Implementasi Algoritma *Rabin Karp* Pada Aplikasi Muroja'ah Hafalan Al-Qur'an Menggunakan *Google Speech API* Berbasis *Android*

Muhammad Khoir Al Alim Manurung[1], Samsudin[2], Ali Ikhwan[3]
[1,2,3]Prodi Sistem Informasi Universitas Islam Negeri Sumatera Utara
e-mail: ***[1]khoiralalim@gmail.com**, [2]samsudin@uinsu.ac.id, [3]aliikhwan053@gmail.com

Abstrak

Aplikasi muroja'ah hafalan Al-Qur'an merupakan sebuah aplikasi berbasis *android* yang digunakan untuk membantu penghafal Al-Qur'an dalam melakukan muroja'ah hafalan. Aplikasi dibangun menggunakan teknologi *Speech Recognition in Artificial Intelligence* (AI) dengan mengimplementasikan *Google Speech API* (*Application Programming Interface*) dan Algoritma *Rabin Karp* untuk mengubah ucapan pengguna menjadi sebuah *string* yang kemudian dibandingkan dengan *string* ayat yang benar. Adapun metode pengumpulan data, penulis menggunakan metode penelitian kualitatif, dengan melakukan observasi, wawancara dan penelitian kepustakaan. Aplikasi yang dibangun berhasil menerapkan fitur *Google Speech API* beserta Algoritma *Rabin Karp* pada aplikasi dengan *output* menampilkan informasi mengenai status tepat atau tidaknya beserta persentase akurasi kesamaan antara data ayat yang benar dengan suara yang diucapkan pengguna.

Kata kunci : Muroja'ah, Algoritma Rabin Karp, *Artificial Intelligence*, *Speech Recognition*, *Google Speech*

Abstract

The muroja'ah application in Al-Qur'an memorizing is an android-based application that is used to help memorizers of the Al-Qur'an in doing muroja'ah memorization. The application is built using Speech Recognition in Artificial Intelligence (AI) technology by implementing the Google Speech API and the Rabin Karp Algorithm to convert the user's speech into a string which is then compared with the correct verse string. As for data collection methods, the author uses qualitative research methods, by conducting observations, interviews, and literature research. The application that was built successfully implemented the Google Speech API feature along with the Rabin Karp Algorithm with the output displaying information about the correct status or not along with the percentage accuracy of the similarity between the correct verse data and the voice spoken by the user.

***Keywords** : Muroja'ah, Rabin Karp Algorithm, Artificial Intelligence, Speech Recognition, Google Speech*

1. PENDAHULUAN

Pada era revolusi 4.0 saat ini, ilmu pengetahuan dan teknologi mengalami perkembangan yang sangat pesat. Yang diiringi dengan meningkatnya kebutuhan manusia, menjadikan teknologi sebagai salah satu kebutuhan penting dalam mendukung berbagai aktivitas manusia secara kompleks. Hal ini dapat dilihat dari munculnya berbagai bidang ilmu pengetahuan dan teknologi yang baru seperti *artificial intelligence* (AI) atau kecerdasan buatan, *mixed reality*, *speech recognition*, *machine learning*, kecerdasan buatan dan lain sebagainya. Bidang ilmu tersebut dikembangkan kedalam sebuah sistem atau inovasi teknologi yang dapat memberikan kemudahan dalam hal pekerjaan manusia maupun dalam menyelesaikan suatu permasalahan

tertentu dengan lebih baik. Salah satu kemajuan teknologi ini ditandai dengan munculnya smartphone dalam berbagai versi terbarukan. *Smartphone* adalah telepon pintar yang memiliki kemampuan *mobile computing*, dan memiliki fitur-fitur pendukung dalam sebuah aplikasi untuk mendukung berbagai keperluan manusia serta memudahkan bagi penggunanya, termasuk dalam hal menghafal dan *muroja'ah* hafalan Al-Qur'an.

Dalam praktik menghafal ayat-ayat Al-Qur'an diperlukan sebuah teknik mengulang ayat-ayat yang dihafalkan dan kemudian disetorkan kepada seorang *mudabbir*, guru maupun orang lain. Dalam melakukan hal tersebut, seorang penghafal Al-Qur'an atau biasa disebut dengan seorang *hafizh* melantunkan ayat hafalannya kepada orang lain untuk dapat dikoreksi apakah hafalannya sudah tepat atau bahkan masih terdapat kesalahan. Jika belum tepat dan masih ada kesalahan maka penghafal akan lebih mudah dalam memperbaiki hafalannya sampai fasih. Cara tersebut dinilai kurang efektif karena seorang *hafizh* membutuhkan dan mencari terlebih dahulu orang yang bisa mengkoreksi hafalannya, sebab jika mengulang hafalan secara mandiri memungkinkan terdapat kesalahan yang tidak disadari. Berdasarkan hal tersebut maka penulis ingin memfasilitasi seorang *hafizh* maupun masyarakat umum dalam menambah dan mengkoreksi hafalannya secara mandiri. Dengan memanfaatkan dan mengembangkan fitur pengenalan ucapan (*speech recognition)* pada *smartphone* menggunakan *Google Speech API*. Pada aplikasi yang dibangun, pengguna dapat mengucapkan sebuah ayat kemudian aplikasi akan mencerna ayat tersebut dan akan diterjemahkan kedalam sebuah *string*. Setelah itu ayat yang diucapkan oleh pengguna akan dibandingkan dengan *database* ayat-ayat yang benar, dalam membandingkannya penulis menerapkan algoritma *Rabin Karp*.

Adapun tujuan pada penelitian ini adalah membangun aplikasi untuk *muroja'ah* hafalan Al-Qur'an menggunakan *Google Speech API* dengan menerapkan algoritma *Rabin Karp* dalam membandingkan ayat yang diucapkan oleh pengguna dengan data ayat yang benar, apakah sudah benar atau belum tepat beserta persentase tingkat persamaannya (*similarity*). Sehingga aplikasi dapat digunakan oleh masyarakat sebagai media untuk melakukan *muroja'ah* hafalan secara mandiri dengan memanfaatkan *smartphone* mereka.

2. METODE PENELITIAN

Adapun metode pengembangan sistem menggunakan metode pengembangan RAD (*Rapid Application Development*). Sedangkan metode pengumpulan data yang dilakukan adalah menggunakan metode penelitian kualitatif, dengan melakukan observasi dan pengamatan secara langsung dengan teknik pengumpulan data berdasarkan hasil wawancara dan penelitian kepustakaan. Adapun observasi yang penulis lakukan adalah dengan melakukan wawancara terhadap *ustadz* dan santri di Pondok Pesantren *Darul Huffazh Al Arief* yang berlokasi dijalan Bejo Gang Bambu II No.5, Bandar Khalifah, Deli Serdang, Sumatera Utara dan juga melakukan wawancara dengan masyarakat umum khususnya generasi milenial yang sedang menghafal Al-Qur'an. Adapun teknik pengumpulan data, penulis melakukan wawancara terhadap penghafal dengan tujuan melakukan analisis kebutuhan dalam *muroja'ah* hafalan Al-Qur'an. Kemudian penulis mewawancarai guru atau *ustadz* sebagai ahli dalam bidang *tahfizh* Al-Qur'an guna mendukung penelitian ini. Selain melakukan teknik wawancara penulis juga melakukan penelitian kepustakaan yaitu studi literatur dengan mencari data dan informasi yang bersumber dari *website*, jurnal maupun buku yang mendukung materi penelitian.

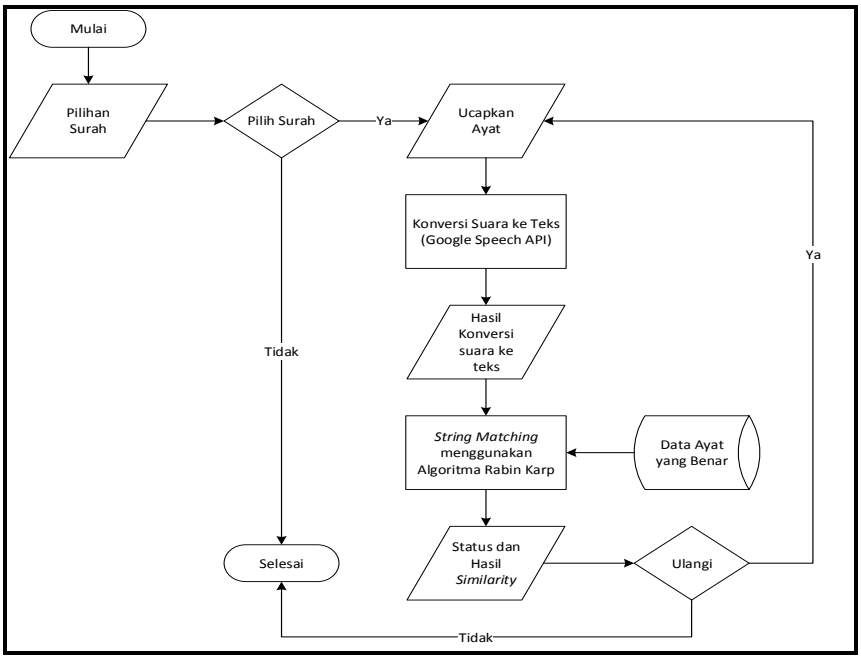
3. HASIL DAN PEMBAHASAN

Aplikasi *muroja'ah* hafalan Al-Qur'an merupakan aplikasi berbasis android untuk digunakan dalam membantu pengguna dalam menghafal Al-Qur'an dengan melakukan *muroja'ah* hafalan Al-Qur'an. Metode pengembangan sistem yang dibangun pada penelitian ini menggunakan metode pengembangan *RAD* (*Rapid Application Development*). RAD adalah

salah satu model dari SDLC (*System Development Lifecycle*) yang merupakan sebuah model dalam proses pengembangan perangkat lunak secara *linear sequential* yang menekankan pada siklus pengembangan yang lebih cepat[1]. Adapun tahapan-tahapan[2] yang dilakukan dalam pengembangan sistem yang dibangun adalah seperti berikut:

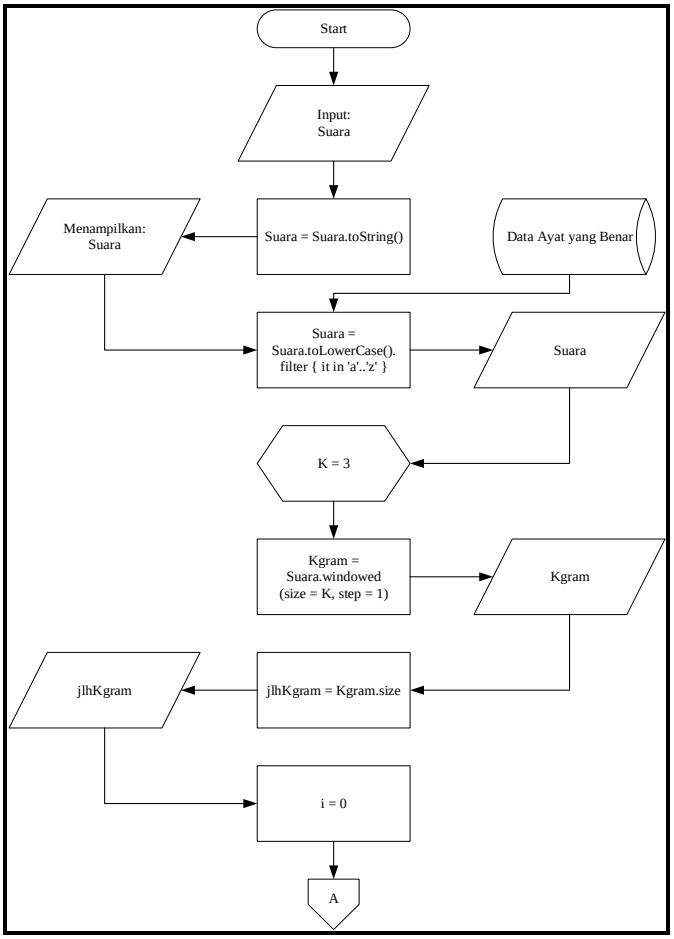
a. *Requirements planning*
   Dalam fase ini, *user* dan *system analyst* mengidentifikasikan tujuan dan kebutuhan-kebutuhan dari sistem yang akan dibangun[3]. Dalam tahapan ini penulis bertujuan membangun aplikasi *muroja'ah muroja'ah* hafalan Al-Qur'an untuk memenuhi kebutuhan masyarakat khususnya para *hafizh* dalam menambah hafalan Al-Qur'an melalui aplikasi yang dibangun. Dalam aplikasi tersebut tersedia fitur *muroja'ah* hafalan ayat Al-Qur'an menggunakan *Google Speech API*. *Google Speech Application Programming Interface* (Google Speech API) adalah sebuah layanan pengembangan pada Model *Machine Learning*, dengan penggabungan *Google Translate Application Programming Interface* dan *Cloud Vision Application Programming Interface.* Pemanfaatannya dapat diterapkan untuk mengembangkan sebuah aplikasi dengan menggunakan teknologi dari *speech recognition*.[4]
b. *RAD design workshop*
   Setelah mengetahui tahap *requirements planning*, penulis melakukan desain perancangan dengan membuat pemodelan terhadap aplikasi.[5] Desain yang dimaksud meliputi perancangan proses aplikasi dan desain *interface*. Dalam perancangan sistem, penulis menggunakan *flowchart,* diagram UML dan *Stroyboard. Flowchart* adalah cara untuk menjelaskan tahap-tahapan dalam suatu pemecahan masalah dengan merepresentasikan symbol-simbol tertentu yang mudah dipahami[6]. Sedangkan diagram UML adalah bahasa visual yang digunakan untuk memodelkan dan mengkomunikasikan sistem melalui diagram dan teks pendukung. Pemodelan UML mencakup berbagai model, seperti *use case* diagram, *class* diagram, *activity* diagram, dan *sequence* diagram.[7].
c. *Implementation*
   Dalam fase ini akan dilakukan implementasi aplikasi yang terdiri dari:
   1. Membangun Aplikasi
      Dalam tahapan ini aplikasi dirancang menggunakan Android Studio menggunakan bahasa pemrograman *Kotlin* dan *Google Speech API*.
   2. Menguji Aplikasi
      Pada tahapan ini penulis melakukan *testing* terhadap keseluruhan aplikasi yang dikembangkan dengan menggunakan metode *blackbox testing* untuk mengetahui keberhasilan dari aplikasi yang dibangun.

Adapun aplikasi yang dibangun berisikan *list* surah Al-Qur'an, selain itu terdapat *activity* atau halaman untuk latihan *murojaah* dimana pada halaman ini nantinya para *hafizh* dapat mengetahui hafalannya sudah benar atau belum beserta persentase persamaan ayat yang diucapkan pengguna. Aplikasi ini mengimplementasikan *Google Speech API* dan menerapkan Algoritma *Rabin Karp* dalam menghitung jumlah persamaan ayat yang diucapkan pengguna. *Google Speech Application Programming Interface* adalah sebuah layanan pengembangan pada Model *Machine Learning*, pemanfaatannya dapat diterapkan untuk mengembangkan sebuah aplikasi dengan menggunakan teknologi dari *speech recognition*[4]. Sementara itu *Speech Recognition* menyatakan sebuah kemampuan untuk melakukan pencocokkan pola yang diperoleh dari kata terhadap sinyal suara dalam bentuk yang mudah dikenali[8]. Sedangkan Algoritma *Rabin Karp* merupakan sebuah algoritma pencarian atau string matching yang mencari pola *substring* dalam sebuah string menggunakan perhitungan *hashing*. Algoritma ini bermanfaat dalam mencocokkan kata dengan banyak pola atau *pattern*[9]. Berikut adalah *flowchart* atau alur kerja aplikasi yang dibangun.

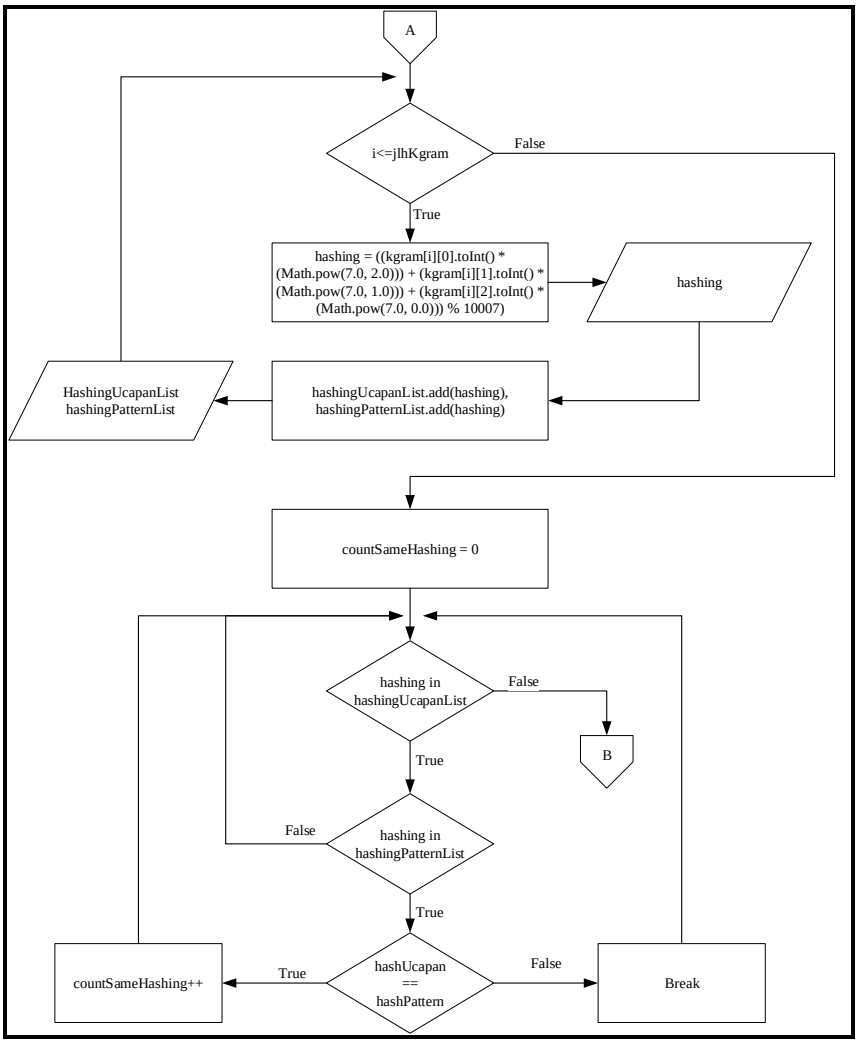


Gambar 1. *Flowchart* Aplikasi *Muroja'ah* yang dibangun

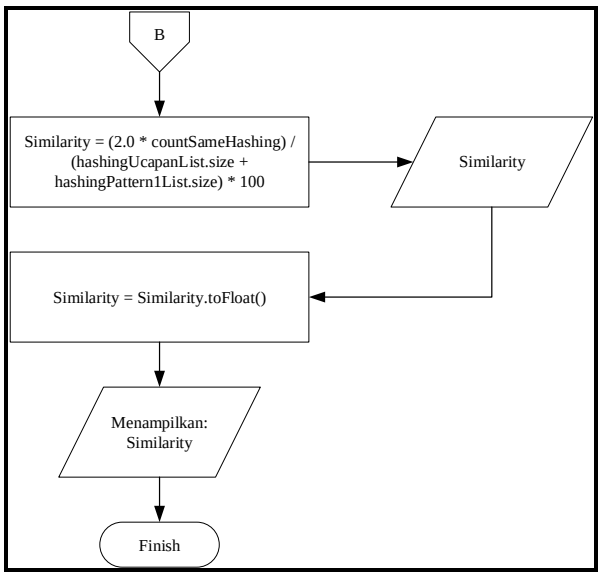
Adapun *flowchart* atau alur proses dari Algoritma *Rabin Karp* pada aplikasi yang dibangun adalah sebagai berikut



Gambar 2.a. *Flowchart* Proses Algoritma *Rabin Karp* Aplikasi *Muroja'ah*



Gambar 2.b. *Flowchart* Proses Algoritma *Rabin Karp* Aplikasi *Muroja'ah*



Gambar 2.c. *Flowchart* Proses Algoritma *Rabin Karp* Aplikasi *Muroja'ah*

Berikut adalah contoh penerapan manual dari tahapan proses Algoritma *Rabin Karp* pada aplikasi *muroja'ah* yang dibangun. Adapun contohnya ialah pada ayat-ayat surah *Al-Mau'un*.

1. *Input* suara/ayat pengguna

Terdapat dua kumpulan *string*, yakni string *pattern* yaitu *string* asli berupa data ayat yang benar. Dan string *speech* berupa *string* dari ayat yang diucapkan pengguna atau *string* uji, dalam hal ini dikenali dengan variabel *speech*. Adapun contoh penerapannya ialah pada ayat-ayat surah Al-Ma'un yang dapat dilihat pada tabel berikut:

Tabel 1. Contoh penerapan *string matching* pada aplikasi

No Ayat	Pattern	Speech
1	aroaitalladziyukadzibubiddin	Aroaitalladzi yukadzibu biddin
2	fadzalikalladziiyadu\'ulyatim	Fadzalikal ladzii yadu'ul yatim
3	walayahudualatoamilmiskin	Wala yahudu ala toa mil miskin
4	fawailullilmushollin	Fawailul lil mushollin
5	allazinahumanshalatihimsahun	Allazinahum an shalatihim sahun
6	alladzinahumyuroun	Alladzinahum yuroun
7	wayamnaunalma\'un	Wayam naunal ma'un

2. Tahap *text preprocessing*

Pada tahap ini *string pattern* dan *string speech* dilakukan proses *filtering* dan pembersihan data. Yaitu dengan menjadikan semua karakter menjadi huruf kecil semua dan menyaring seluruh karakter menjadi kumpulan huruf dari huruf a kecil sampai z kecil dan menghapus spasi beserta simbol-simbol lainnya seperti tanda koma atas, koma, titik, dan lain sebagainya.[10] Adapun hasilnya dapat dilihat pada tabel berikut:

Tabel 2. Tahap *text preprocessing*

No Ayat	Pattern	Speech
1	aroaitalladziyukadzibubiddin	aroaitalladziyukadzibubiddin
2	fadzalikalladziiyaduulyatim	fadzalikalladziiyaduulyatim
3	walayahudualatoamilmiskin	walayahudualatoamilmiskin
4	fawailullilmushollin	fawailullilmushollin
5	allazinahumanshalatihimsahun	allazinahumanshalatihimsahun
6	alladzinahumyuroun	alladzinahumyuroun
7	wayamnaunalmaun	wayamnaunalmaun

3. Tahap *parsing Kgram*

Pada tahap ini dilakukan proses pembentukan pola kata pada *string* dengan cara membagi string menjadi beberapa bagian[11]. Dalam hal ini penulis memecah menjadi 3 (tiga) bagian potongan-potongan. Yang dapat dilihat pada tabel berikut:

Tabel 3. Tahap *parsing* Kgram

No Ayat	Pattern	Speech
1	[aro, roa, oai, ait, ita, tal, all, lla, lad, adz, dzi, ziy, iyu, yuk, uka, kad, adz, dzi, zib, ibu, bub, ubi, bid, idd, ddi, din]	[aro, roa, oai, ait, ita, tal, all, lla, lad, adz, dzi, ziy, iyu, yuk, uka, kad, adz, dzi, zib, ibu, bub, ubi, bid, idd, ddi, din]
2	[fad, adz, dza, zal, ali, lik, ika, kal, all, lla, lad, adz, dzi, zii, iiy, iya, yad, adu, duu, uul,	[fad, adz, dza, zal, ali, lik, ika, kal, all, lla, lad, adz, dzi, zii, iiy, iya, yad, adu, duu, uul,

No Ayat	Pattern	Speech
	uly, lya, yat, ati, tim]	uly, lya, yat, ati, tim]
3	[wal, ala, lay, aya, yah, ahu, hud, udu, dua, ual, ala, lat, ato, toa, oam, ami, mil, ilm, lmi, mis, isk, ski, kin]	[wal, ala, lay, aya, yah, ahu, hud, udu, dua, ual, ala, lat, ato, toa, oam, ami, mil, ilm, lmi, mis, isk, ski, kin]
4	[faw, awa, wai, ail, ilu, lul, ull, lli, lil, ilm, lmu, mus, ush, sho, hol, oll, lli, lin]	[faw, awa, wai, ail, ilu, lul, ull, lli, lil, ilm, lmu, mus, ush, sho, hol, oll, lli, lin]
5	[all, lla, laz, azi, zin, ina, nah, ahu, hum, uma, man, ans, nsh, sha, hal, ala, lat, ati, tih, ihi, him, ims, msa, sah, ahu, hun]	[all, lla, laz, azi, zin, ina, nah, ahu, hum, uma, man, ans, nsh, sha, hal, ala, lat, ati, tih, ihi, him, ims, msa, sah, ahu, hun]
6	[all, lla, lad, adz, dzi, zin, ina, nah, ahu, hum, umy, myu, yur, uro, rou, oun]	[all, lla, lad, adz, dzi, zin, ina, nah, ahu, hum, umy, myu, yur, uro, rou, oun]
7	[way, aya, yam, amn, mna, nau, aun, una, nal, alm, lma, mau, aun]	[way, aya, yam, amn, mna, nau, aun, una, nal, alm, lma, mau, aun]

4. Tahap perhitungan *Hashing* dan perhitungan hasil *similarity*

Pada tahap ini dilakukan proses untuk mentransformasikan string menjadi sebuah *hash value* atau nilai yang unik berdasarkan bilangan *American Standard Code for Information Interchange* (ASCII)[12]. Berikut adalah tabel karakter ASCII[13].

Tabel 4. Tabel Karakter ASCII

Char	Bilangan ASCII	**Char**	Bilangan ASCII	**Char**	Bilangan ASCII	**Char**	Bilangan ASCII
a	97	**i**	105	**p**	112	**x**	120
b	98	**j**	106	**q**	113	**y**	121
c	99	**k**	107	**r**	114	**z**	122
d	100	**g**	103	**s**	115		
e	101	**l**	108	**t**	116		
f	102	**m**	109	**u**	117		
g	103	**n**	110	**v**	118		
h	104	**o**	111	**w**	119		

Sebagai contoh pada Kgram pertama yaitu [aro]. Dalam hal ini basis ditentukan dengan nilai 7 (tujuh) karena angka tersebut paling sesuai dengan hasil persentase dari *similarity* yaitu dengan hasil maksimal 100%.

Tabel 5. Tahap menghitung *hashing* pada Kgram

Char	**Nilai ASCII (decimal)**	**Char ASCII * basis ^ posisi karakter dimulai dari belakang**
a	97	(97 * (7^2)) +
r	114	(114 * (7^1)) +
o	111	(111 * (7^0))
Hasil hashing [aro]		**= 5662**

Berikut perhitungan *hashing* lengkap pada seluruh ayat surah *Al-Ma'un.* Dan kemudian aplikasi akan menghitung jumlah *hash* pada masing-masing *Kgram* di *string pattern* dan juga *string speech.* Setelah itu aplikasi akan membandingkan dan menghitung jumlah *hash* yang sama. Dan terakhir adalah menghitung *similarity*. Dengan rumus (2 * jumlah *hash* yang sama) / (jumlah *hashing speech* + jumlah *hashing pattern*) * 100.

Tabel 6. Tahap menghitung jumlah *hasing* dan hasil *similarity*

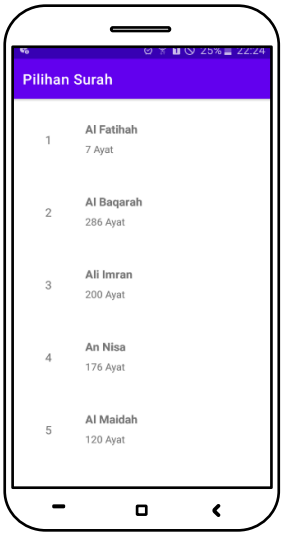
Nomor Ayat	***Pattern***	***Speech***	**Jumlah Hash yang Sama**	**Hasil *Similarity***
1	1. aro=5662.0 2. roa=6460.0 3. oai=6223.0 4. ait=5604.0 5. ita=6054.0 6. tal=6471.0 7. all=5617.0 8. lla=6145.0 9. lad=6071.0 10. adz=5575.0 11. dzi=5859.0 12. ziy=6834.0 13. iyu=6109.0 14. yuk=6855.0 15. uka=6579.0 16. kad=6022.0 17. adz=5575.0 18. dzi=5859.0 19. zib=6811.0 20. ibu=5948.0 21. bub=5719.0 22. ubi=6524.0 23. bid=5637.0 24. idd=5945.0 25. ddi=5705.0 26. din=5745.0	1. aro=5662.0 2. roa=6460.0 3. oai=6223.0 4. ait=5604.0 5. ita=6054.0 6. tal=6471.0 7. all=5617.0 8. lla=6145.0 9. lad=6071.0 10. adz=5575.0 11. dzi=5859.0 12. ziy=6834.0 13. iyu=6109.0 14. yuk=6855.0 15. uka=6579.0 16. kad=6022.0 17. adz=5575.0 18. dzi=5859.0 19. zib=6811.0 20. ibu=5948.0 21. bub=5719.0 22. ubi=6524.0 23. bid=5637.0 24. idd=5945.0 25. ddi=5705.0 26. din=5745.0	26	100%

Tahap Implementasi

Tahapan implementasi merupakan implementasi sistem yang dibangun dengan penerapan metode pemrograman terhadap hasil kebutuhan sistem.[14] Tahap implementasi adalah proses menciptakan dan mengimplementasikan aplikasi yang ingin dibangun secara keseluruhan dari sudut pandang *hardware* (perangkat keras) dan *software* (perangkat lunak).

a. Tampilan awal daftar surah

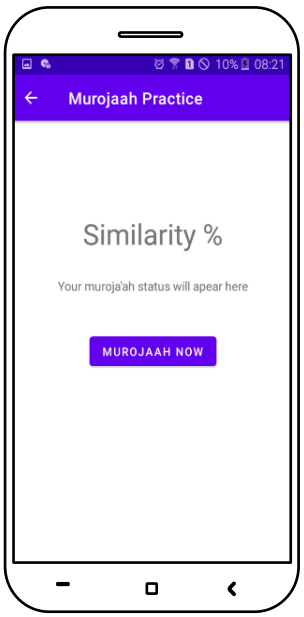
   Tampilan awal merupakan tampilan pembuka dari aplikasi *muroja'ah*. Tampilan ini tampil ketika pertama kali aplikasi ini dijalankan. Tampilan ini berisikan *recyclerview* atau daftar-daftar surah dalam bentuk *list.* Berikut adalah implementasi tampilan awal aplikasi yang dibangun yang terdapat pada gambar berikut ini:



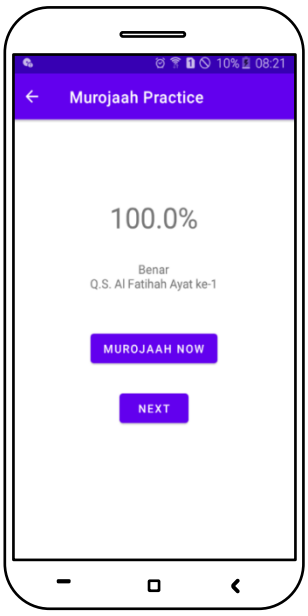
Gambar 7. Implementasi tampilan awal aplikasi

b. Tampilan latihan murojaah

Tampilan latihan murojaah adalah tampilan yang didapatkan setelah *user* memilih surah yang ingin dikoreksi hafalannya. Tampilan ini berfungsi untuk melihat status dan tingkat persamaan ayat yang diucapkan dengan data ayat yang benar. Adapun implementasi tampilan latihan murojaah sebelum dan sesudah diinputkan suara bisa dilihat pada gambar berikut:



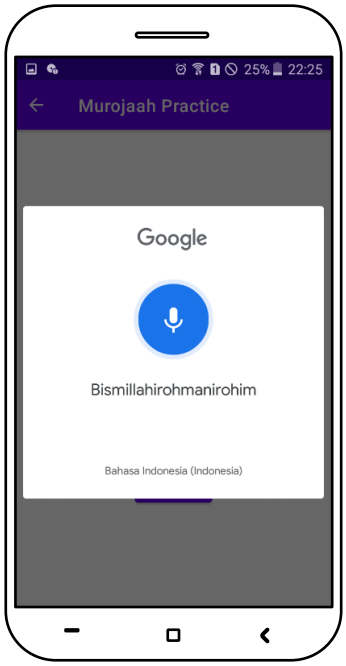
Gambar 8. Implementasi tampilan sebelum *input* suara



Gambar 9. Implementasi tampilan setelah *input* suara

c. Tampilan *google speech*

Tampilan *google speech* adalah tampilan yang akan terlihat jika *user* memilih *button* Murojaah Now. Tampilan ini berfungsi untuk mengenali ucapan pengguna dan kemudian mengkonversi suara atau ucapan tersebut kedalam sebuah teks. Adapun implementasi tampilan *google speech* ini dapat dilihat pada gambar berikut.



Gambar 10. Implementasi tampilan *google speech*

Tahap *Testing*

Dalam tahap *testing* ini, penulis melakukannya setelah tahap implementasi selesai, dengan menjalankan program aplikasi yang dibangun untuk melihat apakah masih terdapat *error* atau program berjalan dengan aman dan lancar. Pada tahap ini, dilakukan pengujian aplikasi dengan metode *blackbox testing.* Metode ini adalah metode untuk melakukan pengujian terhadap program aplikasi berdasarkan fungsi-fungsi yang ada didalam program aplikasi yang dibangun[15]. Adapun pengujian yang penulis lakukan menggunakan *device* atau perangkat Android versi Lollipop 5.1.1 *Samsung Galaxy J3* 2016 *Smartphone* dan diperoleh hasil pengujian seperti berikut:

Tabel 6. Pengujian *Blackbox Testing* pada sistem yang dibangun

No	Jenis	Kegiatan	Hasil
1	*RecyclerView*	Menampilkan daftar *list* surah Al-Qur'an berserta nomor surah dan jumlah ayatnya	Sesuai
2	*On Click surah's item list* pada *RecyclerView*	Menampilkan halaman praktik murojaah (*Murojaah Practice Activity*) sesuai dengan surah yang dipilih	Sesuai
3	*On Back Pressed* pada *Murojaah Practice Activity*	Mengembalikan halaman atau *activity* ke halaman awal (*main activity*)	Sesuai
4	Murojaah *Now Button*	Menanpilkan tampilan *google speech*	Sesuai
5	*Google Speech*	Mengkonversi suara ke teks	Sesuai

6	*Similarity Textview*	Menampilkan tingkat perbandingan atau persamaan ucapan menggunakan Algoritma Rabin Karp dalam satuan persen	Sesuai
7	*Status Textview*	Menampilkan status apakah ucapan sudah benar atau tidak	Sesuai
8	*Next Button*	Membersihkan data	Sesuai

Berdasarkan hasil *blackbox testing* yang penulis lakukan terhadap lima orang yang berbeda menyatakan bahwa sistem aplikasi yang dibangun telah berjalan sesuai dengan fungsi yang semestinya. Maka berdasarkan hasil tersebut dapat disimpulkan bahwa sistem aplikasi yang penulis bangun dan kembangkan telah tervalidasi melalui pengujian *blackbox testing.* Dan hasil pengujian tersebut mengungkapkan bahwa sistem aplikasi yang dibangun telah berhasil berjalan sesuai dengan fungsi yang semestinya.

4. KESIMPULAN

Berdasarkan pembahasan pada bab-bab sebelumnya, maka dapat ditarik beberapa kesimpulan diantaranya ialah sebagai berikut:

1. Aplikasi yang dibangun adalah aplikasi muroja'ah hafalan Al-Qur'an berbasis *Android* dengan menerapkan *Google Speech* API dan Algoritma *Rabin Karp.* Aplikasi akan memberikan informasi berupa status muroja'ah pengguna beserta tingkat akurasi persamaan ayat yang diucapkan.
2. Aplikasi dibangun menggunakan *Android* Studio dengan bahasa pemrograman *Kotlin*.
3. Sampel surah yang diterapkan pada aplikasi yang dibangun sebanyak jumlah ayat pada enam surah. Surah tersebut terdiri dari Surah Al-Fatihah, Al-Maun, Al-Kausar, Al-Ikhlas, Al-Falaq, dan An-Nas.
4. Aplikasi *muroja'ah* hafalan Al-Qur'an mengimplementasikan teknologi *Speech Recognition in Artificial Intelligence* dengan mengimplementasikan layanan dari *Google Speech API* yang berguna untuk mengenali suara/ayat yang diucapkan pengguna dengan mengkonversi suara tersebut kedalam sebuah *string*/teks Bahasa Indonesia.

Aplikasi *muroja'ah* hafalan Al-Qur'an menerapkan proses *string matching* menggunakan algoritma *Rabin Karp* sebagai proses dalam membandingkan antara ayat yang diucapkan oleh pengguna dengan data ayat yang benar yang terdapat pada *file String.xml*. Sehingga aplikasi akan menampilkan persentase akurasi kesamaan ayat. Dalam proses membandingkan ayat tersebut terdapat beberapa proses yang dilakukan yaitu pertama melakukan proses *preprocessing* dengan menjadikan seluruh kalimat menjadi huruf kecil dan membersihkan atau *filter* kalimat yang diucapkan oleh pengguna hanya terdiri dari huruf abjad tanpa spasi, simbol, dan karakter-karakter lainnya. Kedua memisahkan karakter yang sudah dibersihkan tadi menjadi tiga bagian Kgram. Ketiga melakukan proses *hashing Kgram* dengan mengkonversi bagian-bagian *Kgram* tersebut menjadi bilangan ASCII (*American Standard Code for Information Interchange*), kemudian menghitung jumlah *hash* yang sama pada setiap *Kgram*. Keempat melakukan perhitungan persamaan (*similarity*) pada kalimat atau ayat yang diucapkan oleh pengguna. Setelah didapatkan *similarity*-nya maka aplikasi akan memberikan informasi mengenai status benar atau belum tepat beserta persentase akurasi persamaannya

5. SARAN

Adapun saran penelitian untuk penelitian lebih lanjut adalah sebagai berikut:

Berdasarkan penelitian yang penulis lakukan, adapun saran untuk penelitian selanjutnya adalah sebagai berikut:

1. Konversi suara yang diucapkan oleh pengguna sebaiknya diubah kedalam bahasa arab.
2. Menambahkan isi ayat pada setiap surah yang belum terdapat pada aplikasi.
3. Menambahkan fitur mencari letak kesalahan jika terdapat kesalahan pada saat *muroja'ah*. Dan dapat mengidentifikasi jika terdapat *missing* atau kesalahan bacaan huruf baik satu huruf maupun lebih.
4. Menambahkan fitur koreksi kesalahan yang lebih kompleks seperti koreksi panjang pendek ayat (*harakat*), *tajwid, qalqalah* dan lainnya yang terdapat pada ilmu baca Al-Qur'an yang benar.

Mengimplementasikan aplikasi pada layanan *cloud provider* lainnya atau membuat *data training* ayat yang benar dengan menerapkan bidang ilmu *Machine Learning* atau *Natural Languages Processing*.

DAFTAR PUSTAKA

[1] Nurman Hidayat and Kusuma Hati, "*Penerapan Metode Rapid Application Development* (RAD) dalam Rancang Bangun Sistem Informasi Rapor *Online* (SIRALINE)," *J. Sist. Inf.*, vol. 10, no. 1, pp. 8–17, 2021, doi: 10.51998/jsi.v10i1.352.

[2] N. Rolly and N. Hakiem, "Pengembangan Aplikasi *Mobile Academic Information System* (AIS) Berbasis Android Untuk Pengguna Dosen Dan Mahasiswa (Studi Kasus : Pusat Teknologi Informasi dan Pangkalan Data (Pustipanda) UIN Syarif Hidayatullah Jakarta)," *J. Tek. Inform.*, vol. 8, no. 1, pp. 16–21, 2015, doi: 10.15408/jti.v8i1.1932.

[3] R. Rudianto, "*The Implementation of the RAD model* in *the Development of Tender Selection Programs Using the AHP Method*," *J. Informatics Telecommun. Eng.*, vol. 3, no. 2, pp. 232–239, 2020, doi: 10.31289/jite.v3i2.3232.

[4] D. I. S. Saputra, S. W. Handani, and G. A. Diniary, "Pemanfaatan *Cloud Speech Api* Untuk Pengembangan Media Pembelajaran Bahasa Inggris Menggunakan Teknologi *Speech Recognition*," *Telematika*, vol. 10, no. 2, pp. 92–105, 2017.

[5] N. Purwati, "Aplikasi *Sampling* (Sampah Lingkungan) Pengrajin Sampah Berbasis *Web* Menggunakan Metode RAD (*Rapid Application Development*)," *EVOLUSI J. Sains dan Manaj.*, vol. 9, no. 1, pp. 78–86, 2021, doi: 10.31294/evolusi.v9i1.10316.

[6] S. Syamsiah, "Perancangan *Flowchart* dan *Pseudocode* Pembelajaran Mengenal Angka dengan Animasi untuk Anak PAUD Rambutan," *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 4, no. 1, p. 86, 2019, doi: 10.30998/string.v4i1.3623.

[7] M. Syarif and W. Nugraha, "Pemodelan Diagram UML Sistem Pembayaran Tunai Pada Transaksi E-Commerce," *J. Tek. Inform. Kaputama*, vol. 4, no. 1, p. 70 halaman, 2020, [Online]. Available: http://jurnal.kaputama.ac.id/index.php/JTIK/article/view/240.

[8] R. M. Baeker, J. Grudin, W. A. S. Buxton, and S. Greenberg, *Readings in Human-Computer Interaction: Toward The Year 2000*. USA: Morgan Kaufmann Publishers Inc, 1995.

[9] A. P. U. Siahaan *et al.*, "*Combination of levenshtein distance and rabin-karp to improve the accuracy of document equivalence level*," *Int. J. Eng. Technol.*, vol. 7, no. 2 Special Issue 27, pp. 17–21, 2018, doi: 10.14419/ijet.v7i2.27.12084.

[10] A. Filcha and M. Hayaty, "Implementasi Algoritma *Rabin-Karp* untuk Pendeteksi *Plagiarisme* pada Dokumen Tugas Mahasiswa," *JUITA J. Inform.*, vol. 7, no. 1, p. 25, 2019, doi: 10.30595/juita.v7i1.4063.

[11] Rahmaddeni, D. Sazali, and Agustin, "Sistem Pendeteksi Tingkat Kesamaan Teks pada

Pengusulan Proposal Penelitian Internal Menggunakan Algoritma *Rabin-Karp*," vol. 4, no. 2, 2018.

[12] D. A. Putra, H. Sujaini, and H. S. Pratiwi, "Implementasi Algoritma *Rabin-Karp* untuk Membantu Pendeteksian Plagiat pada Karya Ilmiah," *J. Sist. dan Teknol. Inf.*, vol. 4, no. 1, pp. 66–74, 2015, [Online]. Available: http://jurnal.untan.ac.id/index.php/justin/article/view/12411.

[13] "ASCII Code - The extended ASCII table," 2021. https://www.ascii-code.com/ (accessed Jul. 06, 2021).

[14] M. P. Putri and H. Effendi, "Implementasi Metode *Rapid Application Development* Pada *Website Service Guide* '*Waterfall Tour South* Sumatera,'" *J. SISFOKOM*, vol. 07, no. September, pp. 130–136, 2018.

[15] A. Andriani and E. Qurniati, "Sistem Informasi Penjualan Pada Toko *Online* Dengan Metode *Rapid Application Development* (RAD)," *J. Speed – Sentra Penelit. Eng. dan Edukasi*, vol. 10, no. 3, pp. 49–54, 2018, [Online]. Available: http://speed.web.id/ejournal/index.php/speed/article/view/392/385.