BAB 2
TINJAUAN PUSTAKA

2.1 Musik

Menurut M.Soeharto dalam buku "Kamus Musik" dijelaskan bahwa musik adalah pengungkapan melalui gagasan melalui bunyi, yang unsur dasarnya berupa melodi, irama, dan harmoni dengan unsur pendukung berupa gagasan, sifat dan warna bunyi [4]. Musik merupakan seni yang timbul dari perasaan atau pikiran manusia sebagai pengungkapan ekspresi diri, yang diolah dalam suatu nada-nada atau suara-suara yang harmonis. Jika musik diartikan sebagai ungkapan sederhana dari suasana hati jiwa atau respon harafiah terhadap peristiwa dari diri pribadi komponis, diperlukan informasi ataupun referensi yang cukup agar kita dapat menarik hubungan langsung antara kehidupan dengan karyanya [4].

2.2 Genre Musik

Genre musik sulit untuk digambarkan karena tidak ada kesepakatan lengkap tentang definisnya. “Genre muncul sebagai istilah dan kata benda yang mendefinisikan kekambuhan dan kesamaan bahwa anggota dari masyarakat membuat berhubungan dengan mengidentifikasi acara musik” [5]. Genre Musik merupakan pengelompokan musik sesuai dengan kemiripannya satu sama lain. Genre Musik dapat dikelompokan sesuai dengan kriteria yang dapat didefinisikan oleh teknik musik, gaya, konteks, dan tema musik. Berdasarkan hasil dari penjabaran jenis musik, terdapat dua istilah yaitu *Sidestream Music* dan *Mainstream Music*, terlihat seberapa besar jenis musik tersebut dapat diterima dan dinikmati oleh masyarakat secara *universal* yang kemudian kaitannya menjadi sasaran industri.

2.2.1 *Sidestream Music*

Sidestream Music yang merupakan nama lain dari "musik alternatif" yang tidak dinikmati secara massal oleh masyarakat dunia, namun berkembang di daerah atau di kelompok tertentu. Pengelompokannya adalah sebagai berikut :

2.2.1.1 Musik Seni (*Art Music*)

Musik Seni merupakan pengelompokan musik ke arah Musik klasik dan musik-musik sejenis yang dinilai secara tingkat nilai sejarahnya selain dari nilai seninya. Musik jenis ini lebih digunakan untuk merujuk jenis-jenis musik yang sangat patuh kepada teori-teori musik dan biasanya sulit untuk dipahami orang awam.

2.2.1.2 Musik Tradisional (*Traditional Music*)

Musik tradisional adalah musik yang hidup di masyarakat secara turun temurun, dipertahankan bukan sebagai sarana hiburan saja, melainkan ada juga dipakai untuk pengobatan dan ada yang menjadi suatu sarana komunikasi antara manusia dengan penciptanya, hal ini adalah menurut kepercayaan masing-masing orang saja. Musik tradisional merupakan perbendaharaan seni lokal di masyarakat.

2.2.1.3 Musik Independen (*Indie Music*)

Musik Independen, yang sering disingkat "*Indie Music*" atau "*Indie*", adalah istilah yang digunakan untuk menggambarkan musik yang dibuat dengan bebas dan diluar label rekaman komersial yang utama (*Major Label*), pendekatannya adalah *Do-It-Yourself* dalam hal rekaman dan publishing. Musik-musik yang dimainkan sangat bebas, karena ada otonomi mutlak bagi para musisi Indie untuk menciptakan musik. Karya-karya dari musik indie berada di iluar *mainstream* atau berbeda dengan corak lagu yang sedang laris di pasaran. Bahkan sering kali Musik yang berasal dari Indie Label lebih cepat merespon tren baru dan lebih idealis dalam tujuannya. Musik Indie dikenal sebagai sesuatu yang berbeda karena karakternya cenderung eksperimental, amatir *cutting edge* dan kreatif [6].

2.2.2 *Mainstream Music*

Mainstream Music adalah musik yang diterima dan banyak disukai oleh masyarakat dunia. Pengelompokannya adalah sebagai berikut:

2.2.2.1 *Black Urban Music*

Musik yang lahir dari kaum *Afro-America*, seperti *Rhythm and Blues (R&B), Jazz, Gospel*, *Hip Hop*, dan *Rap* kini menjadi musik dominan yang berkembang di Amerika Serikat sebagi kiblat musik dunia.

1. *Jazz*

Jazz adalah jenis musik yang tumbuh dari penggabungan *blues, ragtime,* dan musik Eropa, terutama musik band. Beberapa subgenre jazz adalah *Dixieland, swing, bebop, hard bop, cool jazz, free jazz, jazz fusion, smooth jazz,* dan *CafJazz.*

2. *Gospel*

Sebenarnya lagu jenis Gospel ini memiliki nuansa mirip dengan *Rock n Roll* (oleh karena *Rock n Roll* sendiri sebenarnya merupakan *fusion* atau gabungan dari *Rock, Jazz,* dan *Gospel*).

3. *Blues*

Blues berasal dari masyarakat Afro-Amerika yang berkembang dari musik Afrika barat. Jenis ini kemudian mempengaruhi banyak genre musik *pop* saat ini, termasuk *ragtime, jazz, big band, rhythm and blues, rock and roll, country,* dan musik *pop*.

4. *Rhytem and Blues*

Rhythm and blues adalah nama musik tradisional masyarakat Afro-Amerika, yaitu musik *pop* kulit hitam dari tahun 1940-an sampai 1960-an yang bukan *jazz* atau *blues.*

2.2.2.2 *Funk*

Funk juga dipelopori oleh musisi-musisi Afro-Amerika, misalnya James Brown, *Parliament-Funkadelic*, dan *Sly and the Family Stone*. Musik jenis *Funk* ini biasanya memiliki nada *beat groovy*, suatu rhythm yang membuat pendengarnya berdecak mengikuti irama. Oleh karena itu, dalam banyak hal, *funk* sering disamakan dengan *groovy*.

2.2.2.3 *Rock*

Rock, dalam pengertian yang paling luas, meliputi hampir semua musik pop sejak awal 1950-an. Bentuk yang paling awal, *rock and roll*, adalah perpaduan dari berbagai genre di akhir 1940-an, dengan musisi-musisi seperti Chuck Berry, Bill Haley, Buddy Holly, dan Elvis Presley. Hal ini kemudian didengar oleh orang di seluruh dunia, dan pada pertengahan 1960-an beberapa grup musik Inggris, misalnya The Beatles, mulai meniru dan menjadi populer.

2.2.2.4 *Electronic (Techno)*

Electronic dimulai lama sebelum ditemukannya *synthesizer*, dengan tape loops dan alat musik elektronik analog di tahun 1950-an dan 1960-an. Para pelopornya adalah John Cage, Pierre Schaeffer, dan Karlheinz Stockhausen.

2.2.2.5 *Ska, Reggae, Dub*

Dari perpaduan musik *R&B* dan musik tradisional mento dari Jamaika muncul ska, dan kemudian berkembang menjadi *reggae* dan *dub*.

2.2.2.6 *Hip hop, Rap, Rapcore*

Musik *hip hop* dapat dianggap sebagai *subgenre R&B*. Dimulai di awal 1970-an dan 1980-an, musik ini mulanya berkembang di pantai timur AS, disebut *East Coast hip hop*. Pada sekitar tahun 1992, musik hip hop dari pantai barat juga mulai terkenal dengan nama *West Coast hip hop*. Jenis musik ini juga dicampur dengan *heavy metal* menghasilkan *rapcore*.

2.2.2.7 *Pop*

Musik *pop* adalah genre penting namun batas-batasnya sering kabur, karena banyak musisi *pop* dimasukkan juga ke kategori *rock, hip hop* maupun *country*.

2.2.2.8 *Folk*

Musik *Folk* (*Ballad*) bisa diartikan musik tradisional atau musik rakyat. Saat ini *Genre Folk* diberi nama *Modern Folk* sebagai bentuk kemajuan. Semua Negara di berbagai belahan dunia mempunyai musik tradisional tidak terkecuali Indonesia.

Biasanya musik ini diciptakan untuk membuang rasa jenuh dan kebosanan saat melakukan pekerjaan rutin. Iramanya disesuaikan dengan kegiatan yang sedang mereka lakukan dan disela-sela waktu istirahat diselingi dengan alat musik seadanya. Contohnya adalah macam-macam musik rakyat yang tumbuh di kalangan kulit putih Amerika Serikat yang disebut *country* dan *western*. Ragam musik lainya yang berakar pada *country* and *western* adalah *blurgrass*, *honky-tonk*, *rockabilly*, *country pop*, *country rock*, dan *outlaw country*.

2.3 Sistem Informasi

Sistem adalah kumpulan komponen dimana masing-masing komponen memiliki fungsi yang saling berinteraksi dan saling tergantung serta memiliki satu kesatuan yang utuh untuk bekerja mencapai tujuan tertentu [7]. Sedangkan informasi adalah menurut Abdul Kadir mendefinisikan informasi sebagai data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakan data tersebut [8]. Sedangkan Data Menurut Jogiyanto adalah kenyataan yang menggambarkan suatu kejadian–kejadian dan kesatuan yang nyata. Atau data adalah : representasi dunia nyata yang mewakili suatu objek seperti manusia (pegawai, mahasiswa, pelanggan), hewan, peristiwa, konsep, keadaan dll, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi atau kombinasinya.

Jika disimpulkan, definisi dari sistem informasi adalah sekumpulan komponen yang memiliki fungsi yang saling ketergantungan dalam satu kesatuan utuh dalam bentuk satu data yang telah diolah dalam sebuah bentuk yang dapat digunakan untuk manusia mencapai tujuan tertentu.

Dari pendapat lain, dapat dijelaskan bahwa sistem informasi merupakan kumpulan dari komponen-komponen peralatan *model requirements*, *function* dan *interface*. Sistem Informasi merupakan kombinasi teratur apapun dari sumber daya manusia, *hardware*, *software*, data, dan jaringan. Daur hidup suatu sistem meliputi: investigasi, analisis, desain, implementasi dan pemeliharaan [9].

2.4 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 [10]. 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.

2.4.1 Android *Life Cycle*

Aplikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar *file* musik, membunyikan *alarm*, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada, klasifikasi tersebut berdasarkan kelas-kelas dasar java yang digunakan [10]. Komponen aplikasi Android dapat dilihat pada Tabel 2.1 Komponen Aplikasi Android.

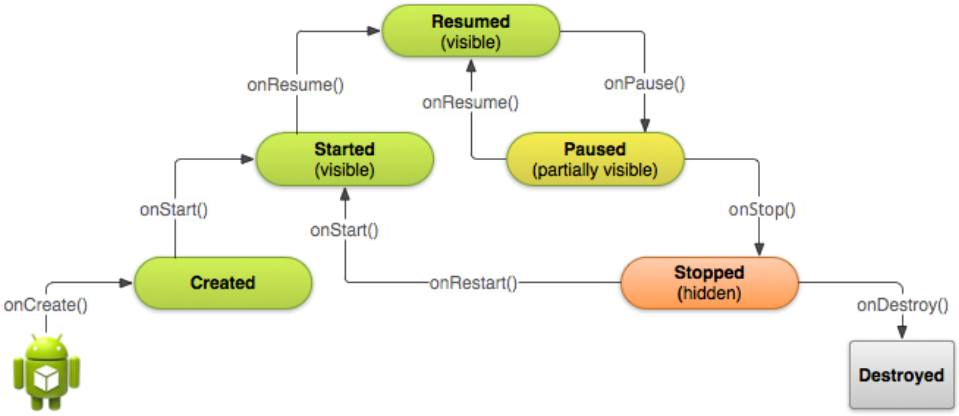
Tabel 2.1 Komponen Aplikasi Android

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

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan *specified permission* sebelum digunakan seperti komponen *Service*, *BroadcastReceiver*. *ContentProvider* [10].

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 [11].

Di dalam android terdapat *activity* dimana komponen ini memberi interaksi antara *user* dan aplikasi yang dibangun melalui *user interface*. *Activity* ini memiliki siklus hidup yang dinamakan *Android Life Cycle*. *Flowchart* siklus hidup tersebut dapat dilihat pada Gambar 2.1 *Android Life Cycle*.



Gambar 2.1 *Android Life Cycle*

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

1. Resumed

Resumed terjadi ketika aplikasi berjalan setelah state paused. State ini akan menjalankan perintah program yang ditulis pada method onResume() [11].

2. Paused

Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini [11].

3. Stopped

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service dibackground [11].

State lain seperti Created dan Started bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem OnCreate() dipanggil, dengan cepat sistem akan memanggil method OnStart(), krmudian diikuti oleh onResume() [10].

2.4.2 Fitur

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

1. Multi-proses dan *App Widgets*

Sistem operasi android tidak melarang prosesor menjalankan lebih dari satu aplikasi dalam satu waktu. Sistem operasi android dapat mengatur aplikasi dan thread yang berjalan secara *multitasking.* Keuntungan yang didapat adalah ketika aplikasi berjalan dan berinteraksi dengan pengguna di layer depan sistem operasi, proses dari aplikasi lain dapat berjalan untuk melakukan pembaruan informasi. Sebagai contoh misalnya ketika pengguna memainkan game, proses lain dapat berjalan di belakang aplikasi seperti memeriksa harga saham dan memunculkan peringatan [11].

App Widgets adalah mini aplikasi yang dapat *embedded* dalam aplikasi seperti home screen. App widgets dapat menjalankan proses *request* seperti musik streaming atau mendeteksi suhu ruangan secara *background* [11].

Multi-proses dapat memberikan manfaat berupa *user experience* yang lebih banyak, namun penggunaan fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [10].

2. *Touch Gestures* dan *Multi-touch*

Touchscreen adalah *user interface* intuitif yang digunakan banyak *smartphone* di dunia. Dengan fitur ini interaksi dapat dibuat lebih mudah karena cukup dengan menggunakan jari tangan. *Multi-touch* adalah kemampuan yang dapat melakukan tracking lebih dari satu tangan dalam satu waktu, Fitur ini sering digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut [11].

3. Hard dan *Soft Keyboard*

Salah satu fitur pada perangkat smartphone adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoperasian android. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol tersebut dalam aplikasi. Tombol non fisik adalah tombol yang dibuat oleh sistem operasi seperti *virtual keyboard*, dan tombol navigasi aplikasi [11].

2.4.3 Prinsip Desain

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

1. Multiple Assets

Android mendukung jutaan *smartphone*, tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu *Multiple Assets* sangat disarankan digunakan untuk mengatasi fragmentasi pada android. Seperti ilustrasi pada Gambar 2.2 Klasifikasi Ukuran Ikon. Android menciptakan beberapa klasifikasi ukuran icon yaitu MDPI, HDPI, XHDPI, XXHDPI dan XXXHDPI. MDPI dan HDPI dikhususkan untuk icon yang akan digunakan pada device berukuran smartphone sedangkan untuk XHDPI, XXHDPI dan XXXHDPI digunakan pada device berukuran tablet [11]. Klasifikasi ukuran ikon dapat dilihat pada Gambar 2.2 Klasifikasi Ukuran Ikon.



Gambar 2.2 Klasifikasi Ukuran Ikon

2. Touch Feedback

Touch Feedback dalam android digunakan sebagai respon setiap objek yang ditekan pengguna. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [11]. *Touch feedback* dapat dilihat pada Gambar 2.3 *Touch Feedback.*



Gambar 2.3 *Touch Feedback*

3. Pattern Gesture

Touch gesture berguna untuk meningkatkan *experience* pengguna dalam menggunakan aplikasi. Terdapat beberapa *gesture* yang didukung oleh android diataranya adalah [11]:

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 [11].

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 [11].

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 [11].

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 [11].

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 [11].

2.5 *User-Generated Content*

User-generated Content atau disingkat UGC, dikenal juga dengan istilah *consumer-generated media* (CGM) atau *user-created content* (UCC) adalah merujuk pada berbagai tipe konten materi atau media yang tersedia untuk umum

dan diproduksi oleh end-*user* atau para non-media profesional. Kemajuan teknologi *web* memungkinkan konten *web*site tidak lagi dimonopoli oleh profesional media, melainkan dapat dibuat oleh para penggunanya. UGC merupakan salah satu ciri dominan *Web* 2.0. Contoh penerapan pada UGC adalah YouTube hampir semua konten yang dimiliki situs tersebut dibuat dan diupload oleh penggunanya [12].

Konsep UGC pada dasarnya telah banyak merubah cara berinteraksi pengguna dengan internet begitu juga dalam media periklanan. Bagi media periklanan jejaring sosial dengan konsep UGC memiliki potensi besar menyediakan market yang lebih terarah dan terpusat bagi mereka [12].

2.6 *Web Service*

Web Service adalah sekumpulan *application logic* beserta objek-objek dan metode-metode yang dimilikinya yang terletak di satu *server* yang terhubung ke internet [13]. Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat platform (*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*). Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain XML, terdapat pula format pertukaran data lain yaitu JSON.

2.7 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 (*generated*) oleh komputer [14]. JSON terbuat dari dua struktur yaitu:

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

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

JSON menggunakan bentuk sebagai berikut:

1. Objek

Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

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

3. Nilai

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.

4. String

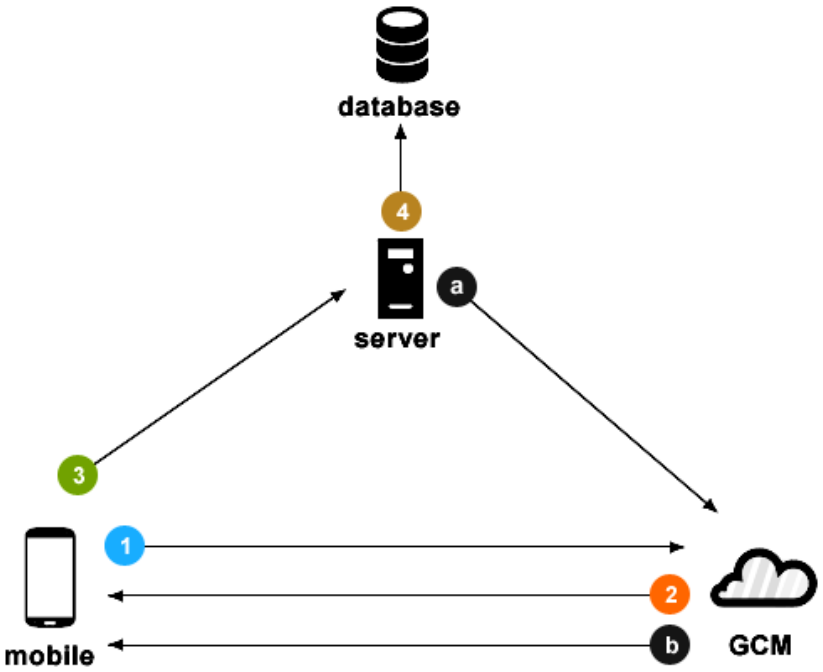
String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam *string* dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan *string* C atau Java.

5. Angka, format oktal dan heksadesimal tidak digunakan.

2.8 *Google Cloud Messaging*

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

aplikasi melalui google *play* [10]. GCM dapat dilihat pada Gambar 2.4 GCM Model GCM Model.



Gambar 2.4 GCM Model [11]

Berdasarkan ilustrasi dari Gambar 2.4 GCM Model terdapat dua *server* yang terlibat dalam proses *push messaging* , pada setiap *server* yang terlibat terdapat proses pengiriman data yang berbeda-beda diantaranya [11]:

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

2.9 Struktur Analisis dan Desain

Struktur analisis dan desain adalah sebuah metodologi yang di gunakan pada rekayasa perangkat lunak untuk mendeskripsikan sistem ke arah 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 [15].

2.9.1 Diagram Konteks

Diagram Konteks, adalah data flow diagram tingkat atas (DFD Top Level), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem) [15].

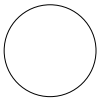
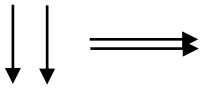
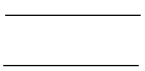
2.9.2 *Data Flow Diagram*

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 2.2 Simbol Diagram Konteks dan DFD

Tabel 2.2 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

2.9.3 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]. Berikut adalah Tabel 2.3 Contoh Spesifikasi Proses.

Tabel 2.3 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)

2.9.4 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 2.4 Contoh Kamus Data.

Tabel 2.4 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.

2.10 *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* [16]. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML.

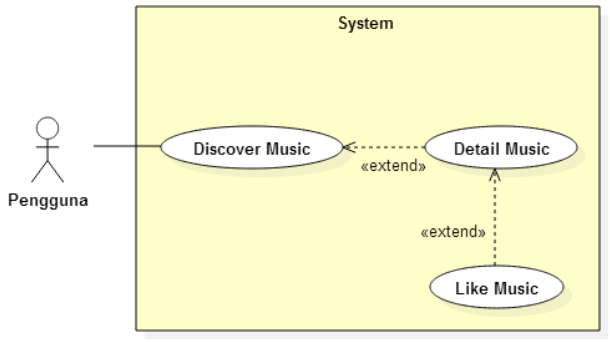
Unified Modelling Language (UML) adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menggunakan *class* dan *operation object* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek [17]. Dalam membangun block UML ada 3 hal yang harus diperhatikan, yaitu *object* (memodelkan konsep), *relationship* (mengkoneksikan *object*), dan *diagram* (*grouping* yang saling mengkoneksikan antara *object* dan *relationship.* Diagram yang umum dipakai dalam analisis dan desain adalah:

1. *Use Case Diagram*

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *Use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use

case yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah use case dapat di- include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang serupa. Sebuah *use case* juga dapat meng-*extend usecase* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

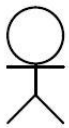
Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, *tester*, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case* [18]. Berikut adalah contoh dari *use case diagram* dapat dilihat pada Gambar 2.5 Contoh *Use Case Diagram* dan simbol yang terdapat pada *use case diagram* dapat dilihat pada Tabel 2.5. Simbol *Use Case Diagram.*

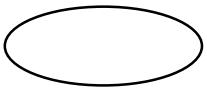
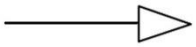


Gambar 2.5 Contoh *Use Case Diagram*

Simbol yang terdapat pada *use case diagram* dapat dilihat pada Tabel 2.5 Simbol *Usecase Diagram.*

Tabel 2.5 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.

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

2. *Use Case Scenario*

Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per *use case*nya. Berikut adalah dasar format penulisan *use case scenario* [18]. Dasar pembangunan *use case scenario* dapat dilihat pada Tabel 2.6 Dasar Pembangunan *Use Case Scenario.*

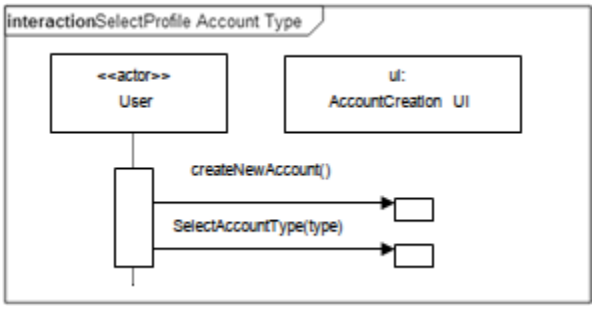
Tabel 2.6 Dasar Pembangunan *Use Case Scenario*

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 [17]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*. Comtoh dari sequence diagram dapat dilihat pada Gambar 2.6 Contoh *Sequence Diagram*.

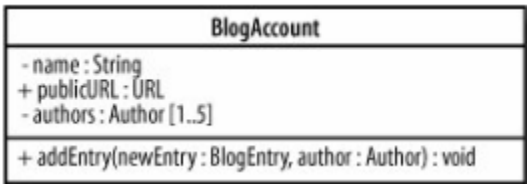


Gambar 2.6 Contoh *Sequence Diagram*

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.



Gambar 2.7 Contoh *Class Diagram*

Class memiliki tiga area pokok:

1. Nama dan stereotype
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

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

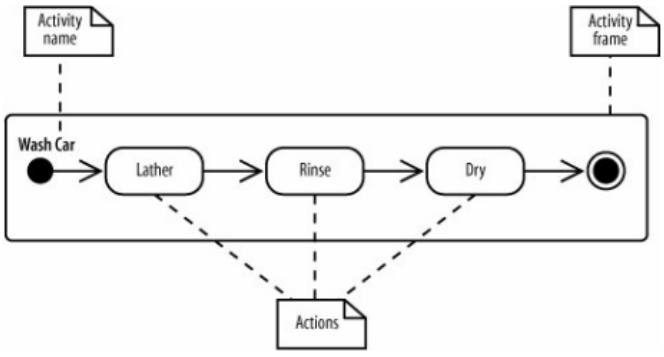
Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*.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

Class memiliki tipe-tipe *relationship*, diantaranya :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain.
2. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu class di share atau direferensikan kepada objek yang ada di class lain.
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Komposisi, yaitu jenis relasi *class diagram* yang kuat dimana jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/*solid*.
5. Depedensi, salah satu jenis relasi *class diagram* yang lemah dimana objek dalam suatu class akan bekerja sangat singkat dengan objek yang ada pada *class* lain.
5. *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 [17]. Contoh dari *activity diagram* dapat dilihat pada Gambar 2.8 Contoh *Activity Diagram*



Gambar 2.8 Contoh *Activity Diagram*

2.11 Pengujian *Black Box*

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

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

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

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

2.12 Pengujian Kuesioner

Kuesioner adalah sebuah daftar pernyataan yang harus diisi oleh orang yang akan dievaluasi (responden). Metode yang digunakan dalam kuesioner pada penelitian ini adalah skala Likert. Dalam skala likert, responden diminta untuk membaca dengan seksama setiap pernyataan yang disajikan, kemudian ia diminta untuk menilai pernyataan-pernyataan tersebut.

Derajat penilaian responden terhadap suatu pernyataan terbagi dalam 5 kategori yang tersusun secara bertingkat, mulai dari Sangat Tidak Setuju (ST), Tidak Setuju (T), Tidak Memutuskan (N), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek

yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada Tabel 2.7 Panduan Pemberian Skor .

Tabel 2.7 Panduan Pemberian Skor [19]

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

Skor yang telah dihitung pada setiap pernyataan kemudian dikalikan dengan masing-masing bobot tersebut sesuai dengan skenario kuesioner yang telah dibuat. Setelah itu total kan seluruh bobot jawaban tersebut kemudian bagi dengan total responden yang nantinya menjadi nilai rata-rata. Nilai rata-rata inilah yang diambil sebagai acuan sikap dimana jika nilai rata-rata kurang dari 3, maka dapat diartikan responden bersikap negatif dan jika nilai rata-rata lebih dari sama dengan 3, maka dapat diartikan responden bersikap positif terhadap tujuan yang ingin peneliti capai. Untuk lebih jelasnya dapat dilihat pada rumus dibawah ini.

$$\overline{X} = \frac{\sum Total}{n}$$

Dimana,

$\overline{X} \geq 3\ bersikap\ positif$

$\overline{X} < 3\ bersikap\ negatif$

Keterangan :

$\overline{X}$ = rata-rata

$\sum Total = jumlah\ seluruh\ nilai\ setelah\ dikalika\ dengan\ bobot$

n = total responden