BAB II
TINJAUAN PUSTAKA

2.1 Futsal

Futsal adalah permainan bola yang dimainkan oleh dua tim, yang masing-masing beranggotakan lima orang. Tujuannya adalah memasukkan bola ke gawang lawan, dengan memanipulasi bola dengan kaki. Selain lima pemain utama, setiap regu juga diizinkan memiliki pemain cadangan. Tidak seperti permainan/olahraga dalam ruangan lainna, lapangan futsal hanya dibatasi garis, bukan net atau papan [8]. Adapun syarat utama permainan futsal terbagi menjadi 4 [3], yaitu:

1. Lapangan

Lapangan garuslah dimainkan di lapangan yang rata, mulus, dan tidak kasar atau tidak bergelombang. Sebaiknya terbuat dari kayu atau bahan buatan.

Ukuran

Pertandingan Internasional:

- Panjang
  Minimun 38m
  Maksimal 42m
- Lebar
  Minimum 20m
  Maksimal 25m

Pertandingan bukan Internasional:

- Panjang
  Minimum 25m
  Maksimal 42m
- Lebar
  Minimum16m

Maksimal 25m

2. Bola

- Berbentuk bundar
- Lingkaran bola tidak lebih dari 64 cm dan tidak kurang dari 62 cm
- Berat bola tidak lebih dari 440 gram dan tidak kurang dari 400 gram
- Memiliki tekanan sama dengan 0,6-0,9 atmosfir (600-900/cm2) pada permukaan laut

3. Pemain

Suatu pertandingan dimainkan oleh dua tim, setiap tim masing-masing tidak lebih dari lima pemain, salah satunya adalah penjaga gawang. Suatu pertandingan tidak boleh dimulai apabila pemain dari salah satu tim kurang dari tiga pemain.

4. Lamanya permainan/pertandingan

Pertandingan berlangsung dua paruh permainan yang waktunya sama masing-masing 20 menit, kecuali ada kesepakatan antara kedua tim.

2.2 Aplikasi Mobile

Aplikasi Mobile adalah adalah proses pengembangan aplikasi untuk perangkat genggam seperti PDA, asisten digital perusahaan atau telepon genggam. Aplikasi ini sudah ada pada telepon selama manufaktur, atau didownload oleh pelanggan dari toko aplikasi dan dari distribusi perangkat lunak mobile platform yang lain [10]. Contohnya mobile programing untuk ponsel:

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

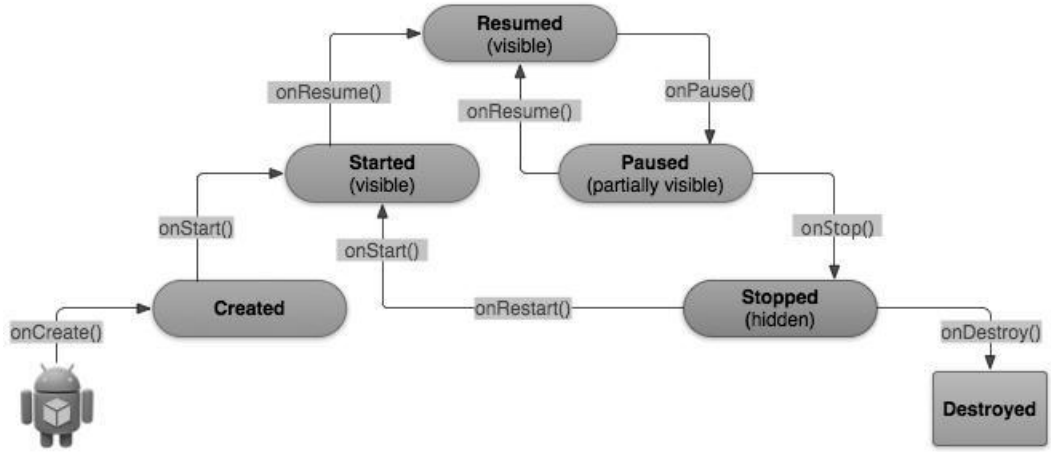
Android Merupakan sebuah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Services (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD) [11].

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.

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 2.2 *Android Life Cycle*.



Gambar 2.1 Android Life Cycle

Kondisi – kondisi pada Gambar 2.1 *Android Lifecycle* diatas bisa bekerja jika menggunakan method-method yang disediakan oleh Android, seperti pada *Tabel 2.1* [12] :

Tabel 2.1 Metode Android Lifecircle

METODE	KETERANGAN	BISA DIMATIKAN SETELAHNYA?	METHOD BERIKUTNYA
onCreate()	Dipanggil saat aktivitas pertama kali dibuat. Di sinilah Kita harus melakukan semua	Tidak	onStart()

	persiapan pada tampilan utama. Yang selanjutnya akan menjalankan method 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 onPause().	Tidak	onPause()
onPause()	Dipanggil bila sistem akan memulai proses untuk	Ya	onResume()

	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.		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

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.

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.

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.

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.

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 [13].

2.5 JAVA

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program sebagaimana Anda membuatnya dengan bahasa seperti Pascal atau C++. Yang lebih menarik, Java juga mendukung sumber daya internet yang saat ini populer, yaitu *World Wide Web* atau yang sering disebut web. Java juga mendukung aplikasi klient/server, baik dalam jaringan local (LAN) maupun jaringan berskala luas (WAN)[7].

Java dikembangkan oleh Sun Microsystems pada Agustus 1991, dengan nama semula Oak. Konon Oak adalah pohon semacam Jati yang terlihat dari jendela tempat pembuatnya, James Gosling, bekerja. Ada yang mengatakan bahwa Oak adalah singkatan dari "Object Application Kernel", tetapi ada yang menyatakan hal itu muncul setelah nama Oak diberikan. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java.

Dalam sejumlah literatur disebutkan bahwa Java merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman, yaitu C, C++, Object-C, SmallTalk, dan Common LISP. Selain itu Java juga dilengkapi dengan unsur keamanan. Yang tak kalah penting adalah bahwa Java menambahkan paradigma pemrograman yang sederhana. Jika telah mengenal C atau C++, yang mengandalkan pointer, Java justru meninggalkannya sehingga anda akan memperoleh kemudahan saat menggunakannya.

Program java bersifat tidak bergantung pada *platform*, artinya, Java dapat dijalankan pada sembarang komputer dan bahkan pada sembarang sistem operasi. Ketidakbergantungan pada platform sering dinyatakan dengan istilah portabilitas. Yang menarik, tingkat portabilitas java tidak hanya sebatas pada program sumber (*source code*), melainkan juga pada tingkat kode biner yang disebut *bytecode*. Dengan demikian bila telah mengkompilasi program java pada komputer berbasis operasi Windows, dapat juga menjalankan hasil kompilasi pada Macintosh secara langsung, tanpa perlu mengkompilasi ulang.

2.6 MySQL

Menurut Abdul Kadir mendefinisikan bahwa : MySQL merupakan *software* yang tergolong database *server* dan bersifat *Open Source*. *Open Source* menyatakan bahwa software ini di lengkapi oleh *source code* (kode yang di pakai untuk membuat *MySQL*), selain tentu saja bentuk *executable*-nya atau kode dapat di jalankan secara langsung di dalam sistem operasi, dan bias diperoleh dengan cara mengunduh di internet secara gratis. Hal lainnya adalah *MySQL* juga bersifat multiplatform. *MySQL* dapat dijalankan pada berbagai sistem operasi.

2.7 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[14]. JSON terbuat dari dua struktur yaitu[14]:

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[14]:

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 Object Oriented Analysis and Design

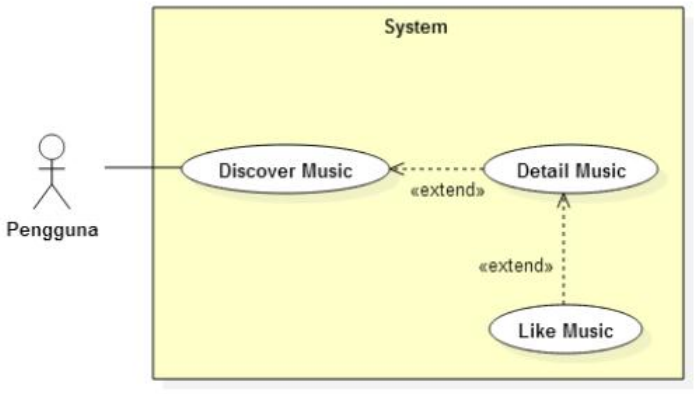
Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan satu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan user. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML [1].

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[1]. 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.

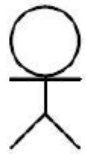
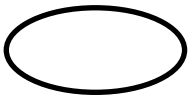
Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, tester, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case*[1]. Berikut adalah contoh dari *use case diagram* dapat dilihat pada Gambar 2.2 Contoh *Use Case Diagram.*



Gambar 2.2 Contoh Use Case Diagram

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

Tabel 2.2 Simbol Usecase Diagram

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

2. *Use Case Scenario*

Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per u*se case*nya. Berikut adalah dasar format penulisan *use case scenario*[1]. Dasar pembangunan *use case scenario* dapat dilihat pada Tabel 2.3.

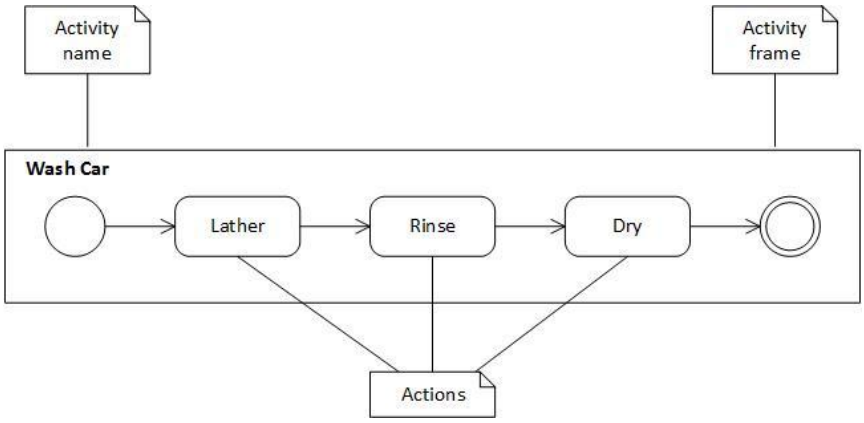
Tabel 2.3 Dasar Pembangunan Use Case Scenario

Use Case Name	Berisi nama dari *Use case* yang akan digunakan	
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari *Use case*	
Description	Menjelaskan gambaran dari *Use case*	
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	

a. Activity Diagram

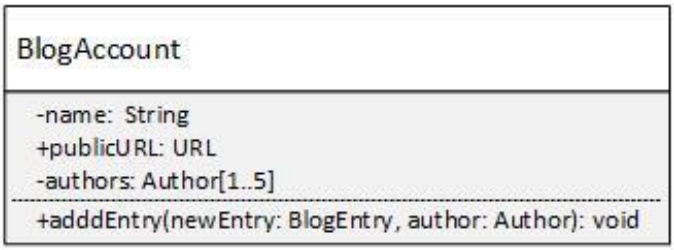
Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Sebuah 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[1]. Contoh dari *activity diagram* dapat dilihat pada Gambar 2.6 Contoh *Activity Diagram.*



Gambar 2.3 Contoh Activity Diagram

b. *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 2.4 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.

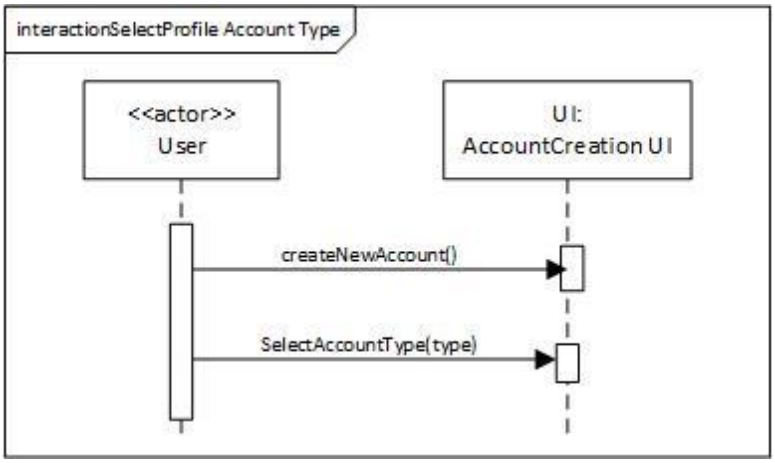
Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*.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

c. *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 2.5 Contoh *Sequence Diagram*.



Gambar 2.5 Contoh *Sequence Diagram*

2.9 Klasemen

Definisi kata klasemen kla.se.menNomina (kata benda) penentuan ke dalam kelas (golongan, tingkatan) [12]. Dan di dunia futsal klasemen berarti penentuan

tingkatan sebuah tim. Tingkatan yang ada di klasemen futsal terbagi menjadi 3 bagian, yaitu tingkatan atas, tengah, dan bawah.

Adapun berikut ini akan dijelaskan metode/cara perhitungan klasemen futsal menurut standar internasional [3].

1. Point terbanyak yang ada di klasemen
2. Lalu dilihat dari selisih (Goals Difference) antara memasukkan gol (Goals Forced) dikurangi kemasukkan gol (Goals Allowed) di klasemen
3. Lalu dilihat dari jumlah memasukkan gol (Goals Forced) di klasemen
4. Jika ada yang sama, dilihat point yang lebih banyak antara kedua tim
5. Jika point sama, dilihat selisih gol antara kedua tim
6. Jika selisih gol sama, dilihat dari jumlah memasukkan gol antara kedua tim

Kesimpulannya:

Tabel 2.4 Pembagian Poin

W	3
D	1
L	0

Keterangan:

P = Play

W = Win

D = Draw

L = Lose

GF = Goal Forced

GA = Goal Allowed

GD = Goal Diference

PTS = Total point

2.10 Broadcast

Broadcast adalah sebuah metode dalam pengiriman data dimana data akan dikirim ke banyak titik sekaligus tanpa melakukan pengecekan apakah alamat yang dituju siap untuk menerima data atau tidak dan juga pengiriman paket juga tidak memperdulikan apakah data tersebut sampai pada alamat yang dituju atau tidak[11].

2.11 Random Generate Number

Random Number Generator adalah alat atau algoritma yang menghasilkan urutan angka yang secara statistik independen dan tidak dapat ditebak.

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), 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 2.5.

Tabel 2.5 Panduan pemberian skor

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.

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