BAB 2
LANDASAN TEORI

2.1. Travelling

Traveling dalam bahasa Indonesia adalah aktivitas melancong; berpindah dalam satu tempat ketempat lainnya dengan berbagai alasan, seperti bisnis, liburan, dan sebagainya. Tetapi menurut Herajeng Gustiayu dalam bukunya yang berjudul The Backpacker's Notes, traveling adalah lebih kepada bagaimana beradaptasi dengan lingkungan baru. Didalam buku tersebut juga dibahas 3 hal dasar yang penting dari membuat perencanaan traveling dan menentukan biaya yang tepat diantaranya adalah transportasi, akomodasi dan makanan [4].

2.2. Sistem Informasi

Sistem adalah kumpulan elemen-elemen yang saling bekerja sama dan berinteraksi untuk memproses masukan kemudian saling berhubungan, berkumpul bersama-sama untuk mencapai suatu sasaran tertentu. Sedangkan informasi adalah data yang telah diproses kemudian diolah menjadi bentuk yang lebih berguna, digunakan untuk mengambil keputusan. Jadi dapat disimpulkan bahwa sistem informasi adalah suatu alat yang membantu dalam menyediakan informasi bagi penerimanya dan untuk membantu dalam pengambilan keputusan [5].

2.2.1. Komponen Sistem Informasi

Sistem informasi dapat digambarkan sebagai sistem yang terdiri dari berbagai komponen. Komponen ini saling berinteraksi satu dengan yang lain membentuk suatu kesatuan untuk mencapai sasaran [5]:

1. Komponen Masukan

   Mewakili sejumlah data yang masuk ke dalam sistem informasi. Input tersebut termasuk metode-metode dan media untuk memperoleh data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

2. Komponen Proses

   Terdiri dari kombinasi prosedur, logika, dan model matematika yang akan memanipulasi data input dan data yang tersimpan di database dengan cara tertentu untuk menghasilkan keluaran (output) yang diinginkan.

3. Komponen Keluaran

   Produk dari sistem informasi adalah keluaran yang merupakan informasi dan dokumentasi yang dapat digunakan untuk semua tingkatan manajemen dan semua pemakai sistem.

4. Komponen Teknologi

   Teknologi merupakan *tool-box* dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan.

5. Komponen *Hardware*

   Hardware berperan penting sebagai suatu peralatan vital bagi sistem informasi yang digunakan untuk menyimpan dan mengolah informasi .

6. Komponen *Software*

   Software berfungsi sebagai kumpulan perintah komputerisasi yang berguna untuk mengolah, menghitung dan memanipulasi data yang diambil dari *hardware* untuk menciptakan suatu informasi.

7. Komponen Basis data

   Database merupakan kumpulan dari data yang saling berhubungan satu sama lainnya, tersimpan pada perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Pengelolaan database umumnya dikenal dengan nama DBMS (*Database Management System*).

8. Komponen Kendali

   Bagian pengendalian dirancang dan diterapkan untuk memelihara sistem dari hal-hal yang dapat merusaknya, seperti faktor-faktor alamiah temperatur, air, api, debu, virus, sabotase/*hijacking*, dan sebagainya.

2.3. Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat seluler layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel Android pertama mulai dijual pada bulan Oktober 2008 [6].

2.3.1. Android Life Cycle

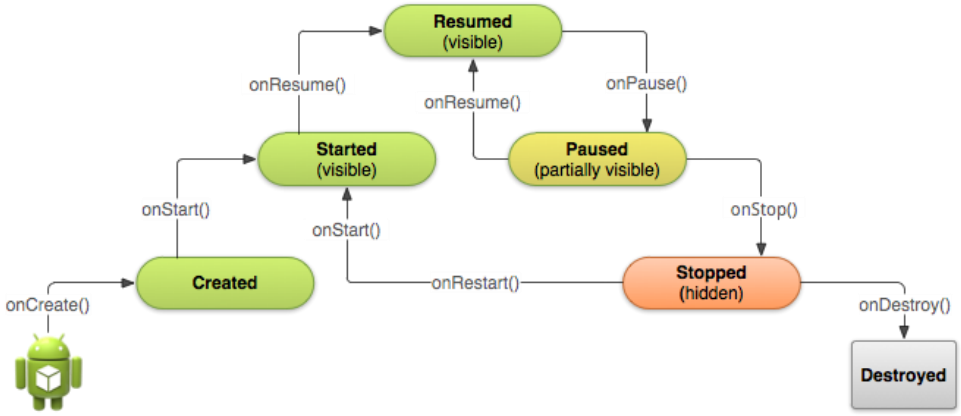
Aplikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada *Tabel 2.1 Komponen Aplikasi Android*, klasifikasi tersebut berdasarkan kelas-kelas dasar java yang digunakan [7].

Tabel 2.1 Komponen Aplikasi Android

Functionality	Java Base Class	Examples
Focused thing a user can do	Activity	Edit a note, play a game
Background process	Service	Play music, update weather icon
Receive messages	BroadcastReceiver	Trigger alarm upon event
Store and retrieve data	ContentProvider	Open a phone contact

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan *specified permission* sebelum digunakan seperti komponen Service, *Broadcast Receiver* dan *Content Provider* [7].

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem android menjalankan kode dalam method Activity dengan menerapkan metode callback tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi desktop atau web, Hal ini dikarenakan aplikasi mobile memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika handling panggilan masuk aplikasi diharuskan menghentikan proses sementara, Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna [6].



Gambar 2.1 Siklus Hidup Android

Terdapat beberapa state dalam siklus hidup android yang terjadi seperti diilustrasikan pada *Gambar 2.1 Siklus Hidup Android*, akan tetapi hanya beberapa dari state tersebut yang menjadi statis diantaranya:

1. Resumed

Resumed terjadi ketika aplikasi berjalan setelah state *paused* . State ini akan menjalankan perintah program yang ditulis pada method onResume() [6].

2. Paused

Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini [6].

3. Stopped

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service di background [6].

State lain seperti *Created* dan *Started* bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem OnCreate() dipanggil, dengan cepat sistem akan memanggil method OnStart(), kemudian diikuti oleh onResume() [7].

2.3.2. Fitur

Android memiliki beberapa fitur utama yang sering digunakan dalam proses pembangunan aplikasi diantaranya adalah:

1. Multi-proses dan App Widgets

Sistem operasi android tidak melarang prosesor menjalankan lebih dari satu aplikasi dalam satu waktu. Sistem operasi android dapat mengatur aplikasi dan thread yang berjalan secara *multitasking.* Keuntungan yang didapat adalah ketika aplikasi berjalan dan berinteraksi dengan pengguna di layer depan sistem operasi, proses dari aplikasi lain dapat berjalan untuk melakukan pembaruan informasi. Sebagai contoh misalnya ketika pengguna memainkan game, proses lain dapat berjalan di belakang aplikasi seperti memeriksa harga saham dan memunculkan peringatan [6].

App Widgets adalah mini aplikasi yang dapat embedded dalam aplikasi seperti home screen. App widgets dapat menjalankan proses request seperti musik streaming atau mendeteksi suhu ruangan secara background [6].

Multi-proses dapat memberikan manfaat berupa user experience yang lebih banyak, namun penggunaan fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [7].

2. Touch Gestures dan Multi-touch

Touchscreen adalah *user interface* intuitif yang digunakan banyak *smartphone* di dunia. Dengan fitur ini interaksi dapat dibuat lebih mudah karena cukup dengan menggunakan jari tangan. *Multi-touch* adalah kemampuan yang dapat melakukan tracking lebih dari satu tangan dalam satu waktu, Fitur ini sering digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut [7].

3. Hard dan Soft Keyboard

Salah satu fitur pada perangkat smartphone adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoperasian android.. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol tersebut dalam aplikasi. Tombol non fisik adalah tombol yang dibuat oleh sistem operasi seperti keyboard virtual, dan tombol navigasi aplikasi [7].

2.3.3. Prinsip Desain

Android memiliki beberapa prinsip desain yang dapat menjadi acuan dalam membuat desain aplikasi android diantaranya adalah :

1. Multiple Assets

Android mendukung jutaan smartphone, tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu *Multiple Assets* sangat disarankan digunakan untuk mengatasi fragmentasi pada android. Seperti ilustrasi pada *Gambar 2.2 Klasifikasi Ukuran Ikon,* android menciptakan beberapa klasifikasi ukuran icon yaitu MDPI, HDPI, XHDPI, XXHDPI dan XXXHDPI. MDPI dan HDPI dikhususkan untuk icon yang akan digunakan pada device berukuran smartphone sedangkan untuk XHDPI, XXHDPI dan XXXHDPI digunakan pada device berukuran tablet [8].



Gambar 2.2 Klasifikasi Ukuran Ikon

2. Touch Feedback

Touch Feedback dalam android digunakan sebagai respon setiap objek yang ditekan pengguna. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [7].



Gambar 2.3 *Touch Feedback*

3. Pattern Gesture

Touch gesture berguna untuk meningkatkan experience pengguna dalam menggunakan aplikasi. Terdapat beberapa gesture yang didukung oleh android diataranya adalah [7]:

a. Touch

Touch adalah gesture default sebagai event untuk item tertentu yang paling sering digunakan dalam perangkat android, hampir seluruh kontrol utama aplikasi ini menggunakan touch.

b. Long Press

Biasanya digunakan untuk seleksi data , dengan gesture ini dimungkinkan untuk memilih satu atau lebih item dalam sebuah tampilan dan menjalankan suatu fungsionalitas tertentu [6].

c. Swipe Or Drag

Swipe adalah menyentuh sebuah titik pada layar dan menggerakkan jari yang tetap tersentuh pada layar ke titik lain pada layar. Swipe biasanya dilakukan untuk kebutuhan berpindah layar atau tergantung dari aplikasi. Swipe dapat dilakukan dari dan ke arah mana saja [6].

d. Double Touch

Pada Smartphone dan tablet android, melakukan dua kali tapping secara berturut-turut pada satu obyek, fungsinya berbeda dengan double klik mouse komputer. Pada android, teknik ini biasanya dipakai untuk melakukan zoom in dan zoom out atau memperbesar / memperkecil sebuah obyek gambar [6].

e. Pinch Open

Teknik lain yang biasa digunakan adalah dengan menggunakan dua jari, di mana kedua jari tersebut menyentuh dua titik pada layar yang terpisah di mana ujung dari dua jari tersebut tidak bersentuhan, kemudian kedua jari tersebut sambil tetap menyentuh layar bergerak saling mendekati. Teknik ini digunakan untuk membuka aplikasi tertentu [6].

f. Pinch Close

Teknik Pinch Close adalah kebalikan dari Pinch Open di mana spread dilakukan dengan berawal dari dua jari yang bersentuhan pada ujungnya dan ditempelkan sebuah titik yang sama pada layar, kemudian kedua jari tersebut bergerak memisahkan atau menjauhi satu sama lain Gerakan ini untuk menutup layar [6].

2.4. Android Studio

Android Studio merupakan IDE yang digunakan para developer Android di seluruh dunia untuk mengembangkan aplikasi Android dengan mengusung kualitas dan performa aplikasi. Android Studio adalah IDE Android resmi yang dibangun

oleh Google [9]. Android Studio memiliki beberapa fitur unggulan yang dapat memudahkan developer Android, diantaranya :

1. ***Instant Run***, untuk setiap developer yang menyukai proses build yang cepat, developer Android bisa melakukan build ulang untuk setiap perubahan yang telah dibuat dengan cepat, layaknya rendering html pada web browser. Dengan akselerasi swapping VM ke swapping aplikasi, fitur Instan Run dapat mengoptimalkan waktu pengembangan aplikasi Android.
2. ***Android Emulator***, emulator Android dapat berjalan 3x lebih cepat dibanding emulator Android sebelumnya. Peningkatan ini tidak terlepas dari peningkatan ADB yang sekarang dapat mentransfer aplikasi dan datanya 10x lebih cepat ke emulator dibanding kompilasi langsung ke device (Smartphone). Layaknya Smartphone, emulator Android juga tersedia Google Play Service sehingga developer Android dapat menguji fungsionalitas API secara langsung.
3. ***Cloud Test Lab Integration,*** tulis sekali dan jalankan dimana saja. Selain emulator Android, developer Android bisa meningkatkan pengembangan aplikasi dimana saja dengan menggunakan Cloud Test Lab yang terdiri dari berbagai device Android.
4. ***App Indexing Code Generation & Test,*** keberadaan aplikasi yang telah developer Android kembangkan sangatlah penting, dengan fitur ini, developer Android dapat mempromosikan aplikasinya dengan cara menambahkan URL yang dibangkitkan secara otomatis melalui fitur App Indexing. Dengan menambahkan URL yang indexable, developer bisa menguji keberadaan aplikasinya secara langsung melalui Android Studio minimum versi 2.0.
5. ***GPU Debuger Preview,*** untuk para developer Android khususnya yang bergelut di bidang pengembangan game. Android Studio menyediakan fitur debugger GPU, sehingga developer Android dapat melihat setiap frame dan state GL. Jika ada permasalahan rendering GL, developer Android bisa menangkap dan menganalisa stream GPU yang ada pada device Android.

2.5. Android SDK (Software Development Kit)

Sesuai dengan namanya merupakan kumpulan tools-tools yang di perlukan untuk mengembangkan aplikasi Android. Seperti pada java yang dapat dijalankan pada multiplatform, SDK Android pun tersedia untuk beberapa platform seperti Windows, Mac OS dan Linux [7].

Adapun aplikasi SDK yang didalamnya terdapat beberapa folder sebagai berikut :

1. **Tools**

Berisikan tools-tools yang diperlukan dalam membangun aplikasi Android seperti emulator, Android Virtual Device (AVD), SDK Manager, ddms dan banyak lagi untuk memudahkan membangun aplikasi Android. Semua tools ini dapat di update mengikuti setiap perkembangan platfom Android.

2. **Platforms**

Direktori ini ditujukkan untuk meletakkan versi platform-platform Android yang telah kita download dari SDK Manager. Seperti Android 1.5, 1.6, 2.0 dan seterusnya.

3. **Add-ons**

Direktori ini nantinya berisikan tools tambahan untuk Android SDK, yang fungsinya untuk menambahkan library yang akan digunakan untuk pengembangan aplikasi. Seperti jika ingin mengembangkan aplikasi menggunakan map dari google maka harus memerlukan platform Android yang sudah terdapat library map. Semua ini dapat di download melalui SDK Manager.

Selain folder-folder diatas, akan terdapat file tambahan jika kita mendownload beberapa platfom dan tool dalam folder Android SDK tersebut. Seperti *Docs, platform tools, dan samples*.

2.6. Android Development Tools

Merupakan plugin Eclipse yang disediakan oleh google untuk membantu dan memudahkan Developer mengembangkan aplikasi Android. Dengan adanya ADT membuat project Android, mendesain tampilan aplikasi, menambahkan komponen-komponen yang berasal dari Android Framework, debug aplikasi menggunakan tools-tools pada SDK Android hingga membuat file installer menjadi lebih mudah berkat plugin ini. Untuk proses penginstallan ADT dapat dilakukan dengan 2 cara Pertama secara Offline dengan mendownload file ADT secara manual pada situs Android dan kedua Secara online dengan menggunakan eclipse maupun Android studio [7].

2.7. Web Services

W3C mendefinisikan web service sebagai sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki interface yang didefinisikan, dideskripsikan, dan dimengerti oleh XML atau JSON dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan message berbasis XML atau JSON melalui protokol internet Web service adalah sebuah software aplikasi yang tidak terpengaruh oleh platform, menyediakan method-method yang dapat diakses oleh network. Web Service juga akan menggunakan XML untuk pertukaran data, khususnya pada dua entities bisnis yang berbeda. Beberapa karakteristik dari web service adalah:

1. *Message-based*
2. *Standards-based*
3. *Programming language independent*
4. *Platform-neutral*

Beberapa key standard di dalam web service adalah: JSON, XML, SOAP, WSDL and UDDI [10].

2.8. JSON

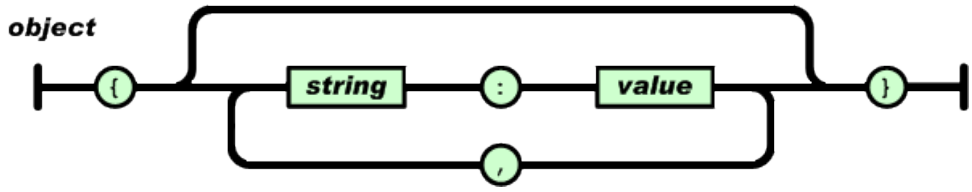
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman 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:

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* [11].
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*) [11].

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

1. Objek

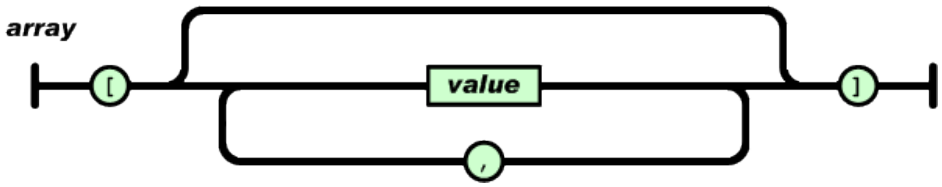
Objek adalah sepasang nama / nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON [11].



Gambar 2.4 Objek JSON

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma). Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat [11].



Gambar 2.5 Array JSON

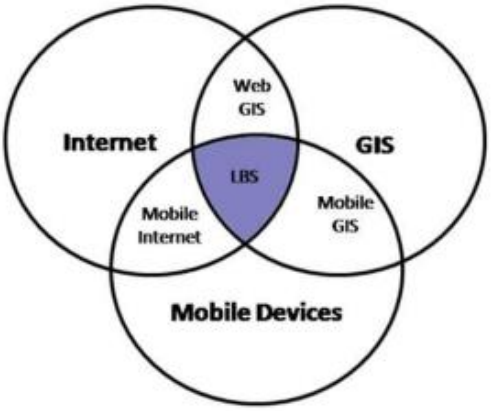
Bentuk data JSON objek dan larik dapat saling dikombinasikan untuk mendukung struktur data yang lebih kompleks. JSON mendukung beberapa tipe data untuk menjadi value seperti Angka, String, Boolean dan nilai NULL [11].

2.9. GPS

Global Positioning System (GPS) merupakan sistem navigasi yang menggunakan satelit dalam penggunaannya. GPS terdiri dari tiga bagian yaitu sistem kontrol, satelit dan pengguna. Sistem kontrol adalah bagian yang mengontrol pergerakan satelit – satelit yang ada dan saling berinteraksi satu sama lain kemudian pengguna adalah alat navigasi yang digunakan seperti perangkat mobile yang kini sudah memiliki fitur GPS di dalamnya. GPS biasanya digunakan untuk menunjukan suatu lokasi yang berada di permukaan bumi dengan tingkat akurasi yang cukup baik yaitu kurang dari 10 meter selama tidak ada benda padat yang dapat menghambat sinyal untuk mendapatkan lokasi pengguna [12].

2.10. Location Based Service

Location Based Service (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan mobile network, yang dilengkapi dengan kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS memungkinkan untuk melakukan komunikasi dan interaksi dua arah. LBS ini dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu *Geographic Information System* (GIS), *Internet Service* dan *mobile device*, seperti pada *Gambar 2.6 Teknologi Location Based Service* [12]:



Gambar 2.6 Teknologi Location Based Service

2.7.1. Metode Penentu Lokasi

LBS merupakan layanan yang mampu bereaksi aktif pada suatu perubahan posisi sehingga dapat mendeteksi letak objek dan memberikan layanan sesuai dengan letak objek yang telah diketahui tersebut. Pada teknologi LBS berbasis jaringan seluler, penentuan posisi pada sebuah peralatan komunikasi ditentukan berdasarkan posisi relatif terhadap lokasi BTS (*Base Transceiver Station*). Dalam menentukan posisi dari sebuah perangkat mobile yang sedang aktif, secara umum terdapat tiga tingkat metode yang digunakan saat ini, di antaranya adalah sebagai berikut [12]:

1. Metode *Basic Positioning* berbasis pada *Cell Identification* (Cell ID)

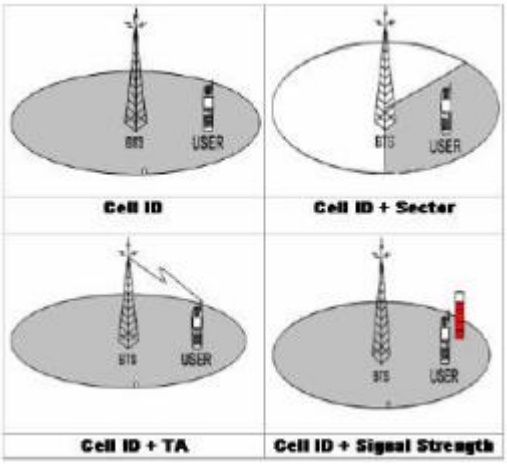
Penentuan posisi didasarkan pada daerah geografis yang tercakup oleh sebuah Cell berhubungan dengan daerah cakupan dari sinyal radio. Ketika sebuah handphone terhubung secara aktif dengan sebuah base station, berarti handphone tersebut diasumsikan berada dalam cell dari base stadion tersebut. Untuk mengukur jarak dan arah handset dari base station tidak dapat diketahi dengan pasti, oleh karena itu untuk lebih meningkatkan lagi akurasi hasil pencarian pada metode ini seringkali dikombinasikan dengan metode lain seperti [12]:

a. *Timing Advance* (TA)

   Dengan menggunakan TA ini metode Cell ID akan ditambahkan sebuah fungsionalitas untuk menghitung *Round Trip Time* (RTT), yaitu waktu transmisi sebuah frame (dari base station ke handphone) dan waktu sebuah *frame* (dari handphone ke station). Dengan tambahan ini jarak antara handphone dan base station dapat ditentukan dengan keakuratan 50 m [12].

b. *Network Measurement Report (NMR)*

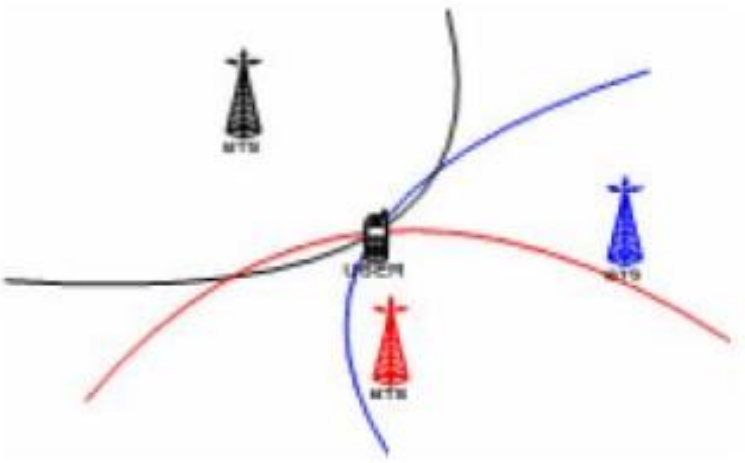
   Berdasarkan pada besar kecilnya sinyal (Receiver Signal Strength) yang diterima handphone yang ada di suatu “sector cell”, maka posisi itu dapat ditentukan dengan lebih akurat [12].



Gambar 2.7 Network Measurement Report

2. Metode *Enhanced Positioning*

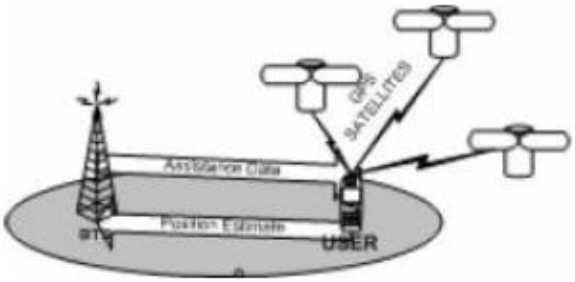
   Metode ini menggunakan pendekatan *Observe Time Difference* (OTD) dengan jaringan GSM yang sering digunakan seperti Enhanced-OTD (EOTD). E-OTD adalah metode pencarian posisi yang berdasarkan pada waktu. Untuk menentukan posisi relative, sebuah handphone harus aktif terhadap tiga base station dan perlu ditentukan terlebih dahulu jarak handphone terhadap masing-masing base station berdasarkan waktu yang ditempuh oleh sebuah sinyal dari handphone ke masing – masing *base station*. Untuk menentukan posisi dari handphone yang sedang aktif digunakan rumus matematika untuk triangulasi. Dengan metode ini akurasinya akan meningkat hingga ketilitian kurang dari 50 m [12].



Gambar 2.8 *Enhaced Positioning*

3. Metode *Advance Positioning*

   Metode ini menggunakan teknologi *Assisted-Global Positioning System* (AGPS). A-GPS ini merupakan metode yang berbasis pada waktu. Pada metode ini, akan dilakukan pengukuran waktu tiba dari sebuah sinyal yang dikirim dari tiga buah satelit GPS. Hal ini berarti handset harus memiliki fasilitas untuk mengakses GPS. A-GPS juga menghasilkan akurasi secara vertical dan estimasi jarak yang baik. Akurasinya pun kurang dari 10 m [12].



Gambar 2.9 *Advance Positioning*

2.7.2. Komponen Location Based Service

Dalam menggunakan *Location Based Service* atau layanan berbasis lokasi terdapat beberapa elemen yang diperlukan, antara lain [12]:

1. *Mobile Devices*

   Sebuah alat yang digunakan untuk meminta informasi yang dibutuhkan seperti perangkat *mobile smartphone* yang memiliki fasilitas navigasi.
2. *Communication Network*

   Jaringan selular yang mengirimkan data pengguna dan permintaan layanan.
3. *Positioning Component*

   Untuk dapat mengolah layanan harus menentukan lokasi pengguna. Posisi pengguna ini dapat diperoleh menggunakan jaringan komunikasi atau dengan menggunakan Global Positioning System (GPS)
4. *Service and Application Provider*

   Penyedia layanan pengguna selular yang bertanggung jawab untuk memproses suatu layanan.
5. *Data and Content Provider*

   Penyedia layanan informasi data yang dapat diminta oleh pengguna.

2.11. Latitude (Garis Lintang)

Latitude adalah garis yang melintang di antara kutub utara dan kutub selatan yang menghubungkan sisi bagian bumi dari timur dan barat. Garis lintang memiliki

posisi membentangi bumi seperti halnya garis khatulistiwa tetapi dengan kondisi nilai tertentu. Latitude dibedakan menjadi dua wilayah yaitu lintang utara dan selatan dimana nilai koordinat di bagian utara selalu positif dan koordinat selatan selalu negatif [12].

2.12. Longitude (Garis Bujur)

Longitude adalah garis bujur yang menghubungkan antara sisi utara dan sisi selatan permukaan bumi. Garis bujur digunakan untuk mengukur sisi barat dan timur koordinat suatu titik di belahan bumi. Longitude dibedakan menjadi dua wilayah yaitu bujur timur dan bujur barat dimana nilai koordinat yang berada di timur selalu negatif dan nilai koordinat dibagian barat selalu positif [12].

2.13. Google Maps

Google Maps adalah sebuah layanan gratis dari Google tentang berbagai macam teknologi pemetaan. Google Maps menampilkan peta dunia secara digital sehingga lebih memudahkan ketika ingin mengetahui suatu lokasi tertentu. Google Maps menyediakan beberapa keunggulan seperti peta jalan, menunjukan rute untuk suatu perjalanan yang diinginkan baik itu melalui jalur mobil sampai pejalan kaki [12].

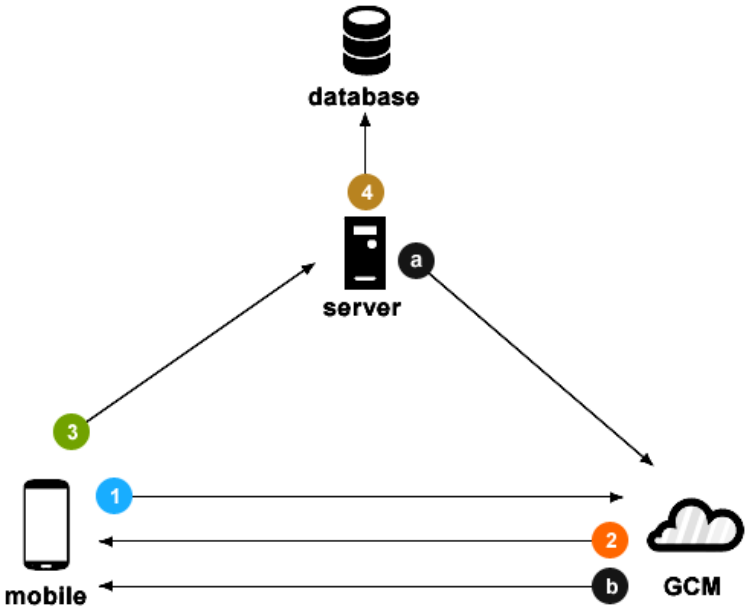
2.14. Google Maps API

Google Maps API adalah sebuah layanan yang memungkinkan Google Maps agar dapat diintegrasikan ke dalam aplikasi yang memang membutuhkan fitur dari Google Maps ini. Google Maps API adalah sebuah library Javascript. Dengan menggunakan Google Maps API ini dapat memudahkan pengembang dalam membuat aplikasi yang membutuhkan teknologi peta di dalamnya dan tidak perlu memikirkan cara membuatnya secara manual karena dengan mengunakan Google Maps API ini sudah semua yang dibutuhkan terkait teknologi peta digital sudah cukup lengkap dan dapat dimanfaatkan untuk aplikasi yang membutuhkannya [12].

Saat ini Google Maps API telah mengalami perkembangan, hingga saat ini Google Maps API telah mencapai versi 3. Namun, Google Maps API versi 3 ini masih hanya bisa digunakan oleh pemrograman berbasis web dengan menggunakan Javascript. Untuk penggunaan Google Maps API pada perangkat mobile seperti misalnya android Google Maps API yang bisa digunakan saat ini telah mencapai versi 2. Perkembangan pada Google Maps Versi 2 ini adalah dengan memiliki fitur–fitur yang lengkap dari versi sebelumnya salah satunya adalah dapat menampilkan peta dengan sudut yang di inginkan dan berupa 3D view. Selain itu pada Google Maps API terdapat beberapa pilihan model peta yang ditampilkan seperti Roadmap, Satellite, Terrain dan Hybrid.

2.15. Google Cloud Messaging

Google Cloud Messaging adalah teknologi untuk membantu pengembang mengirim data dari server menuju device android secara langsung melalui server google, pemanfaatan teknologi ini biasanya digunakan untuk *push messaging*. Pengiriman pesan menggunakan teknologi ini mengharuskan adanya API key, key ini terkait langsung dengan akun google yang digunakan untuk mempublikasikan aplikasi melalui *google play* [7].



Gambar 2.10 GCM Model

Berdasarkan ilustrasi dari Gambar 2.10 GCM Model terdapat dua server yang terlibat dalam proses *push messaging*, pada setiap server yang terlibat terdapat proses pengiriman data yang berbeda-beda diantaranya [7]:

1. Device android mengirimkan *sender id* dan *application id* menuju GCM Server untuk registrasi.
2. Setelah berhasil registrasi, GCM server akan mengirimkan *registration id* menuju device android.
3. Setelah device android mendapatkan registration id, device tersebut akan mengirimkan *registration id* menuju server aplikasi.
4. Server aplikasi akan menyimpan *registration id* yang dikirimkan untuk digunakan sebagai *id* saat menggunakan *push messaging*.

2.16. Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan Bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin [13].

Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "*Tulis sekali, jalankan di mana pun*". Saat ini java Merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web [13].

2.17. PHP

PHP adalah salah satu bahasa *Server-side* yang didesain khusus untuk aplikasi web. PHP dapat disisipkan diantara bahasa HTML dan karena bahasa *Server side*, maka bahasa PHP akan dieksekusi di server, sehingga dikirimkan ke browser adalah "hasil jadi" dalam bentuk HTML, dan kode PHP anda tidak akan terlihat [14].

PHP termasuk dalam *Open Source Product*. Jadi anda dapat merubah *source code* dan mendistribusikan secara bebas. Adapun kelebihan-kelebihan dari PHP yaitu:

1. PHP mudah dibuat dan kecepatan akses tinggi.
2. PHP dapat berjalan dalam web server yang berada dan dalam system operasi yang berbeda pula. PHP dapat berjalan disistem operasi UNIX, Windows98, Windows NT dan Macintosh.
3. PHP diterbitkan secara gratis.
4. PHP juga dapat berjalan pada web server Microsoft Personal Web Server, Apache, IIS, Xitami dan sebagainya.
5. PHP adalah termasuk bahasa yang embeddeb (bisa ditempel atau diletakkan dalam tag HTML).
6. PHP termasuk *server-side* programming.

Sistem database yang didukung oleh PHP seperti Oracle, Sybase, mSQL, MySQL, Solid, Generic ODBC, Postgres SQL [14].

2.18. MySql

Dalam bahasa SQL data akan diatur dalam baris dan kolom pada table, karena pada database di dalamnya pasti akan anda temukan beberapa table. Ada banyak server database yang bisa digunakan, namun salah satu server database yang popular dan banyak digunakan orang adalah MySQL. Selain karena bisa didapat dengan mudah dan gratis, MySQL juga dapat berjalan pada sistem operasi apa pun. Berikut beberapa kelebihan yang dimiliki MySQL [15]:

1. Program database MySQL memang merupakan pasangan setia yang sering diintegrasikan dengan Apache.
2. MySQL memiliki kemampuan server multithread sehingga dapat menangani banyak permintaan (*requet*) layanan secara bersamaan.
3. Dapat berjalan pada banyak sistem operasi (Linux, Windows, SunOS, MacOS).
4. Memiliki metode enkripsi yang baik.
5. Mampu menangani data yang besar dan banyak.
6. Menggunakan autentifikasi pengguna (*user*) dan kata sandi (*password*) sehingga menjadikannya lebih aman.

2.19. BPMN (Business Poress Modeling Notation)

BPMN (Business Porcess modeling notation) adalah Representasi grafis untuk menentukan proses bisnis dalam suatu pemodelan proses bisnis. Tujuan utama dari BPMN adalah menyediakan suatu notasi standar yang mudah dipahami oleh semua pemangku kepentingan bisnis. Dari analisis bisnis yang menciptakan draft permulaan dari proses-proses sampai dengan pengembang-pengembang teknis yang bertanggung jawab untuk mengimplementasikan teknologi yang membantu pelaksanaan proses [16].

Terdapat 4 kategori dari elemen-elemen dalam BPMN, yaitu:

1. *Flow Objects*
   a. *Events*, sebuah event direpresentasikan dengan lingkaran. Events dapat berupa Start, Intermediate, atau End.



Gambar 2.11 *Flow Object (Events)*

   b. *Activities*, sebuah aktivitas direpresentasikan dengan persegi dengan sudut melingkar dan memperlihatkan pekerjaan yang harus dilakukan.



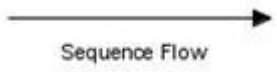
Gambar 2.12 ***Flow Object (Acitvities)***

c. *Gateways*, sebuah gateway direpresentasikan dengan belah ketupat dan memperlihatkan pilihan yang berbeda. Gateway juga menjelaskan mengenai percabangan dan penggabungan dari path yang ada.



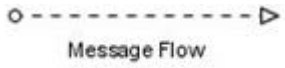
Gambar 2.13 ***Flow Object (Gateways)***

2. *Connecting* Objects
   a. *Sequence Flow*, sequence flow direpresentasikan dengan garis lurus dengan panah tertutup dan menjelaskan mengenai urutan aktivitas yang akan dijalankan.



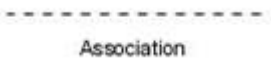
Gambar 2.14 ***Connecting Objects (Sequence Flow)***

   b. *Message Flow*, message flow direpresentasikan dengan garis putus-putus dan panah terbuka. Message flow menjelaskan pertukaran pesan yang sedang terjadi.



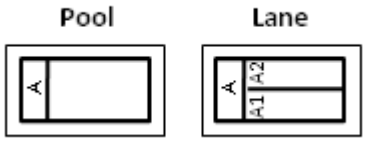
Gambar 2.15 ***Connecting Objects (Message Flow)***

   c. *Association*, association direpresentasikan dengan garis putus-putus. Association digunakan untuk mengasosiasikan sebah artifak, data, maupun flow object.



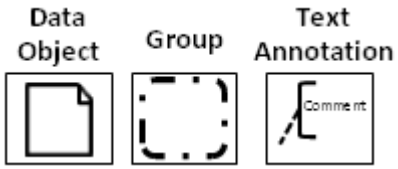
Gambar 2.16 ***Connecting Objects (Association)***

3. *Swimlanes*
   a. *Pool,* pool direpresentasikan dengan persegi besar yang didalamnya dapat berisi flow objects, connecting object, maupun artifak.
   b. *Lane,* lane merupakan bagian lebih mendetail dari pool.



Gambar 2.17 Swimlanes (Pool & Lane)

4. *Artifacts*
   a. *Data Objects*, data object digunakan untuk menjelaskan mengenai data yang dibutuhkan atau dihasilkan dari sebuah aktivitas.
   b. *Group*, group direpresentasikan dalam persegi dengan sudut melingkar dan garis luar putus-putus. Group untuk melakukan grouping aktivitas.
   c. *Annotation*, annotation digunakan untuk menjelaskan model atau diagram.



Gambar 2.18 *Artifacts (Data Object, Group, Text Annotation)*

2.20. Object Oriented Analysis and Design

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan satu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan *user* [17]. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML.

2.13.1. Unified Modelling Languange

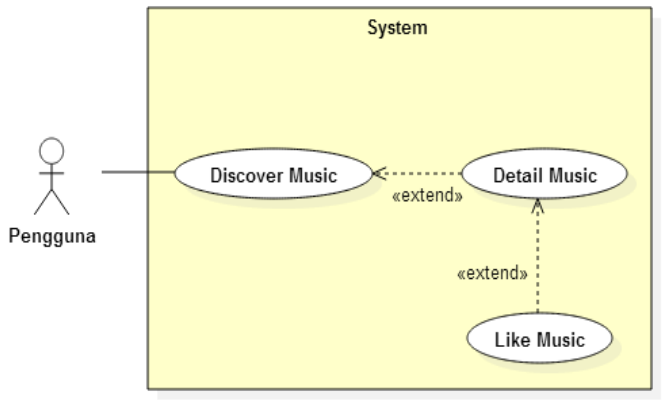
Unified Modelling Language (UML) adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menggunakan *class* dan *operation object* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa

berorientasi objek [18]. Dalam membangun block UML ada 3 hal yang harus diperhatikan, yaitu *object* (memodelkan konsep), *relationship* (mengkoneksikan *object*), dan *diagram* (*grouping* yang saling mengkoneksikan antara *object* dan *relationship.* Diagram yang umum dipakai dalam analisis dan desain adalah:

1. Use Case Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *Use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah use case dapat di- include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang serupa. Sebuah *use case* juga dapat meng-*extend usecase* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

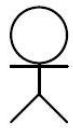
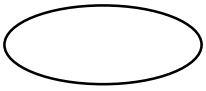
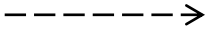
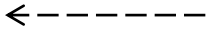
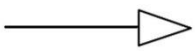
Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, *tester*, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case* [19]. Berikut adalah contoh dari *use case diagram* dapat dilihat pada *Gambar 2.19 Contoh Use Case Diagram* dan simbol yang terdapat pada *use case diagram* dapat dilihat pada *Tabel 2.2 Simbol Usecase Diagram.*



Gambar 2.19 Contoh *Use Case Diagram*

Simbol yang terdapat pada *use case diagram* dapat dilihat pada *Tabel 2.2 Simbol Usecase Diagram.*

Tabel 2.2 Simbol *Usecase Diagram*

Simbol	Nama Simbol	Keterangan
	Actor	Menggambarkan pihak yang berhubungan dengan sistem baik itu merupakan pengguna atau sistem lainnya yang berada dari sistem yang sedang dibahas.
	Use Case	Simbol yang menggambarkan suatu kegiatan (aktifitas) yang terjadi pada sistem
	Asosiasi	Menghubungkan keterkaitan antara aktor dengan use case
	*Include*	Relasi use case dimana proses bersangkutan akan dilanjutkan ke proses yang dituju
	*Extend*	Relasi use case tambahan ke sebuah use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu
	Generalisasi	Sebuah relasi di mana fungsi yang satu adalah fungsi yang umum dari yang lainnya.

2. Use case Scenario

Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per *use case*nya. Berikut adalah dasar format penulisan *use case scenario* [19]. Dasar pembangunan *use case scenario* dapat dilihat pada *Tabel 2.3 Dasar Pembangunan Use Case Scenario.*

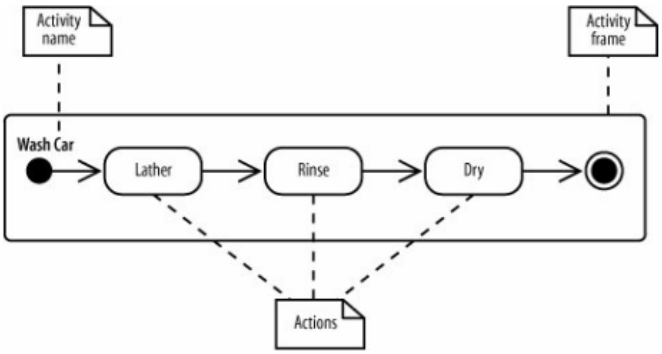
Tabel 2.3 Dasar Pembangunan *Use Case Scenario*

Use Case Name	Berisi nama dari *Use case* yang akan digunakan	
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari *Use case*	
Description	Menjelaskan gambaran dari *Use case*	
Related Use Case	Daftar *Use case* yang berhubungan dengan *Use case* tersebut	
Successful End Condition	Kondisi *Use case* jika berhasil	
Failed End Condition	Kondisi *Use case* jika gagal	
Actors	Daftar aktor yang dapat mengakses *Use case*	
Trigger	Aktifitas yang dilakukan untuk mengawali *Use case*	
Main Flow	Step	Action
	1	Deskripsi urutan aksi dari aktifitas *Use case*
	2	
	3	
Extension	Step	Branching Action
	2.1	Deskripsi urutan aksi lain selain urutan aksi utama
	2.2	

3. Activity Diagram

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas

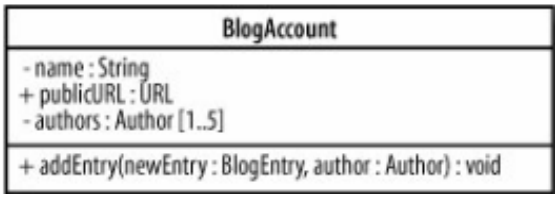
menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis *horizontal* atau *vertikal*. *Activity diagram* dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [18]. Contoh dari *activity diagram* dapat dilihat pada *Gambar 2.20 Contoh Activity Diagram*.



Gambar 2.20 Contoh *Activity Diagram*

4. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) satu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.



Gambar 2.21 Contoh *Class Diagram*

Class memiliki tiga area pokok:

1. Nama dan stereotype
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Class memiliki tipe-tipe *relationship*, diantaranya :

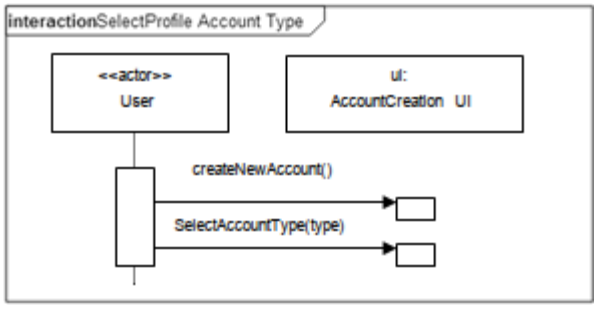
1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain.
2. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu class di share atau direferensikan kepada objek yang ada di class lain.
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Komposisi, yaitu jenis relasi *class diagram* yang kuat dimana jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/*solid*.

5. Depedensi, salah satu jenis relasi *class diagram* yang lemah dimana objek dalam suatu class akan bekerja sangat singkat dengan objek yang ada pada *class* lain.

5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan *Output* tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *Output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message [18]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*. Comtoh dari sequence diagram dapat dilihat pada *Gambar 2.22 Contoh Sequence Diagram*.



Gambar 2.22 Contoh *Sequence Diagram*

2.21. Object Relational Mapping

Object Relational Mapping (ORM) adalah salah satu teknik untuk memetakan basis data relasional ke model objek. Pemetaan dapat dikategorikan menjadi 4, yaitu [20]:

1. *Basic mapping*
2. *Relationship mapping*
3. *Composition mapping*
4. *Inheritance mapping*

2.22. Pengujian Blackbox

Metode pengujian *blackbox* fokus pada keperluan penelusuran kesalahan fungsional dari software. Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan performa
5. Kesalahan inisialisasi dan terminasi

Teknik pengujian black box terdiri dari 10 jenis diantaranya *Equivalence Partitioning, Boundary Value Analysis/Limit Testing, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Requirement Testing, Performance Testing, Endurance Testing, Cause-Effect Relationship Testing.* Salah satunya yang akan dibahas adalah *Equivalence partitioning*.

Equivalence partioning merupakan metode ujicoba blackbox yang membagi domain input dari program menjadi beberapa kelas data dari kasus ujicoba yang dihasilkan. Kasus uji penanganan single yang ideal menemukan sejumlah kesalahan (misalnya : kesalahan pemrosesan dari seluruh data karakter) yang merupakan syarat lain dari suatu kasus yang dieksekusi sebelum kesalahan umum diamati.

2.23. Pengujian Kuesioner

Kuesioner adalah sebuah daftar pernyataan yang harus diisi oleh orang yang akan dievaluasi (responden). Metode yang digunakan dalam kuesioner pada penelitian ini adalah skala Likert. Skala Likert adalah skala pengukuran yang digunakan untuk mengukur sikap seseorang, dengan menempatan kedudukan sikapnya pada kesatuan perasaan kontinum yang berkisar dari "sangat postif" hingga ke "sangat negatif" terhadap sesuatu (objek psikologis) [21]. Dalam skala likert, responden diminta untuk membaca dengan seksama setiap pernyataan yang disajikan, kemudian ia diminta untuk menilai pernyataan-pernyataan tersebut.

Derajat penilaian responden terhadap suatu pernyataan terbagi dalam 5 kategori yang tersusun secara bertingkat, mulai dari Sangat Tidak Setuju (ST), Tidak Setuju (T), Ragu-ragu (N), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada tabel berikut.

Tabel 2.4 Panduan Pemberian Skor

Jenis Pernyataan	Bobot Pendapat				
	SS	S	N	T	TS
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

Dalam menyusun penskalaan dengan metode Likert, dapat ditempuh langkah-langkah sebagai berikut :

1. Tentukan secara tegas sikap terhadap topik apa yang akan diukur.
2. Tentukan secara tegas, dimensi yang menyusun sikap tersebut, kognitif, afektif, atau konatif.
3. Susun pernyataan-pernyataan atau item yang merupakan alat pengukur dimensi yang menyusun sikap yang akan diukur sesuian dengan indikator.
4. Untuk Setiap pilihan respons jawaban diberi skor berdasarkan kriteria sebagai berikut : apabila item postif, maka angka terbesar diletakkan pada respon "sangat setuju", sedangkan bila item negatif, maka angka terbesar diletakkan ad respon "sangat tidak setuju". Skor yang diberikan pada jawaban dijumlahkan.

5. Untuk mengetahui posisi setiap responden tentang suatu variabel, tentukan skor maksimal dan skor minimal yang mungkin dicapai oleh setiap responden.

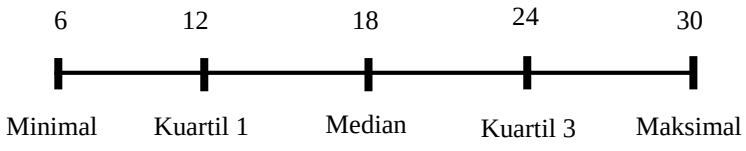
Contoh misalkan kuisoner tersebut diisi oleh 5 responden yang dianggap sebagai sampel penelitian dengan hasil terlihat pada Tabel berikut. Berapa skor terkecil dan terbesar untuk satu orang responden dan total semua responden.

Tabel 2.5 Hasil Skor Item Pertanyaan Dari Setiap Responden

No. Responden	Butir / Item Pertanyaan						Total
	1	2	3	4	5	6	
1	5	3	4	1	2	2	18
2	4	2	4	2	4	1	17
3	5	3	3	1	3	2	17
4	4	3	4	1	2	2	16
5	4	3	3	2	3	1	16
Total							84

Berdasarkan data tersebut, langkah-langkah yang dapat ditempuh untuk mengetahui bagaimana sikap tiap responden terhadap kualitas produk adalah :

1. Menentukan skor maksimal, yaitu skor jawaban terbesar di kali banyak item.
   5 x 6 = 30
2. Menentukan skor minimal, yaitu skor jawaban terkecil di kali banyak item.
   1 x 6 = 6
3. Menentukan nilai median, yaitu hasil penjumlahan skor maksimal dengan skor minimal dibagi dua.
   (30 + 6) : 2 = 18
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan skor minimal dengan median dibagi dua.
   (6 + 18) : 2 = 12
5. Menentukan nilai kuartil 3, yaitu hasil penjumlahan skor maksimal dengan median dibagi dua. (30 + 18) : 2 = 24.
6. Buatlah skala yang menggambarkan skor minimal, nilai kuartil 1, median, kuartil 3 dan skor maksimal.



Keterangan :

- Kategori sikap sangat positif, yaitu daerah yang dibatasi oleh kuartil 3 dan skor maksimal. (Kuartil 3 = x = skor maksimal).
- Kategori sikap positif, yaitu daerah yang dibatasi oleh median dan kuartil 3. (Median = x < Kuartil 3).
- Kategori sikap negatif, yaitu daerah yang dibatasi oleh kuartil 1 dan median. (Kuartil 1 = x < Median).
- Kategori sikap sangat negatif, yaitu daerah yang dibatasi oleh skor minimal dan kuartil 1. (Skor minimal = x < kuartil1).

Catatan :

Range pada skala diatas hanya memunculkan 4 kategori sikap, hal ini dikarenakan skala sikap Likert tidak mengijinkan adanya pernyataan item netral (ragu-ragu). Jadi pernyataan yang ada dalam skala likert hanya dua yaitu, pernyataan item positif dan pernyataan item negatif [21].

7. Carilah batas – batas skor untuk masing-masing kategori sikap. Berdasarkan gambar skala tadi, maka range skor dari keempat kategori adalah :

Tabel 2.6 Range Skor Dari Kategori Sikap

Sikap Sangat Positif	Kuartil 3 ≤ x ≤ Skor Maksimal	24 -30
Sikap Positif	Median ≤ x ≤ Kuartil 3	18 - 24
Sikap Negatif	Kuartil 1 ≤ x ≤ Median	12 - 18
Sikap Sangat Negatif	Skor Minimal ≤ x ≤ Kuartil 1	6 – 12

8. Buatlah tabel distribusi frekuensi sikap tiap responden terhadap kualitas produk.

Tabel 2.7 Tabel Distribusi Frekuensi

Kategori Sikap	Kategori Skor	Frekuensi	Persentase (%)
Sikap Sangat Postif	24 -30	-	-
Sikap Positif	18 - 24	1	20%
Sikap Negatif	12 - 18	4	80%
Sikap Sangat Negatif	6 – 12	-	-
		5	100

9. Berdasarkan tabel di atas terlihat bahwa sikap dari tiap responden tentang kualitas produk, tersebar pada kategori sikap yang sangat positif 80% dan sikap

positif 20%. Persentase tersebut memberikan arti bahwa sebanyak 4 orang (80% dari keseluruhan responden), lalu sebanyak 1 orang (20% dari keseluruhan responden) memandang sikap negatif terhadap kualitas produk yag dihasilkan.