BAB 2
TINJAUAN PUSTAKA

2.1 Komunitas

Komunitas adalah sebuah kelompok sosial dari beberapa organisme yang berbagi lingkungan, umumnya memiliki ketertarikan dan habitat yang sama. Dalam komunitas manusia, individu-individu di dalamnya dapat memiliki maksud, kepercayaan, sumber daya, preferensi, kebutuhan, risiko, kegemaran dan sejumlah kondisi lain yang serupa. Komunitas berasal dari bahasa Latin *communitas* yang berarti "kesamaan", kemudian dapat diturunkan dari *communis* yang berarti "sama, publik, dibagi oleh semua atau banyak"[7]. Menurut Crow dan Allan, Komunitas dapat terbagi menjadi 3 komponen[7]:

1. Berdasarkan Lokasi atau Tempat

Wilayah atau tempat sebuah komunitas dapat dilihat sebagai tempat dimana sekumpulan orang mempunyai sesuatu yang sama secara geografis. Dan saling mengenal satu sama lain sehingga tercipta interaksi dan memberikan konstribusi bagi lingkungannya.

2. Berdasarkan Minat

Sekelompok orang yang mendirikan suatu komunitas karena mempunyai ketertarikan dan minat yang sama, misalnya agama, pekerjaan, suku, ras, hobi maupun berdasarkan kelainan seksual. Komunitas berdasarkan minat memiliki jumlah terbesar karena melingkupi berbagai aspek, contoh komunitas pecinta animasi dapat berpartisipasi diberbagai kegiatan yang berkaitan dengan animasi, seperti menggambar, mengkoleksi action figure maupun film.

3. Berdasarkan Komuni

Komuni dapat berarti ide dasar yang dapat mendukung komunitas itu sendiri.

2.2 Aplikasi Mobile

Aplikasi *Mobile* adalah sebuah aplikasi yang memungkinkan Anda melakukan mobilitas dengan menggunakan perlengkapan seperti PDA, telepon seluler atau *Handphone*. Dengan menggunakan aplikasi *mobile*, Anda dapat dengan mudah melakukan berbagai macam aktifitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing* dan lain sebagainya[8]. Contohnya mobile programing untuk ponsel[9]:

1. Javafx mobile, digunakan di smartphone Android.
2. J2ME (Java 2 Mobile Edition) yang paling umum dan paling banyak yang support, blackberry pun masih menggunakan J2ME dengan tambahan API. J2ME adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen. Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya dari baterei, layar yang kecil dan bandwith jaringan yang rendah.
3. C++ dalam symbian framework, ini khusus untuk handphone dengan OS symbian.
4. C# .NET, untuk handphone dengan OS Windows mobile
5. Flash Lite, game flash di mobile.

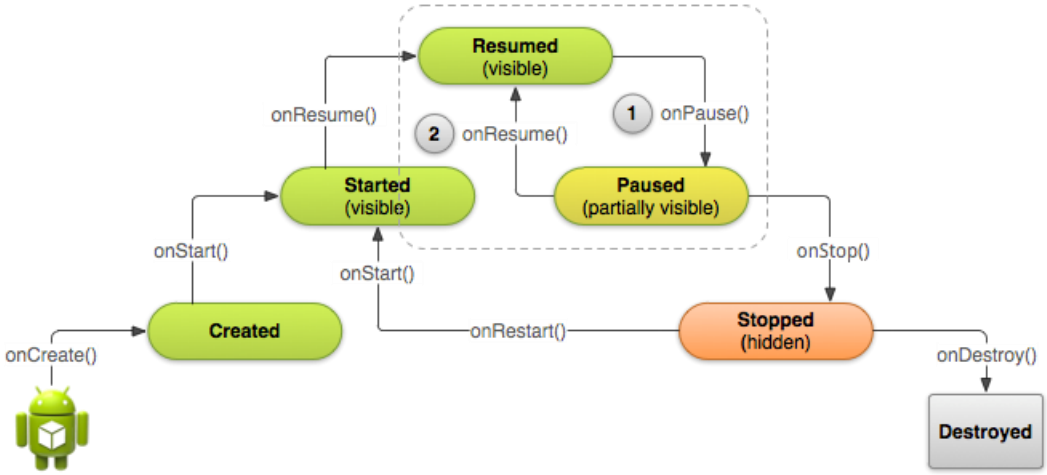
2.3 Android

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.

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[11]:

1. Resumed

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

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.

3. Stopped

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

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().

Android memiliki beberapa fitur utama yang sering digunakan dalam proses pembangunan aplikasi diantaranya adalah[11]:

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.

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*.

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

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.

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.

2.4 Website

Situs web (bahasa Inggris: *website*) adalah suatu halaman web yang saling berhubungan yang umumnya berada pada peladen yang sama berisikan kumpulan informasi yang disediakan secara perorangan, kelompok, atau organisasi. Sebuah situs web biasanya ditempatkan setidaknya pada sebuah server web yang dapat diakses melalui jaringan seperti Internet, ataupun jaringan wilayah lokal (LAN) melalui alamat Internet yang dikenali sebagai URL. Gabungan atas semua situs yang dapat diakses publik di Internet disebut pula sebagai World Wide Web atau lebih dikenal dengan singkatan WWW[12].

Meskipun setidaknya halaman beranda situs Internet umumnya dapat diakses publik secara bebas, pada prakteknya tidak semua situs memberikan kebebasan bagi publik untuk mengaksesnya, beberapa situs web mewajibkan pengunjung untuk melakukan pendaftaran sebagai anggota, atau bahkan meminta pembayaran untuk dapat menjadi aggota untuk dapat mengakses isi yang terdapat dalam situs web

tersebut, misalnya situs-situs yang menampilkan pornografi, situs-situs berita, layanan surel (*e-mail*), dan lain-lain. Pembatasan-pembatasan ini umumnya dilakukan karena alasan keamanan, menghormati privasi, atau karena tujuan komersil tertentu[12].

Sebuah halaman web merupakan berkas yang ditulis sebagai berkas teks biasa (plain text) yang diatur dan dikombinasikan sedemikian rupa dengan instruksi-instruksi berbasisHTML, atau XHTML, kadang-kadang pula disisipi dengan sekelumit bahasa skrip. Berkas tersebut kemudian diterjemahkan oleh peramban web dan ditampilkan seperti layaknya sebuah halaman pada monitor komputer[12].

Halaman-halaman web tersebut diakses oleh pengguna melalui protokol komunikasi jaringan yang disebut sebagai HTTP, sebagai tambahan untuk meningkatkan aspek keamanan dan aspek privasi yang lebih baik, situs web dapat pula mengimplementasikan mekanisme pengaksesan melalui protokol HTTPS[12].

2.5 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generated*) oleh komputer[15]. JSON terbuat dari dua struktur yaitu[13]:

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[13]:

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.

2.6 Structure Analysis and Design

Struktur analisis dan desain adalah sebuah metodologi yang di gunakan pada rekayasa perangkat lunak untuk mendeskripsikan sistem kearah fungsional. Pendekatan ini memecahkan masalah-masalah dalam aktifitas bisnis menjadi bagian-bagian kecil yang dapat disatukan kembali menjadi satu kesatuan yang utuh untuk memecahkan masalah[14].

Tools yang dapat digunakan pada pendekatan analisis pengembangan sistem secara terstruktur adalah[14]:

1. ERD (*Entity Relationship Diagram*)

Merupakan satu model jaringan yang menggunakan susunan data yang disimpan dalam sistem. ERD merupakan model jaringan data yang menekankan pada struktur dan hubungan antardata[14].

Elemen-elemen ERD[14]:

1. Entitas

Entity (entitas) adalah sesuatu apa saja yang ada di dalam sistem, nyata maupun abstrak di mana data tersimpan atau di mana terdapat data. Entitas diberi nama

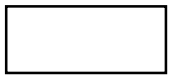
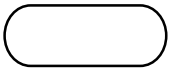
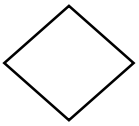
dengan kata benda dan dapat dikelompokkan dalam 4 (empat) kelas, yaitu : *role* (peran), *events* (kejadian), *locations* (lokasi), *tangible things* / *concepts* (sesuatu yang tidak nyata / konsep). Contoh: pegawai, pembelian, kampus, buku, pembayaran, mahasiswa dan dosen.

2. Relationship

Relationship adalah hubungan alamiah yang terjadi antara entitas. Contohnya: Entitas Mahasiswa dengan NIM = "14534" dan NamaMhs = "Dodo" yang mempunyai relasi dengan Entitas Kuliah dengan KodeKul = "SI-140" dan NamaMK = "Basis Data", sehingga struktur data dari Relasi ini bahwa mahasiswa tersebut mengambil mata kuliah pada suatu perguruan tinggi.

3. Atribut

Deskripsi kelompok data yang mempunyai karakteristik yang sama (data yang mendeksripsikan entity dan relationship), merupakan field yang akan disimpan. Berikut simbol-simbol ERD pada tabel 2.1.

Simbol	Nama Simbol	Fungsi
	Entitas	Menggambarkan keberadaan sebuah entitas (entitas kuat)
	Atribut	Menggambarkan atribut yang dimiliki oleh suatu entitas atau relasi
	Relasi	Menggambarkan keterhubungan antar relasi
	Garis Relasi	Menggambarkan hubungan entitas dan relasi atau entitas dengan atribut

Tabel 2.1 Simbol-simbol ERD

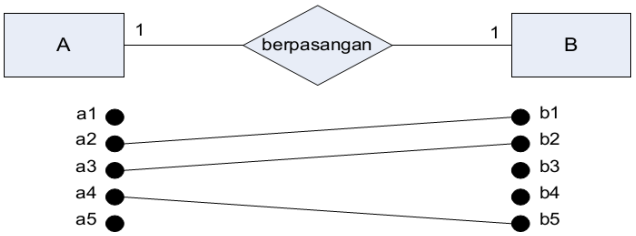
Hubungan antara entitas akan menyangkut dua komponen yang menyatakan jalinan ikatan yang terjadi, yaitu[14]:

1. Derajat (kardinalitas).

Kardinalitas menyatakan jumlah anggota entitas yang terlibat di dalam relasi yang terjadi. Dalam hal ini relasi yang terjadi akan membentuk relasi hubungan (relationship instances). Derajat kardinalitas dibagi menjadi 3, yaitu:

1. Derajat kardinalitas 1 : 1

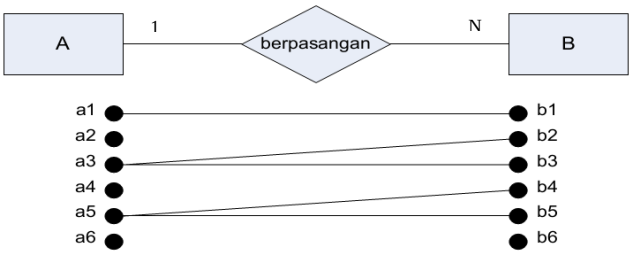
Derajat hubungan 1:1 terjadi bila setiap anggota entitas A hanya boleh berpasangan dengan satu anggota dari entitas B, dan sebaliknya tiap anggota entitas B hanya boleh berpasangan dengan satu anggota dari entitas A. Contoh dari kardinalitas 1:1 dapat dilihat pada Gambar 2.2 Derajat kardinalitas 1:1.



Gambar 2.2 Derajat kardinalitas 1:1

2. Derajat kardinalitas 1:N

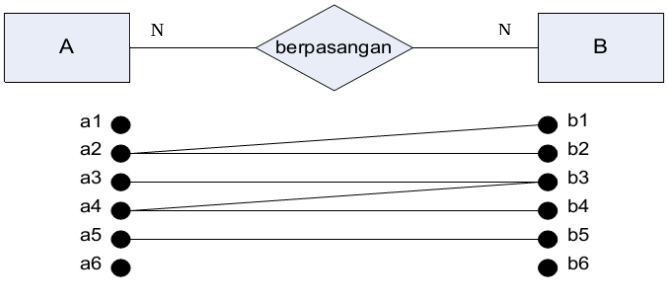
Derajat hubungan ini terjadi bila tiap anggota entitas A boleh berpasangan dengan lebih dari satu anggota entitas B. Sebaliknya tiap anggota entitas B hanya boleh berpasangan dengan satu anggota entitas A. Contoh dari kardinalitas 1:N dapat dilihat pada Gambar 2.3 Derajat kardinalitas 1:N.



Gambar 2.3 Derajat kardinalitas 1:N

3. Derajat kardinalitas N:N

Derajat hubungan antar entitas N:N terjadi bila tiap anggota entitas A dapat berpasangan dengan lebih dari satu anggota entitas B. Sebaliknya setiap anggota entitas B juga dapat berpasangan dengan lebih dari satu anggota entitas A. Contoh dari kardinalitas N:N dapat dilihat pada Gambar 2.4 Derajat kardinalitas N:N.



Gambar 2.4 Derajat kardinalitas N:N

2. Partisipasi hubungan.

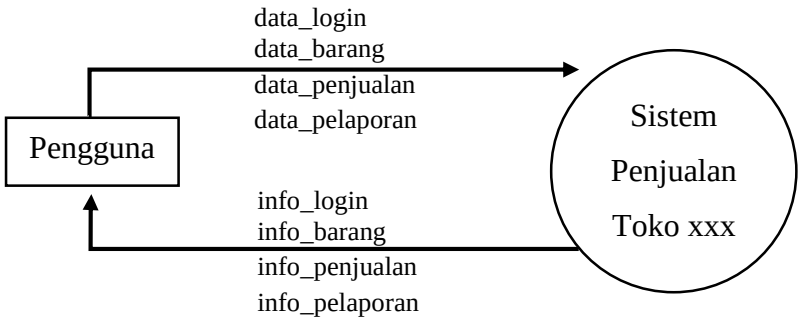
Partisipasi hubungan (*participation*) menyatakan sifat keterlibatan tiap anggota entitas dalam relasi terjadinya hubungan. Partisipasi atau keterlibatan tiap anggota entitas dalam membentuk instance hubungan dapat bersifat wajib (mandatory) dan tidak wajib (non mandatory). Dalam pemodelan data, interpretasi jenis partisipasi hubungan dituliskan dalam aturan data. Contoh misal hubungan antara entitas dosen dan mata kuliah. Berikut keterhubungan kardinalitas dan partisipasi pada tabel 2.2.

Tabel 2.2 Kardinalitas dan partisipasi

Kardinalitas	Partisipasi
1-1 (one to one)	0..1 (optional one)
1-N (one to many)	0..N atau 1..N (optinal many)
N-1 (many to one)	1 (mandatory one)
N-N (many to many)	N (mandatory many)

2. Diagram Konteks

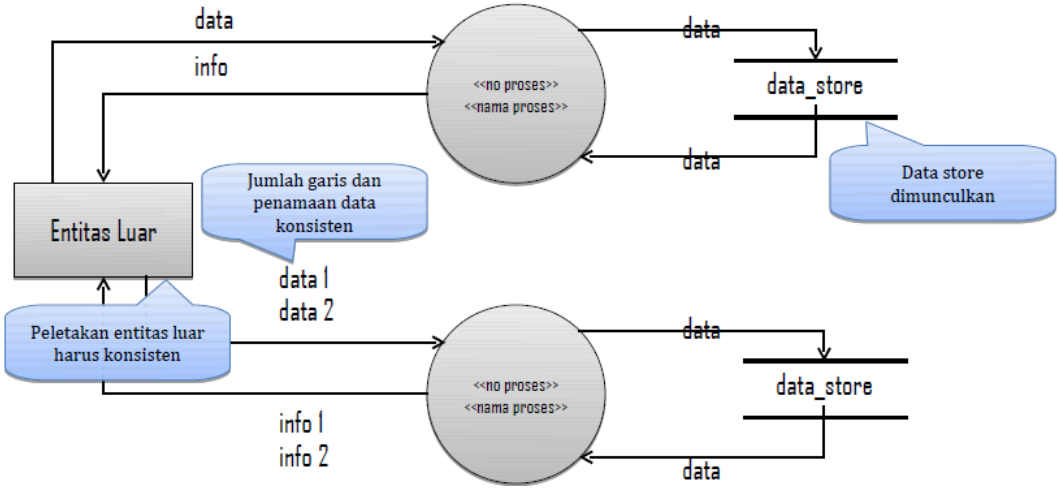
Diagram konteks adalah diagram yang menggambarkan sebuah sistem secara menyeluruh yang selanjutnya akan lebih didetailkan oleh DFD. Konteks diagram juga biasa disebut sebagai DFD level 0[14]. Contoh dari diagram konteks dapat dilihat pada gambar 2.5 Contoh diagram konteks.



Gambar 2.5 Contoh diagram komteks

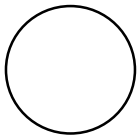
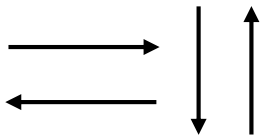
3. Data Flow Diagram (DFD)

Data Flow Diagram merupakan serangkaian diagram yang menggambarkan kegiatan-kegiatan yang ada dalam satu sistem. Teknik pembangunan DFD dimulai dengan menggambarkan sistem secara global dan dilanjutkan dengan analisis masing-masing bagian. Pada awalnya digambarkan konteks diagram yang menggambarkan sebuah sistem secara menyeluruh yang akan diinvestigasi. Konteks diagram tersebut dapat dikatakan sebagai DFD level 0. Analisis sistem yang lebih detail selanjutnya dapat dilakukan dengan menggambarkan DFD level 1,2 dan seterusnya[16]. Contoh penggambaran pembuatan DFD dapat dilihat pada gambar 2.6 Contoh DFD. Simbol-simbol yang digunakan di diagram konteks dan DFD dapat dilihat pada tabel 2.3 Simbol diagram konteks dan DFD.



Gambar 2.6 Contoh DFD

Tabel 2.3 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 pada suatu sistem
	Aliran data/informasi	Menggambarkan aliran data antar proses, data store, dan entitas luar
	Data Store	Menggambarkan kumpulan dokumen-dokumen sejenis yang disimpan

4. Spesifikasi Proses

Spesifikasi Proses merupakan tabel yang berisi keterangan deskripsi dari semua proses yang terdapat di DFD. Logika proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau *pseudo code* (tidak boleh campuran)[14]. Contoh dari spesifikasi proses dapat dilihat pada Tabel 2.4 Contoh Spesifikasi Proses.

Tabel 2.4 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)

5. Kamus Data (*Data Dictionary*)

Kamus Data merupakan fakta tentang data dan kebutuhan-kebutuhan informasi dari sistem informasi. Dengan menggunakan *data dictionary*, analis sistem dapat mendefinisikan data yang mengalir dalam sistem dengan lengkap[14]. Contoh dari kamus data dapat dilihat pada Tabel 2.5 Contoh Kamus Data.

Tabel 2.5 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.

2.7 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[15]. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML.

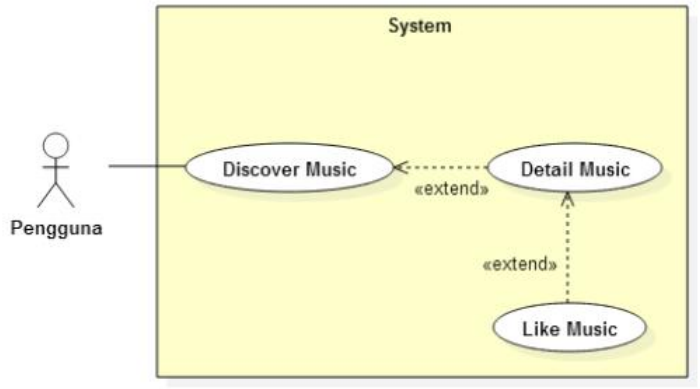
Unified Modelling Language (UML) adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menggunakan *class* dan *operation object* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek[16]. 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.

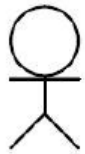
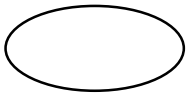
Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, tester, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case*[17]. Berikut adalah contoh dari *use case diagram* dapat dilihat pada Gambar 2.7 Contoh *Use Case Diagram.*



Gambar 2.7 Contoh Use Case Diagram

Simbol yang terdapat pada *use case diagram* dapat dilihat pada Tabel 2.6 Simbol *Usecase Diagram.*

Tabel 2.6 Simbol Usecase Diagram.

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

2. *Use Case Scenario*

Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per u*se case*nya. Berikut adalah dasar format penulisan *use case scenario*[17]. Dasar pembangunan *use case scenario* dapat dilihat pada Tabel 2.7.

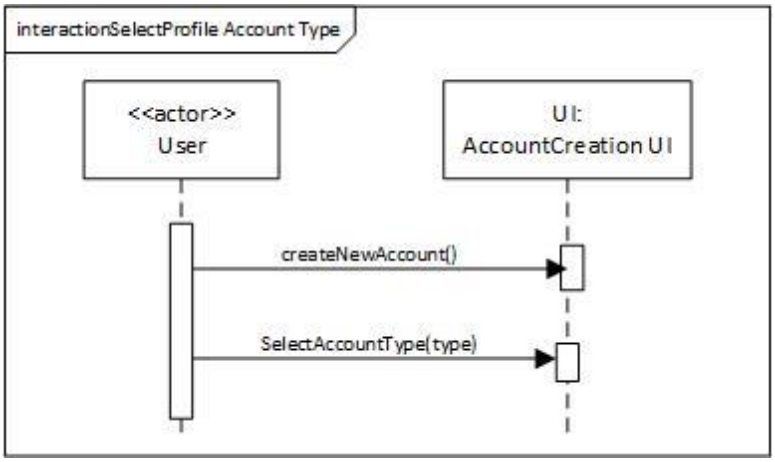
Tabel 2.7 Dasar Pembangunan Use case scenario

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*	
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
	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.

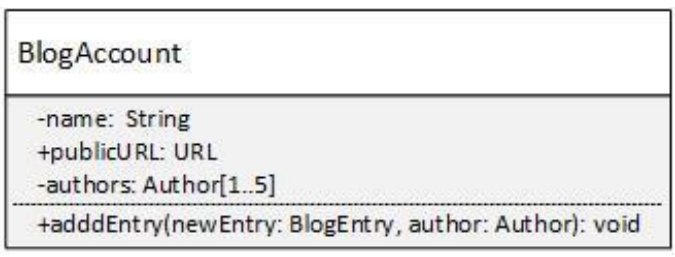
Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message [18]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*. Comtoh dari sequence diagram dapat dilihat pada Gambar 2.8 Contoh *Sequence Diagram*.



Gambar 2.8 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 2.9 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:

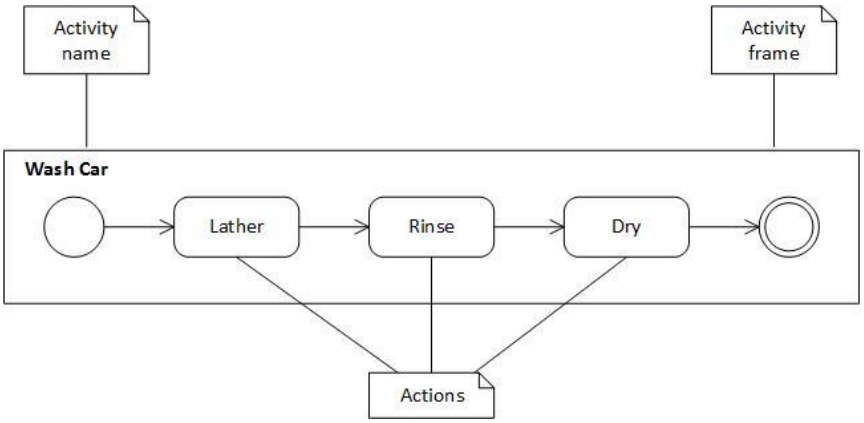
1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

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[18]. Contoh dari *activity diagram* dapat dilihat pada Gambar 2.10 Contoh *Activity Diagram.*



Gambar 2.10 Contoh Activity Diagram

2.8 Citra

Secara harfiah, citra (*image)* adalah gambar pada bidang dwimatra (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh oleh alat-alat optik, misalnya mata pada manusia, kamera,

pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam [19].

Citra sebagai keluaran dari suatu sistem perekaman data dapat bersifat [19]:

1. Optik berupa foto.
2. Analog berupa sinyal *video* seperti gambar pada monitor televisi.

2.9 Kompresi

Kompresi adalah proses pengubahan sekumpulan data menjadi bentuk kode dengan tujuan untuk menghemat kebutuhan tempat penyimpanan data dan waktu untuk transmisi data[19]. Jika data tersebut hendak dipergunakan kembali, maka harus dilakukan dekompresi, yaitu pengubahan kode-kode menjadi data awal.

Saat ini, telah banyak tersedia algoritma untuk melakukan kompresi, antara lain: Huffman, LIFO, LZHUF, LZ77 dan variannya (LZ78, LZW, GZIP), Dynamic Markov Compression (DMC), BlockSorting Lossless, Run-Length, Shannon-Fano, Arithmetic, PPM (Prediction by Partial Matching), Burrows-Wheeler Block Sorting, dan Half Byte. Namun, ada beberapa faktor yang dijadikan pertimbangan dalam memilih algoritma yang akan digunakan dalam kompresi data, yaitu :

1. Sumber daya yang dibutuhkan (Memori, kecepatan PC).
2. Kecepatan kompresi.
3. Ukuran hasil kompresi.
4. Besarnya redudansi.
5. Ketepatan hasil dekompresi.
6. Kompleksitas algoritma.

Berdasarkan output hasil kompresi, metode kompresi dapat dikelompokkan menjadi dua, yaitu :

1. Kompresi Loseless

Hasil dekompresi dari data hasil kompresi akan tepat sama persis dengan data sebelum kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip. Kelemahan dari metode ini adalah ratio kompresi yang rendah.

Teknik ini digunakan jika dibutuhkan data dimana setelah dikompresi harus dapat didekompresi lagi dan menghasilkan data yang tepat sama dengan data asli.

Contoh pada data teks, data program/biner, dan beberapa image random seperti GIF dan PNG.

2. Kompresi Lossy

Hasil dekompresi dari data hasil kompresi tidak tepat sama persis, tetapi persepsi terhadap semantik data tetap sama. Contoh: Mp3, streaming media, JPEG, MPEG, dan WMA. Kelebihan dari metode ini adalah ukuran file lebih kecil dibanding loseless namun masih tetap memenuhi syarat untuk digunakan. Biasanya teknik ini membuang bagian-bagian data yang sebenarnya tidak begitu berguna, tidak begitu dirasakan, tidak begitu dilihat sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.

Kompresi Citra

Semakin besar ukuran sebuah citra, semakin besar memori yang dibutuhkan. Namun, kebanyakan citra mengandung duplikasi data, yaitu :

1. Suatu piksel memiliki intensitas yang sama dengan piksel tetangganya, sehingga penyimpanan setiap piksel memboroskan memori
2. Mengandung region yang sama, sehingga bagian yang sama ini tidak perlu dikodekan berulang kali Kompresi citra bertujuan untuk meminimalkan kebutuhan memori untuk merepresentasikan citra digital dengan mengurangi duplikasi data di dalam citra sehingga memori yang dibutuhkan menjadi lebih sedikit. Hal ini nantinya akan berpengaruh pada waktu pengiriman citra yang menjadi lebih singkat.

2.10 Pengujian Black Box

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

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

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

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

2.11 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), Ragu-ragu (N), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada Tabel 2.8.

Tabel 2.8 Panduan pemberian skor[20]

Jenis Pernyataan	Bobot pendapat				
	SS	S	R	TS	STS
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.

$$\bar{X} = \frac{\sum Total}{\text{n}} \text{.....I}$$

Dimana,

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

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

Keterangan :

$\bar{X}$ = rata-rata

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

n = total responden