RANCANG BANGUN APLIKASI *MESSAGING* BERBASIS *VOICE INTERACTION* BAGI PENDERITA TUNANETRA PADA SISTEM OPERASI ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
LEO TIOFAN JUSTICIA
NIM: 125150101111017

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

RANCANG BANGUN APLIKASI *MESSAGING* BERBASIS *VOICE INTERACTION* BAGI PENDERITA TUNANETRA PADA SISTEM OPERASI ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
LEO TIOFAN JUSTICIA
NIM: 125150101111017

Skripsi ini telah diuji dan dinyatakan lulus pada
4 Mei 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng. Herman Tolle, S.T, M.T	Faizatul Amalia, S.Pd., M.Pd
NIP/NIK: 197408232000121001	NIP/NIK: 2013098608212001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph. D
NIP/NIK: 197105182003121001

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, 4 Mei 2017

LEO TIOFAN JUSTICIA

NIM: 125150101111017

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus karena atas penyertaanya penulis dapat menyelesaikan penelitian melalui tugas akhir skripsi yang berjudul Rancang Bangun Aplikasi *Messaging* Berbasis *Voice Interaction* Bagi Penyandang Tunanetra Pada Sistem Operasi Android. Penelitian ini telah menghabiskan waktu lebih kurang satu tahun. Pada proses pelaksanaanya, peneliti banyak menghadapi tantangan dan rintang. Namun, hingga akhir proses penelitian ini, berbagai tantangan dan rintangan tersebut berhasil dihadapi dan diatasi oleh penulis dengan bantuan dan peran serta berbagai pihak. Oleh sebab itu, penulis juga ingin mengucapkan terimakasih kepada pihak-pihak yang berperan dan membantu proses penyelesaian penelitian ini, yaitu kepada:

1. Bapak Dr. Eng. Herman Tolle, S. T., M.T. selaku dosen pembimbing I yang telah memberikan waktu, tenaga, ilmu dan nasihat yang tulus dalam membimbing dan mengarahkan penulis hingga dapat menyelesaikan penelitian ini.
2. Ibu Faizatul Amalia, S.Pd., M.Pd. selaku dosen pembimbing II yang telah memberikan waktu, tenaga, ilmu dan nasihat yang tulus dalam membimbing dan mengarahkan penulis hingga dapat menyelesaikan penelitian ini.
3. Ayah dan Ibu penulis terkasih, Hiras Silalahi dan Martianna Nainggolan S.Kes beserta anggota keluarga penulis lainnya yang telah membantu memberikan dukungan terus menerus dalam berbagai aspek baik bagi diri penulis maupun bagi penelitian ini.
4. Teman-teman PMK Daniel yang telah memberikan dukungan melalui doa dan semangat kepada penulis.
5. Teman-teman Fakultas Imu Komputer yang teah memberikan dukungan melalui doa dan semangat kepada penulis.

Hingga diselesaikannya penelitian ini, penulis merasa masih memiliki banyak kekurangan. Oleh sebab itu, penulis mengharapkan saran dan kritik yang membangun dari pembaca untuk proses pengembangan berikutnya dari penelitian ini.

Malang, 4 Mei 2017

Penulis

leosilalahi62@gmail.com

ABSTRAK

Rancang Bangun Aplikasi *Messaging* Berbasis *Voice Interaction* Bagi Penderita Tunanetra Pada Sistem Operasi Android. Fakultas Ilmu Komputer, Universitas Brawijaya. 2017. Penulis: Leo Tiofan Justicia, Dr. Eng. Herman Tolle, S.T, M.T. dan Faizatul Amalia, S.Pd., M.Pd.

Teknologi *smartphone* yang ada pada saat ini, hadir dengan menawarkan sejumlah kemudahan bagi proses komunikasi yang dilakukan oleh manusia. Melalui pemanfaatan teknologi *smartphone* manusia mampu berkomunikasi secara *realtime* tanpa batasan jarak komunikasi. Namun, penyandang tunanetra justru menghadapi hal tersebut dengan sejumlah perbedaan. Penyandang tunanetra merupakan kategori disabilitas yang memiliki keterbatasan fisik untuk melakukan pekerjaan yang memerlukan kemampuan visualisasi. Sehingga dengan adanya keterbatasan tersebut, para penyandang tunanetra tidak dapat mengoperasikan *smartphone* untuk berkomunikasi dengan sesamanya. Untuk mengatasi masalah tersebut diperlukan sebuah aplikasi yang dapat memungkinkan para penyandang tunanetra berkomunikasi dengan sesamanya melalui *smartphone*. Melalui penelitian ini dilakukan pengembangan aplikasi *messaging* berbasis *voice interaction* untuk membantu para penyandang tunanetra dalam berkomunikasi dengan sesamanya. Aplikasi ini memungkinkan penggunanya untuk memberikan kontrol terhadap aplikasi melalui sentuhan ataupun perintah suara. Pengguna akan menerima umpan balik berupa keluaran suara untuk setiap perintah suara yang diberikannya. Berdasarkan MOS (*Mean Opinion Score*) *Testing* yang telah dilakukan, diperoleh nilai MOS sebesar 3,7. Hal ini menunjukkan rata-rata pengguna menyatakan bahwa layanan *voice interaction* yang tedapat pada aplikasi memiliki kualitas yang baik (*Good*).

Kata kunci: tunanetra, *messaging*, *voice interaction*, aplikasi Android

ABSTRACT

The Development Of Voice Interaction Based Messaging Application For People With Visual Impairment On The Android Operating System. Faculty of Computer Science. Brawijaya University. 2017. Author: Leo Tiofan Justicia, Dr. Eng. Herman Tolle, S.T, M.T. and Faizatul Amalia, S.Pd., M.Pd.

Smartphone technologies that exist today, comes to offer a number of easines for the communications process that is performed by humans. Through the use of smartphone technology, humans are able to communicate in realtime communication without distance limitations. However, people with visual impairment actually confront it with a number of differences. People with visual impairment is a disability category who have physical limitations for work that requires visualization capabilities. So with these limitations, they can't operate a smartphone to communicate with others. Through this research, developed a voice-interaction-based messaging application to help people with visual impairments to communicate with others using a smartphone. This application allows users to control the application by touch or voice commands. Users will receive feedbacks in the form of speech output for each given voice command. Based on MOS (Mean Opinion Score) Testing that has been done, obtained MOS value by 3,7. It means, the average user stating that the voice-interaction service that contained in the application has a good quality.

Keywords: *visual impairment, messaging, voice interaction, Android application*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Bagian ini menjelaskan seluruh hal yang menjadi dasar dan ruang lingkup proses penelitian di dalam skripsi. Setiap elemen dalam dalam bagian ini akan menjadi acuan dalam penulisan seluruh bab berikutnya. Elemen-elemen tersebut terdiri dari latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian hingga sistematika pembahasannya.

1.1 Latar Belakang

Manusia merupakan makhluk sosial. Artinya, manusia tidak dapat hidup tanpa berhubungan dengan sesamanya. Untuk memenuhi kodratnya tersebut, manusia menggunakan proses komunikasi agar dapat tetap terhubung satu sama lain. Melalui proses komunikasi, manusia juga mampu mempelajari banyak hal yang dapat memperluas wawasannya. Manusia mampu mengenali serta memahami sesamanya, lingkungan sekitarnya atau bahkan lingkungan yang belum pernah dilihatnya sekalipun melalui proses komunikasi. Sehingga, tidak heran hingga saat ini komunikasi menjadi kebutuhan yang sangat penting dalam kehidupan manusia (Garibay, et al., 2014).

Seiring dengan perkembangan teknologi, proses komunikasi yang dimiliki oleh manusia juga ikut mengalami perubahan. Jika pada awalnya proses komunikasi yang dilakukan oleh manusia terbatas pada ruang dan waktu, maka melalui pemanfaatan teknologi manusia mampu berkomunikasi dengan sesamanya kapan saja dan dimana saja. Sebagai contohnya adalah pemanfaatan teknologi *smartphone* dalam berkomunikasi.

Smartphone merupakan salah satu teknologi komunikasi yang cukup populer saat ini (Neira, 2015). Melalui keberadaannya, *smartphone* menawarkan sejumlah kemudahan bagi proses komunikasi yang dilakukan oleh manusia. Manusia tidak lagi berkomunikasi dengan batasan waktu dan jarak seperti halnya saat menggunakan media komunikasi bendera, tanda asap, merpati pos, surat, dan sebagainya. Kini, melalui pemanfaatan *smartphone* manusia mampu berkomunikasi secara *realtime* tanpa batasan jarak komunikasi (Charon Int Trading Ltd, 2014).

Namun, penyandang disabilitas menghadapi perkembangan teknologi *smartphone* dengan sejumlah perbedaan. Penyandang disabilitas merupakan mereka yang memiliki keterbatasan fisik, mental dan sensorik jangka panjang. Keterbatasan yang dimiliki penyandang tunanetra seringkali menimbulkan hambatan untuk berpartisipasi secara penuh dan efektif di dalam masyarakat dengan dasar kesetaraan satu sama lain (United Nations, 2006). Keterbatasan tersebut juga menyebabkan penyandang tunanetra menjadi pengguna *smartphone* yang tidak efektif dan efisien. Sehingga, tidak heran hingga saat ini penyandang disabilitas tetap sering terabaikan dalam proses komunikasi.

Penyandang disabilitas dapat diklasifikasikan berdasarkan jenis keterbatasan yang dimilikinya. Klasifikasi disabilitas berdasarkan jenis keterbatasannya terdiri

dari disabilitas melihat, mendengar, berjalan atau naik tangga, mengingat atau konsentrasi, mengurus diri sendiri dan sebagainya. Penyandang tunanetra (jenis disabilitas melihat) merupakan kategori penyandang disabilitas yang jumlahnya paling banyak di Indonesia (Kementrian Kesehatan RI, 2014). Pada tahun 2012, penyandang tunanetra di Indonesia berjumlah 3,5 juta orang. Angka ini menempatkan Indonesia pada posisi kedua sebagai negara dengan angka penyandang tunanetra tertinggi di dunia (Merdeka.com, 2012).

Penyandang tunanetra merupakan kategori disabilitas yang memiliki keterbatasan fisik untuk melakukan pekerjaan yang memerlukan kemampuan visualisasi. Misalnya, untuk membaca teks penyandang tunanetra memerlukan bantuan sistem *Braille* ataupun sistem penghasil suara digital (Neto & Fonseca, 2014). Keterbatasan tersebut tentunya menciptakan sebuah batasan bagi penyandang tunanetra dalam berkomunikasi dengan memanfaatkan teknologi *smartphone*. Hal ini disebabkan karena teknologi *smartphone* yang populer digunakan saat ini tidak dirancang untuk digunakan oleh penyandang tunanetra. Untuk mengoperasikannya, *smartphone* masih membutuhkan kemampuan visualisasi dari penggunanya.

Faktor keterbatasan dalam interaksi visual menyebabkan hanya sedikit penyandang tunanetra yang bisa memanfaatkan teknologi *smartphone* dalam berkomunikasi jarak jauh. *Smartphone* pertama yang dirancang khusus bagi penyandang tunanetra, *Braille Phone,* juga masih memerlukan sejumlah tahap pengembangan lagi sebelum akhirnya dirilis (Lane, 2013). Oleh karena itu, hingga saat ini penyandang tunanetra memiliki ketergantungan terhadap bantuan orang lain jika ingin menggunakan *smartphone* dalam berkomunikasi. Untuk mengatasi masalah keterbatasan dalam pengoperasian *smartphone* sekaligus mendukung proses komunikasi jarak jauh penyandang tunanetra melalui *smartphone*, dibutuhkan sebuah perangkat lunak yang akan dirancang dan dikonstruksi melalui skripsi ini.

1.2 Rumusan Masalah

Pertanyaan penelitian yang dibahas dalam bagian ini adalah sebagai berikut:

1. Bagaimana rancangan sistem aplikasi *messaging* pada sistem operasi Android yang mampu mendukung komunikasi jarak jauh penyandang tunanetra?
2. Bagaimana implementasi sistem aplikasi *messaging* berbasis *voice interaction* pada sistem operasi Android dengan menggunakan pustaka Android *Speech-To-Text* dan Android *Text-To-Speech*?
3. Bagaimana tingkat kemudahan berdasarkan *Quality-of-Experience* dari penggunaan sistem aplikasi *messaging* berbasis *voice interaction* pada *smartphone* Android?

1.3 Tujuan

Tujuan yang ingin dicapai dari penelitian ini terbagi menjadi 2 bagian, yaitu:

1. Tujuan Umum (*Aim*)
   Tujuan umum yang ingin dicapai dari penelitian ini adalah membangun sistem aplikasi *messaging* berbasis *voice interaction* pada sistem operasi Android untuk mendukung komunikasi jarak jauh bagi penyandang tunanetra.
2. Tujuan-tujuan Khusus (*Objectives*)
   a. Merancang sistem aplikasi *messaging* berbasis *voice interaction* dengan pemodelan berorientasi objek.
   b. Mengimplementasikan sistem aplikasi *messaging* berbasis *voice interaction* dengan pendekatan berorientasi objek.
   c. Memastikan sistem aplikasi *messaging* berbasis *voice interaction* yang telah dibangun memiliki tingkat kemudahan penggunaan yang baik berdasarkan parameter *Quality-of-Experience*.

1.4 Manfaat

Melalui dilakukan penelitian ini, terdapat sejumlah manfaat yang dihasilkan, diantaranya adalah:

1. Terciptanya sebuah perangkat lunak *messaging* berbasis *voice interaction* yang mampu mendukung penyandang tunanetra dalam melakukan komunikasi jarak jauh.
2. Penyandang tunanetra tidak lagi harus bergantung pada bantuan orang lain dalam menggunakan teknologi *smartphone* untuk berkomunikasi.

1.5 Batasan Masalah

Ruang lingkup masalah yang dibahas di dalam penelitian ini dibatasi oleh hal-hal berikut:

1. Aplikasi *messaging* berbasis *voice interaction* dirancang untuk digunakan pada sistem operasi *Android* dengan minimal *API* 15 (Android 4.1.2 atau *Ice Cream Sandwich*).
2. Mekanisme layanan *voice interaction* yang terdapat pada aplikasi dibentuk melalui penggunaan layanan Android *Speech-To-Text* dan Android *Text-To-Speech.*
3. Bahasa yang digunakan pada layanan *voice interaction* yang terdapat pada aplikasi adalah Bahasa Inggris.

1.6 Sistematika Pembahasan

Struktur pembahasan yang digunakan dalam skripsi ini adalah sebagai berikut:

1. BAB 1 Pendahuluan

   Bagian ini menjelaskan seluruh hal yang menjadi dasar dan ruang lingkup proses penelitian di dalam skripsi. Setiap elemen dalam bagian ini akan menjadi acuan dalam penulisan bab-bab berikutnya.

2. BAB 2 Landasan Kepustakaan

   Bagian ini menguraikan seluruh aspek pengetahuan yang dibutuhkan dalam skripsi berdasarkan masalah atau tema yang diangkat. Hal-hal yang diuraikan tersebut dapat berbentuk teori, konsep, model, metode, sistem dari literatur ilmiah hingga penelitian-penelitian terkait sebelumnya.

3. BAB 3 Metodologi

   Bagian ini menguraikan satu atau sekumpulan metode yang menjadi aliran proses pengerjaan skripsi. Setiap tahap dalam aliran proses tersebut akan mengarahkan penelitian ini dari masalah atau topik yang diangkat menuju jawaban atau solusinya.

4. BAB 4 Analisis dan Perancangan

   Bagian ini menjelaskan seluruh aspek yang menjadi jawaban atau solusi atas masalah atau topik yang diangkat dalam skripsi ditinjau dari sudut pandang proses pemodelannya. Proses pemodelan tersebut terbagi menjadi dua tahap, yaitu tahap analisis dan perancangan.

5. BAB 5 Implementasi dan Pengujian

   Bagian ini menjelaskan proses pengimplementasian model yang telah dibentuk sebelumnya menjadi bentuk yang lebih konkret. Proses implementasi ini nantinya akan diikuti oleh proses pengujian untuk memastikan hasil implementasi tersebut dapat menjadi jawaban atau solusi yang nyata atas masalah atau topik yang diangkat dalam skripsi.

6. BAB 6 Penutup

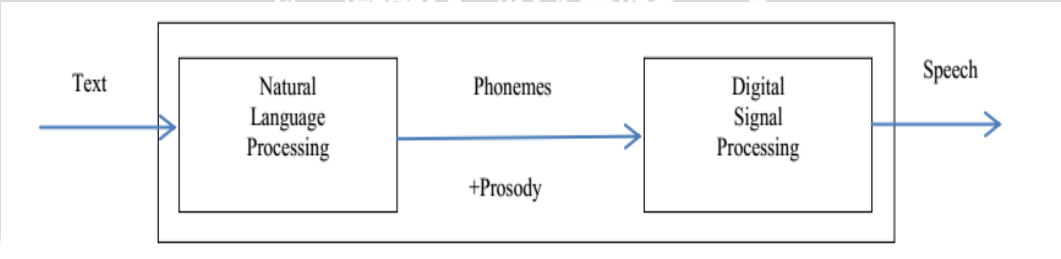
   Bagian ini menjelaskan kesimpulan akhir dari penelitian yang telah dilakukan dan saran yang ditujukan untuk pengembangan pada penelitian berikutnya. Proses pengambilan kesimpulan dan pemberian saran tersebut dilakukan dengan memanfaatkan seluruh aspek pengetahuan yang didapatkan melalui proses penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

Bagian ini menguraikan seluruh aspek pengetahuan yang dibutuhkan dalam skripsi berdasarkan masalah atau tema yang diangkat. Hal-hal yang diuraikan tersebut dapat berbentuk teori, konsep, model, metode, sistem dari literatur ilmiah hingga penelitian-penelitian terkait sebelumnya. Pada bagian ini akan diuraikan studi terkait *Text to Speech*, *Speech Recognition, Google Cloud Messaging* (GCM) dan pengujian perangkat lunak.

2.1 Text to Speech

Speech Synthesizers merupakan sebuah sistem yang dibuat dengan tujuan untuk menciptakan sebuah gelombang suara buatan dari suatu bentuk data secara otomatis. Secara khusus, *Text to Speech* (TTS) *System* merupakan salah satu bentuk *speech synthesizer system* berbasis komputer yang berfungsi untuk mengubah data teks menjadi gelombang suara. Sebuah TTS *System* terdiri dari dua buah modul utama, yaitu modul *Natural Language Processing* (NLP) dan modul *Digital Signal Processing* (DLP). Modul NLP berfungsi untuk memproduksi transkripsi fonetik dari sebuah masukan dalam bentuk teks. Dalam sejumlah kasus, modul ini juga dapat berfungsi untuk menghasilkan intonasi, durasi, intensitas dan *power* gelombang suara buatan yang sesuai dengan masukan teksnya. Sedangkan modul DLP berfungsi untuk mengubah hasil keluaran modul NLP ke dalam bentuk *speech* (Dagba & Boco, 2014).



Gambar 2.1 Struktur *Text-to-Speech* (TTS)

Sumber: (Dagba & Boco, 2014)

2.1.1 Android *Text-To-Speech*

Android *Text-To-Speech* merupakan sebuah fitur bawaan yang memungkinkan perangkat Android untuk "mengatakan" atau "membacakan" teks dengan bahasa yang berbeda-beda. Fitur ini pertama kali dikenalkan sejak Android versi 1.6 dan dibuka bagi para pengembang melalui *TTS API* (*Application Programming Interface). TTS API* terdiri dari dua aspek utama, yaitu *Languages* dan *Resources. TTS Resource* merupakan bahasa yang didukung dalam proses "pembacaan teks". Hingga saat ini, *TTS Engine* mendukung "pembacaan" teks dengan menggunakan bahasa Inggris, Prancis, Jerman, Italia dan Spanyol. Setiap "pembacaan" teks yang dilakukan oleh *TTS Engine* merupakan "pembacaan" yang bersifat *language specific resource.* Hal ini didasari oleh adanya perbedaan

pelafalan kata menurut bahasa yang digunakan, seperti kata "Paris" yang akan berbeda-beda ketika dilafalkan dengan menggunakan Bahasa Prancis dan Bahasa Inggris (Android Developer, 2009).

Meskipun seluruh perangkat Android secara *default* sudah didukung oleh *TTS Engine,* sejumlah perangkat memiliki keterbatasan media penyimpanan dan mungkin kekurangan *language-spesific resource.* Jika pengguna ingin meng-*install resource* tersebut, *TTS API* memungkinkan aplikasi untuk melakukan pengecekan ketersedian *resource* tersebut dan melakukan inisiasi pengunduhan hingga pemasangannya (*installation)* (Android Developer, 2009). Untuk melakukan pengecekan terhadap ketersediaan *TTS Resource* dapat dilakukan melalui penggunaan kode berikut:

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

Kode 2.1 Kode sumber untuk melakukan pengecekan ketersediaan *TTS Resource*

Sumber: (Android Developer, 2009)

Sebuah pengecekan yang berhasil akan ditandai oleh kode hasil CHECK_VOICE_DATA_PASS. Kode hasil tersebut mengindikasikan bahwa perangkat Android sudah siap untuk "membacakan" teks setelah pembuatan objek android.speech.tts.TextToSpeech. Namun, jika pengecekan tersebut tidak berhasil, maka aplikasi harus mengizinkan pengguna untuk mengetahuinya dan melakukan pemasangan *(installation)* data yang dibutuhkan agar perangkat dapat menjadi *multi-lingual talking machine.* Pengunduhan dan pemasangan data tersebut dapat dilakukan melalui penggunaan ACTION_INSTALL_TTS_DATA *intent. Intent* ini akan mengarahkan pengguna untuk masuk ke Android *Market* dan menginisiasikan proses pengunduhan *TTS Resource.* Proses pemasangan akan dilakukan secara otomatis setelah proses pengunduhan selesai (Android Developer, 2009). Contoh implementasi hal tersebut dapat dilihat melalui Kode 2.2.

Aspek berikutnya yang dimiliki oleh *TTS Engine* adalah *Languages* dan *Local*. Setelah seluruh komponen dari *TTS* yang diinginkan berhasil dimuat, maka untuk memulai proses pembacaan teks, dapat dilakukan melalui penggunaan *method speak()* (Android Developer, 2009). Kode sumber untuk memuat bahasa dan melakukan proses pembacaan teks, masing-masing dapat dilihat melalui Kode 2.3 dan Kode 2.4.

```
1  private TextToSpeech mTts;
2  protected void onActivityResult(
3          int requestCode, int resultCode, Intent data) {
4      if (requestCode == MY_DATA_CHECK_CODE) {
5          if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS)
6  {
7              // success, create the TTS instance
8              mTts = new TextToSpeech(this, this);
9          } else {
10             // missing data, install it
11             Intent installIntent = new Intent();
```

```
12              installIntent.setAction(
13                  TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
14              startActivity(installIntent);
15          }
16      }
17  }
```

Kode 2.2 Kode sumber untuk melakukan otomatisasi pengecekan dan pengunduhan *TTS Resource Data*

Sumber: (Android Developer, 2009)

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

Kode 2.3 Kode sumber untuk memuat Bahasa

Sumber: (Android Developer, 2009)

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

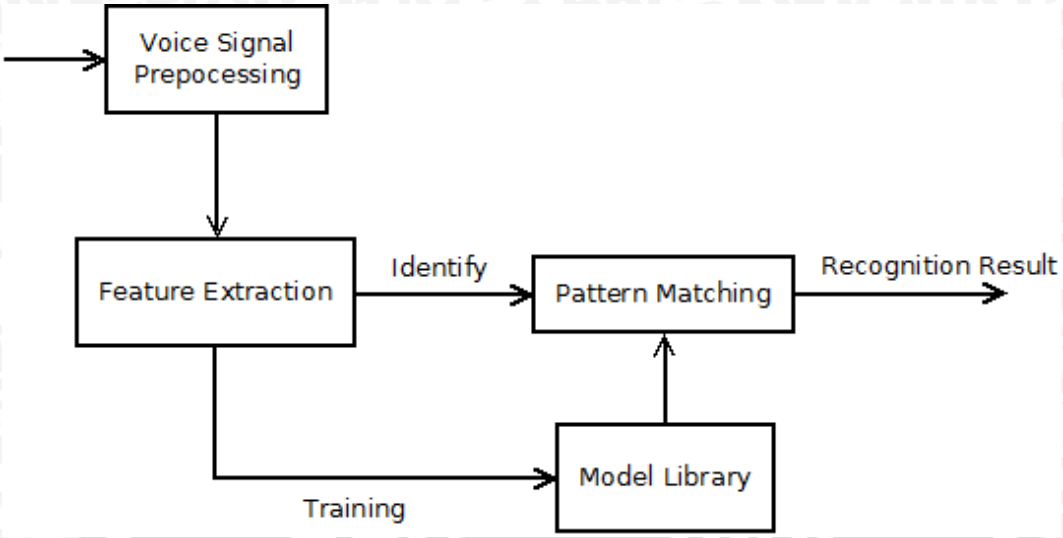
Kode 2.4 Kode sumber untuk membuat aplikasi melakukan pembacaan teks

Sumber: (Android Developer, 2009)

2.2 *Speech Recognition*

Speech recognition adalah suatu mesin yang dibuat dengan tujuan untuk mengidentifikasi dan memahami serta melakukan reaksi sesuai dengan perintah yang dikatakan oleh manusia. Sistem ini bekerja dengan cara menggunakan suara sebagai objek pengamatannya. Suara sebagai objek pengamatan tersebut akan diidentifikasi dan dipahami oleh mesin melalui pemrosesan sinyal suara dan pengenalan pola. Hingga pada saat ini, teknologi *speech recognition* sudah menjadi teknologi tingkat tinggi yang memampukan mesin untuk mengubah sinyal suara yang diberikan oleh manusia menjadi teks yang sesuai ataupun perintah yang sesuai (Meng, et al., 2012).

Sejak Bell Labs Speech Recognition System memulai penelitian tentang *speech recognition* sekitar tahun 1950 hingga perkembangannya saat ini, *speech recognition system* sudah banyak dipengaruhi oleh sejumlah disiplin ilmu. Proses perkembangan *speech recognition* sangat dekat dengan disiplin ilmu pengetahuan tentang suara (*acoustic),* pengklasifikasian bunyi yang diucapkan *(phonetics),* ilmu bahasa (*linguisctics),* teori informasi (*information theory),* pengenalan pola (*pattern recognition)* dan ilmu tentang neurobiologi. Masing-masing disiplin ilmu tersebut memiliki perannya tersendiri dalam mendukung proses perkembangan *speech recognition*. Hingga saat ini, s*peech recognition system* pertama di dunia yang memiliki performansi tinggi, tidak spesifik dan memiliki perbendaharaan kosakata yang tinggi menerapkan *HMM Model, Artificial Neural Network (ANN)* dan metode *VQI-IMM.* Pada dasarnya, *speech recognition system* ini terbentuk di atas sebuah *pattern recognition system* yang di dalamnya termasuk *feature extraction, pattern matching* dan *reference model library.* Struktur dasar sistem tersebut dapat dilihat pada gambar berikut (Meng, et al., 2012):



Gambar 2.2 Prinsip Dasar *Speech Recognition System*

Sumber: (Meng, et al., 2012)

2.2.1 Android *Speech Recognizer*

Android *Speech Recognizer* merupakan sebuah layanan *speech recognition* yang disediakan oleh Google pada *platform* Android melalui Android SDK. Layanan ini memiliki *word error rate* sebesar 13,5% (McGraw, et al., 2016) dan diperkenalkan melalui Android API level 8 atau yang lebih dikenal dengan kode versi FROYO. Android *Speech Recognizer* bekerja dengan melakukan *streaming audio* ke *remote server* milik Google. Kemudian *server* Google akan melakukan proses pengenalan suara dan mengirimkan hasilnya dalam bentuk teks kembali ke *client*. Melalui dokumentasi yang dibuat oleh Google, Google menyatakan bahwa layanan ini tidak dimaksudkan untuk digunakan sebagai layanan pengenalan suara yang digunakan secara terus menerus karena akan mengonsumsi baterai dan *bandwidth* dalam jumlah besar. Namun, melalui Android API level 23, Google memberikan pengembangan yang cukup signifikan terhadap layanan *speech recognizer* miliknya. Google memperkenalkan opsi penggunaan secara *offline* pada layanan *speech recognizer*-nya sehingga dengan kata lain, proses pengenalan suara tidak perlu lagi dilakukan pada *server* Google, cukup pada *client* atau *smartphone* Android sehingga dapat meminimalisir penggunaan baterai dan meniadakan penggunaan *bandwith* (Android Developer, 2015).

Untuk mengakses layanan Android *Speech Recognizer* dapat dilakukan dengan memanfaatkan *class SpeechRecognizer*. *Class SpeechRecognizer* dapat digunakan apabila aplikasi yang dibuat telah memiliki *RECORD_AUDIO permission*. Pada *class* ini terdapat sebuah *factory method* dengan nama *createSpeechRecognizer* yang memungkinkan kita untuk membuat sebuah *SpeechRecognizer* baru. Penggunaan *factory method* dari *class SpeechRecognizer* ini lebih dianjurkan oleh Google dibandingkan dengan melakukan *instantiation* secara langsung pada *class SpeechRecognizer* itu sendiri (Android Developer,

2015). Pembuatan *SpeechRecognizer* baru dapat dilakukan melalui penggunaan kode berikut:

```
1  speechRecognizer = SpeechRecognizer.createSpeechRecognizer(context);
```

Kode 2.5 Kode sumber untuk membuat *instance SpeechRecognizer* baru

Sumber : (Android Developer, 2015)

Untuk memulai proses pengenalan suara, dapat dilakukan melalui penggunaan kode berikut:

```
1  speechRecognier.startListening(recognizerIntent);
```

Kode 2.6 Kode sumber untuk memulai proses pengenalan suara

Sumber : (Android Developer, 2015)

RecognizerIntent merupakan sebuah Intent yang berfungsi sebagai parameter untuk mendukung sebuah layanan *speech recognition*. Sebuah layanan *speech recognition* dapat dikonfigurasi dengan memberikan sejumlah pilihan *extras* pada recognizerIntent (Android Developer, 2015). Untuk membuat sebuah SpeechRecognizer yang dapat berjalan secara *offline* dan menggunakan Bahasa Inggris sebagai bahasa pengenalan *default*-nya, dapat dilakukan melalui penggunaan recognizerIntent seperti yang terlihat pada kode berikut:

```
1  recognizerIntent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
2  recognizerIntent.putExtra(RecognizerIntent.EXTRA_PREFER_OFFLINE, false);
3  recognizerIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, "en-US");
```

Kode 2.7 Kode sumber untuk membuat SpeechRecognizer yang berjalan secara *offline* dan menggunakan Bahasa Inggris sebagai bahasa *default*-nya.

Sumber : (Android Developer, 2015)

2.3 Firebase

Firebase merupakan sebuah layanan infrastruktur *backend-as-a-service* (BaaS) yang diakuisisi oleh Google pada Oktober 2014 silam (Tamplin, 2016). Firebase menawarkan kemudahan kepada para pengembang perangkat lunak dalam membangun aplikasi yang lebih baik serta mengembangkan bisnis yang sukses melalui seluruh fitur komplementernya. Saat ini, Firebase sudah memiliki fitur *Analytics*, *Cloud Messaging*, *Authentication*, *Realtime Database*, *Storage*, *Hosting*, *Test Lab*, *Crash Reporting*, *Notifications*, *Remote Config*, *App Indexing*, *Dynamic Links*, *Invites*, *AdWords*, dan *AdMob*. Seluruh fitur tersebut dikemas dalam sebuah SDK Firebase tunggal sehingga dengan kemudahan yang ditawarkan para pengembang perangkat lunak dapat fokus untuk memecahkan masalah *customer* melalui perangkat lunak yang dibuatnya dan tidak menghabiskan banyak waktu dalam membangun infrastruktur yang kompleks (Google, 2016).

Untuk menambahkan layanan Firebase pada sebuah *project* Android Studio, maka kebutuhan minimum yang diperlukan adalah Android 2.3 (Gingerbread)

dan Google Play Service 10.0.1 (Google, 2017). Proses integrasi pustaka Firebase dengan *project* Android Studio dapat dilakukan dengan cara menambahkan google-service *plugin* pada *root-level file* build.grade seperti yang ditunjukkan oleh kode di bawah ini:

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

Kode 2.8 Kode Sumber untuk menambahkan google-service *plugin* pada *root-level file* build.gradle

Sumber: (Google, 2017)

Kemudian menambahkan baris *apply plugin* dan *dependencies* dari *Firebase* SDK yang ingin digunakan pada berkas modul Gradle seperti yang terlihat pada kode sumber di bawah ini:

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:10.0.1'
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Kode 2.9 Kode Sumber untuk menambahkan baris *apply plugin* dan *dependencies* dari Firebase SDK pada berkas modul Gradle

Sumber: (Google, 2017)

2.3.1 Firebase *User Authentication*

Firebase *User Authentication* merupakan sebuah layanan siap pakai yang dimiliki oleh Firebase SDK. Firebase *User Authentication* memungkinkan aplikasi untuk melakukan *user authentication* menggunakan berbagai macam metode *authentication* seperti *email and password based authentication*, *federated identity provider identegration* (*authentication* menggunakan akun Google, Facebook, Twitter atau Github), *custom authentication system integration* hingga *anonymous authentication*. Firebase *User Authentication* ini bekerja dengan cara mengirimkan *server response* dari Firebase *Server* berdasarkan *credential* yang dikirimkan oleh *client* ke Firebase *Server*. *Credential* tersebut dapat berupa alamat *email* dan *password* ataupun sebuah *token* OAuth dari sebuah *federated identity provider*. Melalui *server response* yang diterima dari Firebase *Server*, aplikasi dapat mengakses informasi dasar profil pengguna dan mengontrol akses pengguna terhadap produk atau layanan Firebase yang terdapat pada aplikasi. Sebelum mulai menggunakan Firebase *User Authentication* pada *project* Android

Studio, maka hal yang perlu dilakukan adalah dengan melakukan *setup sign method* pada Firebase *Console*. Setelah hal tersebut dilakukan, maka aplikasi dapat mengirimkan *user credential* ke Firebase *server* melalui Firebase SDK untuk mendapatkan informasi dasar profil pengguna dan mengontrol akses pengguna pada aplikasi (Google, 2017). Untuk mendapatkan informasi pengguna yang sedang *login* dapat dilakukan dengan menggunakan kode sumber di bawah ini:

```
1  mAuthListener = new FirebaseAuth.AuthStateListener() {
2      @Override
3      public void onAuthStateChanged(@NonNull FirebaseAuth
   firebaseAuth) {
4          FirebaseUser user = firebaseAuth.getCurrentUser();
5          if (user != null) {
6              // User is signed in
7              Log.d(TAG, "onAuthStateChanged:signed_in:" +
   user.getUid());
8          } else {
9              // User is signed out
10             Log.d(TAG, "onAuthStateChanged:signed_out");
11         }
12         // ...
13     }
14 };
```

Kode 2.10 Kode sumber untuk mendapatkan informasi pengguna yang sedang *login*.

Sumber: (Google, 2017)

2.3.2 Firebase *Realtime Database*

Firebase *Realtime Database* merupakan sebuah layanan NoSQL *cloud-hosted database* yang dimiliki oleh Firebase SDK. Layanan ini menawarkan penyimpanan data yang dapat disinkronisasikan secara *realtime* terhadap seluruh *client* yang terhubung. Layanan ini memiliki 3 kemampuan inti yaitu *realtime*, *offline* dan *accessible from client devices* (Google, 2017).

Maksud dari *realtime* adalah jika terdapat perubahan pada data pada *database*, maka seluruh *client* yang terhubung akan secara otomatis mendapatkan perubahannya dalam hitungan milidetik. Kemudian *offline*, yaitu aplikasi yang menggunakan Firebase *Realtime Database* akan tetap responsif bahkan saat *offline*. Hal ini disebabkan karena Firebase SDK dapat mempertahankan data dan perubahannya pada media penyimpanan *client*. Pada saat *client* terhubung ke jaringan internet, maka Firebase SDK akan melakukan penyesuaian otomatis atas catatan perubahan data yang disimpan pada media penyimpanan *client* dengan kondisi terkini dari Firebase *server*. Kemampuan inti yang terakhir adalah *accessible from client devices*. Layanan ini menawarkan kemudahan untuk mengakses Firebase *Realtime Database* secara langsung dari sebuah perangkat *mobile* atau sebuah peramban *web* tanpa membutuhkan *server application* (Google, 2017).

Untuk melakukan penulisan pada Firebase *Realtime Database*, maka dapat dilakukan melalui kode berikut:

```
1  FirebaseDatabase database = FirebaseDatabase.getInstance();
2  DatabaseReference myRef = database.getReference("message");
```

```
3   myRef.setValue("Hello, World!");
```

Kode 2.11 Kode sumber untuk melakukan penulisan pada Firebase *Realtime Database*

Sumber: (Google, 2017)

Sedangkan untuk melakukan pembacaan dari Firebase *Realtime Database*, maka dapat dilakukan melalui kode sumber berikut:

```
1   // Read from the database
2   myRef.addValueEventListener(new ValueEventListener() {
3       @Override
4       public void onDataChange(DataSnapshot dataSnapshot) {
5           // This method is called once with the initial value and
    again
6           // whenever data at this location is updated.
7           String value = dataSnapshot.getValue(String.class);
8           Log.d(TAG, "Value is: " + value);
9       }

10      @Override
11      public void onCancelled(DatabaseError error) {
12          // Failed to read value
13          Log.w(TAG, "Failed to read value.", error.toException());
14      }
15  });
```

Kode 2.12 Kode sumber untuk melakukan pembacaan dari Firebase *Realtime Database*

Sumber: (Google, 2017)

2.4 Pengujian Perangkat Lunak

Tujuan dari dilakukannya proses pengujian perangkat lunak adalah meningkatkan kualitas dari perangkat lunak atau dapat pula disebut untuk mencapai tingkat *realiability* yang lebih terjamin. Pada umumnya, pengujian dilakukan pada setiap tahap atau fase dari pengembangan perangkat lunak guna memvalidasi dan memverifikasi perangkat lunak itu sendiri serta melakukan kontrol terhadap kualitasnya (Kapur, et al., 2014). Namun, pengujian perangkat lunak pada penelitian ini hanya dilakukan pada tahap atau fase paling akhir dari proses pengembangan perangkat lunak. Metode pengujian yang dilakukan adalah metode *Black-box Testing, Mobile Compatibility Testing* dan *Mean Opinion Score (MOS) Testing.*

2.4.1 *Black-box Testing*

Menurut Pressman (2012), *Black-box testing* (pengujian kotak hitam) atau yang dikenali pula dengan istilah pengujian perilaku merupakan metode pengujian yang didasarkan pada verifikasi dan validasi kebutuhan fungsional dari sistem perangkat lunak. Pengujian ini berfokus pada ranah informasi dari setiap kebutuhan fungsional tanpa memperhatikan struktur kendalinya. Sehingga dalam proses pelaksanaannya, metode pengujian ini hanya memerlukan pemberian masukan pada setiap kondisi dari seluruh pelaksanaan kebutuhan fungsional yang ada. Metode pengujian ini nantinya akan menghasilkan kesalahan-kesalahan dari perangkat lunak dengan kategori sebagai berikut:

1. fungsi yang salah atau hilang,
2. kesalahan antarmuka,
3. kesalahan dalam struktur data atau akses basis data eksternal,
4. kesalahan perilaku atau kinerja dan
5. kesalahan inisialisasi dan penghentian

Pada penelitian ini, teknik pengujian dari metode *black-box testing* yang digunakan adalah teknik pengujian validasi. Pada penerapan teknik pengujian ini, dilakukan proses validasi kesesuaian seluruh kondisi rancangan dengan seluruh kondisi hasil implementasinya untuk setiap *use case.*

2.4.2 *Mobile Compatibility Testing*

Mobile compatibilty testing merupakan salah satu metode pengujian aplikasi *mobile.* Pengujian ini memiliki tujuan untuk mengetahui tingkat ketergantungan suatu aplikasi terhadap perbedaan lingkungan (*environment*) dimana aplikasi tersebut dijalankan. Pada umumnya, proses pengujian ini dilakukan terhadap 3 masalah *compatibility* utama, yaitu *platform compatibility, device feature compatibility* dan *native API compatibility. Platform compatibility* mengacu pada pengecekan apakah aplikasi dapat berjalan dengan baik pada sejumlah *mobile platform* yang berbeda dengan beragam versi. *Device feature compatibility* mengacu pada pengecekan apakah aplikasi *compatible* terhadap sejumlah perangkat *mobile* dengan *hardware feature* yang berbeda. Sedangkan *native API compatibility* mengacu pada proses memastikan apakah aplikasi *compatible* terhadap *native API* yang berbeda versi (Zhang, et al., 2015).

Pada penelitian ini, *mobile compatibility testing* yang dilakukan hanya berfokus pada masalah Android *platform compatibility.* Hal ini dilakukan karena aplikasi yang dikembangkan dirancang khusus untuk digunakan pada sistem operasi Android. Selain itu, aplikasi yang dikembangkan juga hanya menggunakan *hadware feature* yang bersifat umum serta dan *native API* yang hanya memiliki versi tunggal. *Evaluator* dari proses pengujian ini adalah peneliti.

2.4.3 *Mean Opinion Score* (*MOS*) *Testing*

Menurut Jie Xu, dkk (2011), *Mean Opinion Score* (*MOS*) *testing* merupakan salah satu metode pengujian yang bersifat subjektif. Metode pengujian MOS digunakan untuk mengetahui tingkat kualitas dari sebuah layanan atau produk berdasarkan pengalaman penggunanya (*Quality-of-Experience*). Pada penelitian ini, *MOS testing* diterapkan untuk mengukur *Quality-of-Experience* dari layanan *voice interaction* pada aplikasi.

Penerapan MOS *testing* tersebut melibatkan 10 orang sebagai respondennya. Jumlah responden ini dipilih dengan mengacu pada rekomendasi International Telecommunication Union (ITU) melalui dokumennya yang berjudul *Subjective Audiovisual Quality Assessment Methods For Multimedia Applications.* Pada dokument tersebut, ITU merekomendasikan jumlah subjek yang memungkinkan pada pengujian melihat dan mendengar adalah sebanyak 6 sampai 40 orang. Jumlah ini sama dengan jumlah subjek pengujian yang umumnya digunakan pada

usability testing dari sebuah layanan (International Telecommunication Union, 1998). Selain itu, pemilihan jumlah responden sebanyak 10 orang juga dilakukan berdasarkan rekomendasi batas minimum dalam analisis statistik serta penelitian eksperimental sederhana (Roscoe, 1975).

Untuk mengetahui tingkat kualitas dari layanan *voice interaction* yang terdapat pada aplikasi, maka setiap subjek pengujian yang ada diharapkan memberikan opininya terkait layanan tersebut dengan menggunakan skala opini yang direkomendasikan oleh ITU. Skala opini yang direkomendasikan oleh ITU dapat dilihat melalui Tabel 2.1.

Tabel 2.1 Skala Opini yang Direkomendasikan Oleh International Telecommunication Union

Opinions	Scales
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

Sumber: (International Telecommunication Union, 1996)

Menurut Jie Xu, dkk, untuk menginterpretasikan rata-rata nilai opini (*MOS*) terkait kualitas layananan *voice interaction* yang terdapat pada aplikasi, dapat dilakukan melalui penggunaan rumus:

$$MOS = \frac{1}{N}\sum_{i=1}^{N} y_i$$

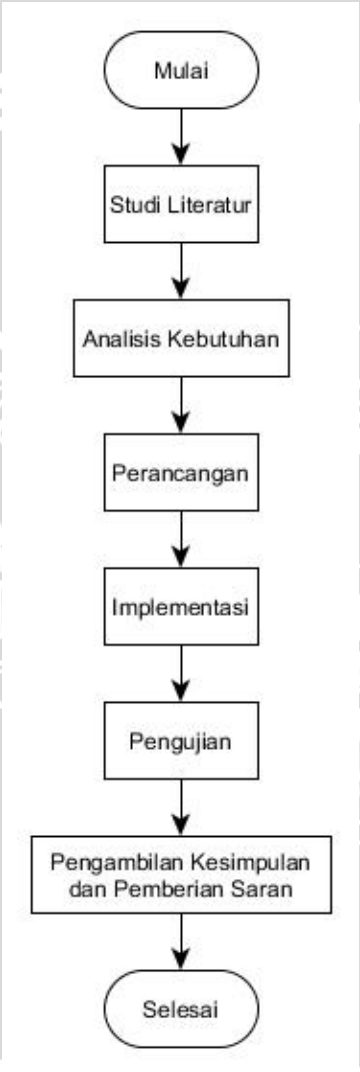
Keterangan:

N = Jumlah responden;

y_i = Nilai opini responden ke-i.

BAB 3 METODOLOGI

Rancang bangun aplikasi *messaging* berbasis *voice interaction* bagi penderita tunanetra pada sistem operasi Android merupakan sebuah penelitian dengan jenis penelitian implementatif pengembangan keminatan *mobile*. Penelitian ini memiliki cakupan pendekatan rancang bangun aplikasi atau sistem baru. Proses pelaksanaan penelitian ini terdiri dari tahap studi literatur, analisis kebutuhan, perancangan, implementasi, pengujian dan pengambilan kesimpulan serta pemberian saran. Aliran proses pelaksanaan penelitian tersebut dapat dilihat melalui Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Studi Literatur

Studi literatur merupakan sebuah proses pengumpulan pengetahuan atau informasi pendukung yang dibutuhkan dalam suatu proses penelitian. Pada umumnya, proses ini dilakukan dengan memanfaatkan jurnal atau *paper* penelitian, buku, *web article* dan berbagai sumber pengetahuan atau informasi yang memiliki tingkat kredibilitas tinggi lainnya. Proses studi literatur pada penelitian ini memiliki cakupan *Text-to-Speech,* Android *Text-to-Speech, Speech Recognition,* Android *Speech Recognition,* Firebase, Firebase *User Authentication,* Firebase *Realtime Database*, Pengujian Perangkat Lunak, *Black-box Testing, Mobile Compability Testing* dan *Mean Opinion Score* (MOS) *Testing.*

3.2 Analisis Kebutuhan

Analisis kebutuhan merupakan salah satu tahap yang terdapat pada proses pengembangan perangkat lunak. Tahap ini dilakukan dengan tujuan untuk mengidentifikasi dan menganalisis kebutuhan dari perangkat lunak yang akan dikembangkan. Umumnya, kebutuhan perangkat lunak dapat diidentifikasi melalaui berbagai cara, diantaranya melalui wawancara *stakeholder,* pengajuan kuesioner, *brainstorming,* observasi dan sebagainya.

Pada penelitian ini, proses analisis kebutuhan yang dilakukan menggunakan pendekatan *Object Oriented Analysis & Design (OOAD)* dengan pemodelan menggunakan *Unified Modeling Language (UML)*. Setiap kebutuhan perangkat lunak yang ada diidentifikasi melalui proses *brainstorming* dan observasi terhadap fungsi-fungsi dasar dari sebuah aplikasi *messaging.* Hasil analisis terhadap kebutuhan-kebutuhan tersebut kemudian didefinisikan dan dispesifikasikan melalui gambaran umum sistem, daftar kebutuhan perangkat lunak, diagram *use case* dan diagram *activity.*

3.3 Perancangan

Perancangan merupakan salah satu tahap yang terdapat pada proses pengembangan perangkat lunak. Tahap ini bertujuan untuk memodelkan suatu sistem perangkat lunak berdasarkan hasil analisis kebutuhan yang telah dilakukan sebelumnya. Pada penelitian ini, proses pemodelan sistem perangkat lunak dilakukan melalui perancangan diagram *sequence,* diagram *class,* basis data, antarmuka dan *page flow.*

3.4 Implementasi

Implementasi merupakan salah satu tahap yang terdapat pada proses pengembangan perangkat lunak. Tahap ini bertujuan untuk merealisasikan suatu model perangkat lunak ke dalam bentuk yang lebih utuh serta siap pakai. Pada penelitian ini, proses implementasi dilakukan dengan menggunakan aplikasi *Android Studio* yang dijalankan di atas sistem operasi *Windows 10*. Proses implementasi tersebut menghasilkan sebuah aplikasi Android *native* yang dibangun menggunakan Java *programming language* dan XML *markup language.*

3.5 Pengujian

Pengujian merupakan salah satu tahap yang terdapat pada proses pengembangan perangkat lunak. Tahap ini bertujuan untuk melakukan validasi dan verifikasi suatu perangkat lunak berdasarkan spesifikasi kebutuhan fungsional dan non-fungsionalnya. Pada penelitian ini, proses pengujian perangkat lunak dilakukan dengan menggunakan metode *Black-box Testing, Mobile (Android Platform) Compatibility Testing,* dan *Mean Opinion Score (MOS) Testing.*

3.6 Pengambilan Kesimpulan dan Pemberian Saran

Pengambilan kesimpulan dan pemberian saran merupakan proses akhir yang dilakukan pada penelitian ini. Proses ini terbagi menjadi 2 bagian, yaitu proses perumusan kesimpulan penelitian dan proses pemberian saran penelitian. Proses perumusan kesimpulan penelitian dilakukan untuk menjawab rumusan masalah yang telah dibuat sebelumnya. Kesimpulan penelitian tersebut dirumuskan dengan mengacu pada penelitian ini secara keseluruhan. Sedangkan proses pemberian saran penelitian dilakukan untuk memberikan saran-saran yang dianggap penting bagi penelitian berikutnya.

BAB 4 ANALISIS DAN PERANCANGAN

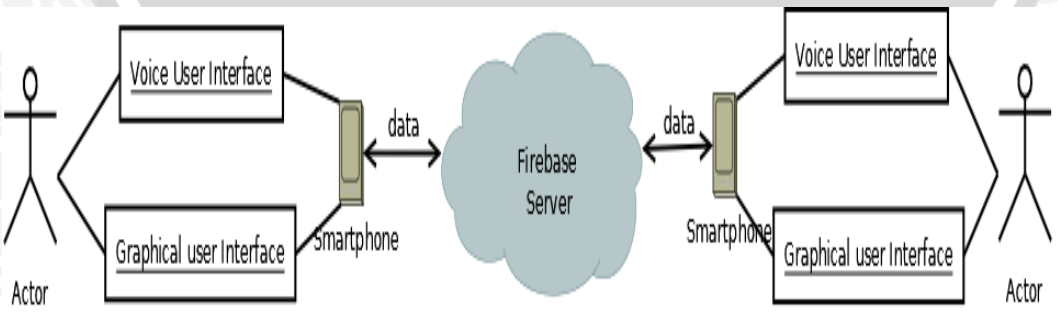
Pada bab ini akan dibahas tentang analisis dan perancangan dari aplikasi *messaging* berbasis *voice interaction* pada sistem operasi Android. Untuk mewujudkan proses analisis dan perancangan tersebut dilakukan pembuatan gambaran umum sistem, penentuan daftar kebutuhan, penggambaran sistem menggunakan bahasa pemodelan *UML Use Case* dan *Activity Diagram* yang dihimpun sebagai tahapan analisis kebutuhan sistem serta pembuatan rancangan arsitektur, penggambaran sistem mengunakan bahasa pemodelan *UML Sequence* dan *Class Diagram*, pembuatan rancangan basis data dan antarmuka sebagai tahap perancangan perangkat lunak.

4.1 Analisis Kebutuhan Sistem

Pada tahap analisis kebutuhan sistem dilakukan pembuatan gambaran umum sistem yang mampu mewujudkan solusi atas permasalahan yang diangkat dalam penelitian ini. Tahap ini terdiri dari:

4.1.1 Gambaran Umum Sistem

Aplikasi *messaging* berbasis *voice interaction* ini merupakan aplikasi yang dirancang khusus untuk membantu para penyandang tunanetra dalam berkomunikasi menggunakan perangkat *smartphone*. Melalui penggunaan aplikasi ini, para pengguna, khususnya penyandang tunanetra mampu saling mengirimkan pesan melalui jaringan internet dengan perantara sebuah *server*. Dalam penerapannya, aplikasi menawarkan pilihan *user interface* dengan sistem *graphical* atau *voice* kepada pengguna. Pada sistem *voice based user interface*, pengguna dapat berinteraksi kepada aplikasi dengan menggunakan sebuah *voice control*. Contoh dari kontrol suara (*voice control)* yang dimaksud adalah mengirimkan pesan, mendengarkan pesan masuk, menghapus pesan masuk, dan sebagainya. Bentuk interaksi terhadap aplikasi yang menggunakan kontrol suara ini akan direspon oleh aplikasi menggunakan keluaran dalam bentuk suara pula sehingga tercipta sebuah mekanisme *voice interaction*. Bahasa yang didukung sebagai bentuk interaksi pengguna terhadap aplikasi maupun interaksi aplikasi terhadap pengguna merupakan bahasa Inggris.



Gambar 4.1 Gambaran Umum Sistem

4.1.2 Daftar Kebutuhan

Aplikasi *messaging* berbasis *voice interaction* yang dirancang memiliki daftar kebutuhan yang dikelompokkan ke dalam dua kategori, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Untuk memberikan identitas terhadap sebuah kebutuhan, digunakan sebuah kode unik dengan ketentuan FRXX untuk identitas sebuah kebutuhan fungsional dan NFRXX untuk identitas sebuah kebutuhan non-fungsional.

Tabel 4.1 Tabel Kebutuhan Fungsional

Kode	Kebutuhan	Use case
FR01	Aplikasi harus mampu mengizinkan *guest* untuk mendaftar sebagai *member* ke dalam sistem.	*Sign Up*
FR02	Aplikasi harus mampu mengizinkan *guest* untuk masuk sebagai *member* ke dalam sistem.	*Sign In*
FR03	Aplikasi harus mampu mengizinkan *guest* dan *member* untuk melihat atau mendengarkan panduan penggunaan aplikasi.	*Help*
FR04	Aplikasi harus mampu mengizinkan *member* untuk menambahkan *member* lain ke dalam daftar kontaknya.	*Add Contact*
FR05	Aplikasi harus mampu mengizinkan *member* untuk menghapus *member* lain dari dalam daftar kontaknya.	*Delete Contact*
FR06	Aplikasi harus mampu mengizinkan *member* untuk mengirimkan pesan.	*Send Message*
FR07	Aplikasi harus mampu mengizinkan *member* untuk melihat atau mendengarkan pesan masuk yang belum	*Read Unread Messages*

	pernah dilihat atau didengarkan sebelumnya.	
FR08	Aplikasi harus mampu mengizinkan *member* untuk melihat atau mendengarkan percakapannya dengan *member* lain.	*Read Conversation*
FR09	Aplikasi harus mampu mengizinkan *member* untuk menghapus percakapannya dengan *member* lain.	*Delete Conversation*
FR10	Aplikasi harus mampu mengizinkan *member* untuk keluar dari sistem.	*Sign Out*

Tabel 4.1 merupakan tabel yang menunjukkan daftar kebutuhan fungsional dari sistem yang dirancang dan akan dibangun. Pada tabel ini, terdapat 10 kebutuhan fungsional yang akan dimodelkan menggunakan bahasa pemodelan *UML Use Case* serta diimplementasikan ke dalam wujud fitur aplikasi. Terhadap hasil implementasinya tersebut, nantinya akan dilakukan pengujian untuk mengetahui tingkat kesesuaiannya dengan hasil rancangan yang telah dibuat.

Tabel 4.2 Tabel Kebutuhan Non-fungsional

Parameter	Kode	Deskripsi Kebutuhan
Compatibility	NFR01	Aplikasi harus dapat dipasang dan digunakan dengan sempurna pada beberapa versi sistem operasi Android.
Usability	NFR02	Aplikasi harus mampu memberikan kemudahan kepada penggunanya, terutama bagi pengguna yang menyandang tunanetra. Kemudahan yang dimaksud dapat diberikan melalui ketersediaan kontrol sentuhan dan kontrol suara untuk berinteraksi dengan aplikasi. Kontrol suara yang diberikan oleh pengguna harus dapat diterima dan diproses oleh aplikasi dengan baik. Setiap kontrol suara tersebut juga harus mendapat feedback dari aplikasi dalam format audio untuk menjamin tingkat user experience yang baik dalam penggunaan aplikasi.

Reliability	NFR03	Aplikasi harus mampu melayani pengiriman pesan pengguna dalam waktu yang tidak terbatas.

Tabel 4.2 merupakan tabel yang menunjukkan daftar kebutuhan non-fungsional dari sistem yang dirancang dan akan dibangun. Masing-masing kebutuhan yang ada pada tabel ini digunakan untuk menjelaskan atribut kualitas *compatibility*, *usability* dan *reliability* dari aplikasi yang akan dibangun.

4.1.3 *Use Case Diagram*

Diagram Use Case merupakan diagram yang digunakan untuk memodelkan perangkat lunak berdasarkan kebutuhan fungsional perangkat lunak itu sendiri. Setiap kebutuhan fungsional yang dimaksud dimodelkan berdasarkan sudut pandang aktornya. Pada diagram use case yang ditunjukkan oleh gambar 4.2 terdapat 2 aktor yaitu *guest* yang merupakan pengguna biasa yang belum memiliki akun dan *member* yang merupakan pengguna yang telah memiliki akun serta terdapat 10 use case yang masing-masing mewakili setiap kebutuhan fungsional perangkat lunak yang telah dirancang sebelumnya.

Gambar 4.2 *Use Case* Aplikasi *Messaging* Berbasis *Voice Interactions*

Di bawah ini terdapat tabel yang menunjukkan skenario *use case sign up* dari aplikasi *messaging* berbasis *voice interactions*.

Tabel 4.3 Skenario *Use Case Sign Up*

Kode *Use Case*	FR01
Nama *Use Case*	*Sign Up*
Tujuan *Use Case*	Mengizinkan *guest* untuk mendaftar sebagai *member* ke dalam sistem.
Aktor	*Guest*
Prakondisi	*Guest* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	-
	Graphical User Interface
	1. *Guest* meng-*click* tombol *sign up.* 2. Aplikasi menampilkan halaman *sign up*. 3. *Guest* mengisi *form sign up.* 4. *Guest* meng-*click* tombol *sign up.* 5. Aplikasi menampilkan *toast feedback.*
Alur Alternatif	*Voice User Interface*
	-
	Graphical User Interface
	Jika masukan *username* atau *email* yang diberikan oleh *guest* telah terdaftar, maka aplikasi akan menampilkan pesan *error* "*Your username or email has been registered, choose another*" kepada *guest*.
Pascakondisi	*Guest* berhasil melakukan pendaftaran ke dalam sistem.

Tabel 4.3 menunjukkan skenario yang dapat dilakukan oleh *guest* dalam melakukan pendaftaran ke dalam sistem. *Guest* dapat mendaftar dengan memberikan masukan *username, email,* dan *password* kepada aplikasi. Kebutuhan fungsional *sign up* dirancang hanya untuk digunakan melalui sistem antarmuka grafis (*graphical user interface system*).

Di bawah ini terdapat tabel yang menunjukkan skenario *use case sign in* dari aplikasi *messaging* berbasis *voice interactions*.

Tabel 4.4 Skenario *Use Case Sign In*

Kode *Use Case*	FR02
Nama *Use Case*	*Sign In*
Tujuan *Use Case*	Mengizinkan *guest* untuk masuk sebagai *member* ke dalam sistem.
Aktor	*Guest*
Prakondisi	1. *Guest* telah membuka aplikasi dan terhubung ke internet. 2. *Guest* telah terdaftar ke dalam sistem.
Alur Utama	*Voice User Interface*
	1. *Guest* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *guest.* 3. *Guest* memberikan perintah suara "*sign in*" kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*what is your username?*" kepada *guest*. 5. *Guest* memberikan masukan suara berupa *username* yang ingin digunakan kepada aplikasi. 6. Aplikasi memberikan umpan balik melalui pesan suara "*what is your password?*" kepada *guest*. 7. *Guest* memberikan masukan suara berupa *password* yang ingin digunakan kepada aplikasi. 8. Aplikasi meminta konfirmasi atas *username* dan *password* yang diberikan *guest* melalui pesan suara "*your username is {username} and your password is {password}. are you sure?*". 9. *Guest* memberikan konfirmasi atas *username* dan *password* yang diberikan sebelumnya melalui masukan suara "*yes*" kepada aplikasi. 10. Aplikasi memberikan umpan balik melalui pesan suara "*sign in successfully*" kepada *guest*.

	Graphical User Interface
	1. Aplikasi menampilkan halaman *sign in.* 2. *Guest* mengisi *form sign in.* 3. *Guest* meng-*click* tombol *sign in.* 4. Aplikasi mengubah *sign in state* untuk *guest*. 5. Aplikasi memberikan
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali perintah suara yang diberikan oleh *guest*, maka aplikasi akan memberikan umpan balik melalui pesan suara "*unknown voice command given*" kepada *guest.* 2. Jika kombinasi masukan *username* dan *password* yang diberikan oleh *guest* tidak tepat, maka aplikasi akan memberikan umpan balik melalui pesan suara "*your username or password isn't correct!*" kepada *guest*. 3. Jika aplikasi gagal mengenali masukan *username* dan *password* yang diberikan oleh *guest,* maka aplikasi akan kembali ke kondisi awal, yakni kondisi yang siap menerima *voice command* baru. 4. Jika aplikasi tidak mengenali konfirmasi atas *username* dan *password* yang diberikan oleh *guest*, maka aplikasi akan membatalkan proses *sign in* dan kemudian memberikan umpan balik melalui pesan suara "*confirmation rejected. sign in process canceled*" kepada *guest*.
	Graphical User Interface
	Jika kombinasi masukan *username* dan *password* yang diberikan oleh *guest* tidak tepat, maka aplikasi akan menampilkan pesan *error* "*Your username or password isn't correct!*" kepada *guest*.
Pascakondisi	*Guest* berhasil masuk sebagai *member* ke dalam sistem.

Tabel 4.4 menunjukkan skenario yang dapat dilakukan oleh *guest* untuk masuk sebagai *member* ke dalam sistem. *Guest* dapat masuk ke dalam sistem dengan menggunakan *username* dan *password* yang telah didaftarkan sebelumnya. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command* "*sign in*" yang diberikan kepada aplikasi.

Skenario *use case help* merupakan skenario yang dapat dilakukan oleh *guest* atau *member* untuk melihat atau mendengarkan panduan penggunaan aplikasi. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command* "*help*" yang diberikan kepada aplikasi. Skenario *use case help* dapat dilihat melalui Tabel 4.5.

Tabel 4.5 Skenario *Use Case Help*

Kode *Use Case*	FR03
Nama *Use Case*	*Help*
Tujuan *Use Case*	Mengizinkan *guest* atau *member* untuk melihat atau mendengarkan panduan penggunaan aplikasi.
Aktor	*Guest, Member*
Prakondisi	*Guest* atau *member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Guest* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *guest.* 3. *Guest* atau *member* memberikan perintah suara "*help*" kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*User manual content is ready to be read. Are you sure want to continue"?* kepada *guest* atau *member.* 5. *Guest* atau *member* memberikan konfirmasi atas pembacaan panduan penggunaan aplikasi melalui masukan suara "*yes*" kepada aplikasi. 6. Aplikasi membacakan panduan penggunaan aplikasi kepada *guest* atau *member.* 7. Setelah satu konten panduan penggunaan aplikasi selesai dibacakan, maka aplikasi akan meminta konfirmasi *guest* atau *member* untuk melanjutkan pembacaan melalui pesan suara "*do you want to continue?*" 8. *Guest* atau *member* memberikan konfirmasi atas pelanjutan pembacaan panduan penggunaan

	aplikasi melalui masukan suara "*yes*" kepada aplikasi. 9. Aplikasi melanjutkan proses membacakan panduan penggunaan aplikasi.
	Graphical User Interface
	1. *User* atau *member* meng-*click* tombol *options menu*. 2. Aplikasi menampilkan *option menu items*. 3. *User* atau *member* meng-*click* tombol *help.* 4. Aplikasi menampilkan halaman *help*.
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali perintah suara yang diberikan oleh *guest* atau *member*, maka aplikasi akan memberikan umpan balik melalui pesan suara "*unknown voice command given*" kepada *guest* atau *member*. 2. Jika *guest* atau *member* memberikan konfirmasi atas pelanjutan pembacaan panduan penggunaan aplikasi melalui masukan suara "*repeat"* kepada aplikasi, maka aplikasi akan mengulangi membacakan panduan penggunaan aplikasi yang terakhir kali dibacakan. 3. Jika aplikasi tidak mengenali konfirmasi atas pembacaan ataupun pelanjutan pembacaan panduan penggunaan aplikasi yang diberikan oleh *guest* atau *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. reading user manual cancelled*" kepada *guest* atau *member*. 4. Jika seluruh panduan penggunaan aplikasi telah selesai dibacakan oleh aplikasi, maka aplikasi akan memberikan umpan balik melalui pesan suara "*all of user manual content already read. reading user manual content is finished."* kepada *guest* atau *member.*
	Graphical User Interface
	-

Pascakondisi	*Guest* atau *member* berhasil melihat atau mendengarkan panduan penggunaan aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case add contact* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.6 Skenario *Use Case Add Contact*

Kode *Use Case*	FR04
Nama *Use Case*	*Add Contact*
Tujuan *Use Case*	Mengizinkan *member* untuk menambahkan *member* lain ke dalam daftar kontaknya.
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.* 3. *Member* memberikan perintah suara "*add contact"* kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*what the username that you want to add to your contact list?"* kepada *member.* 5. *Member* memberikan masukan suara kepada aplikasi berupa *username* dari *member* yang ingin ditambahkan ke dalam daftar kontak. 6. Aplikasi meminta konfirmasi atas proses penambahan kontak melalui pesan suara "*are you sure to add {username} to your contact list?*" . 7. *Member* memberikan konfirmasi atas proses penambahan kontak melalui masukan suara "*yes*" kepada aplikasi. 8. Aplikasi menambahkan kontak dengan *username* yang sesuai dengan masukan sebelumnya ke dalam daftar kontak yang dimiliki oleh *member.*

	9. Aplikasi memberikan umpan balik melalui pesan suara "*{username} successfully added to your contact list*" kepada *member*.
	Graphical User Interface
	1. *Member* membuka *tabview contacts.* 2. Aplikasi menampilkan halaman *tabview contacts.* 3. *Member* mengisi *form add contact.* 4. *Member* meng-*click* tombol *add contact*. 5. Aplikasi menampilkan pesan "*{username} successfully added to your contact list".*
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali perintah suara *member*, maka aplikasi akan memberikan umpan balik melalui pesan suara "*unknown voice command given*" kepada *member*. 2. Jika aplikasi tidak mengenali konfirmasi atas proses penambahan kontak yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. add contact process cancelled*" kepada *member*. 3. Jika *username* yang diberikan *member* telah terdaftar pada daftar kontaknya, maka aplikasi akan memberikan umpan balik melalui pesan suara "*contact already exist*" kepada *member*. 4. Jika *username* yang diberikan *member* tidak *valid*, maka aplikasi akan memberikan umpan balik melaui pesan suara "*contact isn't valid*" kepada *member*.
	Graphical User Interface
	1. Jika *username* yang diisikan oleh *member* pada *form add contact* telah terdaftar pada daftar kontaknya*,* maka aplikasi akan menampilkan pesan *error "contact already exist!"* kepada *member.* 2. Jika *username* yang diisikan oleh *member* pada *form add contact* tidak *valid* maka aplikasi akan menampilkan pesan *error* "*Contact isn't valid!"*

	kepada *member.*
Pascakondisi	*Member* berhasil menambahkan kontak baru ke dalam daftar kontaknya.

Tabel 4.6 menunjukkan skenario yang dapat dilakukan oleh *member* untuk menambahkan *member* lain berdasarkan *username*-nya ke dalam daftar kontak yang dimiliki oleh *member* tersebut. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command "add contact"* yang diberikan kepada aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case delete contact* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.7 Skenario *Use Case Delete Contact*

Kode *Use Case*	FR05
Nama *Use Case*	*Delete Contact*
Tujuan *Use Case*	Mengizinkan *member* untuk menghapus salah satu kontak yang terdapat pada daftar kontaknya.
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.* 3. *Member* memberikan perintah suara *"delete contact"* kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*what the username that you want to delete from your contact list?"* kepada *member.* 5. *Member* memberikan masukan suara kepada aplikasi berupa *username* dari *member* yang ingin dihapus dari dalam daftar kontak. 6. Aplikasi meminta konfirmasi atas proses penghapusan kontak melalui pesan suara "*are you sure want to delete {username} from your contact list?".* 7. *Member* memberikan konfirmasi atas proses

	penghapusan kontak melalui masukan suara "*yes*" kepada aplikasi. 8. Aplikasi menghapus kontak dengan *username* yang sesuai dengan masukan sebelumnya dari dalam daftar kontak *member.* 9. Aplikasi memberikan umpan balik melalui pesan suara *"contact deleted successfully"* kepada *member.*
	Graphical User Interface
	1. *Member* membuka *tabview contacts.* 2. Aplikasi menampilkan halaman *tabview contacts.* 3. *Member* melakukan *longclick* pada salah satu *contact*. 4. Aplikasi menampilkan *dialog* untuk meminta konfirmasi *member* atas penghapusan *contact.* 5. *Member* meng-*click* tombol *Yes.* 6. Aplikasi menampilkan pesan *"contact deleted successfully".*
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali masukan suara yang diberikan oleh *member*, maka aplikasi akan memberikan umpan balik melaui pesan suara "*unknown voice command given*" kepada *member.* 2. Jika aplikasi tidak mengenali konfirmasi atas proses penghapusan kontak yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. delete contact process cancelled*" kepada *member*. 3. Jika *username* yang diberikan *member* tidak *valid*, maka aplikasi akan memberikan umpan balik melaui pesan suara "*Contact isn't valid*!" kepada *member*.
	Graphical User Interface
	Jika *member* meng-*click tombol No* dalam proses pemberian konfirmasi atas penghapusan *contact,* maka aplikasi akan membatalkan proses penghapusan kontak.

Pascakondisi	*Member* berhasil menghapus salah satu kontak dari daftar kontaknya.

Tabel 4.7 menunjukkan skenario yang dapat dilakukan oleh *member* untuk menghapus salah satu kontak berdasarkan *username* yang tedapat pada daftar kontaknya. Pada *voice user interface,* skenario ini dipicu melalui *voice command "delete contact"* yang diberikan kepada aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case send message* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.8 Skenario *Use Case Send Message*

Kode *Use Case*	FR06
Nama *Use Case*	*Send Message*
Tujuan *Use Case*	Mengizinkan *member* untuk mengirimkan pesan kepada *member* lain yang terdapat pada daftar kontaknya.
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi, terhubung ke internet dan menentukan sistem *user interface* yang diinginkan.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up*" kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.* 3. *Member* memberikan perintah suara "*send message*" kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*what the receiver username of your message?*" kepada *member*. 5. *Member* memberikan masukan suara kepada aplikasi berupa *username* dari *member* yang menjadi penerima pesan. 6. Aplikasi memberikan umpan balik melalui pesan suara "*what the message?*" kepada *member.* 7. *Member* memberikan masukan suara kepada aplikasi berupa pesan yang akan dikirimkan. 8. Aplikasi meminta konfirmasi atas proses pengiriman pesan melalui pesan suara "*are you*

	sure want to send message to {username} with the message {message}?" . 9. *Member* memberikan konfirmasi atas proses pengiriman pesan melalui masukan suara "*yes*" kepada aplikasi. 10. Aplikasi mengirimkan pesan sesuai dengan *username* penerima pesan dan isi pesan yang telah diberikan sebelumnya. 11. Aplikasi memberikan umpan balik melalui pesan suara "*message sent successfully"* kepada *member.*
	Graphical User Interface
	1. *Member* membuka halaman *tabview contacts* atau halaman *tabview conversations.* 2. Aplikasi menampilkan halaman *tabview contacts* atau halaman *tabview conversations.* 3. *Member* meng-*click* salah satu *contact* yang terdapat pada halaman *tabview contacts* atau salah satu conversation yang terdapat pada halaman *tabview conversations.* 4. Aplikasi menampilkan halaman utama *conversation.* 5. *Member* mengisi *form* pesan. 6. *Member* meng-*click* tombol *send*. 7. Aplikasi menampilkan pesan "*message delivered successfully*".
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali perintah suara yang diberikan oleh *member*, maka aplikasi akan memberikan umpan balik melaui pesan suara "*unknown voice command given*" kepada *member.* 2. Jika aplikasi tidak mengenali konfirmasi atas proses pengiriman pesan yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. send message process cancelled*" kepada *member*. 3. Jika *username* yang diberikan *member* tidak valid,

	maka aplikasi akan memberikan umpan balik melaui pesan suara "*receiver username is not valid. send message process cancelled*" kepada *member*.
	Graphical User Interface
	-
Pascakondisi	*Member* berhasil mengirimkan pesan kepada *member* lain.

Tabel 4.8 menunjukkan skenario yang dapat dilakukan oleh *member* untuk mengirimkan pesan kepada *member* lain yang terdapat pada daftar kontaknya. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command "send message"* yang diberikan kepada aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case read unread messages* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.9 Skenario *Use Case Read Unread Messages*

Kode *Use Case*	FR07
Nama *Use Case*	*Read Unread Messages*
Tujuan *Use Case*	Mengizinkan *member* untuk melihat atau mendengarkan pesan masuk yang belum pernah dilihat atau didengarkan sebelumnya.
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.* 3. *Member* memberikan perintah suara "*check new message*" kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*Unread messages is ready to be read. Are you sure want to continue?*" kepada *member.* 5. *Member* memberikan konfirmasi atas proses

	pembacaan pesan masuk yang belum pernah dibaca sebelumnya melalui pesan suara "*yes*" kepada aplikasi. 6. Aplikasi membacakan pesan masuk yang belum pernah dibaca sebelumnya kepada *member*. 7. Setelah satu pesan selesai dibacakan, maka aplikasi akan meminta konfirmasi *member* untuk melanjutkan pembacaan melalui pesan suara "*do you want continue?".* 8. *Member* memberikan konfirmasi atas pelanjutan pembacaan pesan melalui masukan suara "*yes*" kepada aplikasi. 9. Aplikasi melanjutkan proses membacakan pesan masuk yang belum pernah dibaca sebelumnya.
	Graphical User Interface
	1. *Member* membuka halaman *tabview conversations.* 2. Aplikasi menampilkan halaman *tabview conversations.* 3. *Member* meng-*click* conversation yang memiliki tanda notifikasi. 4. Aplikasi menampilkan halaman utama *conversation.*
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali perintah suara yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melaui pesan suara "*unknown voice command given*" kepada *member.* 2. Jika *member* memberikan konfirmasi atas pelanjutan pembacaan pesan yang belum pernah dibaca sebelumnya melalui masukan suara "*repeat"* kepada aplikasi, maka aplikasi akan mengulangi membacakan pesan yang terakhir kali dibacakan. 3. Jika aplikasi tidak mengenali konfirmasi atas pembacaan ataupun pelanjutan pembacaan pesan yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara

	"*confirmation rejected. read unread message cancelled*" kepada *member*. 4. Jika tidak terdapat pesan yang belum pernah dilihat atau didengarkan, maka aplikasi akan memberikan umpan balik melalui pesan suara "*you do not have any unread message"* kepada *member.* 5. Jika seluruh pesan yang belum pernah dilihat atau didengarkan telah selesai dibacakan, maka aplikasi akan memberikan umpan balik melalui pesan suara "*all message already read. read unread message process is finished.*" kepada *member.*
	Graphical User Interface
	-
Pascakondisi	*Member* berhasil mendengarkan pesan yang belum pernah dilihat atau didengarkan sebelumnya.

Tabel 4.9 menunjukkan skenario yang dapat dilakukan oleh *member* untuk melihat atau mendengarkan pesan yang belum pernah dilihat atau didengarkan sebelumnya. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command "read unread message"* yang diberikan kepada aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case read conversation* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.10 Skenario *Use Case Read Conversation*

Kode *Use Case*	FR08
Nama *Use Case*	*Read Conversation*
Tujuan *Use Case*	Mengizinkan *member* untuk melihat atau mendengarkan percakapannya dengan *member* lain.
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.*

	3. *Member* memberikan perintah suara "*read conversation*" kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*what is the interlocutors username?*" kepada *member.* 5. *Member* memberikan masukan suara kepada aplikasi berupa *username* dari *member* yang menjadi lawan percakapannya. 6. Aplikasi memberikan umpan balik melalui pesan suara "*conversation is ready to be read. do you want to continue?*" kepada *member.* 7. *Member* memberikan konfirmasi atas proses pembacaan percakapan melalui pesan suara "*yes*" kepada aplikasi. 8. Aplikasi membacakan pesan yang terdapat pada *conversation member* dengan *member* lain yang sesuai dengan *username* yang telah diberikan sebelumnya. 9. Setelah satu pesan selesai dibacakan, maka aplikasi akan meminta konfirmasi *member* untuk melanjutkan pembacaan melalui pesan suara "*do you want to continue?".* 10. *Member* memberikan konfirmasi atas pelanjutan pembacaan pesan melalui masukan suara "*yes*" kepada aplikasi. 11. Aplikasi melanjutkan proses membacakan percakapan.
	Graphical User Interface
	1. *Member* membuka halaman *tabview conversations.* 2. Aplikasi menampilkan halaman *tabview conversations.* 3. *Member* meng-*click* salah satu *conversation.* 4. Aplikasi menampilkan halaman utama *conversation.*
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali perintah suara yang

	diberikan oleh *member*, maka aplikasi akan memberikan umpan balik melalui pesan suara "*unknown voice command given*" kepada *member.* 2. Jika *member* memberikan konfirmasi atas pelanjutan pembacaan percakapan melalui masukan suara "*repeat"* kepada aplikasi, maka aplikasi akan mengulangi membacakan pesan yang terakhir kali dibacakan. 3. Jika aplikasi tidak mengenali konfirmasi atas pembacaan ataupun pelanjutan pembacaan pesan yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. reading conversation process cancelled*" kepada *member*. 4. Jika tidak terdapat percakapan antara *member* dengan *member* lain yang sesuai dengan masukan *username* sebelumnya, maka aplikasi akan memberikan umpan balik melalui pesan suara "*conversation is not exist. reading conversation process cancelled "* kepada *member.* 5. Jika *username* yang diberikan *member* tidak valid, maka aplikasi akan memberikan umpan balik melaui pesan suara "*interlocutors username is not valid. reading conversation process cancelled*" kepada *member*. 6. Jika seluruh pesan yang terdapat di dalam *conversation* telah selesai dibacakan, maka aplikasi akan memberikan umpan balik melalui pesan suara "*all message already read. reading conversation process is finished"* kepada *member*.
	Graphical User Interface
	-
Pascakondisi	*Member* berhasil melihat atau mendengarkan percakapannya dengan *member* lain.

Tabel 4.10 menunjukkan skenario yang dapat dilakukan oleh *member* untuk melihat atau mendengarkan percakapannya dengan *member* lain. Pada sistem *voice user interface, s*kenario ini dipicu melalui *voice command "read conversation"* yang diberikan kepada aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case delete conversation* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.11 Skenario *Use Case Delete Conversation*

Kode *Use Case*	FR09
Nama *Use Case*	*Delete Conversation*
Tujuan *Use Case*	Mengizinkan *member* untuk menghapus percakapannya dengan *member* lain.
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.* 3. *Member* memberikan perintah suara "*delete conversation*" kepada aplikasi. 4. Aplikasi memberikan umpan balik melalui pesan suara "*what is the interlocutors username?*" kepada *member.* 5. *Member* memberikan masukan suara kepada aplikasi berupa *username* dari *member* yang ingin dihapus dari dalam daftar percakapan. 6. Aplikasi meminta konfirmasi atas proses penghapusan percakapan melalui pesan suara "*are you sure want to delete your conversation with {username}?*". 7. *Member* memberikan konfirmasi atas proses penghapusan percakapan melalui masukan suara "*yes*" kepada aplikasi. 8. Aplikasi menghapus percakapan *member* dengan *member* lain yang sesuai dengan *username* yang diberikan sebelumnya. 9. Aplikasi memberikan umpan balik melalui pesan suara "*Conversation deleted successfully"* kepada

	member.
	Graphical User Interface
	1. *Member* membuka halaman *tabview conversations.* 2. Aplikasi menampilkan halaman *tabview conversations.* 3. *Member* melakukan *longclick* pada salah satu *conversation.* 4. Aplikasi menampilkan *dialog* untuk meminta konfirmasi *member* atas penghapusan *conversation.* 5. *Member* memilih tombol *Yes.* 6. Aplikasi menampilkan pesan "*conversation deleted successfully*".
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali masukan suara yang diberikan oleh *member*, maka aplikasi akan memberikan umpan balik melaui pesan suara "*unknown voice command given*" kepada *member.* 2. Jika aplikasi tidak mengenali konfirmasi atas proses penghapusan percakapan yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. delete conversation process cancelled*" kepada *member*. 3. Jika tidak terdapat percakapan *member* dengan *member* lain yang sesuai dengan *username* yang diberikan, maka aplikasi akan memberikan umpan balik melalui pesan suara "*conversation isn't exist. delete conversation process cancelled"* kepada *member.*
	Graphical User Interface
	Jika *member* meng-*click tombol No* dalam proses pemberian konfirmasi atas penghapusan percakapan, maka aplikasi akan membatalkan proses penghapusan percakapan.

Pascakondisi	*Member* berhasil menghapus percakapan.

Tabel 4.11 menunjukkan skenario yang dapat dilakukan oleh *member* untuk menghapus percakapannya dengan *member* lain. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command "delete conversation"* yang diberikan kepada aplikasi.

Di bawah ini terdapat tabel yang menunjukkan skenario *use case sign out* dari aplikasi *messaging* berbasis *voice interactions.*

Tabel 4.12 Skenario *Use Case Sign Out*

Kode *Use Case*	FR010
Nama *Use Case*	*Sign Out*
Tujuan *Use Case*	Mengizinkan *member* untuk keluar dari sistem
Aktor	*Member*
Prakondisi	*Member* telah membuka aplikasi dan terhubung ke internet.
Alur Utama	*Voice User Interface*
	1. *Member* memberikan masukan suara "*wake up"* kepada aplikasi. 2. Aplikasi memberikan umpan balik melalui pesan suara "*I'm ready to receive you command now*" kepada *member.* 3. *Member* memberikan perintah suara "*sign out"* kepada aplikasi. 4. Aplikasi memberikan umpan balik untuk meminta konfirmasi melaui pesan suara "*Are you sure want to sign out?"* kepada *member.* 5. *Member* memberikan konfirmasi atas proses *sign out* melalui masukan suara "*yes"* kepada aplikasi. 6. Aplikasi mengubah *sign-in state* milik *member*. 7. Aplikasi memberikan umpan balik melalui pesan suara "*sign out successfully"* kepada *member.*
	Graphical User Interface
	1. *Member* meng-*click* tombol *options menu.* 2. Aplikasi menampilkan *options menu items.* 3. *Member* meng-*click* tombol *sign out.*

	4. Aplikasi menampilkan *dialog* untuk meminta konfirmasi *member* atas proses *sign out.* 5. *Member* meng-*click* tombol *Yes.* 6. Aplikasi mengubah *sign-in state* milik *member*. 7. Aplikasi menampilkan pesan "*sign out successfully*" kepada *member.*
Alur Alternatif	*Voice User Interface*
	1. Jika aplikasi tidak mengenali masukan suara yang diberikan oleh *member*, maka aplikasi akan memberikan umpan balik melaui pesan suara "*unknown voice command given*" kepada *member.* 2. Jika aplikasi tidak mengenali konfirmasi atas proses *sign out* yang diberikan oleh *member,* maka aplikasi akan memberikan umpan balik melalui pesan suara "*confirmation rejected. sign out process cancelled*" kepada *member*.
	Graphical User Interface
	1. Jika *member* meng-*click tombol No* dalam proses pemberian konfirmasi atas proses *sign out,* maka aplikasi akan membatalkan proses *sign out.*
Pascakondisi	*Member* berhasil keluar dari sistem.

Tabel 4.12 menunjukkan skenario yang dapat dilakukan oleh *member* untuk keluar dari sistem. Pada sistem *voice user interface,* skenario ini dipicu melalui *voice command "sign out"* yang diberikan kepada aplikasi.

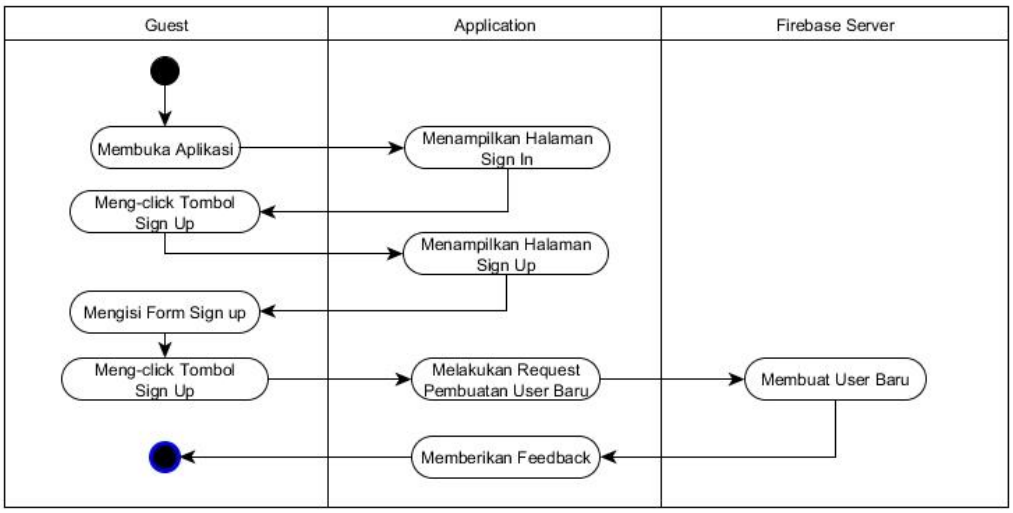
4.1.4 *Activity Diagram*

Activity Diagram merupakan diagram yang digunakan untuk memodelkan perangkat lunak menurut aktifitas-aktifitas yang terjadi pada sistem ataupun yang melibatkan sistem. Diagram ini dibuat dengan mengacu pada diagram *Use Case* yang telah dibuat sebelumnya. Berdasarkan hal ini, aktifitas-aktifitas yang terdapat pada masing-masing *activity diagram* dapat dikelompokkan ke dalam 3 bagian berdasarkan pelakunya, yaitu *guest* atau *member, application,* dan *firebase server.*

4.1.4.1 *Activity Diagram Sign Up*

Diagram ini memodelkan proses pendaftaran yang melibatkan *guest, application,* dan *firebase server*. Proses ini membutuhkan masukan data dari *guest* untuk digunakan sebagai data akun *member* yang akan dibuat. Data tersebut dapat diberikan melalui interaksi *guest* dengan sistem antarmuka grafis

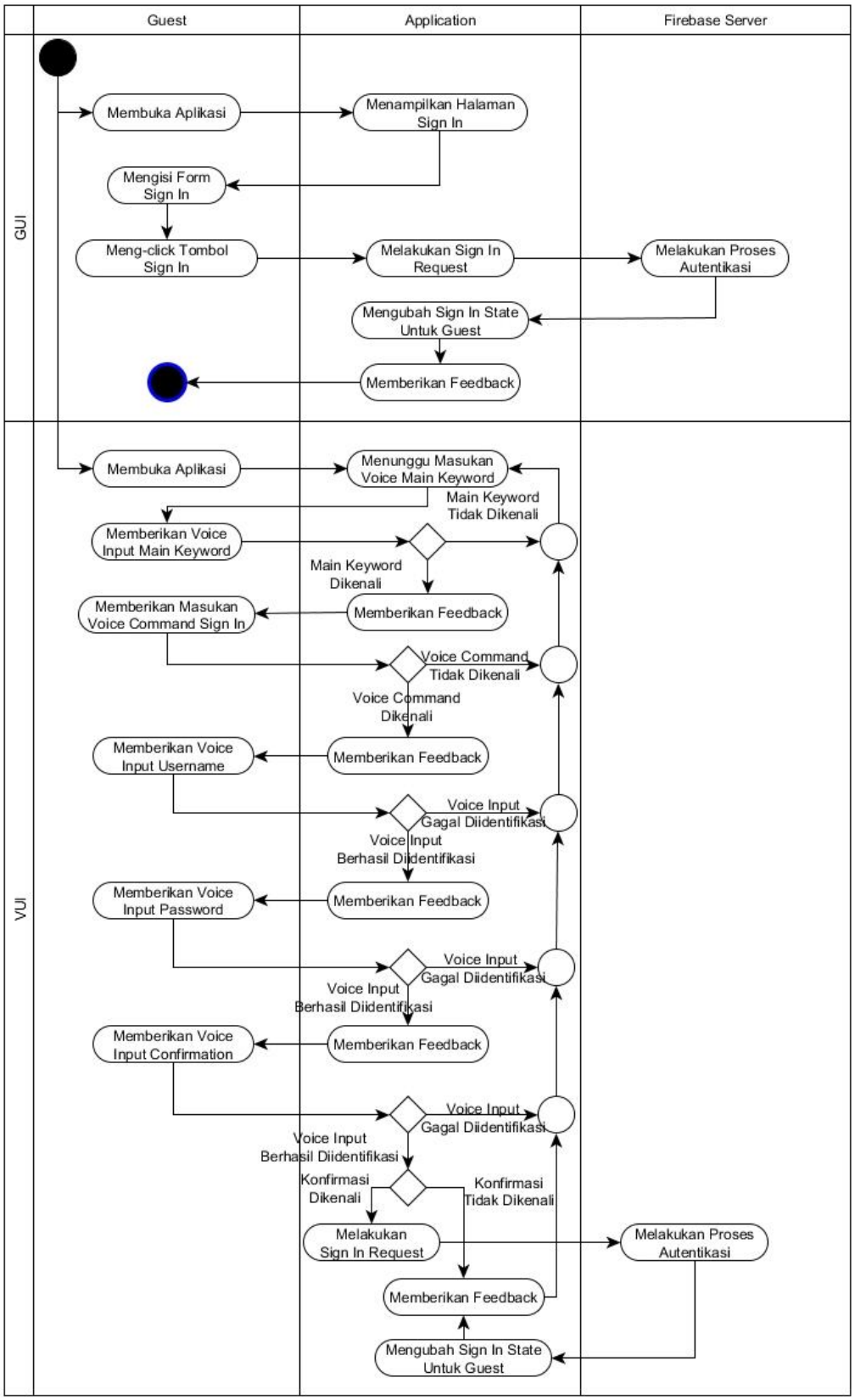
(*graphical user interface system*). *Activity diagram sign up* dapat dilihat melalui Gambar 4.3.



Gambar 4.3 *Activity Diagram Sign Up*

4.1.4.2 *Activity Diagram Sign In*

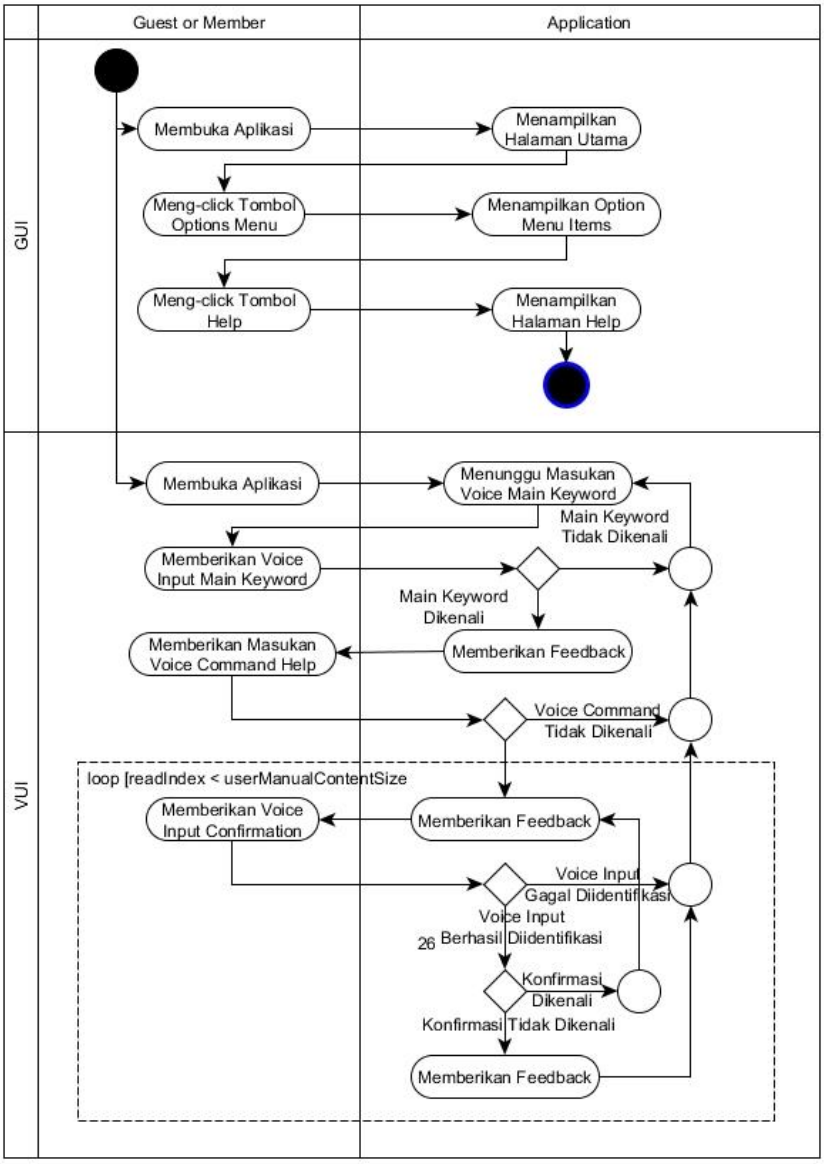
Diagram ini memodelkan proses *sign in* yang melibatkan *guest*, *application*, dan *firebase server*. Pada sistem *voice user interface,* aktifitas *sign in* ini dimulai melalui *voice command sign in* yang diberikan oleh *guest* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* melalui pesan suara kepada *guest* dalam bentuk *conversation* untuk meminta data yang dibutuhkan dalam proses *sign in*. Setelah *guest* memberikan data tersebut melalui masukan suara, aplikasi akan mengirimkan *sign in request* kepada *firebase server* berdasarkan data masukan *guest* sebelumnya. Kemudian, *firebase server* akan mengirimkan respon atas *sign in request* tersebut kepada aplikasi. Respon tersebut nantinya digunakan untuk mengubah *sign in state* untuk *guest.* A*ctivity diagram sign in* dapat dilihat melalui Gambar 4.4.



Gambar 4.4 *Activity Diagram Sign In*

4.1.4.3 *Activity Diagram Help*

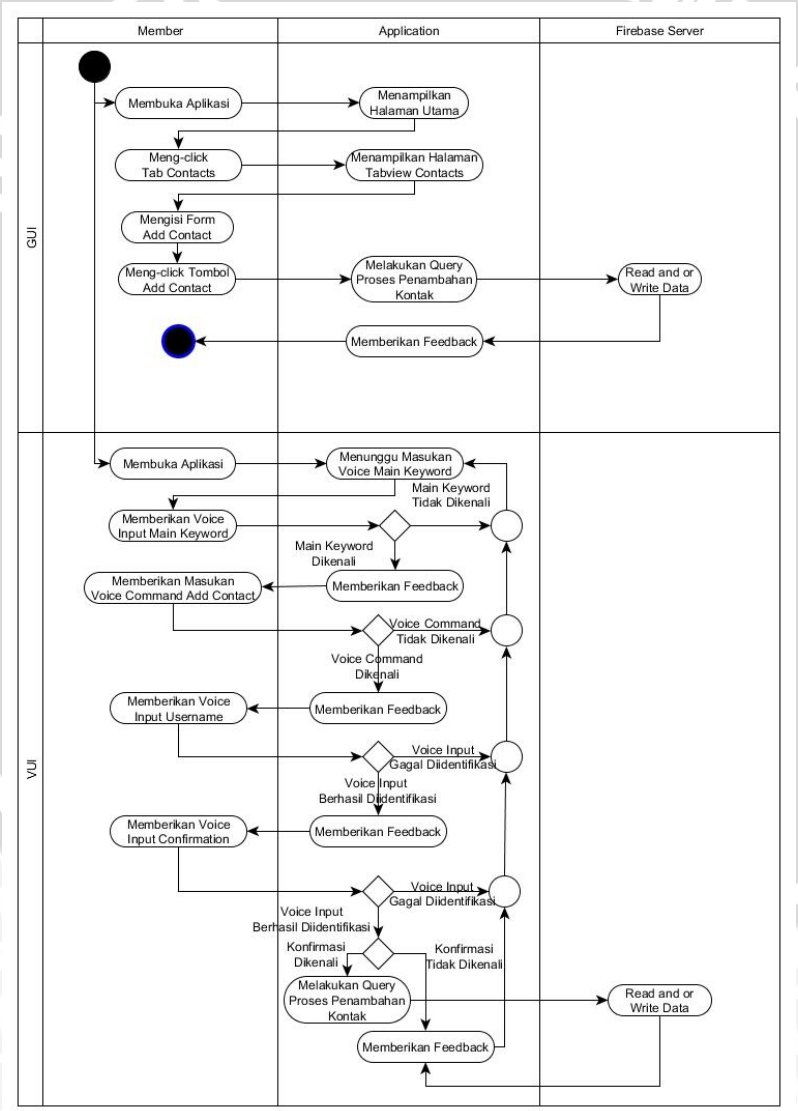
Gambar 4.5 menunjukkan *activity diagram help.* Diagram ini memodelkan proses untuk melihat atau mendengarkan panduan penggunaan aplikasi yang melibatkan *user* atau *member.* Pada sistem *voice user interface,* aktifitas mendengarkan panduan penggunaan aplikasi dimulai melalui *voice command help* yang diberikan oleh *guest* atau *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *guest* atau *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation. Feedback* tersebut bertujuan untuk membacakan *user manual contents* kepada *guest* atau *member.*



Gambar 4.5 *Activity Diagram Help*

4.1.4.4 *Activity Diagram Add Contact*

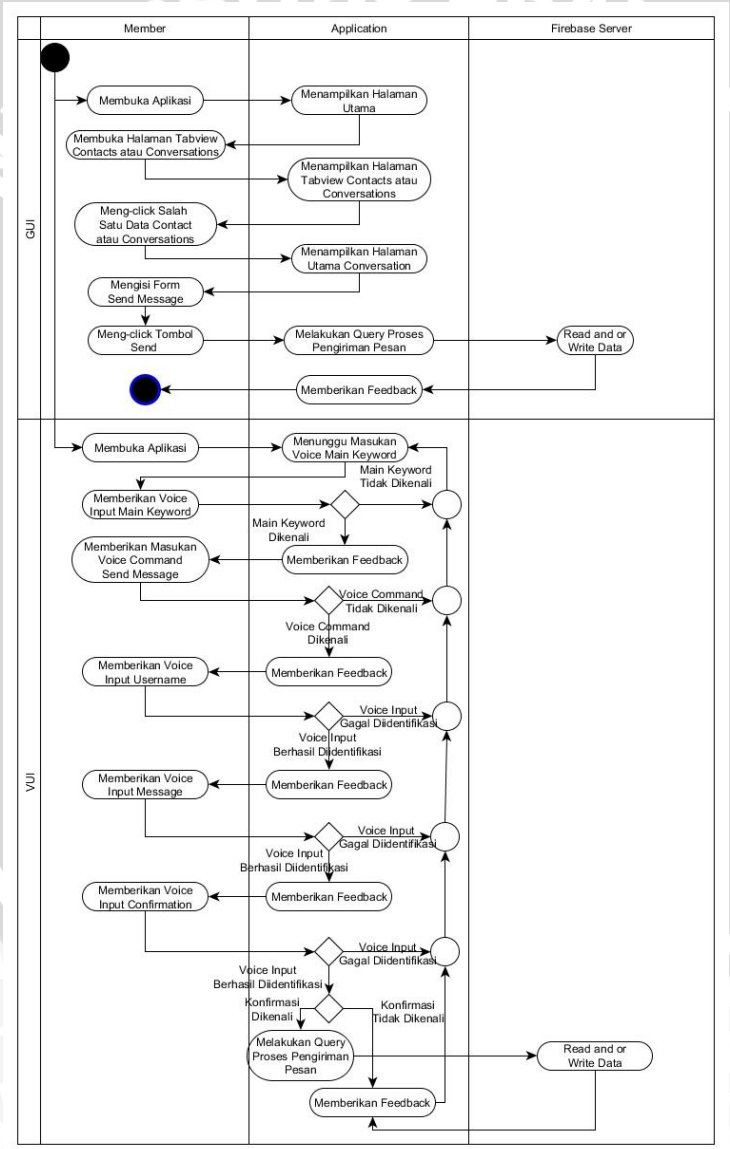
Gambar 4.6 menunjukkan *activity diagram add contact*. Diagram ini memodelkan proses penambahan kontak baru yang melibatkan *member*, *application*, dan *firebase server*. Pada sistem *voice user interface,* aktifitas penambahan kontak baru dimulai melalui *voice command add contact* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation. Feedback* tersebut bertujuan untuk meminta data yang dibutuhkan dalam proses penambahan kontak. Setelah data tersebut diberikan oleh *member*, maka aplikasi akan melakukan *query* proses penambahan kontak. Kemudian aplikasi akan memberikan *feedback* melalui pesan suara kepada *member* sesuai dengan hasil *query* yang telah dilakukan sebelumnya.



Gambar 4.6 A*ctivity Diagram Add Contact*

4.1.4.5 *Activity Diagram Send Message*

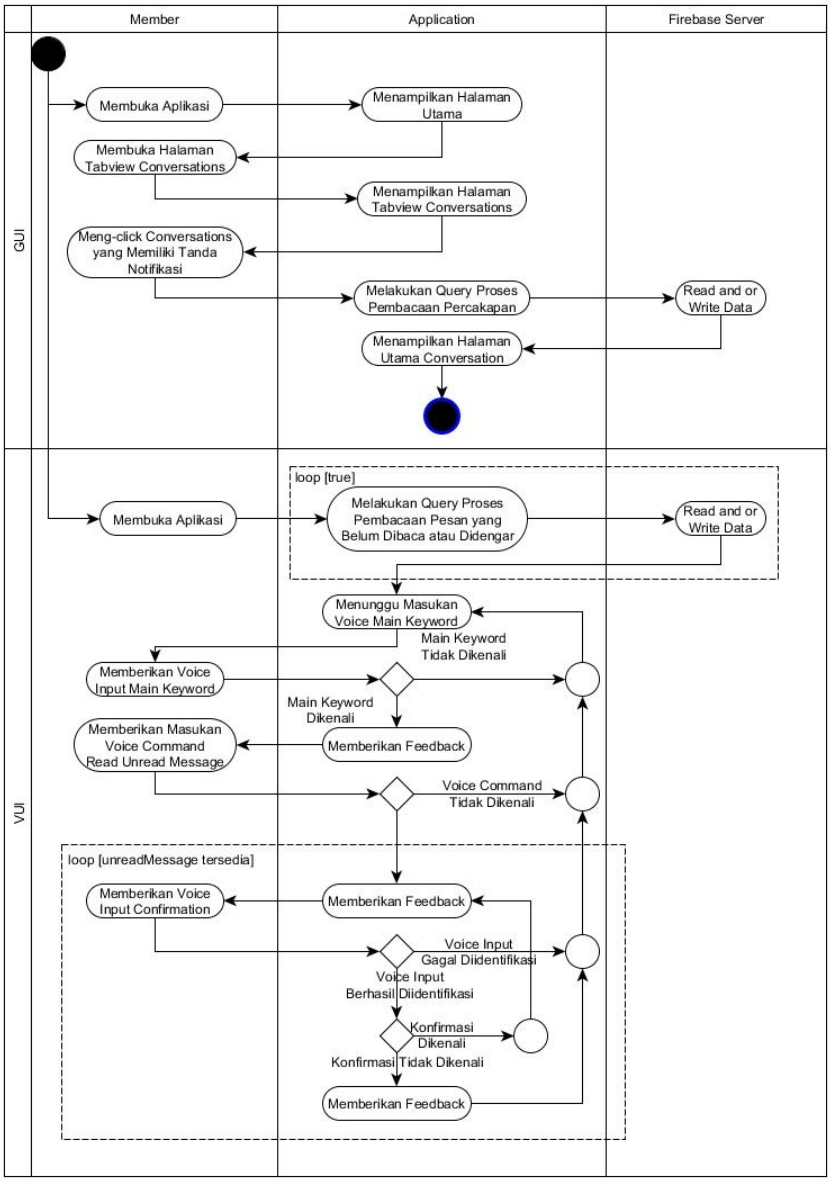
Gambar 4.7 menunjukkan *activity diagram send message*. Diagram ini memodelkan aktifitas pengiriman pesan yang melibatkan *member*, *application* dan *firebase server*. Pada sistem *voice user interface,* aktifitas pengiriman pesan dimulai melalui *voice command send message* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation*. *Feedback* tersebut bertujuan untuk meminta data yang dibutuhkan dalam proses pengiriman pesan. Setelah data tersebut diberikan oleh *member*, maka aplikasi akan melakukan *query* proses pengiriman pesan. Kemudian aplikasi akan memberikan *feedback* melalui pesan suara kepada *member* sesuai dengan hasil *query* yang telah dilakukan sebelumnya.



Gambar 4.7 *Activity Diagram Send Message*

4.1.4.6 *Activity Diagram Read New Unread Message*

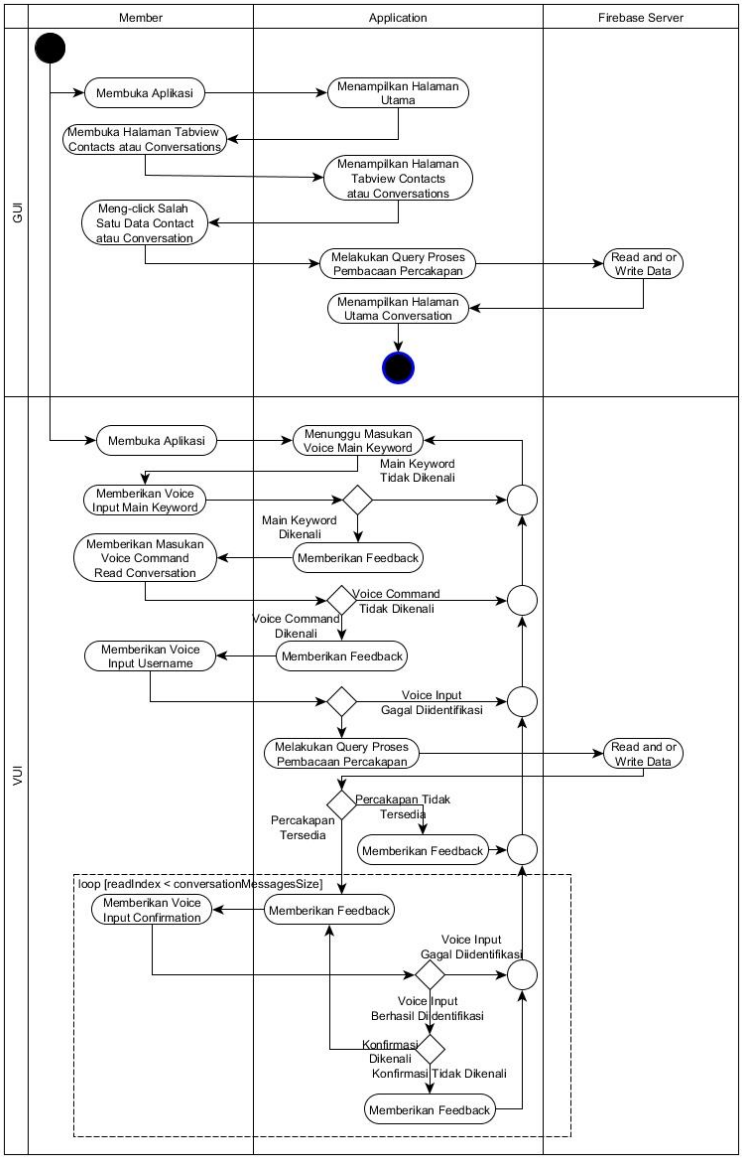
Gambar 4.8 menunjukkan *activity diagram read unread message*. Diagram ini memodelkan aktifitas melihat atau mendengarkan pesan masuk yang belum pernah dilihat atau didengarkan sebelumnya. Aktifitas ini melibatkan *member, application* dan *firebase server.* Pada sistem *voice user interface,* ini dimulai melalui *voice command read unread message* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation*. *Feedback* tersebut bertujuan untuk membacakan pesan-pesan masuk baru yang belum pernah didengarkan oleh *member* sebelumnya.



Gambar 4.8 *Activity Diagram Read New Unread Message*

4.1.4.7 *Activity Diagram Read Conversation*

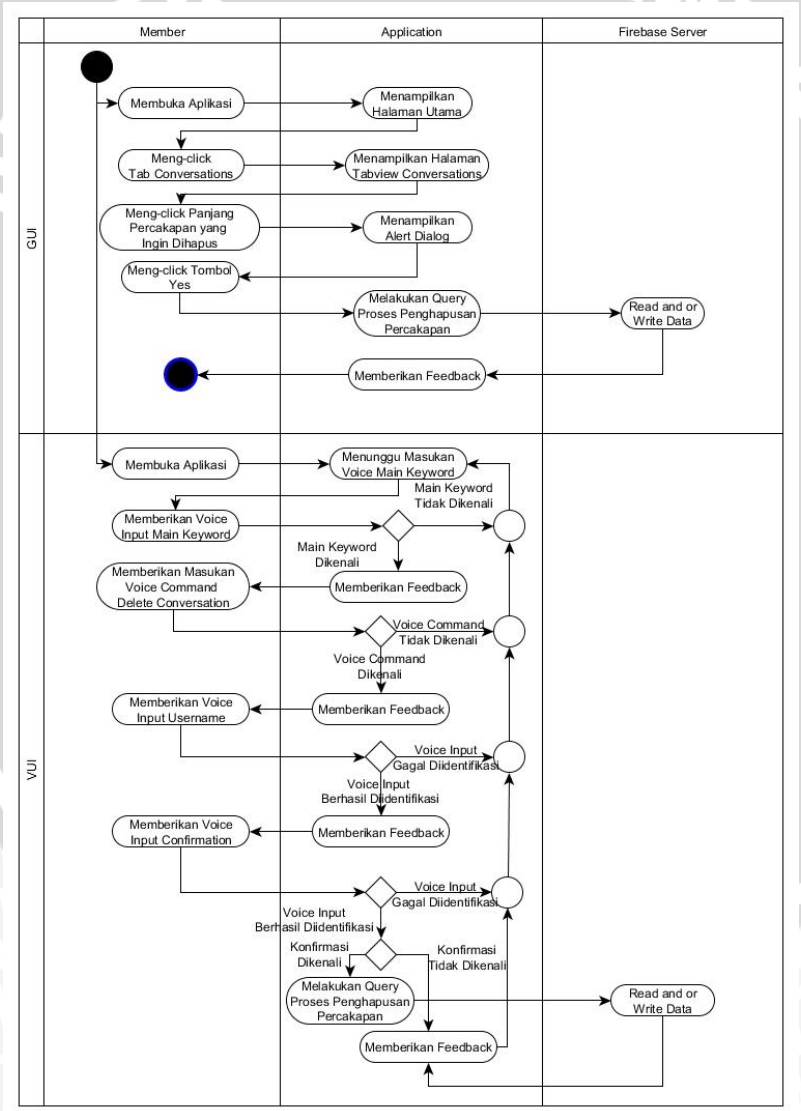
Gambar 4.9 menunjukkan *activity diagram read conversation*. Diagram ini memodelkan aktifitas pembacaan pesan-pesan dalam sebuah *conversation* yang melibatkan *member* dan *application*. Pada sistem *voice user interface,* aktifitas pembacaan percakapan dimulai melalui *voice command read conversation* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation*. *Feedback* tersebut bertujuan untuk meminta data yang dibutuhkan dalam proses pembacaan percakapan. Setelah data tersebut diberikan, maka aplikasi akan melakukan query proses pembacaan percakapan dan proses pembacaan pesan-pesan yang terdapat dalam sebuah percakapan berdasarkan data yang diberikan sebelumnya.



Gambar 4.9 *Activity Diagram Read Conversation*

4.1.4.8 *Activity Diagram Delete Conversation*

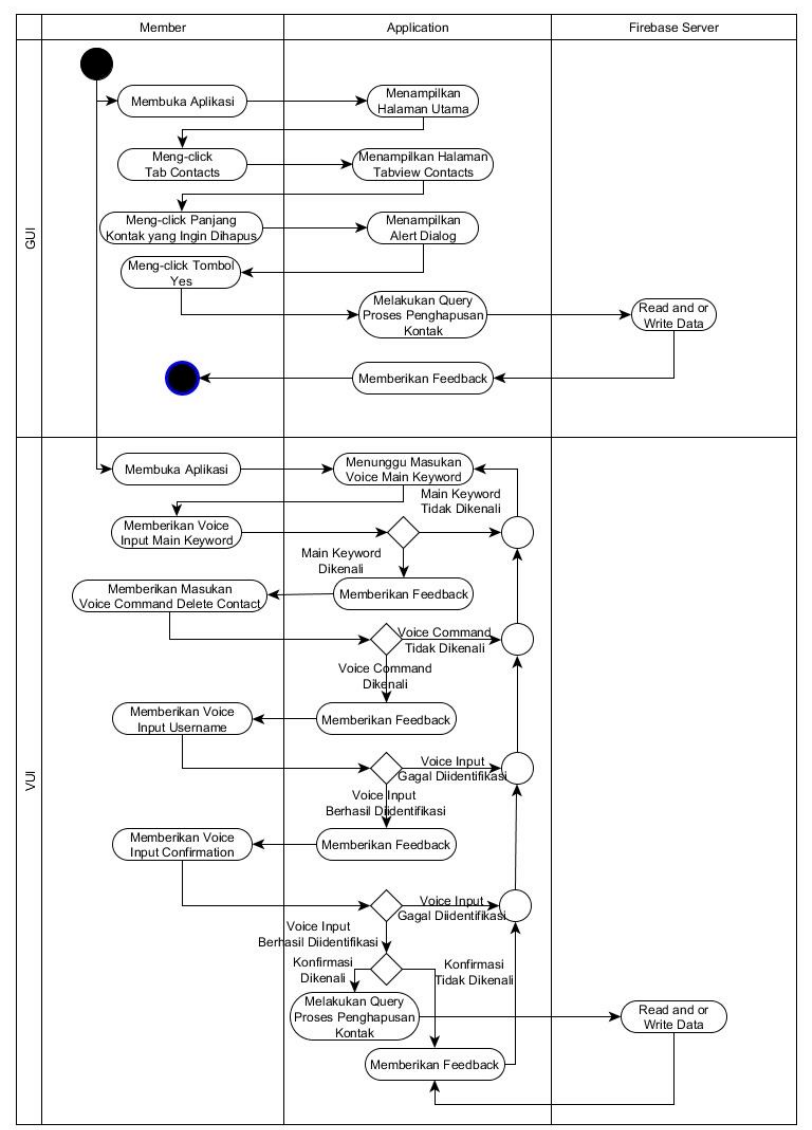
Gambar 4.10 menunjukkan *activity diagram delete conversation*. Diagram ini memodelkan aktifitas penghapusan pecakapan yang melibatkan *member*, *application*, dan *firebase server*. Pada sistem *voice user interface,* aktifitas penghapusan percakapan dimulai melalui *voice command delete conversation* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation*. *Feedback* tersebut bertujuan untuk meminta data yang dibutuhkan dalam proses penghapusan percakapan. Setelah data tersebut diberikan oleh *member*, maka aplikasi akan melakukan *query* proses penghapusan percakapan. Kemudian aplikasi akan memberikan *feedback* melalui pesan suara kepada *member* sesuai dengan hasil *query* yang telah dilakukan sebelumnya.



Gambar 4.10 *Activity Diagram Delete Conversation*

4.1.4.9 *Activity Diagram Delete Contact*

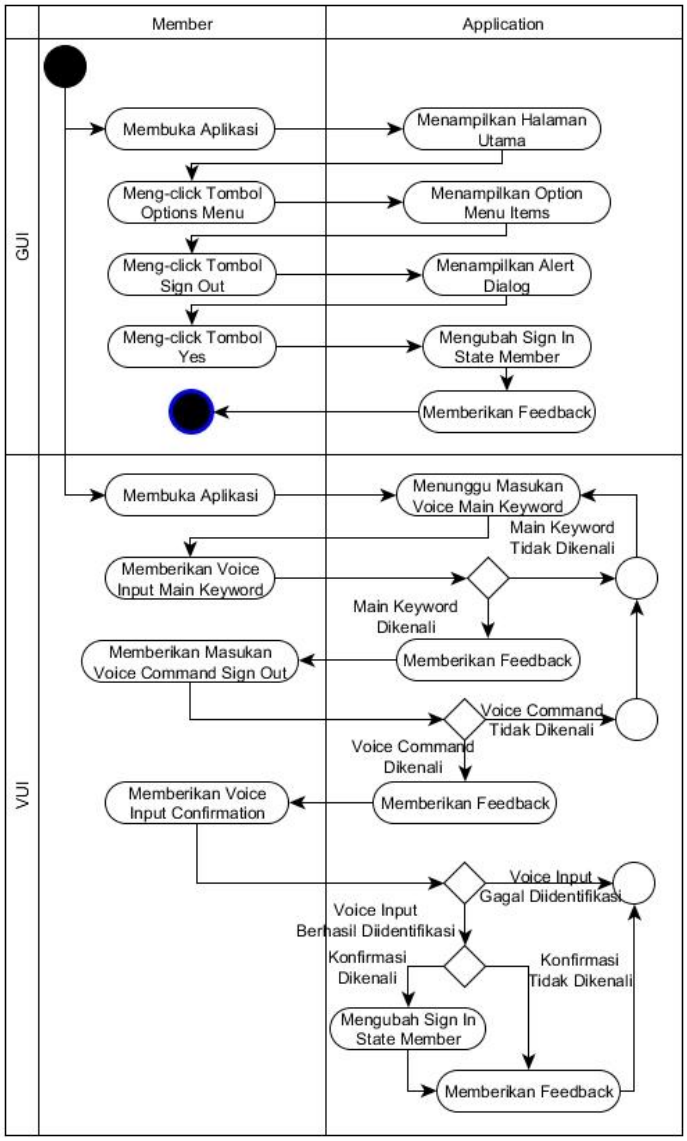
Gambar 4.11 menunjukkan *activity diagram delete contact*. Diagram ini memodelkan aktifitas penghapusan kontak yang melibatkan *member*, *application*, dan *firebase server*. Pada sistem *voice user interface,* aktifitas penghapusan kontak dimulai melalui *voice command delete contact* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation*. *Feedback* tersebut bertujuan untuk meminta data yang dibutuhkan dalam proses penghapusan kontak. Setelah data tersebut diberikan oleh *member*, maka aplikasi akan melakukan *query* proses penghapusan kontak. Kemudian aplikasi akan memberikan *feedback* melalui pesan suara kepada *member* sesuai dengan hasil *query* yang telah dilakukan sebelumnya.



Gambar 4.11 *Activity Diagram Delete Contact*

4.1.4.10 *Activity Diagram Sign Out*

Gambar 4.12 menunjukkan *activity diagram sign out*. Diagram ini memodelkan proses *sign out* yang melibatkan *member* dan *application*. Pada sistem *voice user interface,* aktifitas *sign out* ini dimulai melalui *voice command sign out* yang diberikan oleh *member* kepada aplikasi. Aplikasi akan memberikan respon atau *feedback* kepada *member* atas *voice command* yang diberikan sebelumnya melalui pesan suara dalam bentuk *conversation*. *Feedback* tersebut bertujuan untuk meminta data yang dibutuhkan dalam proses *sign out*. Setelah data tersebut diberikan oleh *member*, maka aplikasi akan *sign in state* milik *member* yang diikuti proses pemberian *feedback* melalui pesan suara kepada *member*.



Gambar 4.12 *Activity Diagram Sign Out*

4.2 Perancangan Perangkat Lunak

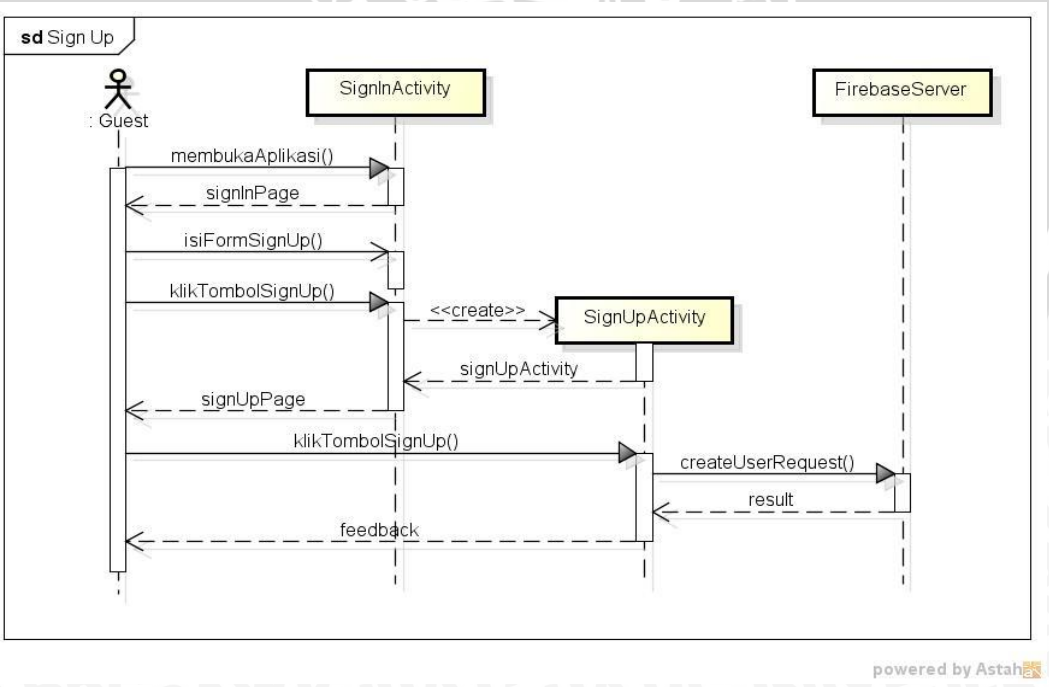
Perancangan perangkat lunak merupakan sebuah proses yang bertujuan untuk memodelkan perangkat lunak berdasarkan hasil analisis kebutuhan yang telah diperoleh sebelumnya. Pada tahap ini, dilakukan 5 perancangan untuk memodelkan perangkat lunak yang akan dibangun. Tahap-tahap tersebut adalah perancangan *sequence diagram*, perancangan *class diagram,* perancangan basis data, perancangan antarmuka dan perancangan *page flow.*

4.2.1 Perancangan *Sequence Diagram*

Sequence Diagram merupakan salah satu *UML Diagram* yang digunakan untuk memodelkan perangkat lunak berdasarkan sudut pandang interaksi objek-objek yang terlibat dalam suatu aktifitasnya. Aktifitas yang dimaksud di sini merupakan aktifitas yang berdasarkan hasil analisis kebutuhan sistem yang telah dilakukan sebelumnya. Interaksi antar objek atau elemen yang terlibat dalam sebuah aktifitas terjadi secara bertahap dan tahapan tersebut dapat dilihat melalui notasi yang bernama *lifeline.* Interaksi tersebut ditandai melalui adanya pertukaran pesan yang dapat dilihat melalui notasi *message.*

4.2.1.1 *Sequence Diagram Sign Up*

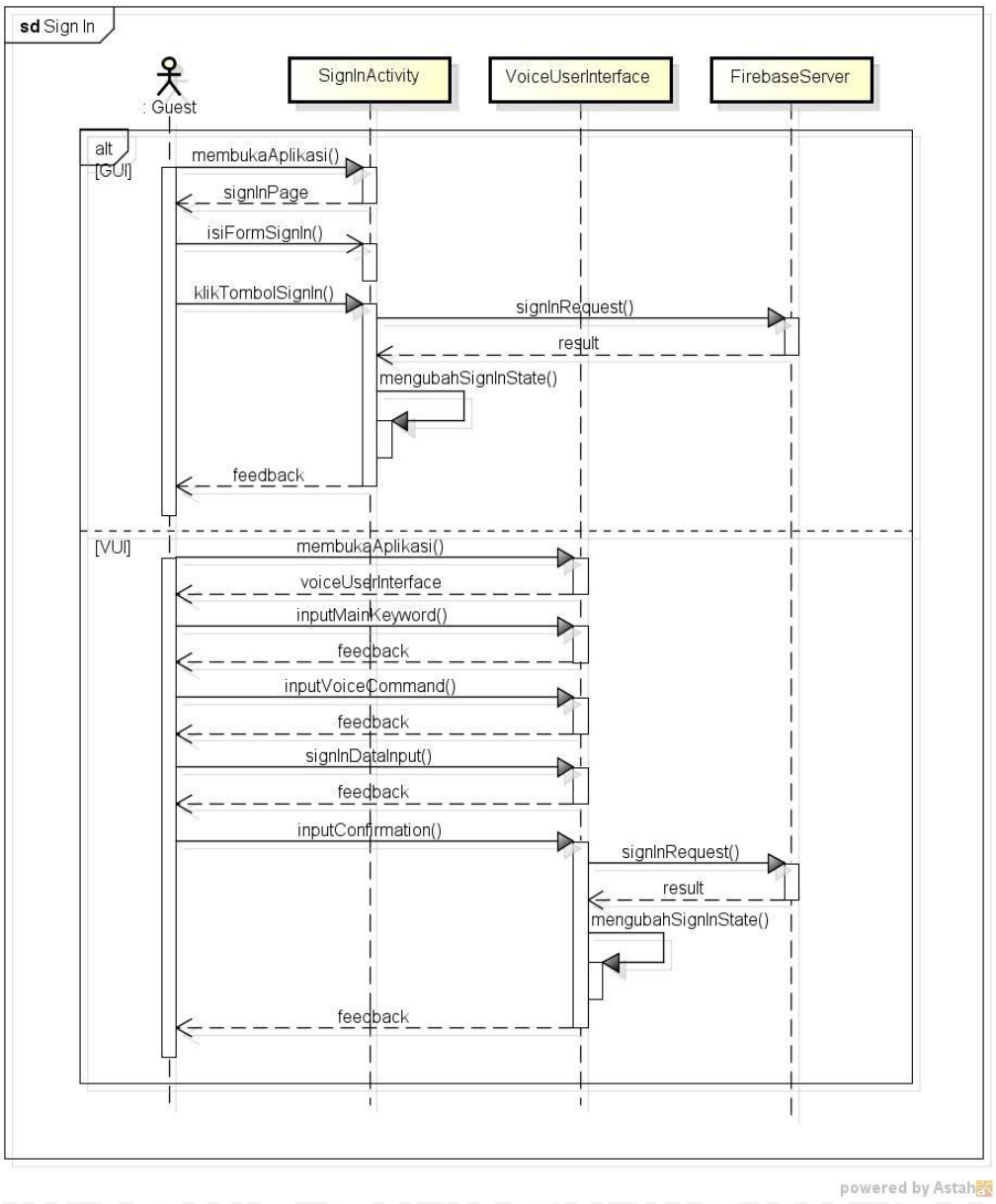
Gambar 4.13 menunjukkan *sequence diagram sign up.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas pendaftaran *(sign up)*. Aktifitas ini akan melibatkan *Guest* sebagai aktornya. Pada aktifitas ini, objek *Guest* akan berinteraksi dengan objek *SignInActivity, SignUpActivity* dan *FirebaseServer.*



Gambar 4.13 *Sequence Diagram Sign Up*

4.2.1.2 *Sequence Diagram Sign In*

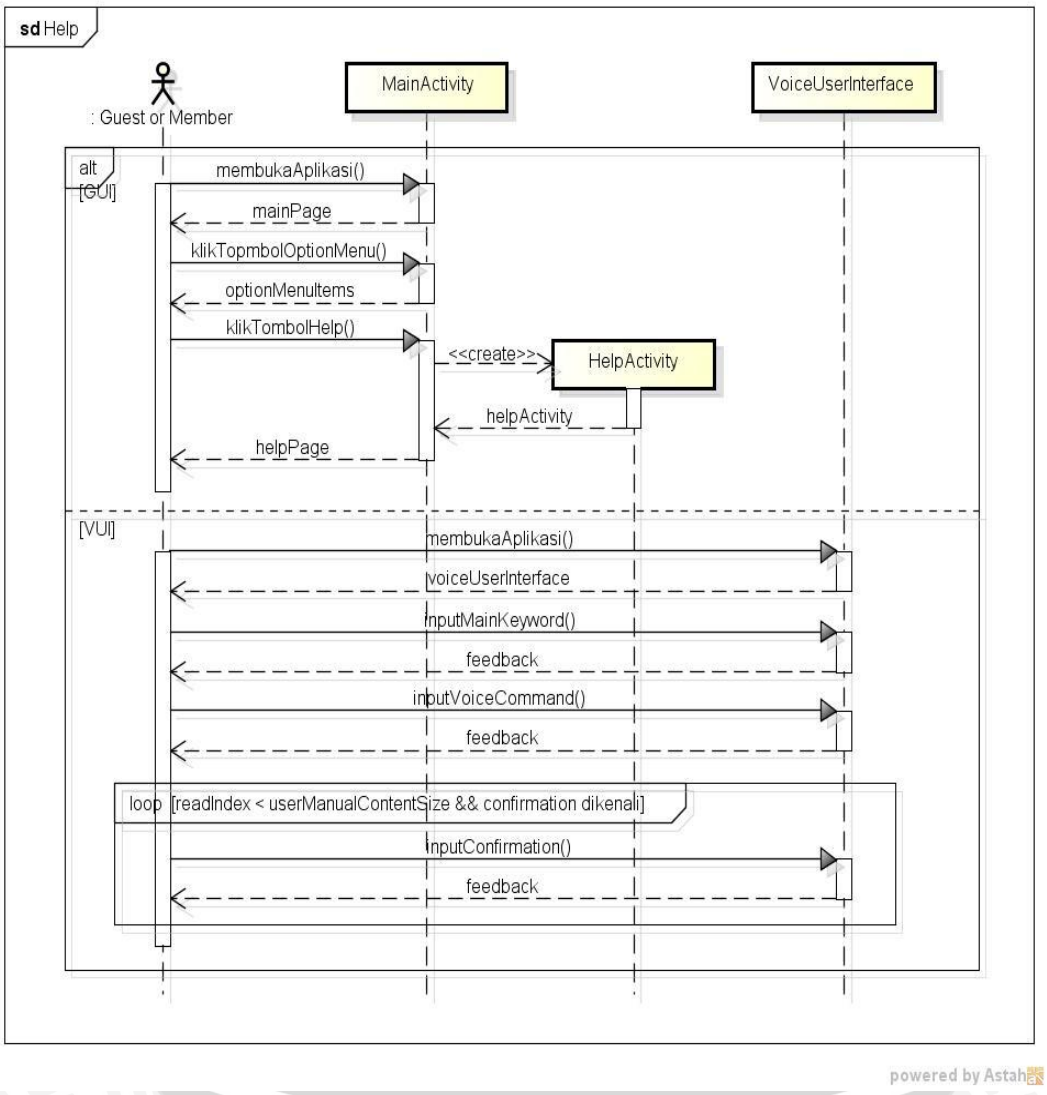
Gambar 4.14 menunjukan *sequence diagram sign in.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas *sign in.* Aktifitas ini akan melibatkan objek *Guest* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Guest* akan berinteraksi dengan objek *VoiceUserInterface* dan *FirebaseServer.* Sedangkan, pada penggunaan sistem *graphical user interface,* objek *Guest* akan berinteraksi dengan objek *SignInActivity* dan *FirebaseServer.*



Gambar 4.14 *Sequence Diagram Sign In*

4.2.1.3 *Sequence Diagram Help*

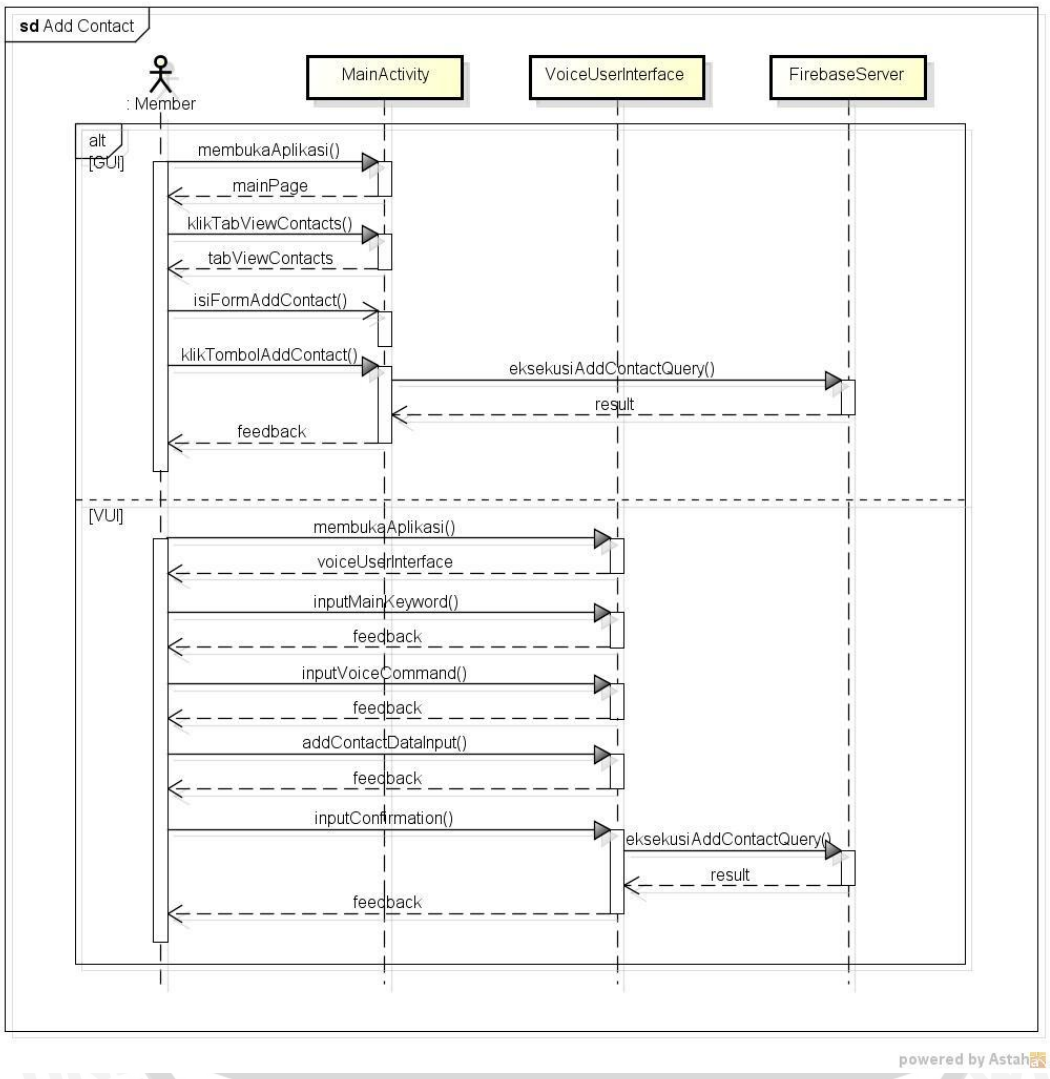
Gambar 4.15 menunjukan *sequence diagram help.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas melihat atau mendengarkan panduan penggunaan aplikasi (*help*). Aktifitas ini akan melibatkan objek *Guest* atau *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Guest* atau *Member* akan berinteraksi dengan objek *VoiceUserInterface.* Sedangkan pada penggunaan sistem *graphical user interface,* objek *Guest* atau *Member* akan berinteraksi dengan objek *MainActivity dan HelpActivity.*



Gambar 4.15 *Sequence Diagram Help*

4.2.1.4 *Sequence Diagram Add Contact*

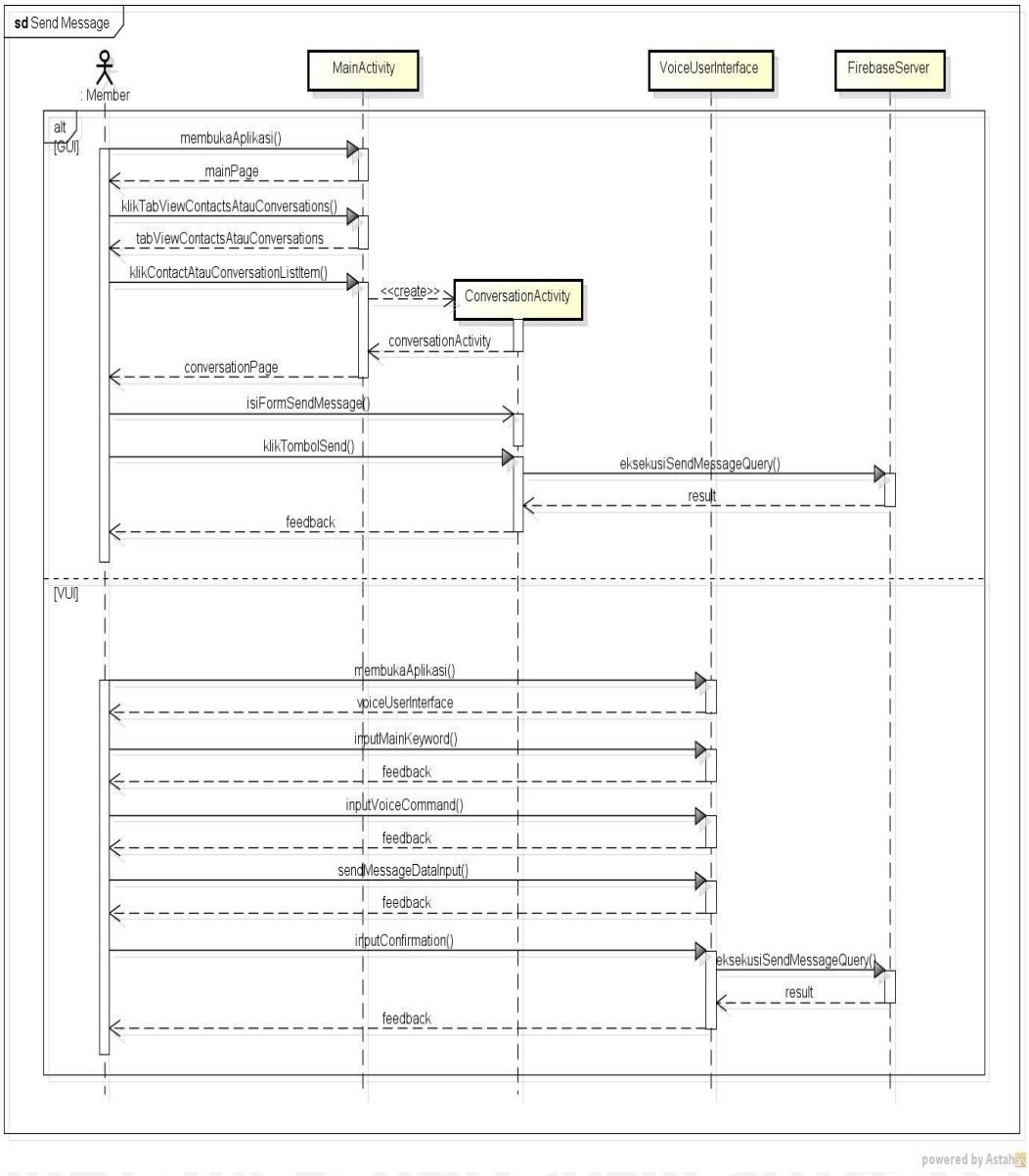
Gambar 4.16 menunjukan *sequence diagram add contact.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas penambahan kontak (*add contact).* Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterface* dan *FirebaseServer.* Sedangkan pada penggunaan sistem *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity* dan *FirebaseServer.*



Gambar 4.16 *Sequence Diagram Add Contact*

4.2.1.5 *Sequence Diagram Send Message*

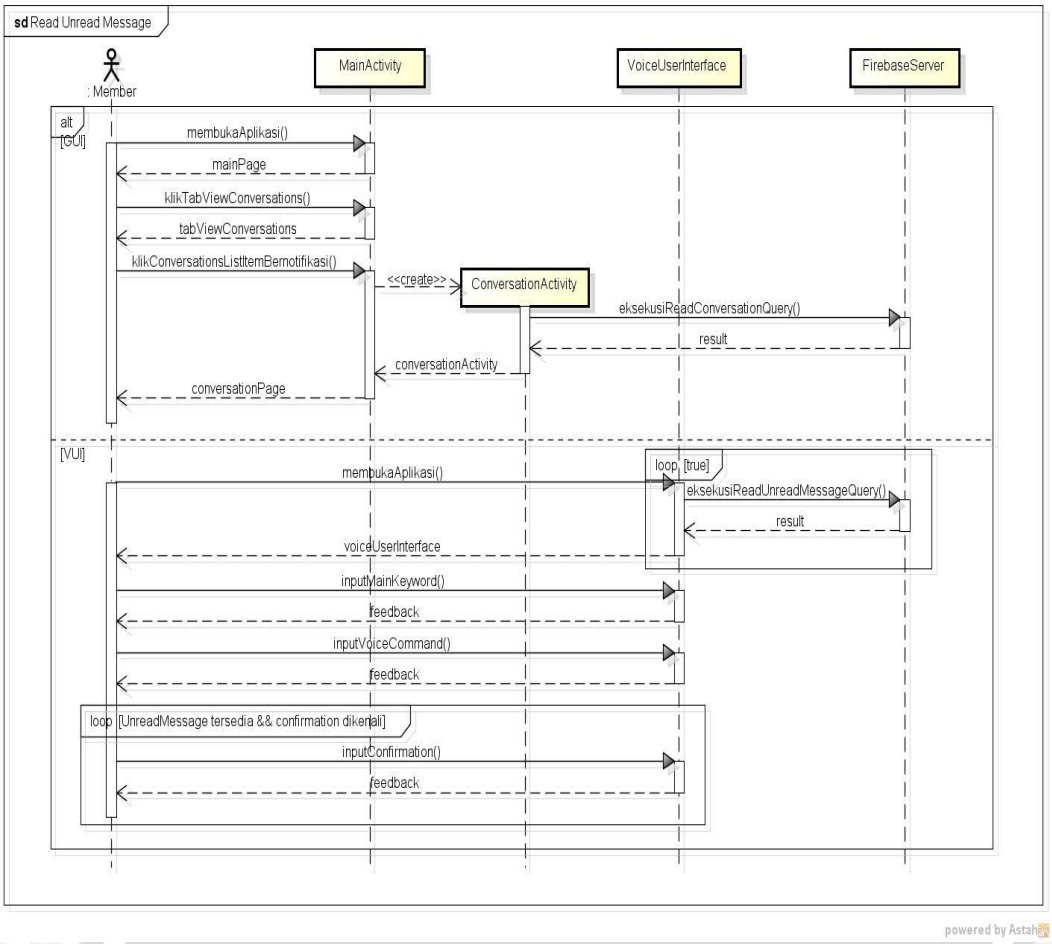
Gambar 4.17 menunjukan *sequence diagram send message.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas pengiriman pesan (*send message).* Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterface* dan *FirebaseServer.* Sedangkan pada penggunaan *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity, ConversationActivity,* dan *FirebaseServer.*



Gambar 4.17 *Sequence Diagram Send Message*

4.2.1.6 *Sequence Diagram Read Unread Message*

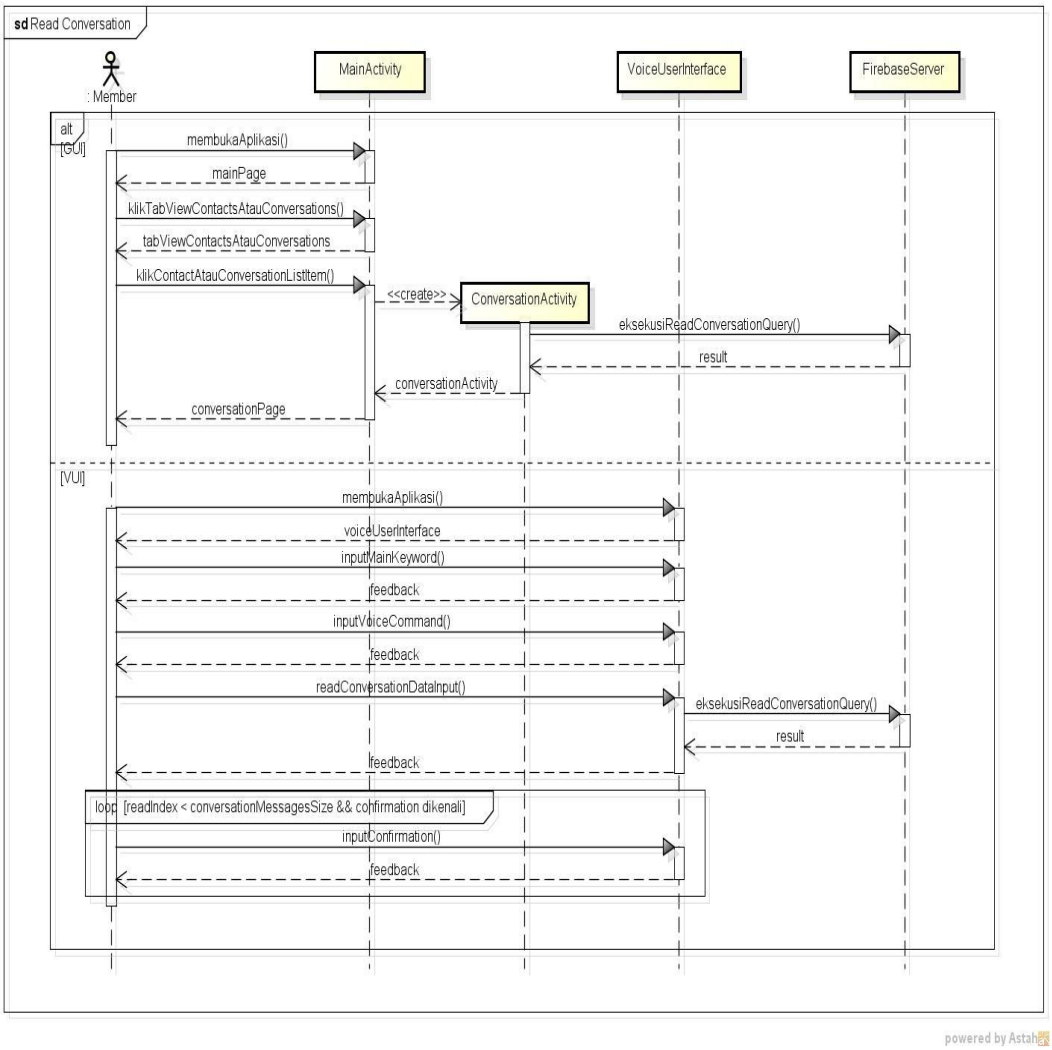
Gambar 4.18 menunjukan *sequence diagram read unread message.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas melihat atau mendengarkan pesan yang belum pernah dilihat atau didengarkan sebelumnya. Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterface* dan *FirebaseServer.* Sedangkan pada penggunaan sistem *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity, ConversationActivity* dan *FirebaseServer.*



Gambar 4.18 *Sequence Diagram Read Unread Message*

4.2.1.7 *Sequence Diagram Read Conversation*

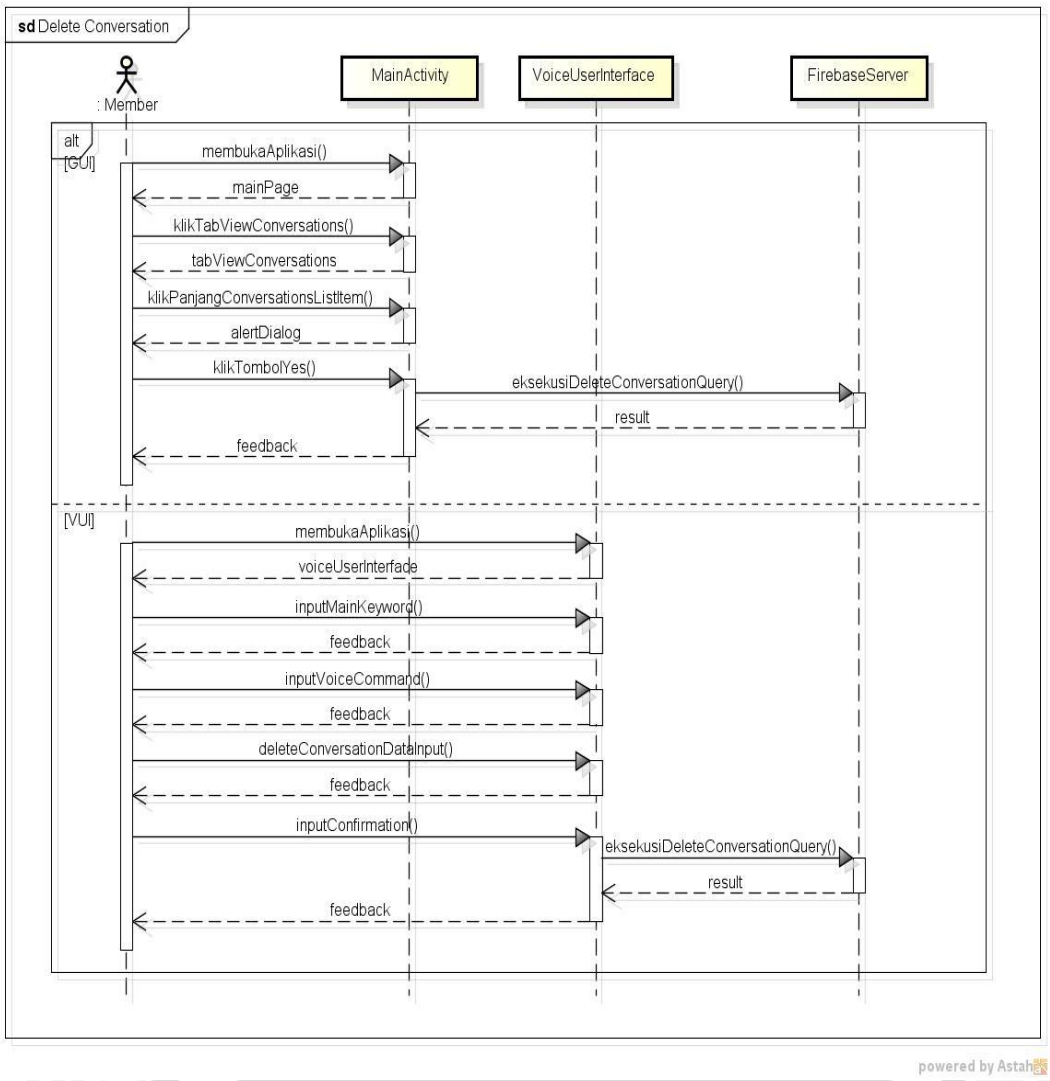
Gambar 4.19 menunjukan *sequence diagram read conversation.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas membaca atau mendengarkan percakapan. Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterface* dan *FirebaseServer.* Sedangkan pada penggunaan sistem *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity, ConversationActivity* dan *FirebaseServer.*



Gambar 4.19 *Sequence Diagram Read Conversation*

4.2.1.8 *Sequence Diagram Delete Conversation*

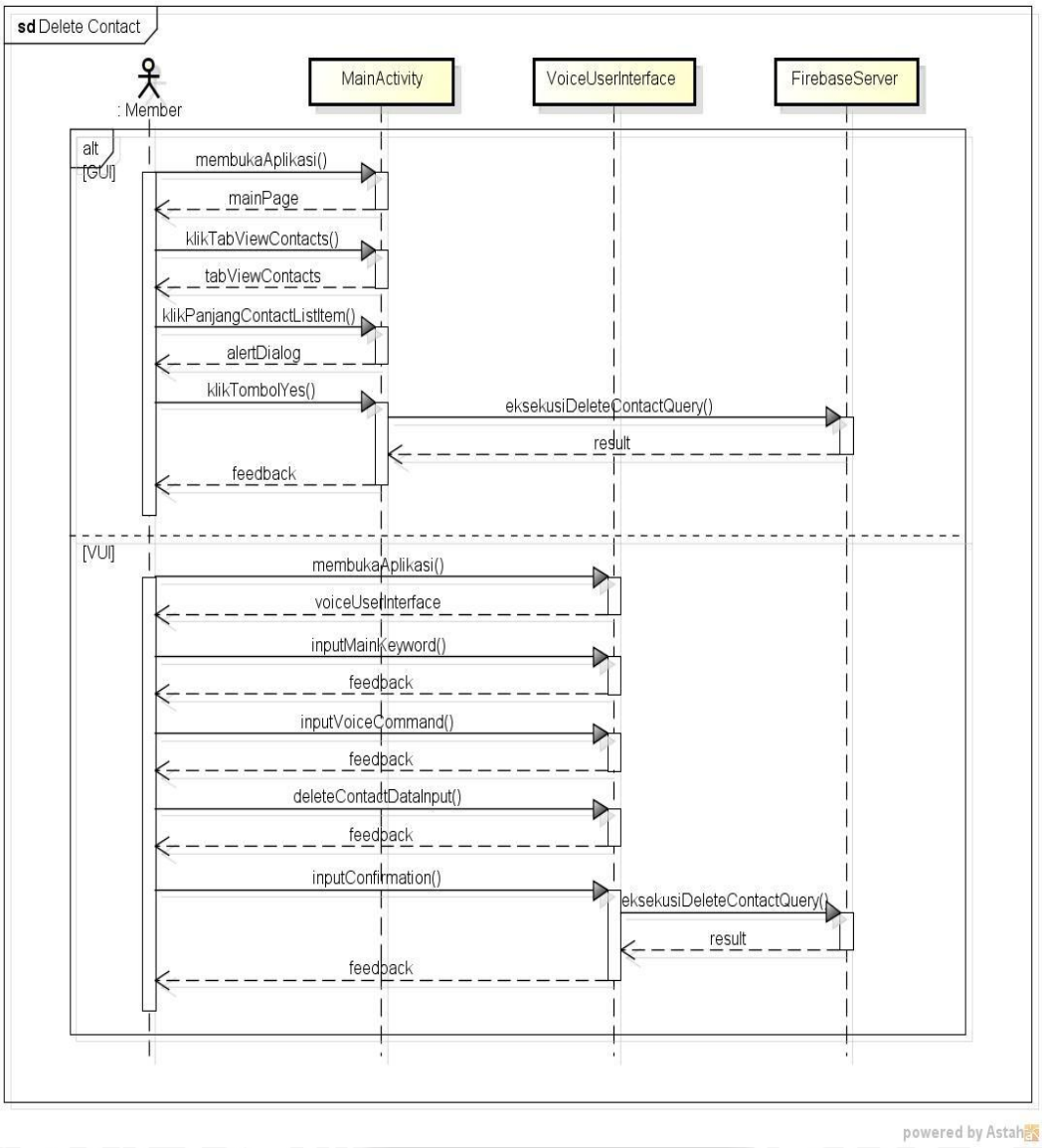
Gambar 4.20 menunjukan *sequence diagram delete conversation.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas menghapus percakapan (*delete conversation).* Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterface* dan *FirebaseServer.* Sedangkan pada penggunaan sistem *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity* dan *FirebaseServer.*



Gambar 4.20 *Sequence Diagram Delete Conversation*

4.2.1.9 *Sequence Diagram Delete Contact*

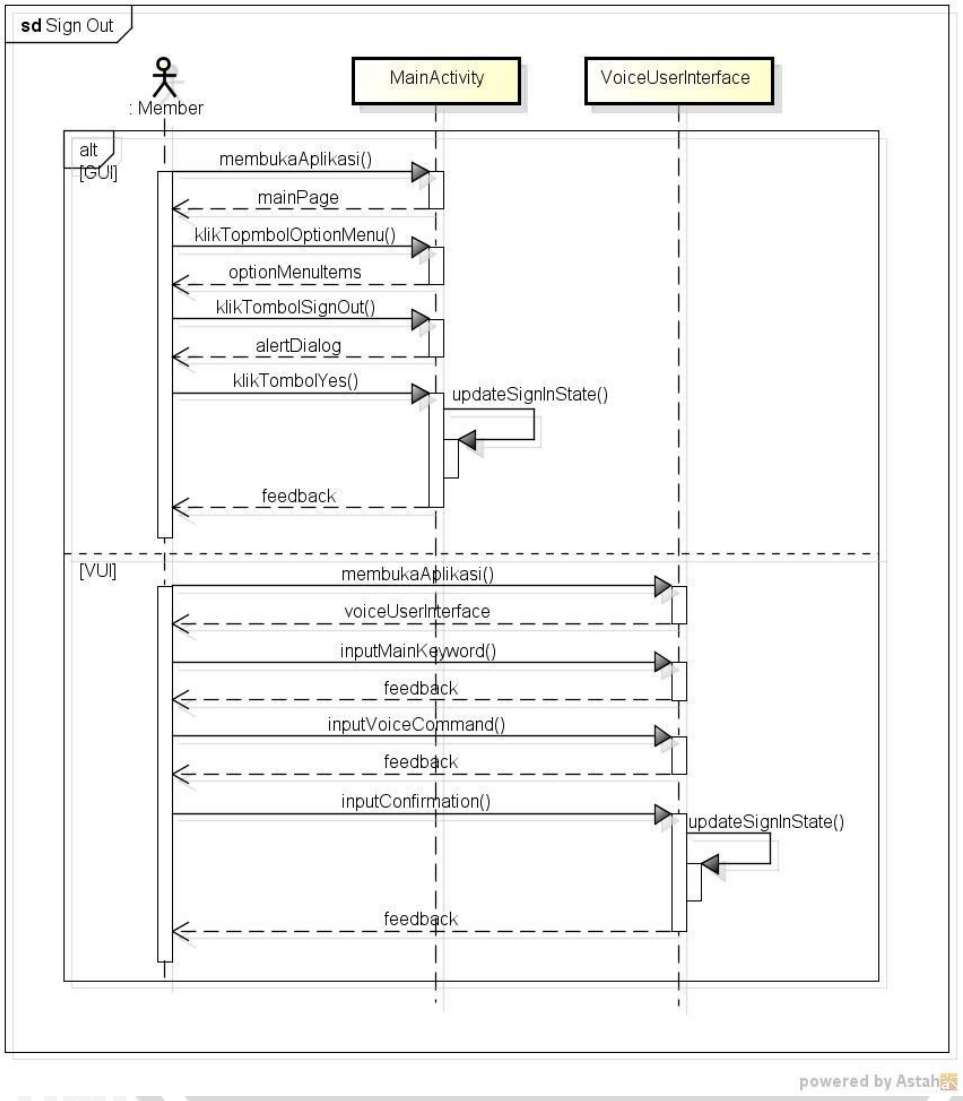
Gambar 4.21 menunjukan *sequence diagram delete contact.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas menghapus kontak (*delete contact).* Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterfaceActivity* dan *FirebaseServer.* Sedangkan pada penggunaan *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity* dan *FirebaseServer.*



Gambar 4.21 *Sequence Diagram Delete Contact*

4.2.1.10 *Sequence Diagram Sign Out*

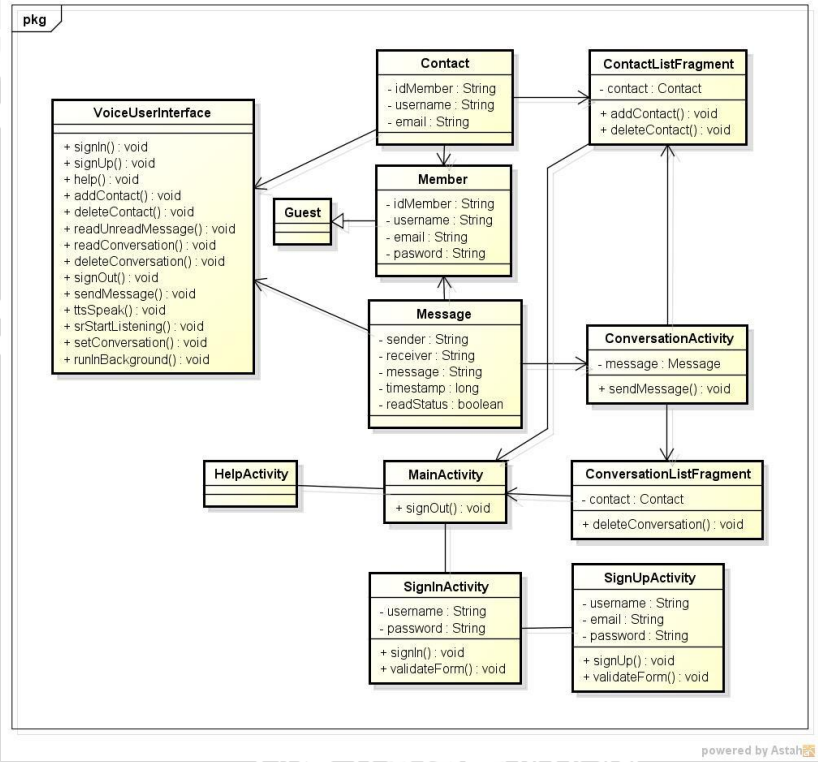
Gambar 4.22 menunjukan *sequence diagram sign out.* Diagram ini menjelaskan interaksi objek-objek yang terlibat pada aktifitas *sign out.* Aktifitas ini akan melibatkan objek *Member* sebagai aktornya. Pada penggunaan sistem *voice user interface,* objek *Member* akan berinteraksi dengan objek *VoiceUserInterface.* Sedangkan pada penggunaan sistem *graphical user interface,* objek *Member* akan berinteraksi dengan objek *MainActivity.*



Gambar 4.22 *Sequence Diagram Sign Out*

4.2.2 Perancangan *Class Diagram*

Class Diagram merupakan diagram yang digunakan untuk memodelkan sebuah sistem berdasarkan sudut pandang *class* yang membentuknya beserta dengan relasi-relasinya. Setiap *class* merupakan cetak biru (*blue print*) dari setiap objek-objek yang saling berinteraksi seperti yang terlihat pada diagram *sequence* sebelumnya. Masing-masing *class* tersebut tersusun atas *attribute* dan *behaviour* dan relasi diantaranya ditandai melalui simbol garis penghubung atau yang dikenal dengan istilah *link.* Di bawah ini merupakan rancangan *class* diagram dari sistem perangkat lunak yang akan dibangun.

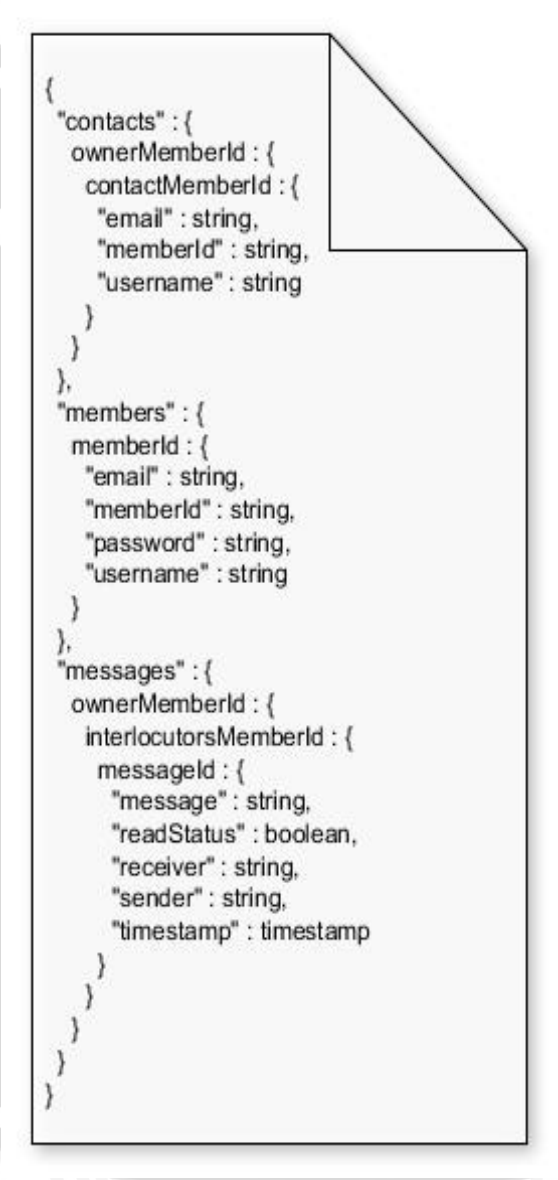


Gambar 4.23 *Class Diagram*

Gambar 4.23 menunjukkan tentang rancangan *class* diagram dari sistem aplikasi *messaging* berbasis *voice interaction*. Diagram ini memuat sebanyak 12 *class* yang berelasi satu sama lain. *Class Guest* dan *Member* merupakan *class* yang nantinya bertindak sebagai *blueprint* dari pengguna sistem perangkat lunak. Sedangkan c*lass Contact* dan *Message* masing-masing akan bertindak sebagai *blueprint* dari pesan dan kontak yang dimiliki oleh setiap pengguna. *Class VoiceUserInterface* merupakan *class* yang nantinya bertindak sebagai elemen yang akan berinteraksi dengan pengguna melalui sistem antar muka suara dan *class SignInActivity, SignUpActivity, MainActivity, ContactListFragment, ConversationListFragment, ConversationActivity* dan *HelpActivity* bertindak sebagai elemen yang akan berinteraksi dengan pengguna melalui sistem antar muka grafis.

4.2.3 Perancangan Basis Data

Perancangan basis data digunakan untuk memodelkan sistem perangkat lunak yang akan dibangun berdasarkan sudut pandang struktur basis datanya (*database*). Pada penelitian ini, perancangan basis data dilakukan dengan menggunakan NoSQL *embedded documents data modeling.* Hal ini dilakukan karena Firebase *real time database* menyimpan data dalam bentuk *single JSON document.* Proses perancangan basis data ini, menghasilkan sebuah rancangan *single document database model* seperti yang ditunjukkan oleh Gambar 4.24.



Gambar 4.24 ***Single Document Database Model***

4.2.4 Perancangan *Page Flow*

Perancangan *page flow* merupakan sebuah cara yang digunakan untuk memodelkan perangkat lunak dari sisi sistem navigasi antarmukanya. Pada aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra, proses perancangan *page flow* dilakukan secara khusus berdasarkan hasil perancangan antarmuka grafis yang telah diperoleh sebelumnya, seperti yang ditunjukkan oleh Gambar 4.25 hingga Gambar 4.30. Perancangan *page flow* tersebut menghasilkan sebuah sistem navigasi yang mampu menghubungkan seluruh antarmuka grafis yang ada. Hasil perancangan *page flow* tersebut dapat dilihat melalui Gambar 4.31.

Gambar 4.25 Perancangan *Page Flow*

4.2.5 Perancangan Antarmuka

Perancangan antarmuka merupakan sebuah cara yang digunakan untuk memodelkan suatu perangkat lunak dari sisi antarmukanya. Pada umumnya, perancangan ini memuat seluruh penjelasan terkait rancangan komponen dan elemen umum dari antarmuka yang akan diterapkan atau diimplementasikan pada perangkat lunak nantinya. Pada perancangan antarmuka aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra terdapat 6 bagian utama yang menjelaskan rancangan komponen dan elemen umum dari antarmuka yang akan diimplementasikan nantinya. Bagian-bagian tersebut adalah *register page, sign in page, main page* yang memuat *contacts and conversations fragments*, *help page* serta *conversation page.*

4.2.5.1 *Sign Up Page*

Sign up page merupakan antarmuka yang dirancang sebagai media interaksi bagi pengguna dalam melakukan proses pendaftaran. Rancangan antarmuka ini dapat dilihat melalui Gambar 4.25. Gambar 4.25 memuat 3 elemen utama yang dimiliki oleh antarmuka *sign up page,* yaitu:

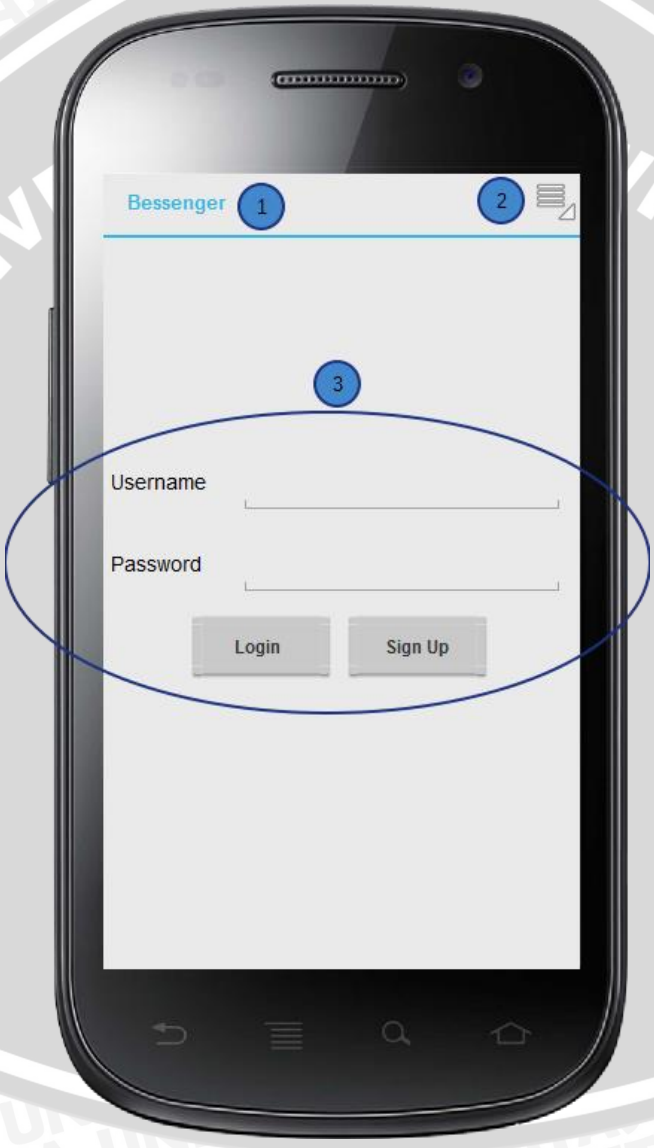
1. *App Bar* yang memuat nama aplikasi.
2. *Options Menu* yang memuat *item help.*
3. *Form* pendaftaran.

Gambar 4.26 Antarmuka *Sign Up Page*

4.2.5.2 *Sign In Page*

Sign In Page merupakan antarmuka yang dirancang sebagai media interaksi bagi pengguna dalam melakukan proses masuk sebagai *member* ke dalam sistem. Rancangan antarmuka ini dapat dilihat melalui Gambar 4.26. Gambar 4.26 memuat 3 elemen utama yang dimiliki oleh antarmuka *sign in page,* yaitu:

1. *App Bar* yang memuat nama aplikasi.
2. *Options Menu* yang memuat *item help.*
3. *Form sign in.*

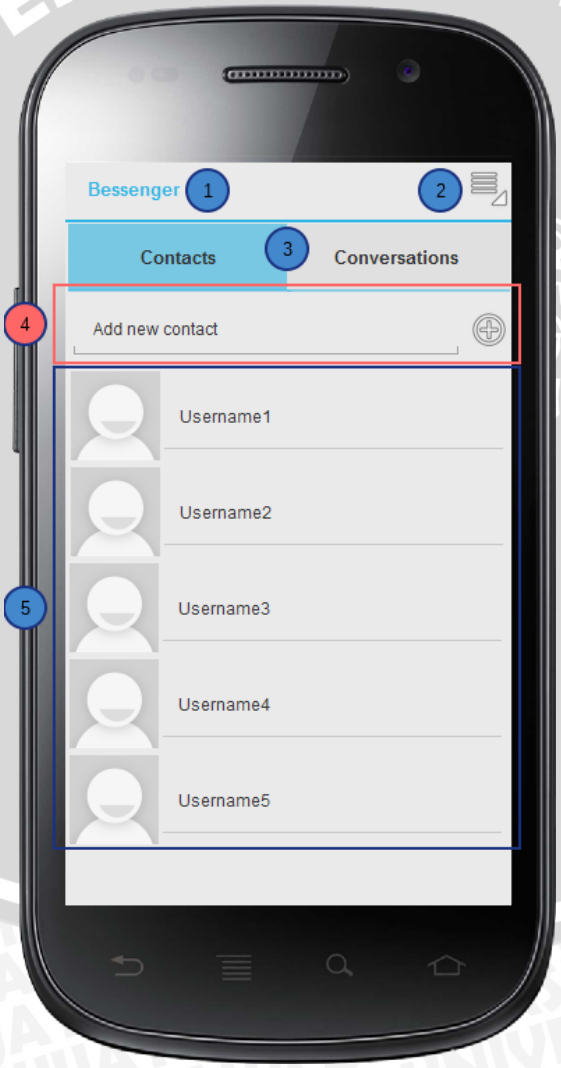


Gambar 4.27 Antarmuka *Sign In Page*

4.2.5.3 *Contact List Fragment*

Contact list fragment merupakan salah satu *fragment* yang terdapat pada antar muka *main page* selain *conversation list fragment.* Antarmuka ini dirancang sebagai media interaksi pengguna dalam melakukan proses manajemen kontak. Rancangan antarmuka ini dapat dilihat melalui Gambar 4.27. Gambar 4.27 memuat 5 elemen utama yang dimiliki oleh *main page* dengan *contact list fragment,* yaitu:

1. *App Bar* yang memuat nama aplikasi.
2. *Options Menu* yang memuat *item help* dan *sign out.*
3. *Action Bar* yang memuat *tab items contacts* dan *conversations.*
4. *Form* penambahan kontak.
5. Daftar kontak.

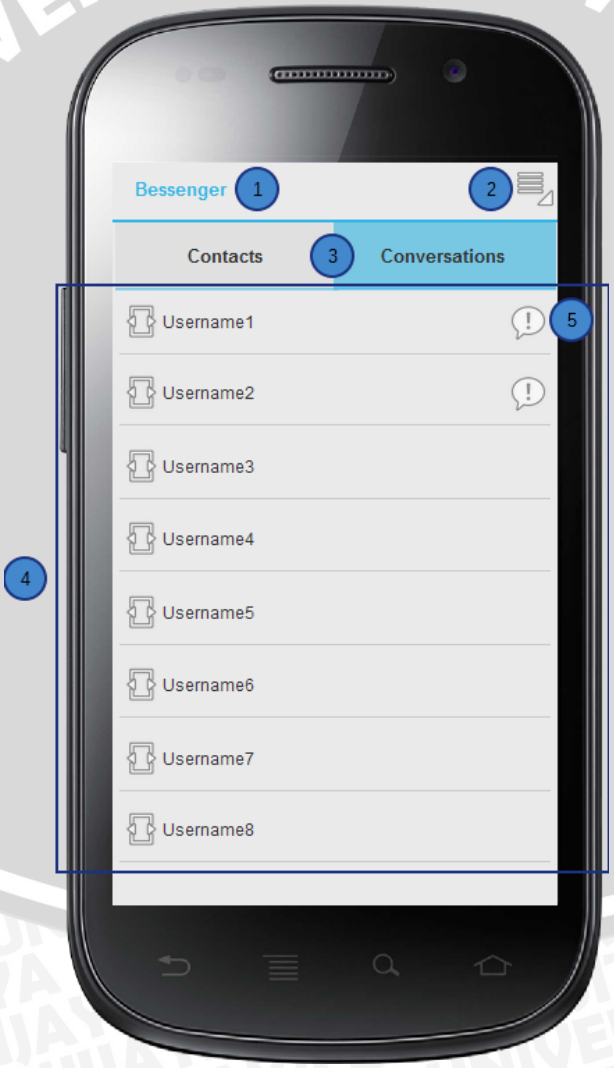


Gambar 4.28 Antarmuka *Contact List Fragment*

4.2.5.4 *Conversation List Fragment*

Conversation list fragment merupakan salah satu *fragment* yang terdapat pada antar muka *main page* selain *contact list fragment.* Antarmuka ini dirancang sebagai media interaksi pengguna dalam melakukan proses manajemen daftar percakapan. Rancangan antarmuka ini dapat dilihat melalui Gambar 4.28. Gambar 4.28 memuat 5 elemen utama yang dimiliki oleh *main page* dengan *conversation list fragment,* yaitu:

1. *App Bar* yang memuat nama aplikasi.
2. *Options Menu* yang memuat *item help* dan *sign out.*
3. *Action Bar* yang memuat *tab items contacts* dan *conversations.*
4. Daftar percakapan.
5. Indikator pesan masuk yang belum dibaca.

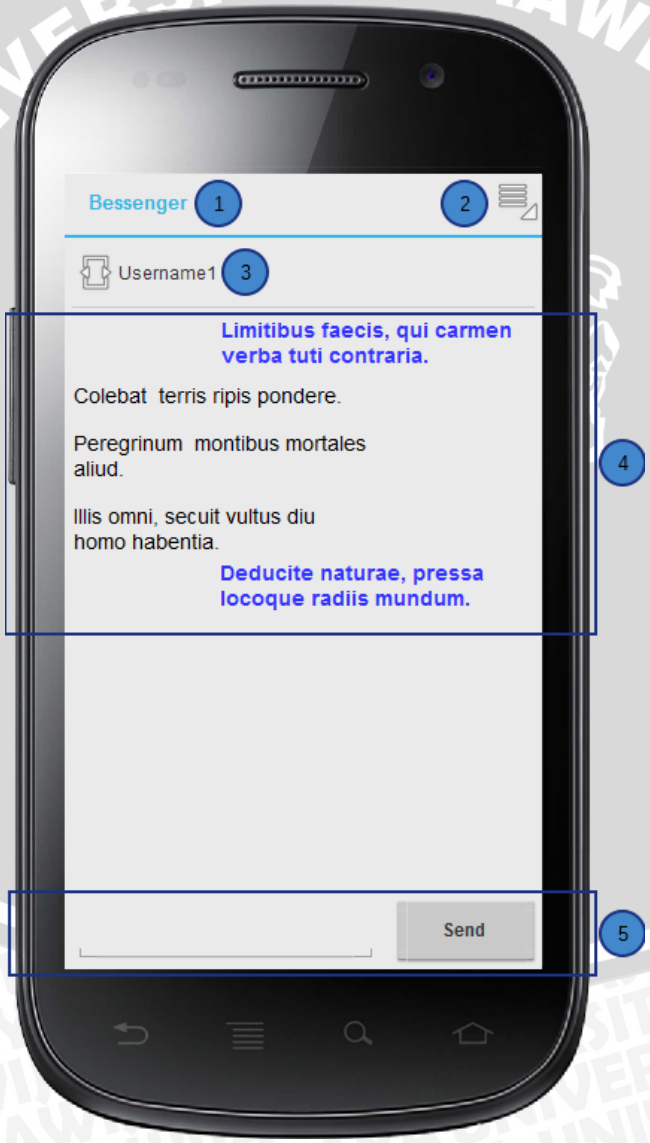


Gambar 4.29 Antarmuka *Conversation list Fragment*

4.2.5.5 *Conversation Page*

Conversation page merupakan antarmuka yang dirancang sebagai media interaksi bagi pengguna dalam melakukan proses percakapan (*chatting)* dan pengiriman pesan. Rancangan antarmuka ini dapat dilihat melalui Gambar 4.29. Gambar 4.29 memuat 5 elemen utama yang dimiliki oleh antarmuka *conversation page,* yaitu:

1. *App Bar* yang memuat nama aplikasi.
2. *Options Menu* yang memuat *item help* dan *sign out.*
3. *TextView* yang menyatakan *chat partner username.*
4. Daftar pesan yang terdapat dalam percakapan.
5. *Form* pengiriman pesan.

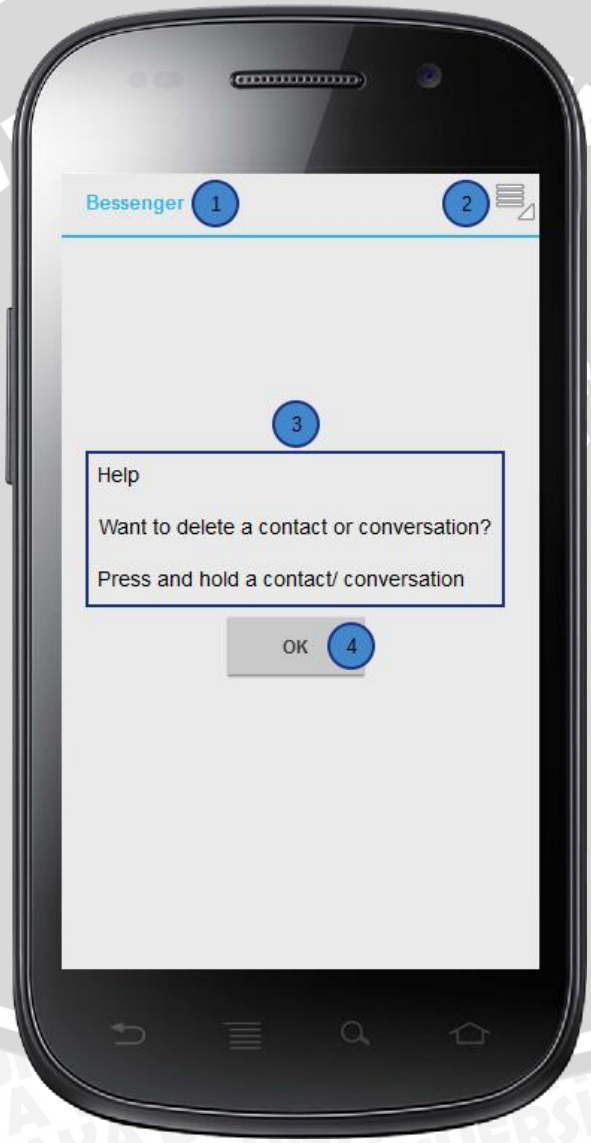


Gambar 4.30 Antarmuka *Conversation Page*

4.2.5.6 *Help Page*

Help page merupakan antarmuka yang dirancang sebagai media interaksi bagi pengguna dalam melihat manual penggunaan aplikasi. Rancangan antarmuka ini dapat dilihat melalui Gambar 4.30. Gambar 4.30 memuat 3 elemen utama yang dimiliki oleh antarmuka *help page,* yaitu:

1. *App Bar* yang memuat nama aplikasi.
2. *Options Menu* yang memuat *item sign out.*
3. Konten manual penggunaan aplikasi.
4. Tombol yang berfungsi untuk menutup *help page.*



Gambar 4.31 Antarmuka *Help Page*

BAB 5 IMPLEMENTASI

Bagian ini menjelaskan seluruh proses dan hasil implementasi dari sistem messaging berbasis voice interaction pada sistem operasi Android. Seluruh proses implementasi yang ada dibuat berdasarkan hasil analisis dan perancangan yang telah diperoleh sebelumnya. Proses dan hasil implementasi tersebut diuraiakan melalui bagian spesifikasi lingkungan pengembangan sistem, batasan-batasan implementasi, basis data, implementasi class, implementasi kode program, implementasi antar muka dan stroyboard. Seluruh proses tersebut menghasilkan sebuah sistem messaging berbasis *voice interaction* pada sistem operasi Android yang siap diuji.

5.1 Spesifikasi Lingkungan Pengembangan Sistem

Proses pengembangan sistem *messaging* berbasis *voice interaction* ini dilakukan pada sebuah lingkungan (*enviroment*) yang terbagi menjadi 2 bagian, yakni perangkat keras (*hardware*) dan perangkat lunak (*software*). Perangkat keras yang digunakan merupakan komputer lipat (laptop) ASUS A555LB dengan spesifikasi RAM sebesar 8GB, Harddisk sebesar 1TB dan dukungan *processor* Intel Core i5-5200U dengan frekuensi maksimum sebesar 2.7 GHz. Sedangkan perangkat lunak yang digunakan adalah Android Studio 2.2.2 dan bahasa pemrograman Java. Perangkat lunak tersebut digunakan di atas sistem operasi Windows 10 Pro versi 64 bit.

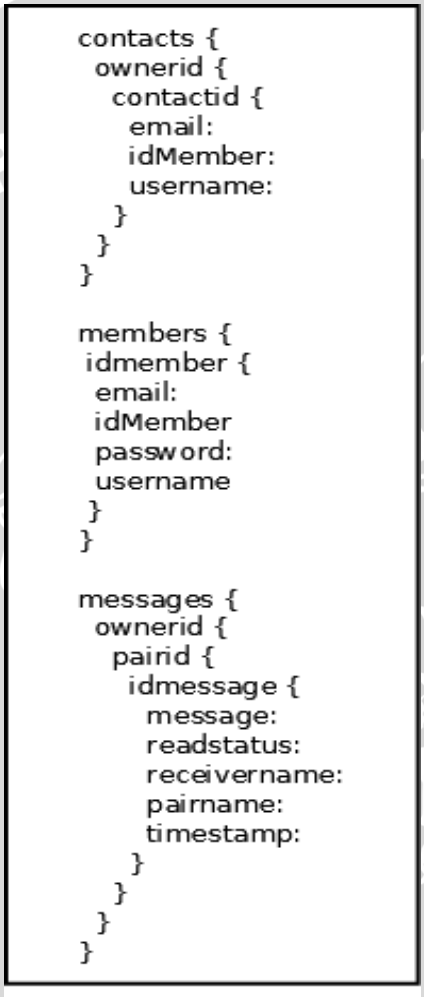
5.2 Batasan-Batasan Implementasi

Proses implementasi sistem messaging berbasis *voice interaction* memiliki batasan-batasan sebagai berikut:

1. Sistem perangkat lunak dibangun dengan bahasa pemrograman Java dan XML (Extensible Markup Language).
2. Sistem perangkat lunak dibangun dengan menerapkan penggunaan Firebase *Realtime Database* dengan struktur penyimpanan data berbentuk JSON-tree.
3. Sistem perangkat lunak yang dibuat hanya dapat dijalankan pada perangkat *smartphone* Android dengan spesifikasi minimum Android 4.1.2 atau yang dikenal dengan nama Android Ice Cream Sandwich.
4. Sistem perangkat lunak memiliki sejumlah fungsi yang dapat dijalankan dalam keadaan *offline* atau tidak terhubung dengan jaringan internet. Namun, untuk melakukan komunikasi dengan server, sistem perangkat lunak harus terhubung terlebih dahulu dengan jaringan internet (*online*).

5.3 Implementasi Perancangan Basis Data

Implementasi penyimpananan pada aplikasi ini menggunakan NoSQL Database yang disediakan oleh Firebase *Realtime Database*. Firebase *Realtime Database* melakukan penyimpanan data dalam bentuk JSON (Java Script Object Notation) *objects*. Proses implementasi basis data ini menghasilkan 3 buah JSON *objects* utama, yaitu *contacts*, *members*, dan *messages*. Masing-masing dari JSON Object ini berfungsi untuk menyimpan data kontak milik pengguna, informasi dasar pengguna serta data pesan yang dimiliki oleh pengguna. Hasil implementasi tersebut dapat dilihat pada JSON *tree* berikut ini:



Gambar 5.1 Database JSON Tree

5.4 Implementasi Perancangan *Class Diagram*

Pada bagian ini, rancangan *class diagram* yang ditunjukkan oleh Gambar 4.25 diimplementasikan ke dalam bentuk berkas-berkas kode program dengan ekstensi .java. Pada umumnya, masing-masing dari *class* yang terdapat pada *class diagram* tersebut diwakili oleh sebuah berkas kode .java. Sejumlah berkas kode tersebut juga berhubungan dengan sebuah berkas dengan ekstensi .xml. Berkas dengan ekstensi .xml ini berperan sebagai *layout* antar muka yang akan ditampilkan kepada pengguna. Daftar berkas-berkas kode dengan ekstensi .java beserta hubungannya dengan berkas dengan ekstensi .xml yang digunakan dalam proses implementasi ini dapat dilihat pada Tabel 5.1.

Tabel 5.1 Tabel Hasil Implementasi Perancangan *Class Diagram*

No	Package	Nama Class	Nama Berkas Program	Nama Berkas Layout
1	ls.mobile.simplechat	LoginActivity	LoginActivity.java	activity_login.xml
2	ls.mobile.simplechat	Register Activity	RegisterActivity.java	activity_register.xml
3	ls.mobile.simplechat	MainActivity	MainActivity.java	activity_main.xml
4	ls.mobile.simplechat	ContactList Fragment	ContactListFragment .java	fragment_contacts.xml
5	ls.mobile.simplechat	Conversation ListFragment	ConversationList Fragment.java	fragment_conversations.xml
6	ls.mobile.simplechat	ConversationActivity	ConversationActivity .java	activity_conversation.xml
7	ls.mobile.simplechat	Collection PagerAdapter	CollectionPagerAdapter	-
8	ls.mobile.simplechat	Conversation ListAdapter	ConversationList Adapter	-
9	ls.mobile.simplechat	ConversationAdapter	ConversationAdapter	-
10	ls.mobile.simplechat	VoiceUser Interface	VoiceUserInterface.java	-

11	ls.mobile.simplechat.firebase.authentication	Authentication	Authentication.java	-
12	ls.mobile.simplechat.firebase.realtime_database	DBObject Contacts	DBObjectContacts.java	-
13	ls.mobile.simplechat.firebase.realtime_database	DBObject Members	DBObjectMembers.java	-
14	ls.mobile.simplechat.firebase.realtime_database	DBObject Messages	DBObjectMessages.java	-
15	ls.mobile.simplechat.firebase.realtime_database	DBObject MessagesUid	DBObjectMessagesUid.java	-
16	ls.mobile.simplechat.nonactivity_class	MyText Watcher	MyTextWatcher.java	-
17	ls.mobile.simplechat.nonactivity_class	MyValue Event Listener	MyValueEventListener	-

18	ls.mobile.simplechat	Member	Member.java	-
19	ls.mobile.simplechat	Contact	Contact.java	-
20	ls.mobile.simplechat	Message	Message.java	-
21	ls.mobile.simplechat	Conversation	Conversation.java	-

5.5 Implementasi Kode Program

Pada bagian ini diuraikan hasil implementasi kode program yang digunakan pada aplikasi *messaging* berbasis *voice interaction*. Uraian implementasi kode yang dimuat berasal dari sejumlah proses yang dianggap berperan penting dalam fitur-fitur yang terdapat pada aplikasi. Proses yang dimaksud adalah proses mendaftar, masuk, menambah kontak, menghapus kontak, mengirim pesan, membaca pesan masuk baru, membaca percakapan, menghapus percakapan dan keluar.

5.5.1 Implementasi Kode Proses Mendaftar

Proses mendaftar atau *register* merupakan proses yang memungkinkan pengguna untuk melakukan pendaftaran akun *member* ke dalam sistem. Proses mendaftar membutuhkan *credential* berupa *user name*, *email* dan *password* milik pengguna. Credential tersebut dapat diberikan oleh pengguna melalui interaksinya dengan *Graphical User Interface* (GUI) yang disediakan oleh aplikasi. Implementasi *code* proses mendaftar dapat dilihat pada Kode 5.1.

```
1   firebaseAuth.createUserWithEmailAndPassword(etEmail.getText().toStrin
    g(), etPassword.getText().toString())
            .addOnCompleteListener(this, new
    OnCompleteListener<AuthResult>() {
2               @Override
3               public void onComplete(@NonNull Task<AuthResult> task) {
4                   Log.d(TAG,
    "createUserWithEmailAndPassword:onComplete: " + task.isSuccessful());
5                   // If sign in fails, display a message to the user.
    If sign in succeeds
6                   // the auth state listener will be notified and logic
    to handle the
7                   // signed in user can be handled in the listener.
8                   if (!task.isSuccessful()) {
9                       Toast.makeText(RegisterActivity.this,
    R.string.auth_failed, Toast.LENGTH_SHORT).show();
10                  } else {
11                      // [START writeNewMember data to database]
12                      String idMember, username, email, password;
13                      idMember =
    firebaseAuth.getCurrentUser().getUid().toString();
14                      username = etUsername.getText().toString();
```

```
15                      email = etEmail.getText().toString();
16                      password = etPassword.getText().toString();
17                      Member member = new Member(idMember, username,
    email, password);
18
19  databaseReference.child("members").child(idMember).setValue(member);
20                      // [END writeNewMember data to database]
21
22                      Toast.makeText(RegisterActivity.this,
    R.string.registration_succeed, Toast.LENGTH_LONG).show();
23
24                      // Redirect to LoginActivity
25                      Intent intent = new
    Intent(getApplicationContext(), LoginActivity.class);
26                      startActivity(intent);
27                  }
28              }
29          });
```

Kode 5.1 Implementasi Kode Proses Mendaftar

Kode 5.1 memiliki penjelasan sebagai berikut:

1. Baris 1 berfungsi untuk melakukan proses pendaftaran melalui layanan Firebase *User Authentication* dengan menggunakan *email* dan *password* yang diberikan oleh pengguna.
2. Baris 3-28 berfungsi untuk melakukan penanganan terhadap status proses pendaftaran. Apabila proses pendaftaran berhasil, maka sistem akan melalukan proses penulisan data pengguna ke *database* yang diikuti dengan pemberian *feedback* kepada pengguna. Sedangkan apabila proses pendaftaran gagal, maka sistem hanya akan melakukan pemberian *feedback* kepada pengguna.

5.5.2 Implementasi Kode Proses Masuk

Proses masuk merupakan proses yang memungkinkan *user* untuk masuk sebagai *member* ke dalam sistem. Proses ini membutuhkan *credential* berupa *username* dan *password* milik pengguna sesuai dengan akun *member* yang telah didaftarkannya sebelumnya. *Credential* tersebut dapat diberikan pengguna melalui interaksinya dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Implementasi kode proses masuk dapat dilihat pada Kode 5.2.

```
1   public void signInWithEmailAndPassword(String email, String
    password){
2       this.firebaseAuth.signInWithEmailAndPassword(email,
    password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
3           @Override
4           public void onSuccess(AuthResult authResult) {
5               sendFeedbackForSignInSuccessResult();
6           }
7       }).addOnFailureListener(new OnFailureListener() {
8           @Override
9           public void onFailure(@NonNull Exception e) {
10              sendFeedbackForSignInFailedResult();
11          }
12      });
13  }
```

Kode 5.2 Implementasi Kode Proses Masuk

Kode 5.2 memiliki penjelasan sebagai berikut:

1. Baris 1 merupakan nama *method* yang digunakan untuk proses masuk.
2. Baris 2-13 berfungsi untuk melakukan penanganan terhadap status proses masuk. Apabila proses masuk berhasil, maka secara otomatis sistem akan memperbarui *authentication state* pada aplikasi dan kemudian memberikan *feedback* kepada pengguna. Sedangkan apabila proses tersebut gagal, maka sistem hanya akan memberikan *feedback* kepada pengguna.

5.5.3 Implementasi Kode Proses Menambah Kontak

Proses menambah kontak merupakan proses yang memungkinkan *member* untuk menambahkan *member* lain ke dalam daftar kontaknya. Untuk melakukan proses ini data yang dibutuhkan adalah *username* dari *member* yang ingin ditambahkan. Proses ini dapat dilakukan melalui interaksi dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Implementasi kode proses menambah kontak dapat dilihat pada Kode 5.3.

```
1   public void addContact(String givenUsername){
2       final String username = givenUsername.toLowerCase();
3
4       /** Query for getting a contact child node based on given
    username */
5       Query query = databaseReference.child("contacts")
6               .child(firebaseAuth.getCurrentUser().getUid())
7               .orderByChild("username").equalTo(username);
8
9       query.addListenerForSingleValueEvent(new MyValueEventListener() {
10          @Override
11          public void onDataChange(DataSnapshot dataSnapshot) {
12              /**
13               * dataSnapshot.getChildrenCount() > 0 means the given
    username is already exist on
14               * user contact list. If the given username isn't already
    exist on user contact
15               * list, try to get member info from members object node
    based on given username
16               */
17              if (dataSnapshot.getChildrenCount() > 0){
18                  sendFeedbackForAddAlreadyExistContactResult();
19              } else {
20                  /** Query for getting a members child node based on
    given username */
21                  Query query =
    databaseReference.child("members").orderByChild("username")
22                          .equalTo(username);
23
24                  query.addListenerForSingleValueEvent(new
    MyValueEventListener() {
25                      @Override
26                      public void onDataChange(DataSnapshot
    dataSnapshot) {
27                          /**
28                           * dataSnapshot.getChildrenCount() == 0 means
    the given username isn't
29                           * registered. If the given username is
    registered, write the username
30                           * and it's details on database.
```

```
31                          */
32                         if (dataSnapshot.getChildrenCount() == 0){
33
34  sendFeedbackForAddNotValidContactResult();
35                         } else {
36                             String uid =
    firebaseAuth.getCurrentUser().getUid();
37                             String idMember = null;
38                             String tempUsername = null;
39                             String email = null;
40                             for (DataSnapshot memberSnapshot :
    dataSnapshot.getChildren()){
41                                 Member member =
    memberSnapshot.getValue(Member.class);
42                                 idMember = member.idMember;
43                                 tempUsername = member.username;
44                                 email = member.email;
45                             }
46                             final String username = tempUsername;
47                             Contact contact = new Contact(idMember,
    username, email);
48
49  databaseReference.child("contacts").child(uid).child(idMember)
50                                 .setValue(contact)
51                                 .addOnCompleteListener(new
    OnCompleteListener<Void>() {
52                                 @Override
53                                 public void onComplete(@NonNull
    Task<Void> task) {
54                                 if (task.isSuccessful()){
55                                     String feedback = username +
                                                " successfuly added to
    your contact list.";
56
57  sendFeedbackForSuccessAddContactResult(feedback);
58                                 }
59                             }
60                         });
61                     }
62                 }
63             });
64         }
65     }
66 });
67 }
```

Kode 5.3 Implementasi Kode Proses Menambah Kontak

Kode 5.3 memiliki penjelasan sebagai berikut:

1. Baris 1 merupakan nama *method* yang digunakan untuk proses menambah kontak.
2. Baris 2 berfungsi untuk inisialisasi variabel *username* dari kontak yang ingin ditambahkan.
3. Baris 3-67 berfungsi untuk melakukan *query* penambahan kontak terhadap *database*. Berdasarkan kode yang terdapat pada baris ini, aplikasi akan memberikan *feedback* kepada pengguna sesuai dengan penanganan terhadap hasil *query* yang telah dilakukan.

5.5.4 Implementasi Kode Proses Menghapus Kontak

Proses menghapus kontak merupakan proses yang memungkinkan *member* untuk menghapus *member* lain dari dalam daftar kontaknya. Untuk melakukan proses ini data yang dibutuhkan adalah *username* dari *member* yang ingin dihapus dari daftar kontak. Proses ini dapat dilakukan melalui interaksi dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Implementasi kode proses menambah kontak dapat dilihat pada Kode 5.4.

```
1   public void deleteContact(String givenUsername){
2           String username = givenUsername.toLowerCase();
3
4           /** Query for getting a contacts/$uid child node based on
    selected contact */
5           Query query = databaseReference.child("contacts")
6                   .child(firebaseAuth.getCurrentUser().getUid())
7                   .orderByChild("username").equalTo(username);
8
9           query.addListenerForSingleValueEvent(new
    MyValueEventListener() {
10              @Override
11              public void onDataChange(DataSnapshot dataSnapshot) {
12                  /**
13                   * dataSnapshot.getChildrenCount() == 0 means the
    given username isn't exist on
14                   * user contact list or not valid. If the given
    username exist, delete the username and it's
15                   * details from user contact list database.
16                   */
17                  if (dataSnapshot.getChildrenCount() == 0){
18                      sendFeedbackForDeleteNotValidContactResult();
19                  } else {
20                      String idContact = null;
21                      for (DataSnapshot contactSnapshot :
    dataSnapshot.getChildren()){
22                          Contact contact =
    contactSnapshot.getValue(Contact.class);
23                          idContact = contact.idMember;
24                      }
25                      databaseReference.child("contacts")

26  .child(firebaseAuth.getCurrentUser().getUid())
27                              .child(idContact).removeValue()
28                              .addOnCompleteListener(new
    OnCompleteListener<Void>() {
29                                  @Override
30                                  public void onComplete(@NonNull
    Task<Void> task) {
31
32  sendFeedbackForSuccessDeleteContactResult();
33                                  }
34                              });
35                  }
36              }
37          });
38      }
```

Kode 5.4 Implementasi Kode Proses Menghapus Kontak

Kode 5.4 memiliki penjelasan sebagai berikut:

1. Baris 1 merupakan nama *method* yang digunakan untuk proses menghapus kontak.

2. Baris 2 berfungsi untuk inisialisasi variabel *username* dari kontak yang ingin dihapus.
3. Baris 4-38 berfungsi untuk melakukan *query* penghapusan kontak terhadap *database*. Berdasarkan kode yang terdapat pada baris ini, aplikasi akan memberikan *feedback* kepada pengguna sesuai dengan penanganan terhadap hasil *query* yang telah dilakukan.

5.5.5 Implementasi Kode Proses Mengirim Pesan

Proses mengirim pesan merupakan proses yang memungkinkan *member* untuk mengirimkan pesan kepada *member* lain yang terdapat pada daftar kontaknya. Untuk melakukan proses ini, data yang dibutuhkan adalah *username* dari *member* yang menjadi tujuan pengiriman pesan dan konten dari pesan itu sendiri. Proses ini dapat dilakukan melalui interakasi dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Implementasi kode proses mengirim pesan dapat dilihat pada Kode 5.5.

```
1   public void sendMessage(String username, final String message){
2           final String receiver = username.toLowerCase();
3           String messageContent = message;
4
5           /** Query for getting a contacts child node based on given
    username  */
6           Query query = databaseReference.child("contacts")
7                   .child(firebaseAuth.getCurrentUser().getUid())
8                   .orderByChild("username").equalTo(receiver);
9
10          query.addListenerForSingleValueEvent(new
    MyValueEventListener() {
11              @Override
12              public void onDataChange(DataSnapshot dataSnapshot) {
13                  /**
14                   * dataSnapshot.getChildrenCount() == 0 means the
    given username is not exist on
15                   * user contact list. If the given username is exist,
    write the message on database.
16                   */
17                  if (dataSnapshot.getChildrenCount() == 0){
18
19  sendFeedbackForInvalidReceiverUsernameSendMessageResult();
20                  } else {
21                      String receiveridTemp = null;
22                      String receiverUsernameTemp = null;
23                      for (DataSnapshot contactSnapshot :
    dataSnapshot.getChildren()){
24                          Contact contact =
    contactSnapshot.getValue(Contact.class);
25                          receiveridTemp = contact.idMember;
26                          receiverUsernameTemp = contact.username;
27                      }
28                      final String receiverid = receiveridTemp;
29                      final String receiverUsername =
    receiverUsernameTemp;
30
31                      /** Query for getting a members child node based
    on current uid */
32                      Query query = databaseReference.child("members")
33                              .orderByChild("idMember")
```

```
34  .equalTo(firebaseAuth.getCurrentUser().getUid());
35                      query.addListenerForSingleValueEvent(new
    MyValueEventListener() {
36                          @Override
37                          public void onDataChange(DataSnapshot
    dataSnapshot) {
38                              String senderUsernameTemp = null;
39                              for (DataSnapshot memberSnapshot :
    dataSnapshot.getChildren()){
40                                  Member member =
    memberSnapshot.getValue(Member.class);
41                                  senderUsernameTemp = member.username;
42                              }
43                              String senderid =
    firebaseAuth.getCurrentUser().getUid();
44                              String senderUsername =
    senderUsernameTemp;
45
46                              /** Write the message to user message
    list on database. */
47                              Message messageObj = new
    Message(senderUsername, receiverUsername,
48                                      message, true);
49                              Map<String, Object> messageValues =
    messageObj.toMap();
50                              String key =
    databaseReference.child("messages").child(senderid)

51  .child(receiverid).push().getKey();
52
53  databaseReference.child("messages").child(senderid).child(receiverid)

54  .child(key).setValue(messageValues);
55
56                              /** Write the message to receiver message
    list on database. */
57                              Message messageObj2 = new
    Message(senderUsername, receiverUsername,
58                                      message, false);
59                              Map<String, Object> messageValues2 =
    messageObj2.toMap();
60                              String key2 =
61  databaseReference.child("messages").child(receiverid)
62                                      .child(senderid).push().getKey();
63
64  databaseReference.child("messages").child(receiverid).child(senderid)

65  .child(key2).setValue(messageValues2);
66
67
68  sendFeedbackForSuccessSendMessageResult();
69                          }
70                      });
71                  }
72              }
73          });
74      }
```

Kode 5.5 Implementasi Kode Proses Mengirim Pesan

Kode 5.5 memiliki penjelasan sebagai berikut:

1. Baris 1 merupakan nama *method* yang digunakan untuk proses mengirim pesan.

2. Baris 2-3 berfungsi untuk inisialisasi variabel *username* target pengiriman pesan dan variabel konten pesan.
3. Baris 5-74 berfungsi untuk melakukan *query* pengiriman pesan terhadap database. Berdasarkan kode yang terdapat pada baris ini, aplikasi akan memberikan *feedback* kepada pengguna sesuai dengan penanganan terhadap hasil *query* yang telah dilakukan.

5.5.6 Implementasi Kode Proses Membaca Pesan Masuk Baru

Proses membaca pesan masuk baru merupakan proses yang memungkinkan *member* untuk melihat atau mendengarkan pesan masuk yang belum pernah dilihat atau didengarkan sebelumnya. Proses ini dapat dilakukan melalui interaksi dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Pada GUI, jumlah pesan masuk yang belum dibaca ditandai oleh sebuah *badge* yang terdapat pada masing-masing *item list* dari daftar percakapan yang ada. Implementasi kode proses membaca pesan masuk baru dapat dilihat pada Kode 5.6.

```
1   public void readUnreadMessage(){
2       unreadMessage.clear();
3
4       /** Query for getting a messages/$uid child node */
5       Query query = databaseReference.child("messages")
6               .child(firebaseAuth.getCurrentUser().getUid());
7
8       query.addListenerForSingleValueEvent(new MyValueEventListener() {
9           @Override
10          public void onDataChange(DataSnapshot dataSnapshot) {
11              /**
12               * dataSnapshot.getChildrenCount() == 0 means the user
    didn't have any conversation.
13               * if the user have a conversation, check for message
    with readStatus = false.
14               */
15              if (dataSnapshot.getChildrenCount() == 0){
16                  //
17              } else {
18                  for (DataSnapshot messageSnapshot :
    dataSnapshot.getChildren()){
19                      final String interlocutorsid =
    messageSnapshot.getKey();
20
21                      /** Query for getting
    messages/$uid/interlocutorsid child node */
22                      Query query = databaseReference.child("messages")

23  .child(firebaseAuth.getCurrentUser().getUid())

24  .child(interlocutorsid).orderByChild("readStatus")
25                              .equalTo(false);
26
27                      query.addListenerForSingleValueEvent(new
    MyValueEventListener() {
28                          @Override
29                          public void onDataChange(DataSnapshot
    dataSnapshot) {
30                              for (DataSnapshot messageSnapshot :
    dataSnapshot.getChildren()){
31                                  String messsageid =
```

```
    messageSnapshot.getKey();
32                                  Message message =
    messageSnapshot.getValue(Message.class);
33                                  DBObjectMessagesUid
    dbObjectMessagesUid =
                                                new DBObjectMessagesUid(
34                                                      interlocutorsid,
35                                                      messsageid,
36                                                      message);
37
38  unreadMessage.add(dbObjectMessagesUid);
39                              }
40                          }
41                      });
42                  }
43              }
44          }
45      });
46  }
```

Kode 5.6 Implementasi Kode Proses Membaca Pesan Masuk Baru

Kode 5.6 merupakan *method* yang digunakan untuk proses membaca pesan masuk baru. Pada kode ini, dilakukan *query* terhadap *database* untuk mendapatkan pesan yang memiliki status belum dibaca. Kemudian, seluruh pesan milik pengguna yang memiliki status belum dibaca tersebut akan disimpan dalam sebuah *list variable*. Nantinya, isi dari *list variable* tersebut akan dikirim kepada pengguna dalam bentuk *feedback.*

5.5.7 Implementasi Kode Proses Membaca Percakapan

Proses membaca percakapan merupakan proses yang memungkinkan *member* untuk melihat atau mendengarkan percakapan yang dimilikinya dengan *lain.* Proses ini dapat dilakukan melalui interaksi dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Pada GUI, halaman percakapan dapat diakses dengan memberikan sekali ketukan terhadap salah satu kontak ataupun salah satu percakapan yang ada. Implementasi kode proses membaca percapan dapat dilihat pada Kode 5.7.

```
1   public void readConversation(String username){
2       conversation.clear();
3
4       /** Query for getting a contacts child node based on given
    username  */
5       Query query = databaseReference.child("contacts")
6               .child(firebaseAuth.getCurrentUser().getUid())

7   .orderByChild("username").equalTo(username.toLowerCase());
8
9       query.addListenerForSingleValueEvent(new MyValueEventListener() {
10          @Override
11          public void onDataChange(DataSnapshot dataSnapshot) {
12              /**
13               * dataSnapshot.getChildrenCount() == 0 means the given
    username is not exist on user
14               * contact list as interlocutors validity parameter. If
    the given
15               * username is exist, check if the given username exist
    on user conversation list
16               * as conversation existing parameter.
17               */
```

```
18              if (dataSnapshot.getChildrenCount() == 0){
19
20  sendFeedbackForInvalidInterlocutorsUsernameReadConversationResult();
21              } else {
22                  String interlocutorsidTemp = null;
23                  for (DataSnapshot contactSnapshot :
    dataSnapshot.getChildren()){
24                      Contact contact =
    contactSnapshot.getValue(Contact.class);
25                      interlocutorsidTemp = contact.idMember;
26                  }
27                  String interlocutorsid = interlocutorsidTemp;
28
29                  /** Query for getting a messages/$uid child node
    based on given username */
30                  Query query = databaseReference.child("messages")

31  .child(firebaseAuth.getCurrentUser().getUid())
32                          .child(interlocutorsid);
33
34                  query.addListenerForSingleValueEvent(new
35  MyValueEventListener() {
36                      @Override
37                      public void onDataChange(DataSnapshot
    dataSnapshot) {
38                          /**
39                           * dataSnapshot.getChildrenCount() == 0 means
    the given username is not
40                           * exist on user conversation list. If the
    given username is exist, read
41                           * the conversation message list from
    database.
42                           */
43                          if (dataSnapshot.getChildrenCount() == 0){
44
45  sendFeedbackForNotExistConversationReadConversationResult();
46                          } else {
47                              for (DataSnapshot messageSnapshot :
    dataSnapshot.getChildren()){
48                                  Message message =
    messageSnapshot.getValue(Message.class);
49                                  conversation.add(message);
50                              }
51
52  sendFeedbackForReadyConversationReadConversationResult();
53                          }
54                      }
55                  });
56              }
57          }
58      });
59  }
```

Kode 5.7 Implementasi Kode Proses Membaca Percakapan

Kode 5.7 memiliki penjelasan sebagai berikut:

1. Baris 1 merupakan nama *method* yang digunakan untuk proses membaca percakapan.
2. Baris 2 berfungsi untuk mengosongkan *conversation list variable.*
3. Baris 4-59 berfungsi untuk melakukan *query* pembacaan percakapan terhadap *database.* Berdasarkan kode yang terdapat pada baris ini,

aplikasi akan memberikan *feedback* kepada pengguna sesuai dengan penanganan terhadap hasil *query* yang telah dilakukan.

5.5.8 Implementasi Kode Proses Menghapus Percakapan

Proses menghapus percakapan merupakan proses yang memungkinkan *member* untuk menghapus percakapannya dengan *member* lain dari daftar percakapan yang dimilikinya. Untuk melakukan proses ini, data yang dibutuhkan adalah *username* dari *member* yang percakapan kita kepadanya ingin dihapuskan. Proses ini dapat dilakukan melalui interaksi dengan *Graphical User Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Implementasi kode proses menghapus percakapan dapat dilihat pada Kode 5.8.

```
1   public void deleteConversation(String username){
2       String interlocutorsUsername = username.toLowerCase();
3
4       /** Query for getting a contacts child node based on given
    username  */
5       Query query = databaseReference.child("contacts")
6               .child(firebaseAuth.getCurrentUser().getUid())
7               .orderByChild("username").equalTo(interlocutorsUsername);
8
9       query.addListenerForSingleValueEvent(new MyValueEventListener() {
10          @Override
11          public void onDataChange(DataSnapshot dataSnapshot) {
12              /**
13               * dataSnapshot.getChildrenCount() == 0 means the given
    username is not exist on user
14               * contact list as interlocutors validity parameter. If
    the given
15               * username is exist, check if the given username exist
    on user conversation list
16               * as conversation existing parameter.
17               */
18              if (dataSnapshot.getChildrenCount() == 0){
19                  sendFeedbackForNotExistDeleteConversationResult();
20              } else {
21                  String interlocutorsidTemp = null;
22                  for (DataSnapshot contactSnapshot :
    dataSnapshot.getChildren()){
23                      Contact contact =
    contactSnapshot.getValue(Contact.class);
24                      interlocutorsidTemp = contact.idMember;
25                  }
26                  final String interlocutorsid = interlocutorsidTemp;
27
28                  /** Query for getting a messages/$uid child node
    based on given username */
29                  Query query = databaseReference.child("messages")

30  .child(firebaseAuth.getCurrentUser().getUid())
31                          .child(interlocutorsid);
32
33                  query.addListenerForSingleValueEvent(new
    MyValueEventListener() {
34                      @Override
35                      public void onDataChange(DataSnapshot
    dataSnapshot) {
36                          /**
37                           * dataSnapshot.getChildrenCount() == 0 means
    the given username is not
38                           * exist on user conversation list. If the
```

```
    given username is exist,
39                          * delete the conversation from database.
40                          */
41                         if (dataSnapshot.getChildrenCount() == 0){
42
43  sendFeedbackForNotExistDeleteConversationResult();
44                         } else {
45                             databaseReference.child("messages")
46  .child(firebaseAuth.getCurrentUser().getUid())
47  .child(interlocutorsid).removeValue()
48                                     .addOnCompleteListener(new
    OnCompleteListener<Void>() {
49                                         @Override
50                                         public void
    onComplete(@NonNull Task<Void> task) {
51                                             if (task.isSuccessful()){
52
53  sendFeedbackForSuccessDeleteConversationResult();
54                                             }
55                                         }
56                                     });
57                         }
58                     }
59                 });
60             }
61         }
62     });
63 }
```

Kode 5.8 Implementasi Kode Proses Menghapus Percakapan

Kode 5.8 memiliki penjelasan sebagai berikut:

1. Baris 1 merupakan nama *method* yang digunakan untuk proses menghapus percakapan.
2. Baris 2 berfungsi untuk inisialisasi variabel *interlocutors username* yang menjadi target pneghapusan percakapan.
3. Baris 4-63 berfungsi untuk melakukan *query* penghapusan pesan terhadap *database.* Berdasarkan kode yang terdapat pada baris ini, aplikasi akan memberikan *feedback* kepada pengguna sesuai dengan penanganan terhadap hasil *query* yang telah dilakukan.

5.5.9 Implementasi Kode Proses Keluar

Proses keluar merupakan proses yang memungkinkan *member* untuk keluar dari sistem. Proses ini dapat dilakukan melalui interkasi dengan *Graphical user Interface* (GUI) ataupun dengan *Voice User Interface* (VUI). Implementasi kode proses keluar dapat dilihat pada Kode 5.9.

```
1  public void signOut(){
2      firebaseAuth.signOut();
3      Intent intent = new Intent(context, LoginActivity.class);
4      intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
5      this.context.startActivity(intent);
6  }
```

Kode 5.9 Implementasi Kode Proses Keluar

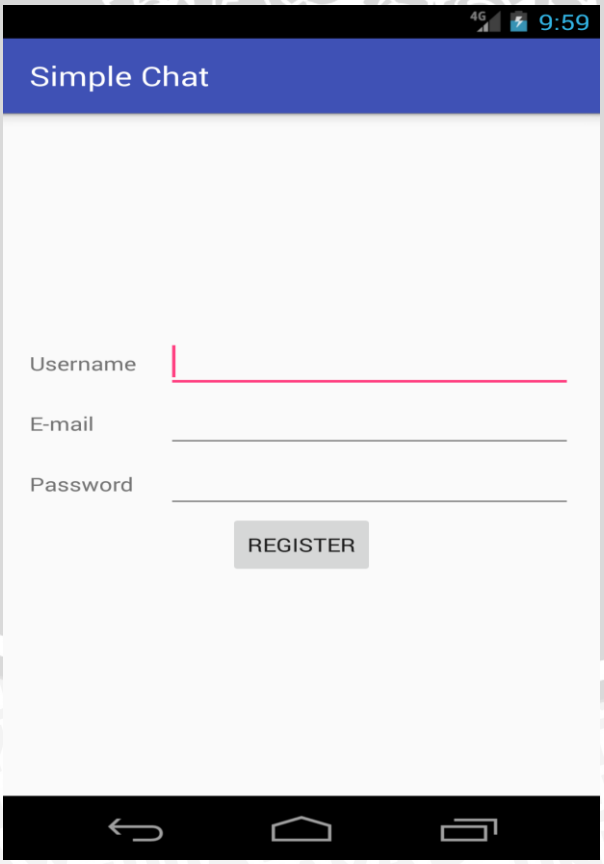
Kode 5.9 merupakan *method* yang digunakan untuk proses keluar. Pada kode ini, dilakukan pemanggilan *method sign out* melalui *instance* dari sebuah *firebase authentication.* Pemanggilan *method sign out* tersebut akan mengubah *login state* milik pengguna. Setelah hal tersebut dilakukan, pengguna akan diarahkan kembali menuju halaman *sign in*.

5.6 Impelementasi Antarmuka

Pada bagian ini diuraikan hasil implementasi antarmuka pada aplikasi *messaging* berbasis *voice interaction*. Hasil tersebut diperoleh melalui proses implementasi yang mengacu pada hasil perancangan *page flow* seperti yang ditunjukkan oleh Gambar 4.27. Hasil implementasi antarmuka yang diperoleh terdiri dari tampilan antarmuka *Register*, *Sign In*, *Contact List*, *Conversation List* dan *Conversation*. Seluruh hasil implementasi tersebut dapat dilihat melalui Gambar 5.1 sampai Gambar 5.5.

5.6.1 Implementasi Antarmuka *Sign Up*

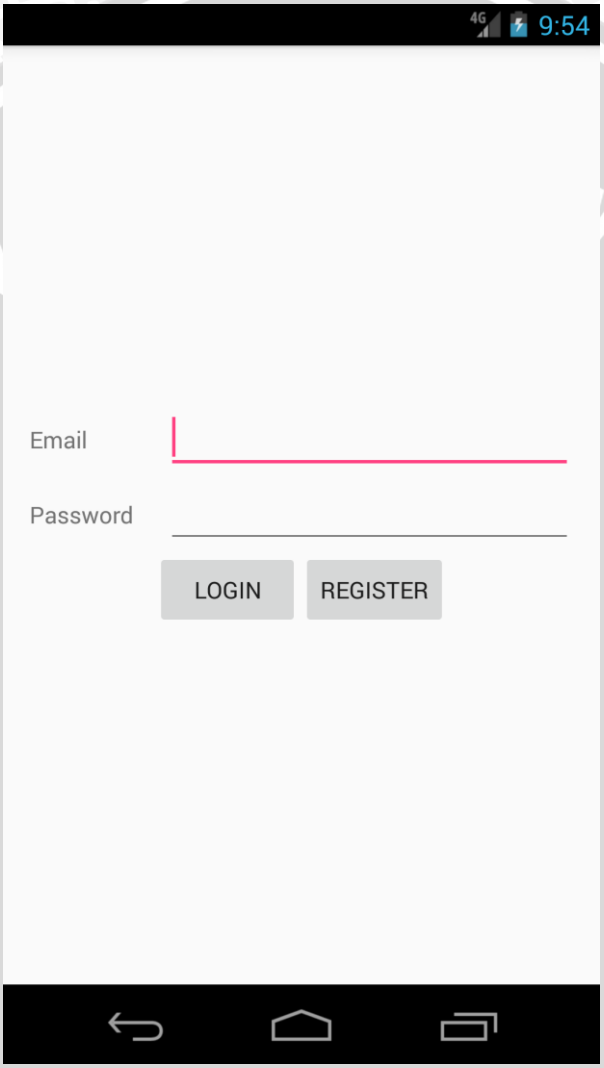
Melalui interaksi dengan tampilan antarmuka *sign up*, *member* mampu melakukan proses pendaftaran. Proses pendaftaran tersebut dapat dilakukan dengan menggunakan form pendaftaran yang telah tersedia pada tampilan ini. Hasil implementasi antarmuka *sign up* dapat dilihat melalui Gambar 5.2.



Gambar 5.2 Tampilan Antarmuka *Sign Up*

5.6.2 Implementasi Antarmuka *Sign in*

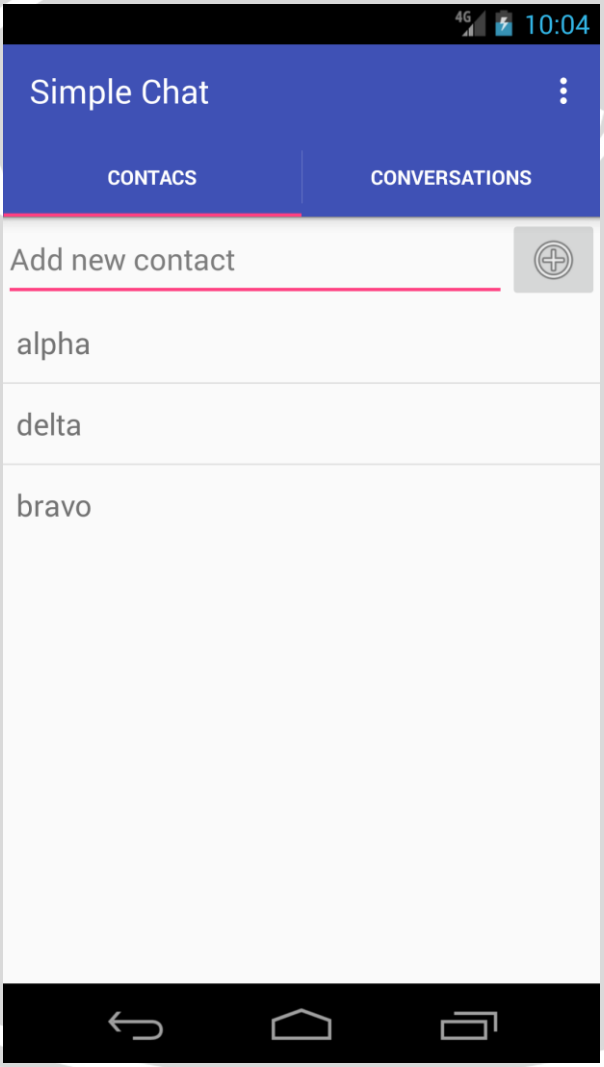
Melalui interaksi dengan tampilan antarmuka *sign in*, *member* mampu melakukan proses *sign in* serta mengakses tampilan antarmuka *register*. Proses *sign in* tersebut dapat dilakukan dengan menggunakan *form sign in* yang telah tersedia pada tampilan ini. Sedangkan untuk mengakses tampilan antarmuka *sign up* dapat dilakukan dengan memberikan sekali ketukan pada tombol *sign up* yang ada. Hasil implementasi antarmuka *sign in* dapat dilihat melalui Gambar 5.3.



Gambar 5.3 Tampilan Antarmuka *Sign In*

5.6.3 Implementasi Antarmuka *Contact List Fragment*

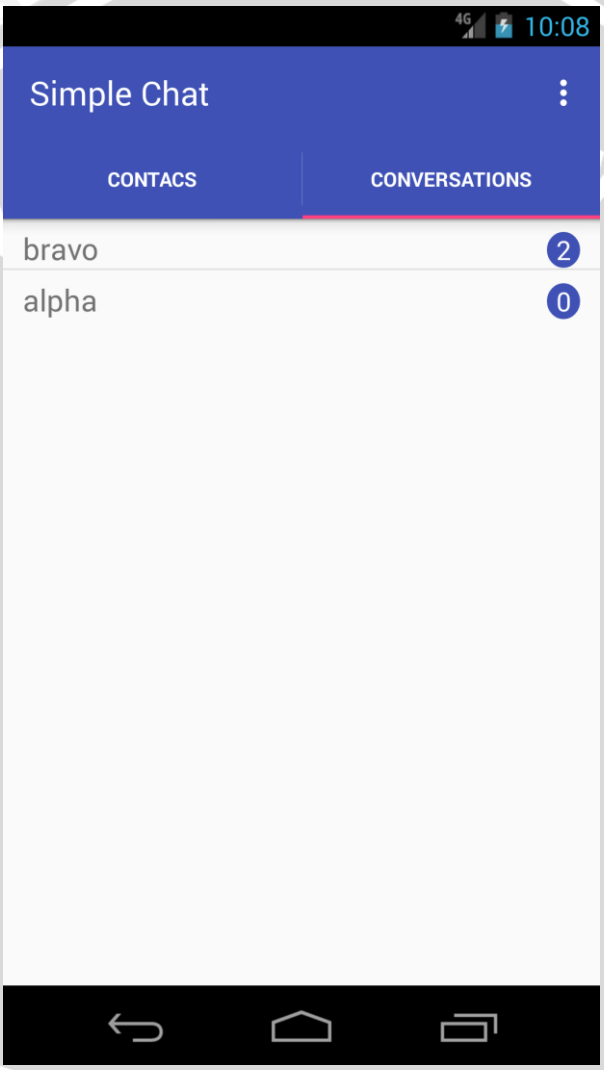
Melalui interkasi dengan tampilan antarmuka *contact list fragment*, *member* mampu melihat daftar kontak yang dimilikinya, menambah kontak baru, menghapus kontak tertentu, serta mengakses tampilan antarmuka *conversation*. Proses menambah kontak dapat dilakukan melalui *form* yang tersedia pada tampilan ini. Sedangkan proses menghapus kontak dapat dilakukan dengan memberikan ketukan panjang terhadap salah satu data kontak yang ada. Serta, untuk mengakses tampilan antarmuka *conversation* dapat dilakukan dengan memberikan sekali ketukan pada data kontak yang ada. Hasil implementasi tampilan antarmuka *contact list fragment* dapat dilihat melalui Gambar 5.4.



Gambar 5.4 Tampilan Antarmuka *Contact List Fragment*

5.6.4 Implementasi Antarmuka *Conversation List Fragment*

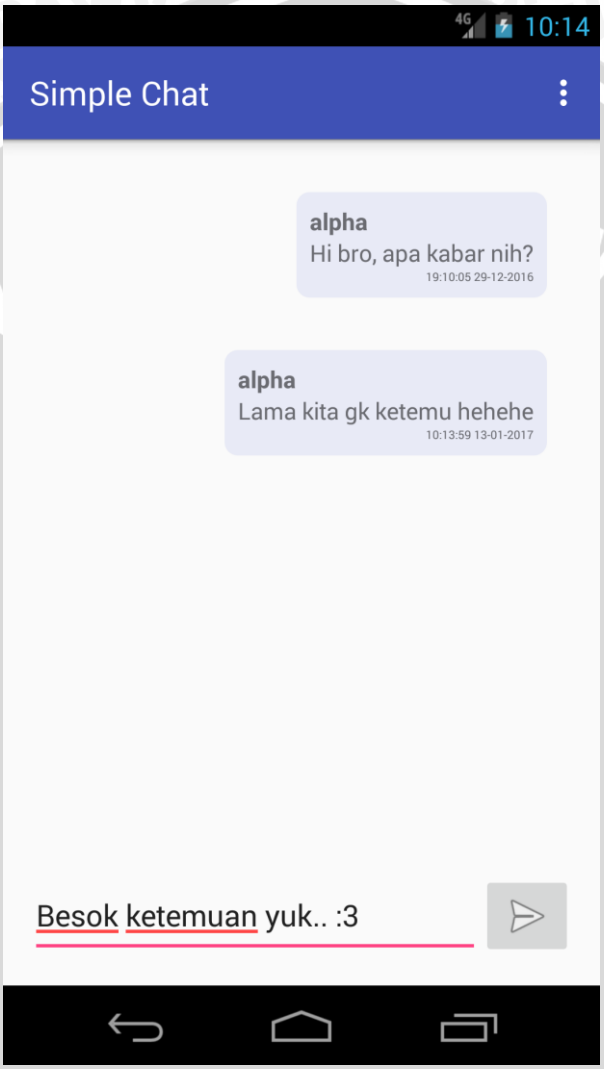
Melalui interaksi dengan tampilan antarmuka *conversation list fragment*, member mampu melihat daftar percakapan yang dimilikinya, menghapus percakapan serta mengakses tampilan antarmuka *conversation*. Proses menghapus percakapan dapat dilakukan dengan memberikan ketukan panjang terhadap salah satu data *conversation* yang ada. Sedangkan untuk mengakses tampilan antarmuka *conversation* dapat dilakukan dengan memberikan sekali ketukan pada data *conversation* yang ada. Hasil implementasi tampilan antarmuka *conversation list fragment* dapat dilihat melalui Gambar 5.5.



Gambar 5.5 Tampilan Antarmuka *Conversation List Fragment*

5.6.5 Implementasi Antarmuka *Conversation*

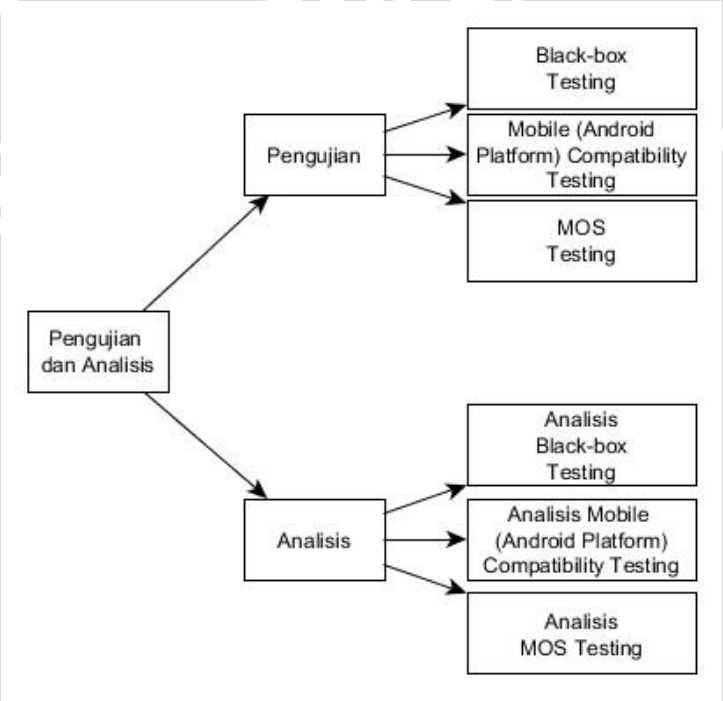
Melalui interaksi dengan tampilan antarmuka *conversation*, member mampu melakukan proses pengiriman pesan serta melihat percakapannya dengan member tertentu. Tampilan antarmuka ini dapat diakses dengan memberikan sekali ketukan terhadap salah satu data *contact* atau *conversation* yang ada. Hasil implementasi tampilan antarmuka *conversation* dapat dilihat melalui Gambar 5.6.



Gambar 5.6 Tampilan Antarmuka *Conversation*

BAB 6 PENGUJIAN DAN ANALISIS

Pada bagian ini dijelaskan proses pengujian dan analisis terhadap aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra pada sistem operasi Android. Proses pengujian dan analisis ini dilakukan dengan tujuan untuk mengetahui tingkat kualitas aplikasi yang telah dibangun. Dalam prosesnya, pengujian ini menggunakan metode pengujian *Black-box Testing, Mobile (Android Platform) Compatibility Testing* dan *Mean Opinion Score* (MOS) *Testing* serta melibatkan sejumlah orang sebagai responden. Struktur pengujian dan analsis tersebut dapat dilihat melalui Gambar 6.1.



Gambar 6.1 Struktur Pengujian dan Analisis Aplikasi *Messaging* Berbasis *Voice Interaction*

6.1 Pengujian

Pada proses pengujian ini terdapat 2 metode pengujian yang digunakan, yaitu metode *Black-box Testing* dan *Mean Opinion Score* (MOS) *Testing.* Masing-masing dari metode ini digunakan untuk menguji dan mengukur tingkat kesesuaian hasil analisa kebutuhan fungsional terhadap hasil implementasi fungsional sistem serta tingkat keakurasian layanan pengenalan suara yang digunakan dalam aplikasi.

6.1.1 *Black-box Testing*

Black-box testing (pengujian kotak hitam) atau yang dikenali pula dengan istilah pengujian perilaku merupakan metode pengujian yang didasarkan pada verifikasi dan validasi kebutuhan fungsional dari sistem perangkat lunak. Proses pengujian dengan menggunakan metode ini diawali dengan perancangan kasus uji yang akan digunakan kemudian diikuti dengan pengujian itu sendiri. Perancangan kasus uji tersebut dilakukan dengan mengacu pada hasil analisa kebutuhan sistem seperti yang ditunjukkan oleh poin 4.1. Daftar kasus uji yang digunakan dalam proses pengujian ini dapat dilihat melalui Tabel 6.1 hingga Tabel 6.17.

Tabel 6.1 Kasus Uji Melakukan Proses Pendaftaran

Nomor Kasus Uji	BBT01
Nama Kasus Uji	Melakukan proses pendaftaran.
Objek Uji	FR01
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan pengguna dapat melakukan pendaftaran akun *member.*
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk tombol *register.* 3. Mengisi seluruh *field* yang terdapat pada *form register.* 4. Mengetuk tombol register.
Hasil yang Diharapkan	Pengguna dapat melakukan proses pendaftar akun *member.*

Tabel 6.2 Kasus Uji Melakukan Proses *Sign In* Melalui GUI

Nomor Kasus Uji	BBT02
Nama Kasus Uji	Melakukan proses *sign in* melalui GUI
Objek Uji	FR02
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan pengguna dapat masuk sebagai *member* ke dalam sistem melalui

	GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengisi seluruh field yang terdapat pada form *sign in.* 3. Mengetuk tombol *sign in.*
Hasil yang Diharapkan	Pengguna dapat masuk sebagai *member* ke dalam sistem melalui GUI.

Tabel 6.3 Kasus Uji Melakukan Proses *Sign In* melalui VUI

Nomor Kasus Uji	BBT03
Nama Kasus Uji	Melakukan proses *sign in* melalui VUI
Objek Uji	FR02
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan pengguna dapat masuk sebagai *member* ke dalam sistem melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.* 3. Memberikan *voice command sign in.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses *sign in.*
Hasil yang Diharapkan	Pengguna dapat masuk sebagai *member* ke dalam sistem melaui VUI.

Tabel 6.4 Kasus Uji Melakukan Proses Menambah Kontak Melalui GUI

Nomor Kasus Uji	BBT04
Nama Kasus Uji	Melakukan proses menambah kontak melalui GUI
Objek Uji	FR05
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat menambahkan *member* lain ke dalam daftar kontaknya melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk *tab contacts.* 3. Mengisi *field* yang terdapat pada *form* penambahan kontak.

	4. Mengetuk tombol *add contact.*
Hasil yang Diharapkan	*Member* dapat menambahkan *member* lain ke dalam daftar kontaknya melalui GUI.

Tabel 6.5 Kasus Uji Melakukan Proses Menambah Kontak Melalui VUI

Nomor Kasus Uji	BBT05
Nama Kasus Uji	Melakukan proses menambah kontak melalui VUI
Objek Uji	FR05
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat menambahkan *member* lain ke dalam daftar kontaknya melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.* 3. Memberikan *voice command add contact.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses menambah kontak.
Hasil yang Diharapkan	*Member* dapat menambahkan *member* lain ke dalam daftar kontaknya melalui VUI.

Tabel 6.6 Kasus Uji Melakukan Proses Menghapus Kontak Melalui GUI

Nomor Kasus Uji	BBT06
Nama Kasus Uji	Melakukan proses menghapus kontak melalui GUI.
Objek Uji	FR06
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk *tab contacts.* 3. Mengetuk panjang salah satu kontak yang ada. 4. Mengetuk tombol *yes* yang terdapat pada jendela *alert dialog.*
Hasil yang Diharapkan	*Member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui GUI.

Tabel 6.7 Kasus Uji Melakukan Proses Menghapus Kontak Melalui VUI

Nomor Kasus Uji	BBT07
Nama Kasus Uji	Melakukan proses menghapus kontak melalui VUI
Objek Uji	FR06
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.* 3. Memberikan *voice command delete contact.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses menghapus kontak.
Hasil yang Diharapkan	*Member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui VUI.

Tabel 6.8 Kasus Uji Melakukan Proses Mengirim Pesan Melalui GUI

Nomor Kasus Uji	BBT08
Nama Kasus Uji	Melakukan proses mengirim pesan melalui GUI.
Objek Uji	FR07
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat mengirimkan pesan ke *member* lain melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk salah satu kontak atau percakapan. 3. Mengisi *field* yang terdapat pada *form* pengiriman pesan. 4. Mengetuk tombol *send message.*
Hasil yang Diharapkan	*Member* dapat mengirimkan pesan ke *member* lain melalui GUI.

Tabel 6.9 Kasus Uji Melakukan Proses Mengirim Pesan Melalui VUI.

Nomor Kasus Uji	BBT09
Nama Kasus Uji	Melakukan proses mengirim pesan melalui VUI.
Objek Uji	FR07
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member*

	dapat mengirimkan pesan ke *member* lain melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.* 3. Memberikan *voice command send message.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses mengirim pesan.
Hasil yang Diharapkan	*Member* dapat mengirimkan pesan ke *member* lain melalui VUI.

Tabel 6.10 Kasus Uji Melihat Pesan Masuk yang Belum Dibaca Melalui GUI

Nomor Kasus Uji	BBT10
Nama Kasus Uji	Melakukan proses melihat pesan masuk yang belum dibaca melalui GUI.
Objek Uji	FR08
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat melihat pesan masuk yang belum dibaca melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk *tab conversations.* 3. Mengetuk data percakapan yang memiliki tanda notifikasi pada daftar percakapan.
Hasil yang Diharapkan	*Member* dapat melihat pesan masuk yang belum dibaca melalui GUI.

Tabel 6.11 Kasus Uji Mendengarkan Pesan Masuk yang Belum Dibaca Melalui VUI.

Nomor Kasus Uji	BBT11
Nama Kasus Uji	Melakukan proses mendengarkan pesan masuk yang belum dibaca melalui VUI.
Objek Uji	FR08
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat mendengarkan pesan masuk yang belum dibaca melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.*

	3. Memberikan *voice command read unread message.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses mendengarkan pesan masuk yang belum dibaca.
Hasil yang Diharapkan	*Member* dapat mendengarkan pesan masuk yang belum dibaca melalui VUI.

Tabel 6.12 Kasus Uji Melihat Percakapan Melalui GUI.

Nomor Kasus Uji	BBT12
Nama Kasus Uji	Melakukan proses melihat percakapan melalui GUI.
Objek Uji	FR09
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat melihat percakapannya dengan *member* lain melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk salah satu data kontak atau data percakapan.
Hasil yang Diharapkan	*Member* dapat melihat percakapannya dengan *member* lain melalui GUI.

Tabel 6.13 Kasus Uji Mendengarkan Percakapan Melalui VUI.

Nomor Kasus Uji	BBT13
Nama Kasus Uji	Melakukan proses mendengarkan percakapan melalui VUI.
Objek Uji	FR09
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat mendengarkan percakapannya dengan *member* lain melalui VUI.
Prosedur Uji	1. Membuka aplikasi 2. Memberikan *voice command wake up.* 3. Memberikan voice command *read conversation.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses mendengarkan percakapan.
Hasil yang Diharapkan	*Member* dapat mendengarkan percakapannya dengan

	member lain melalui VUI.

Tabel 6.14 Kasus Uji Menghapus Percakapan Melalui GUI

Nomor Kasus Uji	BBT14
Nama Kasus Uji	Melakukan proses menghapus percakapan melalui GUI.
Objek Uji	FR10
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat menghapus percakapannya dengan *member* lain melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Mengetuk *tab conversations.* 3. Mengetuk panjang salah satu data percakapan yang terdapat pada daftar percakapan. 4. Mengetuk tombol *yes* yang terdapat pada jendela *alert dialog.*
Hasil yang Diharapkan	*Member* dapat menghapus percakapannya dengan *member* lain melalui GUI.

Tabel 6.15 Kasus Uji Menghapus Percakapan Melalui VUI.

Nomor Kasus Uji	BBT15
Nama Kasus Uji	Melakukan proses menghapus percakapan melalui VUI.
Objek Uji	FR10
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat menghapus percakannya dengan *member* lain melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.* 3. Memberikan *voice command delete conversation.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses menghapus percakapan.
Hasil yang Diharapkan	*Member* dapat menghapus percakapannya dengan *member* lain melalui VUI.

Tabel 6.16 Kasus Uji *Sign Out* melalui GUI.

Nomor Kasus Uji	BBT16
Nama Kasus Uji	Melakukan proses *sign out* melalui GUI.
Objek Uji	FR11
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat keluar dari sistem melalui GUI.
Prosedur Uji	1. Membuka aplikasi. 2. Membuka *options menu.* 3. Mengetuk *options menu item sign out.*
Hasil yang Diharapkan	*Member* dapat keluar sistem melalui GUI.

Tabel 6.17 Kasus Uji *Sign Out* Melalui VUI.

Nomor Kasus Uji	BBT17
Nama Kasus Uji	Melakukan proses *sign out* melalui VUI.
Objek Uji	FR11
Tujuan Pengujian	Memastikan bahwa aplikasi memungkinkan *member* dapat keluar dari sistem melalui VUI.
Prosedur Uji	1. Membuka aplikasi. 2. Memberikan *voice command wake up.* 3. Memberikan *voice command sign out.* 4. Memberikan seluruh data *input* yang diminta pada *voice conversation* proses *sign out.*
Hasil yang Diharapkan	*Member* dapat keluar dari sistem melalui VUI.

6.1.2 *Mobile* (Android *Platform*) *Compatibility Testing*

Mobile (Android *Platform*) *Compatibility Testing* merupakan sebuah metode yang bertujuan untuk mengetahui tingkat ketergantungan suatu aplikasi terhadap perbedaan versi Android *platform* yang menjadi lingkungan *(environment)* dimana aplikasi tersebut dijalankan. Pada pengujian ini dilakukan pengecekan seluruh fungsional aplikasi saat dijalankan pada suatu versi Android *platform* yang menjadi rujukan. Daftar kasus uji yang digunakan dalam proses pengujian ini dapat dilihat melalui Tabel 6.18 hingga Tabel 6.21.

Tabel 6.18 Android 4.1.2 *Compatibility Testing*

Nomor Kasus Uji	ACT01
Nama Kasus Uji	Android 4.1.2 *Compability Testing*

Objek Uji	NFR01
Tujuan Pengujian	Memastikan seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.1.2
Prosedur Pengujian	Mencoba seluruh fitur yang terdapat pada aplikasi.
Hasil yang Diharapkan	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.1.2

Tabel 6.19 Android 4.2.2 *Compatibility Testing*

Nomor Kasus Uji	ACT02
Nama Kasus Uji	Android 4.2.2 *Compability Testing*
Objek Uji	NFR01
Tujuan Pengujian	Memastikan seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.2.2
Prosedur Pengujian	Mencoba seluruh fitur yang terdapat pada aplikasi.
Hasil yang Diharapkan	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.2.2

Tabel 6.20 Android 4.4.2 *Compatibility Testing*

Nomor Kasus Uji	ACT03
Nama Kasus Uji	Android 4.4.2 *Compability Testing*
Objek Uji	NFR01
Tujuan Pengujian	Memastikan seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.4.2
Prosedur Pengujian	Mencoba seluruh fitur yang terdapat pada aplikasi.
Hasil yang Diharapkan	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.4.2

Tabel 6.21 Android 5.0.2 *Compatibility Testing*

Nomor Kasus Uji	ACT04
Nama Kasus Uji	Android 5.0.2 *Compability Testing*
Objek Uji	NFR01
Tujuan Pengujian	Memastikan seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 5.0.2

Prosedur Pengujian	Mencoba seluruh fitur yang terdapat pada aplikasi.
Hasil yang Diharapkan	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 5.0.2

6.1.3 *Mean Opinion Score* (MOS) *Testing*

Mean Opinion Score (MOS) *Testing* merupakan salah satu metode pengujian yang bersifat subjektif. Melalui proses pengujian ini, dilakukan pengukuran tingkat kualitas pengalaman atau *Quality of Experience* (QoE) dari layanan *voice user interface* yang terdapat pada aplikasi. Pada pengujian ini, pengguna akan diminta menggunakan *voice user interface* yang terdapat pada aplikasi dengan minimal penggunaan *sign in, add contact, read unread message* dan *send message voice commands.* Beberapa dari pengguna tersebut diminta untuk berperan sebagai penyandang tunanetra dengan cara melakukan skenario pengujian yang ada dengan kondisi mata tertutup. Kemudian seluruh pengguna akan diminta untuk memberikan nilai atas tingkat kualitas pengalaman mereka saat menggunakan *voice user interface* tersebut. Penilaian yang dilakukan oleh pengguna tersebut dilakukan dengan merujuk pada nilai skala opini seperti yang ditunjukkan oleh Tabel 6.22.

Tabel 6.22 Skala Opini yang Direkomendasikan Oleh International Telecommunication Union

Opinions	Scales
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

Proses pengujian ini melibatkan 10 orang secara acak sebagai respondennya. Hasil pengujian tersebut dapat dilihat melalui Tabel 6.23.

Tabel 6.23 Hasil MOS *Testing*

No.	Responden	Umur	Bahasa Asli	Opinion Score
1	Responden 1	22	Bahasa Indonesia	4
2	Responden 2	22	Bahasa Indonesia	5
3	Responden 3	22	Bahasa Indonesia	4
4	Responden 4	20	Bahasa Indonesia	4
5	Responden 5	20	Bahasa Indonesia	4
6	Responden 6	21	Bahasa Indonesia	4

7	Responden 7	22	Bahasa Indonesia	3
8	Responden 8	22	Bahasa Indonesia	2
9	Responden 9	20	Bahasa Indonesia	4
10	Responden 10	23	Bahasa Indonesia	3

6.2 Analisis

Pada bagian ini diuraikan proses analisis terhadap seluruh pengujian yang telah dilakukan terhadap aplikasi *messaging* berbasis *voice interaction* bagi penyandang tuna netra. Proses analisis tersebut dilakukan berdasarkan hasil pelaksanaan kasus uji yang telah dibuat sebelumnya. Bagian ini terdiri dari analisis *Black-box testing* dan analisis *Mean Opinion Score* (MOS) *testing.*

6.2.1 Analisis *Black-box Testing*

Analisis *Black-box Testing* diperoleh berdasarkan pelaksanaan kasus uji untuk pengujian dengan metode *Black-box Testing* seperti yang ditunjukkan oleh Tabel 6.1 hingga Tabel 6.17. Berdasarkan kasus uji tersebut, dilakukan penyocokan antara hasil yang diharapkan dengan hasil yang didapatkan sehingga dapat diperoleh validitas untuk setiap fungsional aplikasi yang ada. Hasil analisis *Black-box Testing* dapat dilihat melalui Tabel 6.24.

Tabel 6.24 Hasil Analisis *Black-box Testing*

Nomor Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan	Validitas
BBT01	Pengguna dapat melakukan proses pendaftar akun *member.*	Pengguna dapat melakukan proses pendaftar akun *member.*	Valid
BBT02	Pengguna dapat masuk sebagai *member* ke dalam sistem melalui GUI.	Pengguna dapat masuk sebagai *member* ke dalam sistem melalui GUI.	Valid
BBT03	Pengguna dapat masuk sebagai *member* ke dalam sistem melaui VUI.	Pengguna dapat masuk sebagai *member* ke dalam sistem melaui VUI.	Valid
BBT04	*Member* dapat menambahkan *member* lain ke dalam daftar kontaknya melalui GUI.	*Member* dapat menambahkan *member* lain ke dalam daftar kontaknya melalui GUI.	Valid
BBT05	*Member* dapat menambahkan *member* lain ke dalam daftar kontaknya	*Member* dapat menambahkan *member* lain ke dalam daftar kontaknya	Valid

	melalui VUI.	melalui VUI.	
BBT06	*Member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui GUI.	*Member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui GUI.	Valid
BBT07	*Member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui VUI.	*Member* dapat menghapus salah satu kontak yang terdapat pada daftar kontaknya melalui VUI.	Valid
BBT08	*Member* dapat mengirimkan pesan ke *member* lain melalui GUI.	*Member* dapat mengirimkan pesan ke *member* lain melalui GUI.	Valid
BBT09	*Member* dapat mengirimkan pesan ke *member* lain melalui VUI.	*Member* dapat mengirimkan pesan ke *member* lain melalui VUI.	Valid
BBT10	*Member* dapat melihat pesan masuk yang belum dibaca melalui GUI.	*Member* dapat melihat pesan masuk yang belum dibaca melalui GUI.	Valid
BBT11	*Member* dapat mendengarkan pesan masuk yang belum dibaca melalui VUI.	*Member* dapat mendengarkan pesan masuk yang belum dibaca melalui VUI.	Valid
BBT12	*Member* dapat melihat percakapannya dengan *member* lain melalui GUI.	*Member* dapat melihat percakapannya dengan *member* lain melalui GUI.	Valid
BBT13	*Member* dapat mendengarkan percakapannya dengan *member* lain melalui VUI.	*Member* dapat mendengarkan percakapannya dengan *member* lain melalui VUI.	Valid
BBT14	*Member* dapat menghapus percakapannya dengan *member* lain melalui GUI.	*Member* dapat menghapus percakapannya dengan *member* lain melalui GUI.	Valid
BBT15	*Member* dapat menghapus percakapannya dengan *member* lain melalui VUI.	*Member* dapat menghapus percakapannya dengan *member* lain melalui VUI.	Valid
BBT16	*Member* dapat keluar sistem melalui GUI.	*Member* dapat keluar sistem melalui GUI.	Valid
BBT17	*Member* dapat keluar dari sistem melalui VUI.	*Member* dapat keluar dari sistem melalui VUI.	Valid

6.2.2 Analisis *Mobile* (Android *Platform*) *Compatibility Testing*

Analisis *Mobile* (Android *Platform*) *Compatibility Testing* diperoleh berdasarkan pelaksanaan kasus uji untuk pengujian dengan metode Android *Compatibility Testing* seperti yang ditunjukkan oleh Tabel 6.18 hingga Tabel 6.21. Berdasarkan kasus uji tersebut, dilakukan penyocokan antara hasil yang diharapkan dengan hasil yang didapatkan. Sehingga, melalui hal tersebut dapat diperoleh kompatibilitas untuk setiap versi Android yang dijadikan rujukan. Hasil analisis Android *Compatibility Testing* dapat dilihat melalui Tabel 6.25.

Tabel 6.25 Hasil Analisis *Mobile* (Android *Platform*) *Compatibility Testing*

Nomor Kasus Uji	Hasil Yang Diharapkan	Hasil Yang Didapatkan	Kompatibilitas
ACT01	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.1.2	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.1.2	Kompatibel
ACT02	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.2.2	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.2.2	Kompatibel
ACT03	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.4.2	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 4.4.2	Kompatibel
ACT04	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 5.0.2	Seluruh fitur yang dimiliki aplikasi dapat bekerja dengan baik pada sistem operasi Android 5.0.2	Kompatibel

6.2.3 Analisis *Mean Opinion Score* (MOS) *Testing*

Analisis *Mean Opinion Score (MOS) Testing* diperoleh berdasarkan perhitungan rata-rata nilai opini yang diberikan oleh pengguna. Untuk mendapatkan rata-rata nilai opini *(MOS)* terkait penggunaan layanan *voice interaction* yang terdapat pada aplikasi, dapat dilakukan melalui penggunaan rumus:

$$MOS = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Melalui penggunaan rumus sebelumnya, maka diperoleh:

$$MOS = \frac{4+5+4+4+4+4+3+2+4+3}{10}$$

$$MOS = \frac{37}{10}$$

$$MOS = 3{,}7$$

Berdasarkan proses analisis *Mean Opinion Score (MOS) Testing,* diperoleh nilai *MOS* dari layanan *voice interaction* sebesar 3,7. Hal ini menunjukkan bahwa rata-rata pengguna menyatakan layanan *voice interaction* yang terdapat pada aplikasi memiliki fungsi yang baik (*Good*).

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil dari seluruh proses yang terdapat pada penelitian ini, diperoleh kesimpulan sebagai berikut:

1. Aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra pada sistem operasi Android berhasil dirancang melalui perancangan *sequence diagram, class diagram*, basis data, antarmuka dan *page flow.* Proses perancangan tersebut dilakukan dengan mengacu terhadap hasil analisis kebutuhan sistem yang telah dilakukan sebelumnya.
2. Aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra pada sistem operasi *Android berhasil* diimplementasikan dengan menggunakan Java *programming language* dan XML *markup language.* Proses implementasi tersebut dilakukan dengan memanfaatkan aplikasi *Android Studio* serta pustaka Android *Speech-To-Text* dan Android *Text-To-Speech.*
3. Berdasarkan pengujian MOS (*Mean Opinion Score),* diperoleh hasil MOS dari layanan *voice interaction* sebesar 3,7. Hal ini menunjukkan bahwa rata-rata pengguna menyatakan bahwa layanan *voice interaction* yang terdapat pada aplikasi memiliki kualitas yang baik (*Good*).

7.2 Saran

Setelah melalui seluruh proses yang terdapat pada penelitian ini, maka saran yang dapat diberikan bagi pengembangan sistem ataupun penelitian sejenis yang bersifat lanjutan adalah sebagai berikut:

1. Pengembangan aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra pada sistem operasi Android dapat dilakukan dengan menambahkan fungsi pengejaan *voice input* menggunakan *International Radiotelephony Spelling Alphabet* atau ICAO *phonetic alhpabet.* Hal ini dirasa perlu untuk meningkatkan fleksibilitas *voice input* yang dapat diberikan oleh pengguna.
2. Pengembangan aplikasi *messaging* berbasis *voice interaction* bagi penyandang tunanetra pada sistem operasi *Android* dapat dilakukan dengan mengimplementasikan *deep learning* pada *voice interaction design* aplikasi. Hal ini dirasa perlu untuk meningkatkan fleksibilitas *voice conversation* yang dapat diakomodir oleh aplikasi.

DAFTAR PUSTAKA

Android Developer, 2009. *An Introduction to Text-To-Speech in Android.* [Online] Available at: http://android-developers.blogspot.co.id/2009/09/introduction-to-text-to-speech-in.html [Accessed 3 August 2016].

Android Developer, 2015. *SpeechRecognizer.* [Online] Available at: https://developer.android.com/reference/android/speech/SpeechRecognizer.html#startListening(android.content.Intent) [Accessed 30 December 2016].

Barus, A. C., Hutasoit, D. I. P., Siringoringo, J. H. & Siahaan, Y. A., 2015. *White Box Testing Tool Prototype Development.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7352537 [Accessed 20 February 2017].

Charon Int Trading Ltd, 2014. *The History Of Communication From Smoke Signals To Smartphones.* [Online] Available at: http://www.thesnugg.com/history-of-communication.aspx [Accessed 30 January 2017].

Dagba, T. K. & Boco, C., 2014. *A Text To Speech System for Fon Language Using Multisyn Algorithm.* [Online] Available at: http://www.sciencedirect.com/science/article/pii/S1877050914010904 [Accessed 20 February 2017].

Dobie, A., Holly, R. & Hildenbrand, J., 2015. *Android's Early Days.* [Online] Available at: http://www.androidcentral.com/androids-early-days [Accessed 11 February 2016].

Garibay, F. R., Olivarria, C. M., Aguilera, A. F. E. & Huegel, J. C., 2014. *MyVox - Device For The Communication Between People: Blind, Deaf, Deaf-Blind And Unimpaired.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6970330 [Accessed 2017 February 20].

Google, 2016. *Firebase.* [Online] Available at: https://firebase.google.com/ [Accessed 3 January 2017].

Google, 2017. *Add Firebase To Your Android Project.* [Online] Available at: https://firebase.google.com/docs/android/setup [Accessed 1 February 2017].

Google, 2017. *Firebase Authentication.* [Online] Available at: https://firebase.google.com/docs/auth/ [Accessed 1 February 2017].

Google, 2017. *Firebase Realtime Database.* [Online] Available at: https://firebase.google.com/docs/database/ [Accessed 1 February 2017].

Google, 2017. *Set Up Firebase Authentication For Android.* [Online] Available at: https://firebase.google.com/docs/auth/android/start/ [Accessed 1 February 2017].

International Telecommunication Union, 1996. *P.800: Method For Subjective Determination Of Transmission Quality.* [Online] Available at: https://www.itu.int/rec/T-REC-P.800-199608-I/en [Accessed 1 February 2017].

International Telecommunication Union, 1998. *P.911: Subjective Audiovisual Quality Assessment Methods for Multimedia Applications.* [Online] Available at: https://www.itu.int/rec/T-REC-P.911-199812-I/en [Accessed 16 5 2017].

Kapur, P. K., Singh, G., Sachdeva, N. & Tickoo, A., 2014. *Measuring Software Testing Efficiency Using Two-Way Assessment Technique.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7014679 [Accessed 20 February 2017].

Kementrian Kesehatan RI, 2014. *Buletin Jendela Data dan Informasi Kesehatan Situasi Penyandang Disabilitas.* [Online] Available at: http://www.depkes.go.id/download.php?file=download/pusdatin/buletin/buletin-disabilitas.pdf [Accessed 20 February 2017].

Lane, R., 2013. *Touching upon a vision.* [Online] Available at: http://www.rolexawards.com/profiles/young_laureates/sumit_dagar/project [Accessed 31 January 2017].

McGraw, I. et al., 2016. *Personalized Speech Recognition On Mobile Devices.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7472820 [Accessed 17 5 2017].

Meng, J., Zhang, J. & Zhao, H., 2012. *Overview of the Speech Recognition Technology.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6300437 [Accessed 20 February 2017].

Merdeka.com, 2012. *Jumlah Tunanetra di Indonesia Setara dengan Penduduk Singapura.* [Online] Available at: http://www.merdeka.com/peristiwa/jumlah-tunanetra-di-indonesia-setara-dengan-penduduk-singapura.html [Accessed 4 February 2016].

Merdeka.com, 2015. *Ini Persentase Pengguna Android dan iOS di Dunia, Lebih Besar Siapa?.* [Online] Available at: http://www.merdeka.com/teknologi/ini-persentase-pengguna-android-dan-ios-di-dunia-lebih-besar-siapa.html [Accessed 11 February 2016].

Mitroff, S. & Dolcourt, J., 2014. *The Android Era: From G1 to Lollipop.* [Online] Available at: http://www.cnet.com/news/history-of-android [Accessed 16 February 2016].

Neira, E. M., 2015. *IEEE COMSOC CTN Special Issue On Ten Trends That Tell Where Communication Technology Are Headed In 2015.* [Online] Available at: http://www.comsoc.org/ctn/ieee-comsoc-ctn-special-issue-ten-trends-tell-where-communication-technologies-are-headed-2015 [Accessed 30 January 2017].

Neto, R. & Fonseca, N., 2014. *Camera Reading For Blind People.* [Online] Available at: http://www.sciencedirect.com/science/article/pii/S2212017314003624 [Accessed 20 February 2017].

Pressman, R., 2012. *Rekayasa Perangkat Lunak: Pendekatan Praktisi.* 7th ed. Yogyakarta: Penerbit Andi.

Roscoe, J., 1975. *Fundamental Research Statistics for The Behavioural Science.* 2nd Edition ed. New York: Holt Rinehart & Winston.

Tamplin, J., 2016. *Firebase Expands To Become A Unified App Platform.* [Online] Available at: https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html [Accessed 1 February 2017].

United Nations, 2006. *United Nations Convention on the Rights of Persons with Disabilities.* [Online] Available at: http://www.ohchr.org/EN/HRBodies/CRPD/Pages/ConventionRightsPersonsWithDisabilities.aspx [Accessed 20 February 2017].

Xu, J., Xing, L., Perkis, A. & Jiang, Y., 2011. *On the Properties of Mean Opinion Scores for Quality of Experience Management.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6123396 [Accessed 20 February 2017].

Zhang, T., Gao, J., Cheng, J. & Uehara, T., 2015. *Compatibility Testing Service For Mobile Applications.* [Online] Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7133527 [Accessed 20 February 2017].