PENGEMBANGAN APLIKASI NOTIFIKASI SMS DAN PANGGILAN TELEPON MENGGUNAKAN ANDROID TEXT TO SPEECH DAN POCKETSPHINX SPEECH RECOGNITION UNTUK PENGGUNA BERKENDARA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Eko Aditya Santoso
NIM: 115060800111008

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PENGEMBANGAN APLIKASI NOTIFIKASI SMS DAN PANGGILAN TELEPON MENGGUNAKAN ANDROID TEXT TO SPEECH DAN POCKETSPHINX SPEECH RECOGNITION UNTUK PENGGUNA BERKENDARA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Eko Aditya Santoso
NIM: 115060800111008

Skripsi ini telah diuji dan dinyatakan lulus pada
19-01-2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr.Eng. Herman Tolle, S.T., M.T.	Aryo Pinandito, S.T., M.MT.
NIP: 19740823 200012 1 001	NIP: 19830519 201404 1 001

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, M.T.
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 7 Januari 2016

Eko Aditya Santoso

NIM: 115060800111008

KATA PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih Lagi Maha Penyayang. Segala puji dan syukur penulis panjatkan atas kehadirat-Nya, karena hanya atas rahmat, pertolongan dan kasih sayang-Nya penulis dapat menyelesaikan skripsi dengan judul "Pengembangan Aplikasi Notifikasi SMS Dan Panggilan Telepon Menggunakan Android Text to Speech Dan Pocketsphinx Speech Recognition Untuk Pengguna Berkendara" dengan baik.

Shalawat serta salam atas junjungan nabi besar kita Nabi Muhammad SAW beserta keluarga dan para sahabat sekalian. Melalui kesempatan ini, penulis menyampaikan rasa hormat dan terimakasih kepada pihak-pihak yang telah memberikan bantuan kepada penulis dalam penyelesaian skripsi ini. Pihak-pihak tersebut antara lain:

1. Bapak Dr.Eng. Herman Tolle, S.T., M.T. dan Bapak Aryo Pinandito, S.T., M.M.T. Selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan, ilmu dan saran dalam penyusunan skripsi ini.
2. Kedua orang tua yang telah memberi motivasi, kasih sayang, dan dukungan moril serta materil kepada penulis. Kemudian juga kepada adik dari penulis yang telah memberikan semangat dan bantuan yang sangat besar dalam penyelesaian skripsi ini.
3. Seluruh dosen Program Studi Informatika/Ilmu Komputer atas kesediaanya membagi ilmunya kepada penulis.
4. Seluruh staf dan karyawan di PTIIK Universitas Brawijaya yang telah membantu kelancaran pengerjaan skripsi.
5. Teman-teman kuliah diantaranya seluruh anggota kelas TIF-H serta semua teman-teman angkatan 2011, terimakasih atas segala bantuannya selama menjadi mahasiswa serta dukungannya dalam menyelesaikan skripsi ini.

Semoga jasa dan amal baik kita semua mendapatkan balasan dari Allah SWT. Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan masih jauh dari sempurna. Oleh karena itu, kritik dan saran untuk kesempurnaan skripsi ini, senantiasa penulis harapkan dari berbagai pihak.

Malang, 7 Januari 2016

Penulis

echo.adityas@gmail.com

ABSTRAK

Eko Aditya Santoso. 2015. Pengembangan Aplikasi Notifikasi SMS Dan Panggilan Telepon Menggunakan Android Text To Speech Dan Pocketsphinx Speech Recognition Untuk Pengguna Berkendara. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.

Dosen Pembimbing : Dr.Eng. Herman Tolle, S.T., M.T. dan Aryo Pinandito, S.T., M.M.T.

SMS dan panggilan telepon yang masuk biasanya akan muncul sebagai pesan notifikasi, baik dalam bentuk nada dering maupun getaran. Dengan adanya peringatan notifikasi tersebut secara tidak langsung akan memaksa pengguna untuk membuka smartphone miliknya, dimana hal itu dapat mengganggu aktivitas akibat dari adanya SMS dan panggilan telepon yang belum tentu benar-benar penting untuk dibuka. Aktivitas tersebut sering dilakukan oleh para pengendara yang dapat menjadi salah satu faktor terjadinya kecelakaan. Guna mengurangi permasalahan tersebut, maka dirancang dan diimplementasikan sebuah aplikasi perangkat bergerak dengan memanfaatkan *Text to Speech* sebagai pembaca notifikasi SMS dan panggilan masuk serta penggunaan *Speech Recognition* sebagai pemroses respon pengguna. Untuk melakukan implementasi kedua modul tersebut maka digunakanlah Android Text to Speech dan *library* PocketSphinx Speech Recognition, dimana hal ini bertujuan agar aplikasi dapat digunakan secara *offline*. Hasil dari pengujian fungsionalitas menunjukkan bahwa sistem telah memenuhi kriteria spesifikasi kebutuhan fungsional pada tahap perancangan. Selain itu hasil pengujian akurasi dengan menghitung nilai F-measure pada modul Speech Recognition menunjukkan nilai akurasi sebesar 87%.

Kata kunci: TTS, Speech, Recognition, Android, SMS.

ABSTRACT

Eko Aditya Santoso. 2015. Pengembangan Aplikasi Notifikasi SMS Dan Panggilan Telepon Menggunakan Android Text To Speech Dan Pocketsphinx Speech Recognition Untuk Pengguna Berkendara. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang.

Dosen Pembimbing : Dr.Eng. Herman Tolle, S.T., M.T. dan Aryo Pinandito, S.T., M.M.T.

SMS and incoming phone calls will usually appear as a notification message, either in the form of ringtones or vibration. Given the warning notification will indirectly force the user to open his smartphone, where it can interfere with the activity as a result of the SMS and phone calls are not necessarily important to be opened. These activities are often carried out by the rider who can be one of the factors of road accident. In order to reduce the risk of road accident, there is a mobile device application that designed and implemented by using Text to Speech as a notification reader and Speech Recognition for user response processing. The implementation process is using Android Text to Speech and PocketSphinx Speech Recognition library, where it is intended that an application can be used offline. Results of functionality testing indicates that the system has met the criteria of functional requirements specification at the design stage. In addition, the test results of accuracy testing by calculating the F-measure on Speech Recognition module shows the accuracy value by 87%.

Keyword: TTS, Speech, Recognition, Android, SMS.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE

DAFTAR PERSAMAAN

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi *smartphone* saat ini yang semakin maju, memungkinkan seseorang bisa mengakses informasi dimanapun hanya dengan genggaman tangan. Di Indonesia sendiri, perkembangan perangkat bergerak dan *smartphone* sangatlah pesat dimana hal itu juga diikuti dengan jumlah persentase penggunanya yang terus meningkat. Namun, dibalik semua kecanggihan tersebut ternyata *smartphone* masih belum sepenuhnya mampu memenuhi kebutuhan penggunanya dalam hal kemudahan dan kenyamanan. Dalam skripsi ini akan dilakukan penelitian untuk meningkatkan kemampuan *smartphone* dalam aspek kemudahan dan kenyamanan tersebut, dimana akan dilakukan penelitian terhadap fitur Short Message Service (SMS) dan panggilan telepon. Karena perlu diketahui, SMS dan panggilan telepon yang masuk akan muncul sebagai pesan notifikasi baik dalam bentuk nada dering maupun getaran. Dengan adanya peringatan notifikasi tersebut secara tidak langsung akan memaksa pengguna untuk membuka smartphone miliknya, dimana hal itu dapat mengganggu aktivitas akibat dari adanya SMS dan panggilan telepon yang belum tentu benar-benar penting untuk dibuka. Di lain tempat, kegiatan tersebut juga sering dilakukan oleh para pengendara yang dapat menjadi salah satu faktor kecelakaan. Sebagai informasi, kecelakaan lalu lintas di Indonesia saat ini cenderung terus mencemaskan. Pada tahun 2011 saja, rata-rata korban tewas akibat kecelakaan sebanyak 68 orang per hari. Data Polda Metro Jaya menyebutkan, angka kecelakaan lalu lintas jalan yang dipicu pemakaian ponsel meningkat sekitar 1,2% pada 2010 jika dibandingkan 2009. Padahal, UU No 22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan (LLAJ) melarang kegiatan berponsel saat berkendara karena bisa mengganggu konsentrasi.

Oleh karena itu, untuk mengurangi kegiatan merugikan tersebut diperlukan adanya sebuah metode yang dapat memaksimalkan penggunaan smartphone Android, yaitu dengan memanfaatkan *Text to Speech* sebagai pembaca notifikasi SMS dan panggilan masuk serta penggunaan *Speech Recognition* sebegai pemroses respon pengguna. Digunakannya *Text to Speech* adalah berdasarkan jurnal karya Sabyasachi Patra dari Institut Narsee Monjee Institute of Management Studies, India. Dalam jurnal tersebut, ia melakukan pengujian *Text to Speech* standar milik Android untuk membaca SMS sebagai upaya menggalakkan *Text-less Safe Driving* untuk mengurangi jumlah kecelakaan. Kemudian penggunaan metode *Speech Recognition* adalah berdasarkan jurnal karya Sanja Primorac and Mladen Russo yang berjudul *Android application for sending SMS messages with* Speech Recognition *interface*. Kedua orang tersebut mengimplementasikan *Speech Recognition* standar milik Google untuk menulis pesan SMS. Namun, dalam penelitian skripsi ini tidak akan digunakan metode *Speech Recognition* milik Android, melainkan menggunakan *library* bernama PocketSphinx yang dapat digunakan secara *offline*. Sehingga pada akhir penelitian,

akan dilakukan pengujian akurasi terhadap modul *Speech Recognition* dalam mengenali pengucapan dari beberapa kata yang sudah ditentukan, pengujian ini dilakukan untuk menentukan kelayakan aplikasi.

Berdasarkan masalah-masalah tersebut, penelitian pada skripsi ini memiliki tujuan untuk merancang dan mengimplementasikan aplikasi berbasis Android yang berguna untuk membaca notifikasi SMS dan panggilan telepon yang masuk dengan Android Text to Speech, serta melakukan pemrosesan respon suara dengan menggunakan *library* PocketSphinx. Hal itu dikarenakan penggunaan Android Text to Speech sebagai pembacanya akan membuat pengguna aplikasi dapat menentukan apakah memang perlu atau tidak untuk membuka *smartphone* miliknya yang dinilai dapat mengganggu aktivitasnya saat itu. Dengan ini diharapkan dengan diterapkannya *Text to Speech* dan *Speech Recognition* pada aplikasi pengguna dapat lebih mudah mengetahui adanya SMS dan panggilan telepon ketika sedang sibuk dan lebih bijak dalam memutuskan apa yang harus dilakukan selanjutnya terhadap SMS dan panggilan telepon masuk, sehingga diharapkan dapat mengurangi resiko terjadinya kecelakaan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat dirumuskan masalah-masalah yang akan dibahas sebagai berikut:

1. Bagaimana rancangan dan implementasi aplikasi pembaca notifikasi SMS dan panggilan telepon dengan menggunakan *engine* Android Text to Speech?
2. Bagaimana implementasi *library* PocketSphinx sebagai modul untuk pengenalan suara?
3. Bagaimana tingkat akurasi dari penerapan *library* PocketSphinx pada aplikasi?

1.3 Tujuan

Tujuan dari penelitian ini adalah untuk merancang dan mengimplementasikan aplikasi notifikasi SMS dan panggilan masuk dengan menggunakan Android Text to Speech sebagai pembaca notifikasi dan *library* PocketSphinx sebagai penerima dan pemroses respon dari pengguna.

1.4 Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah yang telah diuraikan, agar permasalahan yang dirumuskan dapat lebih terfokus, maka penelitian ini dibatasi dalam hal:

1. Modul *Text to Speech* yang digunakan adalah standar bawaan milik Android.
2. Modul *Speech Recognition* menggunakan *library* tambahan yang bernama PocketSphinx.

3. Kata yang dikenali oleh sistem hanya akan terbatas pada kata “Ya”, “Tidak”, “Satu”, “Dua”, dan “Tiga”.

1.5 Manfaat Penelitian

Adapun manfaat yang diharapkan dari hasil penelitian ini adalah sebagai berikut:

1. Membuat aplikasi yang memudahkan seseorang agar dapat mengetahui adanya SMS dan panggilan telepon ketika sedang sibuk.
2. Meningkatkan produktivitas seseorang dengan tidak membuka *smartphone* miliknya untuk hal yang belum tentu penting.

1.6 Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam laporan tugas akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dari penelitian serta sistematika penuliasan. Tahap ini merupakan landasan awal dari adanya penelitian skripsi ini.

BAB II Landasan Kepustakaan

Menguraikan tentang dasar teori dan referensi secara luas serta informasi yang dipelukan dalam pengembangan, perancangan, dan implementasi dari permasalahan yang akan dibahas.

BAB III Metodologi

Membahas tentang metode yang digunakan dalam penelitian dan langkah-langkah yang akan dilakukan dalam penulisan yang terdiri dari studi literatur, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Analisis dan Perancangan

Membahas mengenai analisis kebutuhan dan perancangan sistem berdasarkan landasan teori yang ada. Dimana tahap ini akan dijadikan sebagai acuan untuk melakukan tahap implementasi dan pengujian.

BAB V Implementasi

Membahas tentang hasil implementasi dari sistem berdasarkan metodologi yang sudah ditentukan sebelumnya, serta berdasarkan perancangan yang sudah dibuat sebelumnya.

BAB V Pengujian

Membahas tentang proses dan hasil pengujian terhadap sistem yang sudah diimplementasi. Sistem diuji dengan berdasarkan metode-metode yang sudah ditentukan pada tahap metodologi sebelumnya.

BAB VI Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangakan serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi pembahasan uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Dalam penelitian ini, akan dilakukan pembahasan mengenai *Text to Speech*, *Speech Recognition, Service*, dan *Broadcast Receiver*.

2.1 Teknologi Pemrosesan Bahasa

Bahasa dapat dibedakan menjadi dua , yaitu Bahasa Alami dan Bahasa Buatan. Bahasa alami adalah bahasa yang biasa digunakan untuk berkomunikasi antar manusia, misalnya bahasa Indonesia, Sunda, Jawa, Inggris, Jepang, dan sebagainya. Bahasa Buatan adalah bahasa yang dibuat secara khusus untuk memenuhi kebutuhan tertentu, misalnya bahasa pemodelan atau bahasa pemrograman computer (Dwi dan Huda, 2013).

Suatu sistem pemrosesan bahasa alami secara lisan dapat dibentuk dari tiga sub-sistem, yaitu sebagai berikut:

1. Sub-Sistem Natural Language Processing (NLP), berfungsi untuk melakukan pemrosesan secara simbolik terhadap bahasa tulisan. Beberapa bentuk aplikasi sub-sistem ini adalah translator bahasa alami (misalnya dari bahasa Inggris ke Bahasa Indonesia), sistem pemeriksaan sintaks bahasa, sistem yang dapat menyimpulkan suatu narasi, dan sebagainya.
2. Sub-Sistem *Text-to-Speech*, berfungsi untuk mengubah teks (bahasa tulisan) menjadi ucapan (bahasa lisan).
3. Sub-Sistem *Speech Recognition*, merupakan kebalikan teknologi *Text to Speech*, yaitu sistem yang berfungsi untuk mengubah atau mengenali suatu ucapan (bahasa lisan) menjadi teks (bahasa tulisan).

2.2 Text to Speech

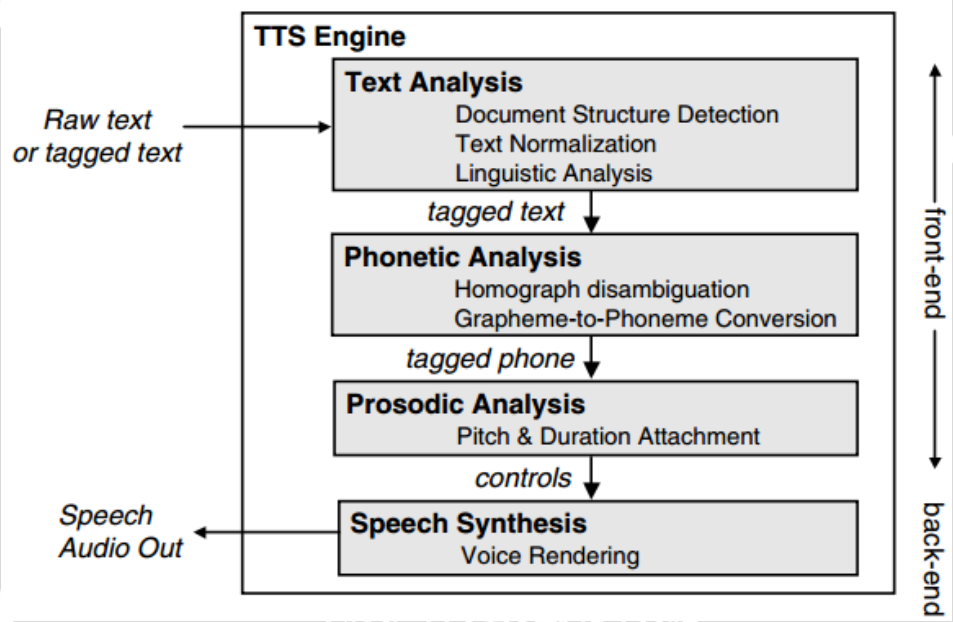
Teknologi bahasa adalah teknologi yang berhubungan dengan penggunaan bahasa, baik bahasa lisan maupun bahasa tulisan. Bahasa merupakan alat komunikasi paling relevan dan tepat sasaran untuk menyampaikan keinginan dan maksud manusia. Bentuk representasinya adalah berupa suara atau ucapan (*spoken language*), tetapi sering pula dinyatakan dalam bentuk tulisan. Sistem pemrosesan bahasa alami secara lisan dapat dibentuk dari sistem *Text to Speech*.

Tujuan dari *Text to Speech* adalah untuk melakukan konversi teks menjadi suara yang terdengar natural dan mudah dimengerti, sehingga dapat menjadi salah satu cara untuk menyampaikan informasi dari mesin kepada manusia.

Karena itu, *Text to Speech* dapat disebut juga sebagai sistem *cut-and-paste*, dan banyak digunakan pada aplikasi telekomunikasi yang berguna untuk membacakan nomor telepon (Schroeter, 2005).

Metode yang digunakan dalam *Text to Speech* adalah dengan memanfaatkan gambaran akustik dari suara untuk kemudian dilakukan sintesis, bersamaan juga dengan dilakukannya analisis bahasa dari teks untuk kemudian diekstrak bagaimana cara pengucapannya. Sistem sintesis pada *Text to Speech* secara umum dinilai berdasarkan tiga hal, yaitu:

1. Tingkat akurasi dari proses penerjemahan masukan teks.
2. Seberapa mudah dimengerti pesan suara yang dihasilkan.
3. Seberapa natural suara yang dihasilkan (suara mendekati kemiripan dengan suara manusia).



Gambar 2.1 Diagram sistem *Text to Speech* secara umum

(Sumber: Schroeter, 2005)

Diagram pemrosesan pada *Text to Speech* dapat dilihat dalam Gambar 2.1. Dapat diketahui bahwa *front-end* merupakan bagian proses masukan (*input*) ke dalam sistem, sedangkan *back-end* merupakan proses keluaran (*output*) dari sistem. Teks masukan, secara opsional dilengkapi dengan penanda yang mengontrol *prosody* atau karakteristik lain. Saat memasuki bagian *front-end,* modul analisis teks akan mendeteksi struktur dokumen seperti daftar, batas paragraph dan akhir kalimat, serta diikuti pula oleh proses normalisasi teks. Teks yang ditandai kemudian memasuki modul *phonetic analysis* yang melakukan proses *homograph disambiguation* dan konversi *grapheme-to-phoneme*. Proses ini disebut juga konversi *letter-to-sound*. Sekumpulan kata yang sudah ditandai kemudian masuk ke dalam modul yang disebut *prosodic analysis* untuk

menentukan tingkat kekerasan suara, durasi, dan amplitudo. Setelah itu kumpulan kata dari simbol-simbol yang berasal dari teks masukan akan diteruskan ke modul *speech synthesis* dimana modul tersebut akan mengendalikan proses *render* untuk menghasilkan suara yag sesuai dengan teks masukan (Schroeter, 2005).

2.2.1 Android Text to Speech

Fitur *Text to Speech* pada Android sudah tersedia sejak versi 1.6 dan mendukung beberapa bahasa seperti bahasa Inggris, bahasa Prancis, bahasa Jerman, bahasa Itali, serta bahasa Spanyol. Selain itu, *Text to Speech* pada Android juga mendukung bermacam-macam logat dalam satu bahasa. Sebagai contoh adalah bahasa Inggris, dimana dalam bahasa tersebut terdapat dua macam logat yang umum digunakan yaitu *American* dan *British*, yang dimana kedua logat tersebut sudah didukung secara penuh oleh *Text to Speech* pada Android (Android Developers Blog, 2009).

Tetapi walaupun semua perangkat Android sudah mendukung fitur *Text to Speech*, beberapa perangkat justru memiliki jumlah kapasitas penyimpanan data yang terbatas yang mengakibatkan keterbatasan sumber daya pada bahasa tertentu. Jika pengguna ingin memasang sumber daya bahasa yang tidak tersedia tersebut, API pada Android Text to Speech mampu menyediakan kemampuan pada aplikasi untuk memeriksa atau mencari bahasa tersebut, apabila bahasa memang tersedia maka pengguna dapat mengunduhnya secara langsung. Kode 2.1 adalah contoh *source code* untuk melakukan pemeriksaan terhadap ketersediaan bahasa.

```
1  Intent checkIntent = new Intent();
2  checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
3  startActivityForResult(checkIntent, MY_DATA_CHECK_CODE);
```

Kode 2.1 *Source code* untuk melakukan pemeriksaan ketersediaan bahasa

Pemeriksaan yang sukses akan ditandai dengan adanya balasan berupa kode hasil CHECK_VOICE_DATA_PASS, mengindikasikan perangkat tersebut sudah siap menjalankan Android Text to Speech. Tetapi jika gagal, pengguna harus tahu bahwa instalasi data sumber daya bahasa perlu dilakukan dengan cara mengunduhnya langsung dari Goole Play Store. Sehingga pengembang aplikasi perlu menggunakan *intent* ACTION_INSTALL_TTS_DATA untuk melakukan proses tersebut.

Perlu diketahui pula bahwa ada hal lain yang merupakan bagian penting dari Android Text to Speech, yaitu bahasa dan *locale*. Sebagai contoh, untuk mengatur bahasa menjadi bahasa Inggris dengan pembicara orang Amerika, maka digunakan *script* seperti yang terlihat dalam Kode 2.2.

```
1  tts.setLanguage(Locale.US);
```

Kode 2.2 *Source code* untuk pilihan bahasa dan *locale* yang akan digunakan

Locale merupakan cara untuk menentukan bahasa, karena faktanya bahasa yang sama dapat berbeda cara pengucapannya antara suatu negara dengan negara lain, atau hal ini sering kita sebut sebagai logat.

Setelah semua persiapan sebelumnya sukses dilakukan, Android Text to Speech sudah dapat digunakan. Dengan menggunakan method speak() seperti dalam Kode 2.3, maka teks yang terdapat pada variabel myText1 dan myText2 dapat diubah menjadi suara.

```
1  String myText1 = "Did you sleep well?";
2  String myText2 = "I hope so, because it's time to wake up.";
3  tts.speak(myText1, TextToSpeech.QUEUE_FLUSH, null);
4  tts.speak(myText2, TextToSpeech.QUEUE_ADD, null);
```

Kode 2.3 *Source code* dengan fungsi speak() untuk membuat aplikasi berbicara sesuai *masukan* teks

Perlu diketahui bahwa fungsionalitas Android Text to Speech berjalan sebagai Service dalam sistem operasi Android, dimana dapat digunakan oleh aplikasi apapun yang memanfaatkan fiturnya. Sebagai pengembang aplikasi yang baik sangat dianjurkan untuk mematikan Service tersebut apabila tidak digunakan lagi, cukup dengan menggunakan perintah tts.shutdown() didalam method onDestroy() pada *Activity.*

2.3 Speech Recognition

Speech Recognition adalah mesin yang bekerja berdasarkan pengucapan kata untuk kemudian dilakukan identifikasi dan merespon reaksi selanjutnya. Teknologi tingkat tinggi ini mampu mengubah sinyal suara menjadi teks atau perintah melalui proses identifikasi dan penalaran. Pengenalan teknologi ini membutuhkan komputasi matematika pada saat mengenali suara yang diucapkan oleh manusia, sehingga kebutuhan akan pengaksesan yang cepat dan langsung (*realtime*) sangat dibutuhkan pada pengembangan teknologi ini. Oleh karena itu, modul ini akan diimplementasikan untuk bekerja secara *offline* sehingga tidak menghambat komputasi pada saat mengenali suara seseorang.

Bidang ilmu Speech Recognition sudah ada selama 60 tahun. Ada banyak kemajuan dalam perkembangannya sejak ditemukannya *speech recognizer* pertama di Bell Labs pada awal tahun 1950-an. Perkembangan *Speech Recognition* meningkat secara bertahap sampai penemuan Hidden Markov Model (HMM) pada awal tahun 1970. HMM adalah salah satu pendekatan statistik yang paling dominan dan sudah diterapkan selama bertahun-tahun. Teori dasar HMM diterbitkan dalam serangkaian makalah klasik oleh Baum dan rekan-rekannya yang kemudian diimplementasikan untuk aplikasi pengenalan suara oleh Baker di Carnegie Mellon University (CMU) dan oleh Jelinek dan rekan-rekannya di IBM di tahun 1970-an (Rabiner dan Juang 1993).

Penerapan HMM pada pengenalan suara berfungsi untuk memprediksi suara yang diucapkan berdasarkan *dictionary file*, *language model*, dan *acoustic model*.

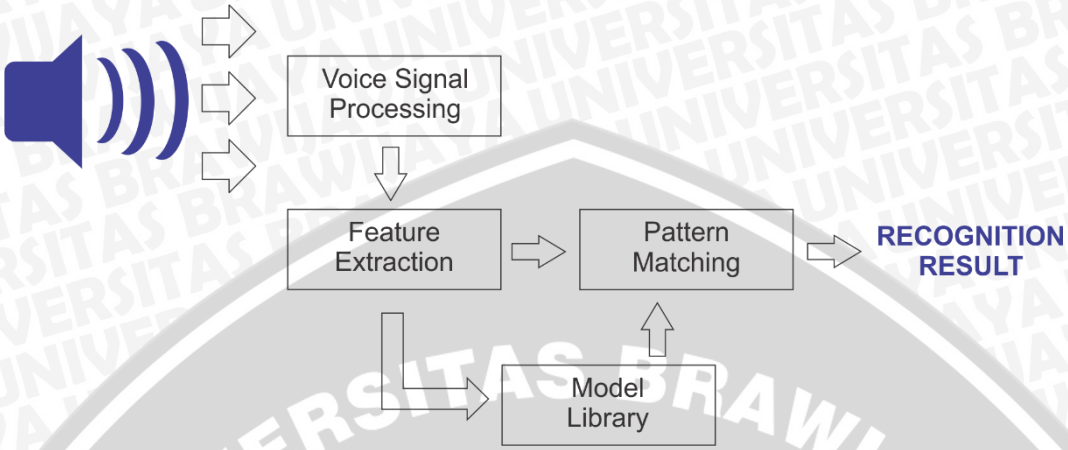
HMM akan memberikan probabilitas berdasarkan kosakata yang akan diucapkan. Probabilitas lalu dihitung berdasarkan pembobotannya sehingga mempunyai nilai terbaik yang memungkinkan kata tersebut dikeluarkan berdasarkan persamaan suara. Pencarian probabilitas terbaik dari masing-masing suku kata diselesaikan dengan Algoritma Viterbi.

Sistem pengenalan ucapan adalah sistem yang berfungsi untuk mengubah bahasa lisan menjadi bahasa tulisan. Masukan sistem ini adalah ucapan manusia, selanjutnya sistem akan mengidentifikasikan kata atau kalimat yang diucapkan dan menghasilkan teks yang sesuai dengan apa yang diucapkan. Penganalisis sintaks akan melakukan transformasi sinyal ucapan dari domain waktu ke domain frekuensi. Pada domain frekuensi, untuk kurun waktu yang singkat, setiap sinyal dapat terlihat memiliki ciri-ciri yang unik. Namun biasanya, pengucapan yang diucapkan oleh seseorang seringkali bervariasi, yakni terpengaruh oleh fonem, kondisi emosi, *noise* dan faktor-faktor lainnya. Sistem *Speech Recognition* akan melakukan pengenalan untuk setiap unit bunyi pembentuk ucapan (fonem), selanjutnya mencoba mencari kemungkinan kombinasi hasil ucapan yang paling dapat dikenali. Sinyal ucapan pertama kali akan dilewatkan pada bagian penganalisis ucapan untuk mendapatkan besaran-besaran atau ciri-ciri yang mudah diolah pada tahap berikutnya. Untuk setiap ucapan yang berbeda akan dihasilkan pola ciri yang berbeda (Munawar 2010).

Secara umum pengenalan suara dibagi 2 jenis diantaranya pengenalan pembicara dan pengenalan ucapan. Pengenalan pembicara merupakan proses untuk mengenali siapa yang sedang berbicara atau mengucapkan informasi berdasarkan pola suara yang diinputkan. Teknologi ini sangat ampuh untuk mengidentifikasi suara seseorang dan bermanfaat untuk digunakan pada layanan seperti control *smarthome*, keamanan, layanan informasi dan akses terhadap keadaan tertentu. Jenis yang kedua yaitu, pengenalan ucapan yang didefinisikan sebagai proses pengubahan sinyal suara menjadi dalam bentuk teks. Pengenalan ucapan ini memiliki kemampuan untuk mencocokkan suara terhadap data yang ada seperti rekaman ataupun pembendaharaan kata (Meng dan Zhang, 2012).

Seperti yang terlihat dalam Gambar 2.2, *Speech Recognition* pada dasarnya adalah sistem pengenalan pola, termasuk didalamnya terdapat fitur ekstraksi, pencocokan pola, serta model *library*. Suara masukan yang tidak dikenali akan diubah menjadi sinyal listrik pada sistem identifikasi. Sistem kemudian akan menetapkan model suara sesuai dengan karakteristik suara manusia, dengan menganalisis sinyal suara masukan dan hasil ekstraksi dari fitur yang diperlukan. Komputer digunakan dalam proses pengenalan sesuai dengan model pengenalan suara untuk membandingkan *template* suara yang tersimpan dalam komputer dan karakteristik sinyal suara masukan. Dengan melakukan pencarian dan pencocokan untuk mengidentifikasi berbagai suara optimal yang cocok dengan *template*. Metode pengenalan suara yang banyak digunakan diantaranya Dynamic Time Warping (DTW), Hidden Markov Model (HMM), Vector Quantization (VQ), Artificial

Neural Network (ANN), Support Vector Machine (SVM) dan sebagainya (Meng dan Zhang, 2012).



Gambar 2.2 Proses pengenalan pola pada *Speech Recognition*

(Sumber: Meng dan Zhang, 2012)

2.3.1 PocketSphinx

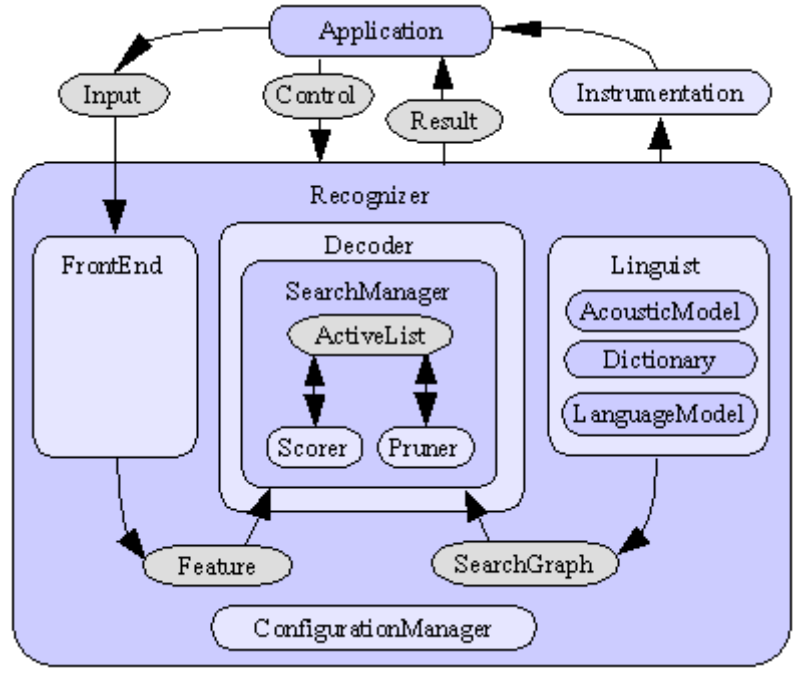
Pocketsphinx merupakan *library* pengenalan ucapan dari sistem Sphinx yang dirancang oleh Carnegie Mellon University (CMU). Dengan kata lain, fungsi *library* Pocketsphinx itu sendiri adalah sebagai *engine* atau *library* yang memudahkan developer yang ingin mengembangkan aplikasi di bidang *Speech Recognition*. *Library* ini menyediakan fasilitas kepada para penggunaanya untuk memodifikasi bahasa yang akan digunakan untuk mengenali suara seseorang. Aplikasi PocketSphinx merupakan sistem pengenalan suara yang biasa digunakan untuk aplikasi desktop, tetapi CMU juga mengembangkan *library* ini untuk digunakan pada *platform* Android. Terlihat dalam Gambar 2.3 yang merupakan gambaran umum dari arsitektur PocketSphinx (Ferdiansyah dan Purwarianti, 2012).

Secara *default*, *library* ini hanya akan mengenali kata-kata dalam Bahasa Inggris. Sehingga untuk membuatnya agar dapat mengenali kata dalam Bahasa Indonesia, diperlukan proses *training* dengan memasukkan kata Bahasa Indonesia kedalam sebuah *dictionary* dan kemudian mendefinisikan bagaimana cara mengucapkan kata tersebut (Ferdiansyah dan Purwarianti, 2012).

```
1   abdominal AE B D AA M AH N AH L
2   abdominal(2) AH B D AA M AH N AH
3   L
4   abduct   AE   B   D   AH   K   T
5   abducted AE B D AH K T IH D
6   abducted(2) AH B D AH K T IH D
7   abductee AE B D AH K T IY
8   abductees AE B D AH K T IY Z
9   abducting AE B D AH K T IH NG
10  abducting(2) AH B D AH K T IH NG
11  abduction AE B D AH K SH AH N
```

Kode 2.4 Contoh bagian dari isi *file dictionary*

Seperti yang terlihat dalam Kode 2.4, contoh *dictionary* tersebut memiliki isi berupa daftar kata beserta bagaimana cara pengucapan kata tersebut. Selain itu, perlu diketahui bahwa contoh *dictionary* tersebut menggunakan Bahasa Inggris.



Gambar 2.3 Arsitektur PocketSphinx

(Sumber: Ferdiansyah dan Purwarianti, 2012)

2.4 Service

Service adalah komponen aplikasi yang dapat melakukan operasi secara tidak terlihat (*background*) serta tidak menyediakan tampilan antarmuka. *Service* yang dijalankan oleh suatu komponen aplikasi akan dapat terus berjalan di *background* walaupun pengguna berganti aplikasi. Selain itu, komponen akan tetap terikat dengan *service* yang menyebabkan keduanya dapat saling berinteraksi. Contoh penerapan dari Service diantaranya adalah pemutar musik serta proses *download* dan *upload* pada jaringan (Android Developers, 2015).

Service yang sedang berjalan akan mendapatkan prioritas tertinggi untuk tidak dimatikan oleh *resource management* milik Android dibandingkan dengan Activity yang sedang dalam kondisi tidak aktif maupun tidak terlihat. Saat dimana Android akan menghentikan *service* adalah apabila terdapat Activity yang sedang membutuhkan sumberdaya, jika hal itu terjadi maka *service* akan dijalankan ulang ketika sumberdaya sudah tersedia.

Service memiliki dua bentuk kondisi, yaitu:

1. *Started*

   Service dimulai (*started*), ketika komponen aplikasi seperti *Activity* menjalankannya dengan memanggil *method* startService(). Sekali dijalankan,

service akan terus berjalan di *background* walaupun komponen yang menjalakannya sudah tidak ada ataupun sudah dimatikan. Biasanya, *service* yang berjalan akan melakukan operasi tunggal dan tidak mengembalikan hasil kepada pemanggilnya (*caller*).

2. *Bound*

   Service berada pada kondisi ini ketika komponen aplikasi mengikatnya dengan memanggil *method* bindService(). *Service* yang terikat akan menghasilkan suatu hubungan layaknya *client-server* yang membuat keduanya dapat saling berinteraksi. *Service* yang terikat hanya akan berjalan selama ada komponen aplikasi yang terikat dengan *service* tersebut, bahkan komponen aplikasi lain. Jadi dapat dikatakan komponen yang mengikat suatu *service* dapat berjumlah lebih dari satu, dan ketika semua komponen tersebut memutus ikatannya maka *service* akan mati atau hancur.

Dua kondisi yang sudah dijelaskan tersebut dapat dikatakan merupakan dua macam daur hidup atau siklus dari *service*. Daur hidup atau siklus dengan kondisi *started* disebut juga sebagai *Unbounded Service*, sedangkan kondisi *bound* disebut juga sebagai *Bounded Service*. Seperti yang terlihat dalam Gambar 2.4, yang membedakan kedua siklus tersebut adalah bagaimana *service* itu dibuat, *Unbounded Service* dibuat dengan menggunakan *method* startService() dan *Bounded Service* dibuat dengan menggunakan *method* bindService(). Selain itu sebelum mematikan Bounded Service dengan perintah onDestroy(), perlu digunakan *method* onUnbind() terlebih dahulu (Meier, 2009).

Untuk membuat *service* diperlukan subkelas dari *service* itu sendiri, yang berguna untuk menangani aspek utama dari daur hidup *service* dan menyediakan mekanisme bagi komponen untuk terikat pada *service* (Android Developers, 2015). Untuk melakukannya, maka digunakan *callback method*, diantaranya adalah:

1. onStartCommand()

   Sistem memanggil *method* ini ketika komponen lain seperti Activity meminta keterangan bahwa *service* akan dimulai dengan memanggil startService(). Sekali *method* ini dijalankan, *service* akan dimulai dan berjalan secara *background*. Jika *method* ini diimplementasikan, maka *service* perlu dihentikan apabila pekerjaan telah selesai, dengan memanggil stopSelf() atau stopService().

2. onBind()

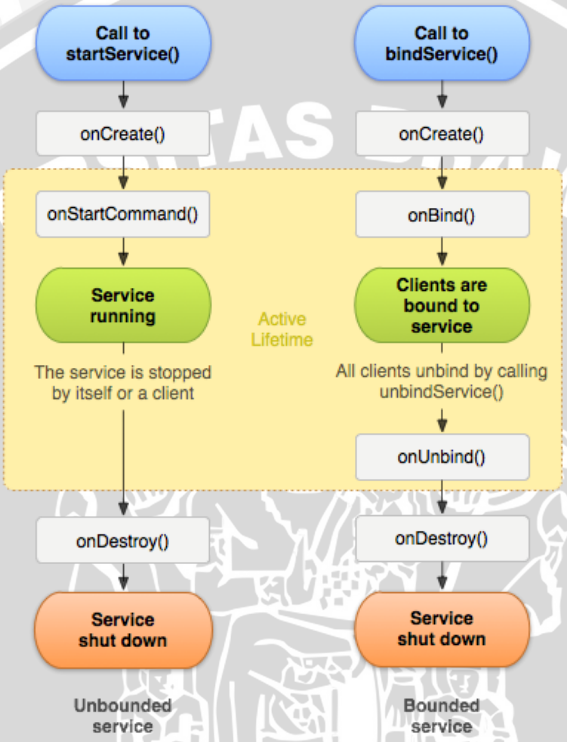
   Sistem memanggil *method* ini ketika komponen lain ingin terikat dengan *service*, dengan memanggil bindService(). Apabila *method* ini diimplementasikan, maka diperlukan sebuah antarmuka yang berguna agar klien dapat melakukan komunikasi dengan *service*.

3. onCreate()

Sistem memanggil *method* ini ketika *service* pertama kali dibuat. Apabila *service* sudah dibuat maka *method* ini tidak perlu dipanggil.

4. onDestroy()

   Sistem memanggil *method* ini ketika *service* tidak lagi digunakan dan apabila akan dihancurkan. Dalam pembuatan aplikasi yang menggunakan *service*, *method* ini perlu digunakan untuk membersihkan sumberdaya yang telah digunakan seperti *thread*, *receiver*, dan lain-lain.



Gambar 2.4 Siklus hidup pada *service*

(Sumber: Android Developers, 2015)

2.5 Broadcast Receiver

Broadcast Receiver merupakan sebuah komponen yang merespon terhadap siaran (*broadcast*) pengumuman yang dikeluarkan oleh sistem. Pesan yang disiarkan disebut sebagai *event* atau *intent*. Aplikasi melakukan *broadcast* untuk memberi tahu aplikasi lainnya bahwa beberapa data telah di unduh ke perangkat yang tersedia, lalu penerima *broadcast* akan menerima pesan untuk melakukan transfer informasi. Contohnya adalah *broadcast* yang memberitahukan bahwa layar sudah mati, notifikasi SMS masuk, dan panggilan telepon. Selain itu, aplikasi yang dibuat juga bisa memulai *broadcast*, misalnya memberitahukan aplikasi lain bahwa beberapa data sudah selesai diunduh dan bisa digunakan. *Broadcast Receiver* tidak memiliki antarmuka layaknya Service, namun *Broadcast Receiver* mampu menampilkan notifikasi di *status bar* untuk memberitahukan user kalau sedang terjadi *broadcast*. Perlu diketahui bahwa proses yang sedang

mengeksekusi *Broadcast Receiver* akan menjadi proses *foreground* dan akan tetap berjalan di sistem kecuali terjadi kekurangan memori yang sangat ekstrim (Android Developers, 2015).

Ada dua kelas utama dari *broadcast* yang dapat diterima, yaitu:

1. *Normal Broadcast*

   Semua penerima *broadcast* berjalan tanpa urutan yang didefinisikan terlebih dahulu, karena seringkali terjadi pada waktu yang bersamaan. Ini lebih efisien, tetapi akan menyebabkan penerima tidak bisa menggunakan hasil ataupun membatalkan API yang digunakan.

2. *Ordered Broadcast*

   Broadcast dikirim kepada penerima setiap satu waktu. Urutan penerima dapat dikendalikan dengan menggunakan atribut android:priority.

Salah satu kegunaan *Broadcast Receiver* adalah untuk mengetahui adanya SMS yang masuk. Kelas BroadcastReceiver akan membuat aplikasi menerima *intent* yang dikirim oleh aplikasi lain menggunakan *method* sendBroadcast(). Ketika ada SMS yang masuk, *method* onReceive() akan langsung dieksekusi. Pesan SMS berada di dalam objek *intent* dengan melalui objek *Bundle*, serta disimpan dalam bentuk objek *array* dengan format PDU. Untuk mengekstrak setiap pesan, digunakan *method* createFromPdu() dari kelas SmsMessage. Setelah itu pesan akan ditampilkan menggunakan *toast*. Nomor telepon pengirim diambil dengan menggunakan *method* getOriginatingAddress(). Broadcast Receiver untuk SMS ini akan terus aktif memeriksa adanya SMS yang masuk walaupun aplikasi sudah tidak berjalan (Meier, 2009).

Dalam Kode 2.5, terlihat *source code* yang dapat digunakan untuk menerima SMS yang masuk. Kelas tersebut turunan dari kelas BroadcastReceiver.

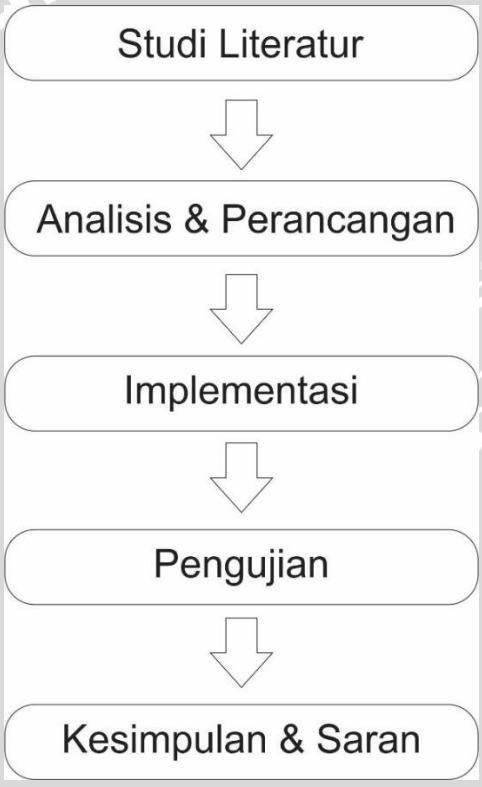
```
1  packagenet.learn2develop.SMS;
2  importandroid.content.BroadcastReceiver;
3  importandroid.content.Context;
4  importandroid.content.Intent;
5  importandroid.os.Bundle;
6  importandroid.telephony.SmsMessage;
7  importandroid.widget.Toast;
8  public classSMSReceiver extends BroadcastReceiver
9  {
10  @Override
11   public voidonReceive(Context context, Intent intent)
12   {
13 Bundle bundle = intent.getExtras();
14       SmsMessage[] msgs = null;
15       String str = "";
16       if(bundle != null)
17       {
18   Object[] pdus = (Object[]) bundle.get("pdus");
19           msgs = newSmsMessage[pdus.length];
20           for(inti=0; i<msgs.length; i++){
21           msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
22          str += "SMS from "+ msgs[i].getOriginatingAddress();
23          str += " :";
```

```
24              str += msgs[i].getMessageBody().toString();
25              str += "\n";
26              }
27          Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
28          }
29      }
30  }
```

Kode 2.5 *Source code* untuk menerima SMS yang masuk

BAB 3 METODOLOGI

Pada bab ini, menjelaskan tentang langkah-langkah yang akan dilakukan dalam penelitian ini. Langkah awal yang perlu dilakukan yaitu, mengumpulkan teori-teori pendukung dan mengemasnya sebagai studi literatur, yang kemudian dilanjutkan dengan melakukan proses analisis kebutuhan serta melakukan perancangan. Setelah kedua hal tersebut dilakukan, langkah selanjutnya yaitu melakukan implementasi berdasarkan rancangan sebelumnya untuk kemudian dilakukan pengujian terhadap implementasi tersebut. Terakhir, kesimpulan dan saran disertakan sebagai catatan atas sistem dan kemungkinan arah pengembangan sistem selanjutnya. Gambar 3.1 menunjukkan diagram tahapan penelitian secara umum.



Gambar 3.1 Tahapan Penelitian

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

1. *Text to Speech* (Android Text to Speech)
2. *Speech Recognition* (PocketSphinx)
3. *Service*
4. *Broadcast Receiver*

Pada tahap ini dilakukan studi terhadap literatur-literatur seperti yang sudah disebutkan sebelumnya, hal ini berguna sebagai acuan dasar pada penelitian skripsi ini serta serta menambah pengetahuan terhadap penelitian yang sedang dilakukan.

3.2 Analisis dan Perancangan

Dalam tahapan ini diperlukan gambaran tentang bagaimana jalannya aplikasi dan analisa tentang kebutuhan sistem yang dibutuhkan, yaitu menjelaskan aktor yang terlibat dan interaksinya terhadap aplikasi. Analisa kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan sistem untuk pengguna yang kemudian akan dimodelkan dalam bentuk diagram Use Case.

Setelah kebutuhan-kebutuhan dari sistem telah terkumpul dari proses analisis kebutuhan, kemudian akan dilakukan tahap perancangan sistem. Sistem perancangan aplikasi pada penelitian ini adalah berdasarkan orientasi objek. Perancangan akan terdiri dari empat model perancangan, yaitu perancangan alur proses sistem yang dimodelkan dengan perancangan diagram Activity, perancangan interaksi yang dimodelkan dengan diagram Sequence, perancangan diagram Class serta perancangan antarmuka sistem.

3.3 Implementasi

Implementasi merupakan tahap pembangunan sistem dan dibuat berdasarkan pada tahap perancangan aplikasi yang telah dilakukan sebelumnya. Implementasi dilakukan secara *native*, dengan menggunakan bahasa pemrograman Java yang ditulis dengan IDE Android Studio. Seperti yang sudah disebutkan pada studi literatur, aplikasi akan menggunakan *Text to Speech* standar milik Android sebagai pembaca notifikasi serta menggunakan *library* PocketSphinx sebagai *Speech Recognition* untuk menerima dan mengenali respon. Selain itu aplikasi akan dijalankan pada *background* sebagai *service* serta mengandalkan *Broadcast Receiver* untuk mengetahui adanya SMS ataupun panggilan yang masuk.

3.4 Pengujian

Tujuan dari pengujian perangkat lunak adalah untuk menemukan dan memperbaiki sebanyak mungkin kesalahan dalam program, karena salah satu pengujian yang baik adalah pengujian yang memiliki probabilitas yang tinggi dalam menemukan kesalahan perangkat lunak. Selain itu, pada tahap pengujian juga akan dilakukan pengujian untuk mengetahui terpenuhinya kinerja dan performa sistem sesuai dengan kebutuhan yang telah ditentukan sebelumnya. Setelah tahap pengujian, dilakukan analisis untuk mengetahui hasil dari pengujian perangkat lunak sehingga bisa didapatkan kesimpulan dan sistem yang telah dibuat. Ada tiga macam pengujian yang dilakukan pada aplikasi ini antara lain:

1. Pengujian Unit

   Pengujian unit menggunakan metode White Box yang berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak, yaitu modul. Teknik yang digunakan dalam melakukan pengujian unit ini adalah teknik Basis Path Testing, dengan memodelkan algoritma pada suatu *flowgraph*, kemudian menetukan jumlah kompleksitas siklomatis (*cyclomatic complexity*), basis set dari jalur independen dan memberikan kasus uji (*test case*) pada setiap basis set yang telah ditentukan.

2. Pengujian Validasi

   Pengujian validasi dengan menggunakan metode Black Box untuk memeriksa hasil eksekusi aplikasi dan fungsionalitasnya. Kesesuain fitur-fitur aplikasi dengan kebutuhan sistem dilakukan untuk mengetahui validitas aplikasi. Pengujian validasi dinyatakan berhasil apabila aplikasi berfungsi dengan cara sesuai dengan yang diharapkan.

3. Pengujian Akurasi

   Pengujian ini dilakukan dengan menghitung tingkat akurasi terhadap kemampuan aplikasi mengenali kata-kata yang sudah ditetapkan, yaitu "Ya", "Tidak", "Satu", "Dua", dan "Tiga". Penentuan nilai akurasi akan dilakukan dengan menghitung nilai *precision*, *recall*, dan F-measure.

Sebagai tambahan, Tabel 3.1 merupakan spesifikasi komputer yang akan digunakan untuk melakukan proses penyusunan *source code* dan pengembangan aplikasi dan Tabel 3.2 merupakan spesifikasi perangkat lunak yang digunakan selama proses pengembangan aplikasi.

Tabel 3.1 Spesifikasi komputer

Sistem Operasi	Windows 8.1
System Model	ASUS A43TA
CPU	AMD A6-3400M Quad-core 1.40 Ghz
Memory	8192 MB RAM
GPU	AMD Radeon HD 6650M

Tabel 3.2 Spesifikasi *smartphone*

Sistem Operasi	Android 5.0
System Model	Sony Xperia M4 Aqua
CPU	Octa-core 1.5 GHz Cortex-A53
Memory	2 GB RAM
GPU	Adreno 405

3.5 Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan analisis, perancangan, dan implementasi serta pengujian telah selesai dilaksanakan. Kesimpulan diambil dari hasil implementasi aplikasi yang telah dibangun. Tahap terakhir penulisan adalah saran yang dimaksudkan untuk memperbaiki kekurangan yang terjadi dan menyempurnakan penulisan serta mengembangkan penelitian lebih lanjut jika diperlukan.

BAB 4 ANALISIS DAN PERANCANGAN

Pada bab ini menjelaskan tentang analisis sistem dan desain dari aplikasi yang akan diteliti. Analisis dan desain sistem memiliki dua tahap, yaitu tahap pertama adalah proses analisis kebutuhan dan tahap kedua adalah proses perancangan perangkat lunak. Tahap analisis kebutuhan terdiri dari gambaran umum alur aplikasi, mendaftar kebutuhan yang diperlukan, serta memodelkan ke dalam diagram Use Case. Pada tahap perancangan, terdiri dari pemodelan diagram Activity untuk menggambarkan proses dan urutan aktivitas dalam sebuah proses, pemodelan diagram Sequence untuk menggambarkan interaksi antar objek, pemodelan diagram Class untuk menggambarkan perancangan struktur kelas yang menyusun aplikasi, serta perancangan antarmuka.

4.1 Analisis Kebutuhan

4.1.1 Gambaran Umum Aplikasi

Aplikasi pembaca SMS dan notifikasi panggilan telepon ini merupakan aplikasi yang dirancang untuk mempermudah para pengendara dan orang-orang sibuk agar dapat menentukan seberapa penting *smartphone* miliknya untuk dibuka. Aplikasi akan membacakan notifikasi SMS dan atau panggilan telepon, untuk kemudian meminta respon selanjutnya kepada pengguna. Pembacaan notifikasi akan memanfaatkan *Text to Speech*, sedangkan untuk menerima respon pengguna akan digunakan *Speech Recognition*. Perlu diketahui, pada aplikasi ini terdapat konfigurasi sistem yang dapat diatur sesuai kebutuhan pengguna seperti seberapa cepat notifikasi dibaca dan bagaimana isi *template* yang digunakan untuk membalas pengirim.

Seperti yang sudah dijelaskan pada batasan masalah, aplikasi hanya akan mengenali kata “Ya”, “Tidak”, “Satu”, “Dua”, dan “Tiga”. Hal itu dikarenakan semua kata tersebut sudah secara umum mewakili semua jawaban yang diperlukan untuk menjawab pertanyaan yang diberikan oleh aplikasi.

Gambaran proses kerja dari kedua modul tersebut, yaitu:

1. Aplikasi ini memanfaatkan *Text to Speech* sebagai pembaca notifikasi. Digunakannya Android Text to Speech adalah karena modul tersebut mampu digunakan secara *offline*, selain itu modul tersebut juga sudah terintegrasi di dalam sistem operasi Android. Oleh karena itu, Android Text to Speech merupakan salah satu komponen penting dalam sistem operasi Android, dan modul tersebut dapat berjalan sebagai Service tersendiri. Sehingga ketika aplikasi ini dijalankan, tidak hanya Service aplikasi itu sendiri yang berjalan secara *background*, tetapi juga Service Android Text to Speech.
2. Untuk menerima respon dari pengguna, maka digunakanlah *library* PocketSphinx. *Library* tersebut merupakan sebuah modul *Speech Recognition*

yang akan mengenali suara pengguna untuk menentukan bagaimana tindakan selanjutnya yang harus dilakukan aplikasi, selain itu alasan digunakannya PocketSphinx adalah karena *library* tersebut dapat digunakan secara *offline*. Seperti yang sudah dijelaskan pada bab Dasar Teori, *library* ini memiliki sebuah *file* yang berisi kumpulan kata beserta bagaimana cara pengucapan kata tersebut. Pada Tabel 4.1, terlihat daftar semua kata yang akan digunakan dan dikenali oleh aplikasi. Kata "Ya" dan "Tidak", berguna untuk menjawab pertanyaan umum yang ditanyakan oleh aplikasi, sedangkan kata "Satu", "Dua", dan "Tiga" digunakan untuk menjawab pilihan terhadap pertanyaan mengenai *template* nomor berapa yang akan digunakan (Ferdiansyah dan Purwarianti, 2012).

Tabel 4.1 Daftar kata yang akan dikenali aplikasi

Kata	Cara Pengucapan
Ya	Y AA
Tidak	T IH D AH K
Satu	S AH T UH
Dua	D UH W AA
Tiga	T IH G AH

4.1.2 Daftar Kebutuhan

Tabel 4.2 Tabel kebutuhan fungsional

Kode	Kebutuhan	Use Case
KF01	Aplikasi mengetahui serta membacakan SMS dan panggilan telepon yang masuk.	Mendengarkan notifikasi.
KF02	Aplikasi memeriksa adanya SMS yang masuk.	Notifikasi SMS masuk.
KF03	Aplikasi memeriksa adanya panggilan telepon masuk.	Notifikasi panggilan telepon masuk.
KF04	Aplikasi menerima respon dari pengguna.	Merespon terhadap SMS atau panggilan masuk.
KF05	Aplikasi harus memiliki konfigurasi yang dapat diatur pengguna.	Mengatur konfigurasi aplikasi.
KF06	Aplikasi menyediakan menu membuat *template* pesan untuk membalas pengirim.	Membuat *template* pesan.

Daftar kebutuhan terdiri dari kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional ditunjukan dengan kode KFXX. Tabel 4.2 merupakan tabel

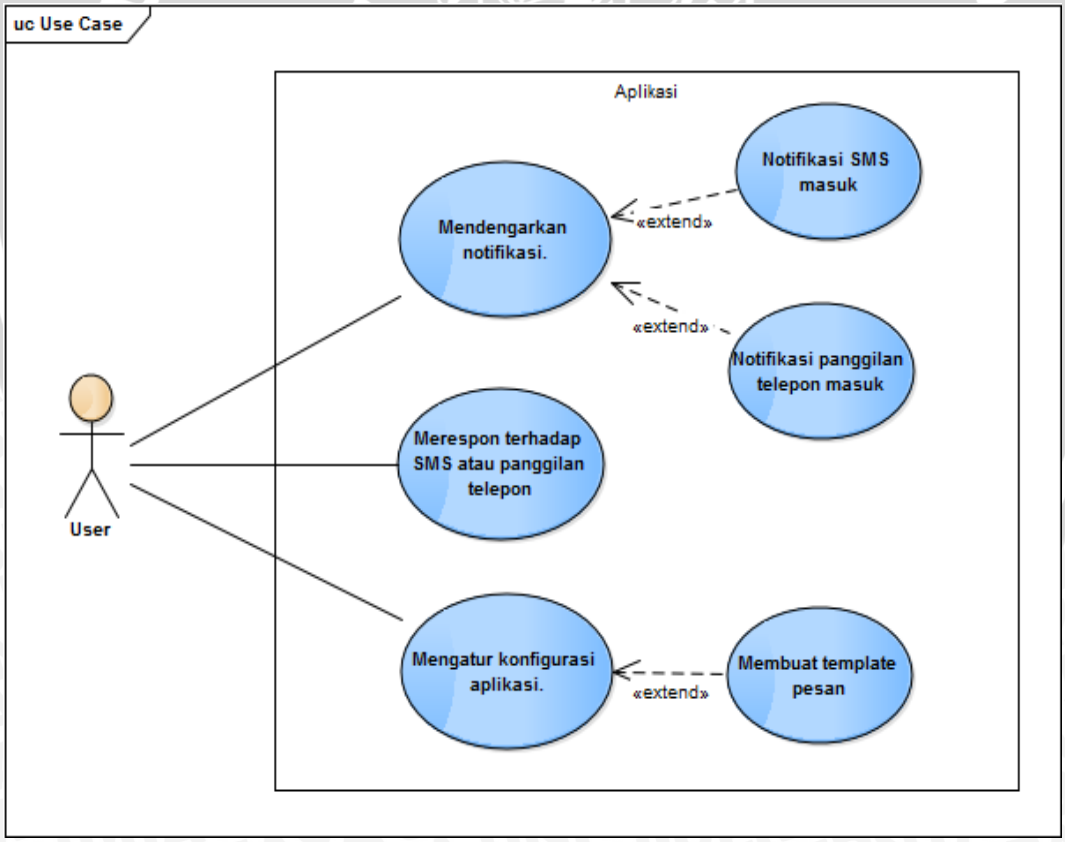
kebutuhan fungsional yang terdiri dari enam kebutuhan sistem yang harus dipenuhi, sedangkan Tabel 4.3 merupakan daftar kebutuhan non-fungsional yang harus dipenuhi.

Tabel 4.3 Tabel kebutuhan non-fungsional

Parameter	Deskripsi Kebutuhan
Compatibility	Aplikasi harus dapat dijalankan pada sistem operasi Android dengan versi minimal 4.0.
Usability	Aplikasi harus dapat digunakan dengan mudah oleh pengguna, dimana pengguna dapat mengetahui dan menggunakan fitur-fitur yang ada pada aplikasi.

4.1.3 Use Case Diagram

Pemodelan Use Case diperoleh dari kebutuhan fungsional yang digunakan untuk menggambarkan interaksi antara aktor dengan sistem yang akan dibuat. Use Case aplikasi dapat dilihat dalam Gambar 4.1.



Gambar 4.1 Use Case aplikasi

Tabel 4.4 menjelaskan mengenai kegiatan yang dilakukan pada saat mendengarkan pembacaan notifikasi. Pertama pengguna harus membuka

aplikasi, setelah itu aplikasi akan berjalan sebagai *service* untuk selalu melakukan pemeriksaan terhadap SMS dan panggilan telepon yang masuk.

Tabel 4.4 Skenario Use Case mendengarkan notifikasi

Kode *Use Case*	**KF01**
Nama *Use Case*	Mendengarkan notifikasi.
Persyaratan Kontek	Aplikasi sudah terpasang pada *smartphone* pengguna.
Tujuan Dalam Kontek	Aplikasi akan membacakan notifikasi SMS dan panggilan masuk untuk mempermudah pengguna.
Prakondisi	Aplikasi sudah terpasang dan berjalan sebagai *service*
Kondisi Akhir Sukses	Aplikasi membacakan notifikasi SMS dan panggilan masuk.
Kondisi Akhir Gagal	-
Aktor	Pengguna
Alur Utama	1. Pengguna membuka aplikasi 2. Menjalankan aplikasi sebagai *service.* 3. Aplikasi mambacakan notifikasi apabila ada SMS atau panggilan masuk

Tabel 4.5 menjelaskan proses pengguna untuk memberikan respon terhadap SMS atau panggilan telepon. Setelah aplikasi berjalan sebagai *service*, aplikasi akan membacakan notifikasi dan kemudian meminta respon dari pengguna untuk menentukan tindakan apa yang harus dilakukan selanjutnya .

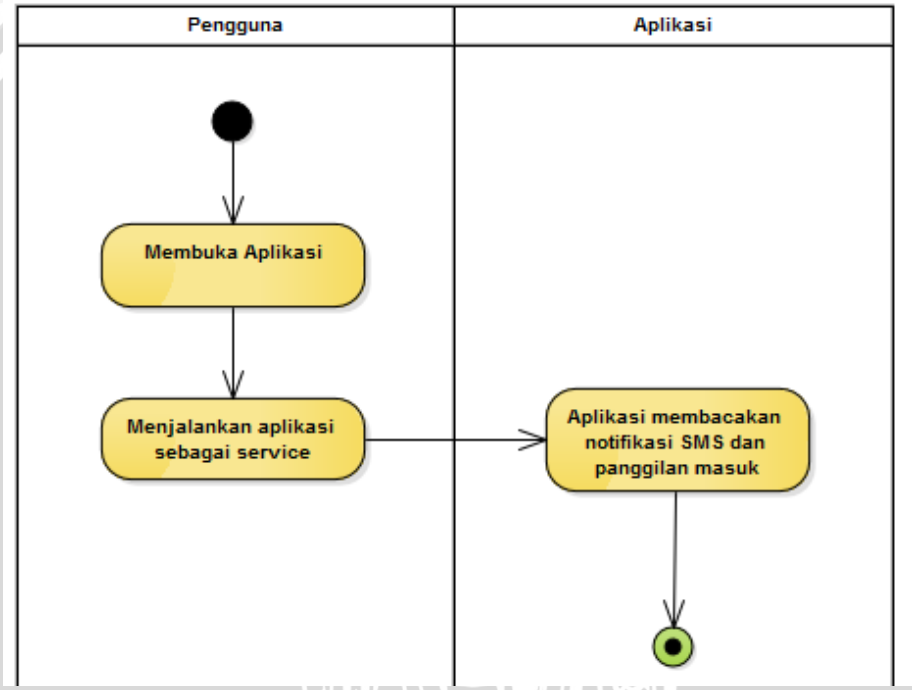
Tabel 4.5 Skenario Use Case merespon terhadap SMS atau panggilan telepon

Kode *Use Case*	**KF04**
Nama *Use Case*	Merespon terhadap SMS atau panggilan telepon.
Persyaratan Kontek	Aplikasi sudah terpasang pada *smartphone* pengguna.
Tujuan Dalam Kontek	Pengguna memberikan respon kepada aplikasi untuk menentukan tindakan selanjutnya.
Prakondisi	Aplikasi sudah terpasang dan dijalankan.
Kondisi Akhir Sukses	Aplikasi mengenali respon dari pengguna.
Kondisi Akhir Gagal	Aplikasi tidak mengenali respon dari pengguna.
Aktor	Pengguna
Alur Utama	1. Aplikasi sudah berjalan sebagai *service*. 2. Aplikasi meminta respon kepada pengguna. 3. Pengguna memberi respon dengan mengucapkan kata. 4. Aplikasi mengenali atau tidak mengenali respon.

4.2 Perancangan

4.2.1 Activity Diagram

Diagram Activity memodelkan aktivitas antara pengguna dengan sistem yang berjalan berdasarkan pada skenario Use Case secara berorientasi objek. Diagram ini menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, keputusan yang mungkin terjadi, serta bagaimana semuanya diakhiri. Oleh karena itu diagram Activity tidak menggambarkan *internal behaviour* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

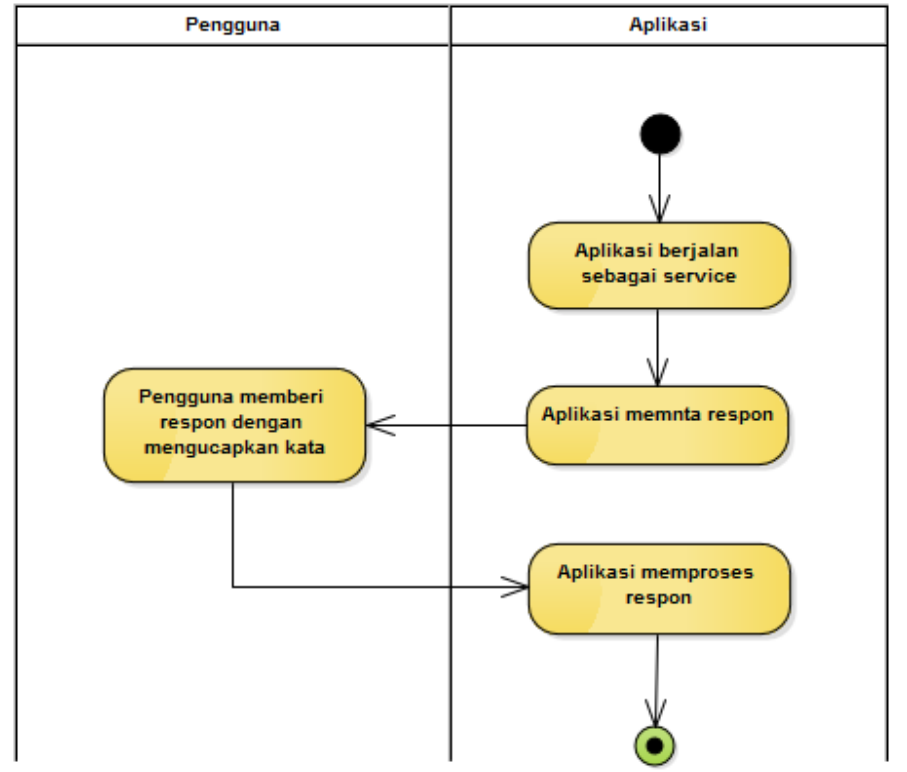


Gambar 4.2 Diagram Activity untuk mendengar pembacaan notifikasi

Setelah membuat skenario Use Case seperti yang terdapat pada Tabel 4.3, berdasarkan table tersebut kemudian dilakukan pemodelan ke dalam diagram Activity. Gambar 4.2 merupakan diagram Activity mendengar pembacaan notifikasi yang digunakan untuk menjelaskan interaksi antara pengguna dan sistem. Pertama pengguna harus menjalankan aplikasi, setelah itu aplikasi akan selalu memeriksa adanya SMS dan panggilan yang masuk. Pada bagian ini peran modul Android Text to Speech sangat penting, karena modul tersebut berperan sebagai pembicara kepada pengguna.

Kemudian dalam Gambar 4.3 merupakan diagram Activity untuk merespon SMS dan panggilan masuk, dimana setelah aplikasi membaca notifikasi aplikasi akan meminta respon dari pengguna untuk menentukan tindakan apa yang harus dilakukan selanjutnya. Pada bagian ini peran modul *library* PocketSphinx sangat

penting, dikarenakan modul tersebut berguna sebagai penerima serta pemroses respon suara dari pengguna.



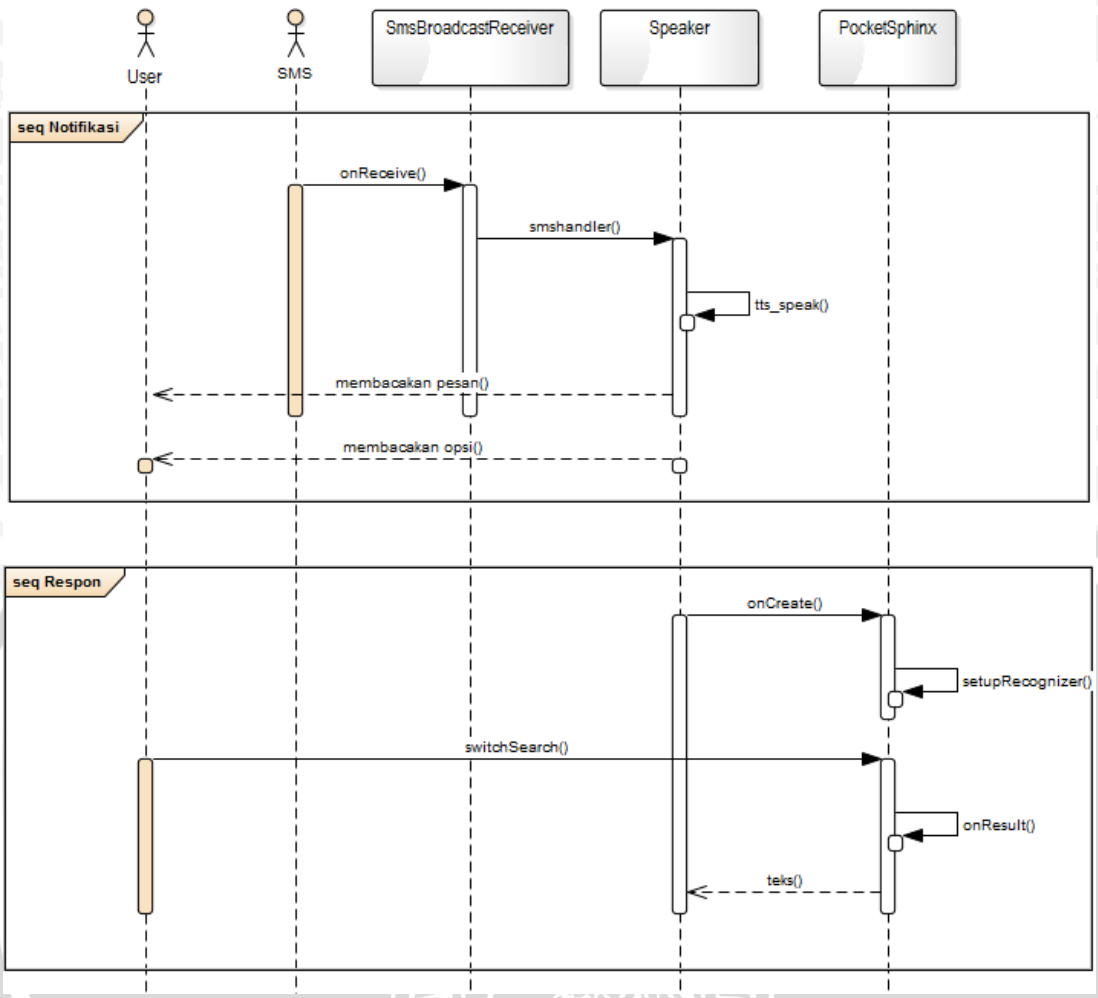
Gambar 4.3 Diagram Activity untuk proses merespon SMS dan panggilan masuk

4.2.2 Sequence Diagram

Diagram Sequence digunakan untuk menggambarkan interaksi antar objek pada aplikasi dan mengindikasikan komunikasi di antara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu serta dideskripsikan dalam urutan dari eksekusi. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

Seperti yang terlihat dalam Gambar 4.4, diagram tersebut memiliki dua segmen yaitu bagian notifikasi dan bagian respon. Pada bagian notifikasi menggambarkan bagaimana proses aplikasi saat membacakan notifikasi. Aplikasi akan menggunakan *Broadcast Receiver* untuk selalu memeriksa SMS dan panggilan yang masuk. Apabila terdapat SMS yang masuk, akan dilakukan inisialisasi terhadap modul *Text to Speech* pada kelas Speaker untuk kemudian akan dilakukan proses pembacaan oleh modul *Text to Speech*.

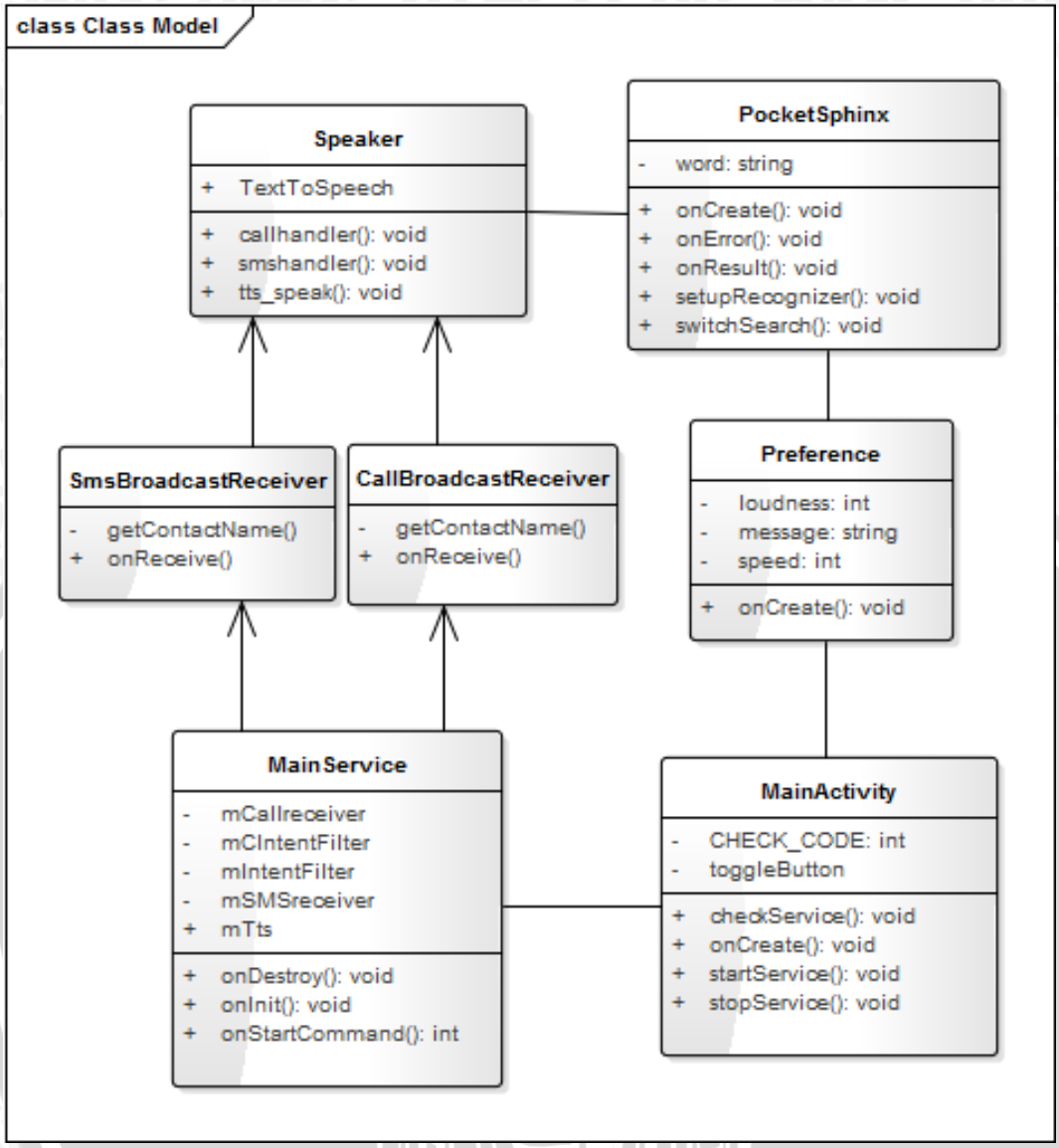
Kemudian dalam bagian respon, menggambarkan proses aplikasi dalam menerima respon. Kelas Speaker akan menjalankan *method* pada kelas PocketSphinx yang berguna sebagai modul *Speech Recognition* untuk menerima dan memproses respon dari pengguna.



Gambar 4.4 Diagram Sequence untuk proses pembacaan notifikasi

4.2.3 Class Diagram

Diagram Class adalah model statis yang menggambarkan struktur dan deskripsi kelas serta hubungannya antar kelas. Diagram yang terdiri dari kumpulan objek-objek dengan dan yang mempunyai struktur umum, *behavior* umum, relasi umum, dan kata yang umum. Kelas ditentukan/ditemukan dengan cara memeriksa objek-objek dalam Diagram Sequence. Sebuah kelas digambarkan seperti sebuah bujur sangkar dengan tiga bagian ruangan dan diberi nama menggunakan kata benda sesuai dengan domain ataupun kelompoknya. Tiga ruangan tersebut terdiri dari nama kelas, atribut, dan operasi/*method*. Bagian paling atas pada notasi kelas digunakan sebagai nama kelas, dan secara opsional juga digunakan sebagai *stereotype*-nya. Bagian tengah digunakan untuk menyimpan atribut, dan bagian paling bawah digunakan untuk menyimpan operasi. Untuk mempresentasikan atribut dapat dilakukan dengan cara menghubungkannya dengan kelas lainnya, untuk itu digunakan notasi relasi sebagai sarana pembentuk atribut relasi dalam diagram kelas yang besar.



Gambar 4.5 Diagram Class aplikasi

Cara yang baik untuk menemukan kelas-kelas adalah mulai dari memperhatikan aliran kejadian (*flow of event*) dari Use Case. Dalam Gambar 4.5 memperlihatkan bagaimana kelas-kelas utama dalam sistem serta bagaimana hubungan antar kelas tersebut. Pada kelas MainActivity akan ditampilkan menu utama dari aplikasi yang berguna untuk menjalankan aplikasi serta menu untuk melakukan konfigurasi. Selain itu, aplikasi akan menjalankan sebuah *service* bernama SmsService, dimana kelas SmsService tersebut memiliki hubungan relasi dengan kelas SmsBroadcastReceiver, CallBroadcastReceiver, dan SpeechRecognizer. Selain itu, kelas MainActivity juga memiliki relasi dengan kelas PreferenceMenu, dimana kelas tersebut berguna untuk menyediakan fitur konfigurasi pada aplikasi.

4.2.3.1 MainActivity

Pada Tabel 4.6, terlihat deskripsi dari *method* kelas MainActivity. Kelas ini merupakan kelas yang mengatur tampilan halaman utama.

Tabel 4.6 Deskripsi *method* pada kelas MainActivity

Method	**Deskripsi**
checkService()	*Method* ini berguna untuk memeriksa kondisi apakah *service* sudah aktif atau belum.
onCreate()	Inisialisasi awal variable dan konfigurasi pada aplikasi.
startService()	*Method* ini digunakan untuk menjalankan *service*.
stopService()	*Method* ini digunakan untuk menghentikan *service*.

4.2.3.2 Service

Pada Tabel 4.7, terlihat deskripsi dari *method* kelas Service. Kelas ini merupakan kelas yang berguna untuk mengatur Service pada aplikasi.

Tabel 4.7 Deskripsi *method* pada kelas Service

Method	**Deskripsi**
onDestroy()	*Method* ini berfungsi untuk menghentikan *service* yang sedang berjalan.
onInit()	Inisialisasi awal *method Service*.
onStartCommand()	*Method* ini berguna untuk menjalankan *service* berdasarkan perintah dari *method* lain.

4.2.3.3 Speaker

Pada Tabel 4.8, terlihat deskripsi dari *method* kelas Speaker. Kelas ini merupakan kelas yang menginisialisasi dan mendeskripsikan modul PocketSphinx.

Tabel 4.8 Deskripsi *method* pada kelas Speaker

Method	**Deskripsi**
smshandler()	*Method* ini berfungsi untuk menangani data dan proses SMS masuk.
callhandler()	*Method* ini berfungsi untuk menangani data dan proses panggilan telepon masuk.
tts_speak()	*Method* ini dijalankan sebagai pembicara oleh modul Text to Speech.

4.2.3.4 PocketSphinx

Pada Tabel 4.9, terlihat deskripsi dari *method* kelas PocketSphinx. Kelas ini merupakan kelas yang menginisialisasi dan mendeskripsikan modul PocketSphinx.

Tabel 4.9 Deskripsi *method* pada kelas PocketSphinx

Method	**Deskripsi**
onCreate()	*Method* ini berfungsi untuk inisialisasi awal dan mengolah suara yang masuk.
onError()	*Method* ini dijalankan apabila proses pengolahan suara gagal.
onResult()	*Method* ini dijalankan apabila proses pengolahan suara berhasil.

4.2.3.5 Preference

Pada Tabel 4.10, terlihat deskripsi dari *method* kelas Preference. Kelas ini merupakan kelas yang menginisialisasi dan mendeskripsikan fungsi pengaturan pada aplikasi.

Tabel 4.10 Deskripsi *method* pada kelas Preference

Method	**Deskripsi**
onCreate()	*Method* ini berfungsi untuk inisialisasi awal pada kelas serta konfigurasi aplikasi.

4.2.3.6 CallBroadcastReceiver

Pada Tabel 4.11, terlihat deskripsi dari *method* kelas CallBroadcastReceiver. Kelas ini merupakan kelas yang menginisialisasi dan mendeskripsikan fungsi Broadcast Receiver untuk menerima telepon pada aplikasi.

Tabel 4.11 Deskripsi *method* pada kelas *CallBroadcastReceiver*

Method	**Deskripsi**
onReceive()	*Method* ini berfungsi untuk menerima adanya informasi *broadcast*, dalam hal ini adalah panggilan telepon.
getContactName()	*Method* ini berguna untuk mengambil data kontak.

4.2.3.7 SmsBroadcastReceiver

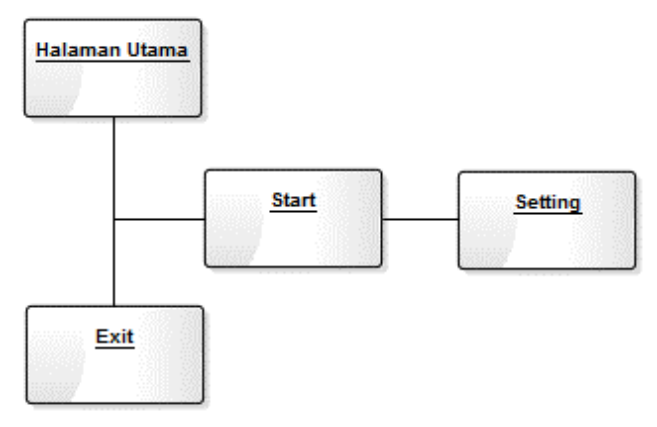
Pada Tabel 4.12, terlihat deskripsi dari *method* kelas SmsBroadcastReceiver. Kelas ini merupakan kelas yang menginisialisasi dan mendeskripsikan fungsi Broadcast Receiver untuk menerima SMS pada aplikasi.

Tabel 4.12 Deskripsi *method* pada kelas PocketSphinx

Method	Deskripsi
onReceive()	*Method* ini berfungsi untuk menerima adanya informasi *broadcast*, dalam hal ini adalah SMS.
getContactName()	*Method* ini berguna untuk mengambil data kontak.

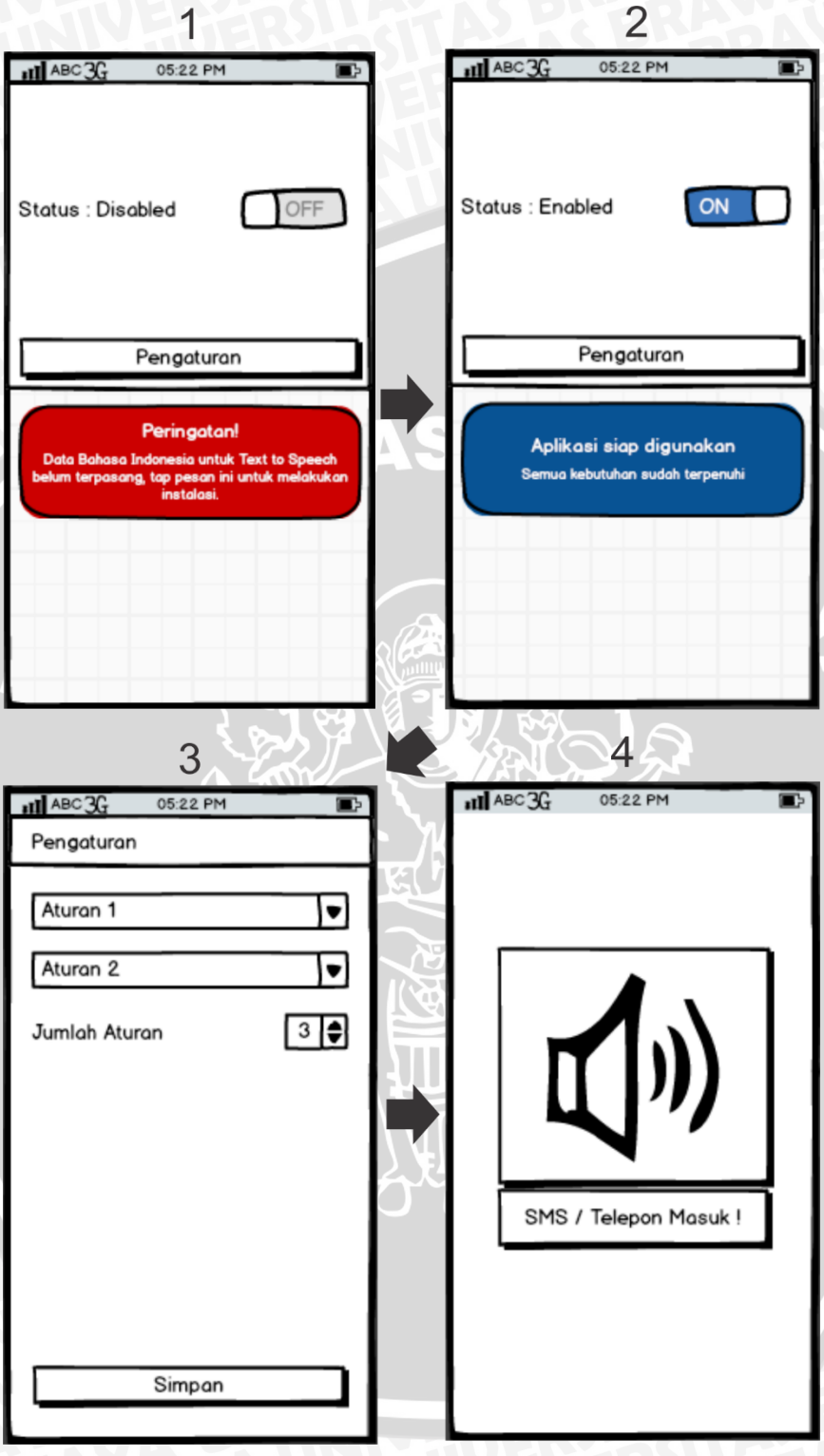
4.2.4 Perancangan Antarmuka

Perancangan antarmuka merupakan rancangan yang dilakukan sebelum melakukan implementasi antarmuka. Untuk mengetahui hubungan dan interaksi setiap halaman antarmuka, maka digunakan *sitemap* seperti yang ditunjukkan dalam Gambar 4.6.



Gambar 4.6 *Sitemap* antarmuka pada aplikasi

Dalam Gambar 4.7 merupakan rancangan antarmuka dari aplikasi serta bagaimana hubungan interaksi setiap antarmuka. Gambar pertama dan kedua merupakan halaman utama aplikasi, dimana pada gambar pertama terlihat bahwa aplikasi tidak aktif dikarenakan komponen yang belum lengkap, yaitu belum terpasangnya data Bahasa Indonesia yang digunakan oleh Android Text to Speech. Setelah data terpasang, maka aplikasi sudah dapat dijalankan, seperti yang terlihat pada gambar kedua. Kemudian pada halaman utama terdapat menu pengaturan untuk melakukan konfigurasi pada aplikasi, bagaimana yang terlihat pada gambar ketiga. Terakhir, gambar keempat merupakan tampilan ketika aplikasi mendeteksi adanya SMS atau panggilan telepon yang masuk.



Gambar 4.7 Tampilan antarmuka serta interaksi masing-masing antarmuka

BAB 5 IMPLEMENTASI

Pada bab ini akan dibahas mengenai tahapan implementasi dari aplikasi. Tahapan implementasi, tersusun dari spesifikasi perangkat lunak sistem, batasan-batasan dalam implementasi, implementasi keals dan aset pada *file* program, implementasi kode program, serta implementasi antarmuka aplikasi.

5.1 Implementasi

5.1.1 Spesifikasi Perangkat Lunak Sistem

Dalam proses pengembangan aplikasi, terdapat beberapa perangkat lunak yang digunakan. Seperti yang terlihat pada Tabel 5.1, aplikasi dikembangkan dalam lingkungan Bahasa pemrograman Java.

Tabel 5.1 Spesifikasi perangkat lunak komputer

Sistem Operasi	Windows 8.1
Programming Language	Java
Software Development Kit	Java Development Kit 8
Programming Environment	Java Runtime Environment 8
Integrated Development Environment	Android Studio
Speech Recognition Library	PocketSphinx

5.1.2 Batasan Implementasi

Dalam implementasi aplikasi terdapat batasan-batasan yang diberikan, yaitu sebagai berikut:

1. Modul *Text to Speech* pada aplikasi menggunakan *engine* standar bawaan sistem operasi Android.
2. Modul *Speech Recognition* menggunakan *library* tambahan bernama PocketSphinx.
3. Aplikasi hanya akan mengenali kata “Ya”, “Tidak”, “Satu”, “Dua”, dan “Tiga”.
4. Aplikasi dapat digunakan tanpa perlu terhubung ke internet.

5.1.3 Implementasi Kelas dan Aset

Pada Tabel 5.2, terlihat semua kelas dan aset yang telah diimplementasikan pada aplikasi. Kelas merupakan komponen terpenting dalam mengembangkan aplikasi, sedangkan aset merupakan komponen pendukung agar aplikasi dapat berjalan dengan semestinya, dalam hal ini aset yang digunakan aplikasi adalah *library* utama PocketSphinx.

Tabel 5.2 Implementasi *class* dan *assets* dalam bentuk *file*

Folder	Nama *File*
src/main/java	MainActivity.java
src/main/java	MainService.java
src/main/java	Speaker,java
src/main/java	SmsBroadcastReceiver.java
src/main/java	CallBroadcastReceiver.java
src/main/java	Setting.java
src/main/java	Guide.java
src/main/java	PocketSphinxActivity.java
libs	pocketsphinx-android-5prealpha-nolib.jar

5.1.4 Implementasi Kode Program

Implementasi kode program berikut hanya akan menjelaskan mengenai alur utama aplikasi, yaitu proses membaca menggunakan *Text to Speech* dan proses pengenalan suara oleh *Speech Recognition*. Perlu diketahui beberapa *method* yang dicantumkan dalam penulisan skripsi ini hanya untuk *method* yang dirasa cukup penting dan paling berpengaruh dalam sistem.

5.1.4.1 Implementasi Alur Proses Text to Speech

Proses *Text to Speech* untuk membaca akan dilakukan apabila ada SMS dan panggilan telepon yang masuk. Proses menggunakan *Text do Speech* ditunjukkan dalam Kode 5.1. Seperti yang sudah dijelaskan pada tahap perancangan, *source code* ini terdapat pada kelas Speaker dimana kelas tersebut terdapat modul Text to Speech untuk melakukan proses bicara.

```
1  public void smshandler(String sender){
2      SharedPreferences model = getSharedPreferences("model",
3  MODE_PRIVATE);
4      SharedPreferences.Editor prefsEditr = model.edit();
5      prefsEditr.putString("type", "1");
6      prefsEditr.apply();
7      String msg = "Anda mendapatkan pesan dari " + sender + ",
8  apakah anda ingin mendengarkan isi pesan?";
9      tts_speak(msg);
10 }
11
12 public void callhandler(String sender){
13     SharedPreferences model = getSharedPreferences("model",
14 MODE_PRIVATE);
15     SharedPreferences.Editor prefsEditr = model.edit();
16     prefsEditr.putString("type", "11");
17     prefsEditr.apply();
18     String msg = "Anda mendapatkan panggilan dari " + sender + ",
19 apakah anda ingin menjawabnya?";
20     tts_speak(msg);
21 }
22
23 public void tts_speak(final String msg){
24     mTts=null;
25     mTts=new TextToSpeech(getApplicationContext(), new
26 TextToSpeech.OnInitListener() {
```

```
27          public void onInit(int status) {
28              if(status != TextToSpeech.ERROR) {
29                  mTts.setLanguage(new Locale("in_ID"));
30                  mTts.setSpeechRate(Float.parseFloat(speed));
31                  mTts.setOnUtteranceProgressListener(new
32  UtteranceProgressListener() {
33                      @Override
34                      public void onDone(String utteranceId) {
35                          Intent intent = new Intent(Speaker.this,
36  PocketSphinxActivity.class);
37                          startActivityForResult(intent, 1);
38                      }
39                      @Override
40                      public void onError(String utteranceId) {
41                      }
42                      @Override
43                      public void onStart(String utteranceId) {
44                      }
45                  });
46                  if(Build.VERSION.SDK_INT >= 21 ){
47                      mTts.speak(msg, TextToSpeech.QUEUE_FLUSH,
48  null,"qwerty");
49                  }else{
50                      mTts.speak(msg, TextToSpeech.QUEUE_FLUSH,
51  null);}
52              }
53          }
54      });
55  }
```

Kode 5.1 Implementasi *source code* alur proses *Text to Speech*

Berikut ini adalah penjelasan dari *source code* yang terdapat dalam Kode 5.1:

1. Baris 1-10 merupakan *method* yang akan dijalankan apabila terdapat SMS yang masuk. Data SMS didapatkan dari kelas SmsBroadcastReceiver yang berfungsi untuk menerima SMS yang masuk.
2. Baris 12-21 merupakan *method* yang akan dijalankan apabila ada panggilan telepon yang masuk. Data penelepon didapatkan dari kelas SmsBroadcastReceiver yang berfungsi untuk menerima panggilan telepon yang masuk
3. Baris 23-55 merupakan *method* yang mendeklarasikan serta menjalankan fungsi *Text to Speech*. Pada baris 25 terlihat objek *Text to Speech* dibentuk dan pada baris 28 akan dilakukan pemeriksaan terhadap *error* yang terjadi pada saat inisialisasi objek *Text to Speech*. Setelah itu pada baris 29 dan baris 30, akan dilakukan inisialisasi terhadap bahasa yang akan digunakan pada pengucapan *Text to Speech* serta pengaturan seberapa cepat pengucapan yang akan dilakukan oleh *Text to Speech*. Setelah itu kata akan diucapkan dengan menggunakan fungsi *speak()* seperti yang terlihat pada baris 48-54.

5.1.4.2 Implementasi Alur Proses Speech Recognition

Proses *Speech Recognition* mengenali suara dilakukan untuk menerima respon dari pengguna. *Source code* dari proses ini ditunjukkan dalam Kode 5.2. Seperti yang sudah dijelaskan pada tahap rancangan, *source code* ini terdapat pada kelas

PocketSphinx yang berguna sebagai penerima dan pemroses respon suara dari pengguna.

```
1   public class PocketSphinxActivity extends Activity implements
2           RecognitionListener {
3
4       private static final String DIGITS_SEARCH = "digits";
5
6       private final int delayTime = 3700;
7       private Handler myHandler = new Handler();
8       private SpeechRecognizer recognizer;
9
10      @Override
11      public void onCreate(Bundle state) {
12          super.onCreate(state);
13
14          new AsyncTask<Void, Void, Exception>() {
15              @Override
16              protected Exception doInBackground(Void... params) {
17                  try {
18                      Assets assets = new
19  Assets(PocketSphinxActivity.this);
20                      File assetDir = assets.syncAssets();
21                      setupRecognizer(assetDir);
22                  } catch (IOException e) {
23                      return e;
24                  }
25                  return null;
26              }
27
28              @Override
29              protected void onPostExecute(Exception result) {
30                  if (result != null) {
31                      Toast.makeText(PocketSphinxActivity.this,
32  "Failed to init recognizer", Toast.LENGTH_LONG).show();
33                  } else {
34                      switchSearch(DIGITS_SEARCH);
35                  }
36              }
37          }.execute();
38
39          myHandler.postDelayed(closeControls, delayTime);
40      }
41      @Override
42      public void onDestroy() {
43          super.onDestroy();
44          recognizer.cancel();
45          recognizer.shutdown();
46      }
47      @Override
48      public void onBeginningOfSpeech() {
49      }
50      @Override
51      public void onEndOfSpeech() {
52
53      }
54      private void setupRecognizer(File assetsDir) throws
55  IOException {
56
57          recognizer = defaultSetup()
58                  .setAcousticModel(new File(assetsDir, "en-us-
59  ptm"))
60                  .setDictionary(new File(assetsDir, "cmudict-en-
```

```
us.dict"))

                .setRawLogDir(assetsDir)

                .setKeywordThreshold(1e-20f)

                .setBoolean("-allphone_ci", true)

                .getRecognizer();
        recognizer.addListener(this);

        File digitsGrammar = new File(assetsDir, "keyword.list");
        recognizer.addKeywordSearch(DIGITS_SEARCH, digitsGrammar);
    }

    private void switchSearch(String searchName) {
        recognizer.stop();

        recognizer.startListening(searchName, 3000);

        audioPlayer();
    }

@Override
    public void onPartialResult(Hypothesis hypothesis) {
        recognizer.stop();
    }

    @Override
    public void onResult(Hypothesis hypothesis) {
        if (hypothesis != null) {
            String text = hypothesis.getHypstr();

            Intent intent=new Intent();
            intent.putExtra("MESSAGE",text);
            setResult(RESULT_OK,intent);
            finish();
        }
    }

    @Override
    public void onError(Exception error) {

    }

    @Override
    public void onTimeout() {

    }
    public void onUserInteraction(){
        myHandler.removeCallbacks(closeControls);
        myHandler.postDelayed(closeControls, delayTime);
    }
    private Runnable closeControls = new Runnable() {
        public void run() {
            Intent intent=new Intent();
            intent.putExtra("MESSAGE","null");
            setResult(RESULT_OK,intent);
            finish();
        }
    };
    public void audioPlayer(){
        MediaPlayer mp = new MediaPlayer();
        int resId=R.raw.tone;
```

```
125             try {
126                 mp = MediaPlayer.create(this, resId);
127                 mp.start();
128             } catch (Exception e) {
129                 e.printStackTrace();
130             }
131         }
132 }
```

Kode 5.2 Implementasi *source code* alur proses *Speech Recognition*

Berikut ini adalah penjelasan dari *source code* yang terdapat dalam Kode 5.2:

1. Baris 8 merupakan inisialisasi objek *Speech Recognition*, dalam hal ini adalah PocketSphinx.
2. Baris 18-21 merupakan proses inisialisasi pengaturan pada sistem *Speech Recognition*.
3. Baris 29-35 merupakan proses awal *Speech Recognition* untuk melakukan pengenalan suara dengan memanggil *method switchSearch()*, dimana *method* tersebut berisi fungsi untuk melakukan proses menangkap suara pengguna.
4. Baris 94-101 merupakan *method* dimana proses pengenalan suara telah selesai melakukan pekerjaannya. Apabila suara yang diucapkan tidak terdapat pada *file training* maka akan menghasilkan nilai *null*.
5. Baris 102-109 merupakan *method* yang dijalankan apabila proses pengenalan suara gagal atau pengguna tidak terlambat mengucapkan kata.

5.1.4.3 Implementasi Menu Konfigurasi Aplikasi

Pada implementasi menu konfigurasi aplikasi ini, bahasa pemrograman yang digunakan berbeda dengan bahasa pemrograman sebelumnya. Bahasa pemrograman yang digunakan adalah bahasa XML, dimana bahasa pemrograman ini banyak digunakan sebagai desain antarmuka aplikasi Android. Implementasi kode dapat dilihat dalam Kode 5.3, dimana metode yang digunakan untuk mengambil data adalah dengan SharedPreferences.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen
3  xmlns:android="http://schemas.android.com/apk/res/android" >
4      <PreferenceCategory
5          android:title="Pengaturan" >
6          <ListPreference
7              android:key="speed"
8              android:title="Kecepatan Membaca"
9              android:entries="@array/speedAlias"
10             android:entryValues="@array/speed"
11             android:defaultValue="1.0"/>
12     </PreferenceCategory>
13     <PreferenceCategory
14         android:title="Template" >
15         <EditTextPreference
16             android:key="template1"
17             android:title="Template 1" />
18         <EditTextPreference
19             android:key="template2"
20             android:title="Template 2" />
```

```
21          <EditTextPreference
22              android:key="template3"
23              android:title="Template 3" />
24      </PreferenceCategory>
25  </PreferenceScreen>
```

Kode 5.3 *Source Code* implementasi menu konfigurasi aplikasi

Berikut ini adalah penjelasan dari *source code* yang terdapat dalam Kode 5.3:

1. Baris 7-11 merupakan inisialisasi untuk pengaturan kecepatan membaca pada modul *Text to Speech*. Secara *default* kecepatan membacanya adalah normal atau bernilai 1.
2. Baris 14-23 merupakan proses inisialisasi untuk pengaturan menu menyimpan data, dimana data yang disimpan adalah *template* pesan pengguna.

5.1.4.4 Implementasi Service

Aplikasi membutuhkan sebuah *service* untuk dapat berfungsi dengan baik. *Source code* dari implementasi ini ditunjukkan dalam Kode 5.4. Seperti yang sudah dijelaskan pada tahap rancangan, *source code* ini terdapat pada kelas MainService yang berguna sebagai pengontrol aplikasi dan bersifat *realtime*.

```
1   public class MainService extends Service{
2       private SmsBroadcastReceiver mSMSreceiver;
3       private CallBroadcastReceiver mCallreceiver;
4       private IntentFilter mIntentFilter;
5       private IntentFilter mCIntentFilter;
6
7       public IBinder onBind(Intent arg0) {
8           return null;
9       }
10      public void onInit(int status) {
11
12      }
13      public int onStartCommand(Intent intent, int flags, int
14  startId) {
15          mSMSreceiver = new SmsBroadcastReceiver();
16          mIntentFilter = new IntentFilter();
17
18  mIntentFilter.addAction("android.provider.Telephony.SMS_RECEIVED");
19          registerReceiver(mSMSreceiver, mIntentFilter);
20
21          mCallreceiver = new CallBroadcastReceiver();
22          mCIntentFilter = new IntentFilter();
23
24  mCIntentFilter.addAction("android.intent.action.PHONE_STATE");
25          registerReceiver(mCallreceiver, mCIntentFilter);
26
27          Toast.makeText(this, "The Service Started",
28  Toast.LENGTH_LONG).show();
29
30          AudioManager mobilemode = (AudioManager)
31  getSystemService(Context.AUDIO_SERVICE);
32          mobilemode.setRingerMode(AudioManager.RINGER_MODE_SILENT);
33
34          return START_STICKY;
35      }
36      public void onDestroy() {
37          super.onDestroy();
```

```
38          unregisterReceiver(mSMSreceiver);
39          unregisterReceiver(mCallreceiver);
40          AudioManager mobilemode =
41  (AudioManager)getSystemService(Context.AUDIO_SERVICE);
42          mobilemode.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
43      }
44  }
```

Kode 5.4 *Source Code* implementasi *service* pada aplikasi

Berikut ini adalah penjelasan dari *source code* yang terdapat dalam Kode 5.4:

1. Baris 7-35 merupakan inisialisasi *service* serta proses yang akan dilakukan setelah *service* dijalankan.
2. Baris 36-44 merupakan proses yang akan dilakukan apabila *service* dihentikan.

5.1.4.5 Implementasi Broadcast Receiver

Aplikasi membutuhkan sebuah *Broadcast Receiver* untuk menerima panggilan telepon dan SMS masuk. *Source code* dari implementasi ini ditunjukkan dalam Kode 5.5 dan Kode 5.6. Seperti yang sudah dijelaskan pada tahap rancangan, *source code* ini terdapat pada kelas CallBroadcastReceiver dan SmsBroadcastReceiver yang berguna sebagai penerima adanya SMS dan panggilan telepon masuk.

```
1   public class CallBroadcastReceiver extends BroadcastReceiver{
2       private static String mLastState;
3
4       public void onReceive(Context context, Intent intent) {
5           Bundle intentExtras = intent.getExtras();
6           if (intentExtras != null) {
7               String state =
8   intent.getStringExtra(TelephonyManager.EXTRA_STATE);
9                   if (!state.equals("IDLE")) {
10                      if (!state.equals(mLastState)) {
11                          mLastState = state;
12                          String incomingNumber =
13  intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER);
14                          String sender = getContactName(context,
15  incomingNumber);
16
17                          SharedPreferences.Editor data =
18  context.getSharedPreferences("data", context.MODE_PRIVATE).edit();
19                          data.putString("sender", sender);
20                          data.putString("body", null);
21                          data.putString("number", incomingNumber);
22                          data.apply();
23
24                          Intent i = new Intent(context,
25  Speaker.class);
26                          i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
27                          context.startActivity(i);
28                      }
29                  }else{
30                      AudioManager am =
31  (AudioManager)context.getSystemService(Context.AUDIO_SERVICE);
32                      switch (am.getRingerMode()) {
33                          case AudioManager.RINGER_MODE_NORMAL:
34
35  am.setRingerMode(AudioManager.RINGER_MODE_SILENT);
```

```
36                              break;
37                          }
38                      }
39              }
40      }
41
42      private String getContactName(Context context,String phone){
43          Uri uri =
44  Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI,
45  Uri.encode(phone));
46          String projection[] = new
47  String[]{ContactsContract.Data.DISPLAY_NAME};
48          Cursor cursor = context.getContentResolver().query(uri,
49  projection, null, null, null);
50          if(cursor.moveToFirst()){
51              return cursor.getString(0);
52          }else {
53              return "nomor tak dikenali";
54          }
55      }
56  }
```

Kode 5.5 *Source Code* implementasi *Broadcast Receiver* penerima panggilan telepon

Berikut ini adalah penjelasan dari *source code* yang terdapat dalam Kode 5.5:

1. Baris 4-40 merupakan inisialisasi *broadcast receiver* sebagai penerima panggilan telepon masuk. Di dalam *method* ini terdapat serangkaian proses untuk mengambil data dan informasi penelepon.
2. Baris 42-55 merupakan *method* yang berguna untuk mengakses informasi kontak telepon untuk dibandingkan dengan nomor telepon yang masuk. Ini berguna untuk mengetahui apakah nomor penelepon sudah ada di dalam kontak atau belum (nomor tidak diketahui).

```
1   public class SmsBroadcastReceiver extends BroadcastReceiver{
2       public static final String SMS_BUNDLE = "pdus";
3
4       public void onReceive(Context context, Intent intent) {
5           Bundle intentExtras = intent.getExtras();
6           if (intentExtras != null) {
7               Object[] sms = (Object[]) intentExtras.get(SMS_BUNDLE);
8               String sender = "";
9               String smsBody = "";
10              String number= "";
11              for (int i = 0; i < sms.length; ++i) {
12                  SmsMessage smsMessage =
13  SmsMessage.createFromPdu((byte[]) sms[i]);
14
15                  smsBody = smsMessage.getMessageBody().toString();
16                  sender = getContactName(context,
17  smsMessage.getOriginatingAddress());
18                  number = smsMessage.getOriginatingAddress();
19              }
20
21              SharedPreferences.Editor data =
22  context.getSharedPreferences("data", context.MODE_PRIVATE).edit();
23              data.putString("sender", sender);
24              data.putString("body", smsBody);
25              data.putString("number", number);
                data.apply();
```

```
26
27              Intent i = new Intent(context,Speaker.class);
28              i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
29              context.startActivity(i);
30
31          }
32      }
33
34      private String getContactName(Context context,String phone){
35          Uri uri =
36  Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI,
37  Uri.encode(phone));
38          String projection[] = new
39  String[]{ContactsContract.Data.DISPLAY_NAME};
40          Cursor cursor = context.getContentResolver().query(uri,
41  projection, null, null, null);
42          if(cursor.moveToFirst()){
43              return cursor.getString(0);
44          }else {
45              return "nomor tidak dikenali";
46          }
47      }
48  }
49
```

Kode 5.6 *Source Code* implementasi *Broadcast Receiver* penerima SMS

Berikut ini adalah penjelasan dari *source code* yang terdapat dalam Kode 5.6:

1. Baris 4-33 merupakan inisialisasi *broadcast receiver* sebagai penerima SMS. Di dalam *method* ini terdapat serangkaian proses untuk mengambil data dan informasi penelepon.
2. Baris 35-49 merupakan *method* yang berguna untuk mengakses informasi kontak telepon untuk dibandingkan dengan nomor telepon yang masuk. Ini berguna untuk mengetahui apakah nomor pengirim SMS sudah ada di dalam kontak atau belum (nomor tidak diketahui).

5.1.5 Implementasi Antarmuka Aplikasi

Bagian ini akan menunjukkan hasil implementasi antarmuka pada aplikasi yang terdiri dari halaman utama, menu pengaturan, dan tampilan notifikasi. Semua tampilan antarmuka ini adalah hasil implementasi dari perancangan antarmuka yang sudah dijelaskan pada tahap perancangan sebelumnya.

5.1.5.1 Halaman Utama

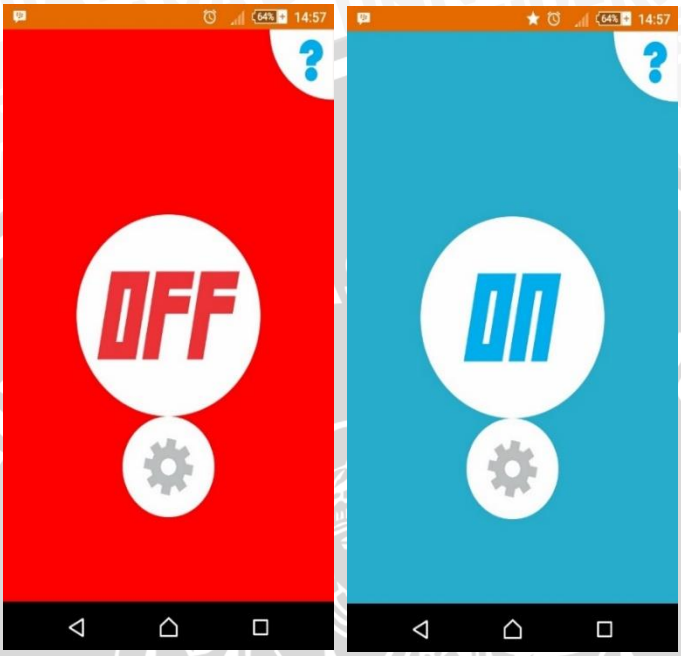
Halaman utama akan ditampilkan setelah pengguna membuka aplikasi. Pada menu ini terdapat tombol untuk mengaktifkan aplikasi dan tombol menu pengaturan, seperti yang terlihat dalam Gambar 5.1.

5.1.5.2 Menu Pengaturan

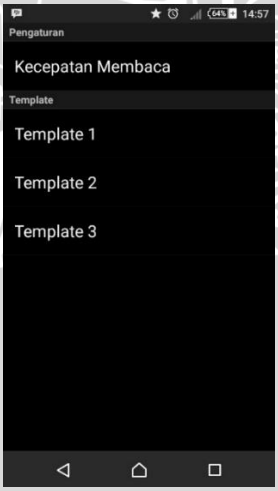
Menu pengaturan berisi pengaturan terhadap aplikasi. Pengaturan yang ada mencakup seberapa cepat Text to Speech membaca dan menyimpan *template* pesan pengguna, seperti yang terlihat dalam Gambar 5.2.

5.1.5.3 Tampilan Notifikasi

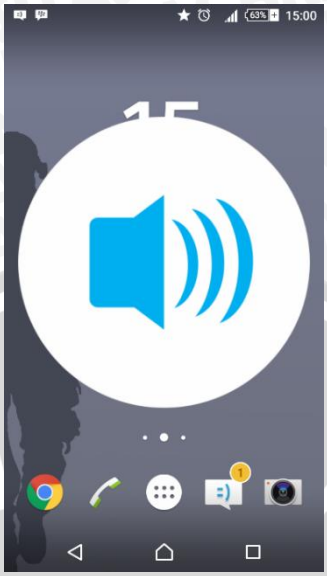
Tampilan ini akan muncul sebagai notifikasi SMS dan panggilan telepon. Dimana tampilan ini berfungsi sebagai interaksi kepada pengguna, seperti yang terlihat dalam Gambar 5.3.



Gambar 5.1 Tampilan halaman utama aplikasi



Gambar 5.2 Tampilan menu pengaturan pada aplikasi



Gambar 5.3 Tampilan notifikasi

BAB 6 PENGUJIAN

Bagian ini menjelaskan tentang pengujian dan analisis yang akan dilakukan pada aplikasi yang telah dibangun, dimana tujuan dari tahap pengujian adalah untuk menemukan kesalahan-kesalahan yang mungkin terjadi dalam sistem serta memastikan sistem yang dikembangkan telah sesuai dengan perancangan sistem yang telah dibuat sebelumnya. Pengujian yang dilakukan terhadap aplikasi adalah pengujian unit, pengujian validasi dan pengujian akurasi. Setelah pengujian selesai akan dilakukan analisis terhadap hasil pengujian yang telah dilakukan untuk mendapatkan kesimpulan.

6.1 Pengujian Unit

Pengujian unit dilakukan untuk melihat apakah algoritma berjalan dengan benar dimana teknik yang digunakan dalam melakukan pengujian unit ini adalah teknik Basis Path Testing. Pada teknik Basis Path Testing, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flowgraph*, kemudian menentukan jumlah kompleksitas siklomatis (*cyclomatic complexity*), basis set dari jalur independen dan memberikan kasus uji (*test case*) pada setiap basis set yang telah ditentukan.

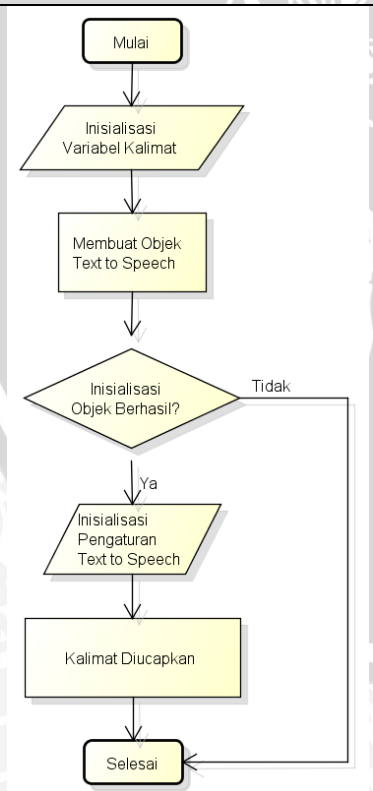
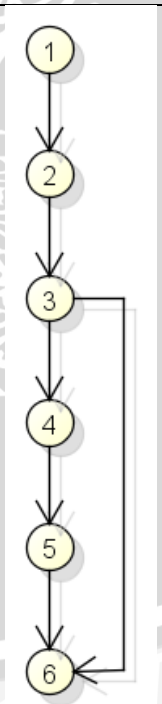
6.1.1 Kasus Uji Unit Proses Text to Speech

Dalam Kode 6.1 akan dijadikan acuan pembuatan *flowchart* dan *flowgraph* dalam Gambar 6.1. *Source code* ini merupakan serangkaian proses *Text to Speech* berdasarkan *source code* pada hasil implementasi sebelumnya.

```
1  public void smshandler(String sender){
2      SharedPreferences model = getSharedPreferences("model",
3  MODE_PRIVATE);
4      SharedPreferences.Editor prefsEditr = model.edit();
5      prefsEditr.putString("type", "1");
6      prefsEditr.apply();
7      String msg = "Anda mendapatkan pesan dari " + sender + ",
8  apakah anda ingin mendengarkan isi pesan?";.....1
9      tts_speak(msg);
10 }
11 public void callhandler(String sender){
12     SharedPreferences model = getSharedPreferences("model",
13 MODE_PRIVATE);
14     SharedPreferences.Editor prefsEditr = model.edit();
15     prefsEditr.putString("type", "11");
16     prefsEditr.apply();
17     String msg = "Anda mendapatkan panggilan dari " + sender + ",
18 apakah anda ingin menjawabnya?";.....1
19     tts_speak(msg);
20 }
21 public void tts_speak(final String msg){
22     mTts=null;
23     mTts=new TextToSpeech(getApplicationContext(), new
24 TextToSpeech.OnInitListener() {.....2
25         public void onInit(int status) {
26             if(status != TextToSpeech.ERROR) {.....3
27                 mTts.setLanguage(new Locale("in_ID"));.....4
```

```
28                  mTts.setSpeechRate(Float.parseFloat(speed));.....4
29                  mTts.setOnUtteranceProgressListener(new
30  UtteranceProgressListener() {
31                      @Override
32                      public void onDone(String utteranceId) {
33                          Intent intent = new Intent(Speaker.this,
34  PocketSphinxActivity.class);
35                          startActivityForResult(intent, 1);
36                      }
37                      @Override
38                      public void onError(String utteranceId) {
39                      }
40                      @Override
41                      public void onStart(String utteranceId) {
42                      }
43                  });
44                  if(Build.VERSION.SDK_INT >= 21 ){
45                      mTts.speak(msg, TextToSpeech.QUEUE_FLUSH,
46  null,"qwerty");.....5
47                  }else{
48                      mTts.speak(msg, TextToSpeech.QUEUE_FLUSH,
49  null);
50                  }
51              }
52          }.....6
53      });}
```

Kode 6.1 Pengujian unit proses *Text to Speech*

Flowchart	*Flowgraph*
	

Gambar 6.1 *Flowchart* dan *Flowgraph* proses *Text to Speech*

Pemodelan *flowgraph* yang telah dilakukan, menghasilkan kompeksitas siklomatis (*cyclomatic complexity*) melalui Persamaan 6.1, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul (*node*).

V(G) = E – N + 2 = 6 – 6 + 2 = 2 (6.1)

Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan, maka ditentukan dua buah basis set dari jalur *independent*, yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 5 – 6

Jalur 2: 1 – 2 – 3 – 6

Setiap jalur yang didapatkan akan dieksekusi sebanyak 1 kali jika syarat yang ada pada perintah *if-else* terpenuhi. Seperti yang terlihat pada Tabel 6.1, terlihat dua kasus pengujian unit untuk alur proses *Text to Speech*.

Tabel 6.1 Kasus uji untuk pengujian unit alur proses *Text to Speech*

No.	Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan
1	Inisialisasi objek *Text to Speech* berhasil	Aplikasi akan mengucapkan kalimat yang sudah ditentukan.	Aplikasi akan mengucapkan kalimat yang sudah ditentukan.
2	Inisialisasi objek *Text to Speech* gagal	Aplikasi tidak akan bisa mengucapkan kalimat.	Aplikasi tidak akan bisa mengucapkan kalimat.

6.1.2 Kasus Uji Unit Proses Speech Recognition

Dalam Kode 6.2 akan dijadikan acuan pembuatan *flowchart* dan *flowgraph* dalam Gambar 6.2. *Source code* ini merupakan serangkaian proses *Speech Recognition* berdasarkan *source code* pada hasil implementasi sebelumnya.

```
1   public class PocketSphinxActivity extends Activity implements
2           RecognitionListener {
3
4       private static final String DIGITS_SEARCH = "digits";
5
6       private final int delayTime = 3700;
7       private Handler myHandler = new Handler();
8       private SpeechRecognizer recognizer;.....1
9
10      @Override
11      public void onCreate(Bundle state) {
12          super.onCreate(state);
13
14          new AsyncTask<Void, Void, Exception>() {
15              @Override
16              protected Exception doInBackground(Void... params) {
17                  try {
18                      Assets assets = new
19  Assets(PocketSphinxActivity.this);
20                      File assetDir = assets.syncAssets();
21                      setupRecognizer(assetDir);.....2
```

```
22                  } catch (IOException e) {
23                      return e;.....8
24                  }
25                  return null;
26              }
27
28              @Override
29              protected void onPostExecute(Exception result) {
30                  if (result == null) {.....3
31                      switchSearch(DIGITS_SEARCH);.....4
32                  } else {
33                      Toast.makeText(PocketSphinxActivity.this,
34  "Failed to init recognizer", Toast.LENGTH_LONG).show();.....8
35                  }
36              }
37          }.execute();
38
39          myHandler.postDelayed(closeControls, delayTime);
40      }
41
42      @Override
43      public void onDestroy() {
44          super.onDestroy();
45          recognizer.cancel();
46          recognizer.shutdown();
47      }
48
49      @Override
50      public void onBeginningOfSpeech() {
51      }
52
53      @Override
54      public void onEndOfSpeech() {
55
56      }
57
58      private void setupRecognizer(File assetsDir) throws
59  IOException {
60
61          recognizer = defaultSetup()
62                  .setAcousticModel(new File(assetsDir, "en-us-
63  ptm"))
64                  .setDictionary(new File(assetsDir, "cmudict-en-
65  us.dict"))
66
67                  .setRawLogDir(assetsDir)
68
69                  .setKeywordThreshold(1e-20f)
70
71                  .setBoolean("-allphone_ci", true)
72
73                  .getRecognizer();
74          recognizer.addListener(this);
75
76          File digitsGrammar = new File(assetsDir, "keyword.list");
77          recognizer.addKeywordSearch(DIGITS_SEARCH, digitsGrammar);
78      }
79
80      private void switchSearch(String searchName) {
81          recognizer.stop();
82
83          recognizer.startListening(searchName, 3000);
84
85
```

```
86          audioPlayer();
87      }
88
89  @Override
90      public void onPartialResult(Hypothesis hypothesis) {
91          recognizer.stop();
92      }
93
94      @Override
95      public void onResult(Hypothesis hypothesis) {
96          if (hypothesis != null) {.....5
97              String text = hypothesis.getHypstr();
98
99              Intent intent=new Intent();
100             intent.putExtra("MESSAGE",text);
101             setResult(RESULT_OK,intent);.....7
102             finish();
103         }
104     }
105
106     @Override
107     public void onError(Exception error) {
108
109     }
110
111     @Override
112     public void onTimeout() {
113
114     }
115     public void onUserInteraction(){
116         myHandler.removeCallbacks(closeControls);
117         myHandler.postDelayed(closeControls, delayTime);
118     }
119     private Runnable closeControls = new Runnable() {
120         public void run() {
121             Intent intent=new Intent();
122             intent.putExtra("MESSAGE","null");.....6
123             setResult(RESULT_OK,intent);
124             finish();
125         }
126     };
127     public void audioPlayer(){
128         MediaPlayer mp = new MediaPlayer();
129         int resId=R.raw.tone;
130         try {
131             mp = MediaPlayer.create(this, resId);
132             mp.start();
133         } catch (Exception e) {
134             e.printStackTrace();
135         }
136     }
}
```

Kode 6.2 Pengujian unit proses *Speech Recognition*

Pemodelan *flowgraph* yang telah dilakukan, menghasilkan kompeksitas siklomatis (*cyclomatic complexity*) melalui Persamaan 6.2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node*) dan N merupakan jumlah simpul (*node*).

$$V(G) = E - N + 2 = 8 - 7 + 2 = 3 \quad (6.2)$$

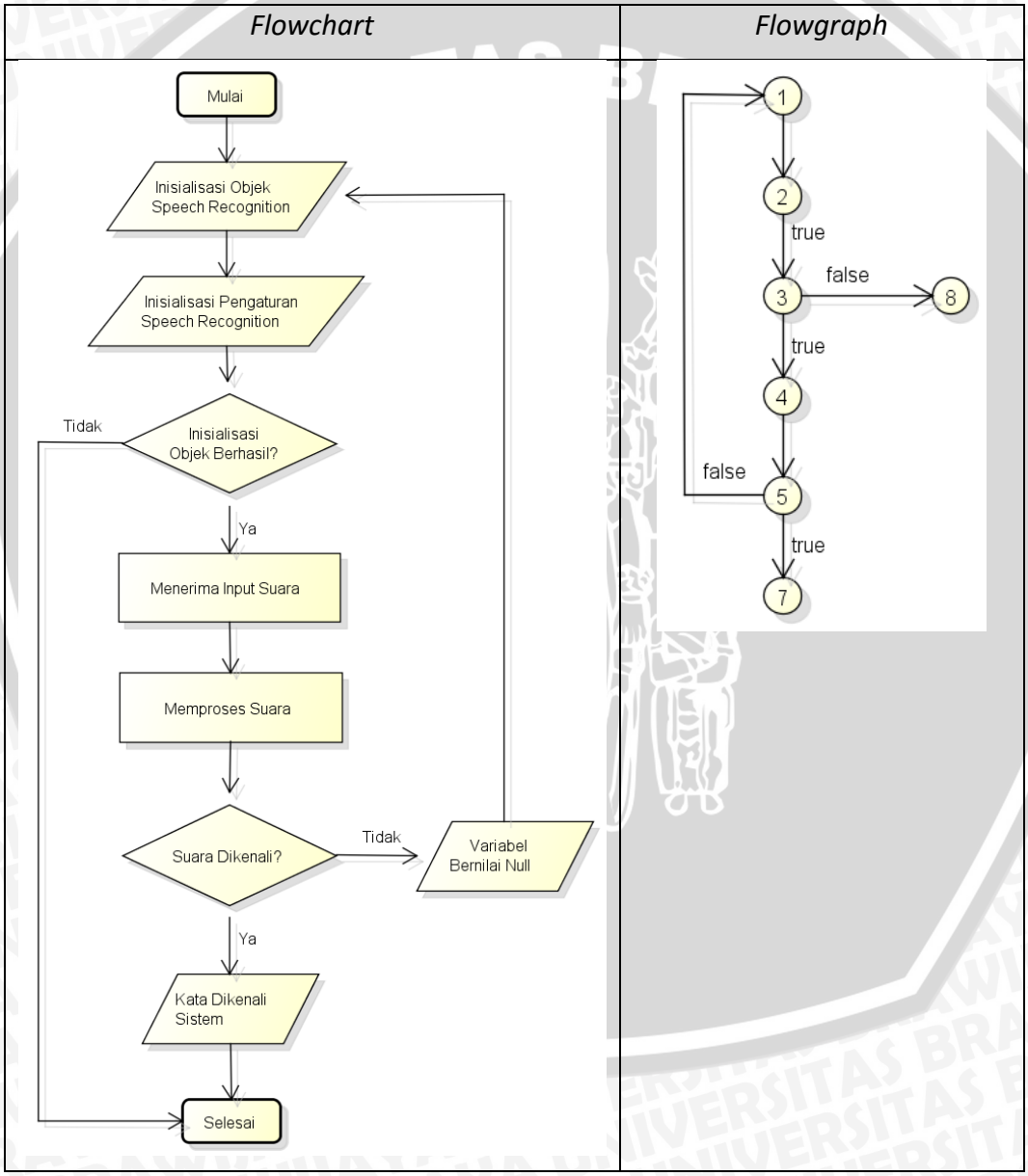
Berdasarkan dari nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan, maka ditentukan dua buah basis set dari jalur *independent*, yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 5 – 7

Jalur 2: 1 – 2 – 3 – 4 – 5 – 1

Jalur 3: 1 – 2 – 3 – 8

Setiap jalur yang didapatkan akan dieksekusi sebanyak 1 kali jika syarat yang ada pada perintah *if-else* terpenuhi. Seperti yang terlihat pada Tabel 6.2, terlihat dua kasus pengujian unit untuk alur proses *Speech Recognition*.



Gambar 6.2 *Flowchart* dan *Flowgraph* proses *Speech Recognition*

Tabel 6.2 Kasus uji untuk pengujian unit alur proses *Speech Recognition*

No.	Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan
1	*Speech Recognition* mengenali suara	Aplikasi mampu mengenali masukan suara dari pengguna.	Aplikasi mampu mengenali masukan suara dari pengguna.
2	*Speech Recognition* gagal mengenali suara	Aplikasi tidak bisa mengenali masukan suara dari pengguna.	Aplikasi tidak bisa mengenali masukan suara dari pengguna.

6.2 Pengujian Validasi

Pengujian validasi dilakukan untuk mengetahui apakah sistem yang dibangun sudah berjalan seperti yang diharapkan. Pengujian dilakukan berdasarkan skenario Use Case dan menggunakan metode Black Box, karena tidak diperlukan konsentrasi terhadap alur jalannya program dan lebih ditekankan untuk menemukan kesesuaian antara kinerja sistem dengan daftar kebutuhan. Untuk mengetahui kesesuaian antara daftar kebutuhan dengan kinerja sistem, maka pada kebutuhan fungsional utama sistem akan dilakukan pengujian dengan kasus uji masing-masing yang sudah ditentukan.

6.2.1 Kasus Uji Mendengarkan Notifikasi

Tabel 6.3 merupakan kasus uji terhadap kebutuhan fungsional KF-01. Dimana kasus uji tersebut bertujuan untuk memastikan bahwa aplikasi akan membacakan notifikasi dan pengguna dapat mendengarkan dengan baik.

Tabel 6.3 Kasus Uji Mendengarkan Notifikasi

Nama kasus uji	**Mendengarkan notifikasi**
Objek uji	Kebutuhan Fungsional (KF01)
Tujuan pengujian	Memastikan bahwa aplikasi akan membacakan notifikasi dan pengguna dapat mendengarkan dengan baik.
Prosedur uji	1. Membuka aplikasi 2. Menyalakan *service* pada aplikasi 3. Mengirim SMS dan panggilan telepon ke alat uji.
Hasil yang diharapkan	Aplikasi akan membacakan notifikasi SMS dan panggilan masuk.

6.2.2 Kasus Uji Menerima Respon Pengguna

Tabel 6.4 merupakan kasus uji terhadap kebutuhan fungsional KF-04. Dimana kasus uji tersebut bertujuan untuk memastikan bahwa aplikasi menerima respon suara dari pengguna dengan baik.

Tabel 6.4 Kasus Uji Menerima Respon Pengguna

Nama kasus uji	Menerima respon pengguna
Objek uji	Kebutuhan Fungsional (KF04)
Tujuan pengujian	Memastikan bahwa aplikasi menerima respon suara dari pengguna dengan baik.
Prosedur uji	1. Membuka aplikasi 2. Menyalakan *service* pada aplikasi 3. Mengirim SMS dan panggilan telepon ke alat uji.
Hasil yang diharapkan	Setelah membacakan notifikasi, aplikasi akan meminta dan menerima respon suara dari pengguna.

6.2.3 Hasil Pengujian Validasi

Seperti yang terlihat pada Tabel 5.6, hasil pengujian validasi berdasarkan skenario Use Case menunjukkan bahwa kedua kasus uji tersebut menghasilkan nilai valid. Itu menandakan bahwa sistem sudah memenuhi semua kebutuhan fungsional.

Tabel 6.5 Hasil pengujian validasi

No.	Nama Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan	Validitas
1	Mendengarkan notifikasi	Aplikasi akan membacakan notifikasi SMS dan panggilan masuk.	Aplikasi dapat membacakan notifikasi SMS dan panggilan masuk.	Valid
2	Menerima respon pengguna	Setelah membacakan notifikasi, aplikasi akan meminta dan menerima respon suara dari pengguna.	Aplikasi dapat meminta dan menerima respon suara dari pengguna	Valid

6.3 Pengujian Akurasi

Pada tahapan ini akan dilakukan pengujian akurasi sistem dalam mengenali kata. Proses pengujian akan dilakukan dengan mengambil suara dari 15 orang yang akan mengucapkan kata “Ya”, “Tidak”, “Satu”, “Dua”, dan “Tiga”. Pengambilan suara hanya akan dilakukan sekali saja untuk menentukan apakah kata-kata tersebut dapat dikenali atau tidak.

Data hasil pengenalan suara dapat dilihat pada Lampiran A.1, kata yang tidak dikenali aplikasi akan ditandai dengan teks berwarna merah atau disebut sebagai *False Negative*. Sedangkan kata yang bisa dikenali aplikasi tetapi sebenarnya adalah salah ditandai dengan teks berwarna kuning atau disebut sebagai *False Positive*.

Penentuan nilai akurasi akan dilakukan dengan menghitung nilai *precision*, *recall*, dan F-measure. *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem dan *recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Sedangkan F-measure digunakan untuk menghitung kinerja dari suatu model atau sistem, dengan mengkombinasikan nilai *precision* dan *recall* (Powers, 2003)**.** Untuk menghitung *precision* digunakan Persamaan 6.3, sedangkan untuk menghitung *recall* digunakan Persamaan 6.4. Nilai *precision* dan *recall* dibutuhkan karena kedua hal tersebut dibutuhkan untuk menghitung nilai F-measure sesuai yang dirumuskan pada Persamaan 6.5.

Tabel 6.6 Tipe data uji

		Nilai Sebenarnya	
		True	False
Nilai Prediksi	True	TP (*True Positive*)	FP (*False Positive*)
	False	FN (*False Negative*)	TN (*True Negative*)

$$Precision = \frac{TP}{TP+FP} \quad (6.3)$$

$$Recall = \frac{TP}{TP+FN} \quad (6.4)$$

$$\text{F-measure} = 2\frac{Precision.Recall}{Precision+Recall} \quad (6.5)$$

Sehingga untuk kasus berdasarkan Lampiran A.1, akan dihitung nilai *precision*, *recall* dan F-measure dengan menggunakan rumusan tersebut. Perhitungannya seperti yang terlihat dalam Persamaan 6.6, Persamaan 6.7, dan Persamaaan 6.8.

Tabel 6.7 Data hasil uji

		Nilai Sebenarnya	
		True	False
Nilai Prediksi	True	59	5
	False	11	0

$$Precision = \frac{59}{59+5} = 0{,}92 \quad (6.6)$$

$$Recall = \frac{59}{59+11} = 0{,}84 \quad (6.7)$$

$$\text{F-measure} = 2\frac{0{,}92\ .\ 0{,}84}{0{,}92 + 0{,}84} = 0{,}87 \quad (6.8)$$

Seperti penjelasan sebelumnya, F-measure digunakan untuk menghitung kinerja dari suatu sistem, dalam hal ini adalah tingkat akurasi. Oleh karena itu, hasil perhitungan tersebut menghasilkan nilai 0,87 atau 87%.

6.4 Analisis Hasil Pengujian

Proses analisis terhadap hasil pengujian dilakukan untuk mendapatkan kesimpulan dari hasil pengujian aplikasi mesin pencari informasi penjualan rumah yang telah dilakukan. Proses analisis mengacu pada hasil pengujian yang telah didapatkan. Proses analisis yang dilakukan adalah analisis pada pengujian unit, validasi dan akurasi.

6.4.1 Analisis Hasil Pengujian Unit

Proses analisis pengujian unit dilakukan dengan melihat kesesuaian fungsi dari implementasi unit fungsi yang diujikan dengan hasil perancangan perangkat lunak yang telah dirancang sebelumnya. Berdasarkan hal tersebut, maka dapat diambil kesimpulan bahwa unit fungsi yang diujikan dari program sudah memenuhi kebutuhan fungsional yang telah dirancang pada tahap perancangan.

6.4.2 Analisis Hasil Pengujian Validasi

Proses analisis pengujian validasi mengacu pada kesesuaian kinerja sistem dengan kebutuhan fungsionalitas sistem. Berdasarkan hasil pengujian validasi dapat disimpulkan bahwa aplikasi telah memenuhi kebetuhan yang diuraikan pada tahap analisis kebutuhan, sebab hasil dari pengujian validasi ini menghasilkan nilai valid.

6.4.3 Analisis Hasil Pengujian Akurasi

Proses analisis pengujian akurasi dilakukan dengan mengambil suara dari 15 orang yang akan mengucapkan kata “Ya”, “Tidak”, “Satu”, “Dua”, dan “Tiga”, dimana pengambilan suara hanya akan dilakukan sekali untuk kemudian ditentukan apakah kata-kata tersebut dapat dikenali atau tidak.

Penentuan nilai akurasi akan dilakukan dengan menghitung nilai F-measure berdasarkan nilai *precision* dan *recall*, dimana Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem dan recall adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Mengacu pada Lampiran A.1, nilai *precision* adalah sebesar 0,92 dan *recall* sebesar 0,84. Setelah kedua nilai tersebut didapatkan, maka dihitung nilai F-measure berdasarkan kedua nilai tersebut, sehingga didapatkan nilai F-measure sebesar 0,87 atau 87%. Nilai tersebut yang akan dijadikan sebagai nilai keakuratan aplikasi dalam mengenali suara.

BAB 7 PENUTUP

8.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut:

1. Pengembangan aplikasi notifikasi SMS dan panggilan telepon dengan menggunakan Android Text to Speech dan *library* PocketSphinx Speech Recognition telah berhasil diimplementasikan sesuai dengan analisis dan perancangan sistem yang telah dibuat. Aplikasi akan berjalan sebagai *service* serta menggunakan *Broadcast Receiver* untuk memeriksa adanya SMS dan panggilan telepon yang masuk secara *realtime*. Selain itu, tujuan digunakannya modul Android Text to Speech adalah sebagai alat untuk menyampaikan notifikasi kepada pengguna secara *offline*. Berdasarkan hasil pengujian validasi dengan metode Black Box, hasilnya menunjukkan bahwa aplikasi ini telah memenuhi kebutuhan fungsional yang telah dirancang sebelumnya.
2. Dengan mengimplementasikan *library* PocketSphinx Speech Recognition, aplikasi sepenuhnya dapat digunakan secara *offline* atau tanpa terkoneksi dengan internet. Hal itu dapat dilakukan dengan menambahkan kata yang perlu dikenali kedalam sebuah *dictionary file* dan kemudian mendeklarasikan bagaimana cara pengucapan kata tersebut.
3. Berdasarkan pengujian akurasi terhadap data uji pada Lampiran A.1 dengan menghitung nilai F-measure, maka prosentase keberhasilan yang berhasil didapatkan dalam mengenali suara adalah 87%.

8.2 Saran

Saran yang dapat diberikan untuk pengembangan selanjutnya dari aplikasi ini adalah:

1. Untuk pengembangan lebih lanjut, peningkatan akurasi dalam mengenali suara perlu dilakukan. Dengan melakukan penambahan model suara pada modul *library* PocketSphinx dan kemudian melakukan *rebuild* pada *library* tersebut.
2. Pengembangan aplikasi selanjutnya dapat dilakukan dengan melakukan implementasi pada *platform* lainnya. Karena *library* PocketSphinx bersifat *open source* dan *multiplatform*.

DAFTAR PUSTAKA

Android Developers. 2015. *Services*. [online] Tersedia di: <http://developer.android.com> [Diakses 14 April 2015].

Android Developers. 2015. *BroadcastReceiver*. [online] Tersedia di: <http://developer.android.com> [Diakses 14 April 2015].

Android Developers Blog. 2009. *An introduction to Text-To-Speech in Android*. [online] Tersedia di: <http://android-developers.blogspot.com> [Diakses 14 April 2015].

Dwi, H dan Huda, M. 2013. *Text Pre-Processing Pada Text To Speech Synthesis System Untuk Penutur Berbahasa Indonesia*. [pdf] Politeknik Elektronika Negeri Surabaya. Tersedia di: http://www.pens.ac.id/post/20130813144934-1378 [Diakses 01 September 2015]

Ferdiansyah, V dan Purwarianti, A. 2012. "Indonesian Automatic Speech Recognition System Using English-Based Acoustic Model", *American Journal of Signal Processing 2012*, hal. 60-63.

Meier, R. 2009. *Professional Android Application Development*, 1st Edition, Wiley Publishing, Inc., Indiana.w

Meng, J dan Zhang, J. 2012. "Overview of the Speech Recognition Technology", *2012 Fourth International Conference on Computational and Information Sciences*, hal. 199-202.

Munawar, B. 2010. "Word Identification Using Hidden Markov Model (HMM) Through Feature Extraction Linear Predictive Coding (LPC)", Undergraduate Theses from Perpustakaan UNIKOM

Patra, S dan Patel, P. 2014. "A Research On Android's Text to Speech Engine", *International Journal of Students' Research in Technology & Management*, Vol. 2, No. 1, hal. 19-25.

Primorac, S dan Russo, M. 2012. "Android Application For Sending SMS Messages With Speech Recognition Interface", *MIPRO 2012 Proceedings of the 35th International Convention*, hal. 1763-1767.

Powers, David M. W. 2003. "Recall and Precision versus the Bookmaker", *Proceedings of the International Conference on Cognitive Science (ICSC-2003)*, hal. 529-534.

Rabiner, L dan Juang, B. H. 1993. *Fundamentals of Speech Recognition*, Prentice Hall, New Jersey.

Schroeter, J. 2005. *Text to-Speech (TTS) Synthesis*, [e-book] AT&T Laboratories. Tersedia melalui: AT&T Laboratories <http://www.research.att.com> [Diakses 13 April 2015].

LAMPIRAN A DATA UJI

A.1 Data Uji

	Jenis Kelamin	Kata				
		Ya	Tidak	Satu	Dua	Tiga
Orang 1	L	Ya	Tiga	Satu	Dua	Tiga
Orang 2	L	Ya	Tidak	Satu	Dua	Tidak
Orang 3	L	Ya	Tidak	Null	Dua	Tiga
Orang 4	P	Ya	Tidak	Satu	Null	Tiga
Orang 5	L	Null	Tidak	Satu	Dua	Tiga
Orang 6	P	Ya	Tidak	Null	Dua	Tiga
Orang 7	P	Ya	Null	Satu	Dua	Null
Orang 8	L	Ya	Tidak	Satu	Null	Tiga
Orang 9	L	Ya	Tidak	Satu	Dua	Tidak
Orang 10	P	Ya	Tidak	Satu	Null	Tiga
Orang 11	L	Ya	Tiga	Satu	Dua	Tidak
Orang 12	L	Ya	Null	Satu	Dua	Tiga
Orang 13	P	Ya	Tidak	Null	Dua	Tiga
Orang 14	L	Ya	Tidak	Satu	Dua	Tiga
Orang 15	L	Ya	Tidak	Satu	Null	Tiga