BAB 2
TINJAUAN PUSTAKA

2.1 Pengertian Bola Basket

Bola basket adalah olahraga berkelompok yang terdiri atas dua tim yang masing-masing tim terdiri dari lima orang. Bola basket biasa dimainkan di ruang olahraga tertutup dan hanya memerlukan lapangan yang *relative* kecil. Bola basket juga mudah dipelajari karena bentuk bolanya yang besar, sehingga tidak menyulitkan pemain ketika memantulkan atau melempar bola tersebut.

Berbeda dengan sepakbola, olahraga ini dilakukan di lapangan yang tidak terlalu luas, sehingga dapat di lakukan diruangan terbuka maupun tertutup. Olahraga ini juga tidak menggunakan kaki tetapi menggunakan tangan. Para pemain dalam tim bekerja sama untuk mencetak *goal* sebanyak-banyaknya ke dalam keranjang. Dalam permainan ini kerjasama dalam tim sangat diperlukan. Selain itu, ketelitian dan ketepatan dalam memasukkan bola ke keranjang lawan juga sangat diperlukan untuk mencapai kemenangan .

Bola yang diperebutkan tidak ditendang tetapi di oper menggunakan tangan. Olahraga ini hampir mirip dengan olahraga sepakbola, akan tetapi menggunakan tangan. Para pemain menggiring bola dengan memantul-mantulkan bola ke lantai, sambil membawa dan mengarahkan bola ke sarang lawan atau yang biasa disebut dengan men*dribble* bola .

Peraturan dalam olahraga ini pun lebih banyak dan lebih rumit daripada olahraga sepakbola. Akan tetapi ternyata olahraga ini banyak penggemarnya, umumnya dikalangan muda. Olahraga ini cukup menyenangkan dan menyehatkan badan [1].

Berdasarkan referensi yang ada dapat disimpulkan bahwa permainan olahraga basket yaitu permainan yang berjumlah 1 timnya yaitu 5 orang dimana aturan bermainnya yaitu kita harus men*dribble* bola (memantulkan ke lantai) ke arah sarang lawan untuk memasukan bola tersebut ke keranjang lawan agar dapat mencapai kemenangan.

2.1.1 Sejarah Olahraga Basket

Olahraga basket diciptakan oleh seorang guru olahraga asal Kanada bernama, Dr. James Naismith pada tahun 1891. Olahraga ini bahkan dikatakan sebagai olahraga yang unik pada masa itu karena konon, olahraga basket diciptakan secara tidak sengaja. Dr. James Naismith mengajar di sebuah perguruan tinggi untuk para siswa professional di YMCA di Springfield, Massachusetts.

Pada waktu itu Dr. James Naismith diberi tugas untuk membuat suatu permainan baru yang dapat dimainkan di ruangan tertutup sebagai kegiatan untuk mengisi waktu para siswa pada masa liburan musim dingin di New England. Setelah melakukan percobaan, akhirnya Naismith membuat suatu permainan yang menggunakan bola yang bentuknya bulat dan ukurannya tidak terlalu kecil. Permainan tersebut tidak menggunakan kekerasaan, tidak menjegal seperti pada permainan sepak bola dan tidak menggunakan gawang sebagai sasarannya melainkan sebuah keranjang yang digantung pada sebuah tihang.

Pada olahraga ini pemain tidak menendang bola tetapi mengoper bola menggunakan tangan, serta men*dribble* bola, gawang diganti dengan sasaran lain yang sempit dan terletak diatas para pemain sehingga tidak terjadi aksi kekerasaan seperti menjegal dan tidak menggunakan kekerasaan, gawang tersebut diganti dengan ring basket atau semacam keranjang. Keterampilan yang digunakan dalam olahraga ini bukan kekuatan tetapi terletak pada ketepatan menembakkan bola kedalam keranjang. Permainan tersebut terinspirasi dari permainan yang pernah Dr. James Naismith mainkan saat kecil di Ontario, Naismith menciptakan permainan yang sekarang dikenal sebagai bola basket pada 15 Desember 1891.

Pada Agustus 1936, saat menghadiri Olimpiade Berlin 1936, Dr. James Naismith dinamakan sebagai Presiden Kehormatan Federasi Bola Basket Internasional. Dr. James Naismith terlahir di Kanada, kemudian pada 4 Mei 1925 ia berpindah kewarnegaraan menjadi warga negara Amerika Serikat. Naismith meninggal dunia 28 November 1939, kurang dari enam bulan setelah menikah untuk kedua kalinya.

Pertandingan resmi bola basket yang pertama, diselenggarakan pada tanggal 20 Januari 1982 di tempat kerja Dr. James Naismith. Basket adalah sebutan yang

diucapkan oleh seorang muridnya. Dalam Bahasa inggris olahraga ini disebut *Basketball* karena Olahraga ini menggunakan unsur basket (keranjang) dan ball (bola). Olahraga ini pun menjadi terkenal di negara Amerika Serikat. Dan mulai banyak digemari oleh masyarakat amerika serikat. Berbagai tim dan *club* basket terbentuk dengan cepat. Pertandingan demi pertandinganpun segera dilaksanakan di kota-kota di seluruh negara bagian Amerika Serikat. Hingga saat ini menjadi olahraga yang dikenal oleh seluruh Negara Internasional [1].

2.1.2 Aturan Dasar Olahraga Basket

Pada awalnya, setiap tim berjumlah sembilan orang dan tidak ada *dribble*, sehingga bola hanya dapat berpindah melalui lemparan. Sejarah peraturan permainan basket diawali dari 13 aturan dasar yang ditulis sendiri oleh James Naismith. Aturan dasar tersebut adalah sebagai berikut. Bola dapat dilemparkan ke semua arah dengan menggunakan salah satu tangan atau dua tangan.

1. Bola dapat dipukul ke segala arah dengan menggunakan salah satu atau kedua tangan, tetapi tidak boleh dipukul menggunakan kepalan tangan(meninju).
2. Pemain tidak diperbolehkan berlari sambil memegang bola. Pemain harus melemparkan bola tersebut dari titik tempat menerima bola, tetapi diperbolehkan apabila pemain tersebut berlari pada kecepatan biasa. Intinya, setiap pemain tidak boleh terlalu lama memegang bola, bola harus di *dribble* atau di oper ke teman satu tim.
3. Bola harus dipegang dengan telapak tangan. Lengan atau anggota tubuh lainnya tidak diperbolehkan memegang bola. Juga tidak boleh menendang bola.
4. Pemain tidak diperbolehkan menyeruduk, menahan, mendorong, memukul pemain lawan dengan cara disengaja. Apabila hal tersebut dilakukan maka termasuk sebagai pelanggaran. Sangsi terhadap pelanggaran-pelanggaran yang dilakukan adalah :
    a. Pelanggaran pertama terhadap peraturan ini akan dihitung sebagai kesalahan.
    b. Pelanggaran kedua akan diberi sangsi berupa diskualifikasi pemain pelanggar hingga keranjang tim nya dimasuki oleh bola lawan.

c. Apabila pelanggaran tersebut dilakukan denga tujuan untuk mencederai lawan, maka pemain pelanggar akan dikenai hukuman tidak boleh ikut bermain sepanjang pertandingan. Pada masa ini, tim tidak diperbolehkan mengadakan pergantian pemain.

5. Apabila ada pemain memukul bola dengan kepalan tangan(meninju), maka dihitung sebagai suatu kesalahan bagi tim.
6. Apabila salah satu pihak melakukan tiga kesalahan berturut-turut, maka kesalahan itu akan dihitung sebagai gol untuk lawannya (berturut-turut berarti tanpa adanya pelanggaran balik yang dilakukan oleh lawan).
7. Apabila bola yang dilemparkan atau dipukul dari lapangan masuk ke dalam keranjang, dan pemain yang menajga keranjangan tidak menyentuk atau mengganggu gol tersebut maka bola yang masuk dihitung sebagai gol. Tetapi apabila bola terhenti di pinggir keranjang atau pemain lawan menggerakkan keranjang, maka hal tersebut tidak akan dihitung sebagai sebuah gol.
8. Apabila bola keluar lapangan pertandingan bola akan dilemparkan kembali ke dalam dan dimainkan oleh pemain pertama yang menyentuhnya. Apabila terjadi perbedaaan pendapat tentang tentang kepemilikan bola, maka wasit yang akan melemparkannya ke dalam lapangan. Pemain yang melembar bola diberi waktu 5 detik untuk melemparkan bola. Apabila ia memegang lebih lama dari waktu tersebut, maka kepemilikan bola akn berpindah. Apabila sah satu pihak melakukan hal yang dapat *menu*nda pertandingan, maka wasit dapat memberi mereka sebuah peringantan pelanggaran.
9. Wasit berhak untuk memperhatikan permainan para pemain dan mencatat jumlah pelanggaran dan memberi tahu wasit pembantu apabila terjadi pelanggaran berturut-turut. Wasit memiliki hak penuh untuk memberikan diskualifikasi pemain yang melakukan pelanggaran sesuai dengan yang tercantum dalam aturan 5.
10. Wasit pembantu memperhatikan bola dan mengambil keputusan apabila bola dianggap telah keluar lapangan, pergantian kepemilikan bola, serta menghitung waktu. Wasit pembantu berhak menentukan sah tidaknya suatu gol dan menghitung jumlah gol yang terjadi.

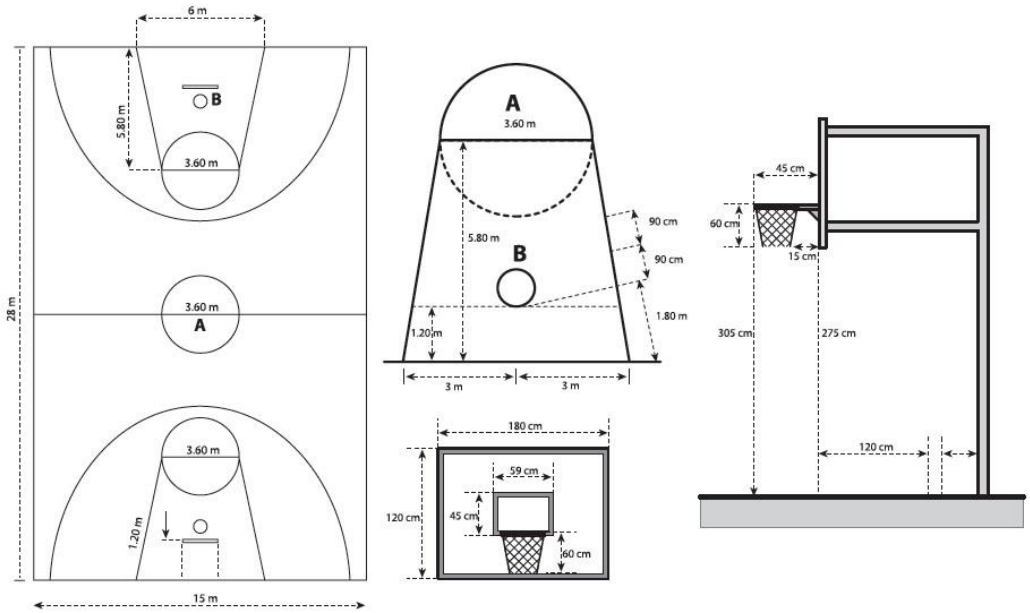
11. Waktu pertandingan adalah 4 babak masing-masing babak waktunya 10 menit.
12. Pihak yang berhasil memasukkan gol terbanyak akan dinyatakan sebagai pemenang [1].

2.1.3 Lapangan Bola Basket

Permainan bola basket dilakukan di sebuah lapangan empat persegi panjang dengan ukuran sebagai berikut :

1. Panjang garis samping lapangan 28 meter.
2. Lebar lapangan 15 meter; sesuatu hal, ukuran panjang dan lebar
3. tersebut dapat dikurangi, tetapi harus seimbang.
4. Garis tengah lingkaran di tengah lapangan 3,6 meter.
5. Tinggi ring basket 2,75 meter.
6. Diameter ring basket 0,45 meter.
7. Ukuran papan pantul panjang x lebar: 1,80m x 1,20m.

Untuk lebih jelasnya, perhatikan gambar berikut :



Gambar 2.1 Ukuran Lapangan Dan Ring Basket

2.1.4 Unsur Dasar Bola Basket

Ada 3 cara dasar menggerakkan bola dalam bola basket, yaitu menggiring (*dribbling*), operan (*passing*) dan tembakan (*shooting*);

1. Menggiring (*dribbling*) adalah cara untuk bergerak dengan bola yang dilakukan oleh seorang pemain. Tujuannya untuk membebaskan diri dari lawan atau mencari posisi bagus untuk mengoper atau menembak.
2. Mengoper (*passing*) adalah cara tercepat dan terefektif memindahkan bola dari satu pemain ke pemain lain.
3. Menembak (*shooting*) adalah gerakan terakhir untuk mendapatkan angka. Umumnya dalam bola basket, tembakan dilakukan setiap 15 – 20 detik dan hampir setengahnya berhasil masuk. Banyaknya tembakan masuk yang terjadi membuat bola basket menarik, atraktif dan menegangkan bagi penonton [1].

2.1.5 Klasemen Basket

Klasemen basket adalah suatu sistem penilaian tertentu sehingga dapat diketahui peringkat terbaik atau sang pemenang. Tabel klasemen merupakan salah satu metode untuk menentukan peringkat [1].

Di dalam dunia basket, klasemen berarti penentuan tingkatan sebuah tim. Adapun berikut ini akan dijelaskan metode atau cara perhitungan klasemen basket *menu*rut standar nasional berdasarkan perhitungan klasemen IBL(*Indonesian Basketball League*).

Tabel 2.1 Metode Perhitungan Klasemen Basket

No	Grup Putih	MP	W	WO	LO	L	Pts	Ptc
1	Aspac Jakarta	17	16	0	0	1	33	4.941
2	Surabaya	17	12	0	0	5	29	4.705
3	Satya Wacana	17	10	1	0	6	28	4.647

1. MP (*Match Played*) adalah *Total* pertandingan yang telah dimainkan.
2. W(*Win*) adalah *total* kemenangan selama melakukan pertandingan.
3. WO (*Win Overtime*) adalah *Total* kemenangan pertandingan pada saat pertambahan waktu.
4. LO (*Lose Overtime*) adalah *total* kekalahan pertandingan pada saat pertambahan waktu.

5. L (*Lose*) adalah *total* kekalahan selama melakukan pertandingan.
6. Pts *Total* adalah *total score* memasukan ke ring lawan dan kemasukan oleh tim lawan selama pertandingan yang telah dimainkan.
7. Pts adalah *Total* Poin selama melakukan pertandingan setiap *win* mendapatkan 2 *point* dan setiap lose mendapatkan 1 *point*. Cara untuk mendapatkan pts ini seperti yang ada pada persamaan 2.1.

$$Pts = (W * 2) + (L * 1) \quad \text{..... (2.1)}$$

8. Ptc adalah presentase kemenangan tim. Cara untuk mendapatkan ptc ini yaitu seperti yang ada pada persamaan 2.2.

$$Ptc = (W + WO)/MP \quad \text{..... (2.2)}$$

Didalam penelitian ini peneliti akan menggunakan metode perhitungan klasemen standar nasional *Indonesian Basket League* (IBL)[1]. Fungsi dari fitur klasemen ini yaitu untuk dapat menilai kemampuan lawan.

2.2 Metode TOPSIS

TOPSIS (*Technique For Others Reference by Similarity to Ideal Solution*) adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang (1981). TOPSIS menggunakan prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal. Solusi ideal positif didefinisikan sebagai jumlah dari seluruh nilai terbaik yang dapat dicapai untuk setiap atribut, sedangkan solusi negatif-ideal terdiri dari seluruh nilai terburuk yang dicapai untuk setiap atribut.

TOPSIS mempertimbangkan keduanya, jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif. Berdasarkan perbandingan terhadap jarak relatifnya, susunan prioritas alternatif bisa dicapai. Metode ini banyak digunakan untuk menyelesaikan pengambilan keputusan secara praktis. Hal ini disebabkan konsepnya sederhana dan mudah dipahami, komputasinya efisien,dan memiliki kemampuan mengukur

kinerja relatif dari alternatif-alternatif keputusan [6]. Tahapan dalam metode TOPSIS yaitu :

1. Menentukan data kriteria dan data alternatif
2. Membuat matriks keputusan ternormalisasi
3. Membuat matriks keputusan yang ternormalisasi terbobot
4. Membuat matriks solusi ideal positif dan matriks solusi ideal negatif
5. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matriks solusi ideal negatif.
6. Menentukan nilai preferensi untuk setiap alternatif

Berikut adalah tahapan langkah - langkah yang lebih detail mengenai proses perhitungan TOPSIS :

1. Menggambarkan alternatif (m) dan kriteria (n) ke dalam sebuah matriks, dimana Xij adalah pengukuran pilihan dari alternatif ke-i dan kriteria ke-j. Matriks ini dapat dilihat pada persamaan 2.3.

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad \text{..... (2.3)}$$

2. Membuat matriks keputusan ternormalisasi

   Setiap normalisasi dari nilai r_{ij} dapat dilakukan dengan perhitungan menggunakan persamaan 2.4.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \quad \text{..... (2.4)}$$

   Keterangan :

   x = nilai matriks

   i = baris matriks

   j = kolom matriks

   m = banyaknya alternatif

3. Membuat matriks keputusan yang ternormalisasi terbobot

   Langkah selanjutnya setelah membuat normalisasi, yaitu membuat matriks keputusan ternormalisasi terbobot. Jadi di tahap ini terdapat perkalian matriks dengan bobot preferensi[6]. menggunakan persamaan 2.5.

$$yij = W_i r_{ij} \quad \text{..... (2.5)}$$

   Keterangan :

   y = bobot ternormalisasi

   i = baris matriks

   j = kolom matriks

   W = nilai bobot untuk setiap alternatif yang ditentukan

   r = hasil ternormalisasi langkah 1

4. Membuat solusi ideal positif dan solusi ideal negatif

   Persamaan yang digunakan untuk dapat menentukan solusi ideal positif dan solusi negatif dapat ditentukan berdasarkan *Rating* bobot ternomalisasi pada persamaan 2.5. proses perhitungan untuk membuat solusi ideal positif seperti pada persamaan 2.6, lalu persamaan 2.7 ini merupakan proses pehitungan untuk mendapatkan solusi ideal negatif.

$$y_j^+ = \begin{cases} \max y_{ij} \,; Jika\ j\ adalah\ atribut\ keuntungan \\ \min y_{ij} \,; Jika\ j\ adalah\ atribut\ biaya \end{cases} \quad \text{..... (2.6)}$$

$$y_j^- = \begin{cases} \min y_{ij} \,; Jika\ j\ adalah\ atribut\ keuntungan \\ \max y_{ij} \,; Jika\ j\ adalah\ atribut\ biaya \end{cases} \quad \text{..... (2.7)}$$

5. Membuat jarak alternatif Ai dengan solusi ideal positif jarak antara alternatif Ai dengan solusi ideal positif

   Ditahap ini yaitu untuk menentukan jarak antara nilai terbobot setiap alternatif terhadap solusi ideal positif. Untuk menentukan jarak antara nilai terbobot setiap alternatif terhadap solusi ideal positif, digunakan persamaan 2.8

$$D_i^+ = \sqrt{\sum_{j=1}^{n}\left(y_i^+ - y_j^i\right)^2} \qquad \text{..... (2.8)}$$

Keterangan :

n = kriteria

Y = bobot ternomalisasi

i = baris matriks

j = kolom matriks

6. Membuat jarak alternatif Ai dengan solusi ideal negatif jarak antara alternatif Ai dengan solusi ideal negatif

   Sedangkan untuk menghitung jarak antara nilai terbobot setiap alternatif terhadap solusi ideal negatif, digunakan persamaan 2.9.

$$D_i^- = \sqrt{\sum_{j=1}^{n}\left(y_i^- - y_j^-\right)^2} \qquad \text{..... (2.9)}$$

   Keterangan :

   n = kriteria

   Y = bobot ternomalisasi

   i = baris matriks

   j = kolom matriks

7. Menentukan nilai preferensi untuk setiap preferensi**,**

   Tahap ini adalah tahap terakhir dalam proses perhitungan TOPSIS. Jadi disini akan menentukan nilai preferensi untuk setiap alternatif yang ada dengan menggunakan persamaan 2.10

$$V_i = \frac{D_i^-}{D_i^- + D_i^+} \qquad \text{..... (2.10)}$$

   Keterangan :

   D_i^- = Jarak nilai terbobot terhadap solusi ideal negatif

   D_i^- = Jarak nilai terbobot terhadap solusi ideal positif

Setelah didapat nilai V_i, maka alternatif dapat diranking berdasarkan urutan V_i. Dari hasil perankingan ini dapat dilihat alternatif terbaik yaitu alternatif yang memiliki jarak terpendek dari solusi ideal dan berjarak terjauh dari solusi ideal negatif.

2.3 Aplikasi

Aplikasi adalah program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan.

Pengertian aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinnya, aplikasi merupakan suatu perangkat komputer yang siap pakai bagi *User* [7].

Kesimpulan peneliti dari pembahasan diatas yaitu aplikasi adalah suatu program yang siap digunakan oleh pengguna untuk mendapatkan hasil yang akurat berdasarkan tujuan dibuatnya aplikasi tersebut.

2.4 *Android*

Android adalah aplikasi sistem operasi untuk telepon seluler yang berbasis Linux. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. Google menyediakan peningkatan versi bertahap utama untuk sistem operasi *Android* setiap enam hingga sembilan bulan, menggunakan nama bertema makanan [8]. Didalam penelitian ini peneliti memilih versi sistem operasi *Android Jelly Bean* (API Level 17) ke atas.

2.5 *Android Life Cycle*

Aplikasi terdiri dari satu atau lebih komponen yang didefinisikan dalam *file* manifiles aplikasi. Komponen-komponen aplikasi tersebut adalah sebagai berikut[9] :

1. *Activity*
2. *Service*
3. *Broadcast receiver*
4. *Content provider*

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, Content Provider*.

Berikut adalah penjelasan tentang komponen-komponen aplikasi *Android* :

1. *Activity*

   Activity akan menyajikan *User Interface* (UI) kepada pengguna sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi *Android* bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* yang bergantung pada tujuan aplikasi dan desain aplikasi itu sendiri. Untuk berpindah dari satu *activity* ke *activity* lain dapat dilakukan menggunakan sebuah *trigger* seperti klik tombol pada tampilan aplikasi.

2. *Service*

   Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*. Sebagai contoh dalam memutar musik, *service* mungkin memutar musik atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Apabila sebuah pemutar musik sedang memutar lagu dari *list* yang ada, aplikasi akan memiliki dua atau lebih *activity* yang memungkinkan pengguna untuk memutar sambil memilih lagu baru. Untuk menjaga musik tetap berjalan sebuah *activity* dapat menjalankan *service*.

3. *Broadcast Receiver*

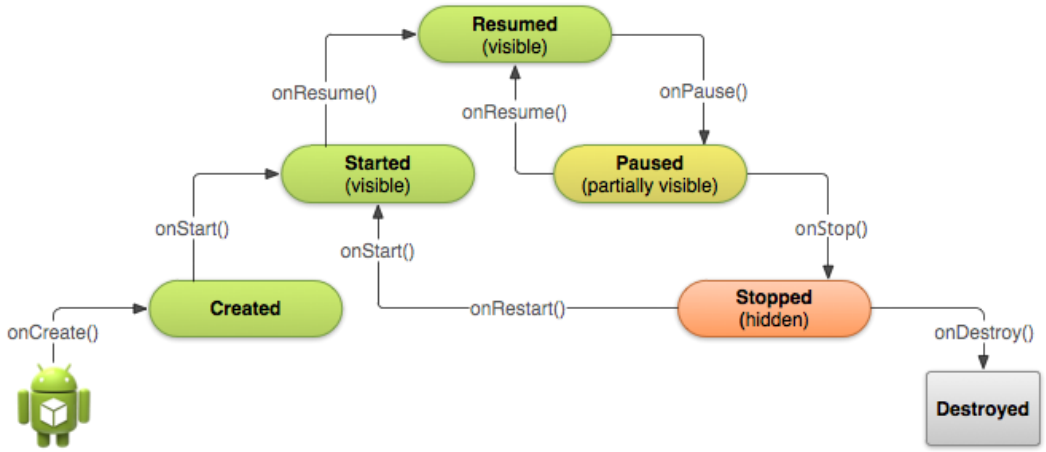
   Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *Broadcast* seperti notifikasi zona waktu telah berubah,

baterai lemah, gambar berhasil diambil oleh kamera, dan lain-lain. Aplikasi juga dapat menginisialiasi *Broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah diunduh ke perangkat dan siap untuk digunakan.

4. *Content Provider*

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

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.



Gambar 2.2 *Android Lifecycle*

Kondisi – kondisi pada Gambar 2.2 *Android Lifecycle* diatas bisa bekerja jika menggunakan *method-method* yang disediakan oleh *Android*.

Tabel 2.2 *Method-Method Android Lifecyle*

Metode	Keterangan	Bisa Dimatikan Setelahnya	Metode Berikutnya
onCreate()	Dipanggil saat aktivitas pertama kali dibuat. Disinilah kita harus melakukan semua persiapan pada tampilan utama. Yang selanjutnya akan menjalankan *method* onStart()	Tidak	onStart()
onRestart()	Dipanggil setelah *Activity* dihentikan, hingga nantinya di panggil lagi dan menjalankan *method* onStart()	Tidak	onStart()
onStart()	Dipanggil tepat sebelum *Activity* terlihat di layar.Diikuti dengan *method* onResume() saat *Activity* sudah muncul dilayar dan *method* onStop() jika *Activity* ingin disembunyikan.	Tidak	onResume() atau onStop()
onResume()	Dipanggil tepat sebelum *Activity* mulai berinteraksi dengan pengguna. Pada kondisi ini, *activity* utama berada di puncak tumpukan *Activity* yang ada. Dan selalu diikuti oleh *method* onPa*Use*().	Tidak	onPa*Use*()
onPa*Use*()	Dipanggil bila sistem akan memulai proses untuk *Activity* lain. *Method* ini biasanya digunakan untuk menerapkan (commit) perubahan yang tidak tersimpan pada data persisten, menghentikan animasi dan hal-hal lain yang mungkin menghabiskan CPU, dan sebagainya.	Ya	OnResume() atau onStop()
onStop()	Dipanggil bila *Activity* tidak lagi terlihat dilayar. Hal ini bisa terjadi karena *Activity* sedang ditutup, atau karena adanya *activity* lain (yang baru). Diikuti oleh *method* onRestart() jika *Activity* kembali dipanggil untuk berinteraksi dengan pengguna, atau *method* onDestroy() jika *activity* benar-benar ingin dihilangkan.	Ya	onRestart() atau onDestroy()
onDestroy()	Dipanggil sebelum *activity* benar-benar ingin tutup. Inilah panggilan terakhir yang akan diterima *Activity*.	Ya	Tidak ada

2.6 *Global Positioning System*

Global Positioning Sytem adalah sistem navigasi berbasis satelit terdiri dari jaringan 24 satelit ditempatkan ke orbit oleh Departemen Pertahanan Amerika

Serikat yang pertama kali diperkenalkan mulai tahun 1978. Layanan GPS dahulu hanya dipergunakan untuk keperluan militer namun mulai terbuka untuk publik. 24 satelit GPS tersebut berada sekitar 12.*0*00 mil di atas bumi bergerak mengelilingi bumi 12 jam dengan kecepatan 7.*0*00 mil per jam. Satelit GPS berkekuatan energi sinar matahari, memiliki baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak ada energi matahari dan memiliki roket penguat kecil pada masing-masing satelit agar dapat menGORbit tepat pada tempatnya.

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

1. Waktu. GPS *receiver* menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.
2. Lokasi. GPS memberikan informasi lokasi dalam tiga dimensi:
    a. Latitude
    b. Longitude
    c. Elevasi
3. Kecepatan. Ketika berpindah tempat, GPS dapat *menu*njukkan informasi kecepatan berpindah tersebut.
4. Arah perjalanan. GPS dapat *menu*njukkan arah tujuan.
5. Simpan lokasi. Tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh GPS *receiver*.
6. Komulasi data. GPS *receiver* dapat menyimpan informasi *track*, seperti *total* perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/jam sampai tujuan, dan sebagainya [10].

2.7 Google *Maps* API

Google*Maps* adalah peta *online* atau membuka peta secara *online*, dapat dilakukan secara mudah melalui layanan gratis dari Google. Bahkan layanan ini

menyediakan API (*Application Programming Interface*) yang memungkinkan *developer* lain untuk memanfaatkan aplikasi ini di aplikasi buatannya. Tampilan Google*Maps* pun dapat dipilih, berdasarkan foto asli atau peta gambar *rute* saja.

Google*Maps* adalah layanan gratis yang diberikan oleh Google dan sangat popular. Google*Maps* adalah suatu peta dunia yang dapat kita gunakan untuk melihat suatu daerah. Dengan kata lain, Google*Maps* merupakan suatu peta yang dapat dilihat dengan menggunakan suatu *browser*. Kita dapat menambahkan fitur Google*Maps* dalam *Web* yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan Google*Maps* API. Google*Maps* API adalah suatu library yang berbentuk JavaScript.

Cara membuat Google*Maps* untuk ditampilkan pada suatu aplikasi sangat mudah hanya dengan membutuhkan pen*get*ahuan mengenai HTML serta JavaScript, serta koneksi Internet yang sangat stabil. Dengan menggunakan Google*Maps* API, kita dapat menghemat waktu dan biaya untuk membangun aplikasi peta digital yang handal, sehingga kita dapat fokus hanya pada data-data yang akan ditampilkan. Dengan kata lain, kita hanya membuat suatu data sedangkan peta yang akan ditampilkan adalah milik Google sehingga kita tidak dipusingkan dengan membuat peta suatu lokasi, bahkan dunia.

Pada Google*Maps* API terdapat 4 jenis pilihan model peta yang disediakan oleh Google, diantaranya adalah:

1. *ROADMAP*, untuk menampilkan peta biasa 2 dimensi
2. *SATELLITE*, untuk menampilkan foto satelit
3. *TERRAIN*, untuk *menu*njukkan *relief* fisik permukaan bumi dan *menu*njukkan seberapa tingginya suatu lokasi, contohnya akan *menu*njukkan gunung dan sungai
4. *HYBRID*, akan *menu*njukkan foto satelit yang diatasnya tergambar pula apa yang tampil pada ROAD*MAP* (jalan dan nama kota) [11].

Dalam penelitian ini. Google *Maps* API menjadi sebuah fungsi yang diterapkan dalam sistem atau perangkat lunak yang akan dibangun. Jadi dengan memanfaatkan API yang terdapat dalam Google *Maps* ini yaitu untuk mengetahui alamat persewaan lapangan basket dan *Event* yang akan diselenggarakan, atau lebih

jelasnya dengan adanya fitur navigasi yang dimiliki oleh Google *Maps*. Pengguna bisa lebih mudah dalam melakukan akses lokasi terhadap tempat persewaan lapangan basket yang akan diadakannya pertandingan adu tanding dari kedua belah pihak tim tersebut dan juga *Event* yang sedang diselenggarakan.

2.8 *Location Based Service*

Location Based Service adalah *service* yang berfungsi untuk mencari dengan teknologi *Global Positioning Service* (GPS) dan Google's *cell-based Location*. *Map* dan layanan berbasis lokasi menggunakan lintang dan bujur untuk menentukan lokasi geografis, namun sebagai *User* kita membutuhkan alamat atau posisi *Realtime* kita bukan nilai lintang dan bujur. *Android* menyediakan *geocoder* yang mendukung *forward* dan *reverse* geocoding. Menggunakan *geocoder*, anda dapat mengkonversi nilai lintang bujur menjadi alamat dunia nyata atau sebaliknya.

Location based service atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan. Dua unsur utama LBS adalah:

1. *Location Manager* (API *Maps*)

   Menyediakan *tools*/*resource* untuk LBS, *Application Programming Interface* (API) *Maps* menyediakan fasilitas untuk menampilkan, memanipulasi *Maps*/peta beserta feature-feature lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada pada com.google.*Android*.*Maps*.

2. *Location Providers* (API *Location*)

   Menyediakan teknologi pencarian lokasi yang digunakan oleh *Device*/perangkat. API *Location* berhubungan dengan data GPS dan data lokasi *Real-time*. API *Location* berada pada paket *Android* yaitu dalam paket *Android*.*Location*. Dengan *Location Manager*, kita dapat menentukan lokasi kita saat ini, *Track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan [12].

Dalam penetilian ini peneliti menggunakan atau memanfaatkan teknologi layanan berbasis lokasi yaitu *Location based service* (LBS). Tujuan penggunaan teknologi LBS ini yaitu agar tim-tim yang akan melakukan pertandingan dapat

mengetahui lokasi lapangangan, *Event* serta mengetahui rute ke tempat persewaan lapangan basket yang akan menjadi tempat bertandingnya tim-tim tersebut dan juga *Event* yang akan diselenggarakan.

2.9 *Geotagging*

Geotagging merupakan proses penambahan informasi geospasial pada berbagai media digital. Media yang telah mengalami proses *Geotagging* akan memiliki informasi koordinat berupa *longitude* (bujur), *latitude* (lintang). Hal tersebut memungkinkan media dapat diposisikan secara tepat pada peta [13].

Pada ponsel-ponsel berkamera yang memiliki GPS *receiver* internal umumnya memiliki fitur ini. Mekanisme GPS *PhotoTagging* adalah pada saat sebuah foto diambil menggunakan kamera (digital atau ponsel) yang memiliki fitur *Geotagging*, kamera atau ponsel tersebut mencatat lebih banyak informasi/data dibandingkan dengan sebuah foto yang 6diambil dengan kamera biasa. Informasi tersebut termasuk waktu dan data ketika sebuah foto diambil, orientasi dari kamera (*portrait* atau *landscape*), apakah menggunakan lampu flash dan detil kamera lainnya yang digunakan seperti Apertur, Exposure, dan Local Length. Semua data ini disimpan pada suatu tempat yang disebut EXIF *Headers*. EXIF (*Exchangeable Image File Format*) *headers* berisi petunjuk foto dengan data yang dapat dibaca oleh aplikasi manajemen foto atau sebuah situs foto tertentu. Selain itu, EXIF *Headers* juga menyediakan sebuah ruang untuk mengisi koordinat *Longitude*, *Latitude*.

Pada penelitian ini penulis memanfaatkan teknologi *Geotagging* yang berguna untuk ketika kapten tim basket melakukan pengambilan gambar konfirmasi bukti pembayaran persewaan, maka lokasi pada saat mengambil foto bukti pembayaran tersebut otomatis akan mengambil *lattiude* dan *longtitude* yang dimana nantinya akan di konversi oleh *geocoder* menjadi informasi alamat lokasi pengguna pada saat mengambil foto tersebut. Tujuan dari penggunaan teknologi ini melainkan sebagai fungsi data keamanan pengusaha lapangan agar dapat mengetahui jejak pengambilan foto bukti konfirmasi pembayaran.

2.10 *Geofence*

Geofence adalah sebuah konsep untuk mendeskripsikan area geografis yang kemudian dimungkinkan untuk menyediakan *context-based Action* secara proaktif [14]. Merupakan generasi selanjutnya dari *location based service*, dimana ketika sebuah perangkat *Mobile* memulai interaksi dialog dengan pengguna jika perangkat *Mobile* memasuki atau keluar dari area yang telah ditentukan [15].

Geofence merelasikan area geografis dengan objek bersamaan dengan sebuah kondisi yang ditentukan terlebih dahulu. Fungsi dari *geofence* yang dibuat dengan lokasi terkini dari perangkat *Mobile* yaitu: ketika pengguna memasuki atau meninggalkan area geografis yang telah dibuat dapat dideteksi secara otomatis, kemudian dari hasil deteksi tersebut dapat dihasilkan luaran yang diinginkan. Dimana luaran tersebut dijalankan secara otomatis ketika semua kondisi yang telah ditentukan terpenuhi.

Pada penelitian yang dilakukan, fungsi trigger *geofence* yang digunakan adalah transisi ketika memasuki area *geofence* yang telah dibuat. Hasil dari trigger *geofence* tersebut akan mengirimkan informasi ke perangkat *Android* pengguna.

2.11 JSON

JSON (*Javascript Object Notation*) adalah *form*at pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. *Form*at ini dibuat berdasarkan bagian dari Bahasa Pemrograman *Javascript*, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan *form*at 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 [16].

JSON terbuat dari dua struktur yaitu:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*object*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau associative array.

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

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena *form*at data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

Dalam penelitian ini peneliti menggunakan metode pertukaran data *form*at JSON karena strukturnya mudah dipahami bagi peneliti sehingga mempercepat dalam membangun *Web service* untuk aplikasi *Android*.

2.12 MySQL

MySQL adalah sebuah *database manajement system* (DBMS) popular yang memiliki fungsi sebagai relational *database manajement system* (RDBMS). Selain itu MySQL *software* merupakan suatu aplikasi yang sifatnya *online* serta server basis data MySQL memiliki kinerja sangat cepat, *reliable*, dan mudah untuk digunakan serta bekerja dengan arsitektur *client server* atau *embedded systems*, dikarenakan faktor *online* dan popular tersebut maka cocok untuk mendemontrasikan proses replikasi basis data. [17].

Sejarah *MySQL* yang merupakan hasil buah pikiran dari Michael ‘Monty’ Widenius, David Axmark, dan Allan Larson dimulai tahun 1995. Mereka bertiga kemudian mendirikan perusahaan bernama *MySQL* AB di Swedia. Tujuan awal ditulisnya program *MySQL* adalah untuk mengembangkan aplikasi *Web* yang akan digunakan oleh salah satu klien *MySQL* AB. Pada saat itu *MySQL* AB adalah sebuah perusahaan konsultan *database* dan pengembang *Event* (masih menggunakan nama perusahaan TcX DataKonsult AB) .

MySQL memiliki kinerja, kecepatan proses, dan ketangguhan yang tidak kalah dibanding *database-database* besar lainnya yang komersil seperti ORACLE, Sybase, Unify, dan sebagainya.

Didalam penelitian ini peneliti akan menggunakan *MySQL* karena DBMS tersebut paling sering digunakan sehingga mudah dalam melakukan perbaikan saat terjadi kesalahan.

2.13 Java

Java merupakan bahasa pemrograman yang sangat populer karena rentang aplikasi yang bisa dibuat menggunakan bahasa ini sangatlah luas, mulai dari komputer hingga *smartphone*.

Bahasa pemrograman Java dikembangkan pertama kali oleh sun *Microsystem* yang dimulai oleh James Gosling dan dirilis pada 1995. Saat ini, Sun *Microsystem* telah diakuisisi oleh Oracle *Corporation*. Java bersifat *Write Once, Run Anywhere* (program yang ditulis satu kali dan dapat berjalan pada banyak *platform*). Dengan demikian, tidak mengherankan bila aplikasi yang dibuat dengan java bisa ditemukan di lingkungan komputer dan *smartphone* tanpa perbedaan yang berarti. Sama seperti pemograman pada umumnya, Java merupakan bahasa pemgrograman yang mampu bekerja dengan sebuah *database* [18].

Dalam penelitian ini peneliti menggunakan bahasa pemograman java karena lebih mudah dalam mencari contoh dokumentasi dan merupakan bahasa yang didukung oleh IDE *Android* Studio resmi dari Google.

2.13.1 Berbagai Kelebihan Java

Java merupakan bahasa pemograman paling populer dan secara resmi dipelajari di bangku perkuliahan. Popularitas tersebut terjadi karena Java memiliki berbagai kelebihan yang dirinci dalam setiap fiturnya.

Berikut ini fitur-fitur Java :

1. Berorientasi Objek : dalam Java, semua adalah Objek.
2. Bersifat *Platform Independent* : Java di-compile dalam bit kode *platform* independen. Itu artinya, aplikasi yang ditulis dengan Java dapat dengan mudah dibuka dengan berbagai sistem komputer dan arsitektur komputer. Ini berbeda dengan aplikasi yang dibauat dengan Visual Basic. Misalnya, yang secara umum hanya bisa dibuka dengan komputer bersistem operasi MS Windows.

Oleh karena itu, tidak heran jika aplikasi yang dibuat dengan Java umumnya bisa dibuka menggunakan komputer, *smartphone*, dan perangkat IT lainnya.

3. Sederhana : Java didesain untuk dapat dengan mudah dipelajari sehingga penetrasi pemograman ini cukup tinggi dikalangan para pelajar.
4. Aman : dengan fitur keamanan Java, Anda dapat membuat sistem yang bebas virus dan *powerfull*
5. Terdistribusi : Java didesain untuk lingkungan distribusi internet.
6. Dinamis : Java lebih dinamis dari C dan C++ karena didesain untuk beradaptasi dengan lingkungan pengembangan.

2.14 Codeigniter

Codeigniter adalah sebuah framework PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi *Website* berbasis PHP dibandingkan jika *menu*lis semua kode program dari awal. Codeigniter menyediakan banyak library untuk mengerjakan tugas-tugas yang umumnya ada pada sebuah aplikasi berbasis *Web*. Selain itu, struktur dan susunan logis dari Codeigniter membuat aplikasi yang dibuat menjadi semakin teratur dan rapi. Dengan demikian developer dapat fokus pada fitur-fitur apa yang dibutuhkan oleh aplikasi dengan membuat kode program seminimal mungkin. Codeigniter pertama kali dibuat oleh Rick Ellis, CEO Ellislab, Inc, sebuah perusahaan yang memproduksi sebuah CMS (*Content Management System*) yang cukup handal, yaitu *ExpressionEngine*. Saat ini, Codeigniter dikembangkan dan dimaintain oleh *ExpressionEngine Development Team* [19].

Dalam penelitian ini untuk bahasa pemograman *backend* peneliti menggunakan Codeigniter karena dengan menggunakan framework ini memudahkan peneliti dalam mengembangkan aplikasi secara cepat tanpa harus melakukan pemograman dari nol. Dengan demikian, karena peneliti telah memiliki dasar pemograman PHP, jadi untuk development aplikasi dapat bekerja lebih cepat.

2.15 *Android* Studio

Android Studio adalah *Integrated Development Environment* (IDE) untuk pengembangan aplikasi *Android*, berdasarkan IntelliJ IDEA. Selain merupakan

editor kode IntelliJ dan alat pengembang yang berdaya guna, *Android* Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi *Android*, misalnya:

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

Setiap proyek di *Android* Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

Modul aplikasi *Android*

2. Modul perpustakaan

3. Modul Google *App Engine* [20].

Dalam penelitian ini peneliti menggunakan IDE *Android* Studio dibandingkan dengan IDE lain karena *Android* Studio telah memiliki dukungan penuh dari Google. Hal tersebut memudahkan pengembang dalam melakukan integrasi seperti penggunaan sebuah *library* yang melalui file gradle cukup dengan mendaftarkan satu baris program saja. Adapun versi *Android* studio yang digunakan peneliti yaitu versi 3.1.

2.16 *Sublime Text*

Sublime Text adalah aplikasi *cross-platform* teks dan *editor* kode, dengan menggunakan API Phyton. Fungsinya juga diperpanjang dengan plugin, Sublime

Text bukan perangkat lunak open source, atau perangkat lunak bebas, namun sebagian besar paket ekstensi memiliki lisensi perangkat lunak bebas dan dibuat dan dikelola oleh komunitas. Didalam penelitian ini peneliti menggunakan *tools* Sublime Text sebagai *editor* kode untuk membangun subsistem *Web* [19].

2.17 *Object Oriented Analysis and Design*

OOAD adalah metode analisis yang memerikasa *Requirement*s dari sudut pandang kelas kelas dan objek yang ditemui dalam ruang lingkup permasalahan yang mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem. OOAD merupakan cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat *menu*rut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas [21].

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa *Requirement* (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur *Event* yang didasarkan pada manipulasi objek-objek sistem atau subsistem.

2.18 *Unified Modeling Languange* (UML)

Unified Modeling Languange (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun [21].

1. *Use Case*

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem secara keseluran yang akan dibuat. *Diagram Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, *diagram Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Yang ditekankan pada *diagram* ini adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". *Use Case* menjelaskan secara sederhana fungsi sistem dari sudut pandang *User*. Adapun komponen-komponen dalam *Use Case Diagram* anataranya :

a. Actor

Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu. Bisa merupakan manusia, sistem, atau *Device* yang memiliki peranan dalam keberhasilan operasi dari sistem.

b. *Use Case*

Use Case merupakan gambaran umum dari fungsi atau proses utama yang menggambarkan tentang salah satu perilaku sistem. Perilaku sistem ini terdefinisi dari proses bisnis sistem yang akan dimodelkan. Tidak semua proses bisnis digambarkan secara fungsional pada *Use Case*, tetapi yang digambarkan hanya fungsionalitas utama yang berkaitan dengan sistem. *Use Case* menitik beratkan bagaimana suatu sistem dapat berinteraksi baik antar sistem maupun diluar sistem.

c. Sytem

Menyatakan batasan sistem dalam relasi dengan actor-actor yang menggunakannya (di luar sistem) dan fitur-fitur yang harus disediakan (dalam sistem). Digambarkan dengan segi empat yang membatasi semua *Use Case* dalam sistem terhadap pihak mana sistem akan berinteraksi. Sistem disertai label yang menyebutkan nama dari sistem, tapi umumnya tidak digambarkan karena tidak terlalu memberi arti tambahan pada *diagram*.

d. Association

Mengidentifikasikan interaksi antara setiap *actor* tertentu dengan setiap *Use Case* tertentu. Digambarkan sebagai garis antara *actor* terhadap *Use Case* yang bersangkutan. Asosiasi bisa berarah (garis dengan anak panah) jika komunikasi satu arah, namun umumnya terjadi kedua arah (tanpa anak panah) karena selalu diperlukan demikian.

e. *Dependency*

Dependensi <<*include*>>

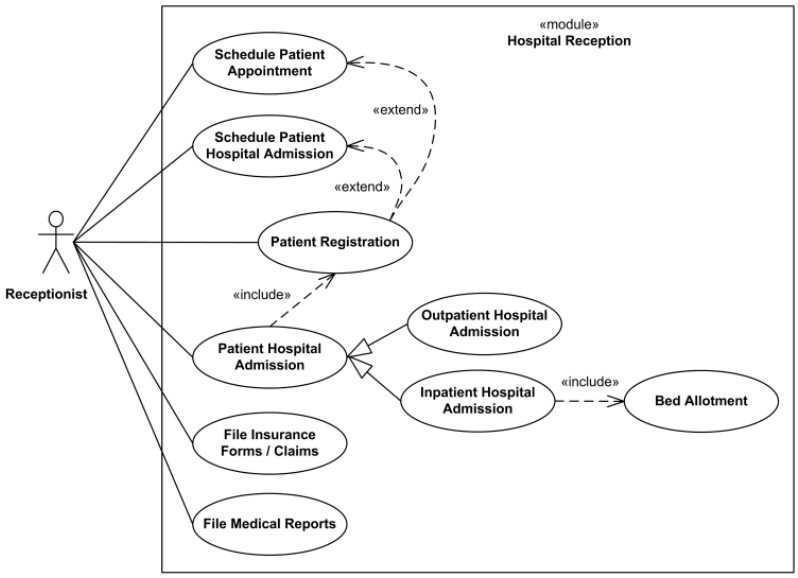
a) Mengidentifikasi hubungan antar dua *Use Case* dimana yang satu memanggil yang lain.
b) Jika pada beberapa *Use Case* terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan *Use Case* tersendiri dengan relasi dependensi setiap *Use Case* semula ke *Use Case* yang baru ini sehingga memudahkan pemeliharaan.
c) Digambarkan dengan garis putus-putus bermata panah dengan notasi <<*include*>> pada garis.
d) Arah mata panah sesuai dengan arah pemanggilan.

Dependensi <<*include*>>

e) Jika pemanggilan memerlukan adanya kondisi tertentu maka berlaku dependensi <<*include*>>.
f) Digambarkan serupa dengan dependensi <<*include*>> kecuali arah panah berlawanan.
g) Note: konsep "*include*" ini berbeda dengan "*include*" dalam Java.

7. Generalization

Mendefinisikan relasi antara dua *actor* atau dua *Use Case* yang mana salah satunya meng-*inherit* dan menambahkan atau *override* sifat dari yang lainnya. Penggambaran menggunakan garis bermata panah kosong dari yang meng-inherit mengarah ke yang di-inherit.



Gambar 2.3 Contoh *Use Case Diagram*

2. Use Case Scenario

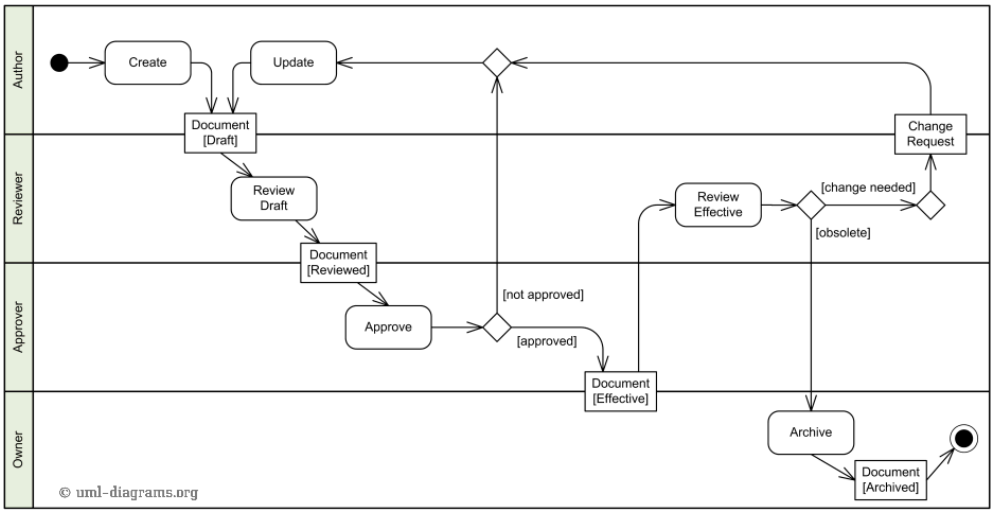
Sebuah diagram yang *menu*njukkan *Use Case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *Use Case Scenario* berbasis teks per *Use Case*nya. Berikut adalah dasar *format* penulisan *Use Case Scenario*.

Tabel 2.3 *Form*at Penulisan *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*	
Use Case Name	Berisi nama dari *Use Case* yang akan digunakan	
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

3. Activity Diagram

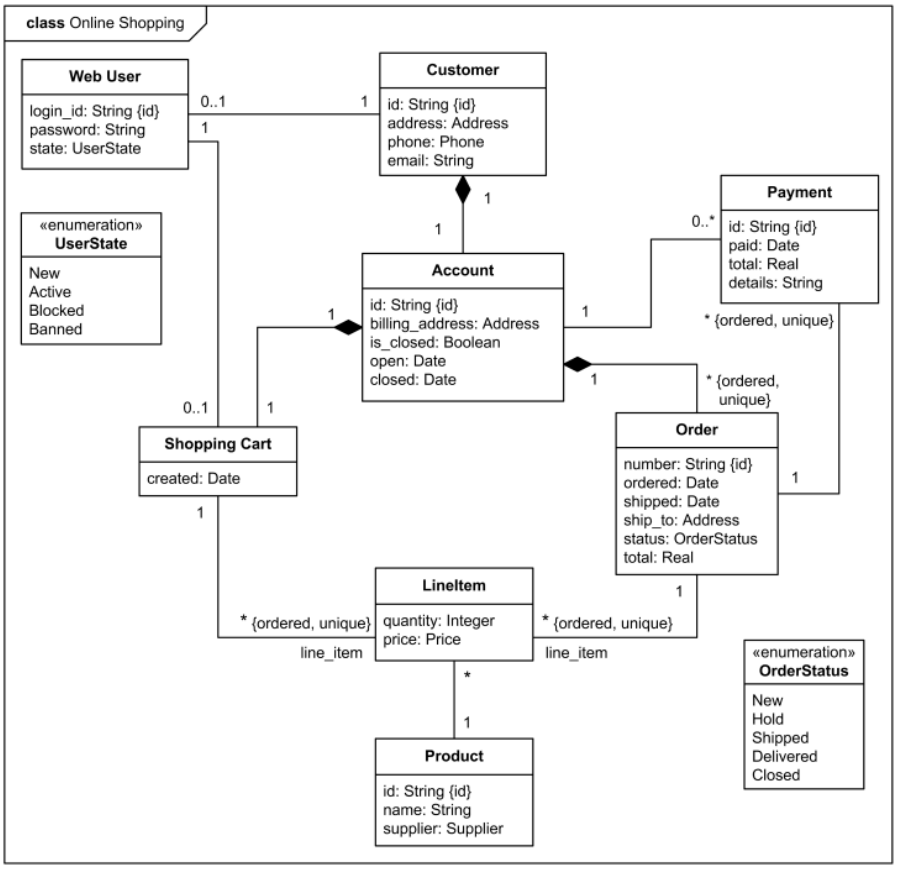
Activity Diagram adalah sebuah tahapan yang lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dimana biasanya dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. *Activity Diagram* ini sendiri memiliki struktur *diagram* yang mirip *Flowchart* atau data *flow diagram* pada perancangan terstruktur. *Activity Diagram* dibuat berdasarkan sebuah atau beberapa *Use Case* pada *Use Case Diagram*.



Gambar 2.4 Contoh *Activity Diagram*

4. Class Diagram

Class Diagram merupakan *diagram* yang selalu ada di pemodelan *sytem* berorientasi objek. *Class Diagram menu*njukkan hubungan antar *Class* dalam *sytem* yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. Kelas pada kelas *diagram* terdiri dari 3 bagian utama yaitu nama kelas, isi *property* dari kelas beserta metode yang ada pada kelas tersebut. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti *public*, *private* dan *protected*.

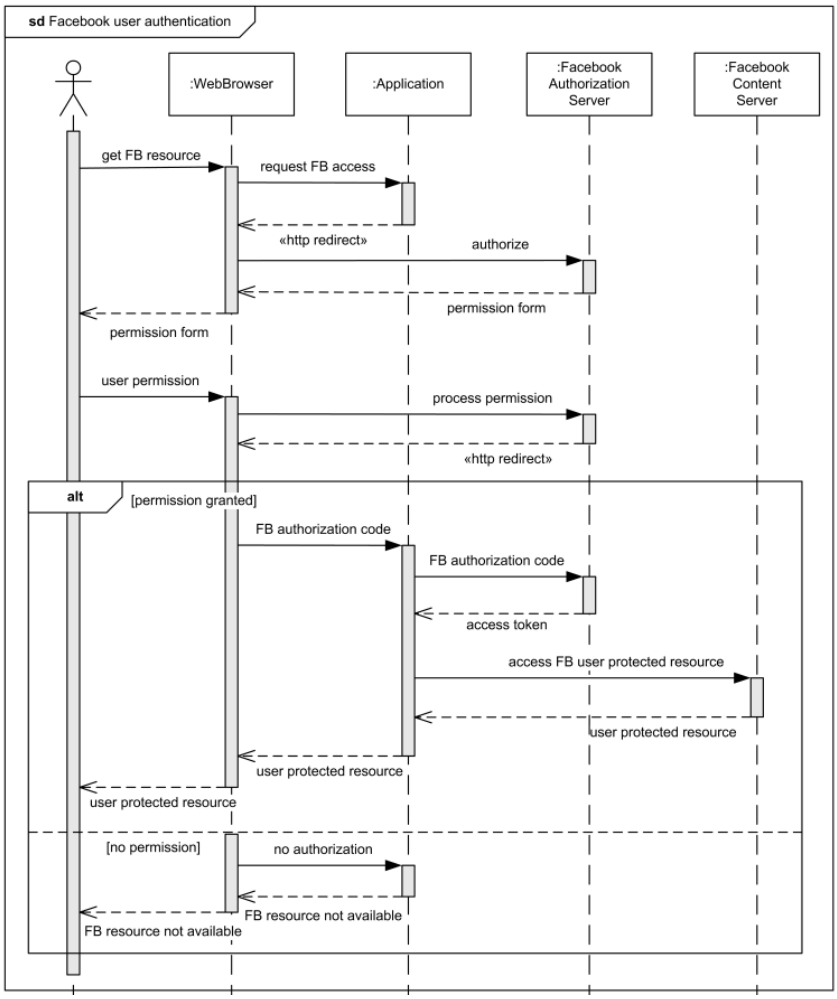


Gambar 2.5 Contoh *Class Diagram*

5. Sequence Diagram

Sequence Diagram adalah suatu *diagram* yang digunakan untuk menggambarkan dan menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *Use Case*. Adapun urutan proses yang dijelaskan yaitu interaksi yang terjadi antar *Class*, operasi yang terlibat, urutan antar operasi, dan *informs* yang diperlukan oleh masing-masing operasi. Komponen utamanya adalah objek yang digambarkan dengan kotak segi empat atau bulat, *message* yang

digambarkan dengan garis penuh, dan waktu yang ditunjukkan dengan *progress vertical*.



Gambar 2.6 Contoh *Sequence Diagram*

Dalam penelitian ini peneliti menggunakan rancangan sistem nya yaitu menggunakan UML karena aplikasi yang dibangun bersifat OOP(*Object Oriented Programming.*