BAB II
TEORI PENUNJANG

Bab ini berisi penjelasan mengenai konsep dasar yang digunakan sebagai acuan atau penunjang dalam perancangan aplikasi serta tools yang digunakan untuk membangun aplikasi. Diantaranya Sarana Umum, Graf, Algoritma Dijkstra, Android, Android SDK, UML, Alpha dan Beta Testing dan Skala likert.

2.1 Sarana Umum

Menurut Kamus Besar Bahasa Indonesia (KBBI), sarana adalah segala sesuatu yang dapat dipakai sebagai alat dalam mencapai maksud atau tujuan [11], sedangkan sarana umum atau fasilitas umum adalah fasilitas yang diadakan untuk kepentingan umum. Dikatakan "umum" karena keberadaan wadah atau tempat ini bersifat mempermudah atau memperlancar terpenuhinya kebutuhan-kebutuhan bersama seluruh masyarakat, misalnya di bidang keamanan, komunikasi, rekreasi, olahraga, pendidikan, kesehatan, administrasi publik, relijius dan sosial-budaya.

2.1.1 Mall

Mall adalah jenis dari pusat perbelanjaan yang berupa bangunan tertutup dengan kenyamanan termal yang telah disesuaikan dengan kebutuhan dan memiliki jalur untuk berjalan jalan yang teratur sehingga berada di antara antar toko-toko kecil yang saling berhadapan. [18]

2.1.2 Kampus

Kampus atau perguruan tinggi adalah lembaga ilmiah yang mempunyai tugas menyelenggarakan pendidikan dan pengajaran di atas perguruan tingkat menengah, dan yang memberikan pendidikan dan pengajaran berdasarkan kebudayaan kebangsaan Indonesia dan dengan cara ilmiah. Bentuk Perguruan Tinggi terdiri atas: universitas, institut, sekolah tinggi, politeknik, akademi dan akademi komunitas. [15]

2.1.3 Hotel

Hotel adalah penyediaan akomodasi secara harian berupa kamar-kamar di dalam 1 (satu) bangunan, yang dapat dilengkapi dengan jasa pelayanan makanan dan minuman, kegiatan hiburan serta fasilitas lainnya. [16]

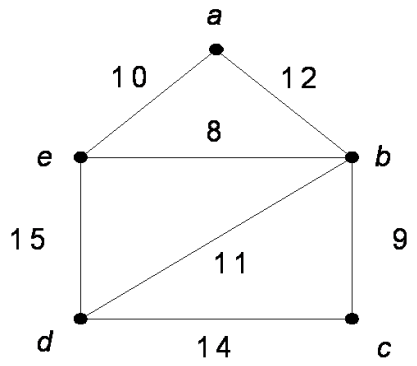
2.1.4 Rumah Sakit

Menurut *World Health Organization*, Pengertian Rumah Sakit adalah bagian integral dari suatu organisasi sosial dan kesehatan dengan fungsi menyediakan pelayanan paripurna (komprehensif), penyembuhan penyakit (kuratif) dan pencegahan penyakit (preventif) kepada masyarakat. Berdasarkan Undang-undang No.44 Tahun 2009 tentang Rumah Sakit, "Rumah Sakit adalah institusi pelayanan kesehatan yang menyelenggarakan pelayanan kesehatan perorangan secara paripurna yang menyediakan pelayanan rawat inap, rawat jalan dan gawat darurat". [17]

2.2 Graf

Graf adalah kumpulan simpul (*nodes*) yang dihubungkan satu sama lain melalui sisi/garis (*edges* atau *arcs*). Representasi visual dari graf adalah suatu objek dinyatakan sebagai noktah, bulatan atau titik, dan hubungan antar objek dinyatakan dengan garis. [12].

Secara matematis graf didefinisikan sebagai pasangan himpunan (V, E), ditulis dengan notasi G = (V, E), yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul (*vertex* atau *node*) dan E adalah himpunan sisi *(edge)* yang menghubungkan sepasang simpul. Simpul *(vertex)* pada graf dapat dinyatakan dengan huruf, bilangan atau gabungan keduanya. Sedangkan sisi-sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan pasangan (u, v) atau dinyatakan dengan lambang e1, e2, e3 dan seterusnya. Dengan kata lain, jika e adalah sisi yang menghubungkan simpul u dengan simpul v, maka e dapat dituliskan sebagai e = (u, v). [12], [1]



Gambar 2.1 Contoh Sebuah Graf Sederhana

2.3 Algoritma Dijkstra

Algoritma Dijkstra adalah sebuah algoritma untuk memecahkan permasalahan jalur terpendek *(shortest path problem)* pada sebuah graf berarah *(directed graf)* atau graf tidak berarah *(undirected graf)* dengan bobot-bobot sisi *(edge weights)* yang bernilai tidak negatif. Bila node dari sebuah graf melambangkan kota-kota dan bobot sisi *(edge weights)* melambangkan jalur antara kota-kota tersebut, maka algoritma Dijkstra dapat digunakan untuk menemukan jalur terpendek antara dua kota. [19]

Algoritma Dijkstra merupakan *greedy algorithm* yang berfungsi menentukan rute terpendek berdasarkan bobot terkecil dari satu titik ke titik lain nya. Karena merupakan *greedy algorithm* atau algoritma rakus maka algortima ini hanya memberikan solusi terbaik tanpa memberikan alternatifnya. Adapun nama algoritma Dijkstra sendiri berasal dari penemunya yaitu Edsger Dijkstra pada tahun 1959. [1]

Berikut ini *pseudocode* algoritma Dijkstra dalam mencari rute terpendek pada sebuah graf :

algorithm *DijkstraShortestWeightedPath*(*G*, *s*)
_ ***pre-cond_:*** *G* is a weighted (directed or undirected) graph, and *s* is one of its nodes.
_ ***post-cond_:*** *π* specifies a shortest weighted path from *s* to each node of *G*, and
d specifies their lengths.
begin
$d(s) = 0, \pi(s) = _$
for other *v*, $d(v)=\infty$and $\pi(v) = nil$
handled = ∅
notHandled = priority queue containing all nodes. Priorities given by $d(v)$.
loop
loop-invariant: See above.
exit when *notHandled* = ∅
let *u* be a node from*notHandled* with smallest $d(u)$
for each *v* connected to *u*
foundPathLength = $d(u) + w_{_u,v_}$
if $d(v) >$ *foundPathLength* then
$d(v) =$ *foundPathLength*
$\pi(v) = u$
(update the *notHandled* priority queue)
end if
end for
move *u* from *notHandled* to *handled*
end loop
return $_d, \pi_$
end algorithm.

Dalam hal ini node yang dikunjungi merupakan node yang terpendek dari setiap tahap algoritma Dijkstra. Jadi jalur atau rute yang dibentuk oleh algoritma Dijkstra tersusun dari node yang telah ditemukan dan telah dikunjungi.

Adapun langkah-langkah dari algoritma Dijkstra sesuai *pseudocode* diatas yaitu:
1. Langkah pertama yaitu menetapkan node awal sebagai status ditemukan (*found*) dan kemudian dikunjungi atau ditangani(*handled*)

2. Langkah kedua yaitu dilakukan pencarian terhadap setiap node yang dapat dicapai secara langsung dari node yang sedang dikunjungi

3. Langkah ketiga yaitu:

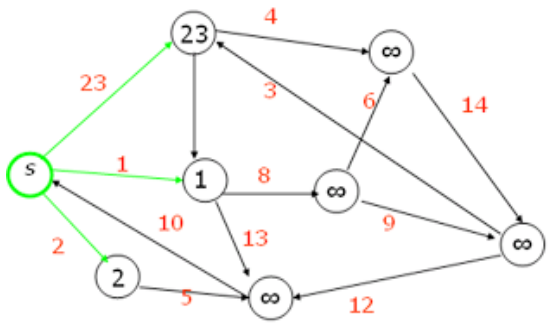
- Apabila node yang didapatkan pada langkah kedua belum pernah ditemukan, maka rubah statusnya menjadi ditemukan

- Apabila node yang didapatkan sudah pernah ditemukan maka lakukan *update* pada bobotnya, ambil bobot yang lebih kecil

4. Langkah keempat yaitu dilakukan pencarian terhadap node yang memiliki bobot paling kecil dari semua node yang berada pada status ditemukan kemudian mengunjunginya.

5. Lakukan *looping* secara berurutan pada langkah kedua, ketiga dan keempat sampai semua node ditemukan. [1]

Contoh graph algoritma Dijkstra :



Gambar 2.2 Contoh Graph Algoritma Dijkstra

2.4 Android

Android adalah sebuah sistem operasi yang berbasis Linux yang mencakup sistem operasi, *middleware,* dan apikasi. Android menyediakan *platform* terbuka bagi para pengembang sehingga dapat menciptakan aplikasi dengan leluasa untuk digunakan oleh para pengguna *smartphone* android. Pada awalnya android dikembangkan oleh Android Inc, kemudian dibeli oleh Google Inc.

Sebagai sistem operasi untuk perangkat mobile, Android disebut sebagai platform mobile pertama yang Lengkap, Terbuka, dan Bebas.

1. Lengkap (*Complete Platform*) : Para desainer dapat melakukan pendekatan yang komperhensif ketika mereka sedang mengembangkan *platform* android. Android

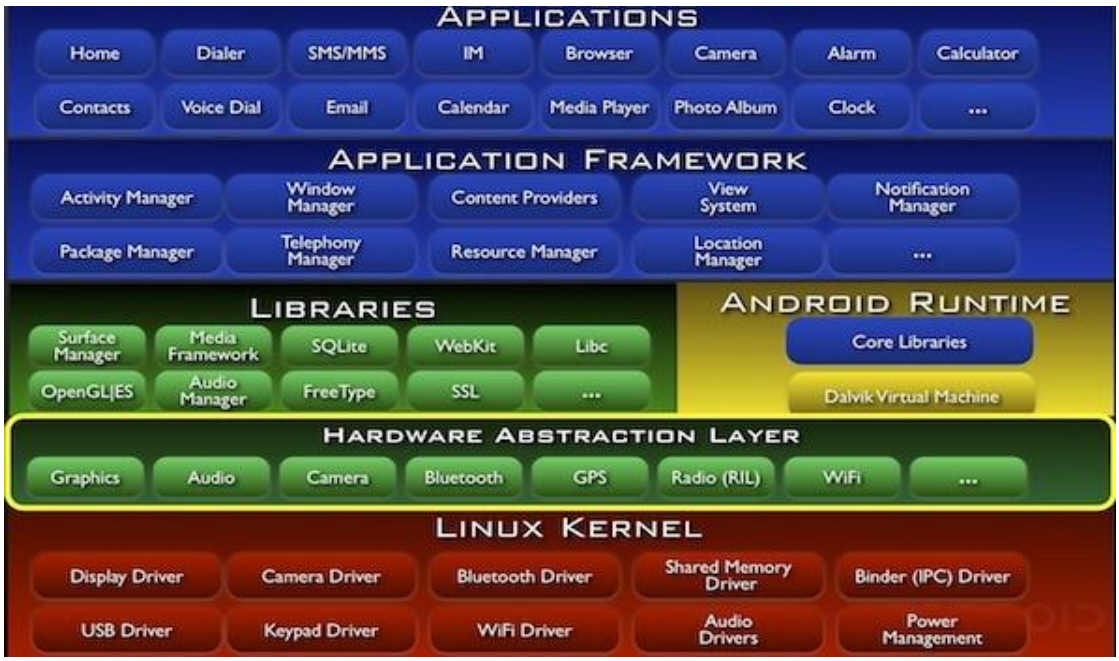
merupakan sistem operasi yang banyak menyediakan *tools* dan membangun *software* dan memungkinkan untuk peluang pengembangan aplikasi.

2. Terbuka (*Open Source Platform*) : *Platform* Android disediakan melalui lisensi *open source*. Pengembang dapat dengan bebas untuk mengembangkan aplikasi.

3. Bebas (*Free Platform*) : Android adalah *platform* atau aplikasi yang bebas untuk *developer*. Tidak ada lisensi atau biaya royalti utuk dikembangkan pada *platform* Android. Tidak ada biaya keanggotaan diperlukan. Tidak diperlukan biaya pengujian. Tidak ada kontrak yang diperlukan. Aplikasi untuk android dapat didistribusikan dan diperdagangkan dalam bentuk apa pun. [6]

2.4.1 Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut :



Gambar 2.3 Arsitektur Android

- ***Applications dan Widgets***

  Applications dan Widgets ini adalah layer di mana kita berhubungan dengan aplikasi saja, di mana biasana kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan *aplikasi tersebut. Di Layer* terdapat aplikasi inti

termasuk klien *email*, program *SMS*, kalender, peta, *browse*r, kontak, dan lain-lain.

- ***Applications Frameworks***

  Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur *alarm*, dan menambahkan status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju *API framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan*(reuse).* Sehingga bisa kita simpulkan *Applications Framework* ini adalah layer di mana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content-providers* yang berupa sms dan panggilan telepon.

  Libraries, ini adalah *layer* di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasi. Berjalan di atas *kernel*, *Layer* ini meliputi berbagai *library C/C++* inti seperti *Libc* dan *SSL*, serta:

  - *Libraries* media untuk pemutaran media audio dan video
  - *Libraries* untuk manajemen tampilan
  - *Libraries Graphics* mencakup *SGL* dan *OpenGL* untuk grafis 2D dan 3D
  - *Libraries SWLite* untuk dukungan *database*
  - *Libraries SSL* dan *WebKit* terintegrasi dengan *web browser* dan *security*
  - *Libraries liveWebcore* mencakup modern *web browser* dengan *engine embeded web view.*
  - *Libraires 3D* yang mencakup implementasi *OpenGL S 1.0 API's*

- ***Android Run Time***

  Layer yang membuat aplikasi Android dapat dijalankan di mana dalam prosesnya menggunakan implementasi *Linux*. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam *Android Run Time* dibagi menjadi dua bagian yaitu :

  - *Core libraries* : Aplikasi Android dibangun dalam bahasa java, sementara *Dalvik* sebagai *virtual* mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa java/C yang ditangani oleh *Core Libraries.*
  - *Dalvik Virtual Machine* : *Virtual* mesin berbasis *reSIGter* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat *linux kernel* untuk melakukan *threading* dan manajemen tingkat rendah.

- ***Linux Kernel***

  Linux Kernel adalah *layer* di mana inti dari sistem operasi dari Android itu berada. Berisi file-file system yang mengatur sistem *processing, memory, resource,driver,* dan sistem-sistem operasi Android lainnya.[9]

2.4.2 Android SDK (Software Development Kit)

Android SDK adalah tools API (*Application Programming Interface*) yang digunakan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. [3]

2.4.3 ADT (Android Development Tools)

Plugins Android Development Tool (ADT) berguna sebagai pengenal Android di dalam IDE *Eclipse*. Dengan ADT *plugins* kita bisa membuat *project* aplikasi Android baru, mengakses *tools emulator*, dan perangkat Android, melakukan kompilasi dan men-*debug* aplikasi, mengekspor aplikasi ke Android *Packages* (APK), membuat sertifikasi digital terhadap kode program APK. [6]

2.5 Google Maps

Google maps merupakan sebuah layanan peta dunia virtual berbasis web yang disediakan oleh Google. Layanan ini gratis dan dapat ditemukan di http://maps.google.com. *Google Maps* menawarkan peta yang dapat digeser *(panned),* diperbesar (*zoom in*), diperkecil (*zoom out),* dapat diganti dalam beberapa mode (*map, satellite, hybrid,* dan lain-lain). [3]

2.6 Eclipse

Eclipse adalah sebuah aplikasi IDE (*Integrated Development Environment*). IDE yang berfungsi untuk membantu dalam pengembangan pembuatan aplikasi. IDE sendiri biasanya terdiri dari *source code editor, build automation tools, debugger* dan lain sebagainya. [3]

2.7 GPS (Global Positioning System) dan A-GPS (Assisted Global Positioning System)

GPS merupakan suatu sistem yang dapat membantu mengetahui keberadaan kita saat ini. Dengan mentransmisikan sinyal dari satelit ke perangkat GPS, maka didapatlah data yang akurat. A-GPS didesain agar perangkat dapat terhubung dengan satelit dengan lebih cepat dan lebih dapat diandalkan daripada menggunakan GPS tunggal, karena data yang diambil telah disimpan di *server* GPS.

Untuk Keakuratan data, A-GPS memiliki keterbatasan yaitu dipengaruhi oleh halangan dari gedung-gedung yang tinggi, kondisi cuaca dan lokasi pengguna. A-GPS menggunakan jaringan seluler 2G dan 3G serta koneksi paket data GPRS atau EDGE. Proses mengunduh data membutuhkan transmisi melalui *provider* layanan jaringan yang dipakai, oleh sebab itu dikenakan biaya pada saat penggunaan A-GPS. [7]

2.8 LBS (Location Based Service)

Sistem Layanan Berbasis Lokasi, atau lebih dikenal dengan *Location-Based Service* (LBS), menggabungkan antara proses dari layanan mobile dengan posisi geografis dari penggunanya. Poin pentingnya adalah ketika posisi target, di mana sebuah target bisa jadi adalah pengguna LBS itu sendiri atau entitas lain yang tergabung dalam suatu layanan. LBS mengarah pada aplikasi-aplikasi dan layanan-

layanan yang menyediakan informasi lokasi mengenai pengguna mobile, untuk menyediakan sebuah layanan atau informasi khusus pada pengguna. Dua unsur utama LBS adalah :

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

Menyediakan *tool* atau *source* untuk LBS, *Application Program Interface* (API). *Maps* menyediakan fasilitas untuk menampilakan, memanipulasi *maps* atau peta beserta fitur-fitur lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada com.google.android.map.

b. *Location Provider (*API *Location)*

Menyediakan teknologi pencarian lokasi yang digunakan oleh *device* atau perangkat. API *Location* berhubungan dengan data GPS *(Global Positioning System)* dan data lokasi *real-time.* API *Location* berapa pada paket android yaitu dalam paket *android.location.* Dengan *Location Manager*, kita dapat menentukan lokasi kita saat ini, *Track* gerakan atau perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan. [6]

2.9 JDK (Java Development Kit)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat digunakan untuk mengembangkan perangkat lunak yang berbasis Java, sedangkan JRE adalah sebuah implementasi dari *Java Virtual Machine* yang benar-benar digunakan untuk menjalankan program java. Biasanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber *compiler* java, *bundling, debuggers, development libraries* dan lain sebagainya.[20]

2.10 SQLite

SQLite adalah salah satu *software* yang *embedded* yang sangat populer, kombinasi SQL *interface* dan penggunaan *memory* yang sangat sedikit dengan kecepatan yang sangat cepat". *SQLite* di Android termasuk dalam Android *Runtime*, sehingga setiap versi dari Android dapat membuat *database* dengan *SQLite.*

Dalam sistem android memiliki beberapa teknik untuk melakukan penyimpanan data. Teknik yang umum digunakan adalah sebagai berikut :

1. *Shared Prefences* yaitu menyimpan data beberapa nilai (*value*) dalam bentuk *groups key* yang dikenal dengan *prefences*.
2. *Files* yaitu menyimpan data dalam bentuk *file*, dapat berupa menulis ke *file* atau membaca dari *file*.
3. *SQLite Databases* yaitu menyimpan data dalam bentuk *Databases*.
4. *Content Providers* yaitu menyimpan data dalam bentuk *content providers service*.

Tidak ada *database* yang otomatis disediakan oleh android, jika kita menggunakan SQLite, kita harus meng-*create database* sendiri, mendefinisikan tabelnya, *index* serta data. Ada tiga metode dalam membuat dan membuka *database*, yaitu :

1. *Constructor*, menyediakan representasi versi *database* dan skema *database* yang kita gunakan.
2. *OnCreate(),* menyediakan SQLite *Database object* yang kita gunakan dalam definisi tabel dan inisialisasi data.
3. *OnUpgrade(),* menyediakan fasilitas konversi *database* dari *database* versi yang lama ke *database* versi yang baru atau sebaliknya. [10]

2.11 UML (Unified Modelling Language)

Unified Modeling Language (UML) adalah bahasa pemodelan visual untuk tujuan umum yang digunakan untuk menentukan, memvisualisasi, membangun, dan mendokumentasikan *artefak* dari perangkat lunak sistem. UML digunakan untuk memahami, mendesain, mengkonfigurasi, memelihara dan mengontrol informasi tentang sistem tersebut. UML dimaksudkan untuk digunakan dengan semua metode pengembangan, tahapan-tahapan siklus hidup, domain-domain aplikasi dan media.

Spesifikasi UML tidak mendefinisikan proses standar tetapi dimaksudkan untuk menjadi berguna dengan proses pengembangan berulang. Hal ini dimaksudkan untuk mendukung proses pengembangan dengan *object oriented*. UML menyimpan informasi tentang struktur statis dan perilaku dinamis suatu sistem. Sebuah sistem dimodelkan sebagai koleksi benda-benda diskrit yang

berinteraksi untuk melakukan kerja yang pada akhirnya bermanfaat. Struktur statis mendefinisikan jenis-jenis objek penting untuk sistem dan pelaksanaannya, serta sebagai hubungan antara objek-objek. Perilaku dinamis mendefinisikan sejarah obyek dari waktu ke waktu dan komunikasi antara objek-objek untuk mencapai tujuan. UML dimaksudkan untuk menjadi tujuan *universal* bahasa pemodelan untuk sistem diskrit seperti yang terbuat dari perangkat lunak, *firmware* atau logika *digital*.[14]

Berikut adalah UML yang penulis gunakan dalam penelitian ini yaitu *use case* diagram, *sequence* diagram, *collaboration* diagram dan *class* diagram.

1. *Use Case Diagram*

Use case diagram digunakan untuk menentukan fungsionalitas perangkat lunak dari perspektif pengguna. *Use case* diagram merupakan gambaran dari semua kasus penggunaan dan bagaimana mereka terkait.. Aktor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh aktor. *Use case* digambar-kan berbentuk elips dengan nama operasi dituliskan di dalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *Use case*. [4]

2. Collaboration Diagram

Collaboration Diagram merupakan diagram yang dapat memperlihatkan bagaimana objek dalam suatu sistem bekerja satu sama lain. Diagram ini menitik beratkan pemahaman atau operasi yang dilakukan objek selama aplikasi dijalankan.[4]

3. Sequence Diagram

Sequence diagram menggambarkan komunikasi dinamis antara sejumlah objek dan untuk menunjukkan rangkaian pesan yang dikirim antar objek juga interaksi antar objek, seuatu yang terjadi pada titik tertentu dalam eksekusi sistem. *Sequence* diagram adalah gambaran tahap demi tahap yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case* diagram.[4]

4. Class Diagram

Class diagram menggambarkan pandangan struktural dari suatu sistem seperti atribut, hubungan dan asosiasi antar kelas lain. Unsur utama dari class diagram berupa kotak, setiap kotak dibagi menjadi bagian horisontal. Bagian atas berisi nama kelas, bagian tengah berisi daftar atribut dari suatu kelas dan bagian ketiga berisi operasi atau perilaku kelas. Sebuah operasi mengacu pada apa yang dapat dilakukan objek pada sebuah kelas. [4]

2.12 Alpha dan Beta Testing

Tujuan *Alpha Testing* atau pengujian alpha adalah untuk mengidentifikasi dan menghilangkan sebanyak mungkin masalah sebelum akhirnya sampai ke *user*, serta dilakukan setelah *software* selesai oleh pengembang [5]. Sedangkan Pengujian beta atau *beta testing* adalah istilah yang digunakan untuk menggambarkan proses pengujian *eksternal* di mana *software* tersebut dikirim ke kelompok atau memilih pelanggan yang menggunakannya dalam lingkungan dunia nyata. Pengujian beta biasanya dilakukan menjelang akhir dari siklus pengembangan produk dan idealnya sebagai validasi bahwa perangkat lunak siap dirilis untuk pelanggan yang sesungguhnya.[13]

2.13 Metode Black-box Testing

Menurut Ron Patton, *Black-box* testing merupakan istilah yang digunakan penguji perangkat lunak untuk menggambarkan bagaimana pendekatan pengujian mereka. Dalam pengujian *black-box*, penguji hanya tahu apa yang seharusnya dilakukan oleh *software*, namun penguji tidak dapat melihat bagaimana hal itu bisa beroperasi. Hal ini yang membuat penulis memilih menggunakan metode *Black box*, agar pengujian berfokus pada persyaratan fungsional perangkat lunak untuk melihat apakah program aplikasi menghasilkan output yang diinginkan atau tidak. [13]

2.14 Skala Likert

Skala Likert umumnya digunakan untuk mengukur sikap atau respon seseorang terhadap suatu objek. Pengungkapan sikap dengan menggunakan skala Likert sangat popular di kalangan para ahli psikologi sosial dan para peneliti. Hal

ini dikarenakan selain praktis, skala Likert yang dirancang dengan baik pada umumnya memiliki reliabilitas yang memuaskan.

Skala Likert berwujud kumpulan pertanyaan-pertanyaan sikap yang ditulis, disusun dan dianalisis sedemikian rupa sehingga respon seseorang terhadap pertanyaan tersebut dapat diberikan angka (skor) dan kemudian dapat diinterpretasikan. Skala Likert tidak terdiri dari hanya satu stimulus atau satu pernyataan saja melainkan selalu berisi banyak item (*multiple item measure*).

Sejauh mana suatu skala Likert akan berfungsi seperti yang diharapkan, yaitu mengungkapkan sikap individu atau sikap kelompok manusia dengan cermat dan akurat, banyak tergantung pada kelayakan pertanyaan-pertanyaan sikap dalam skala itu sendiri. Oleh karena itu pernyataan-pernyataan yang dibuat untuk mengukur sikap harus dirancang dengan hati-hati. Stimulus harus ditulis dan dipilih berdasarkan metode konstruksi yang benar dan skor terhadap respon seseorang harus diberikan dengan cara-cara yang tepat. [2]

Adapun rumusan untuk mendapatkan hasil dari skala *likert* adalah sebagai berikut:

$$P = \frac{S}{Skorideal}\ x\ 100$$

Keterangan:

P = Nilai presentase yang dicari

S = Jumlah frekuensi dikalikan dengan nilai skala jawaban

Skorideal = Skala tertinggi jawaban dikalikan dengan jumlah *sample*