BAB 2
TINJAUAN PUSTAKA

2.1. Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teoriteori yang berkaitan dalam pembangunan aplikasi *chatbot* sebagai Media Latihan Percakapan Bahasa Inggris Berbasis Android.

2.1.1. Pengolahan Bahasa Alami (Natural Language Processing)

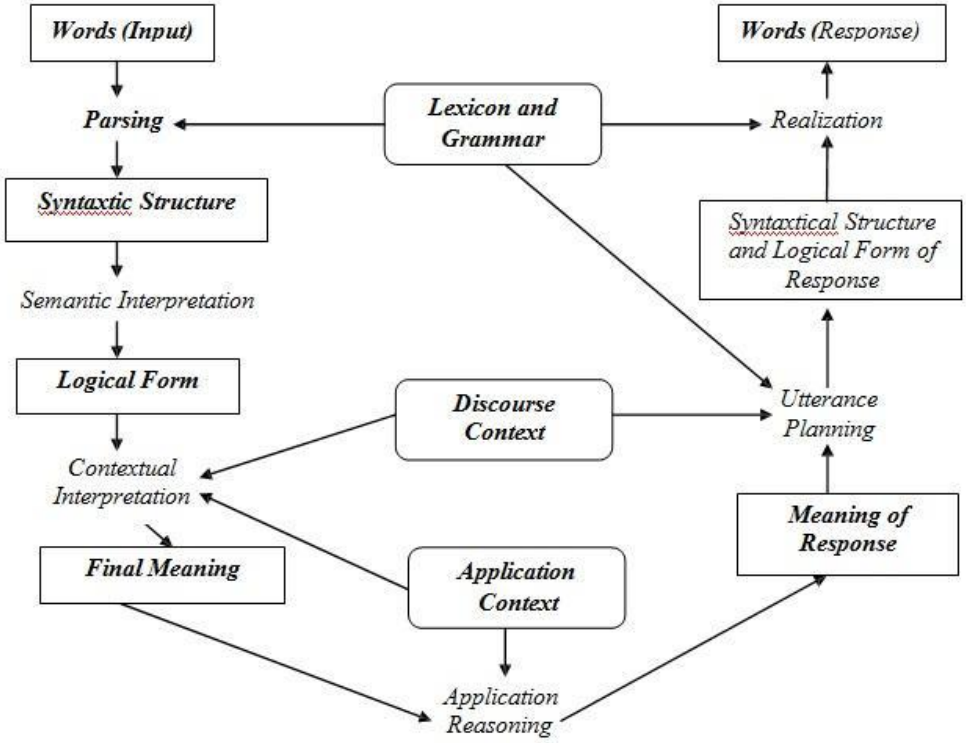
Pengolahan bahasa alami (*NLP*) adalah bidang ilmu komputer dan *linguistik* berkaitan dengan interaksi antara komputer dan manusia. Seluler generasi bahasa sistem komputer yang mengubah informasi dari database ke dalam bahasa manusia yang dapat dibaca.

Natural language processing, biasanya disingkat dengan *NLP*, mencoba membuat komputer mampu memahami suatu perintah yang dituliskan dalam bentuk bahasa sehari-hari dan diharapkan komputer juga merespon dalam bahasa yang mirip dengan bahasa natural. Setelah komputer bisa memahami perintah dalam bahasa *natural*, maka diharapakan sistem komputer juga dapat memberikan respon dalam bahasa *natural* pula. [13]

Sebuah sistem *natural language* harus memperhatikan pengetahuan terhadap bahasa itu sendiri baik dari segi kata yang digunakan bagaimana kata-kata tersebut digabung untuk menghasilkan suatu kalimat, apa arti suatu kata, apa fungsi sebuah kata dalam sebuah kalimat dan sebagainya. *Natural Language processor* tidak memperdulikan bagaimana suatu kalimat diinputkan ke komputer. Tugasnya adalah mengekstrak informasi dari kalimat. Inti dari sistem *NLP* adalah *parser*. *Parser* adalah bagian dari program atau *system* yang membaca setiap kalimat, kata demi kata, untuk menentukan "*what is what*".

NLP tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk teks atau suara menjadi data digital dan/atau sebaliknya pula; melainkan bertujuan untuk memahami arti dari kalimat yang diberikan dalam bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu. [13]

Untuk mencapai tujuan ini dibutuhkan tiga tahap proses. Proses yang pertama ialah *parsing* atau analisa *sintaksis* yang memeriksa kebenaran struktur kalimat berdasarkan suatu *grammar* (tata bahasa) dan *lexicon* (kosa kata) tertentu. Proses kedua ialah *semantic interpretation atauinterpretasi* semantik yang bertujuan untuk merepresentasikan arti dari kalimat secara context-independent untuk keperluan lebih lanjut. Sedangkan proses ketiga ialah *contextual* interpretation atau interpretasi kontekstual yang bertujuan untuk merepresentasikan arti secara *context dependent* dan menentukan maksud dari penggunaan kalimat. Gambaran sebuah organisasi sistem NLP dapat dilihat pada gambar 2.1. [13]



Gambar 2.1 Organisasi Sebuah Sistem NLP

Jenis aplikasi yang bisa dibuat pada bidang bidang *natural language* adalah *text - based application dan dialogue - based applications*.

1. *Text - based application*

   Text -based application ini adalah Mencakup segala macam aplikasi yang melakukan proses terhadap text tertulis seperti misalnya buku,

berita di surat kabar, *e-mail* dan lain sebagainya. Contoh penggunaan dari *text - based application* ini adalah :

a. Mencari topik tertentu dari buku yang ada pada perpustakaan.
b. Memberikan respon atas *input* yang diberikan.
c. Mencari isi dari surat atau *e-mail*.
d. Menterjemahkan dokumen dari satu bahasa ke bahasa yang lain.

2. *Dialogue - based application*

   Dialogue - based application idealnya pedekatan ini melibatkan bahasa lisan atau pengenalan suara, akan tetapi bidang ini juga memasukkan interaksi dengan cara memasukkan teks pertanyaan melalui *keyboard*. Aplikasi yang sering ditemui untuk bidang ini adalah :

   a. Sistem tanya jawab, dimana *natural language* digunakan dalam mendapatkan informasi dari suatu database.
   b. Sistem otomatis pelayanan melalui telepon.
   c. Kontrol suara pada perlatan sistem.
   d. Sistem *problem solving* yang membantu untuk melakukan penyelesaian masalah yang umum dihadapi dalam suatu pekerjaan.[13]

2.1.2. Teknologi Chatbot

Chatbot adalah sebuah program komputer yang dirancang untuk mensimulasikan sebuah percakapan atau komunikasi yang interaktif kepada user (manusia) melalui bentuk teks, suara, dan atau visual. Percakapan yang terjadi antara komputer dengan manusia merupakan bentuk respon dari program yang telah dideklarasikan pada database program pada komputer. Respon yang dihasilkan merupakan hasil pemindaian kata kunci pada inputan user dan menghasilkan respon balasan yang dianggap paling cocok, atau pola kata-kata yang dianggap paling mendekati, dari database tentunya. Dalam bahasa sehari-harinya *ChatBot* merupakan Aplikasi atau Program komputer yang dirancang untuk meniru manusia itu sendiri, batasan yang diambil dari *Chatbot* adalah mampu meniru komunikasi manusia. Jadi jika manusia sedang bercakap-cakap dengan program ini, seakan-akan ada dua pribadi manusia yang saling

berkomunikasi. Nyatanya tidak, manusia berkomunikasi dengan robot. Robot sudah dirancang untuk merespon segala jenis pertanyaan dan pernyataan yang diinputkan oleh manusia (*user*). Hal ini terjadi karena sebelumnya sudah dideklarasikan pada database, berupa entitas-entitas kata, pola kalimat, dan berbagai jenis pernyataan dan pertanyaan.[14]

Chatbot yang pertama adalah ELIZA yang dibuat pada tahun 1964 sampai 1966 oleh Professor Joseph Weizenbaum di MIT (*Massachusetts Institute of Technology*), dengan tujuan untuk mempelajari komunikasi natural language antara manusia dengan mesin. Eliza bertindak seolah-olah dia adalah seorang psikolog yang dapat menjawab pertanyaan-pertanyaan dari pasien dengan jawaban yang cukup masuk akal atau menjawabnya dengan pertanyaan balik.

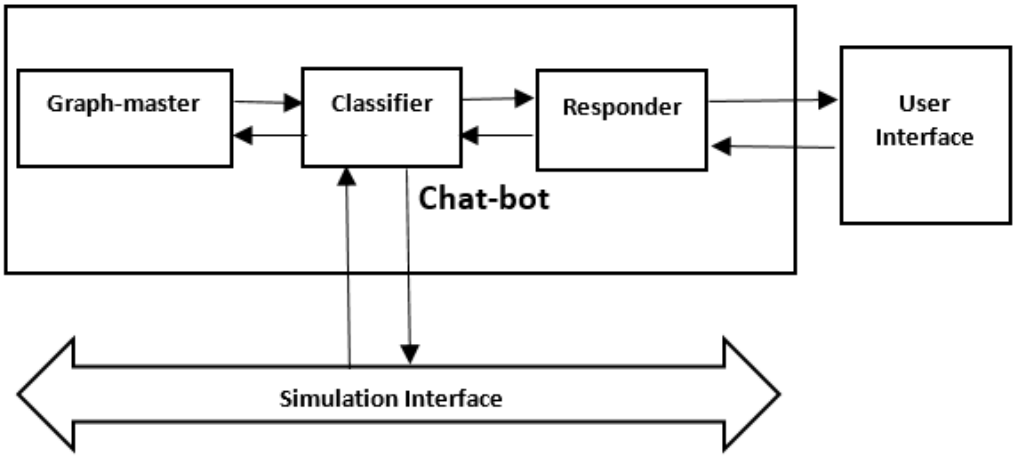
ELIZA adalah pelopor atau dapat disebut sebagai nenek moyangnya *chatbot*. ELIZA (juga dikenal sebagai Dr. Eliza) terkenal sebagai program *chat* yang "memiliki profesi" sebagai seorang psikiater. ELIZA mensimulasikan percakapan antara seorang psikiater dengan pasiennya menggunakan metode biasa yang dapat mencerminkan perasaan pasien dengan mengajukan pertanyaan-pertanyaan seperti: "*How do you ...*", "*Why do you feel like ...*", "*What do you think about ...*". Program ini akan mencari pola kata-kata tertentu pada input yang diberikan oleh pengguna, dan kemudian memberikan output yang sesuai.

ELIZA dapat memberikan respon yang tepat hanya jika pernyaataan pengguna ditulis dalam ejaan yang benar. Meskipun begitu, pengguna dapat berbicara secara santai dengan ELIZA.[14]

2.1.2.1. Komponen Chatbot

Paket *software chat-bot* terdiri dari tiga komponen penting: *Responder, Classifier* dan *Graph master*. Antarmuka antara rutin utama dan pengguna adalah *Responder*. Responden mentransfer data dari pengguna ke *Classifier*. *Responder* mengontrol *input* dan *output*nya. Fungsi *Classifier* adalah menormalkan dan menyaring data masukan. Masukan pengguna diganti dan dibagi menjadi komponen logis oleh *Classifier*. *Classifier* memindahkan kalimat yang dinormalisasi ke dalam komponen *Graph-master* dari *chat-bot*. *Classifier* memproses *output* dari komponen *master Graph* dari *chat-bot*. *Classifier* juga

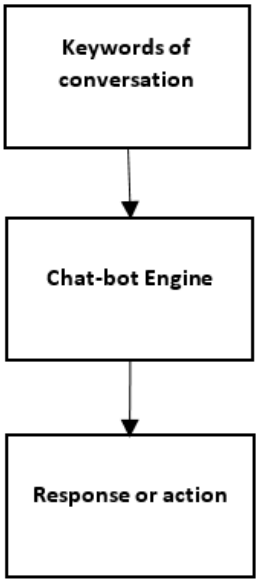
menangani instruksi sintaks *database* (misalnya AIML). Komponen *graph-master chat-bot* menangani pola pencocokan. *Graph-master* bertanggung jawab untuk mengatur penyimpanan konten otak *chat-bot*. Komponen *Graph-master* menyimpan isi otak sebagai grafik. *Graph-master* juga menangani proses pencocokan pola dan algoritma pencocokan pola. Diagram komponen *chat-bot* ditunjukkan pada gambar 2.2 [15].



Gambar 2.2 Komponen Chatbot

2.1.2.2. Text Processing, Response dan Action

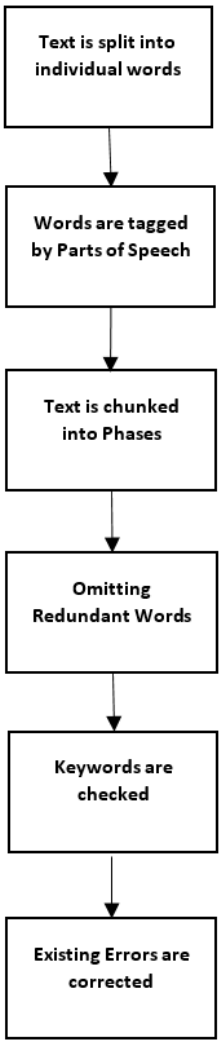
Input teks pengguna dipecah menjadi beberapa kata terpisah untuk menandai label *part-of-speech* sehubungan dengan posisi dan tetangga mereka dalam teks masukan. Pada tahap berikutnya, dengan bantuan berbagai jenis tata bahasa, kata-kata yang diberi label secara individual digabungkan untuk mengembangkan ungkapan. Pada tahap operasi chinking, kata kunci penting diambil dari frasa dengan menghapus kata-kata yang tidak diinginkan. Kata kunci diperiksa dan dikoreksi jika tidak benar. Tahap fase pemrosesan teks yang berbeda digambarkan pada gambar 2.3 [16].



Gambar 2.3 Fase Respon dan Tindakan

2.1.2.3. Text Processing

Chat-bot dapat dirancang untuk memberikan respons yang diharapkan terhadap percakapan teks bahasa manusia. Mesin *Chat-bot* dilengkapi dengan kata kunci yang diambil dari pemrosesan teks bahasa alami. *Output*nya adalah respon, yang akan ditunjukkan kepada pengguna. Diagram yang menggambarkan respons dan fase pengambilan tindakan ditunjukkan pada gambar 2.4 [16].



Gambar 2.4 Fase dari Text Processing

2.1.3. Speech Recognition

Pengenalan ucapan atau pengenalan suara dalam istilah bahasa Inggrisnya, *Automatic Speech recognition* (ASR) adalah suatu pengembangan teknik dan sistem yang memungkinkan komputer untuk menerima masukan berupa kata yang diucapkan. Teknologi ini memungkinkan suatu perangkat untuk mengenali dan memahami kata-kata yang diucapkan dengan cara digitalisasi kata dan mencocokkan sinyal digital tersebut dengan suatu pola tertentu yang tersimpan dalam suatu perangkat[8].

Kata-kata yang diucapkan diubah bentuknya menjadi sinyal digital dengan cara mengubah gelombang suara menjadi sekumpulan angka yang kemudian

disesuaikan dengan kode-kode tertentu untuk mengidentifikasikan kata-kata tersebut. Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk tulisan atau dapat dibaca oleh perangkat teknologi sebagai sebuah komando untuk melakukan suatu pekerjaan, misalnya penekanan tombol pada telepon genggam yang dilakukan secara otomatis dengan komando suara.

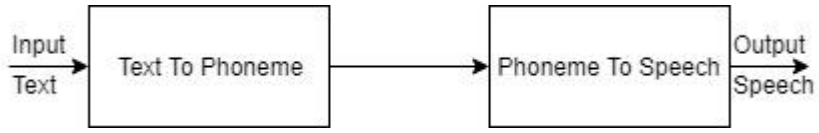
Berikut ini adalah dua tahap utama dalam pengenalan ucapan:

1. Pemrosesan sinyal: Tahap ini melibatkan menangkap kata-kata yang diucapkan ke mikrofon dan menggunakan *analog-to-digital converter* (ADC) untuk menerjemahkannya ke dalam data digital yang dapat diproses oleh komputer. ADC memproses data digital untuk menghilangkan gangguan dan melakukan proses lain seperti pembatalan gema agar dapat mengekstrak fitur-fitur yang relevan untuk pengenalan suara.
2. Pengenalan ucapan: Sinyal dibagi menjadi segmen-segmen menit yang dicocokkan dengan fonem-fonem bahasa yang akan dikenali. Fonem adalah unit ucapan terkecil, kira-kira setara dengan huruf-huruf alfabet. Misalnya, fonem dalam kata kucing adalah / k /, / ae /, dan / t /. Dalam bahasa Inggris, misalnya, ada sekitar 4 fonem, tergantung pada variasi bahasa Inggris yang digunakan.[8]

2.1.4. Text-to-Speech

Text-to-speech merupakan teknologi yang mengkonversi teks ke kata yang diucapkan dalam bentuk audio, dengan menganalisis dan memproses teks menggunakan *Natural Language Processing* (NLP) dan kemudian menggunakan Teknologi *Digital Signal Processing* (DSP) untuk mengonversi teks yang diproses menjadi representasi audio ucapan yang disintesis dari teks.[8]

Sistem Text to Speech pada prinsipnya terdiri dari dua sub sistem [8], yaitu:



Gambar 2.5 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 kodekode 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.

2.1.5. Android

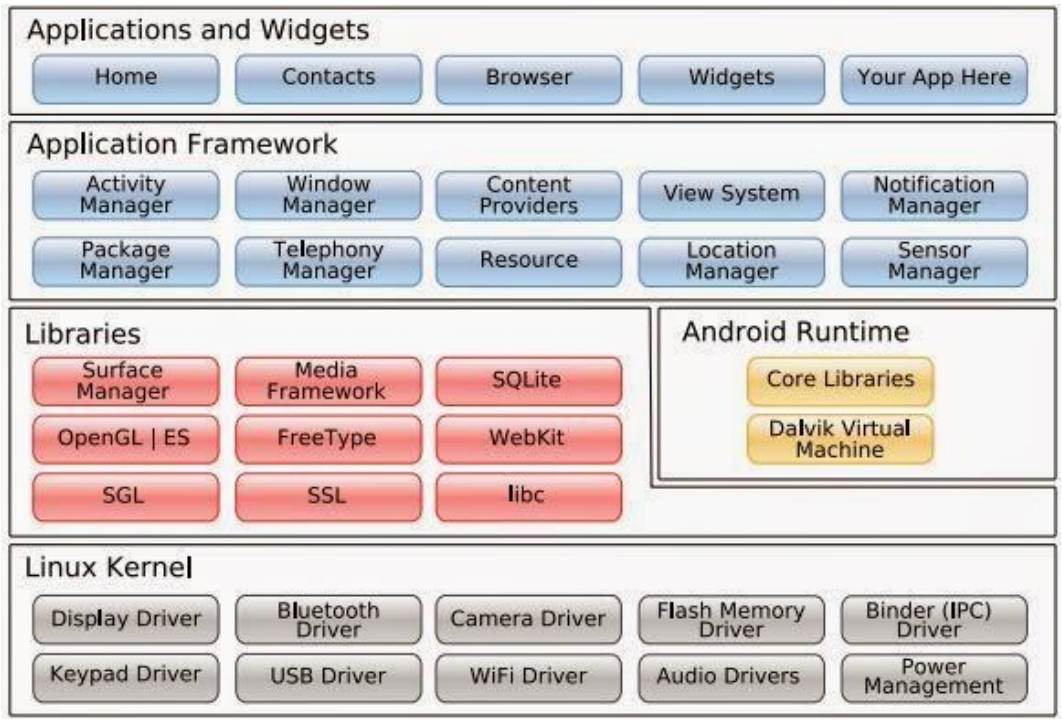
Android adalah sebuah sistem operasi untuk perangkat mobile berbasis Linux yang mencakup sistem operasi, middleware dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc., yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/smartphone. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan open source pada perangkat mobile. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan open platform perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Services (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD). Tidak hanya menjadi sistem operasi di smartphone, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi Tablet PC.[17]

2.1.5.1. Arsitektur Android

Secara garis bersar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut [17]:



Gambar 2.6 Arsitektur Android

1. *Applications* dan *Widgets*

   Applications dan *Widgets* ini adalah layer dimana pengguna berhubungan dengan aplikasi saja, dimana biasanya pengguna men-download aplikasi kemudian melakukan instalasi dan menjalankan aplikasi tersebut. Di layer terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *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 33 service *background*, mengatur alarm, menambahkan status notificatios, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi pada kategori inti. Arsitektur aplikasi dirancang supaya pengembang dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*). Sehingga pengembang bisa menyimpulkan Applications Frameworks ini adalah layer dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang, seperti contont-providers yang berupa sms dan lain sebagainya. Komponen-komponen yang termasuk didalam *Applications Frameworks* diantaranya *Views, Content Provider, Resouce Manager, Notification Manager, dan Activity Manager*.

3. *Libraries*

   Libraries ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi kebanyakan mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas kernel, layer ini meliputi berbagai library C/C++ inti seperti sebagai libc dan SSL, serta:

   a. *libraries* media untuk pemutaran media audio dan video
   b. *libraries* untuk manajemen tampilan
   c. *libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
   d. *libraries* SQLite untuk dukungan *database*
   e. *libraries* SSL dan WebKit terintegrasi dengan *web browser* dan security

f. *libraries LiveWebcore* mencakup modern web browser dengan engine embeded web view

g. *libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's.

4. Android Run Time

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* 34 (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Didalam *Android Run Time* dibagi menjadi dua bagian yaitu:

a. *Core Libraries*: Aplikasi Android dibangun dalam bahasa java, sementara Dalvik sebagai virtual mesinya bukan Virtual Machine Java, sehingga diperlukan sebuah libraries yang berfungsi untuk menerjemahkan bahasa java/c yang dihandle oleh Core Libraries.

b. *Dalvik Virtual Machine*: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat linux kernel untuk thereading dan manajemen tingkat rendah.

5. Linux Kernel

Linux kernel adalah layer dimana inti dari operating sistem dari Android itu berbeda. Berisi file-file system yang mengatur *sistem processing, memory, resource, drivers*, dan sistem-sistem *operating sistem* android lainnya. *Linux karnel* yang digunakan android adalah *linux karnel* 2.6.[17]

2.1.6. Basis Data

Database atau Basis Data adalah sekumpulan data yang saling terhubung satu dengan yang lainnya atau sekumpulan tabel yang saling terhubung satu dengan yang lainnya.dan fungsi dari database adalah menyimpan suatu data pada tabel-tabel dan dikumpulkan menjadi satu dengan *database*. *Database* juga bisa di umpamakan sebagai sebuah rumah dengan beberapa kamar-kamar dan sebuah property seperti almari meja belajar tempat tidur itu bisa di sebut dengan data querynya.

Basis data terdiri dari 2 kata, yaitu basis dan data. Basis dapat diartikan sebagai tempat berkumpul, sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, barang, hewan, peristiwa, dan sebagainya yang direkam dalam bentuk angka, huruf, symbol, teks, gambar, bunyi atau kombinasinya.[14]

Ada beberapa operasi dasar yang dapat dilakukan berkenaan dengan basis data, antara lain:

1. Pembuatan basis data baru (*create database*)
2. Penghapusan basis data (*drop database*)
3. Pembuatan file/tabel baru ke suatu basis data (*create table*)
4. Penghapusan file/tabel dari suatu basis data (*drop table*)
5. Penambahan/pengisian data baru ke sebuah file/tabel (*insert*)
6. Pengambilan/pencarian data dari sebuah file/tabel (*retrieve/search*)
7. Pengubahan data dari sebuah file/tabel (*update*)
8. Penghapusan data dari sebuah file/tabel (*delete*)

Istilah-istilah dalam database :

1. *Entity*

   Entity adalah orang, tempat, kejadian atau konsep yang informasinya direkam. Misalnya pada kampus terdapat entity mahasiswa, matakuliah, dosen, nilai test dan lain-lain.

2. *Atribute*

   Setiap *entity* mempunyai *atribute* atau sebutan untuk mewakili suatu *entity*. Seorang mahasiswa dapat dilihat dari atributnya, misalnya nim, nama, alamat, jenis kelamin dan lain-lain. *Atribute* juga disebut sebagai data elemen, data field, data item.

3. *Data value* (nilai atau isi data)

   Data value adalah data actual atau informasi yang disimpan pada tiap data elemen atau *atribute*. *Atribute* nama mahasiswa menunjukan tempat dimana informasi nama mahasiswa disimpan, sedang *data value* adalah dali, andi, merupakan isi data nama mahasiswa tersebut.

4. *Record* (tupel)

   Record yaitu kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entity secara lengkap. Satu *record* mewakili satu data atau informasi tentang seseorang misalnya, nomor induk mahasiswa, nama, alamat, jenis kelamin dan seterusnya.

5. *File*

   File adalah kumpulan record-record sejenis yang mempunyai panjang elemen yang sama, atribute yang sama, namun berbeda-beda data valuenya.

6. *Database*

   Database adalah kumpulan file-file yang mempunyai kaitan antara satu file dengan file yang lain sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan, instansi dalam batasan tertentu.

7. *Database Management System* (DBMS)

   Database Management System (DBMS) adalah program untuk pengelolaan disebut DBMS. *Database* adalah kumpulan datanya, sedangkan program pengelolanya berdiri sendiri dalam satu paket program yang berfungsi untuk membaca data, mengisi data, menghapus data serta melaporkan data dalam *database*.[14]

2.1.7. Object Oriented Programming (OOP)

Object-Oriented Programming (OOP) adalah sebuah pendekatan untuk pengembangan / development suatu software dimana dalam struktur software tersebut didasarkan kepada interaksi *object* dalam penyelesaian suatu proses/tugas. Interaksi tersebut mengambil form dari pesan-pesan dan mengirimkannya kembali antar *object* tersebut. *Object* akan merespon pesan tersebut menjadi sebuah tindakan / action atau metode[14].

Object-oriented programs terdiri dari objects yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, users dari software programs dilibatkan dari logika proses untuk menyelesaikan tugas. Adapun karakteristik yang dimiliki OOP yaitu :

1. *Objects*

   Object adalah sebuah structure yang menggabungkan data dan prosedur untuk bekerja bersama-sama.

2. *Abstraction*

   Ketika manusia berinteraksi dengan object-object di dunia ini, manusia sering hanya konsentrasi dengan sebuah bagian dari propertiesnya. Ketika membangun *objects* dalam aplikasi OOP, adalah penting untuk menggabungkan konsep abstraction ini. Jika membangun aplikasi shipping, maka harus membangun object produk dengan atribut seperti ukuran dan berat. Warna adalah contoh informasi yang tidak ada hubungannya dan harus dibuang. Tetapi ketika akan membangun order entry application, warna menjadi penting dan harus termasuk atribut object produk.

3. *Encapsulation*

   Ciri penting lainnya dari OOP adalah *encapsulation*. *Encapsulation* adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan hidden. Jika ingin mendapat data, harus berinteraksi dengan object yang bertanggung jawab atas data tersebut.

4. *Polymorphism*

   Polymorphisms adalah kemampuan 2 buah object yang berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Dalam OOP, diterapkan tipe *polymorphism* melalui proses yang disebut overloading. Dapat dilakukan dalam implementasi metode yang berbeda pada sebuah object yang mempunyai nama yang sama.

5. *Inheritance*

   Penggunaan *inheritance* dalam OOP untuk mengklasifikasikan *objects* dalam program sesuai karakteristik umum dan fungsinya.[14]

2.1.8. Application Programming Interface

Application Programming Interface (API) adalah sebuah teknologi yang memfasilitasi pertukaran informasi atau data antara dua atau lebih aplikasi perangkat luan. API adalah antarmuka virtual antara dua fungsi perangkat lunak

yang saling bekerja sama, seperti antara sebuah *word processor* dan sebuah *spreadsheet*. Sebuah API mendefinisikan bagaimana cara programmer memanfaatkan suatu fitur tertentu dari sebuah komputer. API tersedia untuk sistem windowing, sistem file, sistem basisdata, serta sistem jaringan.[18]

2.1.9. Languagetool API

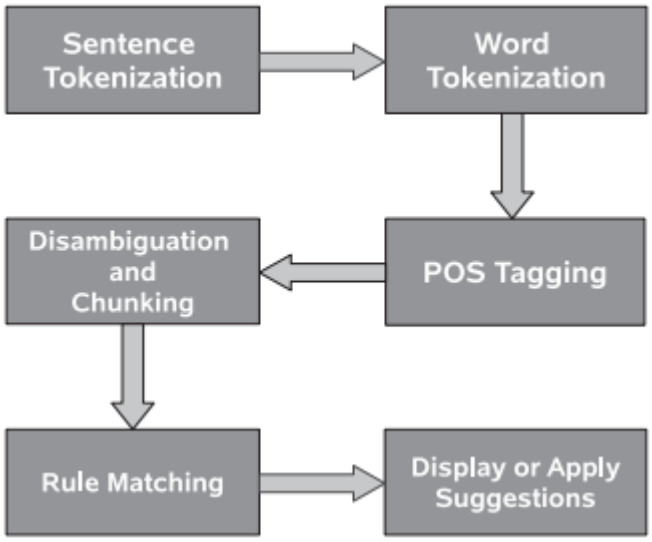
LanguageTool adalah pemeriksa gaya dan tata bahasa berdasarkan aturan bersifat gratis dan open source. Sebagian besar aturan ditulis secara manual baik dalam XML atau Java, oleh sebab itu pengembangan terhadap aturan membutuhkan pengetahuan tentang bahasa pemrograman tersebut. Program ini tersedia sebagai versi yang berdiri sendiri dan dapat digunakan di beberapa aplikasi lain seperti LibreOffice dan TeXstudio.

Ketika sebuah teks diperiksa, LanguageTool menggunakan splitter kalimat, tokenizer, dan pemberian tag *part-of-speech* kalimatnya sendiri untuk menetapkan teks *part-of-speech* ke setiap token dalam input. Setiap kalimat kemudian diperiksa dengan gaya dan aturan tata bahasa.[19]

LT (*LanguageTool*) diciptakan oleh Daniel Naber dengan Python sebagai karya diplomanya, dan kemudian porting ke Java dan diperpanjang. Saat ini dikelola oleh Daniel Naber dan penulis saat ini. Ada juga pengelola aturan yang mengurus penerapan aturan untuk masing-masing bahasa. [20]

Ada dua jenis alat proofreading yang dapat dibangun: statistik atau *rule-based*. Alat proofreading statistik (atau *machine-learned*) dapat berupa *positive-match checker* atau *negative-match checker*. Adapun *Checker rule-based* juga bisa berupa *positive-match* atau *negative-match*. LT menggunakan metode *positive-match checker rule-based*, Pengecekan pencocokan positif(*positive-match checker*) didasarkan pada konsep intuitif dari kesalahan tata bahasa atau gaya, yaitu kesalahan dipilih sendiri (berdasarkan rata-rata intuisi pengguna bahasa, kamus, teori linguistik atau analisis corpora). Dengan cara ini *positive-match checker* dapat digunakan untuk memodelkan kesalahan gaya yang paling parah dan sering ditemukan di sebagian besar anilisis corpora (terutama dalam register sehari-hari dan / atau dalam naskah pidato).[20]

LT ditulis dalam Java dan mengimplementasikan alur kerja dasar berikut (lihat gambar 2.7). Teks input di-tokenkan ke dalam kalimat dan kata-kata individual. Kata *tokenizer* hanya membagi string di setiap spasi dan tanda baca (untuk beberapa bahasa, seperti Belanda, tanda kutipnya, ketika tidak didahului dengan spasi, tetap menjadi bagian dari kata sebelumnya).



Gambar 2.7 LT Basic Workflow[20]

Perlakuan dashes bervariasi dari bahasa ke bahasa; di beberapa dari mereka, itu dianggap sebagai bagian dari kata. Meskipun pengaturan ini sangat sederhana, ia bekerja cepat dan memenuhi kebutuhan kami. Perlu dicatat, bagaimanapun, bahwa arsitektur memungkinkan untuk menggunakan *tokenizer* lain, misalnya, untuk bahasa seperti Thai yang membutuhkan leksikon untuk membagi teks input menjadi token individu.

Kalimat *tokenizer* mendukung Segmentasi Aturan Exchange (SRX) 2.0 standar XML. Ini memungkinkan pengguna untuk menyesuaikan segmentasi serta data segmentasi pertukaran dengan alat lain seperti perangkat lunak terjemahan yang dibantu komputer. Kata-kata kemudian secara opsional ditandai oleh tagger POS(*part-of-speech*), dan, juga opsional, disambiguasi atau chunked (ditandai pada tingkat urutan kata, bukan hanya satu kata).[20]

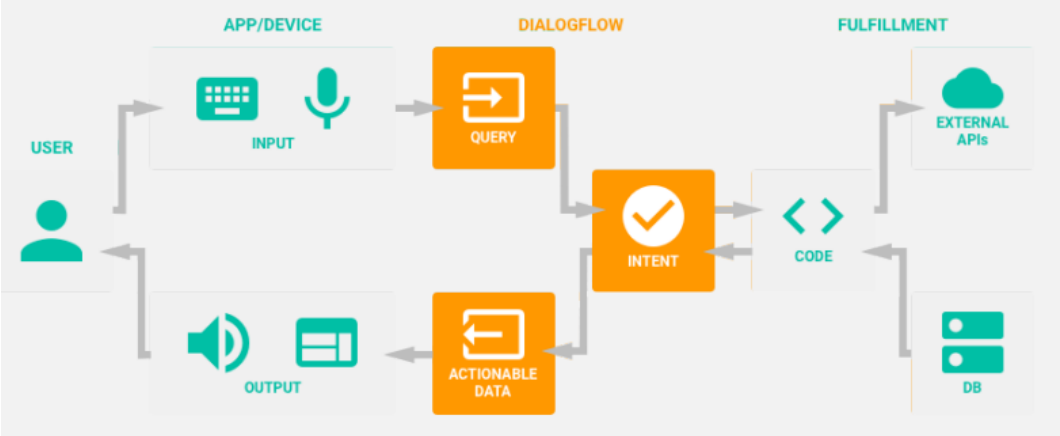
Ke daftar kata yang disusun dalam kalimat, aturan pencocokan kesalahan diterapkan. Ada dua jenis aturan dasar yang tersedia: standar, aturan-kode XML

yang dienkode pada tingkat kalimat individu dan aturan berbasis Java untuk menangkap kesalahan yang lebih sulit dicocokkan dengan aturan XML. Misalnya, aturan tingkat paragraf hanya mungkin di Java; mereka saat ini termasuk mendeteksi tanda kutip yang tidak seimbang, tanda kurung dan sejenisnya, serta mempertahankan varian kata yang sama dalam paragraf yang sama (untuk Jerman).

Pencocokan aturan menghasilkan saran koreksi (jika tersedia), dan hasil pemeriksaan dihasilkan pada output. Sebagai alternatif, teks input dikoreksi secara otomatis dengan menggunakan saran yang ditentukan dalam aturan. Saran yang dihasilkan mungkin rumit dan mencakup infleksi kata-kata atau grup kata tertentu.[20]

2.1.10. Dialogflow API

Dialogflow adalah platform untuk mengembangkan *chat-bots* berdasarkan percakapan bahasa alami. Konsep penting seperti *Intents* and *Contexts* digunakan untuk memodelkan perilaku *chat-bot*. Maksudnya adalah pemetaan antara apa yang pengguna masukan dan respon atau tindakan apa yang harus dilakukan oleh bot. Konteks digunakan untuk membedakan input pengguna yang mungkin memiliki maksud yang berbeda tergantung pada input pengguna sebelumnya. Saat pengguna memasukkan data ke dalam platform Dialogflow.com, pertama kali diperiksa jika cocok dengan maksud yang telah ditentukan sebelumnya. Dialogflow.com memiliki fitur bernama "*Default fallback intent*" untuk menangani input pengguna yang tidak sesuai dengan maksud yang telah ditentukan sebelumnya. Kasus pencocokan suatu maksud dapat dibatasi dengan menyebutkan daftar konteks yang seharusnya bisa berjalan. Kasus pencocokan suatu maksud dapat membuat dan menghapus konteksnya. Sistem maksud dan konteks ini membuat ketentuan untuk mengembangkan *chat-bots* yang bisa memiliki arus besar dan kompleks. Keterbatasan Dialogflow.com adalah: *chat-bot* tidak dapat dirancang sedemikian rupa sehingga sebuah tujuan dapat dicocokkan hanya jika konteks tertentu tidak ada [21]. Secara umum proses yang ada pada dialogflow ditunjukan pada gambar 2.8.



Gambar 2.8 Proses Dialogflow API

2.1.10.1. Komponen Dialogflow API

Dialogflow API terdiri dari beberapa komponen diantaranya *Agent, Intent, Entities, dan Contexts*. Berikut adalah penjelasan tiap-tiap komponen.

1. Agent

Agent paling tepat digambarkan sebagai modul NLU (Pemahaman Bahasa Alami). Mereka dapat digunakan oleh aplikasi, produk, atau layanan Anda untuk mengubah permintaan pengguna alami menjadi data yang dapat ditindaklanjuti. Transformasi ini terjadi ketika input pengguna cocok dengan salah satu tujuan di dalam agen Anda. Maksud adalah komponen agent yang ditetapkan sebelumnya atau yang ditetapkan pengembang yang memproses permintaan pengguna.

Agent dapat dirancang untuk mengelola alur percakapan dengan bantuan konteks, prioritas maksud, pengisian slot, tanggung jawab, dan pemenuhan melalui webhook. Diagram berikut menunjukkan penanganan permintaan pengguna. Seorang agent mencakup komponen DIALOGFLOW.

2. Intent

Intent adalah pemetaan antara apa yang dikatakan pengguna dan tindakan apa yang harus diambil oleh perangkat lunak Anda. Antarmuka *Intent* memiliki bagian-bagian berikut:

a. Training Phase

b. Action

c. Response

d. Context

3. Entities

Entities adalah alat canggih yang digunakan untuk mengekstraksi nilai parameter dari input bahasa alami. Setiap data penting yang ingin Anda dapatkan dari permintaan pengguna, akan memiliki *entities* yang sesuai.

Entities yang digunakan dalam *agent* tertentu akan bergantung pada nilai parameter yang diharapkan akan dikembalikan sebagai hasil dari fungsi *agent*. Dengan kata lain, pengembang tidak perlu membuat *entities* untuk setiap konsep yang mungkin disebutkan dalam *agent* - hanya untuk mereka yang diperlukan untuk data yang dapat ditindaklanjuti.

Ada 3 jenis *entities*: sistem (didefinisikan oleh Dialogflow), pengembang (didefinisikan oleh pengembang), dan pengguna (dibangun untuk setiap individu pengguna akhir dalam setiap permintaan) *entities*. Masing-masing dapat diklasifikasikan sebagai pemetaan (memiliki nilai referensi), enum (tidak memiliki nilai referensi), atau komposit (mengandung *entities* lain dengan alias dan mengembalikan nilai jenis objek) *entities*.

4. Contexts

Contexts mewakili konteks saat ini dari permintaan pengguna. Ini berguna untuk membedakan frasa yang mungkin tidak jelas atau memiliki arti yang berbeda tergantung pada preferensi pengguna, lokasi geografis, halaman saat ini di aplikasi, atau topik percakapan.

Misalnya, jika pengguna mendengarkan musik dan menemukan band yang menarik minat mereka, mereka mungkin mengatakan sesuatu seperti: "Saya ingin mendengar lebih banyak dari mereka". Sebagai pengembang, mereka dapat menyertakan nama band dalam *contexts* dengan permintaan, sehingga agent dapat menggunakannya dalam maksud lain.

Atau katakanlah seseorang adalah produsen perangkat rumah pintar, dan ia memiliki aplikasi yang mengontrol lampu dan peralatan rumah tangga dari jarak jauh. Seorang pengguna mungkin berkata, "Hidupkan lampu teras depan", diikuti dengan "Matikan". Dengan mengatur *contexts* aplikasi akan mengerti bahwa frase kedua mengacu pada cahaya dari permintaan pertama.

Kemudian, jika pengguna berkata, "Hidupkan mesin kopi", dan kemudian "Matikan", itu akan menghasilkan tindakan yang berbeda dari sebelumnya, karena *contexts* baru.

2.1.11. Google Speech Recognition API

Penggunaan *Google Speech Recognition API* memungkinkan developer untuk mengkonversi ucapan kedalam teks. Untuk dapat menggunakan fitur tersebut, developer android dapat menggunakan interface dan class yang telah disediakan Google API (package android.speech) [22].

1. .*Interfaces* RecognitionListener digunakan untuk menerima pemberitahuan dari SpeechRecognizer ketika peristiwa pengenalan (recognition) terkait terjadi.
2. *Classes* Terdapat beberapa class yang terdapat pada package android.speech, diantaranya:
    a. RecognitionService, kelas ini menyediakan kelas dasar untuk implementasi layanan pengenalan
    b. RecognitionService.Callback, kelas ini menerima callback dari layanan pengenalan suara dan mengirimkannya ke pengguna
    c. RecognizerIntent, konstanta untuk mendukung pengenalan suara melalui memulai Intent
    d. RecognizerResultsIntent, konstanta untuk intents yang berkaitan dengan menunjukkan hasil pengenalan suara
    e. SpeechRecognizer, kelas ini menyediakan akses ke layanan pengenalan suara.

Untuk dapat memanfaatkan fitur Google Speech Recognition API pada program diperlukan beberapa langkah, diantaranya mengimpor class yang berhubungan dengan speech recognition, yaitu `android.speech.RecognizerIntent;`

Pada dasarnya memicu Intent (android.speech.RecognizerIntent) berfungsi untuk menunjukan kotak dialog untuk mengenali suara masukan. Activity ini kemudian mengubah suara menjadi teks dan mengirim kembali hasil dari pemanggilan Activity. Ketika memanggil Intent android.speech.RecognizerIntent,

developer harus menggunakan `startActivityForResult()` untuk menunjukkan hasil pengenalan suara. [22]

2.1.12. Google Text To Speech API

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) [22].

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 textto-speech e) TextToSpeechService, kelas abstrak dasar untuk implementasi mesin TTS f) UtteranceProgressListener, listener untuk kejadian yang berkaitan dengan kemajuan suatu ucapan melalui antrian sintesis. [22]

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 initListenere.

```
private EditText write;
ttobj = 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. `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.[22]

2.1.13. Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan artifact bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak) dari system

perangkat lunak, seperti pada pemodelan bisnis dan system non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan system yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.[14]

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism.

1. *View*

   View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain: *use case view, logical view, component view, concurrency view, dan deployment view.*

   a) *Use case view*

      Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. View ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

   b) *Logical view*

      Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,*dan *relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam class diagrams untuk struktur statis dan dalam state, *sequence, collaboration,* dan *activity diagram* untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

c) *Component view*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. *View* ini digambarkan dalam *component vie*w dan digunakan untuk pengembang (developer).

d) *Concurrency view*

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis (*state, sequence, collaboration, dan activity diagrams)* dan diagram implementasi (*component dan deployment diagrams)* serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji *(tester*).

e) *Deployment view*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (nodes) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrato*r), dan penguji (*tester*).[14]

2. Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya dialokasikan untuk view tertentu. Adapun jenis diagram antara lain :

a) *Use Case Diagram*

Use case adalah abstraksi dari interaksi antara sistem dan actor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata user.

Sedangkan *use case diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

Use case diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem.

b) Class Diagram

Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya class diagram dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class- class yang ada. Sebuah sistem biasanya mempunyai beberapa class diagram. Class diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem. Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar class yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya.
c. *Public*, dapat dipanggil oleh siapa saja.

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

c) Component Diagram

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi software dari sebuah atau lebih class. Komponen dapat berupa *source code*, komponen biner, atau

executable component. Sebuah komponen berisi informasi tentang logic class atau class yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga component diagram merepresentasikan dunia nyata yaitu *component software* yang mengandung *component, interface dan relationship*.

d) Deployment Diagram

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware, menunjukkan hubungan komputer dengan perangkat (nodes) satu sama lain dan jenis hubungannya. Di dalam nodes, *executeable component* dan object yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh node tertentu dan ketergantungan komponen.

e) State Diagram

Menggambarkan semua state (kondisi) yang dimiliki oleh suatu object dari suatu class dan keadaan yang menyebabkan state berubah. Kejadian dapat berupa object lain yang mengirim pesan. State class tidak digambarkan untuk semua class, hanya yang mempunyai sejumlah state yang terdefinisi dengan baik dan kondisi class berubah oleh state yang berbeda.

f) Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antara object, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

g) Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti sequence diagrams. Dalam menunjukkan pertukaran pesan, collaboration diagrams menggambarkan object dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan sequence

diagrams, tapi jika penekanannya pada konteks gunakan collaboration diagram.

h) Activity Diagram

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi.[14]

2.1.14. SQLite

SQLite adalah sebuah embedded engine database SQL yang tidak seperti kebanyakan database SQL lainnya, dimana SQLite tidak memiliki proses server yang terpisah. SQLite melakukan read dan write secara langsung pada disk file. Sebuah database SQL yang lengkap dengan beberapa tabel, *index, trigger, view* yang hanya berbentuk file biasa pada disk. Format file database yang digunakan adalah cross-platform, dimana SQLite bebas menyalin database antara sistem 32-bit dan 64-bit atau antara arsitektur big-endian dan little-endian [23].