BAB II
TINJAUAN PUSTAKA

Dalam BAB ini berisi berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi pembaca pesan *Short Message Service* (SMS) dan *Instant Messenger* berbasis *Text to Speech*.

2.1 Komunikasi

Manusia adalah makhluk hidup dan makhluk sosial yang sangat perlu untuk melakukan komunikasi, karena hanya dengan komunikasi manusia dapat mengenal satu sama lain dan saling bertukar berbagai macam informasi. Komunikasi juga bertujuan untuk keberlangsungan hidup manusia, karena apabila tanpa ada suatu komunikasi, maka manusia tidak akan bisa untuk melangsungkan kehidupannya. Misalnya dalam bidang ilmu pengetahuan dan teknologi, dimana komunikasi memiliki peranan penting dalam pencapaian tujuan organisasi.

Di dalam buku karya Onong Uchjana Effendi yang berjudul "Ilmu Komunikasi Teori dan Praktek" menjelaskan bahwa : "Istilah komunikasi atau dalam bahasa Inggris *communication* berasal dari bahasa latin atau *communication* dan bersumber dari *communis* yang berarti sama. Sama disini maksudnya adalah satu makna. Jadi, jikadua orang terlibat dalam komunikasi, maka komunikasi akan terjadi atau berlangsung selama ada kesamaan makna mengenai apa yang dikomunikasikan, yakni baik si penerima maupun si pengirim sepaham dari suatu pesan tertentu." [7]

2.2 Aplikasi

Aplikasi adalah penggunaan dalam suatu komputer,instruksi (*instructiom*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi *output* [8]. Menurut Kamus Besar Bahasa Indonesia, aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan

tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer. Program merupakan kumpulan *instruction set* yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini.

Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada *instructionset*. Program aplikasi merupakan program siap pakai yang dirancang 14 untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasi ialah program pemproses kata dan *Web Browser*. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan *app*.

Secara historis, aplikasi adalah *software* yang dikembangkan oleh sebuah perusahaan. *App* adalah *sofware* yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang parallel dengan yang terjadi antar sistem operasi yang dimunculkan.

2.3 Short Message Service (SMS)

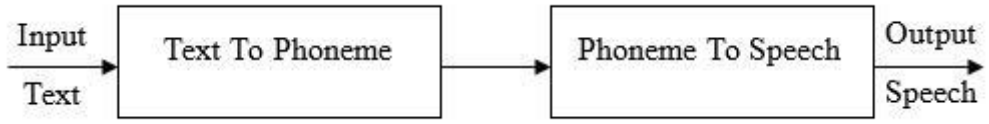
Short Message Service (SMS) adalah kemampuan untuk mengirim dan menerima pesan dalam bentuk teks dari dan kepada ponsel. Teks tersebut bisa terdiri dari kata-kata atau nomor atau kombinasi *alphanumeric*. SMS diciptakan sebagai standar pesan (*message*) oleh ETSI (*Europesan Telecommunication Standards Institute*), yang juga membuat standar GSM (*Global System for Mobile*) yang diimplementasikan oleh semua operator *Global System for Mobile* (GSM). SMS yang pertama dikirimkan pada Desember 1992 dari PC ke sebuah ponsel melalui jaringan *Global System for Mobile* (GSM) Vodafone di UK. Setiap Pesan maksimal terdiri dari 160 karakter jika menggunakan *alphabet* Latin, dan 70 karakter jika menggunakan *alphabet* nonLatin seperti huruf Arab atau China [9].

2.4 Instant Messenger

Instant Messenger (IM) atau yang biasa disebut aplikasi chatting merupakan perangkat lunak yang memfasilitasi pengiriman pesan singkat (*instant messaging*), suatu bentuk komunikasi secara langsung antara dua orang atau lebih menggunakan teks yang diketik. Teks dikirim melalui komputer yang terhubung melalui sebuah jaringan, misalnya internet. Setelah penggunaan email yang mengubah cara orang berkomunikasi dari cara konvensional untuk mengirimkan surat, teknologi pengiriman pesan singkat (instant messaging) diciptakan untuk menutupi kelemahan email yang kadang-kadang kurang cepat dan tidak waktu nyata (real-time) [10].

2.5 Text To Speech (TTS)

Sistem *Text to Speech* pada prinsipnya terdiri dari dua sub sistem[11], yaitu:



Gambar II.1 Skema *Text To Speech*

1. Bagian Konverter Teks ke Fonem (*Text to Phoneme*)

   Bagian Konverter Teks ke Fonem berfungsi untuk mengubah kalimat masukan dalam suatu bahasa tertentu yang berbentuk teks menjadi rangkaian kode-kode bunyi yang biasanya direpresentasikan dengan kode fonem, durasi serta *pitch*-nya. Bagian ini bersifat sangat *language dependant*. Untuk suatu bahasa baru, bagian ini harus dikembangkansecara lengkap khusus untuk bahasa tersebut.

2. Bagian Konverter Fonem ke Ucapan (*Phoneme to Speech*)

   Bagian Konverter Fonem ke Ucapan menerima masukan berupa kode-kode fonem serta *pitch* dan durasi yang dihasilkan oleh bagian sebelumnya. Berdasarkan kode-kode tersebut, bagian Konverter Fonem ke Ucapan akan menghasilkan bunyi atau sinyal ucapan yang sesuai dengan kalimat yang ingin diucapkan. Ada beberapa alternatif teknik

yang dapat digunakan untuk implementasi bagian ini. Dua teknik yang banyak digunakan adalah *formant synthesizer* serta *diphone concatenation*. Sub sistem ini harus memiliki pustaka setiap unit ucapan dari suatu bahasa. *Formant synthesizer* bekerja berdasarkan suatu model matematis yang akan melakukan komputasi untuk menghasilkan sinyal ucapan yang diinginkan. *Synthesizer* jenis ini telah lama digunakan pada berbagai aplikasi. Walaupun dapat menghasilkan ucapan dengan tingkat kemudahan interpretasi yang baik, *synthesizer* ini tidak dapat menghasilkan ucapan dengan tingkat kealamian yang tinggi.

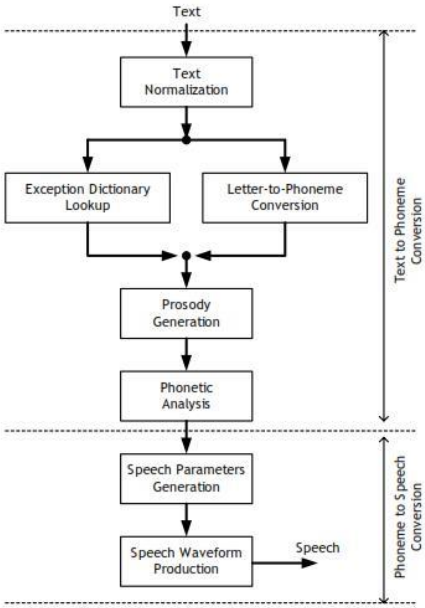
Synthesizer yang menggunakan teknik *diphone concatenation* bekerja dengan cara menggabung-gabungkan segmen-segmen bunyi yang telah direkam sebelumnya. Setiap segmen berupa *diphone* (gabungan dua buah fonem). *Synthesizer* jenis ini dapat menghasilkan bunyi ucapan dengan tingkat kealamian (*naturalness*) yang tinggi.

Struktur sistem seperti di atas pada prinsipnya merupakan konfigurasi tipikal yang digunakan pada berbagai sistem *Text to Speech* berbagai bahasa. Namun demikian, pada setiap sub-sistem terdapat sifat-sifat serta proses-proses yang sangat spesifik dan sangat tergantung dari bahasanya.

Konversi dari teks ke fonem sangat dipengaruhi oleh aturan-aturan yang berlaku dalam suatu bahasa. Pada prinsipnya proses ini melakukan konversi dari simbol-simbol tekstual menjadi simbol-simbol fonetik yang merepresentasikan unit bunyi terkecil dalam suatu bahasa. Setiap bahasa memiliki aturan cara pembacaan dan cara pengucapan teks yang sangat spesifik. Hal ini menyebabkan implementasi unit konverter teks ke fonem menjadi sangat spesifik terhadap suatu bahasa. Untuk mendapatkan ucapan yang lebih alami, ucapan yang dihasilkan harus memiliki intonasi (*prosody*). Secara kuantisasi, prosodi adalah perubahan nilai *pitch* (frekuensi dasar) selama pengucapan kalimat dilakukan atau pitch sebagai fungsi waktu. Pada prakteknya, informasi pembentuk prosodi berupa data-data *pitch* serta durasi pengucapannya

untuk setiap fonem yang dibangkitkan. Nilai-nilai yang dihasilkan diperoleh dari suatu model prosodi. Prosodi bersifat sangat spesifik untuk setiap bahasa, sehingga model yang diperlukan untuk membangkitkan data-data prosodi menjadi sangat spesifik juga untuk suatu bahasa. Beberapa model umum prosodi pernah dikembangkan, tetapi untuk digunakan pada suatu bahasa masih perlu banyak penyesuaian yang harus dilakukan.

Konverter fonem ke ucapan berfungsi untuk membangkitkan sinyal ucapan berdasarkan kode-kode fonem yang dihasilkan dari proses sebelumnya. Sub sistem ini harus memiliki pustaka setiap unit ucapan dari suatu bahasa. Pada sistem yang menggunakan teknik *diphone concatenation*, sistem harus didukung oleh suatu *diphone database* yang berisi rekaman segmen-segmen ucapan yang berupa *diphone*. Ucapan dalam suatu bahasa dibentuk dari satu set bunyi yang mungkin berbeda untuk setiap bahasa, oleh karena itu setiap bahasa harus dilengkapi dengan diphone database yang berbeda. Tahapan-tahapan utama konversi dari teks menjadi ucapan dapat dinyatakan dengan *diagram* seperti terlihat pada gambar II.2.



Gambar II.2 Urutan Proses Konversi dari Teks ke Ucapan

2.6 Basis Data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk penyimpanan data agar dapat diakses dengan mudah dan cepat.

Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun *Database Management System* (DBMS) [12]. Kebutuhan basis data dalam sistem informasi meliputi memasukan, menyimpan, dan mengambil data serta membuat laporan berdasarkan data yang telah disimpan.

2.6.1 Database Management System (DBMS)

DBMS (*Database Management System*) atau dalam bahasa indonesia sering disebut sebagai Sistem Manajemen Data adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal dapat menyediakan fasilitas untuk mengelola akses data, mampu menangani integrasi data, dan mampu menangani *backup* data.

DBMS sudah mulai berkembang sejak tahun 1960an. Kemudian sekitar tahun 1970an mulai berkembang teknologi *Relational* DBMS yaitu DMBS berbasis relasional model. Relasional model pertama kali dikembangkan oleh Edgar J. Codd pada tahun 1970. Secara sederhana relasional model dapat dipahami sebagai suatu model yang memandang data sebagai sekumpulan tabel yang saling terkait.

Pada tahun 1980an mulai berkembang *Object Oriented* DBMS (OODBMS). OODBMS berkembang seiring dengan perkembangan teknologi pemrograman berorientasi objek. Secara umum dapat diartikan bahwa OODBMS merupakan DBMS yang memandang data sebagai suatu objek.

Terdapat beberapa macam DBMS versi komersial yang paling sering digunakan di dunia saat ini, diantaranya Oracle, Microsoft SQL Server, IBM DB2, dan Microsoft Access. Sedangkan DBMS versi *open source* yang cukup berkembang dan sering digunakan saat ini adalah diantaranya MySQL, PostgreSQL,

Firebird, dan SQLite. Hampir semua DBMS mengadopsi SQL sebagai bahasa untuk mengelola data pada DBMS.

2.6.2 Structured Query Language (SQL)

SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada DBMS. Meskipun SQL diadopsi dan diacu sebagai bahasa standar oleh hampir sebagian besar RDBMS yang beredar saat ini, tetapi tidak semua standar yang tercantum dalam QSL diimplementasikan oleh seluruh DBMS tersebut. Sehingga terkadang terdapat perbedaan prilaku (hasil yang ditampilkan) oleh DBMS yang berbeda padahal *query* yang dimasukan sama.

2.7 Skala Likert

Skala likert merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat dan persepsi seseorang tentang suatu gejala atau fenomena tertentu. Instrumen penelitian yang menggunakan skala likert dapat dibuat dalam bentuk *checklist* ataupun pilihan ganda [13]. Seperti menentukan lokasi kedudukan seseorang dalam suatu kontinium sikap terhadap objek sikap, mulai dari sangat negative sampai dengan positif. Untuk setiap satu pernyataan responden memilih salah satu tanggapan berupa SS = Sangat Setuju, S = Setuju, Kurang Setuju=KS, TS = Tidak Setuju, STS = Sangat Tidak Setuju. Tanggapan responden dikodekan kedalam bilangan 1 sampai dengan 5. Pemberian bilangan ini bergantung kepada bentuk pernyataan yang dijawab oleh responden.

Ada dua macam bentuk pernyataan dalam skala likert, pertama adalah pernyataan yang diharapkan untuk disetujui oleh responden, disebut bentuk positif. Yang kedua adalah pernyataan yang diharapkan untuk tidak disetujui oleh responden, bentuk negatif. Biasanya kuesioner berisikan pernyataan positif dan negatif agar responden berfikir dulu sebelum memberi jawaban.

Contoh kuesioner sikap maysarakat menggunakan skala *likert* mengenai aplikasi *mobile* android yang akan dibangun. Petunjung menjawab:

A = Sangat Setuju,

B = Setuju,

C = Ragu,

D = Tidak Setuju,

E = Sangat Tidak Setuju

Dengan pernyataan :

1. Dengan adanya aplikasi mobile Android yang dapat mengurangi terjadinya kesalahan melaporkan tempat kejadian kecelakaan dan lebih tergambarnya keadaan korban dari masyarakat
2. Mempermudah petugas pelayanan kesehatan mobilitas Dinas Kesehatan Kota Bandung menentukan puskesmas atau rumah sakit mana yang siap dan paling dekat dengan korban kecelakaan

Dari contoh kuesioner diatas didapat skor untuk skala *likert* sebagai berikut :

Tabel II.1 Contoh Skor Skala *Likert*

Pilihan Jawaban	Pernyataan	
	Positif	Negatif
Sangat Setuju	5	1
Setuju	4	2
Ragu-Ragu	3	3
Tidak Setuju	2	4
Sangan Tidak Setuju	1	5

Pada bentuk positif sangat setuju memperoleh sekor tinggi dan sangat tidak setuju memperoleh sekor rendah. Pada bentuk negatif, sangat setuju memperoleh sekor terendah dan sangat tidak setuju memperoleh sekor tinggi. Skala Likert digunakan untuk mengukur penilaian pengguna kendaraa bermotor terhadap aplikasi pembaca pesan *Short Message Service* dan *Intant Messenger*. Untuk mencari nilai presentase dari masing-masing jawaban kuesioner digunakan rumus skala likert sebagai berikut :

$$P = \frac{S}{Skor\ Ideal}\ x\ 100\%$$

Berikut ini adalah keterangan dari rumus skala likert mencari nilai presentase dari jawaban kuesioner :

P = Nilai Presentasi yang dicari

S = Jumlah frekuensi dilakukan dengan skor yang ditetapkan jawaban

Skor Ideal = Nilai tertinggi dikalikan dengan jumlah sampel. Nilai tertingginya yaitu 5.

Skala Likert digunakan untuk mengukur penilaian pengguna kendaraa bermotor terhadap aplikasi pembaca pesan *Short Message Service* dan *Intant Messenger*. Skala Likert ini akan digunakan pada Bab 4 implementasi dan pengujian.

2.8 Object Oriented Programming

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah [14].

2.8.1 Objek

Objek merupakan pemodelan sistem yang lebih natural dibandingkan dengan prosedur. Dalam dunia nyata, objek dapat dilihat di mana saja. Setiap Objek mempunyai dua karakteristik, yaitu keadaan (*state*) dan tingkah laku (*behavior*). Keadaan (*state*) digunakan untuk menyimpan informasi Objek. keadaan (*state*) disebut juga dengan attribute atau *field*. misalnya kucing sebagai objek bisa mempunyai state lapar/kenyang. haus/tidak haus, nilai berat badan, warna bulu, dan lain-lain. Sedangkan tingkah laku Objek digunakan untuk menentukan keda apa saja yang dapat dilakukan Objek tersebut. kerja disebut juga dengan metode (*method*). metode Objek, misalnya makan. tidur, dan berak. metode yang dilakukan dapat berpengaruh terhadap Objek itu sendiri atau Objek lainnya. misalnya metode makan pada Objek kucing membuat kucing tersebut kenyang dan Objek tikus mati[15].

2.8.2 Kelas (Class)

Class merupakan cetak biru atau template objek. Jadi. *class* bukanlah Objek real. namun merupakan konsep Objek. Misalnya dalam *game* yang Anda buat

terdapat dua ekor kucing diberi nama Katty dan Ronald. Maka kucing adalah *class*, sedangkan Katty dan Ronald merupakan objek dengan tipe kucing. Dalam pendefinisian *class*. kita menentukan *field* dan metode yang akan digunakan. Misalnya dalam *class* kucing kita menentukan dua buah *field*: nama dan berat badan. Dalam deklarasi *class*, nama dan berat badan kucing tidak ditentukan karena setiap kucing bisa memiliki nama dan berat badan yang berbeda-beda. Kedua *field* tersebut baru ditentukan saat kita membuat objek kucing. Misalnya dibuat objek kucing bernama Katty berat badan 5 kg, serta seekor kucing lagi dengan nama Ronald berat badan 7 kg[15].

2.8.3 Inheritance

Setiap objek dapat diklasifikasikan secara bertingkat untuk memudahkan pernahaman dan pembuatan objek itu sendiri. Misalnya dari pemahaman sistem, diketahui bahwa kucing dan anjing mempunyai kesamaan, yaitu sama-sama hewan dan merupakan mamalia. Maka kita dapat membuat tingkatan *class* tersebut.

Inheritance atau pewarisan menggambarkan penurunan sifat dari *class*. Dengan klasifikasi di atas, anjing dan kucing mewarisi sifat dari *class* di atas (*superclass*), yaitu mamalia. Inheritance sangat fleksibel dan dapat diterapkan pada berbagai sistem. misalnya untuk program aplikasi penjualan, perpustakaan, dan sebagainya[15].

2.8.4 Polymorphism

Polimorfisme merupakan kondisi di mana sesuatu mempunyai beberapa bentuk. Dalam pemrograman OOP, penerapan polimorfisme dilakukan menggunakan nama sama, namun mempunyai implementasi berbeda. Mari kita buat contoh. Dengan menerapkan polimorfisme. metode berkembang biak pada Objek dengan tipe kucing dan Objek dengan tipe burung bisa berbeda. Anjing berkembang biak dengan melahirkan, sedangkan burung dengan bertelur.

Contoh lain, objek mahasiswa dapat melakukan metode mencuci yang berbeda.beda tergantung Objek yang dicuci. Metode mencuci motor, mencuci piring, dan mencuci baju tentu dilakukan dengan cara berbeda, *Polimorfisme* pada

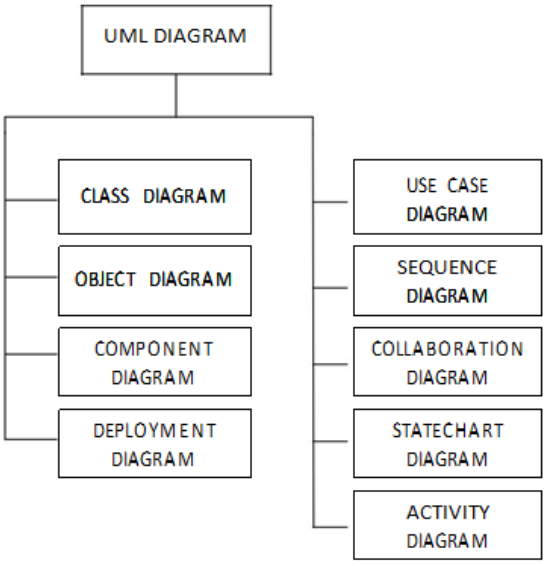
OOP digunakan untuk memudahkan pemrograman karena lebih natural. Dengan polimorfisme. kita tidak harus menggu- nakan nama yang berbeda untuk metode yang mirip secara bahasa. Jadi, cukup menggunakan metode mencuci saja. tidak harus diberi nama mencuci_baju, mencuci_piring, dan mencuci_motor[15].

2.9 UML (Unified Modelling Language)

UML merupakan kependekan dari *Unified Modeling Language* yaitu sebuah bahasa pemodelan grafis yang digunakan sebagai standar untuk memodelkan sistem dengan metodologi pemodelan berorientasi objek. UML distandarkan oleh *Object Management Group* (OMG). UML pertarnakali dipopulerkan oleh Grady Booch dan James Rumbaugh pada akhir tahun 1994. Kemudian Ivar Jacobson yang merupakan pimpinan dari *object oriented development* bergabung. Pada tahun 1996 UML mulai digunakan sebagai tools untuk memodelkan sistem pada IBM dan i-logix[15].

2.9.1 UML *Diagram*

Diagram-diagram yang terdapat pada UML sangat banyak, berikut ini beberapa *diagram* yang sering digunakan dalam pengembangan sistem dapat dilihat pada gambar II.3.



Gambar II.3 UML *Diagram*

2.9.2 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) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut(metoda/fungsi) [15].

Class diagram menggambarkan struktur dan deskripsi *class*, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu Nama (dan *stereotype*), Atribut dan 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.

Hubungan Antar *Class* :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah navigability menunjukkan arah *query* antar *class*. Lambang :

2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”) atau biasa disebut relasi mempunyai sebuah. Lambang :

3. *Composition*, yaitu sebuah kelas tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi composition. Lambang

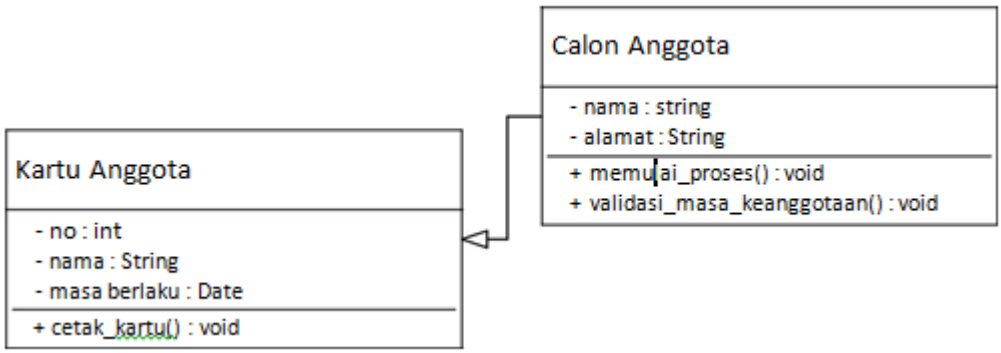
4. *Dependency*, yaitu hubungan yang menunjukan operasi pada suatu kelas yang menggunakan *class* yang lain. Lambang :



5. Generalisasi / 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 diwaris.



Berikut Contoh *Class Diagram* dapat dilihat pada Gambar II.4 [14].



Gambar II.4 Contoh *Class Diagram*

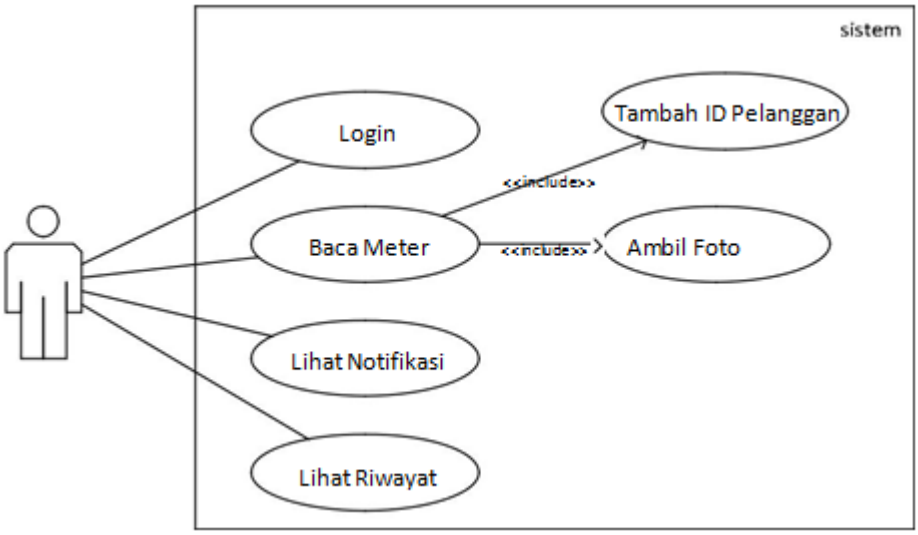
2.9.3 User Case Diagram

Use Case Diagram yaitu *diagram* yang digunakan untuk menggambarkan hubungan antara sistem dengan aktor. *Diagram* ini hanya menggambarkan secara global. Karena *use case diagram* hanya menggambarkan sistem secara global, maka elemen-elemen yang digunakan pun sangat sedikit, berikut ini elemen-elemen yang digunakan pada *use case diagram*[15] :

1. Sistem, merupakan batasan-batasan proses yang sudah kita deskripsikan dałam sebuah sistem.
2. Aktor, elemen yang menjadi pemicu sistem. Aktor bisa berupa orang, mesin ataupun sistem lain Yang berinteraksi dengan use case.
3. *Use Case*, potongan proses Yang merupakan bagian dari sistem.
4. *Association*, menggambarkan interaksi antara *use case* dan aktor.

5. *Depedency*, menggambarkan relasi (*relationship*) antara dua *use case*. Ada 2 (dua) tipe dari *dependency* yaitu, *include* dan *extends*. Include merupakan tipe dari *dependency* yang menghubungkan dua *use case* dimana, satu *use case* membutuhkan *use case* yang satunya sedangkan *extends* adalah tipe dari *dependency* yang menghubungkan dua *use case* dimana satu use case terkadang akan memanggil *use case* yang satunya, tergantung pada kondisi.
6. *Generalization*, menggambarkan pewarisan antara dua aktor atau use case dimana salah satu aktor atau use case mewarisi properties ke aktor atau *use case* yang satunya.

Berikut Contoh *Use Case Diagram* dapat dilihat pada Gambar II.5 [15].



Gambar II.5 Contoh Use Case *Diagram*

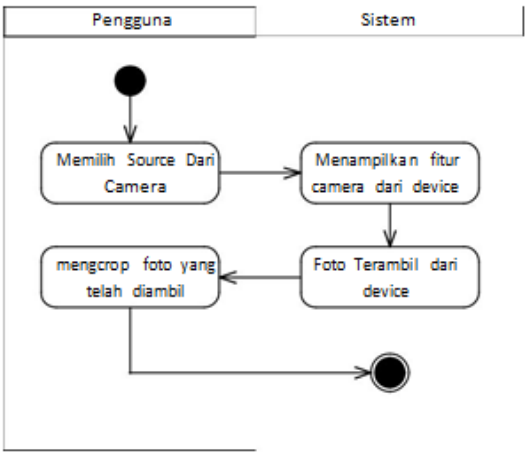
2.9.4 Activity Diagram

Activity Diagram yaitu *diagram* yang digunakan untuk menggambarkan alur kerja (aktivitas) pada use case (proses), logika, proses bisnis dan hubungan antara aktor dengan alur-alur kerja use case. Jika sudah terbiasa dengan flowchart, maka tidak akan merasa kesulitan dalam mempelajari *activity diagram*, karena *activity diagram* identik dengan flowchart, hanya saja ada beberapa notasi

tambahan yang digunakan untuk kasus- kasus tertentu. Berikut ini di jelaskan elemen-elemen dari *activity diagram*[15].

1. *Activity*, yaitu elemen yang digunakan untuk menggambarkan aktivitas.
2. *Transitions*, yaitu elemen yang digunakan untuk menggambarkan transisi dari elemen yang satu ke elemen yang lainnya
3. *Decisions*, yaitu elemen yang digunakan untuk percabangan logika. Elemen ini sering kita jumpai pada *flowchart* terutama *flowchart* yang digunakan untuk menggambarkan sebuah algoritma.
4. *Merge Point*, yaitu elemen yang digunakan untuk menggabungkan percabangan proses. Elemen ini merupakan kebalikan dari elemen *decisions*, dimana jika *decisions* digunakan untuk percabangan, sedangkan *merge point* digunakan sebagai penggabungan dari percabangan.
5. *Start Point*, yaitu elemen yang digunakan untuk memulai *activity diagram*.
6. *End Point*, yaitu elemen yang digunakan untuk mengakhiri *activity diagram*.
7. *Swimlines*, yaitu elemen yang digunakan untuk memisahkan antara aktor dan sistem ataupun antara aktor yang satu dengan aktor yang lain atau antara sistem yang satu dengan sistem yang lain.

Berikut Contoh *Activity Diagram* dapat dilihat pada Gambar II.6.



Gambar II.6 Contoh *Activity Diagram*

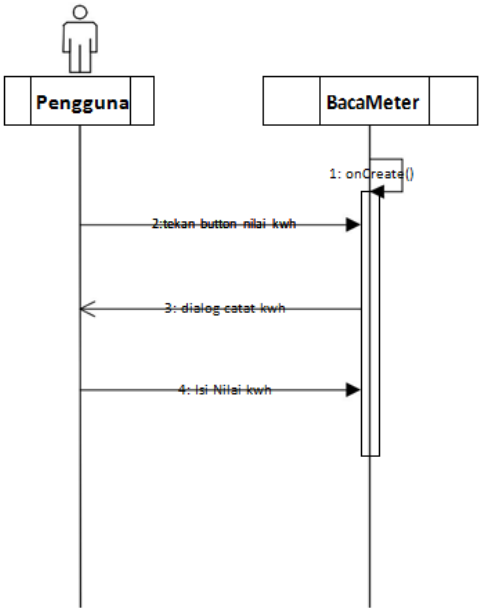
2.9.5 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Sequence *diagram* terdiri atas 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* [15].

Berikut Contoh *sequence diagram* dapat dilihat pada gambar 2.9.



Gambar II.7 Contoh *Sequence Diagram*

Pengetahuan tentang *Object Oriented Programming* (OOP) ini diperlukan karena metode analisis yang digunakan dalam pembangunan aplikasi pertolongan

kecelakaan lalu lintas di kota bandung adalah metode *Object Oriented Programming* (OOP) untuk sistem *mobile* android.

2.10 Google Text To Speech

Penggunaan *Google Text To Speech API* memungkinkan *developer* untuk mengkonversi teks kedalam ucapan (suara). Untuk dapat menggunakan fitur tersebut, *developer* android dapat menggunakan *interface* dan *class* yang telah disediakan Google *API* (*package android.speech.tts)* [16].

1. *Interfaces*

   Terdapat beberapa *interface* yang terdapat pada *package android.speech.tts*, diantaranya:

   a. *SynthesisCallback*
   b. *TextToSpeech.OnInitListener*
   c. *TextToSpeech.OnUtteranceCompletedListener*

2. *Classes*

   Terdapat beberapa *class* yang terdapat pada *package android.speech.tts*, diantaranya:

   a. *SynthesisRequest*, berisi data yang dibutuhkan oleh mesin untuk mensintesis ucapan
   b. *TextToSpeech*, mensintesis ucapan dari teks untuk diputar langsung atau untuk membuat file suara
   c. *TextToSpeech.*Engine, konstanta dan nama parameter untuk mengendalikan *text-to-speech*
   d. *TextToSpeech.EngineInfo*, informasi tentang terpasangnya mesin *text-to-speech*
   e. *TextToSpeechService*, kelas abstrak dasar untuk implementasi mesin TTS
   f. *UtteranceProgressListener*, *listener* untuk kejadian yang berkaitan dengan kemajuan suatu ucapan melalui antrian sintesis.

Untuk dapat memanfaatkan fitur *Google Text To Speech API* pada program diperlukan beberapa langkah, diantaranya mengimpor *class* yang berhubungan dengan *text to speech*, yaitu android.speech.tts.TextToSpeech; Untuk menggunakan kelas *TextToSpeech*, perlu menginisialisasi sebuah objek dari *class TextToSpeech* dan menentukan *initListener*.

```
private EditText write;
tobj = new TextToSpeech(getApplicationContext(),
new TextToSpeech.OnInitListener() {
@Override
public void onInit(int status) {
}
```

Pada *listener* ini, harus menentukan properti untuk objek *text to speech*, seperti bahasa, *pitch* dan lain sebagainya. Bahasa dapat diatur dengan memanggil *method setLanguage()*, yaitu: *ttobj.setLanguage(Locale.UK);*.

Method setLanguage mengambil objek lokal sebagai parameter. Berikut contoh daftar dari beberapa objek lokal (tergantung masing-amsing *device*): US, CANADA_FRENCH, GERMANY, ITALY, JAPAN, CHINA. Setelah mengatur bahasa, dapat memanggil method speak, yaitu: *ttobj.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null);*.

Terlepas dari *method speak*, ada beberapa *method* lain yang disediakan pada *class text to speech* yaitu *addSpeech(String text, String filename), getLanguage(), isSpeaking(), setPitch(float pitch), setSpeechRate(float speechRate), shutdown(),* dan *stop()*.

Selain menggunakan *class android.speech.tts.TextToSpeech;* yang disediakan oleh Google, developer dapat menggunakan *HTTP GET (REST) request*: http://translate.google.com/translate_tts?ie=UTF-8&tl=en&q=text untuk mengkonversi teks menjadi suara. Dimana *tl* merupakan bahasa yang digunakan, misalnya, *ja* untuk bahasa *Japanese* atau *en* untuk bahasa *English*, sedangkan *q* merupakan *query string* (teks) yang akan dikonversi menjadi suara (ucapan). Namun dalam penggunaannya, metode ini hanya terbatas pada 100 karakter.

2.11 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linuk yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi baru. Android merupakan generasi baru platform mobile, platform yang memberikan pengembangan untuk melakukan pengembangan sesuai dengan yang diharapkan. Pada saat perilisan perdana android, 5 November 2007, android bersama open handset alliance menyatakan mendukung pengembangan open source pada perangkat lunak mobile. Di lain pihak, google merilis kode-kode android dibawah lisensi apache, sebuah lisensi perangkat lunak dan open platform perangkat seluler [17].

2.11.1 Android SDK (Software Development Kit)

Android SDK adalah *tools API* (*Application Programming Interface*) yang diperlukan 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. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan *API* untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan [17].

2.11.2 ADT (Android Development Tools)

Android Develpoment Tools adalah *plugin* yang di desain untuk IDE Eclipse yang memberikan kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk Eclipse akan 23 memudahkan dalam membuat aplikasi *project* Android, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya [17].

2.11.3 Arsitektur Android

Secara garis besar Arsitektur Android dapat di jelaskan dan di gambarkan sebagai berikut [17]:

1. *Application* dan *Widgets*

   Application dan *Widgets* adalah layer dimana *user* berhubungan dengan aplikasi saja, dimana biasanya user men-*download* aplikasi, melakukan instalasi dan menjalankan aplikasi.

2. *Application Frameworks*

   Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau member kemampuan untuk membangun aplikasi yang inovatif

3. *Libraries*

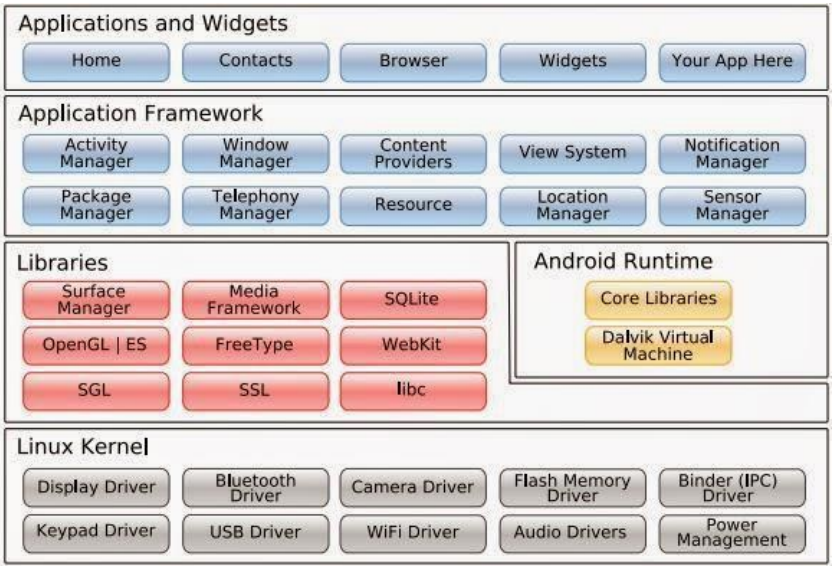
   Libraries adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

4. Android *Runtime*

   Layer yang membuat aplikasi Android dapat djalankan dimana dalam prosesnya menggunakan Implementasi Linux.

5. Linux Kernel

   Linux Kernel adalah *layer* dimana inti dari *operating system* dari Android itu berada.



Gambar II.8 Arsitektur Android

2.11.4 Fundamental Aplikasi

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data *file resource* yang dibutuhkan oleh aplikasi, dimana prosesnya di-*package* oleh *tools* yang dinamakan "*apt tools*" ke dalam paket Android sehingga menghasilkan *file* dengan ekstensi apk. Ada empat jenis komponen pada aplikasi Android yaitu :

1. *Activities*
2. *Service*
3. *Broadcast Receiver*
4. *Content Provider*

2.11.5 Versi Android

1. Android Versi 1.1

   Android pertama yaitu versi 1.1 di rilis pada 9 Maret 2009 oleh Google. Android versi ini dilengkapi dengan fitur yang disupport oleh Google *Mail Service* dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. Android Versi 1.5 (*Cupcake*)

   Android *Cupcake* atau Android Versi 1.5, Versi kedua android ini mendukung berbagai fitur yang lebih lengkap diantaranya adalah upload Video ke Youtube atau Gambar ke Picasa langsung dari telepon selular. Bluetooth A2DP yang sudah terintegrasi.

3. Android Versi 1.6 (*Donut*)

   Satu lagi nama unik os android yaitu android donut. nama lain dari Versi Android 1.6. Android Donut lebih mengembangkan aplikasi - aplikasi standart pada hp diantaranya Proses searching yang lebih baik, Fitur pada galery yang lebih "*user friendly*", Mendukung Resolusi Layar WVGA, Peningkatan Android *Market* dan Aplikasi , juga mampu Upgradeable ke versi 2.1 (Eclair).

4. Android Versi 2.0/2.1 (*Éclair*)

   Versi Android selanjutnya adalah versi 2.0 / 2.1 atau yang disebut juga Android Eclair yang merupakan urutan generasi ke 4 dari banyak versi Android yang ada, terdapat penambahan google map di versi ini., juga beberapa penambahan fitur yang lebih canggih dan bagus seperti Kamera 3,2 Megapixel yang didukung oleh "*flash*", Daftar kontak baru yang elegan, HTML5 telah terdukung pada perubahan *UI* dengan *browser* baru dan terakhir Bluetooth 2.1 dengan kecepatan *trasfer file* yang lebih cepat.

5. Android Versi 2.2 (*Froyo*)

   Android versi 2.2 *Froyo* pertamakali diluncurkan pada tahun 2010. android versi ini sudah dilengkapi dengan beragam fitur baru seperti *adobe flash* dan dikalim memiliki kecepatan 3 sampai 5 kali lebih cepat dari versi terdahulunya. sedangkan fitur terbarunya adalah Mampu merekam video dengan *HD Quality*, Bisa meletakkan aplikasi di dalam *MMC/SD Card*, Bisa untuk dijadikan *Hotspot*, Performa yang meningkat, Kemampuan *auto update* dalam Android *Market.*

6. Android Versi 2.3 (*Gingerbread*)

   Android versi 2.3 Gingerbread mampu meningkatkan kinerja dan peforma berbagai macam aplikasi atau fitur fitur yang umum dalam device android seperti *game*, audio, video, kamera dll. Fungsi dan penerapan copy paste juga telah dioptimalkan. Android gingerbread juga sudang mendukung *User interface* hemat energi, *Keyboard virtual* dengan *word selection, Power Management, App control*, Dukungan jumlah kamera yang lebih dari satu.

7. Android Versi 3.0 (*Honeycomb*)

   Android versi 3.0/3.1 Honeycomb adalah salah satu versi android yang dikhususkan bagi pengguna tablet, os android versi ini tidak bisa digunakan pada ponsel/hp anda. fitur fiturnya juga telah disesuaikan bagi pengguna tablet. sistem ini didesain khusus dengan kecanggihannya membuat tablet anda berjalan dengan kinerja maksimal.

8. Android Versi 4.0 (*Ice Cream Sandwich*)
9. Android versi 4.0 *Ice Cream Sandwich* atau disingkat ICS adalah android pertama yang mempunyai fitur baru membuka kunci dengan pengenalan wajah.android ini juga dikalim mampu menghasilan interface yang lebih halus dan bersih. terdapat juga beberapa penambahan font baru pada android versi ICS ini.
10. Android Versi 4.1 (*Jelly Bean*)

    Jelly Bean-Android versi 4.1 yang diluncurkan pada acara Google I/O membawa fitur fitur baru yang menawan, beberapa fitur yang diperbaharui dalam sistem operasi ini antara lain, pencarian dengan menggunakan *Voice Search* yang lebih cepat, informasi cuaca, lalu lintas, hasil pertandingan olahraga yang cepat dan tepat, selain itu versi 4.1 ini juga mempunyai fitur keyboard virtual yang lebih baik. Permasalahan umum yang sering ditemui pengguna Android adalah baterai, namun Baterai dalam sistem Android *Jelly Bean* versi 4.1 ini diklaim cukup hemat.
11. Android Versi 4.4 (*KitKat*)

    Google telah mengeluarkan versi tterbarunya yaitu android kitkat, namun belum banyak user yang sudah merasakannya, masih jarang ponsel/tablet yang mendukung versi android ini dikarenakan peluncurannya masih baru. fitur fitur barunya antara lain Fitur SMS yang terintegrasi langsung kedalam Aplikasi Google Hangouts. Terdapat fasilitas Could Printing, dimana pengguna dapat Printing secara nirkabel / mengirim perintah ke Laptop / PC yang terhubung dengan printer.Desain ikon dan tema yang lebih unik dan realistik.
12. Android Versi 5.0 (*Lollipop*)

    Versi ini memfokuskan pada desain dan performa, sehingga lebih nyaman digunakan, ditambah pula kapasitas baterai yang lebih tinggi daripada versi sebelumnya. OS ini dinilai lebih baik apabila digunakan untuk gaming, dan aplikasi-aplikasi yang memerlukan resolusi tinggi.

13. Android Versi 6.0 (*Marshmallow*)

    Versi *marshmallow* (6.0) mejadi sorotan utama dari OS paling baru ini adalah *native finger scanner support* serta *indivdual app permission. Native finger scanner support* akan memberikan dukungan kepada vendor smartphone untuk menampilkan fiture finger scanner pada produk gadget milik mereka dengan bantuan OS teranyar ini. *Indivdual app* permission pada OS ini akan memberikan akses khusus pada user untuk menentukan apakah aplikasi tersebut mendapat izin dari pengguna gadget. Dari fitur-fiur yang disediakan dari OS seri M ini dinilai memiliki tingakt privasi yang sangat tinggi.

Alasan dari pembahasan tentang android ini karena aplikasi pembaca pesan *Short Message Service (SMS)* dan *Instant Messanging* berbasis *text to speech* untuk masyarakat menggunakan *mobile* android oleh karena itu perlu adanya pengetahuan tentang android seperti apa saja yang perlu diperhatikan dan disiapkan dalam membangun aplikasi android.

2.12 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan *IntelliJ* IDEA. Selain merupakan *editor* kode *IntelliJ* dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya[18]:

1. Sistem pembuatan berbasis *Gradle* yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh

6. Alat penguji dan kerangka kerja yang ekstensif
7. Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK

   Dukungan bawaan untuk Google *Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine.*



Gambar II.9 Logo Android Studio

2.12.1 Struktur Proyek

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

1. Modul aplikasi Android
2. Modul perpustakaan
3. Modul Google *App Engine*

Secara *default*, Android Studio menampilkan *file* proyek Anda dalam tampilan proyek Android. Tampilan ini diatur menurut modul untuk memberi akses cepat ke *file* sumber kunci proyek Anda. Semua *file* versi terlihat di bagian atas di bawah ***Gradle Scripts*** dan masing-masing modul aplikasi berisi folder berikut:

1. *manifests*: Berisi file AndroidManifest.xml.
2. *java*: Berisi file kode sumber Java, termasuk kode pengujian JUnit.
3. *res*: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar bitmap.

Struktur proyek Android pada disk berbeda dari representasi rata ini. Untuk melihat struktur *file* sebenarnya dari proyek ini, pilih *Project* dari menu tarik turun. Anda juga bisa menyesuaikan tampilan *file* proyek untuk berfokus pada aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih tampilan *Problems* dari tampilan proyek Anda akan menampilkan tautan ke *file* sumber yang berisi kesalahan pengkodean dan sintaks yang dikenal, misalnya tag penutup elemen XML tidak ada dalam *file* tata letak.

Alasan dari pembahasan tentang android Studio ini karena aplikasi pertolongan kecelakaan lalu lintas di kota bandung menggunakan mobile android untuk masyarakat sehingga dibutuhkan aplikasi untuk pengembangan android yaitu android studio maka perlu adanya pengetahuan tentang android studio.

2.12.2 SQLite

Android memiliki fasilitas untuk membuat *database* yang dikenal dengan SQLite. SQLite adalah salah satu *software* yang bersifat *embedded* yang sangat populer. SQLite memiliki kelebihan yaitu menggunakan kombinasi SQL *interface* dan penggunaan *memory* yang sangat sedikit dengan kecepatan yang sangat cepat. SQLite di android termasuk dalam Android *runtime*, sehingga setiap versi android dapat membuat *database* dengan SQLite.

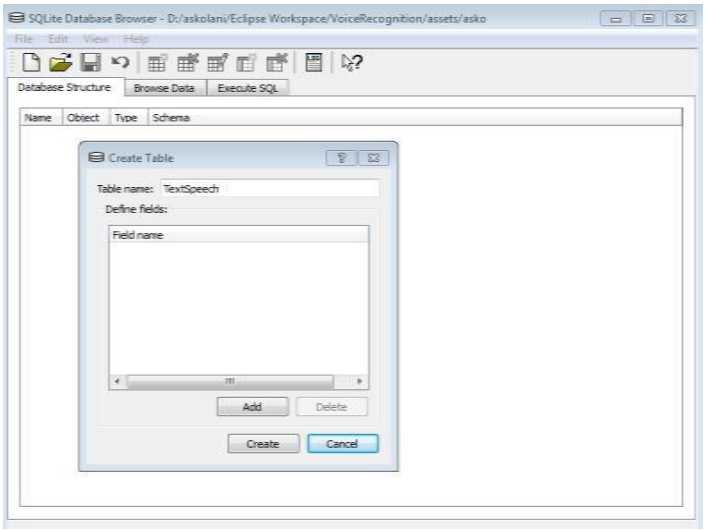
Untuk membuat dan membuka *database* pada android yang paling baik adalah menggunakan *libraries* Importandroid.database.sqlite.SQLiteOpenHelper yang menyediakan tiga metode yaitu:

1. *Constructor*, menyediakan representasi versi dari *database* dan skema *database* yang digunakan:
2. *onCreate(),* menyediakan SQLiteDatabase *object* yang digunakan dalam definisi tabel dan inisialisasi data
3. *onUpgrade(),* menyediakan fasilitas konversi *database* dari *database* versi lama ke *database* versi baru atau sebaliknya.
4. *Serverless*, SQLite tidak memerlukan proses pada *server* atau sistem untuk menjalankannya, melainkan hanya sebuah *file* yang diakses oleh *library* SQLite.

5. *Cross Platform*, semua instan *database* berada dalam sebuah *file* yang *cross-platform*, tidak memerlukan administrasi.
6. *Zero Configuration*, Tidak ada *server* berarti tidak perlu setup, membuat sebuah *database* instan adalah semudah anda membuat *file* biasa.
7. *Small Runtime Footprint*, untuk membangun *database* SQLite hanya membutuhkan kurang dari satu *megabyte library* (kode program) dan hanya membutuhkan beberapa *megabyte memory*, bahkan dengan beberapa *adjustment* baik ukuran *library* maupun memory dapat diperkecil.
8. *Self-Contained*, sebuah *library* mengandung keseluruhan dari system *database*, yang langsung terintegrasi pada sebuah aplikasi program.
9. *Ransactional*, SQLite *transaction* memperbolehkan aksi penyimpanan melalui beberapa proses *thread*.
10. *Full Featured*, SQLite mensupport hampir sebagian besar standar SQL92 (SQL2).
11. *Highly Reliable*, Tim pengembang SQLite mengembangkan melalui kode program yang sangat serius serta telah melewati proses *testing*.

Eclipse tidak mendukung pembuatan *database* SQlite di dalam fitur-fiturnya. Untuk itu diperlukan *software* pembantu yang dapat membuat sebuah *database* kosong yang nantinya *database* tersebut dapat digunakan oleh Android untuk dimodifikasi atau diakses oleh suatu program Android tertentu.

Contoh *software* pendukung untuk pembuatan *database* SQLite kosong yang dapat digunakan adalah SQLite *Database Browser*. SQLite *Database Browser* merupakan aplikasi *open source* yang dapat memvisualisasikan struktur *database* dan datanya. Tampilan GUI yang *user friendly* membuat pengguna tidak harus mengerti tentang bahasa SQL. Selain itu mudah dalam memodifikasi data yang sangat membantu *programmer*. *Software* ini tersedia pada *platform* Windows, Linux dan Mac OS X. Berikut adalah tampilan antarmuka dari SQLite *Database Browse:*



Gambar II.10 Tampilan Antarmuka SQLite *Database Browser*

Pembuatan *database* dilakukan dengan membuat *file* SQLite baru lalu di dalam *file* tersebut diciptakan tabel dengan nama yang nantinya diakses menggunakan *method* yang ada di Eclipse untuk Android. Berikut adalah langkah-langkah pembuatan *database* SQLite pada Eclipse:

1. Pembuatan Kelas *DataHelper*

   Sebuah kelas yang berfungsi untuk mengatur pembuatan *database* dan *update*-nya.

2. Pembuatan Kelas *DataWrite*

   a. Deklarasi variabel untuk *field* data/kolom, contoh:

   ```
   public static final String KEY_ROWID = "_id";
   public static final String KEY_TRIGGER = "trigger";
   public static final String KEY_OUTPUT = "output";
   public static final String KEY_PROTOKOL = "protokol";
   public static final String KEY_FEEDBACK = "feedback";
   ```

   b. Deklarasi nama *database*, tabel dan versi, contoh:

   ```
   private static final String DATABASE_NAME =
   "textspeechmain";
   private static final String DATABASE_TABLE =
   "textspeech1";
   private static final int DATABASE_VERSION = 1;
   ```

c. Membuat *adapter database*

```
public DatabaseWrite(Context ctx)
{
this.context = ctx;
}
```

d. *Inserting* data baru untuk *database*, contoh:

```
public Cursor getById(long rowId) throws SQLException
{
Cursor mCursor = db.query(true, DATABASE_TABLE, new
String[]
KEY_ROWID,
KEY_INPUT,
KEY_OUTPUT,
KEY_KAMERA,
KEY_KECEPATAN,
KEY_JARAK,
KEY_LENGAN},
KEY_ROWID + "=" + rowId,
null,
null,
null,
null,
null);
if (mCursor != null) {
mCursor.moveToFirst();
}
return mCursor;
```

e. Membuat *method* untuk membuka dan menutup *database*

```
public DatabaseWrite open() throws SQLException {
DBHelper = new DatabaseHelper(context); db =
DBHelper.getWritableDatabase();
return this;
}
public void close() {
DBHelper.close();
}
```

f. Menghapus Data

```
public boolean deleteById(long rowId)
{
return db.delete(DATABASE_TABLE, KEY_ROWID +
"=" + rowId, null) > 0;
}
```

2.13 Metode Pengujian Sistem

Metode pengujian sistem terdiri dari Pengujian *black box* dan beta yang dilakukan untuk mengetahui efektifitas dari perangkat lunak (*software*) yang digunakan.

2.13.1 Pengujian Black Box

Pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Pengujian ini dianalogikan seperti melihat suatu kotak hitam yang hanya bisa dilihat penampilan luarnya saja, tanpa tahu ada apa dibalik bungkus hitamnya. *Black box* testing melakukan evaluasi untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi tidak benar atau hilang.
2. Kesalahan interface atau antarmuka.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan kinerja atau perilaku dan kesalahan inisialisasi dan terminasi.

2.13.2 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif dimana diuji secara langsung ke lapangan, dengan menggunakan kuesioner mengenai tanggapan pengguna terhadap aplikasi yang telah dibangun. Adapun metode penilaian pengujian yang digunakan yaitu metode kuantitatif berdasarkan data sampel dari pengguna .