mendukung *behavior* pararel.

Diagram *activity* dapat dipisahkan ke dalam partisi-partisi yang menampilkan *action* mana yang dilakukan oleh sebuah *class* atau organisasi. Model ini biasa disebut dengan *swim lanes* [16].

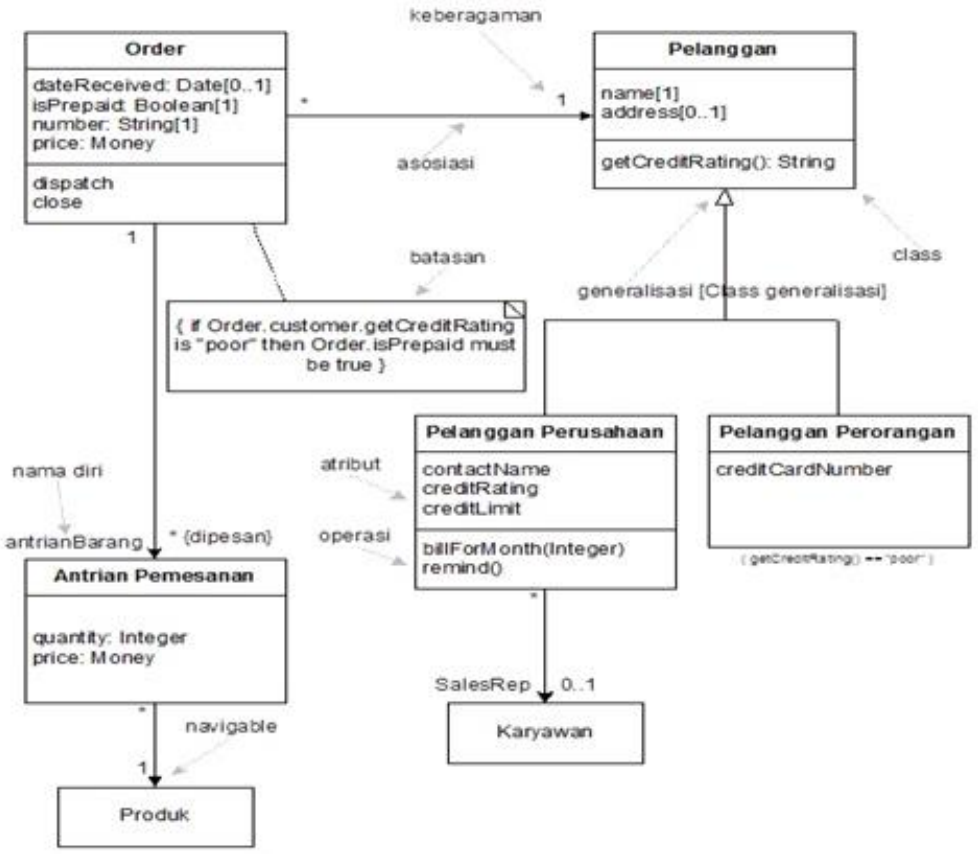


Gambar 2.10 Contoh Diagram *Activity* [16]

3. Diagram *Class*

Diagram *class* mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. Diagram *class* juga menunjukan properti dan operasi sebuah *class* dan batasan-batasan yang terdapat pada sebuah hubungan-hubungan objek. UML menggunakan istilah fitur sebagai istilah umum yang meliputi properti dan operasi sebuah *class* [16].

Didalam class dibagi menjadi tiga bagian. Nama *class*, atributnya, dan operasinya.

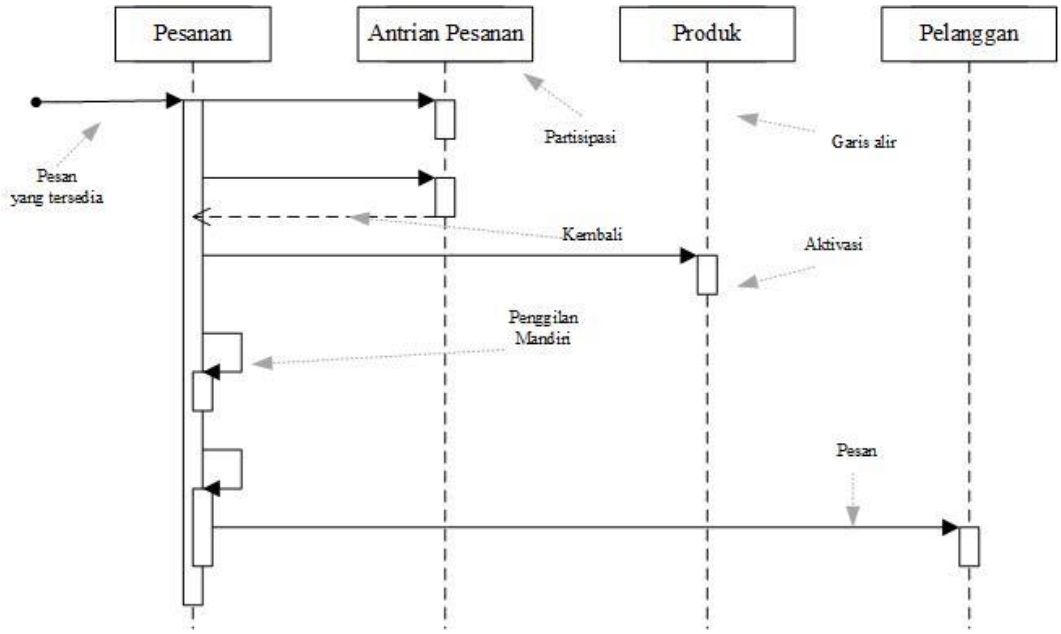


Gambar 2.11 Contoh Diagram *Class* [16]

4. Diagram *Sequence*

Diagram *sequence* secara khusus menjabarkan *behavior* sebuah skenario tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan

yang melewati objek-objekdi dalam *use case.* Diagram *sequence* menunjukkan interaksi dengan menampilkan setiap partisipan dengan garis alir secara vertikal dan pengurutan pesan dari atas ke bawah [16].



Gambar 2.12 Contoh Diagram *Sequence* [16]

2.15 Metode *Agglomerative Hierarchical Clustering*

Penglompokan hierarki aglomeratif merupakan metode pengelompokan pendekatan bawah – atas. Pengelompokan dimulai dari masing-masing data sebagai kelompok lalu secara rekursi mencari kelompok terdekat untuk menjadi satu kelompok yang lebih besar. Algoritma pengelompokan hierarki aglomeratif dijabarkan dengan langkah sebagai berikut [23] :

1. Hitung matriks kedekatan berdasarkan jenis jarak yang digunakan

   d(u,v) = $\sqrt{\sum_{i=0}^{n}(|Ui - Vi|)^2}$

2. Ulangi langkah 3 sampai 4, hingga hanya satu kelompok yang tersisa

3. Gabungkan dua kelompok terdekat berdasarkan parameter kedekatan yang ditentukan
4. Perbaharui matriks kedekatan untuk merefleksikan kedekatan di antara kelompok baru dan kelompok asli yang dusah digabung.

Contoh implementasi pengelompokan hierarki aglomeratif sebagai berikut [23] :

Tabel 2.1 Set Data Dua Dimensi

Data	X	Y
1	1	1
2	4	1
3	1	2
4	3	4
5	5	4

Langkah pertama adalah menghitung jarak pada semua pasangan dua data :

d(1,2) = $\sqrt{|4-1|^2+|1-1|^2}$ = 3

d(1,3) = $\sqrt{|1-1|^2+|2-1|^2}$ = 1

dan seterusnya sehingga dihasilkan matriks jarak sebagai berikut :

Tabel 2.2 Matriks jarak

Data	1	2	3	4	5
1	0	3	1	5	7
2	3	0	4	4	4
3	1	4	0	4	6
4	5	4	4	0	2
5	7	4	6	2	0

Lalu memilih jarak dua kelompok yang terkecil : min(d_{uv}) = d_{13} = 1

Maka dipilih kelompok 1 dan 3, sehingga kelompok 1 dan 3 digabungkan. Selanjutnya hitung kembali jarak-jarak antara kelompok (1,3) dengan kelompok yang tersisa yaitu 2,4 dan 5.

$d_{(13)2} = \min\{d_{12}, d_{32}\} = \min\{3,4\} = 3$

$d_{(13)2} = \min\{d_{12}, d_{32}\} = \min\{3,4\} = 3$

$d_{(13)2} = \min\{d_{12}, d_{32}\} = \min\{3,4\} = 3$

setelah itu, menghapus baris dan kolom yang bersesuaian dengan kelompok 1 dan 3, dan menambah baris dan kolom untuk kelompok (13). Sehingga didapat matriks jarak baru sebagai berikut :

Tabel 2.3 Matriks jarak Baru

Data	13	2	4	5
13	0	4	5	7
2	4	0	4	4
4	5	4	0	2
5	7	4	2	0

Selanjutnya pilih kembali jarak dua kelompok terkecil dan lakukan langkah yang sama dengan sebelumnya hingga hanya tersisa satu kelompok.

2.16 Metode *Technique For Others Reference by Similarity to Ideal Solution* (TOPSIS)

Technique For Others Reference by Similarity to Ideal Solution atau TOPSIS merupakan salah satu sistem pendukung keputusan multikriteria. Metode TOPSIS memiliki keuntungan yaitu merupakan salah satu metode yang *simple* dan konsep rasional yang mudah dipahami, dan mampu mengukur kinerja relatif dalam membentuk form matematika sederhana. Metode TOPSIS memiliki langkah-langkah sebagai berikut [17]:

1. Membuat matriks keputusan yang ternormalisasi.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$$

Keterangan :

i = 1,2,...,m

j = 1,2,...,n

2. Membuat matriks keputusan yang ternormalisasi terbobot.

$$y_{ij} = w_i r_{ij}$$

Keterangan :

i = 1,2,...,m

j = 1,2,...,n

3. Menentukan matriks solusi ideal posifit dan matriks solusi ideal negatif.

$$A^+ = (y_1^+, y_2^+, \dots, y_n^+);$$
$$A^- = (y_1^-, y_2^-, \dots, y_n^-);$$

$$y_j^+ = \begin{cases} max_i y_{ij} \\ max_i y_{ij} \end{cases} \qquad y_j^- = \begin{cases} min_i y_{ij} \\ min_i y_{ij} \end{cases}$$

4. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan negatif.

   a. Jarak antara alternatid Ai dengan solusi ideal positif dengan rumus sebagai berikut :

   $$D_i^+ = \sqrt{\sum_{j=1}^{n} (y_i^+ - yij)^2}; \quad \text{i = 1,2,...,m}$$

   b. Jarak antara alternatid Ai dengan solusi ideal negatif dengan rumus sebagai berikut :

   $$D_i^- = \sqrt{\sum_{j=1}^{n} (yij - y_i^-)^2}; \quad \text{i = 1,2,...,m}$$

5. Menentukan nilai preferensi untuk setiap alternatif.

$$V_i = \frac{D_i^-}{D_i^- + D_i^+} \qquad \text{i = 1,2,...,m}$$

Nilai Vi yang lebih besar menunjukan bahwa alternatif Ai lebih dipilih.

2.17 Metode *Weighted Product*

Metode *weighted product* merupakan salah satu metode yang digunakan untuk menyelesaikan masalah *Multi Attribute Decision Making* (MADM). Metode WP menggunakan perkalian untuk menghubungkan nilai atribut atau kriteria, dimana nilai setiap atribut atau kriteria harus dipangkatkan terlebih dahulu dengan bobot atribut yang bersangkutan. Preferensi untuk alternatif Ai diberikan dengan rumus sebagai berikut [17]:

$$s_i = \prod_{j=1}^{n} x_{ij}^{w_j}$$ Dengan i = 1, 2, ..., m dan j = 1, 2, ..., n

Keterangan :

Π = product

S_i = skor / nilai dari setiap alternatif

X_{ij} = nilai alternatif ke – i terhadap atribut ke-j

w_j = bobot dari setiap atribut

untuk perangkingan atau mencari alternatif terbaik dilakukan dengan rumus berikut :

$$v_i = \frac{\prod_{j=1}^{n} X_{ij}^{w_j}}{\prod_{j=1}^{n} \left(X_{ij}^{*}\right)^{w_j}}$$

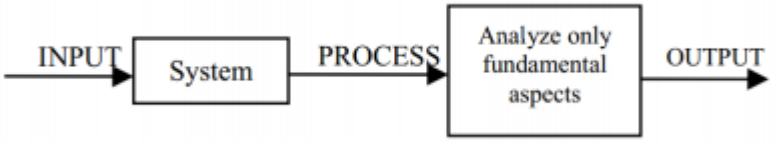
Nilai Vi terbesar menyatakan bahwa alternatif Ai terpilih.

2.18 Pengujian *Black-Box*

pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinsikan. Metode *Black Box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program.

Pengujian *Black Box* dapat menemukan kesalahan dalam kategori berikut [18]:

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam strutur data atau akses basisdata eksternal
4. Inisialisasi dan kesalahan terminasi
5. validitas fungsional
6. kesensitifan sistem terhadap nilai input tertentu
7. batasan dari suatu data



Gambar 2.13 Sitem kerja dari Teknik Pengujian *Black Box*[18]

2.19 Pengujian Beta

Pengujian beta dianggap sebagai bentuk pengujian penerimaan perangkat lunak oleh pengguna eksternal. Dibagikan kepada pengguna yang berada di luar tim pemrogram sebagai penguji beta. Perangkat lunak dibagikan ke sekelompok orang sehingga pengujian bisa memastikan bahwa produknya memiliki kesalahan atau bug yang kecil [19].

2.20 Skala *Likert*

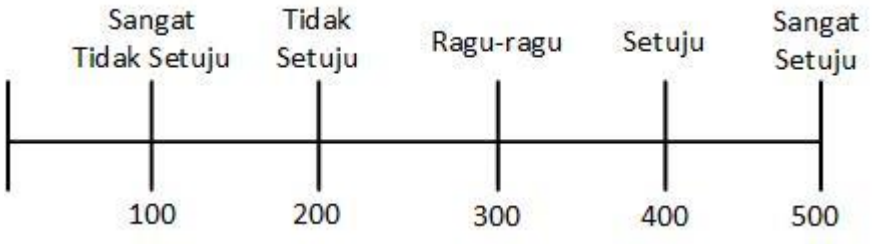
Skala *likert* adalah metode yang digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau kelompok orang tentang fenomena sosial. Dengan skala *likert*, variabel yang akan diukur dijabarkan menjadi indikator variabel. Selanjutnya infikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrumen pernyataan atau pertanyaan. Instrumen penelitian dapat dibuat dalam bentuk checklist ataupun pilihan ganda.

Jawaban setiap item instrumen yang menggunakan skala *likert* mempunyai gradasi dari sangat positif sampai sangat negatif dimana jawaban yang diberikan akan diberikan skor. Jawaban dapat berupa kata-kata sebagai berikut [20]:

a. Sangat setuju dengan skor 5
b. Setuju dengan skor 4
c. Ragu-ragu dengan skor 3
d. Tidak setuju dengan skor 2
e. Sangat tidak setuju dengan skor 1

Langkah-langkah skala likert adalah :

1. Menghitung jumlah skor ideal (kriterium) yaitu dengan rumus :
   Kriterium = nilai bobot maksimal X jumlah responden
2. Menghitung jumlah jawaban dari respoden dalam bentuk persentase, digunakan rumus :
   $P = \frac{total\ nilai}{kriterium}\ x\ 100\%$
   Keterangan :
   P : nilai persentasi yang dicari
   Total nilai : jumlah frekuensi X nilai bobot
3. Selanjutnya skor dimasukkan kedalam bentuk *Interval Rating Scale*



Gambar 2. 14 *Interval Rating Scale*