PENGEMBANGAN APLIKASI IDENTIFIKASI KESALAHAN BACA ALQURAN MENGGUNAKAN *SPEECH RECOGNIZER* DENGAN METODE *LEVENSTEIN DISTANCE* PADA PLATFORM ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan   
memperoleh gelar Sarjana Komputer

Disusun oleh:

Galang Putratama

NIM: 155150201111060



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2019

PENGESAHAN

PENGEMBANGAN APLIKASI IDENTIFIKASI KESALAHAN BACA ALQURAN MENGGUNAKAN *SPEECH RECOGNIZER* DENGAN METODE *LEVENSTEIN DISTANCE* PADA PLATFORM ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun Oleh :

Galang Putratama

NIM: 155150201111060

Skripsi ini telah diuji dan dinyatakan lulus pada

2 Januari 2019

Telah diperiksa dan disetujui oleh:

|  |  |
| --- | --- |
| Dosen Pembimbing 1  Agi Putra Kharisma S.T, M.T  NIK: 201304 8604301 001 | Dosen Pembimbing 2  Lutfi Fanani S.Kom., M.T., M.Sc  NIK: 201607 890217 1 001 |

Mengetahui

Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 4 Januari 2019

­

Galang Putratama

NIM: 155150201111060

PRAKATA

*Assalamu’alaikum Wr. Wb.*

Pertama-tama kami panjatkan puji syukur kehadirat Allah SWT yang telah memberikan rahmat, taufik dan hidayahnya sehingga penulis dapat menyelesaikan skripsi ini dengan baik. Adapun dokumen skripsi yang penulis buat ini berjudul “Pengembangan Aplikasi Identifikasi Kesalahan Baca Alquran Menggunakan *Speech Recognizer* dengan Metode *Levenstein Distance* pada Platform Android*”.*

Sejak awal sampai dengan akhir penulisan ini, tidak sedikit bantuan yang kami terima dan karenanya dalam kesempatan ini kami ingin menyampaikan terima kasih kepada:

1. Allah S.W.T. atas limpahan berkah dan hidayah serta bimbingan-Nya dalam pelaksanaan praktek kerja lapangan.
2. Bapak Agi Putra Kharisma, S.T, M.T. selaku dosen pembimbing 1 yang telah menyediakan waktu untuk berdiskusi dan berbagai saran serta masukkannya.
3. Bapak Lutfi Fanani S.Kom., M.T., M.Sc. selaku dosen pembimbing 2 yang telah menyediakan waktu untuk berdiskusi dan berbagai saran serta masukkannya.
4. Bapak Agus Wahyu Widodo, S.T, M.Cs. selaku Ketua Program Studi Informatika / Ilmu Komputer Fakultas Ilmu Komputer Universitas Brawijaya
5. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
6. Orangtua dan seluruh keluarga atas segenap dukungan yang diberikan.
7. Segenap civitas akademika Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya.

Demi kesempurnaan skripsi ini, penulis mengharapkan kritik dan saran yang bersifat membangun. Dan harapan penulis, dokumen skripsi ini dapat bermanfaat bagi semua pihak.

*Wassalamu’alaikum Wr. Wb.*

Malang, 4 Januari 2019

Penulis

galang.ptrm@gmail.com

ABSTRAK

Galang Putratama, Pengembangan Aplikasi Identifikasi Kesalahan Baca Alquran Menggunakan *Speech Recognizer* Dengan Metode Levenstein Distance Pada Platform Android

Pembimbing: Agi Putra Kharisma dan Lutfi Fanani

Alquran adalah kitab suci umat Islam yang harus dibaca dengan memperhatikan kebenaran pengucapan huruf. Karena kesalahan dalam membaca Alquran berakibat pada rusaknya makna dari Alquran itu sendiri. Proses belajar membaca Alquran membutuhkan seorang pembimbing untuk memperbaiki kesalahan yang mungkin terjadi. Namun, dengan keterbatasan waktu, maka terbatas juga kesempatan seseorang untuk berlatih membaca Alquran dengan pembimbing. Salah satu cara yang penulis tawarkan adalah melalui media aplikasi Android untuk mendeteksi kesalahan baca ayat Alquran. Dalam pengembangan aplikasi digunakan metode *Waterfall* yang di awali dari wawancara dengan seorang imam masjid dan 2 orang mahasiswa. Kemudian aplikasi dirancang dengan metode *Object Oriented Design* dan diimplementasikan dengan metode *Object Oriented Programming*. Hasil implementasinya adalah aplikasi memanfaatkan *Speech Recognizer* untuk menerjemahkan bacaan pengguna menjadi teks Bahasa Arab. Sementara algoritme *Levenstein Distance* dan *library Diff Match Patch* digunakan untuk mendeteksi kesalahan baca pengguna. Pada tahap pengujian akurasi ditemukan bahwa Aplikasi ini berhasil menyelesaikan 3 dari 4 kasus pengujian. Sementara pada tahap pengujian *usability* dengan metode *System Usability Scale* aplikasi ini mendapatkan skor 76 dengan tingkat kepuasan B+ atau di atas rata-rata. Aplikasi harus dikembangkan lebih lanjut karena tingkat akurasi yang didapatkan masih terlalu rendah.

Kata kunci : Alquran, Aplikasi, Android, Bahasa Arab, *Speech Recognizer, Levenstein Distance*

ABSTRACT

Galang Putratama, *Development of Application for Error Detection in Koran Recitation using Speech Recognizer and Levenstein Distance Algorithm on Android Platform*

*Supervisors*: Agi Putra Kharisma *and* Lutfi Fanani

*Alquran is Muslims holy book which have to be read by paying attention to its letters pronunciation. Because even a little mistake in pronunciation resulting in damage to the meaning of Alquran itself. The process of learning to read the Koran requires a guide to correct mistakes that might occur. However, due the limited of time, then there is also limited opportunity for someone to practice reading Koran with a guide. One solution that the author offers is through the media based on Android application to detect errors in reading the verses of Koran. In the development of the application, the Waterfall method was used which began with interviews with a mosque imam and 2 students. Then the application is designed with the Object Oriented Design method and implemented with the Object Oriented Programming method. The result of its implementation is that the application utilizes Speech Recognizer to translate user readings into Arabic text. While Levenstein Distance algoritm and Diff Match Patch Library handle the error detections in user’s pronunciation. At the accuracy testing stage it was found that this application successfully completed 3 of the 4 test cases performed. While at the usability testing stage with the System Usability Scale method this application gets a score of 76 with a level of satisfaction B + or above average. The application must be further developed because the level of accuracy obtained is still too low.*

*Keywords : Alquran, Mobile Application, Android, Arabic Language, Speech Recognizer, Levenstein Distance*

DAFTAR ISI

[PENGESAHAN ii](#_Toc534370113)

[PERNYATAAN ORISINALITAS iii](#_Toc534370114)

[PRAKATA iv](#_Toc534370115)

[ABSTRAK v](#_Toc534370116)

[ABSTRACT vi](#_Toc534370117)

[DAFTAR ISI vii](#_Toc534370118)

[DAFTAR TABEL x](#_Toc534370119)

[DAFTAR GAMBAR xi](#_Toc534370120)

[DAFTAR LAMPIRAN xiii](#_Toc534370121)

[BAB 1 PENDAHULUAN 1](#_Toc534370122)

[1.1 Latar belakang 1](#_Toc534370123)

[1.2 Rumusan masalah 2](#_Toc534370124)

[1.3 Tujuan 3](#_Toc534370125)

[1.4 Manfaat 3](#_Toc534370126)

[1.5 Batasan Masalah 3](#_Toc534370127)

[1.6 Sistematika Pembahasan 4](#_Toc534370128)

[BAB 2 LANDASAN KEPUSTAKAAN 5](#_Toc534370129)

[2.1 Kajian Pustaka 5](#_Toc534370130)

[2.2 Alquran 5](#_Toc534370131)

[2.3 *API* Alquran 6](#_Toc534370132)

[2.4 Android 6](#_Toc534370133)

[2.4.1 Android Studio 6](#_Toc534370134)

[*2.4.2* Sensor *Microphone* 6](#_Toc534370135)

[2.5 *Speech Recognizer* 7](#_Toc534370136)

[2.6 Metode *Levenshtein Distance* 9](#_Toc534370137)

[2.7 *Library Diff match patch* 10](#_Toc534370138)

[2.8 Wawancara 11](#_Toc534370139)

[2.9 *Validation Testing* 11](#_Toc534370140)

[2.10 Pengujian *Usability* 12](#_Toc534370141)

[2.11 Pengujian Akurasi 12](#_Toc534370142)

[BAB 3 METODOLOGI 13](#_Toc534370143)

[3.1 Studi Literatur 14](#_Toc534370144)

[3.2 Analisis Kebutuhan 14](#_Toc534370145)

[3.3 Perancangan 15](#_Toc534370146)

[3.4 Implementasi 15](#_Toc534370147)

[3.5 Pengujian dan Analisis Hasil 15](#_Toc534370148)

[*3.5.1 Validation Testing* 15](#_Toc534370149)

[*3.5.2* Pengujian *Usability* 15](#_Toc534370150)

[*3.5.3* Pengujian Akurasi 16](#_Toc534370151)

[3.6 Kesimpulan dan Saran 16](#_Toc534370152)

[BAB 4 ANALISIS KEBUTUHAN 17](#_Toc534370153)

[4.1 Gambaran Umum Aplikasi 17](#_Toc534370154)

[4.2 Identifikasi Aktor 18](#_Toc534370155)

[4.3 Elisitasi Kebutuhan 18](#_Toc534370156)

[4.4 Penomoran Kode Kebutuhan 19](#_Toc534370157)

[4.5 Analisis Kebutuhan 19](#_Toc534370158)

[*4.5.1* Kebutuhan Fungsional 19](#_Toc534370159)

[*4.5.2* Kebutuhan Non Fungsional 20](#_Toc534370160)

[*4.5.3 Usecase Diagram* 20](#_Toc534370161)

[*4.5.4 Usecase Scenario* 21](#_Toc534370162)

[BAB 5 PERANCANGAN 23](#_Toc534370163)

[5.1 *Activity Diagram* 23](#_Toc534370164)

[*5.1.1 Activity Diagram* Melihat Daftar Surat 23](#_Toc534370165)

[*5.1.2 Activity Diagram* Melihat Daftar Ayat 24](#_Toc534370166)

[*5.1.3 Activity Diagram* Melihat Kesalahan Baca 25](#_Toc534370167)

[5.2 *Sequence diagram* 26](#_Toc534370168)

[5.2.1 *Sequence diagram* Melihat Daftar Surat 27](#_Toc534370169)

[5.2.2 *Sequence diagram* Melihat Daftar Ayat 28](#_Toc534370170)

[*5.2.3 Sequence diagram* Melihat Kesalahan Baca 29](#_Toc534370171)

[5.3 *Class Diagram* 30](#_Toc534370172)

[5.4 Perancangan Antarmuka 31](#_Toc534370173)

[*5.4.1* Perancangan Antarmuka Melihat Daftar Surat 31](#_Toc534370174)

[5.4.2 Perancangan Antarmuka Melihat Daftar Ayat 32](#_Toc534370175)

[*5.4.3* Perancangan Antarmuka Melihat Kesalahan Baca 33](#_Toc534370176)

[*5.4.4* Perancangan *Screenflow* Aplikasi 34](#_Toc534370177)

[BAB 6 IMPLEMENTASI 35](#_Toc534370178)

[6.1 Spesifikasi Lingkungan Aplikasi 35](#_Toc534370179)

[6.1.1 Spesifikasi Perangkat Keras 35](#_Toc534370180)

[6.1.2 Spesifikasi Perangkat Lunak 36](#_Toc534370181)

[6.2 Batasan Implementasi 36](#_Toc534370182)

[6.3 Implementasi Kode Program 36](#_Toc534370183)

[6.3.1 Kode Program Melihat Daftar Surat 36](#_Toc534370184)

[6.3.2 Kode Program Melihat Daftar Ayat 39](#_Toc534370185)

[6.3.3 Kode Program Melihat Kesalahan Baca 42](#_Toc534370186)

[6.4 Implementasi Antarmuka 44](#_Toc534370187)

[6.4.1 Implementasi Antarmuka Melihat Daftar Surat 44](#_Toc534370188)

[6.4.2 Implementasi Antarmuka Melihat Daftar Ayat 45](#_Toc534370189)

[6.4.3 Implementasi Antarmuka Melihat Kesalahan Baca 46](#_Toc534370190)

[BAB 7 PENGUJIAN DAN ANALISIS HASIL 47](#_Toc534370191)

[7.1 *Validation Testing* 47](#_Toc534370192)

[7.1.1 Kasus Uji Melihat Daftar Surat 47](#_Toc534370193)

[7.1.2 Kasus Uji Melihat Daftar Ayat 48](#_Toc534370194)

[7.1.3 Kasus Uji Melihat Kesalahan Baca 48](#_Toc534370195)

[7.1.4 Analisis Hasil *Validation Testing* 49](#_Toc534370196)

[7.2 Pengujian *Usability* 49](#_Toc534370197)

[*7.2.1* Analisis Hasil Kuisioner *System Usability Scale* 50](#_Toc534370198)

[*7.2.2* Analisis Hasil Wawancara 52](#_Toc534370199)

[7.3 Pengujian Akurasi 52](#_Toc534370200)

[*7.3.1* Analisis Hasil Pengujian Akurasi 53](#_Toc534370201)

[BAB 8 PENUTUP 55](#_Toc534370202)

[8.1 Kesimpulan 55](#_Toc534370203)

[8.2 Saran 56](#_Toc534370204)

[DAFTAR PUSTAKA 57](#_Toc534370205)

[LAMPIRAN 59](#_Toc534370206)

DAFTAR TABEL

SKRIPSI i

Tabel ‎4.1 Identifikasi Aktor 18

Tabel ‎4.2 Penomoran Kode Kebutuhan 19

Tabel ‎4.3 Kebutuhan Fungsional 20

Tabel ‎4.4 Kebutuhan Non Fungsional 20

Tabel ‎4.5 *Usecase Scenario* Melihat Daftar Surat 21

Tabel ‎4.6 *Usecase Scenario* Melihat Daftar Ayat 22

Tabel ‎4.7 *Usecase Scenario* Melihat Kesalahan Baca 22

Tabel ‎5.1 Objek *Sequence diagram* 26

Tabel ‎5.2 Keterangan Antarmuka Melihat Daftar Surat 31

Tabel ‎5.3 Keterangan Antarmuka Melihat Daftar Ayat 32

Tabel ‎5.4 Keterangan Antarmuka Melihat Kesalahan Baca 33

Tabel ‎6.1 Spesifikasi Perangkat Keras Komputer 35

Tabel ‎6.2 Spesifikasi Perangkat Keras Perangkat Bergerak 35

Tabel ‎6.3 Spesifikasi Perangkat Lunak 36

Tabel ‎7.1 Kasus Uji Melihat Daftar Surat 47

Tabel ‎7.2 Kasus Uji Melihat Daftar Ayat 48

Tabel ‎7.3 Kasus Uji Melihat Kesalahan Baca 48

Tabel ‎7.4 Rekapitulasi Hasil *Validation Testing* 49

Tabel ‎7.5 Pernyataan *System Usability Scale* 50

Tabel ‎7.6 Rekapitulasi Skor *System Usability Scale* 51

Tabel ‎7.7 Klasifikasi Pengujian Akurasi 52

Tabel ‎7.8 Validasi Hasil Pengujian Akurasi 53

Tabel ‎7.9 Rekapitulasi Hasil Pengujian Akurasi 54

Tabel ‎8.1 Rekapitulasi Hasil Wawancara 59

DAFTAR GAMBAR

[Gambar ‎2.1 Contoh *Condensor Microphone* 7](#_Toc534370338)

[Gambar ‎2.2 Contoh *Digital Sampling* 7](#_Toc534370339)

[Gambar ‎2.3 Contoh *Markov Model* 8](#_Toc534370340)

[Gambar ‎2.5 Contoh Klasifikasi Pengujian *Equivalence Partitioning Testing* 12](#_Toc534370341)

[Gambar ‎3.1 Diagram Alir Metodologi Penelitian 13](#_Toc534370342)

[Gambar ‎4.1 Gambaran Umum Aplikasi 17](#_Toc534370343)

[Gambar ‎4.2 *Usecase Diagram* 21](#_Toc534370344)

[Gambar ‎5.1 *Activity Diagram* Melihat Daftar Surat 23](#_Toc534370345)

[Gambar ‎5.2 *Activity Diagram* Melihat Daftar Ayat 24](#_Toc534370346)

[Gambar ‎5.3 *Activity Diagram* Melihat Kesalahan Baca 25](#_Toc534370347)

[Gambar ‎5.4 *Sequence diagram* Melihat Daftar Surat 27](#_Toc534370348)

[Gambar ‎5.5 *Sequence diagram* Melihat Daftar Ayat 28](#_Toc534370349)

[Gambar ‎5.6 *Sequence diagram* Melihat Kesalahan Baca 29](#_Toc534370350)

[Gambar ‎5.7 *Class Diagram* Aplikasi 30](#_Toc534370351)

[Gambar ‎5.8 *Wireframe* Antarmuka Melihat Daftar Surat 31](#_Toc534370352)

[Gambar ‎5.9 *Wireframe* Antarmuka Melihat Daftar ayat 32](#_Toc534370353)

[Gambar ‎5.10 *Wireframe* Antarmuka Melihat Kesalahan Baca 33](#_Toc534370354)

[Gambar ‎5.11 *Screenflow* Aplikasi 34](#_Toc534370355)

[Gambar ‎6.1 Kode Program PilihSurat*Activity*.kt 37](#_Toc534370356)

[Gambar ‎6.2 Kode Program PilihSurat*Adapter*.kt 37](#_Toc534370357)

[Gambar ‎6.3 Kode Program Surat.kt 38](#_Toc534370358)

[Gambar ‎6.4 Kode Program PilihAyat*Activity*.kt 39](#_Toc534370359)

[Gambar ‎6.5 Kode Program PilihAyat*Adapter*.kt 40](#_Toc534370360)

[Gambar ‎6.6 Kode Program AyatDiakritikList.kt 41](#_Toc534370361)

[Gambar ‎6.7 Kode Program Tahsin*Activity*.kt 42](#_Toc534370362)

[Gambar ‎6.8 Kode Program diff\_match\_patch.kt 43](#_Toc534370363)

[Gambar ‎6.9 Antarmuka Melihat Daftar Surat 44](#_Toc534370364)

[Gambar ‎6.10 Antarmuka Melihat Daftar Ayat 45](#_Toc534370365)

[Gambar ‎6.11 Antarmuka Menampilkan Kesalahan Baca 46](#_Toc534370366)

[Gambar ‎7.1 *System Usability Scale Letter Grade* 51](#_Toc534370367)

DAFTAR LAMPIRAN

[Lampiran 1 - Rekapitulasi Hasil Wawancara Pengujian Usability 59](#_Toc532456264)

# PENDAHULUAN

## Latar belakang

Alquran adalah kitab suci umat Islam yang berisi firman Allah swt dan diturunkan kepada Nabi Muhammad saw (KBBI, 2018). Membaca Alquran merupakan salah satu ibadah yang paling utama baik mengerti artinya maupun tidak. Perintah untuk membaca Alquran sendiri merupakan bagian dari wahyu pertama yang diterima Nabi Muhammad saw untuk menekankan seberapa pentingnya (Shihab, 1994). Dalam konteks membaca Alquran, Alquran harus dibaca dengan tartil (perlahan-lahan) yaitu dengan cara membaguskan pengucapan hurufnya dan mengetahui tempat-tempat berhenti (pemenggalan kata/waqof). Hal tersebut dikarenakan jika terjadi kesalahan dalam pengucapan huruf atau pemenggalan kata, dapat berakibat kepada rusaknya susunan bacaan Alquran atau berubahnya makna bacaan dari yang diinginkan Allah swt (Salim, 2015).

Proses belajar membaca Alquran tidak bisa dilakukan sendiri, karena membutuhkan seorang pembimbing untuk mempertanggungjawabkan kebenaran apa yang diajarkan dan memperbaiki kesalahan pembacaan yang mungkin terjadi. Maka tidak ada alasan yang membenarkan seorang mukmin mempelajari bacaan Alquran secara otodidak tanpa seorang pembimbing (Salim, 2015). Waktu yang bisa diluangkan seseorang untuk belajar membaca Alquran dengan guru pembimbing cukup terbatas. Dengan terbatasnya waktu untuk bertemu guru pembimbing, maka terbatas juga kesempatan seseorang untuk melatih bacaan Alquran mereka. Dengan demikian, maka masalah yang hendak diselesaikan dari penelitian ini adalah bagaimana caranya memfasilitasi setiap orang yang ingin melatih bacaan Alquran mereka ketika tidak bersama guru pembimbing. Salah satu solusi yang penulis tawarkan adalah melalui media aplikasi berbasis Android untuk mendeteksi kesalahan baca ayat Alquran.

Android sendiri adalah sebuah sistem operasi terbuka yang dibuat dan dirancang untuk berjalan pada perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet (Azzam, 2017). Karena sifatnya yang terbuka, Android lebih mudah dikembangkan oleh banyak orang. Android menyediakan akses yang sangat luas kepada penulis untuk menggunakan *library* dan *tools* yang dapat digunakan untuk membangun aplikasi. Oleh karena itu, penulis memilih Android untuk dijadikan platform pengembangan aplikasi untuk mendeteksi kesalahan baca ayat Alquran.

Sudah ada beberapa aplikasi di Android untuk belajar membaca Alquran, diantaranya adalah Aqra. Aqra digunakan untuk mengkoreksi bacaan Alquran pengguna menggunakan rekaman suara. Kekurangan pada Aplikasi Aqra adalah tidak bisa memberitahu pengguna di bagian mana letak kesalahan pembacaan Alquran, karena tampilan yang diberikan hanya berupa tulisan ‘benar’ apabila bacaan Alquran pengguna benar dan ‘salah’ apabila terdapat kesalahan pembacaan oleh pengguna. Kemudian kekurangan berikutnya adalah pengguna tidak bisa memilih ayat Alquran, pengguna hanya bisa membaca ayat Alquran yang ditampilkan secara acak pada tampilan aplikasi. Dan saat ini aplikasi Aqra hanya mempunyai tiga surat Alquran yakni An Nas, Al Falaq, dan Al Ikhlash (Wahyudiyanta, 2016). Oleh karena itu aplikasi yang ingin dikembangkan penulis adalah aplikasi yang dapat mendeteksi dan memberitahu pengguna di mana letak kesalahan pembacaan Alquran yang dilakukan oleh pengguna. Dan memberikan pengguna keluasan untuk memilih ayat mana yang ingin dibaca.

*Speech Recognizer* merupakan layanan dari Platform Android yang berfungsi untuk menerjemahkan suara manusia menjadi sebuah teks (Google, 2018). *Speech Recognizer* penulis pilih karena sudah mendukung penerjemahan suara ke teks Bahasa Arab dan memiliki dokumentasi API yang cukup baik. Sementara Metode *Levenstein Distance* adalah algoritme untuk mengukur perbedaan antara dua buah *string* dengan cara menentukan jumlah minimum operasi tambah huruf, hapus huruf ataupun tukar huruf untuk merubah dari *string* A menjadi *string* B, (Babar, 2017). Aplikasi yang ingin dikembangkan penulis akan memanfaatkan *Speech Recognizer* untuk menerjemahkan *input* suara pengguna menjadi teks Bahasa Arab. Hasil terjemahan tersebut akan dianalisis oleh aplikasi menggunakan metode *Levenstein Distance* untuk mencocokkan hasil terjemahan teks Bahasa Arab tersebut dengan teks ayat Alquran. Sehingga dari hasil metode *Levenstein Distance* ini akan mungkin terlihat perbedaan antara teks hasil olahan *Speech Recognizer* dengan teks ayat Alquran yang asli. Perbedaan yang terdeteksi adalah kelebihan huruf, kekurangaan huruf dan perbedaan ayat yang dibaca, hal tersebut yang akan diidentifikasi sebagai kesalahan pembacaan Alquran yang dilakukan pengguna.

Dalam pengembangan aplikasi nanti penulis akan melakukan pengujian kebutuhan Non Fungsional yang meliputi pengujian *Usability* dan pengujian akurasi*.* Pengujian *Usability* menitikberatkan pada kemudahan aplikasi ketika digunakan oleh pengguna. Sementara Pengujian Akurasi menitikberatkan pada hasil *ouput* letak kesalahan baca pengguna yang dideteksi dengan Algoritme *Levenstein Distance.*

## Rumusan masalah

Berdasarkan paparan latar belakang yang tersebut, maka penulis akan merumuskan beberapa masalah penelitian sebagai berikut :

1. Bagaimana memodelkan kebutuhan fungsional dan non fungsional dari apilikasi pendeteksi kesalahan baca Alquran?
2. Bagaimana rancangan aplikasi pendeteksi kesalahan baca Alquran berbasis Android?
3. Bagaimana implementasi *Speech Recognizer* dan metode *Levenstein Distance* pada aplikasi pendeteksi kesalahan baca Alquran tersebut?
4. Bagaimana hasil pengujian akurasi aplikasi pendeteksi kesalahan baca Alquran tersebut?

## Tujuan

Penelitian skripsi ini bertujuan untuk memenuhi hal-hal berikut :

1. Dapat memodelkan kebutuhan fungsional dan non fungsional aplikasi pendeteksi kesalahan baca Alquran.
2. Dapat mengetahui rancangan aplikasi pendeteksi kesalahan baca Alquran berbasis Android.
3. Dapat mengetahui cara mengimplementasikan *Speech Recognizer* dan metode *Levenstein Distance* pada aplikasi pendeteksi kesalahan baca Alquran.
4. Dapat mengetahui hasil uji akurasi aplikasi pendeteksi kesalahan baca Alquran.

## Manfaat

Aplikasi hasil dari penelitian ini nantinya diharapkan dapat memberikan kemudahan bagi setiap orang yang ingin berlatih membaca Alquran ketika tidak sedang bersama guru pembimbing dengan membantu mendeteksi kesalahan baca yang mungkin terjadi selama proses pembelajaran.

## Batasan Masalah

Penelitian ini memiliki batasan masalah yang dapat dirumuskan sebagai berikut :

1. Kesalahan baca yang bisa dideteksi aplikasi hanya terbatas pada kesalahan kekurangan huruf, kelebihan huruf dan perbedaan ayat yang dibaca.
2. Aplikasi tidak mendukung penerjemahan suara ke dalam Bahasa Arab yang memiliki diakritik atau harkat.
3. Aplikasi tidak bisa mendeteksi kesalahan baca tajwid.
4. Aplikasi hanya bisa digunakan jika perangkat Android terkoneksi ke internet.
5. Aplikasi dibuat hanya sebagai pelengkap pembelajaran baca Alquran, bukan pengganti guru pembimbing atau guru mengaji.

## Sistematika Pembahasan

Sistematika pembahasan pada penelitian ini terdiri dari delapan bab yang dimulai dari bab pendahuluan hingga bab penutup. Deskripsi singkat dari masing-masing bab tercantum sebagai berikut:

##### BAB 1 PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika penulisan penelitian.

##### BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tentang dasar teori dari semua komponen yang digunakan dalam penelitian mulai dari tahap perancangan, pengembangan hingga pengujian.

##### BAB 3 METODOLOGI PENELITIAN

Bab ini membahas langkah-langkah penelitian yang akan dilakukan, mencakup subjek, objek, metode dan algoritme yang akan digunakan dalam penelitian.

##### BAB 4 ANALISIS KEBUTUHAN

Bab ini membahas tentang menentukan kebutuhan dari aplikasi yang akan dihasilkan setelah penelitian ini, baik kebutuhan fungsional maupun non fungsional.

##### BAB 5 PERANCANGAN

Bab ini menguraikan tentang langkah-langkah rancangan aplikasi yang menjadi objek penelitian.

##### BAB 6 IMPLEMENTASI

Bab ini membahas tentang langkah-langkah implementasi untuk mewujudkan aplikasi yang menjadi objek penelitian.

##### BAB 7 PENGUJIAN DAN EVALUASI

Bab ini berisi tentang tahap-tahap pengujian fungsionalitas dan non fungsionalitas dari aplikasi dan evaluasi tercapai atau tidaknya tujuan penelitian.

##### BAB 8 PENUTUP

Bab ini menguraikan kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi yang dikembangkan dalam penelitian ini serta saran-saran untuk pengembangan lebih lanjut.

# LANDASAN KEPUSTAKAAN

Dalam bab Landasan kepustakaan ini akan menjelaskan tinjauan singkat materi dan studi yang memiliki hubungan dengan penelitian ini yang digunakan sebagai landasan dalam pembuatan aplikasi pendeteksi kesalahan baca Alquran berbasis android menggunakan *Speech Recognizer* dan metode *Levenstein Distance*.

## Kajian Pustaka

Kajian pustaka merupakan pembahasan untuk menganalisis penelitian terdahulu yang memiliki kemiripan topik bahasan dengan penelitian yang akan dilakukan penulis.

Pertama-tama penelitian yang dilakukan oleh Amalia Khasanah dengan yang memiliki judul “Perancangan Aplikasi Alquran Menggunakan *Voice Recognition* sebagai Media Pencarian Terjemahan Alquran Berbasis Android” dengan memanfaatkan *voice recognition* yang dapat mendeteksi suara dengan cara digitalisasi kata dan mencocokkan sinyal digital tersebut dengan suatu pola yang tersimpan dalam *database* yang sudah disediakan oleh perangkat sehingga dapat memunculkan hasil terjemahan ayat Alquran tersebut (Khasanah, 2014).

Selain itu pada penelitian kedua yang dilakukan oleh Achmad Fauzan dalam penelitiannya yang berjudul “Pembangunan Aplikasi Iqro’ Berbasis Android Menggunakan *Google Speech*” yang membahas tentang metode pembelajaran iqro interaktif dengan konsep gamifikasi. Dalam penelitian tersebut pengguna diminta untuk menjawab soal-soal menggunakan suara yang kemudian dikonversi dengan *Google Speech* menjadi teks yang akan dicocokkan dengan kunci jawaban (Fauzan, 2018).

Kemudian pada penelitian ketiga yang dilakukan oleh Hassan Al-Haj dengan judul “*Pronunciation Modeling for Dialectal Arabic Speech Recognition*” yang membahas tentang metode untuk memodelkan pengucapan Bahasa Arab dengan tanda baca. Dalam penelitian tersebut Hassan menjelaskan bahwa penulisan Bahasa Arab pada dasarnya hanya menggunakan huruf tanpa diakritik atau biasa disebut penulisan Arab gundul. Absennya diakritik ini dapat menyebabkan ambiguitas dalam pengucapan Bahasa Arab (Hassan, 2009).

## Alquran

Alquran merupakan kitab suci umat Islam yang berisikan firman-firman Allah SWT, dan diwahyukan dalam Bahasa Arab kepada Nabi Muhammad saw. Membaca Alquran termasuk salah satu bentuk ibadah yang paling utama. Jika ditinjau secara historis, Allah menurunkan Alquran kepada nabi Muhammad ditaklah secara utuh, melainkan dalam waktu yang cukup lama, yaitu sejak Muhammad diangkat menjadi seorang Rasul sampai beliau wafat. Selama masa tersebut diyakini oleh umat Islam sebagai masa turunnya wahyu yang berisi petunjuk dan ajaran dari segala bidang kehidupan, yang membahas masalah aqidah, ibadah ataupun muamalah (Shihab, 1994).

## *API* Alquran

API (*Applicaion Programming Interface*) Alquran adalah sebuah layanan *web* yang menyediakan *database* Alquran dan bisa digunakan menggunakan metode HTTP GET untuk mengembalikkan data berbentuk JSON (Islamic Network, 2018). Situs penyedia layanan yang penulis gunakan adalah dari situs [Tanzil Net yang diakses melalui alamat *tanzil.net* dan situs API Fatimah Bot yang diakses melalui alamat *api*.*banghasan*](http://api.banghasan).*com. S*itus API Fatimah Bot sebagai penyedia layanan API untuk daftar surat Alquran dan situs Tanzil Net sebagai penyedia *database* ayat-ayat Alquran.

## Android

Android adalah sistem operasi terbuka yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Sistem Android ini memiliki basis Linux sebagai pondasi dasarnya. Linux merupakan sistem operasi yang memang khusus dirancang untuk komputer (Azzam, 2017). Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan *library* dan *tools* yang dapat digunakan untuk membangun aplikasi. Android memiliki sekumpulan *tools* untuk membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi. Hingga saat ini Android sudah dikembangkan hingga versi terbarunya adalah Android 9 Pie.

### Android Studio

Android Studio merupakan alat pengembangan aplikasi *mobile* yang terpadu dan terintegrasi untuk pengembangan aplikasi Android, berdasarkan Intellij IDEA (Android, 2016). Tujuannya dibuatnya Android Studio adalah untuk mempercepat pengembangan dan membantu anda membuat aplikasi Android berkualitas tinggi (Android, 2018). Karena Android Studio dibuat khusus untuk pengembang Android, maka fitur di dalamnya meliputi pengeditan kode yang lengkap, *debugging*, pengujian, dan alat pembuatan profil.

### Sensor *Microphone*

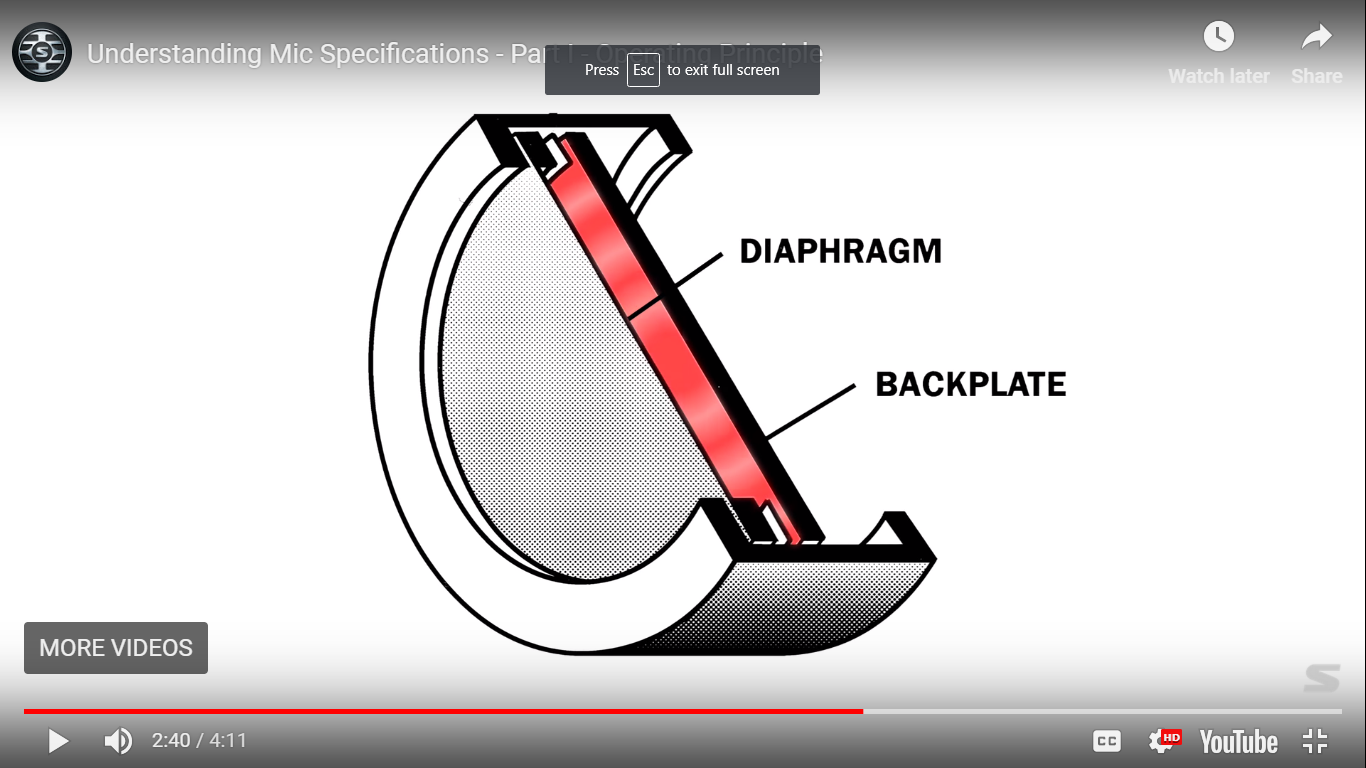
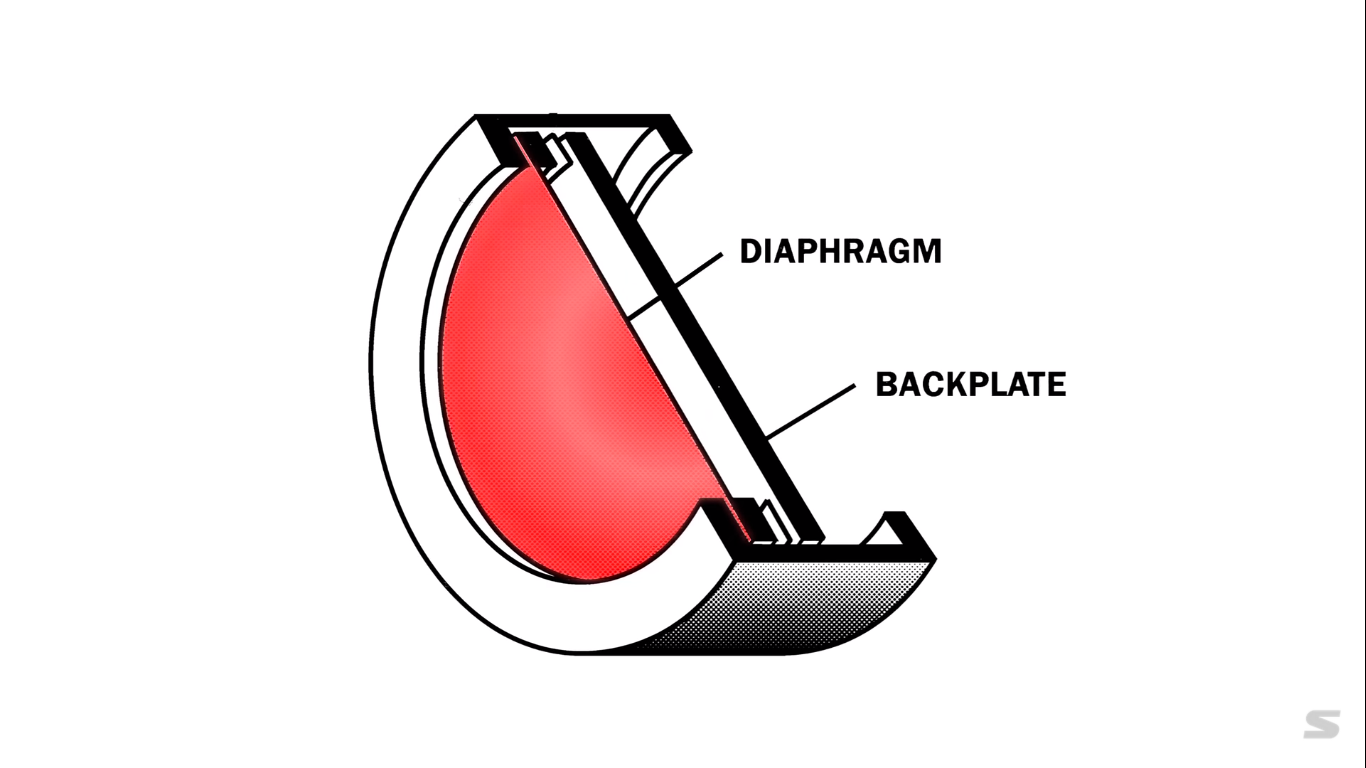
Sensor *Microphone* adalah sebuah alat di dalam perangkat Android yang berfungsi untuk merekam suara analog menjadi data digital yang bisa digunakan dalam aplikasi (Milette, 2017). Sensor *Microphone* biasa digunakan untuk merekam suara pengguna ketika sedang menelpon dan ketika pengguna melakukan navigasi perintah suara saat menyetir mobil.

Jenis sensor *Microphone* pada perangkat Android umumnya menggunakan *Condensor Microphone. Condensor Microphone* bekerja dengan 2 tahapan :

Rambatan gelombang suara melalui udara akan menggetarkan membran diafragma dan membuat membran tersebut bergerak turun-naik.

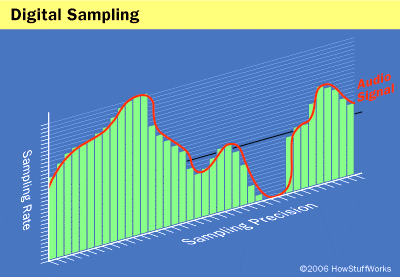
Pergerakan membrane diafragma tersebut membuat perubahan medan listrik pada *backplate* di belakangnya.

Perubahan medan listrik ini menghasilkan sinyal digital dari gelombang suara analog yang menggetarkan membran diafragma pada tahapan pertama (Kalwinder, 2013).



Gambar ‎2.1 Contoh *Condensor Microphone*

Sementara untuk merubah sinyal analog menjadi sinyal digital, dilakukan Sensor *Microphone* menggunakan teknik *digital sampling* dengan cara sinyal analog direkam beberapa kali setiap satuan waktu (frekuensi), yang disebut dengan satuan Hertz (Grabianowski, 2006). Misalkan *digital sampling* dilakukan dengan frekuensi batas pendengaran manusia yaitu 20.000Hz artinya dalam 1 detik dilakukan *sampling* sebanyak 20.000 kali oleh sensor *microphone*. Untuk menggambarkannya tertera pada gambar 2.2, kurva berwarna merah adalah sinyal analog dari suara dan grafik warna hijau adalah hasil *sampling* yang dilakukan sensor *microphone.*

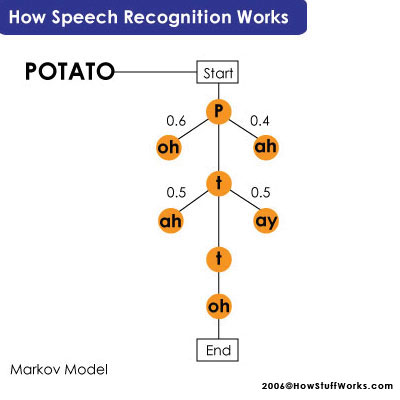


Gambar ‎2.2 Contoh *Digital Sampling*

## *Speech Recognizer*

*Speech Recognizer* merupakan sebuah *service* dari Platform Android yang berfungsi untuk menerjemahkan suara manusia menjadi sebuah teks. Layanan ini berbentuk API (*Application Program Interface*) yang dapat diimplementasikan ke dalam aplikasi Android. Menurut terjemahan situs *Android Developers*, *Speech Recognizer* ini bekerja dengan cara mengirimkan data *audio* yang ditangkap oleh sensor mikrofon ke *remote servers* untuk menerjemahkan suara data *audio* menjadi sebuah teks (Android, 2018). Layanan *Speech Recognizer* ini sudah mendukung 120 bahasa di dunia termasuk Bahasa Arab.

Teknologi *Speech Recognizer* modern umumnya menggunakan metode *statistical modeling systems* yang menggunakan fungsi matematika untuk memprediksi hasil yang paling mungkin dari suara. Contoh yang paling umum adalah metode *Markov Model* yang bekerja dengan cara menyatukan setiap satuan bunyi (fonem) secara berantai hingga membentuk suatu kata. Setiap rantai akan terhubung ke percabangan fonem yang berbeda. Dalam tahap ini, program akan memberikan skor probabilitas ke setiap cabang fonem berdasarkan kamus Bahasa maupun hasil *training* (Grabianowski, 2006)*.*



Gambar ‎2.3 Contoh *Markov Model*

Alasan penulis menggunakan *Speech Recognizer* ini adalah karena layanan ini sudah mendukung Bahasa Arab yang sesuai dengan Bahasa dalam Alquran, layanan ini memiliki dokumentasi yang cukup lengkap di situs Android Developers dan layanan ini bisa digunakan secara gratis.

Contoh penggunaan *SpeechRecognizer* dalam Android

|  |  |
| --- | --- |
| **1**  **2** | **val** speech = SpeechRecognizer.createSpeechRecognizer(**this**) speech.setRecognitionListener(**this**) |
| **3**  **4**  **5**  **6** | **val** intnt = Intent(RecognizerIntent.*ACTION\_RECOGNIZE\_SPEECH*) intnt.putExtra(RecognizerIntent.*EXTRA\_PARTIAL\_RESULTS*, **true**)  intnt.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE\_MODEL*, RecognizerIntent.*LANGUAGE\_MODEL\_FREE\_FORM*) intnt.putExtra(RecognizerIntent.*EXTRA\_LANGUAGE*, **"ar-SA"**) |

1. Baris 1-2 : Inisiasi Objek *SpeechRecognizer* dan *class* yang mengimplementasi *listener.*
2. Baris 3 : Inisiasi *intent* untuk menangkap *output SpeechRecognizer*
3. Baris 4-6 : Menambahkan objek *Extra* pada *intent* yang berupa penggunaan model Bahasa, fungsi *Partial\_Result* dan objek Bahasa Arab.

## Metode *Levenshtein Distance*

Metode *Levenshtein Distance* adalah perhitungan matriks untuk menghitung jumlah banyaknya perbedaan antara dua buah *string*. Jumlah perbedaan kedua *string* ini disebut *distance* yang ditentukan oleh jumlah berapa banyak operasi perubahan huruf untuk merubah dari *string* A menjadi *string* B, dan dihitung dengan tabel perhitungan *Levenshtein Distance* (Babar, 2017).

Ada tiga macam operasi utama yang bisa dilakukan oleh algoritme ini yaitu :

Operasi Mengubah Karakter, Operasi ini merupakan operasi menukar sebuah huruf dengan huruf lain. Misalkan penulis menuliskan *string* “pinter‟ menjadi “pintar‟. Dalam kasus ini huruf “e” harus diganti dengan huruf “a” (Adriyani, 2012).

Operasi Menambah Karakter, Operasi ini untuk menambahkan huruf ke dalam sebuah *string*. Misalkan *string* “diri” menjadi *string* “dirimu”, dilakukan penambahan huruf “mu‟ di akhir *string*. Penambahan huruf tidak harus dilakukan di akhir *string*, namun bisa ditambah di awal maupun diselipkan di tengah *string* (Adriyani, 2012).

Operasi Menghapus Karakter, Operasi ini untuk menghilangkan huruf dari suatu *string*. Misalkan *string* “barur‟ huruf terakhir dihilangkan sehingga menjadi *string* “baru‟. Kali ini dilakukan penghapusan huruf “r‟ di akhir *string* (Adriyani, 2012).

Penerapan metode *Levenstein Distance* dalam pemrograman ini terbagi menjadi 7 langkah sebagai berikut :

Jika ada dua buah kata a=”**BARU**” dengan b=”**BATU**”, hitunglah berapa perbedaan huruf (*distance*) kedua kata tersebut?

Langkah 1,

Siapkan variabel P untuk menyimpan panjang *string* *source* (S)

Siapkan variabel Q untuk menyimpan panjang *string* *target* (T)

Jika P = 0 atau Q=0 maka *exit*

Buat sebuah matriks ber-ordo [0 – P][0 – Q], beri nama matriks *distance* d[i,j]

Inisialisasi baris 0 s/d P + 1

Inisialisasi kolom 0 s/d Q + 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | B | A | R | U |
|  | 0 | 1 | 2 | 3 | 4 |
| B | 1 |  |  |  |  |
| A | 2 |  |  |  |  |
| T | 3 |  |  |  |  |
| U | 4 |  |  |  |  |

Langkah 2, Simpan setiap karakter dari *String* S ke dalam *Char Array* x[] (looping dari karakter ke 1-P) hasilnya simpan pada array x[i]. Hasilnya adalah x[] = [B, A, R, U].

Langkah 3, Simpan setiap karakter dari *String* T ke dalam *Char Array* y[] (looping dari karakter ke 1-Q) hasilnya simpan pada array y[j]. Hasilnya adalah y[] = [B, A, T, U].

Langkah 4, inisialisasi variable *cost* intuk menyimpan jarak. Kemudian cocokkan setiap karakter pada array x[] dengan y[].

Jika x[i] == y[i], maka cost = 0

Jika x[i] != y[i], maka cost = 1

Langkah 5, kemudian pilih nilai minimum dari

d[I-1, j] + 1 > d[1-1, 1] + 1 = 1

d[I, j-1] + 1 > d[1, 1-1] + 1 = 2

d[i-1, j-1] + cost > d[1-1, 1-1] + 0 = 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | B | A | R | U |
|  | 0 | 1 | 2 | 3 | 4 |
| B | 1 | **0** |  |  |  |
| A | 2 |  |  |  |  |
| T | 3 |  |  |  |  |
| U | 4 |  |  |  |  |

Langkah 6, Lanjutkan langkah 5 pada seluruh baris dan kolom, hingga matriks d[i,j] terisi seperti berikut :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | B | A | R | U |
|  | 0 | 1 | 2 | 3 | 4 |
| B | 1 | **0** | **1** | **2** | **3** |
| A | 2 | **1** | **0** | **1** | **2** |
| T | 3 | **2** | **1** | **1** | **2** |
| U | 4 | **3** | **2** | **2** | **1** |

Kemudian ambil nilai di sudut kanan-bawah dari matriks sebagai nilai *distance.* Dari matriks d[i,j] yang dihasilkan, diketahui perbedaan “BARU” dengan “BATU” adalah **1 huruf**.

## *Library Diff match patch*

*Diff match patch* adalah sebuah *library* dari Google yang digunakan untuk memanipulasi teks (Fraser, 2018). *Library* ini mendukung Bahasa pemrograman C++, C#, Java, Javasript, Dart, Lua, Objective-C dan Phyton. *Library* ini mengimplementasikan algoritme Myer yang umumnya dianggap sebagai algoritme terbaik untuk mencari perbedaan 2 buah *string.* Dalam *library* ini juga mendukung pemrosesan semantik untuk membuat *output* lebih mudah dibaca. Tiga fungsi utama *library Diff match patch* :

1. *Diff,* membandingkan 2 buah teks, dan mengembalikkan daftar perbedaan Antara 2 teks tersebut.
2. *Match,* digunakan untuk mencari sebuah *string* diantara blok teks menggunakan *fuzzy.*
3. *Patch,* digunakan untuk menambal serangkaian kata ke dalam sebuah teks.

Dalam implementasi penelitian ini penulis memfokuskan pada fungsi *Diff* dan fungsi-fungsi yang digunakan adalah :

1. diff\_main(text1, text2) → diffs, fungsi untuk mengkomputasi perbedaan Antara *text1* dan *text2*. Kemudian mengambalikkan daftar perbedaan diantara kedua *text* yang meliputi *insertion, deletion* dan *equality.*
2. diff\_cleanupSemantic(diffs) → null, fungsi untuk menyederhanakan hasil balikan dari fungsi diff\_main() supaya *output* nya lebih mudah dibaca oleh manusia.

## Wawancara

Wawancara adalah sebuah kegiatan tanya jawab yang dilakukan seorang pewawancara sebagai penanya dengan narasumber sebagai orang yang ditanya (KBBI, 2017). Tujuan dari wawancara ialah menggali informasi dari narasumber yang terpercaya. Wawancara dilakukan dengan pewawancara menyampaikan sejumlah pertanyaan kepada narasumber. Pewawancara akan menggunakan jawaban dari narasumber untuk dianalisis sebagai hasil wawancara.

Narasumber ialah seorang subyek yang akan memberikan beberapa fakta khusus tentang dirinya atau keluarganya (Felix, 2015). Narasumber yang mengungkapkan fakta-fakta terkait dirinya sendiri disebut Responden. Jumlah minimum responden yang dibutuhkan dalam suatu penelitian sangat bergantung kepada apa saja hal yang ingin diketahui peneliti, apa maksud dari penelitian, apa kegunaan penelitian, apa yang dianggap memiliki kredibilitas dan apa yang bisa dikerjakan dengan waktu dan sumberdaya yang sudah tersedia (Patton, 1990).

## *Validation Testing*

*Validation Testing* adalah pengujian spesifikasi fungsional dari perangkat lunak. Pengujian ini dilakukan dengan cara mengamati hasil eksekusi sistem melalui data uji sambil memeriksa fungsional dari perangkat lunak (Prastyo, 2009). Dalam pengujian ini akan terlihat apakah semua kebutuhan fungsional dari sistem sudah terpenuhi atau belum. *Validation Testing* termasuk juga dalam kategori *Blackbox Testing.*

## Pengujian *Usability*

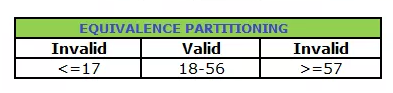
Pengujian *usability* adalah pengujian untuk mengukur tingkat kemudahan pengguna dalam menggunakan aplikasi (Farouqi, 2018). Selain itu pengujian *usability* bisa juga digunakan untuk mendeteksi *pain points* atau kesulitan yang dialami pengguna selama menggunakan aplikasi.

Salah satu metode yang dapat digunakan untuk mengukur skor *usability* adalah menggunakan *System Usability Scale (SUS)* yang merupakan metode karya John Brooke di tahun 1986 (Sauro, 2011). Metode SUS adalah sebuah kuesioner dengan 10 pernyataan dengan 5 buah skala skor di setiap pernyataannya. Pengguna diminta untuk mengisi SUS setelah menggunakan sistem.

Dalam menghitung nilai SUS dengan cara Skor untuk setiap pernyataan dengan angka ganjil, akan dihitung menggunakan rumus . Sementara itu, untuk setiap urutan pernyataan dengan angka genap, akan dihitung menggunakan rumus ( dengan ketentuan bahwa adalah angka pada skala pernyataan yang dipilih responden (Sauro, 2011).

## Pengujian Akurasi

Pengujian Akurasi adalah pengujian kebutuhan non fungsional yang terkait dengan tingkat kebenaran *output* yang dihasilkan aplikasi. Metode yang penulis gunakan adalah *Equivalence Partitioning Testing* yang termasuk dalam kategori *black box testing*. Metode ini mengklasifikasikan *input* menjadi beberapa kategori yang dianggap memberikan *output* yang sama (Rajkumar, 2018). Metode ini dapat mengurangi jumlah *test case* pada sebuah pengujian, karena dalam satu kategori cukup diwakilkan oleh satu *test case* dan *test case* lainnya dalam kategori tersebut dianggap akan memberikan *output* yang sama.Sebagai contoh dalam sebuah sistem hanya menerima *input* angka valid dengan *range* 18-56. Maka *inputan* angka <=17 dan >=57 akan dinyatakan tidak valid (Rajkumar, 2018).



Gambar ‎2.4 Contoh Klasifikasi Pengujian *Equivalence Partitioning Testing*

# METODOLOGI

Dalam bab Metodologi akan menjelaskan semua tahapan yang akan dilakukan sepanjang penelitian pengembangan aplikasi pendeteksi kesalahan baca Alquran berbasis Android. Metodologi penelitian ini tidak terlepas dari Siklus Hidup Pengembangan Sistem atau yang disebut SDLC (*Software Development Life Cycle*). Pada penelitian kali ini penulis menggunakan *SDLC* *Waterfall. SDLC* *Waterfall* merupakan dasar dari suatu cara untuk pengembangan sistem dengan fase yang berurutan (Sommerville, 2011). Berikut gambar 3.1 adalah metodologi yang akan digunakan dalam penelitian ini.

Studi Literatur

Analisis Kebutuhan

Perancangan

Pengujian & Analisis Hasil

Kesimpulan & Saran

Mulai

Selesai

Implementasi

Gambar ‎3.1 Diagram Alir Metodologi Penelitian

## Studi Literatur

Studi Literatur adalah sebuah proses metode untuk mengumpulkan data-data dari penelitian sebelumnya yang serupa. Sumbernya dari berbagai literatur yang berupa buku, jurnal, dokumentasi internet dan *paper*. Semua literatur yang digunakan untuk mendukung penelitian ini, tercantum dalam daftar pustaka di bagian akhir penelitian.

Buku-buku yang digunakan sebagai literatur pustaka adalah buku yang membahas tentang topik yang sesuai dengan landasan pustaka penelitian ini. Tahun penerbitan buku tidak dibatasi.

Jurnal dan *paper* yang digunakan adalah yang memiliki korelasi dengan penelitian yang dilakukan penulis. Tahun terbitnya jurnal dan *paper* paling lama adalah 5 tahun sebelum penelitian penulis dilakukan.

Dokumen Internet yang digunakan sebagai literatur adalah dokumen dari sumber yang relevan dan terpercaya. Dokumen dikatakan relevan apabila terdapat kesamaan topik dengan penelitian penulis. Dan dokumen dikatakan terpercaya apabila sumbernya berkompeten, misalnya lembaga ilmu pengetahuan, dosen, peneliti, perusahaan, jurnalis dan sebagainya. Topik-topik literatur yang digunakan adalah yang memiliki korelasi dengan penelitian penulis yaitu :

Alquran

Android

*Speech Recognizer*

Metode *Levenstein Distance*

*Library Diff Match Patch*

Metode Wawancara

*Validation Testing*

Pengujian *Usability*

Pengujian Akurasi Aplikasi

## Analisis Kebutuhan

Analisis kebutuhan dilakukan guna mengetahui kebutuhan aplikasi yang ingin dibuat. Tahap awal analisis kebutuhan adalah tahap Elisitas Kebutuhan untuk mengidentifikasi masalah yang ingin diselesaikan. Pada penelitian ini, penulis menggunakan teknik elisitasi kebutuhan tradisional yang berupa metode wawancara langsung dengan calon pengguna aplikasi. Calon pengguna aplikasi yang dimaksudkan penulis meliputi :

Imam Masjid Raden Patah Universitas Brawijaya

Mahasiswa Filkom Universitas Brawijaya

Hasil dari wawancara kepada calon pengguna tersebut akan dijadikan dasar dalam menentukan kebutuhan fungsional aplikasi. Setelah itu akan dilakukan analisis kebutuhan fungsional untuk melakukan identifikasi aktor, memodelkan *usecase* dan *usecase scenario* dari aplikasi. Oleh karena itu, pada penelitian kali ini penulis hanya akan memilih tiga orang narasumber yang mewakili calon pengguna yang ahli membaca Alquran dan calon pengguna umum.

## Perancangan

Perancangan dilakukan memodelkan sebuah aplikasi yang akan menyelesaikan suatu masalah dari hasil analisis kebutuhan. Metode Perancangan yang penulis gunakan adalah OOD (*Object Oriented Design*) menggunakan *tools* UML (Unified Modelling Language) yang berupa *Activity Diagram, Class Diagram dan Sequence diagram*. Penulis juga akan melakukan perancangan algoritme untuk memodifikasi algoritme *Levenstein Distance* agar dapat mendeteksi kesalahan baca Alquran dan menghasilkan *output* yang diharapkan. Kemudian perancangan antarmuka aplikasi dengan menggunakan metode *wireframing* dan *screenflow.*

## Implementasi

Implementasi merupakan fase untuk mengeksekusi semua perancangan yang telah dilakukan. Pembahasan implementasi mengacu pada fase perancangan aplikasi. Implementasi dilakukan dengan metode OOP (*Object Oriented Programming*) menggunakan Bahasa pemrograman Kotlin, Java dan *XML* untuk antarmuka. Implementasi yang dilakukan meliputi implementasi algoritma dan implementasi antarmuka.

## Pengujian dan Analisis Hasil

Pengujian dan analisis aplikasi dilakukan untuk menunjukkan bahwasannya aplikasi bisa bekerja sesuai harapan. Pengujian dilakukan melalui tiga cara, yaitu: *Validation Testing,* Pengujian *Usability* dan Pengujian Akurasi.

### *Validation Testing*

*Validation Testing* pada penelitian kali ini dilakukan dengan cara menguji setiap kebutuhan fungsional dari aplikasi menggunakan kasus uji dari setiap *usecase scenario* yang terdapat pada perancangan aplikasi. Apabila hasil pengujian sesuai dengan hasil yang diharapkan maka kasus uji dinyatakan berhasil/valid.

### Pengujian *Usability*

Pengujian *Usability* Aplikasi akan dilakukan dibagi menjadi tiga tahap yaitu pengujian *usability*, kuesioner, dan wawancara. Pengujian *Usability* diberikan dalam bentuk meminta 5 orang pengguna untuk menggunakan aplikasi selama 5 menit. Setelah itu, pengguna diminta untuk mengisi kuisioner *System Usability Scale (SUS)* sebanyak 10 pernyataan berdasarkan hasil pengalaman menggunakan aplikasi selama 5 menit tersebut.

Menurut Nielsen, jika ingin mengidentifikasi permasalahan dari desain sebuah sistem, diperlukan lima orang sebagai peserta tes. Pengujian dengan menggunakan lima orang memungkinkan pengembang menemukan lebih dari 80% masalah *usability* dan apabila pengembang menggunakan lebih banyak peserta tes maka peningkatannya tidak terlalu signifikan yaitu hanya sekitar 20% (Nielsen, 2000).

Terakhir adalah tahap wawancara yang dilakukan terhadap lima responden usai mereka mengisi kuesioner SUS. Pertanyaan wawancara dibagi menjadi tiga kategori yaitu pertanyaan seputar tampilan aplikasi, fungsional aplikasi, output aplikasi, performa aplikasi dan masalah terbesar yang dihadapi ketika menggunakan aplikasi.

Data yang diperoleh dari pengujian *usability*, merupakan data kualitatif dan kuantitatif. Data kualitatif didapatkan melalui metode wawancara. Jawaban pengguna akan diidentifikasi untuk mengetahui letak *pain points* pengguna ketika menggunakan aplikasi. *Pain points* yang ditemukan bisa akan digunakan sebagai acuan untuk menyempurnakan aplikasi lebih lanjut. Sementara data kuantitatif didapatkan melalui metode kuesioner SUS. Nilai pada kuesioner dapat menggambarkan kualitas aplikasi yang dihasilkan.

### Pengujian Akurasi

Pengujian Akurasi menitikberatkan pada *output* dari algoritme *Levenstein Distance* dan *Library Diff Match Patch* dalam mendeteksi kesalahan baca Alquran. Pengujian dilakukan dengan metode *Equivalence Partitioning Test Case* dengan membagi klasifikasi kasus uji menjadi empat, yaitu kasus uji bacaan benar, bacaan kekurangan huruf, bacaan kelebihan huruf dan bacaan dari ayat yang berbeda. Dengan menggunakan objek pengujian dari ayat ke-5 surat Al-Bayyinah.

## Kesimpulan dan Saran

Tahap kesimpulan dan saran dilakukan setelah peneliti menyelesikan seluruh tahap penelitian. Kesimpulan penelitian adalah hasil dari pengujian dan analisis, kesimpulan ini menyatakan apakah rumusan masalah telah terselesaikan atau belum. Sementara Tahapan paling akhir adalah penulisan saran, penulisan saran bertujuan untuk memberi masukan guna memperbaiki kesalahan yang terjadi sepanjang penelitian di samping itu saran juga memberikan pertimbangan dan pandangan bagi penelitian selanjutnya.

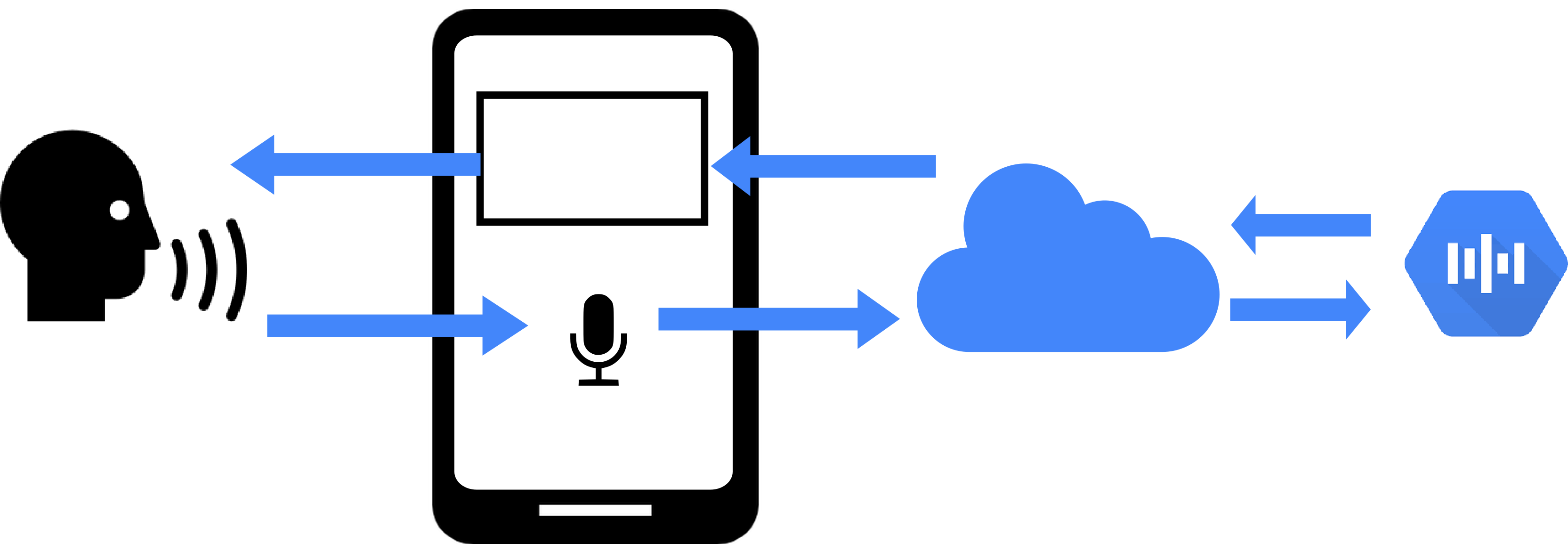
# ANALISIS KEBUTUHAN

Dalam bab Analisis Kebutuhan ini akan membahas tentang proses analisis kebutuhan dari aplikasi Identifikasi Kesalahan Baca Alquran yang akan dibagi menjadi tujuh bagian, yaitu : gambaran umum aplikasi, identifikasi aktor, elisitas kebutuhan, analisis kebutuhan fungsional, analisis kebutuhan non fungsional, *usecase* dan *usecase scenario.*

## Gambaran Umum Aplikasi

Aplikasi Identifikasi Kesalahan Baca Alquran adalah aplikasi berbasis Android yang dibangun dengan Bahasa pemrograman Kotlin, Java dan *XML*. Aplikasi ini direncanakan akan memiliki 3 fitur utama yang berupa : pemilihan Surat dan Ayat, identifikasi kesalahan bacaan dari suara pengguna dan pemberian rekomendasi koreksi kesalahan bacaan. Gambaran alur kerja aplikasi tercantum dalam Gambar 4.1.

Aplikasi



*Speech Recognizer*

Algoritme *Levenstein Distance*

*Remote Server*

Internet

Pengguna

**Gambar ‎4.1 Gambaran Umum Aplikasi**

Untuk menerjemahkan suara pengguna ke dalam teks Bahasa Arab, aplikasi ini akan memanfaatkan mikrofon pada perangkat Android dan *Speech Recognizer* dalam implementasinya. Mikrofon pada perangkat Android digunakan untuk merekam suara pengguna ketika membaca ayat Alquran. Hasil rekaman suara tersebut kemudian dikirimkan ke *server* Google untuk diterjemahkan ke dalam teks Bahasa Arab oleh *Speech Recognizer*. Kemudian *server* Google akan mengirimkan kembali hasil terjemahan berupa teks Bahasa Arab. Dalam proses tersebut, aplikasi harus terkoneksi dengan internet agar bisa mengirim dan menerima data dari *server* Google. Hasil terjemahan tersebut adalah berupa teks yang bisa langsung ditampilkan pada antarmuka aplikasi.

Pada fitur identifikasi kesalahan bacaan, penulis akan memanfaatkan algoritme *Levenstein Distance* dan *Library Diff Match Patch* untuk mendeteksi ada atau tidaknya kesalahan bacaan dari pengguna. Proses identifikasi ini diawali dari mencocokkan teks hasil terjemahan *Speech Recognizer* dengan teks ayat Alquran asli. Kedua teks tersebut akan dicocokkan dengan algoritme *Levenstein Distance* huruf per hurufnya. Apabila ada ketidakcocokan diantara hurufnya akan terdeteksi sebagai kesalahan bacaan. Hasil deteksi ini yang nanti digunakan untuk melakukan koreksi bacaan pengguna.

Terakhir pada fitur koreksi kesalahan bacaan, aplikasi akan memberikan koreksi dari setiap huruf yang terdeteksi sebagai kesalahan bacaan oleh pengguna. Koreksi yang bisa diberikan adalah seputar kesalahan pengucapan huruf, dan kesalahan dari pengucapan panjang/pendeknya huruf. Koreksi nantinya akan diberikan perkata sesuai dengan kesalahan yang terdeteksi.

## Identifikasi Aktor

Dalam bagian identifikasi aktor akan dijelaskan siapa saja pengguna yang terlibat dalam aplikasi yang akan dibangun. Aktor adalah pihak di luar system yang terlibat dan berinteraksi langsung dengan sistem. Berikut aktor yang terlibat dalam aplikasi Identifikasi Kesalahan Baca Alquran yang terlampir pada Tabel 4.1.

Tabel ‎4.1 Identifikasi Aktor

|  |  |  |
| --- | --- | --- |
| **No** | **Aktor** | **Deskripsi** |
| 1 | Pengguna | Semua orang yang dapat menggunakan aplikasi. Meliputi orang yang sudah ahli membaca Alquran maupun yang belum ahli membaca Alquran |

## Elisitasi Kebutuhan

Pada bagian elisitasi kebutuhan akan dijelaskan proses penentuan kebutuhan fungsional dan non fungsional dari aplikasi. Proses penentuan kebutuhan dilakukan melalui wawancara dengan calon pengguna aplikasi. Pada penelitian ini, penulis melakukan wawancara dengan tiga orang yang berpotensi menjadi pengguna aplikasi. Calon pengguna aplikasi dibedakan menjadi calon pengguna *expert* dan umum. Calon pengguna *expert* meliputi orang-orang yang ahli dalam membaca Alquran, contohnya guru mengaji dan imam masjid. Sementara calon pengguna umum meliputi pengguna yang belum ahli dalam membaca Alquran.

## Penomoran Kode Kebutuhan

Penomoran kode kebutuhan dilakukan untuk membedakan Antara kebutuhan fungsional dan kebutuhan non fungsional. Berikut daftar penomoran kode kebutuhan dari aplikasi identifikasi kesalahan baca Alquran terdapat pada Tabel 4.2.

Tabel ‎4.2 Penomoran Kode Kebutuhan

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Kode** | **Nama Kode** | **Deskripsi** |
| 1 | TQ | Singkatan dari nama aplikasi | Kode ini diletakkan di awal penulisan kebutuhan, untuk menandakan nama aplikasi |
| 2 | F | F adalah singkatan dari Fungsional | Kode ini digunakan untuk menandakan kebutuhan fungsional |
| 3 | NF | NF adalah singkatan dari Non Fungsional | Kode ini digunakan untuk menandakan kebutuhan non fungsional |
| 4 | NOMOR AKTOR | Berisi angka 1 digit | Kode ini digunakan untuk menandakan nomor urut dari aktor.  1. Pengguna |
| 5 | NOMOR KEBUTUHAN | Berisi angka 2 digit | Kode ini digunakan untuk menandakan nomor urut dari kebutuhan aplikasi |

Contoh :

1. **TQ-F-1-01** : Representasi dari kebutuhan fungsional urutan ke-01 dari aplikasi Identifikasi kesalahan baca Alquran dengan **Pengguna** sebagai aktor.
2. **TQ-NF-0-01** : Representasi dari kebutuhan non fungsional urutan ke-01 dari aplikasi Identifikasi kesalahan baca Alquran tanpa aktor.

## Analisis Kebutuhan

Pada bagian analisis kebutuhan akan mendeskripsikan kebutuhan fungsional dan kebutuhan non fungsional dari aplikasi.

### Kebutuhan Fungsional

Kebutuhan fungsional akan mendeskripsikan fungsi apa saja yang bisa dijalankan oleh aplikasi Identifikasi kesalahan baca Alquran. Untuk saat ini Kebutuhan fungsional yang telah ditetapkan adalah kebutuhan fungsional pada *prototype* (purwarupa) dari aplikasi yang akan dibuat. Berikut daftar kebutuhan fungsional dari purwarupa aplikasi identifikasi kesalahan baca Alquran terdapat pada Tabel 4.3.

Tabel ‎4.3 Kebutuhan Fungsional

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Kode** | **Nama Fungsi** | **Deskripsi** |
| 1 | TQ-F-1-01 | Menampilkan daftar surat | Aplikasi harus dapat menampilkan daftar dari 114 surat yang ada di Alquran, termasuk dengan nomor surat, nama surat, jumlah ayat, asma surat dan kota tempat turunnya surat. |
| 2 | TQ-F-1-02 | Menampilkan daftar ayat | Aplikasi harus dapat menampilkan daftar semua ayat dari surat yang dipilih oleh pengguna. |
| 3 | TQ-F-1-03 | Mendeteksi kesalahan baca pengguna | Aplikasi harus bisa mendeteksi kesalahan baca Alquran yang dilakukan pengguna |

Pada tabel 4.3 tersebut baru merupakan kebutuhan fungsional dari *prototype* dari aplikasi yang sebenarnya. Setiap kebutuhan fungsional akan dimodelkan ke dalam *usecase diagram* dengan *usecase scenario* nya.

### Kebutuhan Non Fungsional

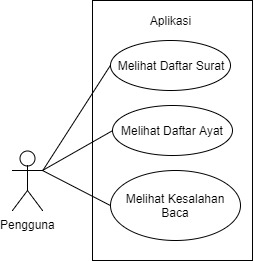
Kebutuhan non fungsional akan membahas tentang perilaku dari aplikasi. Berikut daftar kebutuhan non fungsional dari purwarupa aplikasi identifikasi kesalahan baca Alquran terdapat pada Tabel 4.4.

Tabel ‎4.4 Kebutuhan Non Fungsional

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Kode** | **Nama Kebutuhan** | **Deskripsi** |
| 1 | **TQ-NF-0-01** | *Usability* | Kemudahan penggunaan, aplikasi harus mudah digunakan oleh berbagai karakteristik pengguna mulai dari pengguna awam dan pengguna expert. |

### *Usecase Diagram*

*Usecase diagram* merupakan diagram yang menggambarkan interaksi Antara aplikasi aktor. Berikut ini adalah *usecase diagram* dari aplikasi Identifikas kesalahan baca Alquran yang dapat dilihat pada Gambar 4.2.



**Gambar ‎4.2 *Usecase Diagram***

Pada gambar 4.2 merupakan *usecase* diagram dari *prototype* aplikasi Identifikasi kesalahan baca Alquran yang memiliki satu aktor yaitu Pengguna. Sementara itu ada tiga *usecase* yang berupa melihat daftar surat, melihat daftar ayat dan melihat kesalahan baca pengguna.

### *Usecase Scenario*

*Usecase scenario merupakan* bagian untuk menjelaskan suatu *usecase* pada aplikasi. *Usecase scenario* ini bertujuan untuk mendapatkan deskripsi secara global mengenai *usecase*, kondisi awal dan akhir yang harus dipenuhi oleh *usecase* setelah fungsionalitas selesai dijalankan. Dalam skenario ini akan diulas bagaimana harapan tanggapan sistem terhadap aksi yang dilakukan oleh aktor. Berikut ini adalah *usecase scenario* dari aplikasi Identifikasi kesalahan baca Alquran yang dapat dilihat pada Tabel 4.5 sampai Tabel 4.7.

Tabel ‎4.5 *Usecase Scenario* Melihat Daftar Surat

|  |  |
| --- | --- |
| **Kode** | TQ-F-1-01 |
| **Nama Fungsi** | Melihat Daftar Surat |
| **Aktor** | Pengguna |
| **Deskripsi** | Pengguna bisa melihat daftar 114 surat dalam Alquran berserta nomor surat, nama surat, jumlah ayat dan tempat turunnya surat |
| **Pre-Condition** | - |
| ***Main Flow*** | Pengguna membuka aplikasi  Aplikasi menampilkan seluruh daftar 114 surat Alquran |
| ***Alternative Flow*** | Pengguna membuka aplikasi  Aplikasi menampilkan pesan *error* dan meminta pengguna untuk melakukan *refresh* |
| ***Post-Condition*** | Aplikasi menampilkan daftar 114 surat Alquran berserta nomor surat, nama surat, jumlah ayat dan tempat turunnya surat |

Tabel ‎4.6 *Usecase Scenario* Melihat Daftar Ayat

|  |  |
| --- | --- |
| **Kode** | TQ-F-1-02 |
| **Nama Fungsi** | Melihat Daftar Ayat |
| **Aktor** | Pengguna |
| **Deskripsi** | Pengguna bisa melihat daftar semua ayat dari surat yang dipilih oleh pengguna. |
| **Pre-Condition** | Pengguna memilih salah satu surat Alquran dari daftar surat |
| ***Main Flow*** | Pengguna memilih salah satu surat Alquran  Aplikasi menampilkan seluruh ayat dari surat yang dipilih |
| ***Alternative Flow*** | - |
| ***Post-Condition*** | Aplikasi menampilkan semua ayat dari surat yang dipilih oleh pengguna |

Tabel ‎4.7 *Usecase Scenario* Melihat Kesalahan Baca

|  |  |
| --- | --- |
| **Kode** | TQ-F-1-04 |
| **Nama Fungsi** | Melihat Kesalahan Baca |
| **Aktor** | Pengguna |
| **Deskripsi** | Pengguna bisa melihat kesalahan baca yang dilakukan |
| **Pre-Condition** | Pengguna membaca salah satu ayat Alquran |
| ***Main Flow*** | Pengguna memilih salah satu ayat Alquran  Pengguna menekan tombol rekam  Pengguna membaca ayat Alquran  Pengguna telah selesai merekam suara  Aplikasi menampilkan daftar kesalahan baca yang dilakukan pengguna |
| ***Alternative Flow*** | Pengguna memilih salah satu ayat Alquran  Pengguna menekan tombol rekam  *Microphone* aplikasi tidak bisa mendengar suara pengguna  Pengguna diminta menekan tombol rekam kembali |
| ***Post-Condition*** | Aplikasi menampilkan daftar kesalahan baca yang dilakukan pengguna |

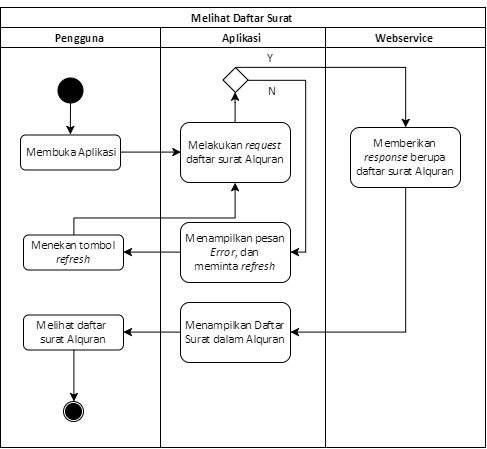
# PERANCANGAN

Dalam bab Perancangan ini akan dibahas seputar rancangan Aplikasi Identifikasi Kesalahan Baca Alquran. Beberapa tahapan perancangan ini dapat dibagi menjadi enam bagian, yaitu : *activity diagram,* *class diagram, sequence diagram,* perancangan algoritme dan perancangan antarmuka Masing-masing tahapan tersebut akan dijelaskan dalam bagian bab perancangan berikut ini.

## *Activity Diagram*

Perancangan *Activity Diagram* adalah sebuah metode pemodelan yang menjelaskan aktivitas apa saja yang dilakukan aktor terhadap aplikasi berdasarkan *usecase scenario* pada Bab 4. Pada Gambar 5.1 – Gambar 5.3 berikut ini adalah perancangan dari *Activity Diagram.*

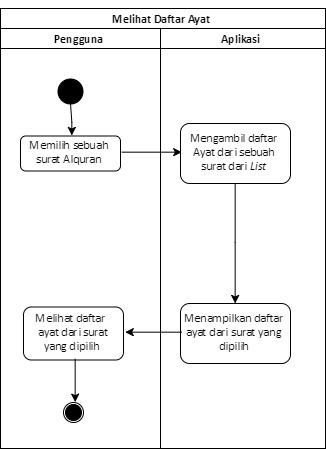
### *Activity Diagram* Melihat Daftar Surat



**Gambar ‎5.1 *Activity Diagram* Melihat Daftar Surat**

Pada Gambar 5.1 adalah *activity* diagram dari *usecase* *scenario* TQ-F-1-01 Melihat daftar surat. Alur aktifitas melihat daftar surat ini di awali ketika pengguna membuka aplikasi, aplkasi secara otomatis akan mengirimkan *request* ke *webservice* untuk mendapatkan daftar surat-surat dalam Alquran. Jika *request* berhasil, maka *webservice* akan memberikan *response* berupa objek JSON yang berisi daftar surat-surat dalam Alquran. Kemudian objek JSON tersebut akan diolah oleh aplikasi dan ditampilkan kepada pengguna dalam bentuk Daftar Surat Alquran. Namun, apabila aplikasi gagal melakukan *request* ke *webservice* maka akan muncul pesan *error* dan tombol *refresh* untuk meminta pengguna melakukan *refresh* dan aplikasi melakukan *request* ulang ke *webservice.*

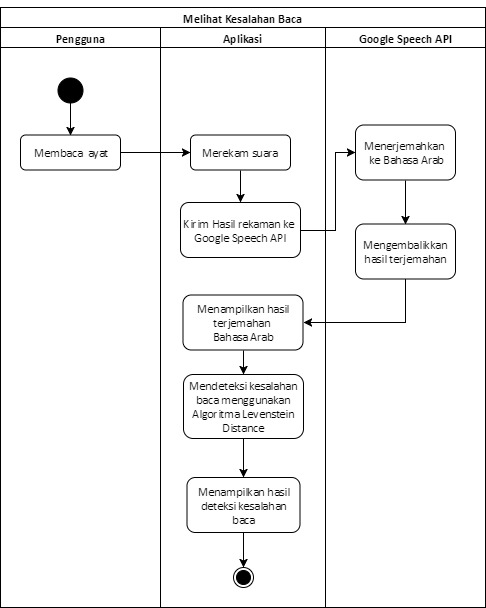
### *Activity Diagram* Melihat Daftar Ayat



**Gambar ‎5.2 *Activity Diagram* Melihat Daftar Ayat**

Pada Gambar 5.2 adalah *activity* diagram dari *usecase* *scenario* TQ-F-1-02 melihat daftar ayat. Alur aktifitas melihat daftar ayat ini di awali ketika pengguna memilih sebuah surat di halaman daftar surat, aplikasi secara otomatis akan memilih beberapa ayat dari *ListAyat* berdasarkan surat yang dipilih pengguna. Kemudian daftar ayat tersebut akan diolah oleh aplikasi dan ditampilkan kepada pengguna dalam bentuk daftar ayat dari sebuah surat.

### *Activity Diagram* Melihat Kesalahan Baca

****

**Gambar ‎5.3 *Activity Diagram* Melihat Kesalahan Baca**

Pada Gambar 5.3 adalah *activity* diagram dari *usecase* *scenario* TQ-F-1-34 melihat kesalahan baca. Alur aktifitas melihat kesalahan baca ini di awali ketika pengguna membaca ayat di halaman tahsin, kemudian aplikasi akan merekam suara pengguna dan mangirimkannya ke *Speech Recognizer*. Aplikasi kemudian menerjemahkan rekaman suara tersebut ke dalam teks Bahasa Arab. Kemudian aplikasi menampilkan teks hasil rekaman tersebut kepada pengguna. Selanjutnya aplikasi akan mendeteksi kesalahan baca pengguna menggunakan algoritme *Levenstein Distance* berdasarkan teks hasil rekaman yang ditampilkan. Setelah itu aplikasi akan menampilkan hasil deteksi kesalahan baca kepada pengguna.

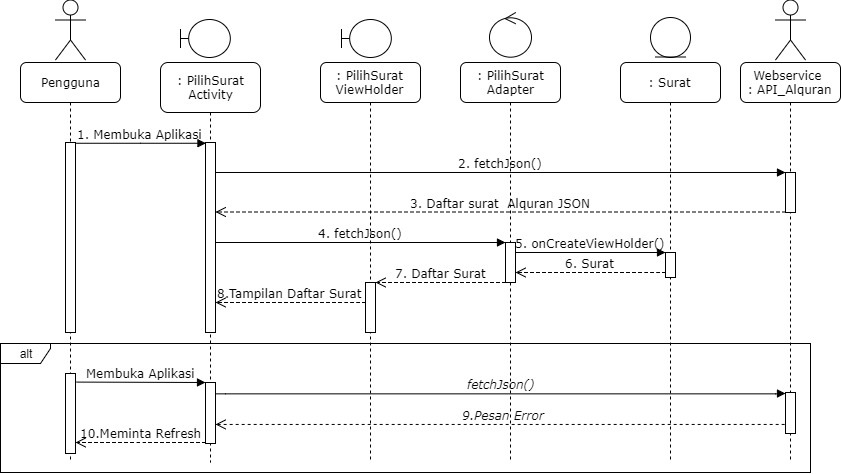
## *Sequence diagram*

Perancangan *Sequence diagram* adalah penjelasan tentang interaksi antar objek dan kelas yang ada dalam aplikasi dan menurut pada garis waktu. Perancangan *sequence diagram* mengacu kepada *usecase scenario* pada Bab 4. Objek-objek *sequence diagram* aplikasi dijelaskan pada tabel 5.1. Pada Gambar 5.2.1 – Gambar 5.2.4 berikut ini adalah perancangan *sequence diagram* aplikasi*.*

Tabel ‎5.1 Objek *Sequence diagram*

|  |  |  |
| --- | --- | --- |
| **No** | **Nama Objek** | **Penjelasan** |
| 1 |  | Pengguna, merupakan entitas aktor di luar sistem yang dapat berinteraksi langsung dengan sistem. |
| 2 |  | QuranAPI dan *Speech Recognizer*, merupakan entitas di luar sistem yang berbentuk *webservice*. QuranAPI menyimpan *database* Alquran berupa Ayat dan terjemahan yang dibutuhkan oleh sistem. Sementara *Speech Recognizer* berfungsi untuk menerjemahkan suara menjadi teks. |
| 3 | : *Activity* | *Activity* adalah objek pada aplikasi Android yang menampilkan dan mengatur halaman aplikasi sebagai tempat interaksi antara pengguna dengan aplikasi. *Activity* digambarkan sebagai *boundary* dari sistem. |
| 4 | : *Adapter* | *Adapter* adalah objek pada aplikasi Android yang bekerja sebagai jembatan antara *View* dengan *Data* yang ingin ditampilkan pada *View* tersebut. *Adapter* digambarkan sebagai *control* dari sistem. |
| 5 | : *Viewholder* | *Viewholder* adalah objek pada aplikasi Android yang berfungsi menampung data yang ingin ditampilkan kepada pengguna. *Viewholder* digambarkan sebagai *boundary* dari sistem karena dapat berinteraksi langsung dengan pengguna. |
| 6 |  | Surat dan Ayat adalah objek *Entity* dari sistem yang menyimpan struktur data dari Surat maupun Ayat Alquran. |
| 7 |  | Service adalah komponen aplikasi Android yang digunakan untuk melakukan operasi jangka panjang di *background process.* Service tidak memiliki *user-interface* dan termasuk dalam *control* dari sistem. |
| 8 |  | *Levenstein Distance* adalah sebuah objek buatan yang berisi algoritme untuk mendeteksi kesalahan baca Alquran oleh pengguna. |

### *Sequence diagram* Melihat Daftar Surat

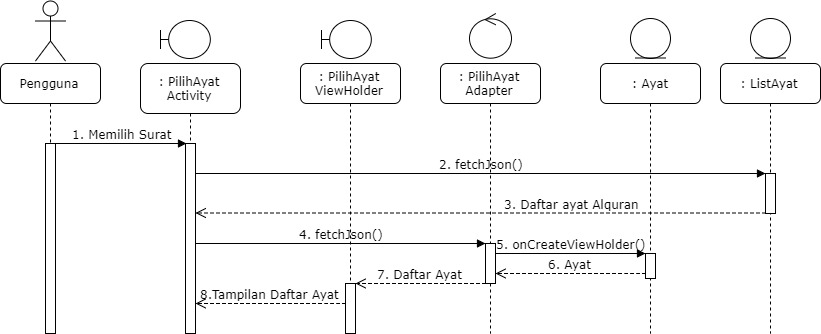


**Gambar ‎5.4 *Sequence diagram* Melihat Daftar Surat**

Pada Gambar 5.4 adalah *sequence diagram* dari *usecase* *scenario* TQ-F-1-01 Melihat daftar surat. Alur aktifitas melihat daftar surat ini adalah :

1. Di awali ketika pengguna membuka aplikasi.
2. Aplikasi secara otomatis mengirimkan *request* ke *webservice* menggunakan fungsi fetchJson() untuk mendapatkan daftar surat-surat dalam Alquran.
3. Jika *request* berhasil, maka *webservice* akan memberikan *response* berupa objek JSON yang berisi daftar surat-surat dalam Alquran.
4. Kemudian objek JSON tersebut dikirimkan kepada *adapter*.
5. Dalam *Adapter*, objek JSON akan di *mapping* ke dalam entitas Surat.
6. Setiap entitas Surat akan dirimkan kembali ke *Adapter*.
7. Dalam *adapter* setiap entitas Surat dimasukkan ke dalam *Viewholder* berbentuk *RecycleView.*
8. *RecycleView* yang berisi daftar surat ditampilkan kepada pengguna melalui *Activity*.
9. Jika aplikasi gagal melakukan *request* ke *webservice* maka akan muncul pesan *error.*
10. Dalam *activity* akan muncul tombol *refresh* untuk meminta pengguna melakukan *refresh* dan aplikasi melakukan *request* ulang ke *webservice.*

### *Sequence diagram* Melihat Daftar Ayat

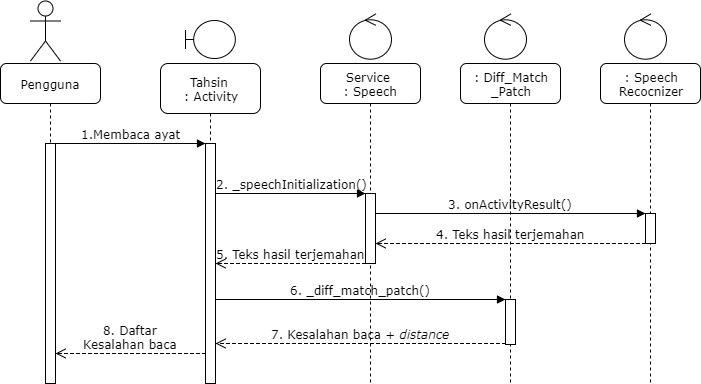
****

**Gambar ‎5.5 *Sequence diagram* Melihat Daftar Ayat**

Pada Gambar 5.5 adalah *sequence diagram* dari *usecase* *scenario* TQ-F-1-02 melihat daftar ayat. Alur aktifitas melihat daftar ayat ini adalah :

1. Di awali ketika pengguna memilih sebuah surat di halaman daftar surat.
2. Aplikasi secara otomatis akan memilih beberapa ayat dari *ListAyat* menggunakan fungsi fetchJson() berdasarkan surat yang dipilih pengguna.
3. Aplikasi mengirim daftar ayat untuk ditampilkan.
4. Kemudian daftar ayat tersebut dikirimkan kepada *adapter*.
5. Dalam *Adapter*, daftar ayat tersebut akan di *mapping* ke dalam entitas Ayat.
6. Setiap entitas Ayat akan dirimkan kembali ke *Adapter*.
7. Dalam *adapter* setiap entitas Ayat dimasukkan ke dalam *Viewholder* berbentuk *RecycleView.*
8. *RecycleView* yang berisi daftar ayat ditampilkan kepada pengguna melalui *Activity*.

### *Sequence diagram* Melihat Kesalahan Baca

****

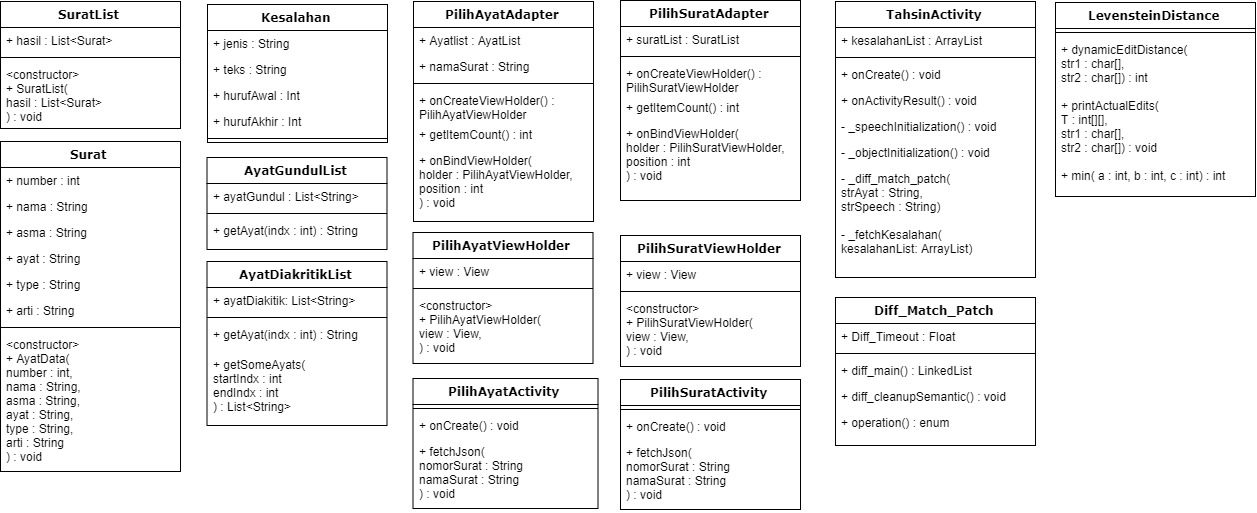
**Gambar ‎5.6 *Sequence diagram* Melihat Kesalahan Baca**

Pada Gambar 5.6 adalah *activity* diagram dari *usecase* *scenario* TQ-F-1-04 melihat kesalahan baca. Alur aktifitas melihat kesalahan baca ini adalah :

1. Di awali ketika pengguna membaca ayat di halaman tahsin,
2. Kemudian aplikasi akan merekam suara pengguna
3. Rekaman suara pengguna dikirimkan ke *Speech Recognizer*.
4. Kemudian *Speech Recognizer* mengembalikkan hasil tersejamah dalam teks Bahasa Arab.
5. Aplikasi menampilkan teks hasil rekaman tersebut kepada pengguna.
6. Selanjutnya aplikasi akan mendeteksi kesalahan baca pengguna menggunakan algoritme *diff match patch* dan *Levenstein Distance* berdasarkan teks hasil rekaman yang ditampilkan.
7. Hasil deteksi kesalahan baca dan *distance* perbedaan bacaan dikembalikkan ke *Activity*
8. *Activity* menampilkan hasil deteksi kesalahan baca kepada pengguna.

## *Class Diagram*

Perancangan *Class Diagram* akan menjelaskan bagaimana antar kelas dan objek yang dirancang pada aplikasi saling terhubung.

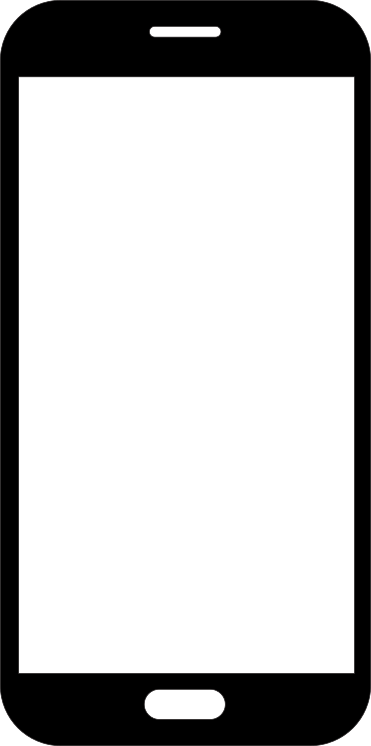
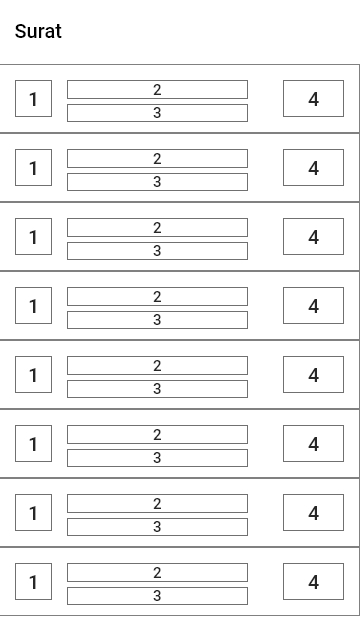


**Gambar ‎5.7 *Class Diagram* Aplikasi**

## Perancangan Antarmuka

Perancangan Antarmuka menjelaskan tentang pemodelan antarmuka aplikasi yang berupa *Low Fidelity Prototype* berbentuk *Wireframe*. Perancangan antarmuka pada penelitian ini mengacu kepada *usecase* *scenario* pada bab 4. Penulis akan menjelaskan *screenflow* dari aplikasi pada gambar 5.11. Sementara itu, pada Gambar 5.8 – Gambar 5.10 adalah *Wireframe* dari perancangan antarmuka aplikasi*.*

### Perancangan Antarmuka Melihat Daftar Surat



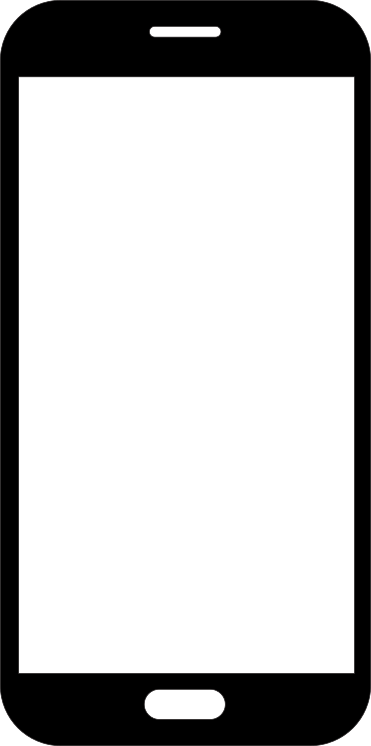
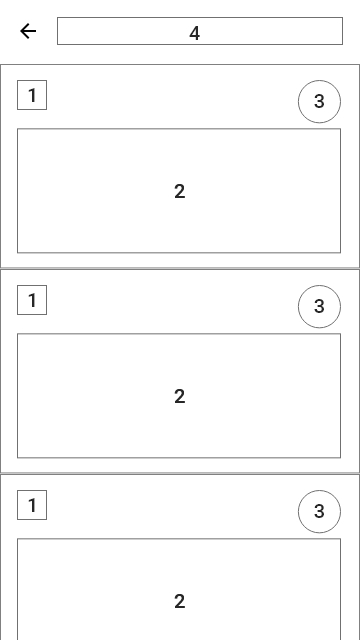
**Gambar ‎5.8 *Wireframe* Antarmuka Melihat Daftar Surat**

Pada gambar 5.8 adalah *Wireframe* dari *usecase* *scenario* nomor TQ-F-1-01 Melihat Daftar Surat. Tampilan melihat daftar surat merupakan tampilan awal dari aplikasi yang menampilkan daftar surat berupa nomor urut surat, nama surat, tempat turunnya surat dan asma surat.

Tabel ‎5.2 Keterangan Antarmuka Melihat Daftar Surat

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Objek** | **Tipe** | **Keterangan** |
| 1 | Nomor urut | *TextView* | Nomor urut surat dalam Alquran |
| 2 | Nama surat | *TextView* | Nama Surat dalam huruf latin |
| 3 | Tipe surat | *TextView* | Tempat turunnya surat beserta jumlah ayatnya |
| 4 | Asma surat | *TextView* | Nama surat yang ditulis dalam Bahasa Arab |

### Perancangan Antarmuka Melihat Daftar Ayat



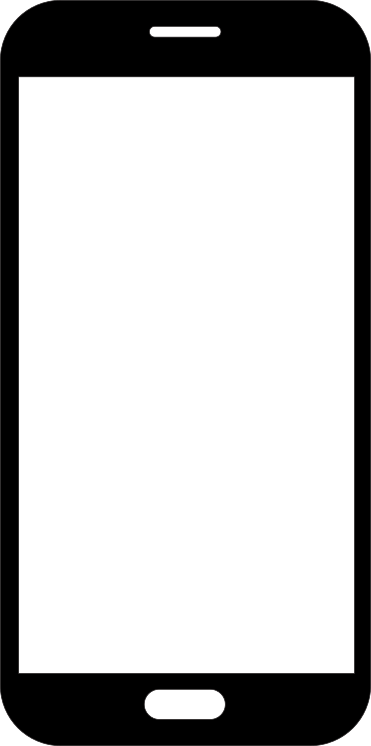
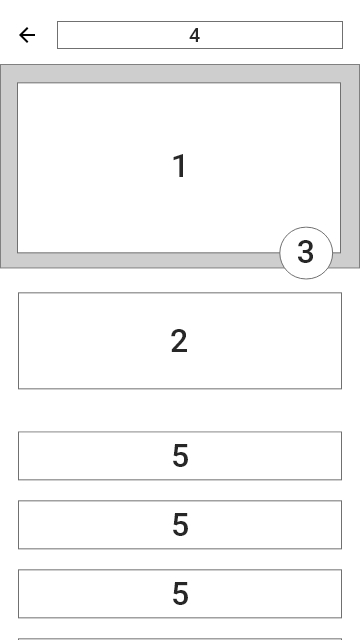
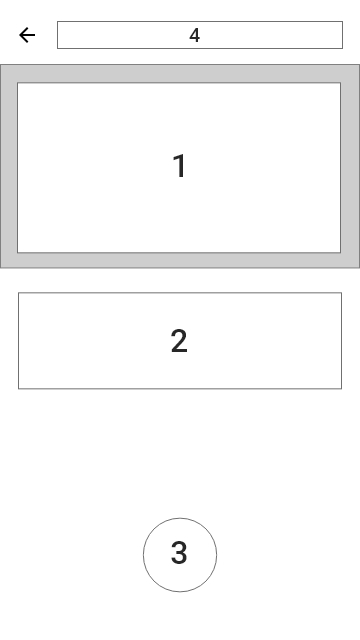
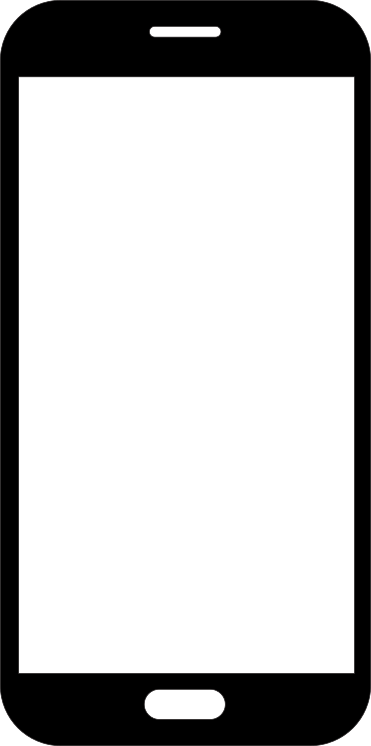
**Gambar ‎5.9 *Wireframe* Antarmuka Melihat Daftar ayat**

Pada gambar 5.9 adalah *Wireframe* dari *usecase* *scenario* nomor TQ-F-1-02 Melihat Daftar Ayat. Tampilan melihat daftar ayat merupakan tampilan setelah pengguna memilih salah satu surat dari halaman Daftar Surat. Antarmuka melihat daftar ayat menampilkan nomor urut ayat, nama surat, teks ayat dan tombol rekam suara. Penjelasan selengkapnya dari *Wireframe* melihat daftar ayat ini terdapat pada Tabel 5.3.

Tabel ‎5.3 Keterangan Antarmuka Melihat Daftar Ayat

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Objek** | **Tipe** | **Keterangan** |
| 1 | Nomor urut | *TextView* | Nomor urut ayat dalam surat tertentu |
| 2 | Text ayat | *TextView* | Text ayat dalam huruf hijaiyah |
| 3 | Tombol rekam | *Button* | Tombol untuk masuk ke halaman Tahsin untuk merekam suara pengguna |
| 4 | Nama surat | *TextView* | Nama surat beserta nomor urutnya dalam Alquran |

### Perancangan Antarmuka Melihat Kesalahan Baca

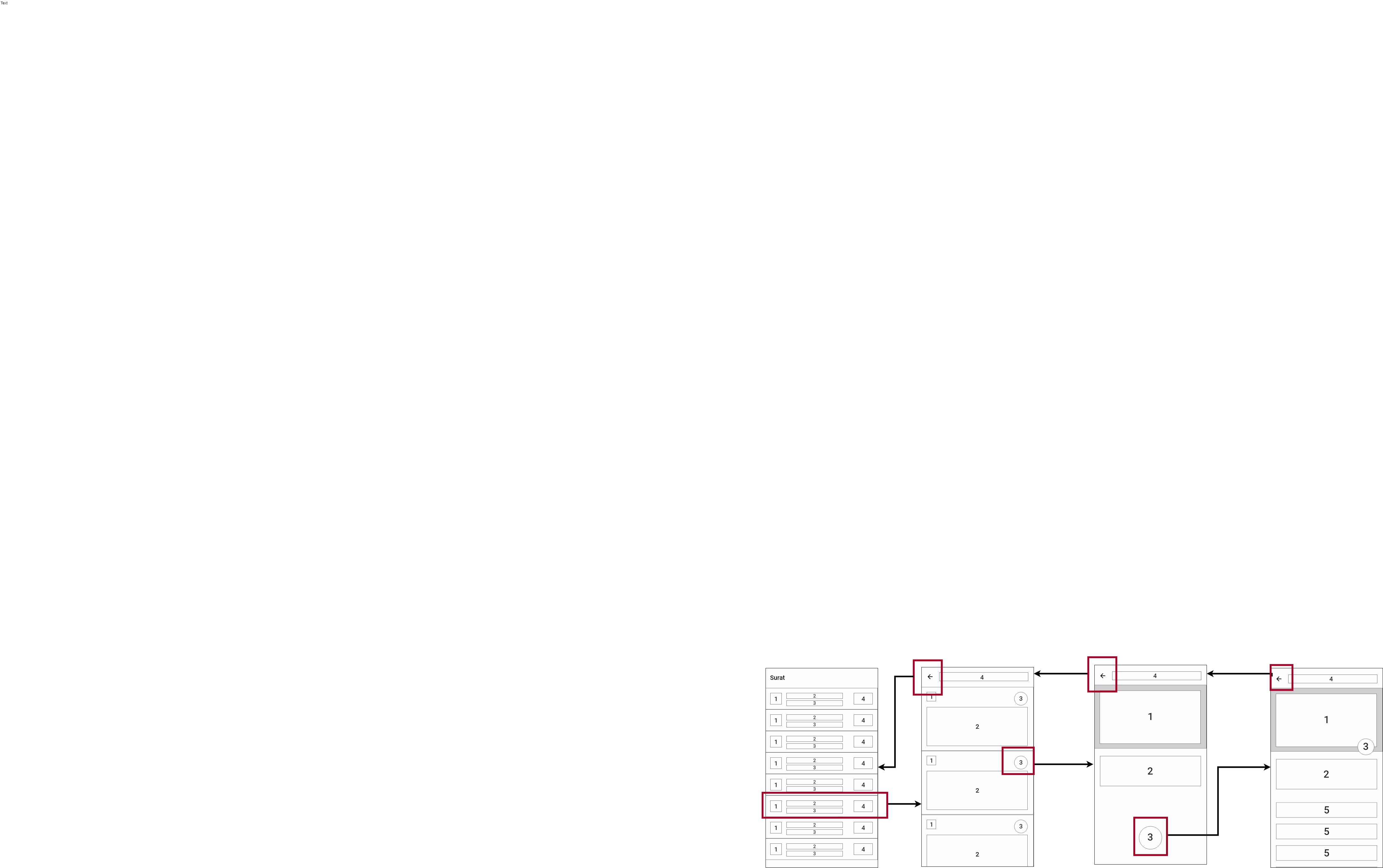
**Gambar ‎5.10 *Wireframe* Antarmuka Melihat Kesalahan Baca**

Pada gambar 5.10 adalah *Wireframe* dari *usecase* *scenario* nomor TQ-F-1-04 Melihat kesalahan baca. Tampilan untuk melihat kesalahan baca menggunakan halaman Tahsin. Pada tampilan ini, setelah pengguna membaca ayat dan merekam suaranya, aplikasi akan menampilkan teks hasil rekaman suara sekaligus hasil identifikasi kesalahan baca yang dilakunan pengguna. Antarmuka melihat kesalahan baca ini menampilkan teks ayat, teks hasil rekaman suara, nama surat, tombol rekam suara dan daftar kesalahan baca.

Tabel ‎5.4 Keterangan Antarmuka Melihat Kesalahan Baca

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Objek** | **Tipe** | **Keterangan** |
| 1 | Teks ayat | *TextView* | Teks ayat dalam huruf hijaiyah |
| 2 | Teks suara | *TextView* | Teks hasil terjemahan suara ke dalam Bahasa Arab |
| 3 | Tombol rekam | *Button* | Tombol yang ditekan dan ditahan untuk mulai merekam suara pengguna |
| 4 | Nama surat | *TextView* | Nama surat beserta nomor urutnya dalam Alquran |
| 5 | Kesalahan baca | *TextView* | Menampilkan pada bagian mana saja pengguna melakukan kesalahan baca |

### Perancangan *Screenflow* Aplikasi



Halaman

Daftar Surat

Halaman

Daftar Ayat

Halaman

Tahsin

Halaman

Tahsin

**Gambar ‎5.11 *Screenflow* Aplikasi**

Pada gambar 5.11 merupakan *screenflow* aplikasi yang terdiri dari 3 halaman berbeda yaitu Halaman Daftar Surat, halaman Daftar Ayat dan halaman Tahsin. Saat pengguna pertama membuka aplikasi, akan langsung masuk ke halaman Daftar surat. Kemudian pengguna memilih salah satu surat dan aplikasi akan menapilkan halaman Daftar ayat sesuai dengan surat yang dipilih pengguna. Pada halaman Daftar ayat pengguna akan menekan tombol rekam. Setelah itu, aplikasi akan membuka halaman Tahsin yang menampilkan ayat yang dipilih pengguna. Kemudian pengguna bisa mulai membaca ayat sambil melakukan perekaman suara dengan menekan tombol rekam di halaman Tahsin. Terakhir, aplikasi akan menampilkan daftar kesalahan baca pengguna di halaman Tahsin yang sama.

# IMPLEMENTASI

Dalam Bab Implementasi ini akan menjelaskan tentang bagaimana rancangan aplikasi pada bab sebelumnya diimplementasikan ke dalam perangkat bergerak di dunia nyata. Beberapa tahapan implementasi akan dibagi menjadi empat bagian, yaitu : batasan implementasi, spesifikasi lingkungan aplikasi, implementasi antarmuka dan implementasi kode program. Setiap tahapan tersebut akan dijelaskan dalam bab implementasi berikut.

## Spesifikasi Lingkungan Aplikasi

Spesifikasi lingkungan aplikasi mengacu pada rancangan arsitektur akan diimplementasikan melalui dua jenis spesifikasi, yaitu spesifikasi perangkat keras dan spesifikasi perangkat lunak.

### Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk aplikasi deteksi kesalahan baca Alquran ini ditunjukkan pada tabel 6.1.

Tabel ‎6.1 Spesifikasi Perangkat Keras Komputer

|  |  |
| --- | --- |
| Komponen | Spesifikasi |
| *Laptop* | Asus A455LN |
| *Processor* | Intel Core i5 5200 |
| *Memory* | 8096MB DDR3L |

Sementara untuk proses instalasi dan pengujian digunakan perangkat bergerak dengan system operasi Android yang memiliki spesifikasi seperti tertera pada tabel 6.2.

Tabel ‎6.2 Spesifikasi Perangkat Keras Perangkat Bergerak

|  |  |
| --- | --- |
| Komponen | Spesifikasi |
| *Smartphone* | Asus Zenfone 3 Max |
| *Processor* | MediaTek MT6737T 1.25GHz quad-core |
| *Memory* | 2048 MB |

### Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk implementasi aplikasi deteksi kesalahan baca Alquran ditunjukkan dalam tabel 6.3.

Tabel ‎6.3 Spesifikasi Perangkat Lunak

|  |  |
| --- | --- |
| Komponen | Spesifikasi |
| *Operating System Laptop* | Windows 10 64-bit |
| *Operating System Mobile* | Android 7.0 Nougat |
| Bahasa Pemrograman | Java, Kotlin |
| *Tools Development* | Android Studio |

## Batasan Implementasi

Aplikasi Deteksi Kesalahan Baca Alquran memiliki batasan dalam proses implementasi, yaitu :

1. Aplikasi hanya dapat diimplementasikan pada perangkat bergerak dengan sistem operasi Android minimal versi 4.1 *Ice Cream Sandwich* sampai Android versi terbaru.
2. Aplikasi hanya dapat di implementasikan pada perangkat Android yang memiliki sensor *microphone.*
3. Aplikasi hanya di implementasikan pada perangkat Android yang selalu terkoneksi dengan internet.
4. Aplikasi hanya menggunakan *webservice* dari *QuranAPI* dan *Speech Recognizer*.

## Implementasi Kode Program

Pada bagian implementasi kode program akan dijelaskan kode program yang telah digunakan guna menjalankan beberapa fungsi utama pada aplikasi. Semua yang dijelaskan akan mengacu kepada perancangan *usecase diagram* dalam gambar 4.2 dan *usecase scenario* pada Bab 4. Sementara kode program yang dijelaskan akan sepenuhnya mengacu kepada objek apa saja yang ada dalam *class diagram* pada gambar 5.7.

### Kode Program Melihat Daftar Surat

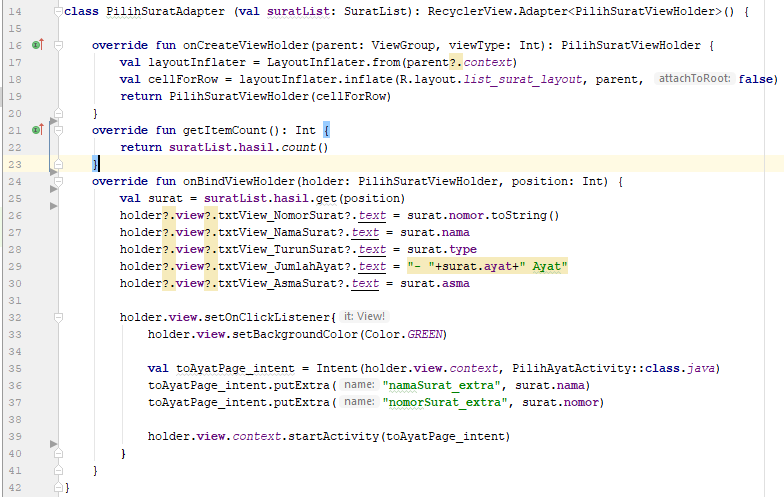
Implementasi kode program melihat daftar surat adalah kode program yang dijalankan ketika pengguna membuka aplikasi. Objek yang digunakan adalah *PilihSuratActivity.kt* sebagai objek *boundary, PilihSuratAdapter.kt* sebagai objek *control* dan *Surat.kt* sebagai objek model (entitas).



**Gambar ‎6.1 Kode Program PilihSurat*Activity*.kt**

Penjelasan Kode Program pada gambar 6.1 sebagai berikut :

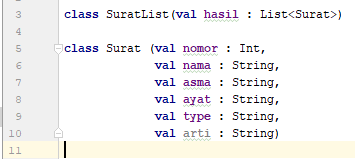
1. Baris 15-20, method onCreate() yang pertama kali dijalankan ketika PilihSurat*Activity* dipanggil.
2. Baris 22-40, method fetchJson() yang melakukan *request* daftar surat kepada QuranAPI dan memanggil objek PilihSurat*Adapter* pada baris 36 untuk menampilkan daftar surat ke dalam tampilan PilihSurat*Activity*.



**Gambar ‎6.2 Kode Program PilihSurat*Adapter*.kt**

Penjelasan Kode Program pada gambar 6.2 sebagai berikut :

1. Baris 16-20, method *onCreateViewholder*() yang dipanggil ketika sebuah *RecycleView* membutuhkan *Viewholder* baru untuk menampilkan data.
2. Baris 21-23, method *getItemCount*() digunakan untuk mengembalikkan jumlah data yang terdapat pada List.
3. Baris 24-41, method *onBindViewholder*() yang digunakan oleh *RecycleView* untuk mengatur posisi tampilan data.



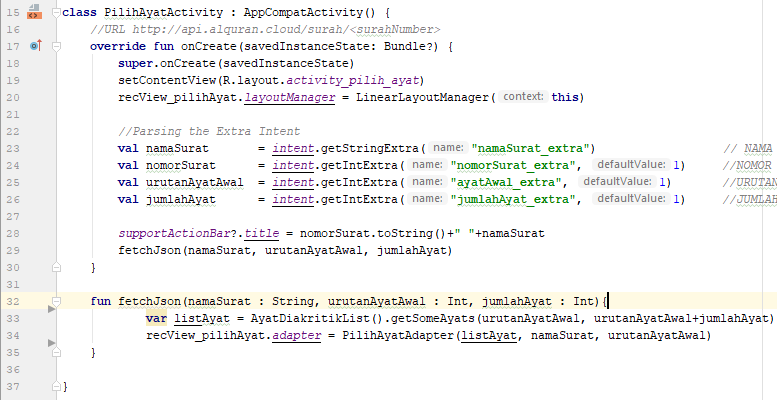
**Gambar ‎6.3 Kode Program Surat.kt**

Penjelasan Kode Program pada gambar 6.3 sebagai berikut :

1. Baris 3, class SuratList untuk menyimpan daftar dari surat-surat dalam Alquran.
2. Baris 5-10, class Surat sebagai objek yang menyimpan informasi tentang sebuah surat.

### Kode Program Melihat Daftar Ayat

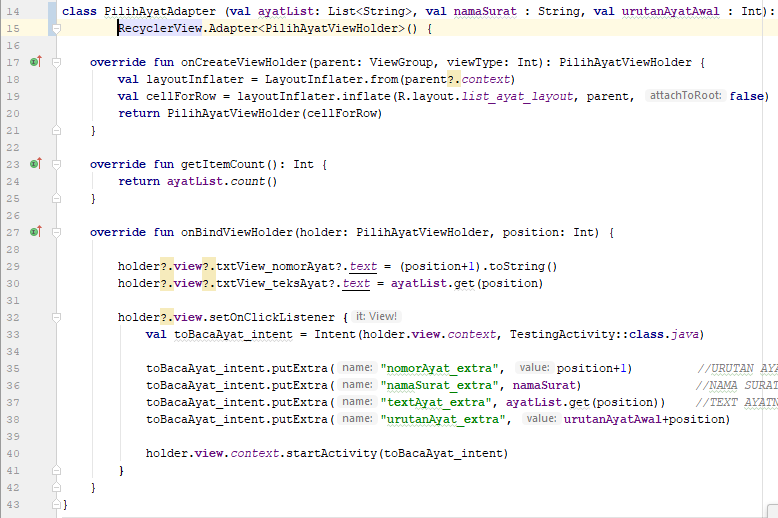
Implementasi kode program melihat daftar ayat adalah kode program yang dijalankan ketika pengguna memilih sebuah surat dari tampilan Pilih Surat. Objek yang digunakan adalah *PilihAyatActivity.kt* sebagai objek *boundary, PilihAyatAdapter.kt* sebagai objek *control* dan *Ayat.kt* sebagai objek model (entitas).



**Gambar ‎6.4 Kode Program PilihAyat*Activity*.kt**

Penjelasan Kode Program pada gambar 6.4 sebagai berikut :

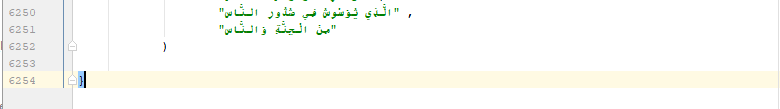
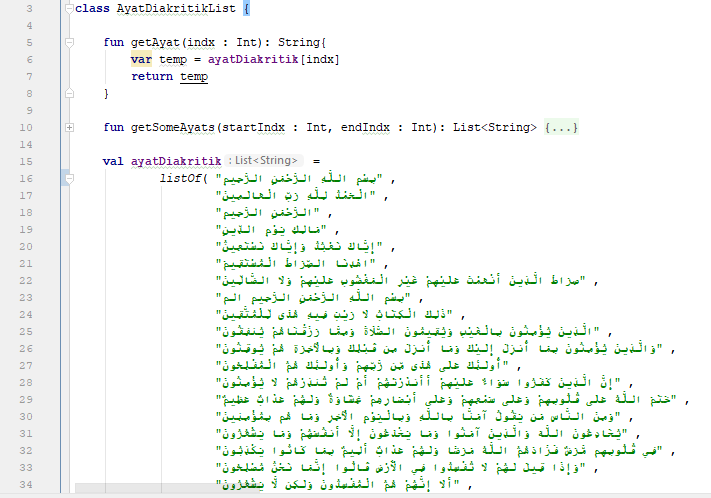
1. Baris 17-30, method *onCreate*() yang pertama kali dijalankan ketika *PilihAyatActivity* dipanggil.
2. Baris 32-34, method fetchJson() yang melakukan pengambilandaftar ayat dari listAyat dan memanggil objek PilihAyat*Adapter* pada baris 34 untuk menampilkan daftar ayat ke dalam tampilan PilihAyat*Activity*.



**Gambar ‎6.5 Kode Program PilihAyat*Adapter*.kt**

Penjelasan Kode Program pada gambar 6.5 sebagai berikut :

1. Baris 17-21, method *onCreateViewholder*() yang dipanggil ketika sebuah *RecycleView* membutuhkan *Viewholder* baru untuk menampilkan data.
2. Baris 23-25, method *getItemCount*() digunakan untuk mengembalikkan jumlah data yang terdapat pada List.
3. Baris 27-42, method *onBindViewholder*() yang digunakan oleh *RecycleView* untuk mengatur posisi tampilan data.



**Gambar ‎6.6 Kode Program AyatDiakritikList.kt**

Penjelasan Kode Program pada gambar 6.6 sebagai berikut :

1. Baris 5-8, *method* getAyat() untuk mengembalikkan satu buah ayat dari listAyat.
2. Baris 10, *method* getSomeAyats() untuk mengembalikkan daftar beberapa ayat diantara *startIndex* dan *endIndex*
3. Baris 15-6252, objek ayatDiakritik sebagai objek *list* yang menyimpan informasi tentang seluruh ayat dalam Alquran.

### Kode Program Melihat Kesalahan Baca

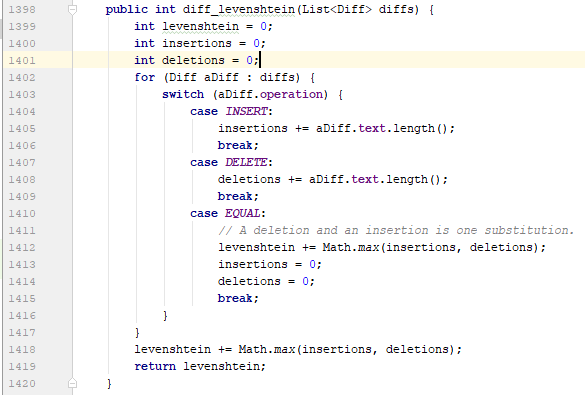
Implementasi kode program melihat kesalahan baca adalah kode program yang dijalankan ketika pengguna melakukan pembacaan ayat hingga aplikasi menampilkan kesalahan baca dari pengguna. Objek yang digunakan adalah *TahsinActivity.kt* sebagai objek *boundary* dan *diff\_match\_patch.kt* sebagai objek *control*.



**Gambar ‎6.7 Kode Program Tahsin*Activity*.kt**

Penjelasan Kode Program pada gambar 6.7 sebagai berikut :

1. Baris 75-76, inisiasi objek Spannable*String* dan tempIndx untuk melakukan pergantian *style* dari sebuah *String* guna menandakan lokasi kesalahan baca yang dilakukan pengguna.
2. Baris 78-96, perulangan dari setiap objek diff class untuk merubah *style* dari sebagian teks pada objek *diff* sesuai dengan kondisi kesalahan baca yang terjadi.
3. Baris 97-102, menampilkan objek ss Spannable*String* ke dalam *textView\_distanceResult* dan menampilkan daftar kesalahan baca ke dalam *textView\_resultText*.



**Gambar ‎6.8 Kode Program diff\_match\_patch.kt**

Penjelasan Kode Program pada gambar 6.8 sebagai berikut :

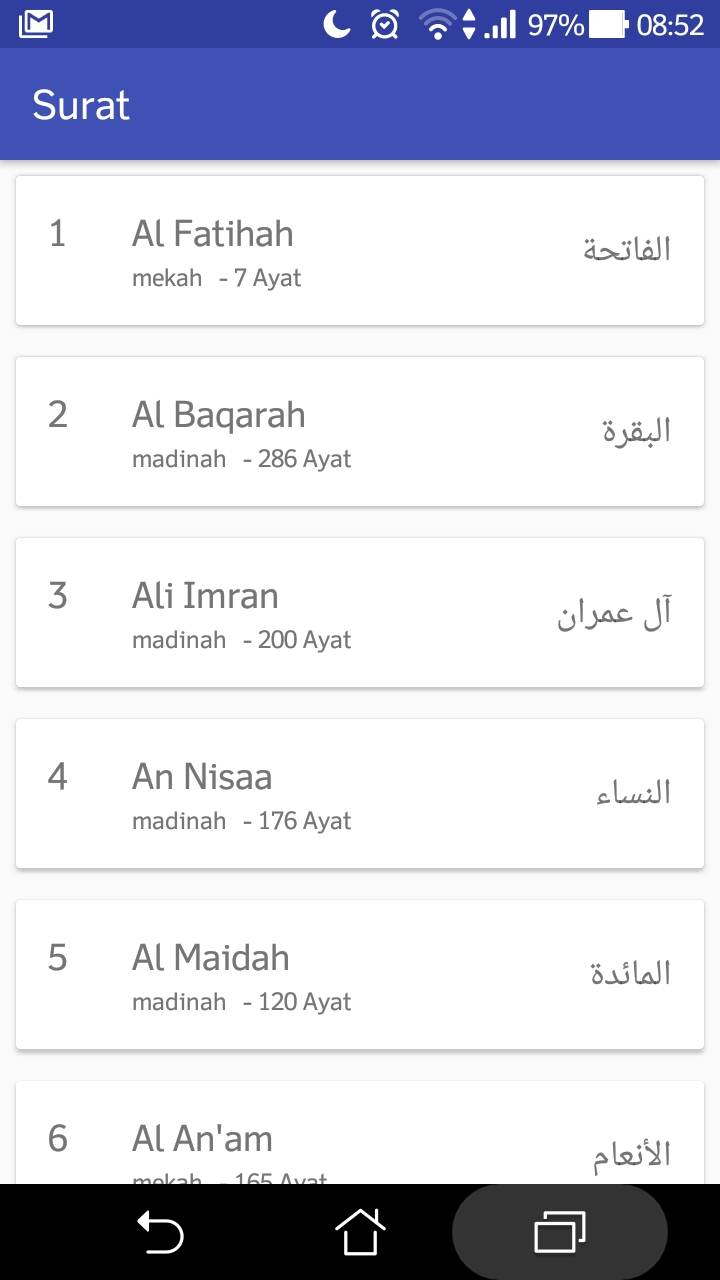
1. Baris 1398-1417, method diff\_levenstein untuk mengetahui *distance* atau banyaknya perbedaan Antara dua buah *string*. Operasi ini digunakan untuk menentukan seberapa banyak perubahan *string* yang dibutuhkan untuk menyamakan kedua buah *string* yang berbeda.

## Implementasi Antarmuka

Pada bagian Implementasi Antarmuka akan dijelaskan tentang hasil mengimplementasikan rancangan antarmuka pada bab 5 ke dalam aplikasi yang sesungguhnya. Hasil implementasi antarmuka dijelaskan pada gambar 6.9 - 6.11.

### Implementasi Antarmuka Melihat Daftar Surat

Implementasi antarmuka melihat daftar surat merupakan tampilan halaman pilih surat dari *PilihSuratActivity.* Tampilan ini merupakan tampilan awal dari aplikasi yang menampilkan seluruh daftar surat dalam Alquran yang berupa nomor urut surat, nama surat, tempat turunnya surat dan asma surat. Ketika pengguna menekan salah satu surat, maka akan di arahkan ke halaman pilih ayat.



**Gambar ‎6.9 Antarmuka Melihat Daftar Surat**

### Implementasi Antarmuka Melihat Daftar Ayat

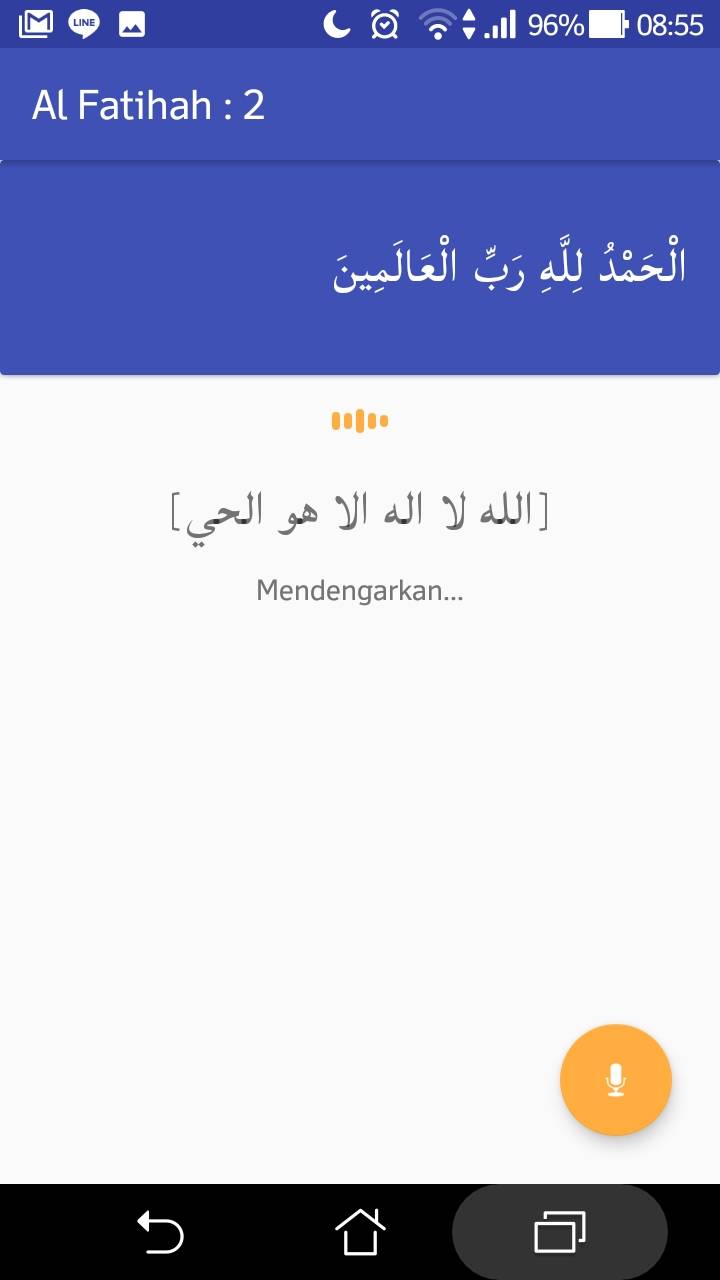
Implementasi antarmuka melihat daftar ayat merupakan tampilan halaman pilih ayat dari *PilihAyatActivity.* Halaman ini merupakan tampilan setelah pengguna memilih salah satu surat dari halaman Pilih Surat. Halaman pilih ayat menampilkan nomor urut ayat, nama surat, teks ayat dan tombol rekam suara. Ketika pengguna menekan tombol rekam suara dari salah satu ayat, maka akan di arahkan ke halaman tahsin.



**Gambar ‎6.10 Antarmuka Melihat Daftar Ayat**

### Implementasi Antarmuka Melihat Kesalahan Baca

Implementasi antarmuka melihat kesalahan baca merupakan tampilan halaman tahsin dari *TahsinActivity.* Tampilan ini menggunakan halaman Tahsin yang sama-sama digunakan juga untuk merekam suara pengguna. Pada tampilan ini, setelah pengguna membaca ayat dan merekam suaranya, aplikasi akan menampilkan teks hasil rekaman suara sekaligus hasil identifikasi kesalahan baca yang dilakunan pengguna. Antarmuka melihat kesalahan baca ini menampilkan teks ayat, teks hasil rekaman suara, nama surat, tombol rekam suara dan daftar kesalahan baca.

**Gambar ‎6.11 Antarmuka Menampilkan Kesalahan Baca**

# PENGUJIAN DAN ANALISIS HASIL

Dalam bab pengujian dan analisis ini akan dijelaskan tentang hasil pengujian dari aplikasi deteksi kesalahan baca Alquran beserta analisisnya. Pengujian yang dilakukan berdasarkan metodologi pada bab tiga yang meliputi pengujian kebutuhan fungsional dan pengujian kebutuhan non fungsional. Pengujian kebutuhan fungsional meliputi *Validation Testing.* Sedangkan pengujian kebutuhan non fungsional akan meliputi Pengujian *Usability* dan Pengujian Akurasi.

## *Validation Testing*

*Validation Testing* adalah sebuah teknik pengujian fungsional yang digunakan untuk menguji fitur atau fungsi dari sistem. Tujuan dari *validation testing* adalah untuk mengetahui apakah kebutuhan fungsional dari aplikasi sudah terpenuhi seluruhnya atau belum. Kebutuhan fungsional yang diuji menggunakan kasus uji sesuai dengan *use case scenario* pada bab empat.

### Kasus Uji Melihat Daftar Surat

Pada kasus uji melihat daftar surat ini dilakukan ketika pengguna membuka aplikasi.

Tabel ‎7.1 Kasus Uji Melihat Daftar Surat

|  |  |
| --- | --- |
| **Nomor Kasus Uji** | UJI-1 |
| **Nama Kasus Uji** | Melihat Daftar Surat |
| **Objek Uji** | *Usecase* TQ-F-1-01 |
| **Tujuan Pengujian** | Aplikasi dapat menampilkan daftar surat Alquran kepada pengguna |
| **Prosedur Pengujian** | Pengguna membuka aplikasi  Aplikasi menampilkan seluruh daftar 114 surat Alquran |
| **Hasil yang diharapkan** | Aplikasi menampilkan seluruh daftar 114 surat Alquran |
| **Hasil yang didapatkan** | Aplikasi menampilkan daftar 114 surat Alquran berserta nomor surat, nama surat, jumlah ayat dan tempat turunnya surat |
| **Status** | Valid |

### Kasus Uji Melihat Daftar Ayat

Pada kasus uji melihat daftar ayat ini dilakukan setelah pengguna memilih salah satu surat dari daftar surat yang ditampilkan pada halaman pilih surat di awal aplikasi.

Tabel ‎7.2 Kasus Uji Melihat Daftar Ayat

|  |  |
| --- | --- |
| **Nomor Kasus Uji** | UJI-2 |
| **Nama Kasus Uji** | Melihat Daftar Ayat |
| **Objek Uji** | *Usecase* TQ-F-1-02 |
| **Tujuan Pengujian** | Aplikasi dapat menampilkan seluruh ayat dari sebuah surat dalam Alquran |
| **Prosedur Pengujian** | Pengguna memilih salah satu surat Alquran  Aplikasi menampilkan seluruh ayat dari surat yang dipilih |
| **Hasil yang diharapkan** | Aplikasi menampilkan seluruh ayat dari surat yang dipilih |
| **Hasil yang didapatkan** | Aplikasi menampilkan semua ayat dari surat yang dipilih oleh pengguna |
| **Status** | Valid |

### Kasus Uji Melihat Kesalahan Baca

Pada kasus uji melihat kesalahan baca ini dilakukan setelah pengguna masuk ke halaman Tahsin dan membaca ayat yang ditampilkan.

Tabel ‎7.3 Kasus Uji Melihat Kesalahan Baca

|  |  |
| --- | --- |
| **Nomor Kasus Uji** | UJI-3 |
| **Nama Kasus Uji** | Melihat Kesalahan Baca |
| **Objek Uji** | *Usecase scenario* TQ-F-1-04 |
| **Tujuan Pengujian** | Aplikasi dapat menampilkan daftar kesalahan baca yang dilakukan pengguna |
| **Prosedur Pengujian** | Pengguna memilih salah satu ayat Alquran  Pengguna menekan tombol rekam  Pengguna membaca ayat Alquran  Pengguna telah selesai merekam suara  Aplikasi menampilkan daftar kesalahan baca yang dilakukan pengguna |
| **Hasil yang diharapkan** | Aplikasi menampilkan daftar kesalahan baca yang dilakukan pengguna.  [ALT] Aplikasi meminta pengguna menekan tombol rekam kembali |
| **Hasil yang didapatkan** | Aplikasi menampilkan daftar kesalahan baca yang dilakukan pengguna  [ALT] Aplikasi meminta pengguna menekan tombol rekam kembali |
| **Status pengujian** | Valid |

### Analisis Hasil *Validation Testing*

Berdasarkan *validation testing* yang telah dilakukan pada tabel 7.1 sampai tabel 7.3 didapatkan hasil rekapitulasi hasil pengujian fingsional yang dapat dilihat pada tabel 7.4**.**

Tabel ‎7.4 Rekapitulasi Hasil *Validation Testing*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Nama Kasus Uji** | **Nomor Kasus Uji** | **Status** |
| 1 | Melihat Daftar Surat | UJI-1 | Valid |
| 2 | Melihat Daftar Ayat | UJI-2 | Valid |
| 3 | Melihat Kesalahan Baca | UJI-3 | Valid |

Dari hasil rekapitulasi pada tabel 7.4 dapat ditarik kesimpulan bahwa hasil pengujian fungsional dari aplikasi deteksi kesalahan baca Alquran baru memenuhi 100% dari seluruh kebutuhan fungsional.

## Pengujian *Usability*

Pengujian *Usability* Aplikasi akan dilakukan dibagi menjadi dua tahap yaitu pengujian *usability*, kuisioner. Pengujian *Usability* diberikan dalam bentuk meminta 5 orang pengguna untuk menggunakan aplikasi selama 5 menit. Kemudian pengguna diminta untuk mengisi kuisioner *System Usability Scale (SUS)* sebanyak 10 pernyataan. Pernyataan SUS yang digunakan bersumber dari situs : measuring.com/sus yang ditulis oleh Jeff Sauro pada tahun 2011. Pernyataan SUS yang digunakan terlampir pada tabel 7.5.

Tabel ‎7.5 Pernyataan *System Usability Scale*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Pernyataan** | **Skor** | | | | |
| **Sangat Tidak Setuju** | **Tidak Setuju** | **Netral** | **Setuju** | **Sangat Setuju** |
| 1 | Saya ingin menggunakan aplikasi ini lagi |  |  |  |  |  |
| 2 | Saya merasa aplikasi ini kompleks |  |  |  |  |  |
| 3 | Saya merasa aplikasi ini mudah digunakan |  |  |  |  |  |
| 4 | Saya butuh bantuan teknis ketika menggunakan aplikasi ini |  |  |  |  |  |
| 5 | Saya merasa fungsi-fungsi dari aplikasi ini berjalan baik |  |  |  |  |  |
| 6 | Saya merasa tampilan aplikasi ini sering tidak konsisten |  |  |  |  |  |
| 7 | Saya merasa banyak orang yang akan mudah menggunakan aplikasi ini |  |  |  |  |  |
| 8 | Saya merasa aplikasi ini tidak praktis |  |  |  |  |  |
| 9 | Saya merasa percaya diri dalam menggunakan aplikasi ini |  |  |  |  |  |
| 10 | Saya harus mencoba berkali-kali untuk bisa memahami aplikasi ini |  |  |  |  |  |

### Analisis Hasil Kuisioner *System Usability Scale*

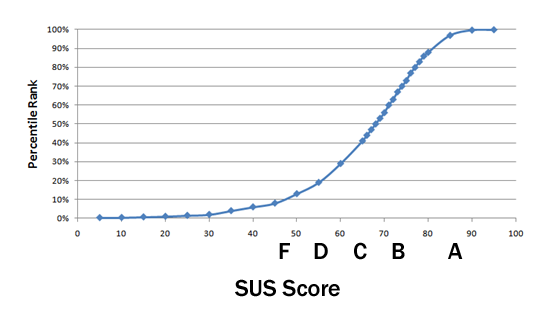
Pada bagian analisis hasil kuisioner *System Usability Scale (SUS)* ini akan menampilkan rekapitulasi skor SUS dari setiap responden. Skor untuk setiap pernyataan dengan angka ganjil, akan dihitung menggunakan rumus . Sementara itu, untuk setiap urutan pernyataan dengan angka genap, akan dihitung menggunakan rumus ( dengan ketentuan bahwa adalah angka pada skala pernyataan yang dipilih responden. Maka hasil dari skala kuisioner SUS dengan 5 responden tertera pada tabel 7.6

Tabel ‎7.6 Rekapitulasi Skor *System Usability Scale*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Responden** | **Skor tiap pernyataan** | | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **1** | 4 | 4 | 4 | 4 | 2 | 4 | 3 | 3 | 4 | 4 |
| **2** | 3 | 3 | 3 | 4 | 3 | 2 | 2 | 2 | 2 | 4 |
| **3** | 4 | 3 | 3 | 1 | 3 | 1 | 4 | 4 | 4 | 2 |
| **4** | 4 | 2 | 3 | 4 | 2 | 2 | 2 | 4 | 2 | 1 |
| **5** | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 3 | 4 |

Setelah setiap pernyataan bernomor ganjil dan bernomor genap dihitung menggunakan rumus tersebut, selanjutnya adalah menjumlahkan semua skor dari setiap pernyataan lalu hasil perjumlahan dikalikan dengan 2,5. Kemudian seluruh nilai SUS yang telah didapatkan dari setiap responden akan dihitung nilai rata-ratanya (Sauro, 2011). Maka hasil dari nilai SUS yang didapatkan pada pengujian adalah sebesar :

Hasil nilai SUS sementara yang didapatkan adalah 76. Jika disandingkan dengan pendapat Jeff Sauro, maka aplikasi ini mendapatkan *grade B+* (di atas rata-rata)karena berada pada *range 75-80* sesuai grafik nilai SUS pada gambar 7.1



**Gambar ‎7.1 *System Usability Scale Letter Grade***

### Analisis Hasil Wawancara

Pada tahap wawancara yang akan dilakukan kepada lima responden yang telah menggunakan aplikasi dan telah selesai mengisi kuesioner SUS. Pertanyaan akan dibagi menjadi lima kategori yaitu pertanyaan yang berkaitan dengan tampilan aplikasi, fungsional aplikasi, output aplikasi dan performa aplikasi. Tujuan dari wawancara adalah untuk mengetahui *pain points* yang dirasakan pengguna ketika menggunakan aplikasi. Rekapitulasi hasil wawancara akan langsung penulis sampaikan dalam bentuk poin-poin yang menjadi ketidaknyamanan pengguna ketika menggunakan aplikasi seperti yang terlampir pada tabel 8.1 pada Bab Lampiran di akhir penelitian.

## Pengujian Akurasi

Pada pengujian akurasi dilakukan dengan cara meminta pengguna membaca beberapa ayat dari Alquran. Dengan metode *Equivalent Partitioning Testing,* klasifikasi cara membaca ayat dibagi ke dalam kategori bacaan benar, bacaan kekurangan huruf, bacaan kelebihan huruf dan bacaan ayat yang berbeda.

Tabel ‎7.7 Klasifikasi Pengujian Akurasi

|  |  |  |  |
| --- | --- | --- | --- |
| **Ke** | | | |
| **Bacaan benar** | **Kekurangan huruf** | **Kelebihan huruf** | **Ayat berbeda** |
| Ayat dibaca dengan benar | Ayat dibaca dengan mengurangi beberapa huruf/kata | Ayat dibaca dengan melebihkan beberapa huruf/kata | Ayat yang dibaca berbeda |

Pengujian dilakukan dengan menggunakan ayat 5 dari surat Al-Bayyinah. Pengujian dinyatakan valid apabila aplikasi berhasil mendeteksi bacaan yang benar, kekurangan bacaan, kelebihan bacaan dan ayat berbeda yang di baca. Hasil validasi pengujian akurasi seperti yang terdapat pada tabel 7.8.

### Analisis Hasil Pengujian Akurasi

Berdasarkan klasifikasi pengujian akurasi pada tabel 7.7. Penulis dapat memvalidasi pengujian pada tabel 7.8 dan menyimpulkan hasil pengujian pada tabel 7.9.

Tabel ‎7.8 Validasi Hasil Pengujian Akurasi

|  |  |  |
| --- | --- | --- |
| **Klasifikasi Bacaan** | **Screenshot Validasi** | **Kesimpulan** |
| Bacaan benar |  | Pengujian **tidak valid**, karena aplikasi mendeteksi kesalahan baca, padahal tidak ada kesalahan baca.  Dikarenakan kesalahan *Speech Recognizer* menerjemahkan huruf ة yang dibaca pengguna sebagai huruf ه |
| Kekurangan huruf |  | Pengujian **valid**, karena kekurangan huruf terdeteksi sebesar 48,39%. |
| Kelebihan huruf |  | Pengujian **valid**, karena kelebihan huruf terdeteksi sebesar 19,39%. |
| Ayat berbeda |  | Pengujian **valid**, karena bacaan ayat terdeteksi sebagai ayat yang berbeda. |

Dari hasil analisis pengujian akurasi pada tabel 7.8 diketahui bahwa dari 4 kategori klasifikasi pengujian yang dilakukan, terdapat 3 pengujian valid dan 1 pengujian tidak valid.

Tabel ‎7.9 Rekapitulasi Hasil Pengujian Akurasi

|  |  |
| --- | --- |
| **Klasifikasi Bacaan** | **Hasil Pengujian** |
| Bacaan benar | Tidak Valid |
| Kekurangan huruf | Valid |
| Kelebihan huruf | Valid |
| Ayat Berbeda | Valid |

Ada beberapa faktor yang menyebabkan beberapa pengujian akurasi tidak valid yaitu :

*Speech Recognizer* salah menerjemahkan suara pengguna, ketika pengguna membaca huruf **ة** namun diterjemahkan sebagai huruf **ه .**

1. Aplikasi mendeteksi kelebihan ‘spasi’ sebagai kesalahan baca, padahal ‘spasi’ tidak termasuk dalam bacaan Alquran.

# PENUTUP

Pada Bab Penutup ini memuat kesimpulan dan saran dari hasil pengembangan Aplikasi Pendeteksi Kesalahan Baca Alquran berdasarkan pemaparan dari bab-bab sebelumnya, yang diharapkan dapat membantu pengembangan selanjutnya.

## Kesimpulan

Berdasarkan dari hasil pada bab analisis kebutuhan, bab perancangan, bab implementasi dan bab pengujian yang dilakukan, maka dapat diperoleh kesimpulan sebagai berikut :

1. Pada tahap analisis kebutuhan Aplikasi Deteksi Kesalahan Baca Alquran telah modelkan sesuai dengan hasil wawancara kepada calon pengguna dan menghasilkan 3 kebutuhan fungsional berupa memilih surat Alquran, memilih ayat Alquran dan melihat kesalahan baca.
2. Pada tahap perancangan Aplikasi Deteksi Kesalahan Baca Alquran dilakukan dengan metode *Object Oriented Design (OOD)* yang menggunakan diagram UML Activity Diagram, *Class Diagram* dan *Sequence Diagram* untuk perancangan arsitektur beserta metode *Wireframing* untuk perancangan antarmuka.
3. Pada tahap implementasi digunakan layanan *Speech Recognizer* untuk menerjemahkan suara pengguna ke dalam teks Bahasa Arab. Algoritme *Levenstein Distance* digunakan untuk menghitung banyaknya kesalahan baca dan library *Diff Match Pacth* digunakan untuk mendeteksi posisi kesalahan baca pengguna. Sementara implementasi antarmuka disesuaikan dengan *prototype Wireframe* yang ada pada bab perancangan.
4. Pada tahap pengujian akurasi ditemukan bahwa Aplikasi Deteksi Kesalahan Baca Alquran ini dinyatakan valid pada 3 kasus uji dan tidak valid pada 1 kasus uji. Pengujian yang valid adalah ketika aplikasi berhasil mendeteksi kekurangan huruf, kelebihan huruf dan perbedaan ayat yang dibaca. Sementara pengujian yang tidak valid ketika aplikasi mendeteksi kesalahan baca padahal pengguna sudah membaca ayat dengan benar. Hal tersebut dikarenakan kesalahan penerjemahan dari *Speech Recognizer* yang tidak bisa diperbaiki dari sisi penelitian ini.

## Saran

Saran yang diberikan untuk pengembangan lanjut Aplikasi Deteksi Kesalahan Baca Alquran ini adalah sebagai berikut :

1. Menggunakan *Speech Recognizer* untuk menerjemahkan suara ke dalam Bahasa Arab terbukti memberikan hasil yang cukup akurat. Namun belum cukup akurat agar bisa dikatakan layak digunakan untuk menerjemahkan bacaan Alquran karena *Speech Recognizer* tidak mendukung penggunaan diakritik (harkat) pada teks Bahasa Arab. Disarankan pada penelitian berikutnya *Speech Recognizer* digantikan dengan metode *Diacritics Engine* pada penelitian Hassan Al-Haj yang berjudul “*Pronunciation Modeling for Dialectal Arabic Speech Recognition*”.
2. Penggunaan *Diacritics Engine* memungkinkan algoritme untuk menerjemahan suara ke dalam teks Bahasa Arab bisa di *training* terus-menerus guna mendapatkan hasil yang lebih akurat. Di samping itu, penggunaan *Diacritics Engine* sudah mendukung penggunaan harkat pada tulisan Bahasa Arab.
3. Untuk meningkatkan kepuasan pengguna, pada penelitian selanjutnya disarankan untuk memperbaiki *pain points* dari aplikasi yang terlampir pada bab Lampiran di akhir penelitian.

DAFTAR PUSTAKA

Adriyani, Ni Made Murni. 2012. *Implementasi Algoritme Levenshtein Distance dan Metode Empiris Untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Dokumen Berbahasa Indonesia*. Bali : JELIKU - Jurnal Elektronik Ilmu Komputer Udayana.

Al-Haj, Hassan. 2009. *Pronunciation Modeling for Dialectal Arabic Speech Recognition*. Pittsburg : Language Technologies Institute School of Computer Science, Carnegie Mellon University.

Android Developers. 2018. *SpeechRecognition.* [online]. Android Developers. Tersedia di : < https://developer.android.com/reference/android/speech/SpeechRecognizer> [Diakses tanggal 31 Agustus 2018]

Azzam, Muhammad. 2017. Pengertian Android Beserta Kelebihan dan Kekurangannya. [online] Nesabamedia. Tersedia di : <https://www.nesabamedia.com/pengertian-android-beserta-kelebihan-dan-kekurangannya/> [Diakses 17 Oktober 2018]

Farouqi, Muhammad Ismali. 2018. Evaluasi *Usability* pada Aplikasi Go-Jek Dengan Menggunakan Metode Pengujian *Usability*. Malang : JPTIIK.

Fauzan, Ach., Arwani, Issa., Fanani, Luthfi., 2017. *Pembangunan Aplikasi Iqro’ Berbasis Android Menggunakan Google Speech*. Malang: ABC Press.

Felix, Tani. 2015. Penelitian Kualitatif #015: Cara Memilih Subyek Tineliti. Kompasiana. [online]. Tersedia di : < https://www.kompasiana.com/mtf3lix5tr/55284dadf17e61d6398b4574/penelitian-kualitatif-015-cara-memilih-subyek-tineliti> [Diakses tanggal 17 Oktober 2018]

Fraser, Neil. 2018. *The Diff Match and Patch Libraries*. Github. [online]. Tersedia di : <https://github.com/google/diff-match-patch> [Diakses tanggal 10 Desember 2018]

Grabianowski, Ed . 2006 . *How Speech Recognition Works .* How Stuff Works. [online] . Tersedia di : <https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition2.htm> [Diakses tanggal 4 Januari 2019]

Islamic Network . 2018 . *Alquran Cloud* . Github . [online]. Tersedia di : <https://alquran.cloud/api> [Diakses tanggal 4 Januari 2019]

Kaur, Kalwinder . 2013 . *Microphone Sound Sensor .* Azo Sensors . [online ] . Tersedia di : <https://www.azosensors.com/article.aspx?ArticleID=229> [Diakses tanggal 4 Januari 2019]

Kementrian Pendidikan dan Kebudayaan , 2018. Kamus Besar Bahasa Indonesia Daring. [online] KBBI Daring. Tersedia di : <http://kbbi.kemdikbud.go.id/entri/Alquran> [Diakses 8 Maret 2018]

Khasanah, Amalia., 2013. *Perancangan Aplikasi Al Qur’an Menggunakan Voice Recognition Sebagai Media Pencarian Terjemahan Al Qur’an Berbasis Android*. Yogyakarta: Sekolah Tinggi Manajemen Informatika Dan Komputer Amikom Yogyakarta.

Milette, Greg. 2017. *Using the Microphone*. O’Reilly . [online]. Tersedia di <https://www.oreilly.com/library/view/professional-android-sensor/9781118240458/9781118240458c14.xhtml> [Diakses tanggal 17 Oktober 2018]

Nielsen, J. 2000. *Why You Only Need to Test with 5 Users*. Nielsen Norman Group . Tersedia di : <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users> [Diakses 21 November 2018]

Patton, M.Q. 1990. *Qualitative Evaluation and Research Methods* (2nd ed.). Newbury Park, CA: Sage, 532 pp.

Prastyo, Azis Yudi. 2009. Implementasi dan Analisis Literatur. Fasilkom UI. [online]. Tersedia di : <http://lontar.ui.ac.id/file?file=digital/123769-SK-721-Implemetasi%20dan%20analisis-Literatur.pdf> [Diakses tanggal 5 November 2018]

Rajkumar. 2018. *Equivalence Partitioning Test Case Design Technique*. [online] Software Testing Material. Tersedia di : <http://www.softwaretestingmaterial.com/equivalence-partitioning-testing-technique//> [Diakses 12 Desember 2018]

Sauro, Jeff. 2011. *Measuring Usability With The System Usability Scale (Sus)*. Measuring U . Tersedia di : <https://measuringu.com/sus/> [Diakses 7 Desember 2018]

Shihab, Quraish. 1994. *Membumikan Alquran : Fungsi & Peran Wahyu dalam Kehidupan Masyarakat .* Bandung : Mizan Pustaka.

Sommerville, Ian. 2011. *Software Engineering 9th Edition*. London : Pearson

Wahyudianta, Imam. 2016. Dua Pelajar ini Buat Aplikasi Pengoreksi Kesalahan Membaca Alquran. [online] Detikinet. Tersedia di : <https://inet.detik.com/cyberlife/d-3300096/dua-pelajar-ini-buat-aplikasi-pengoreksi-kesalahan-membaca-alquran/> [Diakses 2 September 2018]

Zarkasyi, Dachlan Salim. 2015. Keutamaan Membaca Al Qur’an Dengan Baik dan Benar, Serta Menghafalnya. [online] Lembaga Qiroati Pusat Semarang. Tersedia di : <<http://www.qiroatipusat.or.id/2015/06/keutamaan-membaca-al-quran-dengan-baik.html>> [Diakses 12 Agustus 2018]

LAMPIRAN

Lampiran 1 - Rekapitulasi Hasil Wawancara Pengujian Usability

Rekapitulasi *pain points* dari hasil wawancara pengguna :

Tabel ‎8.1 Rekapitulasi Hasil Wawancara

|  |  |
| --- | --- |
| **Kategori** | ***Pain points*** |
| Tampilan | 1. Tampilan tombol rekam di halaman pilih ayat dikira langsung rekam di tempat 2. Tidak ada arti ayat dalam Bahasa Indonesia 3. Pada ayat pertama di tiap surat masih tergabung antara ayat dengan bacaan basmalah, sehingga membingungkan. 4. *Background* untuk membaca surat lebih nyaman berwarna putih 5. Pesan *error* yang ditampilkan aplikasi masih sulit dimengerti pengguna 6. Durasi tampilan rekam suara dibuat lebih lama |
| Fungsional | 1. Merasa bacaan sudah benar, namun disalahkan 2. Tidak ada fitur pencarian surat 3. Bingung darimana hasil presentasi persen kesalahan 4. Tampilan daftar kesalahan baca dikira bisa ditekan dan menunjukkan lokasi kesalahan baca 5. Tidak ada panduan penggunaan aplikasi 6. Ingin tombol untuk merekam suara dibuat *hold to speak* 7. Tidak ada klasifikasi kesalahan yang dideteksi 8. Saran untuk menambahkan fitur untuk tambah audio ke dalam aplikasi |
| *Output* | 1. Bingung apa artinya ‘Hapus’ & ‘Tambah’ 2. Membutuhkan *feedback*, bagaimana bacaan yang benar 3. Output dari sistem membingungkan, karena huruf arabnya gundul |
| Performa | 1. Huruf yang tampil secara *realtime* ketika pengguna membaca ayat masih agak lambat |