BAB 2
TINJAUAN PUSTAKA

2.1 *Workout*

Workout merupakan aktivitas fisik yang bertujuan untuk menjaga kebugaran tubuh dan membuat tubuh agar tetap sehat. Dengan tubuh yang bugar, akan sangat bermanfaat untuk mengurasi risiko berbagai penyakit seperti kelebihan berat badan, mencegah penyakit jantung, menurunkan tekanan darah tinggi, dan mengurangi depresi. Aktivitas ini bisa dilakukan baik di tempat olahraga maupun di rumah karena banyak variasi gerakan yang tidak perlu menggunakan alat bantu dan hanya mengandalkan beban tubuh saja. Beberapa jenis *workout* yang familier di masyarakat adalah *push-up, squat, sit-up, plank,* dan lain-lain. Bahkan yoga, kardio, dan senam aerobik termasuk ke dalam aktivitas *workout*[1].

Terdapat sejumlah beragam variasi dalam menjalankan latihan di rumah, seperti melakukan *push up, sit up, squat, burpees*, dan berbagai gerakan lainnya. Selain itu, kita dapat menggunakan peralatan seadanya seperti sapu lantai, ember, dan bahkan kursi dapat dimanfaatkan untuk melaksanakan latihan *tricep deep*. Pada dasarnya, hal yang sangat penting bagi individu yang menjalankan latihan di rumah adalah untuk selalu menjaga keamanan dan menghindari risiko cedera. Oleh karena itu, disarankan agar selalu melakukan pemanasan sebelum memulai latihan dan melakukan pendinginan setelah latihan selesai, karena tindakan ini akan membantu mengurangi kemungkinan terjadinya cedera[2].

2.2 Aplikasi

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

komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. Aplikasi adalah sebuah perangkat lunak yang menggabungkan penggunaan perangkat keras yang dirancang untuk menjalankan perintah dari penggunanya untuk mengatur dan mengolah data[7].

Aplikasi penggunaan dalam suatu komputer, instruksi(instruction) atau pernyataan(statement) yang disusun sedemikian sehingga komputer dapat memproses input menjadi output.Aplikasi banyak membantu manusia di berbagai bidang seperti pendidikan, bisnis, hingga militer[8].

Aplikasi *mobile* adalah aplikasi yang dibuat untuk perangkat-perangkat bergerak (*mobile)* seperti *Tablet, Smartphone, SmartWatch*, dan sebagainya. Aplikasi dibuat menggunakan bahasa pemrograman tertentu untuk membuatnya.

2.3 *Application Programming Interface*

Application Programming Interface (API) adalah sebuah antarmuka yang dibuat oleh pengembang yang bertujuan untuk membuat fungsi-fungsi dalam sistem dapat diakses secara terprogram. Penggunaan API di dalam sebuah sistem memungkinkan untuk mempermudah sebuah sistem berkolaborasi dengan sistem yang lain. Contohnya adalah penggunaan banyak *platform* seperti android, iOS, dan Website yang dapat dengan mudah bertukar data melalui API[9].

2.4 Android

2.4.1 Definisi Android

Gambar 2.1 Android

Android adalah sistem operasi berbasis Linux yang dikembangkan oleh Google. Sistem operasi ini dirancang khusus untuk perangkat mobile, seperti smartphone, tablet, dan perangkat wearable. Android menyediakan berbagai fitur dan platform yang memungkinkan pengembang untuk membuat aplikasi mobile

yang inovatif. Android merupakan sebuah *mobile platform* generasi baru yang memungkinkan pengembang untuk melakukan pengembangan sesuai dengan keinginannya[10].

2.4.2 Sejarah Android

Ketika Android pertama kali dirilis pada tanggal 5 November 2007, Android bersama Open Handset Alliance dengan tegas menyuarakan dukungannya terhadap pengembangan *open source* di ranah perangkat *mobile*. Di sisi lain, Google mengambil langkah signifikan dengan menerbitkan kode-kode sumber Android di bawah lisensi Apache, sebuah lisensi yang mempromosikan konsep perangkat lunak yang terbuka dan mendukung platform perangkat seluler yang bersifat terbuka pula[11].

Versi pertama Android yang dirilis secara publik adalah Android 1.0 pada tahun 2008. Android 1.0 masih memiliki antarmuka yang sederhana dan terbatas fitur dibandingkan dengan versi-versi yang lebih baru. Namun, seiring berjalannya waktu, Android mengalami perkembangan pesat dengan peningkatan fitur dan fungsionalitas yang signifikan[12].

Setiap versi Android memiliki nama kode berdasarkan nama-nama makanan manis, seperti *Cupcake, Donut, Eclair, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, Pie,* dan versi terbaru, yaitu Android 10, Android 11, dan Android 12.[12]

2.4.3 Versi-versi android

2.4.3.1 Android Cupcake (1.5)

Android Cupcake dirilis pada April 2009, merupakan versi Android pertama yang memiliki nama kode berdasarkan makanan manis. Versi ini menambahkan fitur-fitur penting seperti *keyboard* virtual yang disempurnakan, kemampuan untuk mengunggah video ke YouTube, dan dukungan untuk format *file* multimedia baru[12].

2.4.3.2 Android Gingerbread (2.3)

Dirilis pada Desember 2010, Android Gingerbread menghadirkan peningkatan performa dan antarmuka pengguna yang lebih baik. Versi ini juga memperkenalkan fitur-fitur baru seperti pembaruan *keyboard*, dukungan untuk *near field communication* (NFC), dan dukungan kamera yang ditingkatkan[12].

2.4.3.3 Android Jelly Bean (4.1-4.3)

Jelly Bean adalah serangkaian versi Android yang dirilis antara Juli 2012 hingga Juli 2013. Versi ini menawarkan peningkatan kecepatan dan responsivitas, antarmuka pengguna yang ditingkatkan dengan proyeksi layar (*screen mirroring*) nirkabel, notifikasi yang ditingkatkan, dan peningkatan fitur aksesibilitas[12].

2.4.3.4 Android KitKat (4.4)

Dirilis pada Oktober 2013, Android KitKat menghadirkan perubahan desain antarmuka yang disebut "*Material Design*". Versi ini juga memperkenalkan fitur-fitur seperti "OK Google" yang memungkinkan pengguna menggunakan perintah suara, dukungan printer nirkabel, dan peningkatan manajemen memori[12].

2.4.3.5 Android Lollipop (5.0-5.1)

Dirilis pada November 2014, membawa perubahan desain yang signifikan dengan Material Design. Versi ini juga memperkenalkan fitur seperti pemberitahuan yang ditingkatkan, mode "*Guest*" untuk penggunaan bersama, dan peningkatan daya tahan baterai melalui "*Project Volta*"[12].

2.4.3.6 Android Marshmallow (6.0)

Dirilis pada Oktober 2015, menawarkan fitur-fitur seperti kontrol perizinan yang lebih baik, *Doze Mode* untuk mengoptimalkan daya tahan baterai, dan *Now On Tap* yang memberikan informasi kontekstual dari aplikasi yang sedang digunakan[12].

2.4.3.7 Android Nougat (7.0-7.1)

Dirilis pada Agustus 2016, menghadirkan fitur-fitur seperti mode *multi*-jendela (*multi-window*), pembaruan notifikasi yang lebih baik, dan peningkatan keamanan dengan *Direct Boot* dan pembaruan sistem yang lebih lancar[12].

2.4.3.8 Android Oreo (8.0-8.1)

Dirilis pada Agustus 2017, membawa perubahan seperti mode tampilan gambar dalam gambar, pembaruan notifikasi, *Autofill* API untuk mengisi formulir dengan mudah, dan peningkatan keamanan dengan *Play Protect*[12].

2.4.3.9 Android Pie (9.0)

Dirilis pada Agustus 2018, menawarkan navigasi gesek (*gestural navigation*), *Adaptive Battery* untuk mengoptimalkan penggunaan daya, *Adaptive Brightness* untuk menyesuaikan kecerahan layar, serta *Digital Wellbeing* untuk membantu pengguna mengelola waktu layar mereka[12].

2.4.3.10 Android 10

Dirilis pada September 2019, memperkenalkan mode gelap sistem, kontrol perizinan yang lebih ketat, navigasi gesek yang diperbarui, dan peningkatan keamanan dan privasi, termasuk izin lokasi yang lebih terbatas[12].

2.4.3.11 Android 11

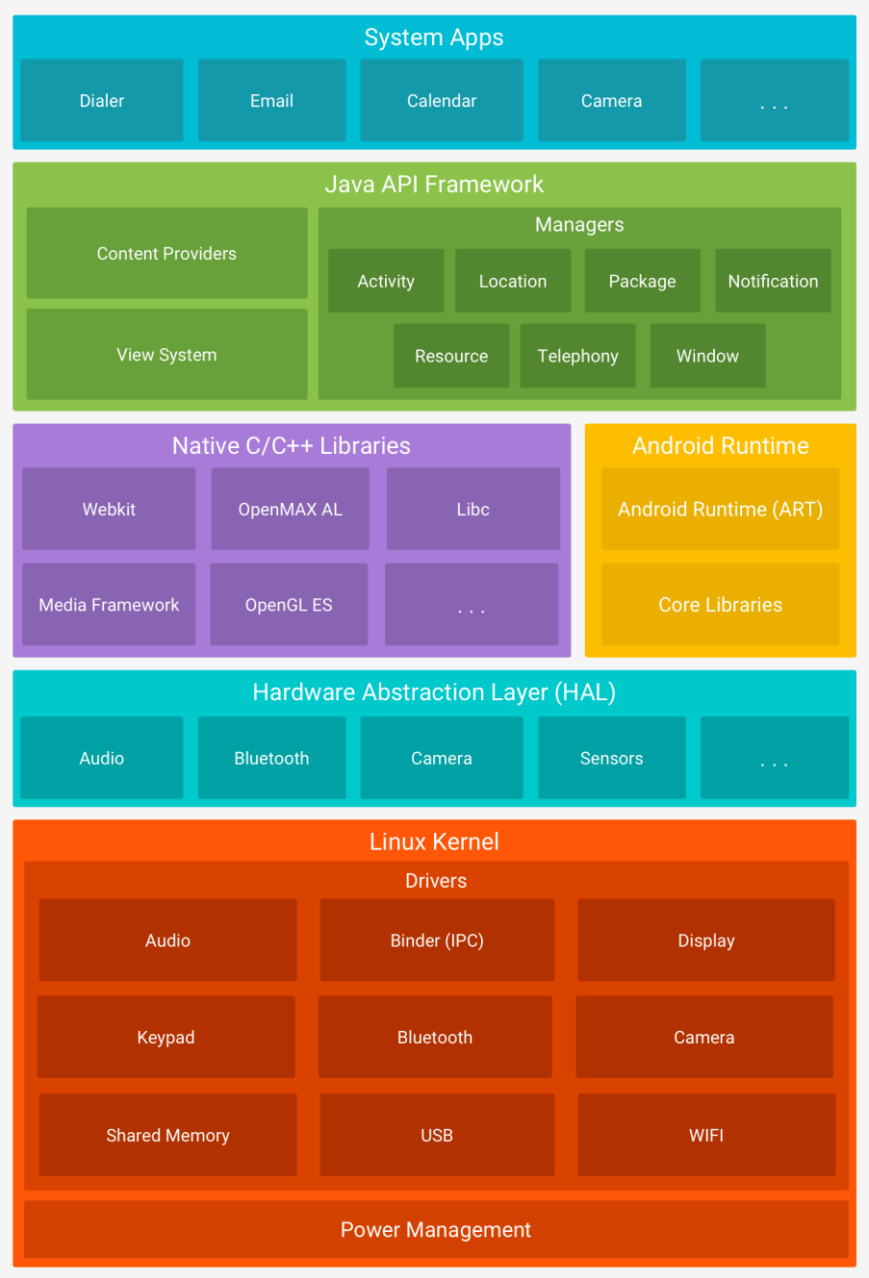
Dirilis pada September 2020, membawa fitur-fitur seperti *Bubbles* untuk percakapan, pengendalian perangkat pintar melalui menu daya, peningkatan pembaruan aplikasi secara otomatis, dan peningkatan keamanan dengan izin satu kali (*one-time permission*)[12].

2.4.3.12 Android 12

Dirilis pada tahun 2021, menghadirkan perubahan desain yang signifikan dengan *Material You*, pembaruan privasi dengan indikator mikrofon dan kamera,

serta fitur-fitur seperti *Smart Reply*, *Auto-rotate* menggunakan kamera, dan optimisasi performa[12].

2.4.4 Arsitektur Platform



Gambar 2.2 Arsitektur Platform Android

Arsitektur Android terdiri dari beberapa komponen utama yang bekerja bersama untuk menyediakan lingkungan yang kuat untuk pengembangan aplikasi *mobile*. Berikut adalah penjelasan tentang setiap komponen utama[13]:

1. *Kernel Linux*

   Android menggunakan *kernel Linux* sebagai lapisan dasar dari sistem operasinya. *Kernel Linux* bertanggung jawab untuk mengelola sumber daya perangkat keras seperti memori, proses, manajemen daya, dan komunikasi perangkat keras.

2. Perpustakaan Perangkat Keras (*Hardware Abstraction Layer* - HAL)

   HAL adalah lapisan perantara antara *kernel Linux* dan perangkat keras fisik. HAL menyediakan antarmuka standar yang memungkinkan sistem operasi untuk berkomunikasi dengan berbagai perangkat keras yang berbeda. Ini memungkinkan pengembang untuk mengakses dan menggunakan fungsionalitas perangkat keras secara umum melalui API Android.

3. *Android Runtime* (ART)

   Android Runtime (ART) adalah lingkungan eksekusi aplikasi di Android. Pada versi Android sebelum 5.0, Android menggunakan *Dalvik Virtual Machine* (DVM). Namun, mulai dari Android 5.0, Android beralih ke ART yang menggunakan kompilasi *ahead-of-time* (AOT). Dalam AOT, kode sumber aplikasi dikompilasi menjadi kode mesin sebelum aplikasi dijalankan. Ini menghasilkan kinerja yang lebih cepat dan penghematan daya baterai.

4. Pustaka C/C++ Bawaan

   Sejumlah komponen dan layanan inti dari sistem Android, seperti ART dan HAL, diterjemahkan ke dalam kode bawaan yang ditulis dalam bahasa C dan C++. Untuk meningkatkan fungsionalitas pustaka bawaan ini dalam aplikasi, platform Android memungkinkan penggunaan kerangka kerja API Java. Contohnya, pengguna dapat menggunakan kerangka kerja API *OpenGL Java Android* untuk mengakses *OpenGL* ES, sehingga mendukung penggambaran dan manipulasi grafik 2D dan 3D dalam aplikasi.

5. *Framework* Aplikasi

   Framework Aplikasi Android adalah kumpulan pustaka dan komponen yang menyediakan berbagai fitur dan fungsionalitas untuk pengembangan aplikasi Android. *Framework* ini termasuk komponen seperti *Activity, Service, Content Provider, dan Broadcast Receiver* yang memungkinkan pengembang untuk membangun aplikasi yang interaktif dan terintegrasi dengan sistem operasi.

6. Aplikasi Sistem

   Android menyediakan sejumlah aplikasi inti seperti email, pesan SMS, kalender, penjelajah web, kontak, dan sebagainya. Aplikasi-aplikasi ini tidak memiliki status khusus yang mengharuskan pengguna untuk memasang aplikasi yang disertai dengan *platform*. Oleh karena itu, pengguna dapat memilih aplikasi pihak ketiga sebagai penjelajah web utama, pengolah pesan SMS, atau bahkan *keyboard* utama (kecuali beberapa pengecualian seperti aplikasi pengaturan sistem).

   Aplikasi-aplikasi sistem ini berfungsi sebagai aplikasi yang digunakan oleh pengguna dan memberikan kemampuan kunci yang dapat diakses oleh pengembang dari aplikasi mereka sendiri. Sebagai contoh, jika aplikasi yang sedang dikembangkan ingin mengirim pesan SMS, tidak perlu membangun fungsi tersebut secara mandiri. Pengembang dapat menjalankan aplikasi SMS mana pun yang sudah dipasang untuk mengirim pesan kepada penerima yang ditentukan.

2.5 Kotlin

Gambar 2.3 Kotlin

Kotlin merupakan bahasa pemrograman sumber terbuka yang dikembangkan oleh JetBrains. Bahasa pemrograman ini mendukung penggunaan *multi-platform*, namun bahasa ini banyak digunakan untuk membuat aplikasi android. Kotlin berjalan di atas *platform Java Virtual Machine* (JVM) yang memungkinkan komputer untuk menjalankan kode berbasis Java atau yang dikompilasi menggunakan Java. Bahasa Kotlin adalah pengembangan dari bahasa Java yang sudah popular sebelumnya. Bahasa Kotlin memiliki fitur-fitur bahasa modern yang lebih dibandingkan bahasa Java[14].

2.6 *Pose Detection API*

Pose Detection API adalah sebuah *tools* API dari ML Kit buatan Google yang menghadirkan kemampuan *machine learning* di perangkat *mobile* untuk aplikasi Android dan iOS. *Tools* ini merupakan solusi praktis dan ringan untuk para pengembang aplikasi untuk mendeteksi pose tubuh secara *real-time* dari video atau gambar[15].

Deteksi pose dilakukan dengan mengidentifikasi *landmark* kerangka pada berbagai bagian tubuh, seperti bahu dan pinggul. *Pose Detection API* tidak memerlukan peralatan khusus atau keahlian *machine learning* untuk mendapatkan hasil yang sangat baik. Namun, deteksi pose hanya dapat dilakukan dengan optimal jika wajah dan seluruh tubuh subjek terlihat dalam gambar[15].

Kemampuan utama dari *Pose Detection API* ini adalah sebagai berikut[15]:

1. Mendukung lintas platform, yaitu bisa digunakan di Android dan iOS.
2. Menampilkan 33 titik kerangka penting, termasuk posisi tangan dan kaki.

3. Dapat menampilkan tingkat keyakinan berupa skor untuk setiap titik yang dikenali seperti bahu, pinggul, siku, dll. Skor ini menunjukkan probabilitas bahwa tempat terkenal tersebut berada dalam bingkai gambar. Skor ini memiliki rentang 0,0 hingga 1,0, dengan 1,0 menunjukkan keyakinan tinggi.
4. SDK yang dioptimalkan. Alat ini menampilkan hasil dengan kecepatan masing-masing ~30 dan ~45 fps. Namun, presisi koordinat tempat terkenal dapat bervariasi. SDK yang akurat akan menampilkan hasil pada kecepatan frame yang lebih lambat, tetapi menghasilkan nilai koordinat yang lebih akurat.
5. Koordinat Z untuk analisis kedalaman. Nilai ini dapat membantu menentukan apakah bagian tubuh pengguna berada di depan atau di belakang pinggul pengguna. Koordinat Z adalah nilai eksperimental yang dihitung untuk setiap tempat terkenal. Nilai diukur dalam "piksel gambar" seperti koordinat X dan Y, tetapi bukan nilai 3D yang sebenarnya. Sumbu Z tegak lurus dengan kamera dan melewati pinggul subjek. Asal sumbu Z adalah perkiraan titik tengah antara pinggul (kiri/kanan dan depan/belakang relatif terhadap kamera). Nilai Z negatif menunjukkan mengarah ke kamera. Koordinat Z tidak memiliki batas atas atau bawah.

Tempat terkenal	Jenis	Posisi	InFrameLikelihood
11	LEFT_SHOULDER	(734,9671, 550,7924, -118,11934)	0,9999038
12	RIGHT_SHOULDER	(391,27032, 583,2485, -321,15836)	0,9999894

Gambar 2.4 Koordinat Landmark

Berikut ini merupakan integrasi *Pose Detection API* pada android secara garis besar dalam bahasa Kotlin:

1. Menginstal *dependencies* dalam file *build.gradle*

```
dependencies {
  // If you want to use the base sdk
  implementation 'com.google.mlkit:pose-detection:18.0.0-beta3'
  // If you want to use the accurate sdk
  implementation 'com.google.mlkit:pose-detection-accurate:18.0.0-beta3'
}
```

Gambar 2.5 Dependencies Pose Detection API

2. Membuat *Pose Detector*, terdapat 2 pilihan mode deteksi. *STREAM_MODE* untuk mendeteksi secara *real-time* dan *SINGLE_IMAGE_MODE* untuk mendeteksi pada gambar statis.

```
// Base pose detector with streaming frames, when depending on the pose-detection sdk
val options = PoseDetectorOptions.Builder()
    .setDetectorMode(PoseDetectorOptions.STREAM_MODE)
    .build()

// Accurate pose detector on static images, when depending on the pose-detection-accurate sdk
val options = AccuratePoseDetectorOptions.Builder()
    .setDetectorMode(AccuratePoseDetectorOptions.SINGLE_IMAGE_MODE)
    .build()
```

Gambar 2.6 Pilihan Mode Deteksi

```
val poseDetector = PoseDetection.getClient(options)
```

Gambar 2.7 Instance Pose Detector

3. Mempersiapkan masukkan gambar. Masukkan gambar bisa berupa *media.image*, *Bitmap*, *ByteBuffer*, atau dari *file* di perangkat.

```
private class YourImageAnalyzer : ImageAnalysis.Analyzer {

    override fun analyze(imageProxy: ImageProxy) {
        val mediaImage = imageProxy.image
        if (mediaImage != null) {
            val image = InputImage.fromMediaImage(mediaImage, imageProxy.imageInfo.rotationDegrees)
            // Pass image to an ML Kit Vision API
            // ...
        }
    }
}
```

Gambar 2.8 Masukkan Gambar media.image

4. Memproses Gambar. Gambar masukkan diteruskan ke *pose detector*

```
Task<Pose> result = poseDetector.process(image)
        .addOnSuccessListener { results ->
            // Task completed successfully
            // ...
        }
        .addOnFailureListener { e ->
            // Task failed with an exception
            // ...
        }
```

Gambar 2.9 Kode Pose Detector

5. Dapatkan informasi *landmark*. Jika seseorang terdeteksi dalam gambar, API deteksi pose akan menampilkan objek pose dengan 33 titik landmark. Jika orang tersebut tidak sepenuhnya berada dalam gambar, akan menetapkan koordinat *landmark* yang hilang di luar *frame* dan memberinya nilai *InFrameConfidence* yang rendah. Jika tidak ada orang yang terdeteksi dalam *frame*, maka *landmark* akan kosong

```
// Get all PoseLandmarks. If no person was detected, the list will be empty
val allPoseLandmarks = pose.getAllPoseLandmarks()

// Or get specific PoseLandmarks individually. These will all be null if no person
// was detected
val leftShoulder = pose.getPoseLandmark(PoseLandmark.LEFT_SHOULDER)
val rightShoulder = pose.getPoseLandmark(PoseLandmark.RIGHT_SHOULDER)
val leftElbow = pose.getPoseLandmark(PoseLandmark.LEFT_ELBOW)
```

Gambar 2.10 Mendapatkan Informasi Koordinat Landmark

2.7 *Unified Modelling Language*

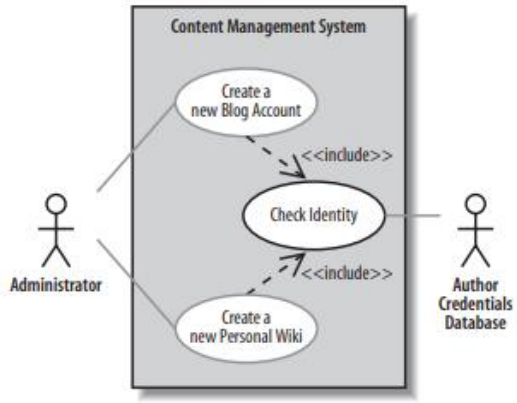
Unified Modelling Language (UML) adalah cara standar untuk memodelkan sistem, terutama sistem perangkat lunak. Pemodelan dapat terdiri dari berbagai elemen, seperti pseudo-kode, kode asli, gambar, diagram, atau penjelasan panjang. Dalam hal ini, pemodelan merupakan segala bentuk yang dapat membantu menggambarkan sistem secara lebih baik[16]. Penggunaan UML diharapkan dapat

mempermudah pengembangan perangkat lunak dengan memenuhi kebutuhan pengguna secara tepat dan efektif, termasuk keamanan, keandalan, dan skalabilitas.

UML juga berguna untuk mentransfer pengetahuan tentang aplikasi sistem dari satu pengembang ke pengembang lainnya. Selain itu, fungsi lainnya adalah sebagai jembatan antara pengembang sistem dan pengguna untuk memahami sistem yang akan dikembangkan. Terlebih lagi, UML mudah dipelajari tidak hanya untuk para pengembang, tetapi juga untuk para pebisnis.

UML memiliki beberapa jenis diagram, yaitu *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram,* dan *Class Diagram.*

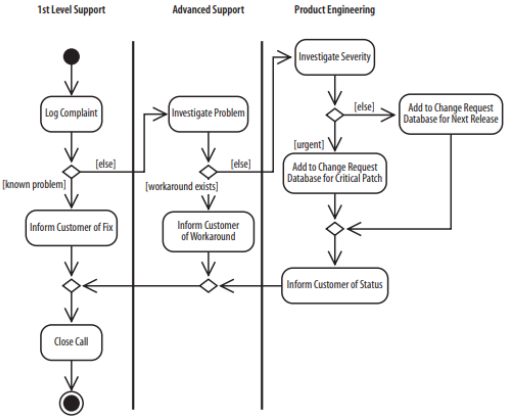
2.7.1 *Use Case Diagram*



Gambar 2.11 Use Case Diagram

Diagram ini menggambarkan hubungan interaksi antara sistem dan aktor yang terlibat pada sistem. *Use Case* dapat mendeskripsikan interaksi apa yang terjadi pada si pengguna sistem dengan sistemnya. Untuk melakukan pemodelan ini, diperlukan adanya suatu diagram yang berisikan aksi-aksi aktor dengan sistemnya itu sendiri.

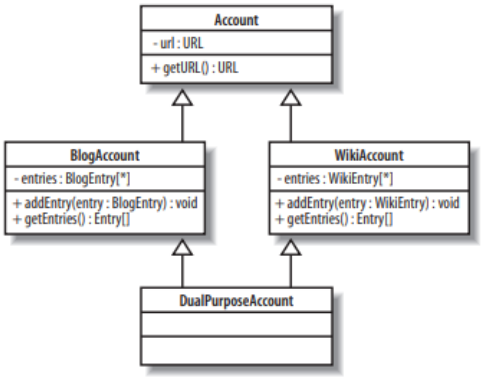
2.7.2 *Activity Diagram*



Gambar 2.12 Activity Diagram

Activity Diagram adalah jenis diagram yang digunakan untuk memodelkan proses yang terjadi dalam sistem. Diagram ini menunjukkan urutan langkah dalam proses secara vertikal, dan sering digunakan dalam pengembangan sistem dengan menggunakan UML, terutara pada pengembangan *Use Case*. Dengan menggunakan *Activity Diagram*, proses berjalan dalam sistem dapat divisualisasikan sehingga dapat membantu pengembang dalam merancang dan memperbaiki proses yang ada.

2.7.3 *Class Diagram*

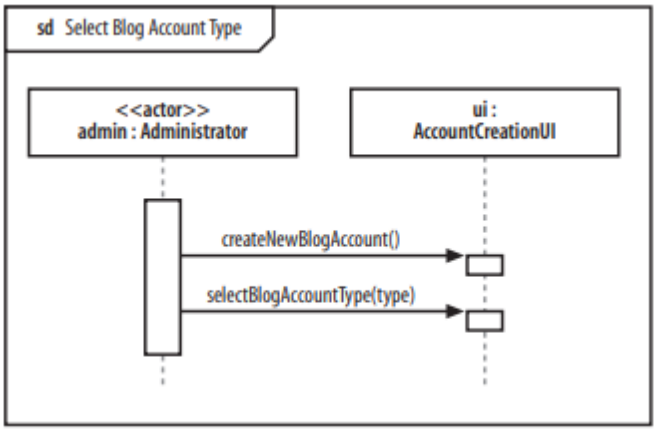


Gambar 2.13 Class Diagram

Class Diagram adalah suatu bentuk diagram yang digunakan untuk merepresentasikan kelas-kelas dan hubungan antara kelas-kelas dalam suatu sistem. Ada dua jenis yang sering digunakan dalam pengembangan perangkat lunak. Yang

pertama adalah *Class Diagram* yang mencerminkan model domain yang merepresentasikan abstraksi basis data. Yang kedua adalah *Class Diagram* yang merupakan bagian dari *pattern* MVC (*Model-View-Controller*), yang memiliki kelas-kelas sebagai antarmuka, kelas kontrol sebagai algoritma program, dan kelas entitas sebagai tabel dalam basis data.

2.7.4 *Sequence Diagram*



Gambar 2.14 Sequence Diagram

Sequence Diagram adalah jenis diagram pada UML yang digunakan untuk menggambarkan interaksi antar objek secara kronologis, atau dalam urutan waktu. Diagram ini dapat memperlihatkan tahapan yang harus dilakukan oleh suatu objek atau beberapa objek untuk mencapai suatu tujuan tertentu, seperti yang telah didefinisikan pada diagram *use case*.

2.8 Skala likert

Metode skala Likert, juga dikenal sebagai skala *rating* terakumulasi, adalah pendekatan pengukuran yang digunakan untuk menilai tanggapan positif dan negatif terhadap pernyataan tertentu. Terdapat empat opsi skala yang sering digunakan dalam kuesioner skala Likert, yang mengharuskan responden memilih di antara pilihan-pilihan tersebut karena opsi netral tidak disediakan[17].

Dalam skala Likert, responden memberikan tingkat persetujuan terhadap pernyataan tertentu dengan memilih salah satu dari pilihan yang tersedia. Umumnya, terdapat lima pilihan dalam skala dengan format sebagai berikut[18]:

Pernyataan Positif (Skor terendah ke tertinggi):

a. Sangat (tidak setuju/buruk/kurang sekali)
b. Tidak (setuju/baik/) atau kurang
c. Netral / Cukup
d. Setuju/Baik/Suka
e. Sangat (setuju/baik/suka)

Pernyataan Negatif (Skor terendah ke tertinggi):

a. Sangat (setuju/baik/suka)
b. (setuju/baik/suka)
c. Netral / Cukup
d. Tidak (setuju/baik/) atau kurang
e. Sangat (tidak setuju/buruk/kurang sekali)


UNIVERSITAS KOMPUTER INDONESIA
UNIKOM