BAB 2
LANDASAN TEORI

2.1 Service (Perbaikan)

Servis sering juga disebut dengan istilah perbaikan(jasa). Pengertian dari perbaikan itu sendiri adalah usaha untuk mengembalikan kondisi dan fungsi dari suatu benda atau alat yang rusak akibat pemakaian alat tersebut pada kondisi semula . Proses perbaikan tidak menuntut penyamaan sesuai kondisi awal, yang diutamakan adalah alat tersebut bisa berfungsi normal kembali. Perbaikan memungkinkan untuk terjadinya pergantian bagian alat/spare part. Terkadang dari beberapa produk yang ada dipasaran tidak menyediakan spare part untuk penggantian saat dilakukan perbaikan, meskipun ada, harga spare part tersebut hampir mendekati harga baru satu unit produk tersebut. Hal ini yang memaksa user/pelanggan untuk membeli baru produk yang sama [4].

Tidak setiap perbaikan dapat diselesaikan dengan mudah, tergantung tingkat kesulitan dan kerumitan assembling atau perakitan alat tersebut, mulai dari tingkatan jenis bahan hingga tingkat kecanggihan fungsi alat tersebut. Tingkat kesulitan tersebutlah yang menumbuhkan perbedaan jenis perbaikan, mulai jenis perbaikan ringan, perbaikan sedang dan perbaikan yang sering dinamakan servis berat. Dari jenis servis diatas ditentukan biaya perbaikan sesuai tingkat kesulitannya.

2.2 Jasa

Setiap manusia memiliki keinginan serta kebutuhan yang berbeda beda dan beragam yang terus meningkat seiring dengan berjalannya waktu, baik jumlah maupun ragam jenisnya Kebutuhan yang tidak terpenuhi membuat orang tahu seseorang menjadi tidak puas, sehinggan ia akan berusaha mencari sesuatu yang memenuhi kebutuhannya. Pemenuhan atas keinginan biasanya dilakukan dengan

berbagai cara seperti dengan memproduksi sendiri, memperoleh dengan mengadakan pertukaran. Telah banyak para ahli yang mengemukakan beberapa definisi-definisi mengenai pengertian jasa. Diantaranya yaitu [4] :

1. Lovelock & Wright 2005:5 (http://digilib.petra.ac.id), mendefinisikan jasa adalah: Kegiatan ekonomi yang menciptakan dan memberikan manfaat bagi pelanggan pada pada waktu dan tempat tertentu, sebagai hasil dari tindakan mewujudkan perubahan yang diinginkan dalam diri atau atas nama penerima jasa tersebut. Definisi tersebut menjelaskan bahwa jasa merupakan kegiatan ekonomi yang menciptakan dan memberikan manfaat bagi pelanggan pada waktu dan tempat tertentu manfaat yang diberikan berupa keuntungan atau laba yang diperoleh pelanggan dari kinerja jasa atau penggunaan barang fisik. Sebagai hasil dari tindakan mewujudkan perubahan yang diinginkan dalam diri atau atas nama penerima.
2. Kotler (Tjiptono 2006:16), mengemukakan pengetian jasa sebagai berikut:
   An service is any act or performance that one party can offer to another that is essentially intangible and does not result in the ownership of anything. Its production may or may not be tied to a physical product .
   Jasa adalah setiap tindakan atau kinerja yang ditawarkan oleh satu pihak ke pihak lain yang secara prinsip tidak berwujud dan tidak menyebabkan perpindahan kepemilikan. Produk jasa dapat terikat atau tidak terikat pada suatu produk fisik.
3. Pengertian jasa secara luas dikemukakan oleh Zeithaml, Bitner & Gremler 2003:3 (www.demandiri.or.id), adalah sebagai berikut:
   Include all economic activities whose output is not a physical product or construction, is generally consumed at the time it is produced, and provides added value in forms (such as convenience, amusement, timelines, comfort or health) that are essentially intangible concerns of its first purchaser.
   Dari definisi diatas dapat diartikan bahwa jasa merupakan keseluruhan aktivitas ekonomi dimana output yang dihasilkan tidak berupa produk yang berbentuk fisik dan pengkonsumsian jasa, umumnya dilakukan pada saat jasa

tersebut diproduksi dan terdapat penambahan nilai pada bentuknya (seperti keyakinan, kesukaan, kenyamanan, kesehatan) merupakan hal yang paling penting yang dipertimbangkan pada saat pembelian pertama. Dari keseluruhan definisi diatas terdapat satu kesamaan pendapat yang menyatakan bahwa pada dasarnya jasa mempunyai ciri-ciri sebagai berikut:

a) Suatu yang tidak berwujud, tetapi dapat memenuhi kebutuhan konsumen.
   Proses produksi jasa dapat menggunakan atau tidak menggunakan bantuan suatu produk fisik.
b) Jasa tidak mengakibatkan peralihan hak atau kepemilikan.
c) Terdapat interaksi antara penyedia jasa dengan pengguna jasa.

2.3 Sistem

Suatu sistem sangatlah dibutuhkan dalam suatu perusahaan atau instansi pemerintahan , karena sistem sangatlah menunjang terhadap kinerja perusahaan atau instansi pemerintah, baik yang berskala kecil maupun besar. Supaya dapat berjalan dengan baik diperlukan kerjasama diantara unsur-unsur yang terkait dalam sistem tersebut. Ada berbagai pendapat yang mendefinisikan pengertian sistem, seperti “Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu[11] Istilah sistem secara umum dapat didefinisikan sebagai kumpulan hal atau elemen yang saling bekerja sama atau yang dihubungkan dengan cara-cara tertentu sehingga membentuk satu kesatuan untuk melaksanakan suatu fungsi guna mencapai suatu tujuan. Sistem mempunyai karakteristik atau sifat – sifat tertentu, yaitu : Komponen Sistem, Batasan Sistem, Lingkungan Luar Sistem, Penghubung Sistem, Masukan Sistem, Keluaran Sistem, Pengolahan Sistem danSasaran Sistem [12].

2.4 Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.Sumber dari informasi adalah data. Data

merupakan bentuk jamak dari bentuk tunggal datum atau data item. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata [11]. Informasi adalah data yang diolah menjadi bentuk yang berguna dan menjadi berarti bagi penerimanya. Kegunaan informasi adalah untuk mengurangi ketidakpastian di dalam proses pengambilan keputusan tentang suatu keadaan. Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya untuk mendapatkan informasi tersebut. Kualitas informasi sangat dipengaruhi atau ditentukan oleh beberapa hal yaitu : Relevan (*Relevancy*), Akurat (*Accurancy*), Tepat waktu (*Time liness*), Ekonomis (*Economy*), Efisien (*Efficiency*), Ketersediaan (*Availability*), Dapat dipercaya (*Reliability*), Konsisten [12]

2.5 Android

Android adalah sistem operasi bersifat *Open Source* berbasis Linux 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]. Android memiliki OS yang sangat baik, cepat dan kuat serta memiliki antarmuka pengguna intuitif yang dikemas dengan pilihan dan fleksibilitas.

Android SDK (*Software Development Kit*) menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada platform android dengan menggunakan bahasa pemrograman Java.

2.5.1 Kelebihan Android

1. User Friendly

Sangat mudah untuk di oprasikan, hanya dengan belajar beberapa hari atau beberapa jam, user sudah dapat menguasai sistem operasi android

2. Notifications

Adanya sistem notifikasi Email, SMS , Voice Dial, Update dan lain sebagainya pada sistem operasi android.

3. Tampilan

Dari segi tampilan, Android tidak kalah bagusnya dari iOs miliknya Apple, karena android hampir mengusung teknologi iOs, hanya saja ini versi murahnya.

4. Open Source

Operating system dibuat open source oleh penciptanya, karena memang berbasis kernel Linux.

5. Aplikasi

Untuk Aplikasi disajikan jutaan pilihan aplikasi yang menarik dari yang gratis hingga berbayar, dan dapat download di Google Play.

2.5.2 Kekurangan Android

1. Update System

untuk melakukan update system tidak mudah, karena harus menunggu dari masing-masing Vendor untuk merilis Update Versi yang terbaru.

2. Batterai Cepat Habis

Dalam menyalakan paket data dan menggunakan widget serta aplikasi yang berjalan secara berlebihan, untuk mengatasinya anda harus mengurangi aktivitas aplikasi pada smartphone .

3. Lemot atau Lag

Sebenarnya kalau ini ada kaitannya dengan spesifikasi dari masing-masing perangkat, namun ada kalanya Android ini tidak bersahabat dengan beberapa aplikasi dikarenakan Ram ataupun Prosesornya yang kurang memadai, jadi sesuaikan aplikasi dengan perangkatnya.

2.5.3 Prinsip Desain Android

Dalam pembangunan aplikasi, android memiliki acuan sebagai prinsip, yaitu :

1. Multiple assets

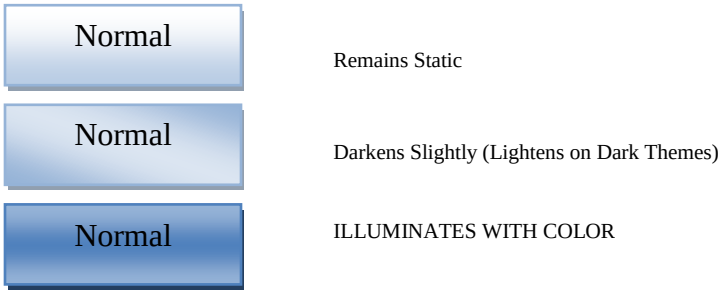
Android mendukung jutaan *smartphone*, tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu *Multiple Assets* sangat disarankan digunakan untuk mengatasi fragmentasi pada android. Android menciptakan beberapa klasifikasi ukuran icon yaitu MDPI, HDPI, XHDPI, XXHDPI dan XXXHDPI. MDPI dan HDPI dikhususkan untuk icon yang akan digunakan pada device berukuran smartphone sedangkan untuk XHDPI, XXHDPI dan XXXHDPI digunakan pada device berukuran tablet[10]. Klasifikasi ukuran ikon dapat dilihat pada gambar 2.1:



Gambar 2.1 Klasifikasi Ukuran Icon

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[10]. *Touch feedback* dapat dilihat pada gambar 2.2 Touch Feedback :



Gambar 2.2 Touch Feedback

3. Pattern Gesture

Touch gesture berguna untuk meningkatkan experience pengguna dalam menggunakan aplikasi. Terdapat beberapa gesture yang didukung oleh android diataranya adalah[10]:

1. 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.

2. 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.

3. 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.

4. 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.

5. 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. 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.

2.6 JSON (Javascript Object Notation)

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, 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 [7].

JSON terbuat dari dua struktur:

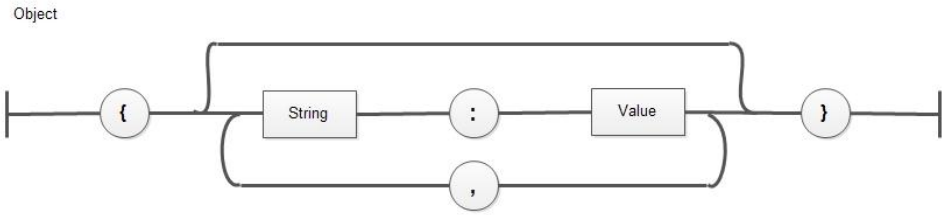
- Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus

(*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

- Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

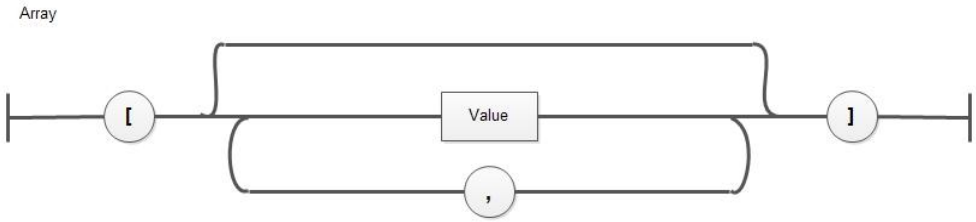
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga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

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).



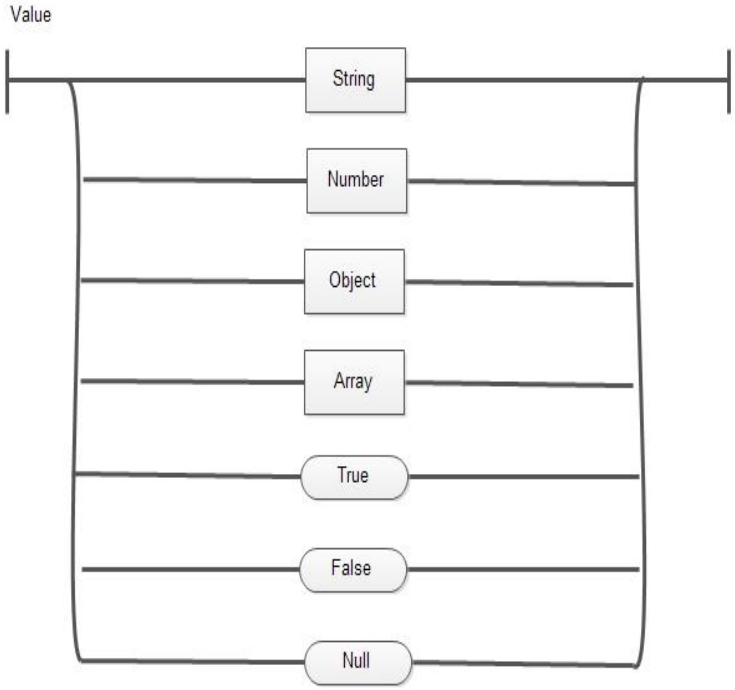
Gambar 2.3 Objek

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



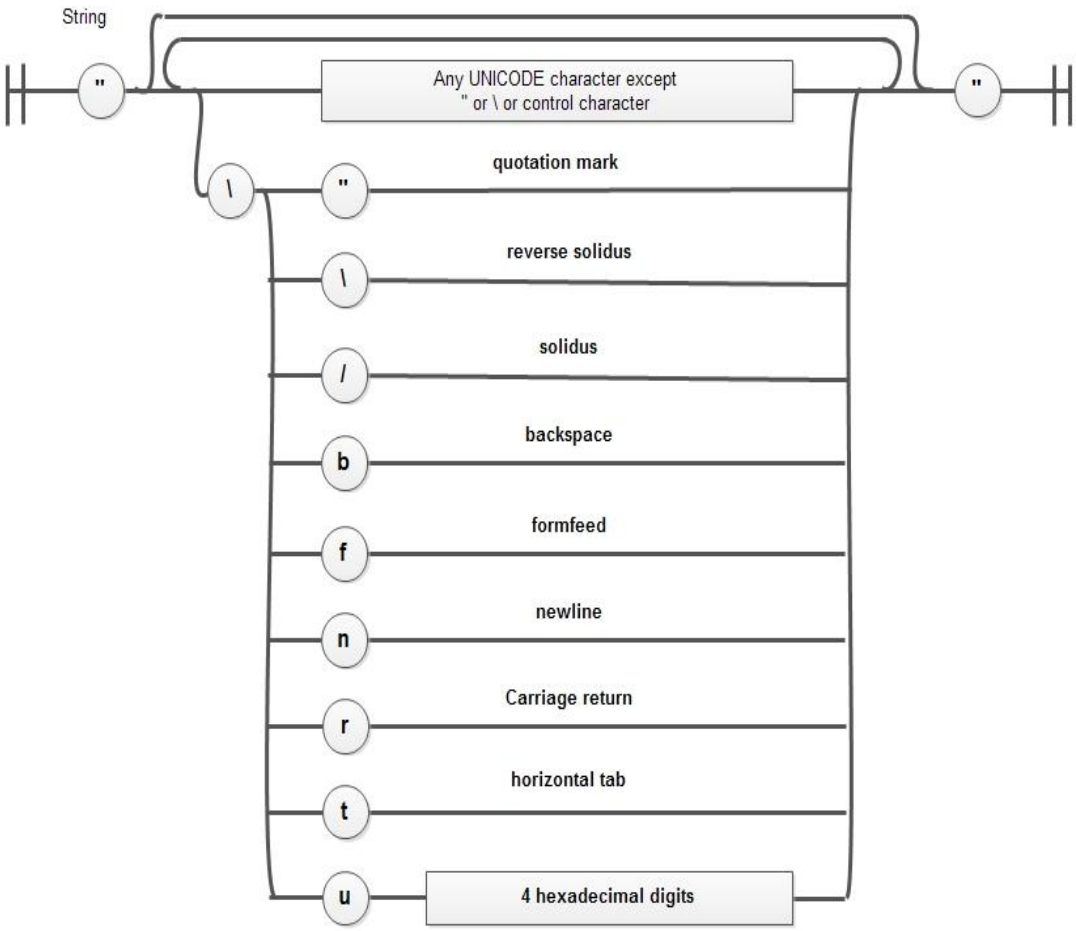
Gambar 2.4 Larik

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.



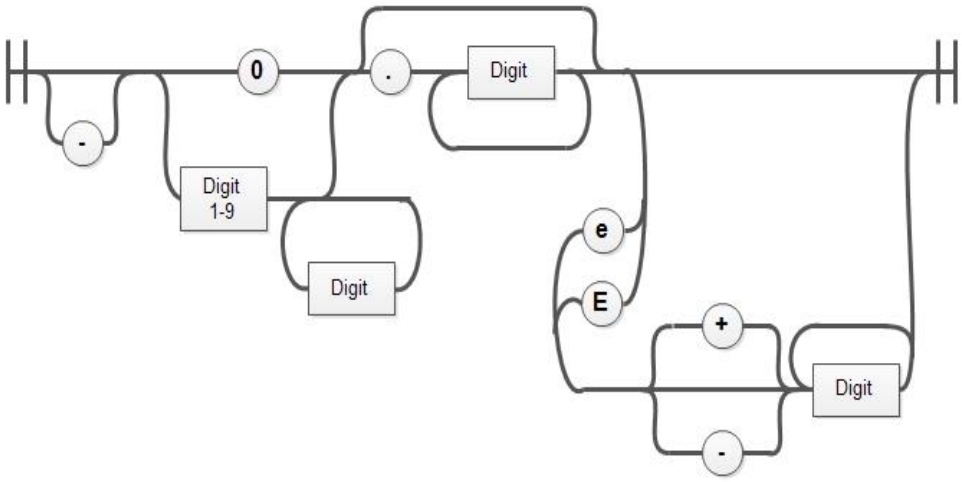
Gambar 2.5 Value

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 sangat mirip dengan string C atau Java.



Gambar 2.6 String

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



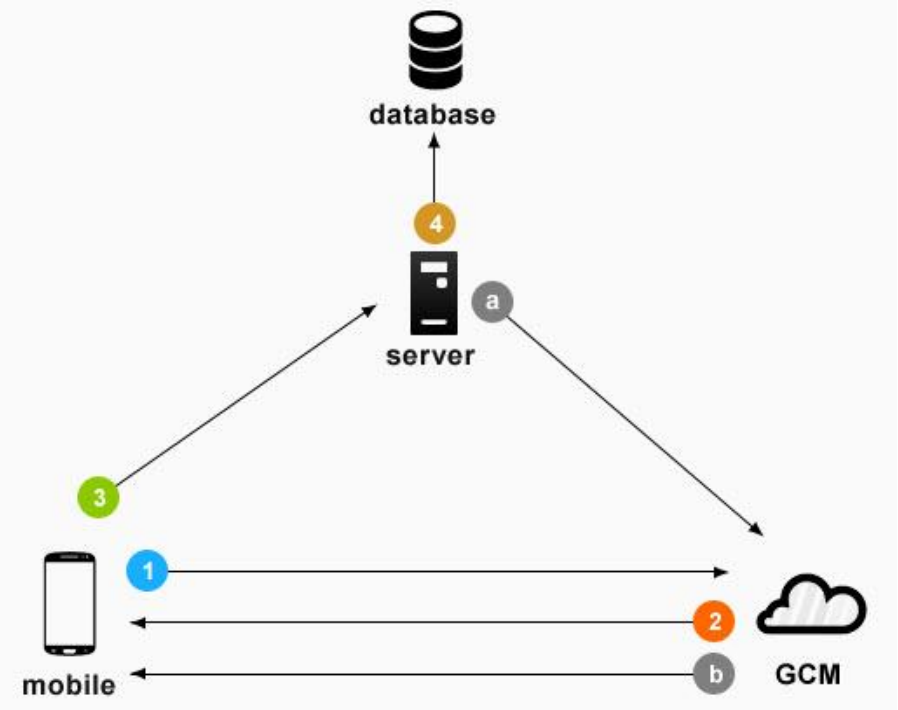
Gambar 2.7 Angka

2.7 Google Cloud Messaging

Google Cloud Messaging untuk Android (GCM) adalah layanan yang membantu pengembang mengirim data dari *server* untuk aplikasi mereka Android pada perangkat Android. Pesan ringan memberitahu aplikasi Android bahwa ada data baru yang akan diambil dari *server* (misalnya, film diunggah oleh seorang teman), atau bias juga pesan yang berisi sampai dengan 4KB data *payload* (sehingga aplikasi seperti *instant messaging* dapat mengkonsumsi pesan langsung). Layanan GCM menangani semua aspek antrian pesan dan pengiriman ke aplikasi target Android berjalan pada perangkat target.

GCM memungkinkan aplikasi android untuk mngirimkan pesan kepada *server* untuk mem*broadcast* sebuah notifikasi kepada seluruh *client* yang ada. Hanya membutuhkan sebuah *account* Gmail maka akan langsung dapat menggunakan fasilitas GCM ini.

Untuk penjelasan lebih lengkap untuk alur kerja google cloud messaging, yaitu [8]:



Gambar 2.8 Google Cloud Messaging

Berdasarkan gambar 2.8 Google Cloud Messaging, terdapat beberapa proses yang terjadi dalam push messaging yaitu :

1. Pertama device Android mengirimkan id pengirim dan id aplikasi ke server GCM untuk registrasi.
2. Apabila registrasi berhasil, server GCM akan menerbitkan id registrasi kepada device Android.
3. Setelah mendapatkan id registrasi, device akan mengirimkan id registrasi ke server.
4. Server akan menyimpan id registrasi pada database untuk penggunaan selanjutnya.

5. Setiap kali notifikasi push dibutuhkan, server akan mengirimkan pesan ke server GCM beserta id registrasi devicenya.
6. Server GCM akan mengirimkan pesan ini ke device Android yang bersangkutan menggunakan id registrasi.

2.8 PHP

PHP adalah kependekan dari PHP Hypertext Preprocessor, bahasa interper yang mirip dengan bahasa C dan Perl yang memiliki kesederhanaan dalam perintah. PHP dapat digunakan bersama dengan HTML sehingga memudahkan dalam pembangunan aplikasi web dengan menciptakan basis data. Interpreter adalah sebuah program yang digunakan untuk membaca file yang akan berisi kode program yang akan dijalankan kemudian interpreter tersebut akan menerima CPU untuk melakukan perintah yang diterimanya. PHP banyak mendukung basis data seperti MySQL, PostgresSQL, Interbase, ODBC, mSQL, Oracle, dan Sybase.

Adapun kelebihan dari PHP dibandingkan dari Bahasa Pemerograman lain yaitu:

1. Bahasa pemrograman PHP adalah sebuah bahasa script yang tidak melakukan sebuah kompilasi dalam penggunaanya.
2. Web Server yang mendukung PHP dapat ditemukan dimana - mana dari mulai apache, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan developer yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.
5. PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan secara runtime melalui console serta juga dapat menjalankan perintah-perintah system.

2.8.1 Penulisan PHP

Penulisan script PHP diawali dan di akhiri dengan sintaks khusus. Ada beberapa cara yang dapat digunakan untuk menuliskan script PHP[13], yaitu :

1. Dengan sintaks “<?php” dan “?>”; sintaks tersebut adalah yang paling umum dari PHP
2. Dengan sintaks “<%” dan “%>”; sintaks tersebut sebebarnya adalah sintaks dari ASP, tetapi dapat digunakan oleh sintaks PHP apabila konfigurasi PHP pilihan asp_tag diaktifkan.
3. Dengan sintaks “<SCRIPT LANGUAGE = "php” dan diakhiri dengan “</SCRIPT>”; sintak itu digunakan untuk deklarasikan sebuah perintah pada HTML.

Berikut contoh programnya :

```
<HTML>
<HEAD>
<TITLE>Pemrograman PHP</TITLE>
</HEAD>
<BODY>
<CENTER>
<?php
echo “Ini contoh program”;
?>
</CENTER>
</BODY>
</HTML>
```

2.8.2 Variabel pada PHP

Variabel merupakan suatu tempat didalam memori komputer yang dialokasikan untuk menyimpan data. Pada PHP, variabel ditandai dengan dollar ($) dan diikuti dengan nama variabelnya. Penamaan variabel mengikuti aturan sebagai berikut[13]:

1. Harus diawali dengan huruf atau underscore (_), dapat diikuti oleh huruf, angka, underscore, atau sembarang karekter ASCII dari 127 hingga 255.
2. Huruf kecil dan besar dibedakan (case sensitive).
3. Tidak boleh mengandung spasi.

Variabel dalam PHP tidak harus diddeklarasikan terlebih dahulu sebelum digunakan, tidak seperti dalam bahasa pemogramman tertentu yang harus mendeklarasikan variabel tertentu.

2.8.3 Fungsi pada PHP

Fungsi adalah kontruksi dalam bahasa pemograman untuk melakukan suatu proses. Dalam pemograman, kita biasa melakukan proses secara berulang kali. Oleh karena itu, fungsi sangat membantu programmer untuk menanggani hal tersebut supaya penulisan kode programnya lebih efisien[13].

Bentuk umum fungsi :

```
Functionnama_fungsi (argumen)
     {
     Kode perintah
     }
```

Fungsi harus terlebih dahulu didefinisikan sebelum digunakan. Pengembalian nilai dari suatu fungsi menggunakan perintah return. Semua jenis data bisa digunakan sebagai nilainya, termasuk list objeck.

2.9 MYSQL

MySQL adalah sebuah perangkat lunak system manajemen basis data SQL (DBMS) yang multithread, dan multi-user. MySQL adalah implementasi dari system manajemen basisdata relasional (RDBMS). MySQL dibuah oleh TcX dan telah dipercaya mengelola system dengan 40 buah database berisi 10.000 tabel dan 500 di antaranya memiliki 7 juta baris. MYSQL menggunakan bahasa dasar SQL untuk mengakses database.

SQL (Structured Query Language) pertama kali diterapkan pada sebuah proyek riset pada laboratorium riset San Jose, IBM yang bernama system R. Kemudian SQL juga dikembangan oleh Oracle, Informix dan Sybase. Dengan menggunakan SQL, proses pengaksesan database lebih user-friendly dibandingan dengan yang lain, misalnya dBase atau Clipper karena mereka masih menggunakan perintah-perintah pemrograman murni. Pernyataan SQL dalam mendefinisikan atribut basis data, table, atribut(kolom), batasan2 terhadap suatu atribut, serta hubungan antar tabel, yaitu :

1. Pernyataan dalam DDL (Data Defination language)

1. Membuat Database

   ```
   Syntax yang digunakan : CREATE DATABASE namadatabase;
   ```

2. Menghapus Database

   ```
   Syntax yang digunakan : DROP DATABASE namadatabase;
   ```

3. Membuat Tabel

   ```
   Syntax yang digunakan : CREATE TABLE namatabel;
   ```

4. Menghapus Database

   ```
   Syntax yang digunakan : DROP DATABASE namadatabase;
   ```

5. Menghapus Tabel

   Syntax yang digunakan : DROP TABEL namatabel;

6. Mendefinisikan null/not null

```
Syntax yang digunakan : CREATE TABLE namatabel
( Field1 TipeData1 NOT NULL,
Field2 TipeData2
);
```

7. Mendefinisikan Nilai Default

```
Syntax yang digunakan : CREATE TABLE namatabel
( Field1 TipeData1,
Field2 TipeData2 DEFAULT nilai
);
```

8. Mendefinisikan Primary Key pada Tabel

   Dapat dilakukan dengan 3 Syntax :

   1)
```
CREATE TABLE namatabel
( Field1 TipeData1 NOT NULL PRIMARY KEY,
Field2 TipeData2
);
```

   2)
```
CREATE TABLE namatabel
( Field1 TipeData1,
Field2 TipeData2,
PRIMARY KEY(Field1)
);
```

   3) ALTER TABEL namatabel ADD CONSTRAINT namaconstraint PRIMARY KEY (NAMAKOLOM)

9. Menghapus Primary Key pada Tabel

   Syntax yang digunakan ada 2 :

   1) `ALTER TABLE namatabel DROP CONSTRAINT NAMACONSTRAINT;`

   2) `ALTER TABLE NAMATABEL DROP PRIMARY KEY;`

10. Menambah Kolom Baru pada Tabel

    `ALTER TABEL NAMATABEL ADD newfield tipe;`

11. Mengubah Tipe Data atau Lebar Kolom pada Tabel

    `Syntax yang digunakan : ALTER TABEL namatabel MODIFY COLUMN field tipe;`

12. Mengubah Nama Kolom

    `ALTER TABEL namatabel CHANGE COLUMN namakolomlama namakolombaru tipe;`

13. Menghapus Kolom pada Tabel

    `ALTER TABEL namatabel DROP COLUMN namakolom;`

14. Mendefinisikan Foreign Key pada Tabel

    Dapat dilakukan dengan 2 Syntax :

    1)
    ```
    CREATE TABLE namatabel
    ( Field1 TipeData1,
    Field2 TipeData2,
    FOREIGN KEY (Field2) REFERENCES namatabelinduk
    (namakolominduk) ON UPDATE CASCADE ON DELETE NO
    ACTION)
    ```

2) `ALTER TABEL namatabel ADD CONSTRAINT namaconstraint FOREIGN KEY (namakolom) REFERENCES namatabelinduk (namakolominduk) ON UPDATE CASCADE ON DELETE NO ACTION;`

15. Menghapus Foreign Key

Syntak yang digunakan : `ALTER TABEL namatabel DROP FOREIGN KEY namaconstraint;`

2. DML (Data Manipulation Language)

DML berfungsi untuk memanipuladi data yang ada di dalam basis data, contohnya untuk pengambilan data, penyisipan data, pengubahan data dan penghapusan data.

Perintah yang digunakan biasanya adalah : INSERT, DELETE, UPDATE, dan SELECT.

1. INSERT

   Menambah baris pada tabel. Syntax yang paling sering digunakan : INSERT INTO namatabel VALUES (nilai1, nilai2, nilai-n);

2. DELETE

   Menghapus baris pada tabel. Syntax : DELETE FROM namatabel [where kondisi];

3. UPDATE

   Mengubah isi beberapa kolom pada tabel. Syntax : UPDATE namatabel SET kolom1=nilai1, kolom2=nilai2 [where kondisi];

4. SELECT

   Menampilkan isi dari suatu tabel yang bisa dihubungkan dengan tabel yang lainnya;

2.10 Web Service

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 internetWeb 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 didalam web service adalah: JSON, XML, SOAP, WSDL dan UDDI[14].

2.11 Location Base Service

Layanan Berbasis lokasi adalah layanan informasi yang dapat diakses melalui mobile device dengan mengunakan mobile network, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari mobile device tersebut. LBS memberikan kemungkinan komunikasi dan interaksi dua arah. Oleh karena itu pengguna memberitahu penyedia layanan untuk mendapatkan informasi yang dia butuhkan, dengan referensi posisi pengguna tersebut. Layanan berbasis lokasi dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu :

Geographic Information System, Internet Service, dan Mobile Devices, hal ini dapat dilihat pada gambar LBS adalah pertemuan dari tiga teknologi [17].

2.11.1 Komponen yang ada pada Location Base Service

Dalam Layanan Berbasis Lokasi terdapat Lima komponen penting yaitu meliputi [17] :

a. Mobile Devices

Suatu alat yang digunakan oleh pengguna untuk meminta informasi yang dibutuhkan. Informasi dapat diberikan dalam bentuk suara, gambar, dan text.

1 Comunication Network

Komponen kedua adalah jaringan komunikasi yang mengirim data pengguna dan informasi yang diminta dari mobile terminal ke Service Provider kemudian mengirimkan kembali informasi yang diminta ke pengguna. Communication network dapat berupa jaringan seluler (GSM, CDMA), Wireless Local Area Network (WLAN), atau Wireless Wide Area Network (WWAN)

2 Positioning Component

Untuk memproses suatu layanan maka posisi pengguna harus diketahui

3 Service and Aplication Provider

Penyedia layanan menawarkan berbagai macam layanan kepada pengguna dan bertanggung jawab untuk memproses informasi yang diminta oleh pengguna.Data and Content Provider Penyedia layanan tidak selalu menyimpan semua data yang dibutuhkan yang bisa diakses oleh pengguna. Untuk itu, data dapat diminta dari data and content provider.

2.12 Struktur Analisis dan Desain

Struktur analisis dan desain adalah sebuah metodologi yang di gunakan pada rekayasa perangkat lunak untuk mendeskripsikan sistem kearah fungsional. Pendekatan ini memecahkan masalah-masalah dalam aktifitas bisnis menjadi bagian-bagian kecil yang dapat disatukan kembali menjadi satu kesatuan yang utuh untuk memecahkan masalah [9].

2.12.1 Diagram Konteks

Diagram konteks adalah diagram yang mencakup masukan-masukan dasar, sistem umum dan keluaran, diagram ini merupkan tingkatan tertinggi dalam diagram aliran data dan hanya memuat satu proses, menunjukan sistem secara keseluruhan, diagram tersebut tidak memuat penyimpanan dan penggambaran aliran data yang sederhana, proses tersebut diberi nomor nol. Semua entitas ekternal yang ditunjukan pada diagram konteks berikut aliran data-aliran datautama menuju dan dari sistem [9].

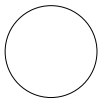
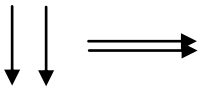
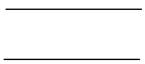
2.12.2 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. DFD terdiri dari empat komponen utama diantaranya adalah

1. Entitas.
2. Proses.
3. Data Store.
4. Data Flow.

DFD dapat dikatakan merupakan breakdown dari diagram konteks. Analisis mengenai sistem yang lebih detail dapat dilakukan dengan melakukan breakdown pada DFD level 1, 2 dan seterusnya. Simbol yang erdapat pada diagram konteks dan DFD dapat dilihat pada Tabel 2.1 Simbol Diagram Konteks dan DFD [15].

Tabel 2.1 Simbol Diagram Konteks dan DFD

Simbol	Nama Simbol	Fungsi
	Entitas Luar	Menggambarkan entitas eksternal yang berhubungan dengan sistem
	Sistem (Konteks) / Proses (DFD)	Menggambarkan proses yang ada dalam suatu sistem
	Aliran Data/Informasi	Menggambarkan aliran data antar proses, data store dan entitas luar
	Data Storage	Mengambarkan tempat penyimpanan data di dalam sistem

2.12.3 Spesifikasi Proses

Spesifikasi Proses adalah tabel yang berisi keterangan atau deskripsi dari semua proses yang terdapat pada DFD. Logika proses yang ditulis pada spesifikasi proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau *pseudecode* tetapi tidak boleh mengkombinasikan kedua bahasa tersebut [15].

Tabel 2.2 Contoh Spesifikasi Proses

No	Proses	Keterangan
1.	No Proses	Menyatakn no proses
	Nama Proses	Menyatakan nama proses
	Source (Sumber)	(Menyatakan sumber data input menuju proses)
	Input	(menyatakan isi data yang masuk ke proses)
	Output	(menyatakan informasi yang keluar dari proses)
	Destination	(menatakan tujuan informasi output dari proses)
	Logika Proses	(menyatakan algoritma dari proses)

2.12.4 Kamus Data

Kamus data adalah tabel yang berisi deskripsi dari data yang mengalir pada DFD, Penjelasan struktur data dapat berupa field tiap data harus sama dengan yang sudah dimodelkan di ERD[15]. Contoh dari kamus data dapat dilihat pada Tabel 2.3 Contoh Kamus Data.

Tabel 2.3 Contoh Kamus Data

Nama aliran data	Data Nama dari data yang digunakan
Where used/How used	Daftar pada poin proses apa data digunakan
Keterangan	Uraian singkat dari data yang digunakan
Struktur Data	Daftar komponen data yang ada pada data yang digunakan
Deskripsi	Jenis data dalam representasi komputer untuk masing-masing data.

2.12.5 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. Tools yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML [16].

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. 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 [16]:

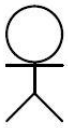
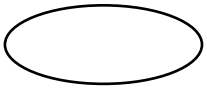
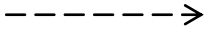
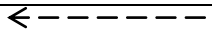
1. Use Case Diagram

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[16]. Berikut adalah simbol yang terdapat pada *use case diagram* dapat dilihat pada Tabel 2.4 Simbol Use Case Diagram.

Tabel 2.4 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 casenya. Berikut adalah dasar format penulisan use case scenario [16]. Dasar pembangunan use case scenario dapat dilihat pada tabel 2.5 Dasar Pembangunan Use Case Scenario.

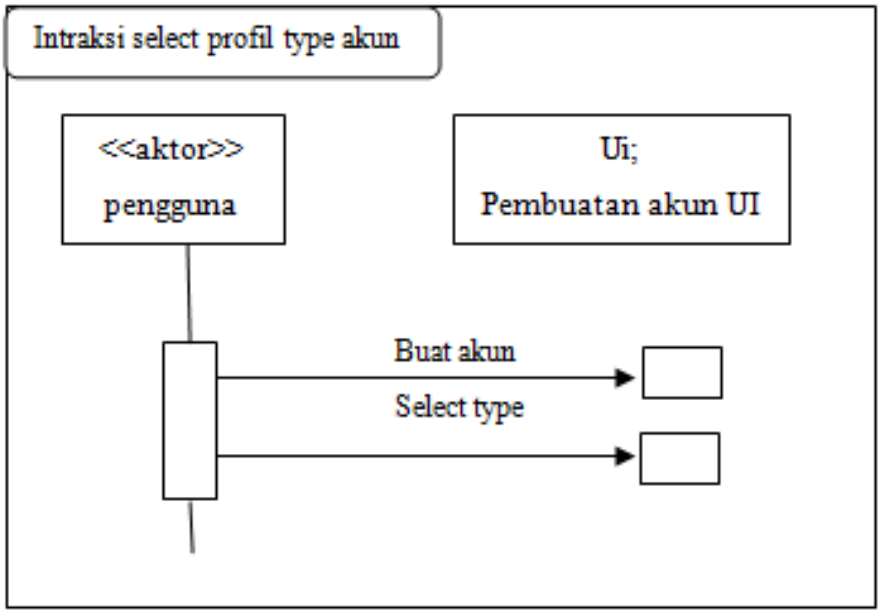
Tabel 2.5 Dasar Pembangunan *Use Case Scenario*

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. 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

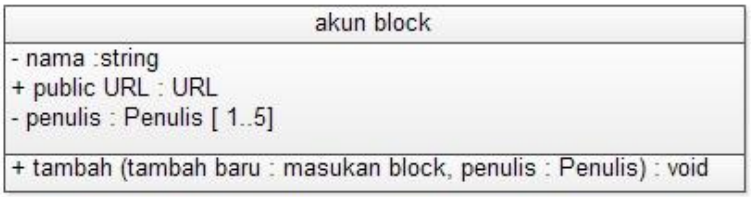
digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari class. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek boundary, controller dan persistent entity[16]. Berikut contoh dari sequence diagram :



Gambar 2.9 Contoh Sequence 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[16].



Gambar 2.10 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 [16].

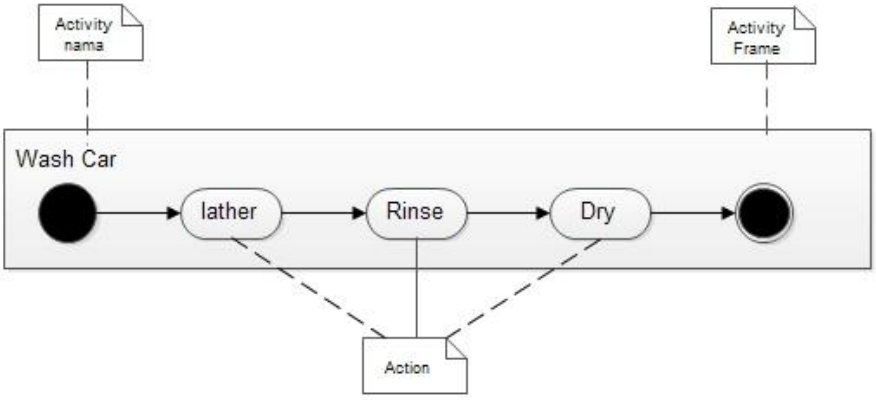
Class memiliki tipe-tipe relationship, diantaranya :

a. Asosiasi, yaitu hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain.
b. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu class di share atau direferensikan kepada objek yang ada di class lain.

c. 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.
d. 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*.
e. 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. 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 [16]. Contoh dari activity diagram dapat dilihat pada Gambar 2.11 Contoh *Activity Diagram*



Gambar 2.11 Contoh *Activity Diagram*

2.12.6 Metode Black-box

Black-box testing merupakan pendekatan pengujian dimana program dianggap sebagai suatu 'black-box' ('kotak hitam') Program test case berbasiskan spesifikasi *Test planning* dapat dimulai sejak awal proses pengembangan sistem.
Black-box testing berfokus pada persyaratan fungsional perangkat lunak. Metode ini memungkinkan *software developer* untuk mendapatkan serangkaian kondisi input yang mempergunakan semua persyaratan fungsional program.

Pengujian blackbox berusaha menemukan kesalahan dalam kategori :

a. Fungsi – fungsi yang tidak benar atau hilang
b. Kesalahan *interface*
c. Kesalahan dalam struktur data atau akses database eksternal
d. Kesalahan kinerja
e. Inisialisasi dan kesalahan terminasi

Equivalence partitioning yaitu metode pengujian *black-box* yang membagi domain input dari suatu program ke dalam kelas data dari mana *test case* dapat dilaksanakan. Input data dan output hasil terdapat di kleas yang berbeda yang sesuai dengan kelas inputnya. Masing-masing kelas *equivalensi partition* diproses dimana program akan memproses anggota kelas-kelas tersebut secara equivalen. *Test cases* dipilih dari masing-masing partisi

Partisi Ekivalensi (*equivalensi partition*) adalah Input data dan output hasil terdapat di kelas yang berbeda yang sesuai dengan kelas inputnya. Masing-masing kelas *equivalensi partition* dimana program akan memproses anggota kelas-kelas tersebut secara *equivale*

Integration testing adalah pengujian keseluruhan sistem atau subsistem yang terdiri dari komponen yang terintegrasi. Tes integrasi menggunakan *black box* dengan *test case* ditentukan dai spesifikasi. Kesulitannya adalah menemukan atau melokasikannya.

2.12.7 Metode Topsis

TOPSIS didasarkan pada konsep dimana alternatif terpilih yang terbaik tidak hanya memiliki jarak terpendek dari solusi ideal positif, namun juga memiliki jarak terpanjang dari solusi ideal negatif (Kusumadewi, 2006:87). Konsep ini banyak digunakan pada beberapa model MADM untuk menyelesaikan masalah keputusan secara praktis. Hal ini disebabkan konsepnya sederhana dan mudah dipahami, komputasinya efisien, dan memiliki kemampuan untuk mengukur kinerja relatif dari alternatif-alternatif keputusan dalam bentuk matematis yang sederhana [2].

Secara umum, prosedur TOPSIS mengikuti langkah-langkah sebagai berikut:

a. Membuat matriks keputusan yang ternormalisasi;
b. Membuat matriks keputusan yang ternormalisasi terbobot;
c. Menentukan matriks solusi ideal positif dan matriks solusi ideal negatif;
d. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matriks solusi ideal negatif;
e. Menentukan nilai preferensi untuk setiap alternatif.

TOPSIS membutuhkan rating kerja setiap alternatif A_i pada setiap kriteria C_j yang ternormalisasi

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

dengan i=1,2,...,m; dan j=1,2,...,n

dimana :

r_{ij} = matriks ternormalisasi [i][j]

x_{ij} = matriks keputusan [i][j]

Solusi ideal positif A^+ dan solusi ideal negatif A^- dapat ditentukan berdasarkan rating bobot ternormalisasi (y_{ij}) sebagai :

y_{ij} = wi.rij ; dengan i=1,2,...,m; dan j=1,2,...,n

$A^+ = (y_1^+, y_2^+, ..., y_n^+)$;

$A^- = (y_1^-, y_2^-, ..., y_n^-)$;

dimana :

y_{ij} = matriks ternormalisasi terbobot [i][j]

w_i = vektor bobot[i] dari proses AHP

y_j^+ = max y_{ij}, jika j adalah atribut keuntungan
min y_{ij}, jika j adalah atribut biaya

y_j^- = min y_{ij}, jika j adalah atribut keuntungan

max y_{ij}, jika j adalah atribut biaya

j = 1,2,...,n

Jarak antara alternatif A_i dengan solusi ideal positif :

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

dimana :

D_i^+ = jarak alternatif Ai dengan solusi ideal positif

y_i^+ = solusi ideal positif[i]

y_{ij} = matriks normalisasi terbobot[i][j]

Jarak antara alternatif A_i dengan solusi ideal negatif :

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

dimana :

D_i^- = jarak alternatif Ai dengan solusi ideal negatif

y_i^- = solusi ideal positif[i]

y_{ij} = matriks normalisasi terbobot[i][j]

Nilai preferensi untuk setiap alternatif (V_i) dapat dilihat pada rumus (2.11).

$$Vi = \frac{D_i^-}{D_i^- + D_i^+} \quad ; \quad i=1,2,...,m$$

dimana :

V_i = kedekatan tiap alternatif terhadap solusi ideal

D_i^+ = jarak alternatif Ai dengan solusi ideal positif

D_i^- = jarak alternatif Ai dengan solusi ideal negatif

Nilai V_i yang lebih besar menunjukkan bahwa alternatif A_i lebih dipilih.