PENGEMBANGAN APLIKASI KAMUS BAHASA ACEH BERBASIS *MOBILE*

TUGAS AKHIR

Diajukan Oleh:

SAHIBUL NUZUL FIRDAUS
NIM. 180705014
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI AR-RANIRY
BANDA ACEH
2023 M/1444 H

PENGEMBANGAN APLIKASI KAMUS BAHASA ACEH BERBASIS *MOBILE*

TUGAS AKHIR

Diajukan kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Ar-Raniry Banda Aceh
Sebagai Salah Satu Beban Studi Memperoleh Gelar Sarjana (S1)
dalam Ilmu/Prodi Teknologi Informasi

Oleh:
SAHIBUL NUZUL FIRDAUS
NIM. 180705014
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui Untuk Dimunaqasyahkan Oleh:

Pembimbing I,	Pembimbing II,
Bustami, M. Sc	**Ima Dwitawati, M.B.A**
NIDN. 2008048601	**NIDN. 0113108204**

Mengetahui,
Ketua Program Studi Teknologi Informasi

Ima Dwitawati, M.B.A
NIDN. 0113108204

PENGEMBANGAN APLIKASI KAMUS BAHASA ACEH BERBASIS *MOBILE*

TUGAS AKHIR/SKRIPSI

Telah Diuji Oleh Panitia Ujian Munaqasah Tugas Akhir/Skripsi
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus
Serta Diterima Sebagai Salah Satu Beban Studi Program Sarjana (S-1)
Dalam Ilmu/Prodi Teknologi Informasi

Pada Hari/Tanggal: Rabu, 7 Maret 2023
14 Syaban 1444 H
di Darussalam, Banda Aceh

Panitia Ujian Munaqasah Tugas Akhir/Skripsi:

Ketua,	Sekretaris,
Bustami, M. Sc	**Ima Dwitawati, M.B.A**
NIDN. 2008048601	**NIDN. 0113108204**
Penguji I,	Penguji II,
Hendri Ahmadian, S.Si., M.IM	**Malahayati, MT**
NIDN. 2004018303	**NIDN. 2027018303**

Mengetahui:
Dekan Fakultas Sains dan Teknologi
UIN Ar-Raniry Banda Aceh,

Dr. Ir. Muhammad Dirhamsyah, M.T., IPU
NIP. 196210021988111001

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR/SKRIPSI

Yang bertanda tangan di bawah ini:

Nama	:	Sahibul Nuzul Firdaus
NIM	:	180705014
Program Studi	:	Teknologi Informasi
Fakultas	:	Sains dan Teknologi
Judul	:	Pengembangan Aplikasi Kamus Bahasa Aceh Berbasis *Mobile*

Dengan ini menyatakan bahwa dalam penulisan tugas akhir/skripsi ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggungjawabkan;
2. Tidak melakukan plagiasi terhadap naskah karya orang lain;
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya;
4. Tidak memanipulasi dan memalsukan data;
5. Mengerjakan sendiri karya ini dan mampu bertanggungjawab atas karya ini.

Bila dikemudian hari ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat dipertanggungjawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenai sanksi berdasarkan aturan yang berlaku di Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh.

Demikian pernyataan ini saya buat dengan sesungguhnya dan tanpa paksaan dari pihak manapun.

Banda Aceh, 13 Januari 2023
Yang Menyatakan,

Sahibul Nuzul Firdaus

ABSTRAK

Nama	:	Sahibul Nuzul Firdaus
NIM	:	180705014
Program Studi	:	Teknologi Informasi
Judul	:	Pengembangan Aplikasi Kamus Bahasa Aceh Berbasis *Mobile*
Tanggal Sidang	:	7 Maret 2023 / 14 Syaban 1444 H
Jumlah Halaman	:	133 Halaman
Pembimbing I	:	Bustami, M. Sc
Pembimbing II	:	Ima Dwitawati, M.B.A
Kata Kunci	:	Aplikas Kamus, Bahasa Aceh, Jaro-Winkler, *Personal Extreame Programming, Recommendation List*

Bahasa Aceh merupakan bahasa yang aktif digunakan untuk komunikasi oleh sebagian besar masyarakat Aceh. Bahasa Aceh sendiri memiliki keunikan yang disebut tanda-tanda diakritik pada sistem atau kaidah penulisan kosakatanya. Masalah utama yang terdapat pada aplikasi kamus bahasa Aceh yang tersedia di toko aplikasi saat penelitian ini dilakukan yaitu masih terdapat kesalahan penulisan atau pengetikan kosakata bahasa Aceh yang tidak dibubuhi dengan tanda-tanda diakritik sesuai kaidah atau sistem penulisan bahasa Aceh. Kesalahan tersebut dapat menyebabkan terjadinya perubahan makna kata dan dapat mengubah identitas ataupun tatanan bahasa Aceh. Pengetikan tanda diakritik sendiri tidak mudah untuk dilakukan dengan *keyboard* fisik ataupun *keyboard virtual* pada umumnya, karena keterbatasan karakter atau huruf yang ditampilkan sehingga membutuhkan teknik khusus untuk mengetiknya. Pengembangan aplikasi yang dilakukan pada penelitian ini bertujuan untuk memberikan solusi terkait permasalahan tanda diakritik yang belum tersedia pada aplikasi kamus bahasa Aceh yang telah ada sebelumnya. Aplikasi kamus bahasa Aceh berbasis *mobile* ini difokuskan pada penambahan jumlah data dan pembuatan fitur *recommendation list* pada fitur pencarian menggunakan algoritma Jaro-Winkler. Fitur *recommendation list* berguna untuk membantu pengguna yang kesulitan mengetik tanda diakritik pada saat melakukan pencarian kosakata di aplikasi. Metode yang digunakan adalah studi literatur dan data *preprocessing* untuk pengumpulan data, serta *Personal Extreme Programming* (PXP) untuk pengembangan aplikasi. Teknologi yang digunakan adalah menerapkan gaya arsitektur REST (*client-server*), bahasa pemograman Dart dan Flutter untuk *mobile*, dan bahasa pemograman Golang dan Gin Web *Framework* untuk *backend*, serta *database* PostgreSQL. Hasil evaluasi

fitur *recommendation list* yang diuji dengan 1000 data *testing* disimpulkan bahwa algoritma Jaro-Winkler cocok digunakan dalam fitur *recommendation list* atau sistem rekomendasi kata pada aplikasi kamus bahasa Aceh dalam kasus dimana kata kunci bahasa Aceh yang diketik tidak mengandung tanda diakritik. Algoritma Jaro-Winkler mampu memberikan nilai *Mean Average Precision* (MAP) yang lebih unggul dari pada algoritma Levenshtein yang digunakan sebagai pembanding dengan menunjukkan nilai MAP yang lebih tinggi di setiap *threshold* yang diuji.

Kata kunci: Aplikas Kamus, Bahasa Aceh, Jaro-Winkler, *Personal Extreame Programming, Recommendation List*

ABSTRACT

Name	:	Sahibul Nuzul Firdaus
Student ID	:	180705014
Department	:	*Information Technology*
Title	:	*Development of a Mobile-Based Aceh Language Dictionary Application*
Date	:	7 *March* 2023 / 14 *Shaban* 1444 H
Thesis Pages	:	133 *Pages*
Supervisor I	:	Bustami, M. Sc
Supervisor II	:	Ima Dwitawati, M.B.A
Keywords	:	*Dictionary Application, Acehnese,* Jaro-Winkler, *Personal Extreame Programming, Recommendation List*

The Acehnese language is the language that is actively used for communication by most Acehnese people. The Acehnese language itself has a uniqueness called diacritical signs in its vocabulary writing system or rules. The main problem found in the Aceh language dictionary application which was available in the application store when this research was conducted was that there were still writing or typing errors in the Acehnese vocabulary which were not affixed with diacritical marks according to the rules or writing system of the Acehnese language. These errors can cause changes in the meaning of words and can change the identity or order of the Acehnese language. Typing the diacritical marks themselves is not easy to do with a physical keyboard or virtual keyboard in general, due to the limited characters or letters displayed, requiring special techniques to type them. The application development carried out in this study aims to provide solutions related to the problem of diacritical marks that are not yet available in the previously existing Aceh language dictionary application. This mobile-based Aceh language dictionary application is focused on increasing the amount of data and creating a recommendation list feature in the search feature using the Jaro-Winkler algorithm. The recommendation list feature is useful for helping users who have difficulty typing diacritical marks when searching for vocabulary in the application. The method used is literature study and data preprocessing for data collection, as well as Personal Extreme Programming (PXP) for application development. The technology used is applying the REST (client-server) architectural style, the Dart and Flutter programming languages for mobile, and the Golang and Gin Web Framework programming languages for the backend, as well as the PostgreSQL database. The

evaluation results of the recommendation list feature tested with 1000 data testing concluded that the Jaro-Winkler algorithm is suitable for use in the recommendation list feature or word recommendation system in the Aceh language dictionary application in cases where the Aceh language keywords typed do not contain diacritical marks. The Jaro-Winkler algorithm is able to provide a Mean Average Precision (MAP) value that is superior to the Levenshtein algorithm used as a comparison by showing a higher MAP value at each tested threshold.

Keywords: Dictionary Application, *Acehnese*, Jaro-Winkler, *Personal Extreame Programming, Recommendation List*

KATA PENGANTAR

بِسْـــــــــــــمِ اللهِ الرَّحْمَنِ الرَّحِيْمِ

Segala puji hanya milik Allah, Tuhan seluruh alam yang tiada Tuhan selain-Nya. Shalawat dan salam semoga selalu tercurah kepada junjungan kita Nabi Muhammad Shallallahu Alaihi Wassalaam, keluarga dan sahabatnya. Alhamdulillah dengan rahmat Allah yang Maha Rahman dan yang Maha Rahim, sehingga penulis dapat menyelesaikan skripsi dengan judul "Pengembangan Aplikasi Kamus Bahasa Aceh Berbasis *Mobile*" ini. Skripsi ini merupakan salah satu syarat untuk menyelesaikan program studi Strata satu Teknologi Informasi pada Fakultas Sains dan Teknologi di Universitas Islam Negeri Ar-Raniry Banda Aceh.

Ucapan terima kasih penulis sampaikan kepada berbagai pihak yang menjadi sebab dari mereka penulis belajar, mendapatkan ilmu, mendapatkan dukungan, serta mendapatkan hal yang bermanfaat lainnya sehingga penulis sampai pada titik menyelesaikan skripsi ini. Terutama dalam konteks ini penulis sampaikan kepada :

- Kedua orang tua dan keluarga besar penulis yang selalu memberikan dukungan, dan doa yang tak ternilai harganya selama ini. Hanya Allah yang mampu membalas kasih sayang mereka yang tak terhingga, semoga selalu Allah limpahkan rahmat kepada mereka dan mendapatkan ridha serta cinta dari-Nya.
- Bapak Bustami, M. Sc. dan Ibu Ima Dwitawati, M.B.A. selaku pembimbing yang selalu tersedia meluangkan waktu, fikiran untuk membimbing penulis demi kesempurnaan skripsi ini. Kemudian juga kepada Bapak Khairan Ar, M.Kom dan Bapak Hendri Ahmadian, S.Si.,M.IM selaku penguji sidang proposal skripsi yang telah memberi koreksi dan saran untuk mencapai tujuan yang sama. Semoga Allah limpahkan rahmat kepada Bapak dan Ibu dan mendapatkan ridha serta cinta dari-Nya.
- Ketua dan Sekretaris Program Studi Teknologi Informasi, Ibu Ima Dwitawati, M.B.A. dan Bapak Khairan Ar, M.Kom, serta Bapak dan Ibu dosen Program Studi Teknologi Informasi yang telah memberikan ilmu pengetahuan dalam

bidang Teknologi Inofrmasi kepada penulis sehingga penulis mampu menyelesaikan skripsi ini.

- Pembimbing Akademik, Ibu Sri Wahyuni, M.T dan Bapak Bustami, M. Sc., yang telah membimbing dan memberikan saran selama masa perkuliahan.
- Staf Prodi Ibu Cut Ida Rahmadiana S,Si. yang telah membantu membantu penulis dalam hal pengurusan administrasi dan surat-surat untuk keperluan penyelesaian skripsi.
- Teman-teman mahasiswa Prodi Teknologi Informasi dan semua pihak yang tidak bisa sebut satu persatu.

Penulis berharap hasil skripsi ini dapat berguna bagi kelestarian bahasa Aceh dengan adanya aplikasi kamus bahasa Aceh yang dikembangkan. Dan dapat dijadikan sebagai referensi bagi penelitian selanjutnya ataupun sebagai referensi untuk mempelajari bahasa Aceh dan menulis bahasa Aceh diberbagai media.

Akhir kata, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, oleh karena itu kritik dan saran yang bersifat membangun sangat penulis harapkan demi mendapatkan hasil yang lebih baik. Semoga perjalanan mempelajari dan berkarya (penyusunan skripsi) pada salah satu ilmu milik-Nya ini dapat menghantarkan penulis agar dapat mengenal-Nya dan kekasih-Nya lebih banyak serta mendapatkan ridho dan cinta-Nya yang Maha Rahman dan Rahim. Shalawat dan salam kepada junjungan kita Nabi Muhammad Shallallahu Alaihi Wassalaam, keluarga dan sahabat-sahabatnya.

Banda Aceh, 13 Januari 2023

Penulis,

Sahibul Nuzul Firdaus

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Bahasa Aceh merupakan salah satu bahasa daerah di Indonesia dan aktif digunakan untuk berkomunikasi dalam kehidupan sehari-hari oleh sebagian besar masyarakat di daerah Aceh. Bahasa Aceh memiliki keunikan pada kaidah-kaidah atau sistem tertentu dalam penulisannya, yaitu pada penulisan yang disebut dengan tanda diakritik dalam bahasa Aceh (Idham & Azwardi, 2008).

Umumnya saat mempelajari suatu bahasa seseorang membutuhkan media belajar untuk menguasai bahasa tersebut, baik itu kamus tradisional (buku) ataupun kamus digital seperti aplikasi *web* dan *mobile*. Kamus digital memiliki kelebihan tersendiri dibandingkan kamus tradisional dalam hal funsionalitasnya. Kamus digital lebih fleksibel dan dapat memberikan berbagai macam fitur canggih yang dapat memudahkan seseorang dalam mempelajari suatu bahasa. Seperti fitur pencarian yang dapat membantu pencarian kata dalam suatu kamus menjadi lebih cepat dibandingkan dengan kamus tradisional.

Saat penelitian ini dilakukan, diketahui sudah tersedia sekitar lima aplikasi kamus bahasa Aceh berbasis *mobile*. Dimana penulis melakukan pencarian dengan kata kunci “kamus aceh” dan “kamus bahasa aceh” pada toko aplikasi *Google Play Store* dan *App Store*. Kelima aplikasi tersebut diantaranya bernama “Kamus Aceh Lengkap”, “Kamus Bahasa Aceh Lengkap”, “Kamus Resmi Bahasa Aceh Offlin”, “Kamus Inggris Aceh”, dan “Kamus Indonesia - Aceh” (Aguswandi, 2021; Azklabs Mobile, 2020; Khaidir, 2018; Nimrod Studio, 2019; techniqueapp, 2022). Selain itu, terdapat juga beberapa penelitian yang merancang aplikasi kamus bahasa Aceh dengan dukungan untuk *platform* Android. Diantaranya yaitu penelitian yang dilakukan oleh Agil M. Caesar pada tahun 2018, dan penelitian yang dilakukan oleh Rizka Pebrijayandi, dan Zalfie Ardian pada tahun 2018 (Agil M. Caesar, 2018;

Pebrijayanti & Ardian, 2018). Namun, aplikasi yang dihasilkan dari penelitian dan aplikasi yang tersedia di toko aplikasi tersebut masih terdapat kekurangan dari segi jumlah data dan terutama terkait kesalahan penulisan atau pengetikan kosakata bahasa Aceh yang tidak dibubuhi dengan tanda diakritik. Penulisan atau pengetikan tanda diakritik sendiri merupakan hal yang sulit untuk dilakukan oleh orang awam *(end user)* dengan *keyboard* fisik maupun *keyboard* virtual pada umumnya, yang mana *keyboard* tersebut memiliki keterbatasan huruf untuk ditampilkan. Dengan kata lain, penulisan tanda diakritik membutuhkan teknik khusus untuk mengetiknya.

Tentunya, kesalahan dan masalah ini tidak baik untuk terus diabaikan, karena dapat mempengaruhi dan dapat mengubah identitas ataupun tatanan bahasa Aceh. Sebagai contoh, kata dalam bahasa Aceh *kuéh* dan *kueh* merupakan kata yang berbeda dengan makna yang juga berbeda. Kata *kuéh* bermakna "kue" dalam bahasa Indonesia, sedangkan *kueh* bermakna "menggali" dalam bahasa Indonesia. Namun, jika kata *kuéh* tidak dibubuhi dengan tanda diakritik pada huruf vokal *e*, maka kata tersebut akan terlihat sama dengan kata *kueh*. Sehingga akan sulit untuk membedakan mana kata yang bermakna "kue" atau "menggali" dalam bahasa Indonesia.

Oleh karena itu, untuk menyempurnakan aplikasi kamus bahasa Aceh yang sudah ada. Penelitian ini berfokus pada pengembangan aplikasi kamus bahasa Aceh berbasis *mobile* dengan penambahan jumlah data dan fitur *recommendation list* pada fitur pencarian kata. Aplikasi ini diharapkan dapat menutup kekurangan dari aplikasi hasil penelitian sebelumnya dan aplikasi yang tersedia di toko aplikasi saat penelitian ini dilakukan, terutama dari segi penulisan tanda diakritik pada kosakata bahasa Aceh. Aplikasi ini juga dilengkapi dengan fitur *recommendation list* pada hasil fitur pencarian kata. Fitur tersebut berguna untuk membantu pengguna mendapatkan hasil pencarian yang sesuai ataupun mendekati dengan kata kunci yang dimaksud, meskipun terdapat kesalahan pengetikan (*typo*) seperti tidak mengetik tanda diakritik pada kata kunci pencarian yang dicari oleh pengguna. Fitur *recommendation list* pada aplikasi ini mengimplementasikan algoritma Jaro-Winkler untuk mencocokkan kata kunci pencarian dengan data kosakata yang tersedia di *database*.

1.2. Rumusan Masalah

Dari penjelasan latar belakang di atas, maka dapat dirumuskan masalah yang muncul yaitu:

1. Bagaimana mengembangkan aplikasi kamus bahasa Aceh berbasis *mobile*?
2. Bagaimana membangun fitur *recommendation list* pada fitur pencarian kata di aplikasi kamus bahasa Aceh berbasis *mobile*?

1.3. Tujuan Penelitian

Berdasarkan latar belakang dan rumusan masalah yang telah penulis uraikan, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan aplikasi kamus bahasa Aceh berbasis *mobile.*
2. Membangun fitur *recommendation list* pada fitur pencarian kata di aplikasi kamus bahasa Aceh berbasis *mobile.*

1.4. Manfaat Penelitian

Berdasarkan latar belakang, rumusan masalah dan tujuan penelitian yang telah penulis uraikan, maka manfaat dari penelitian ini yaitu:

1. Dapat berkontribusi dalam menjaga dan melestarikan bahasa Aceh dengan memanfaatkan Teknologi Informasi.
2. Aplikasi *mobile* kamus bahasa Aceh yang dikembangkan pada penelitian ini dapat digunakan sebagai media belajar bahasa Aceh dan dapat menjadi referensi dalam menulis bahasa Aceh di berbagai media.
3. API *(Application Programming Interface)* yang dibuat pada penelitian ini dapat digunakan sebagai sumber data untuk menyediakan fitur terjemahan bahasa Aceh di berbagai aplikasi *web, mobile* atau *desktop*. Sehingga dapat memberi pengalaman baru bagi pengguna dalam menggunakan teknologi.

1.5. Batasan Masalah

Agar pembahasan dari penelitian ini sesuai dengan judul dan latar belakang yang telah penulis uraikan, maka penulis membatasi masalah yang akan dibahas pada paper ini sebagai berikut:

1. Aplikasi yang dibangun terbatas pada perangkat berbasis *mobile* yaitu pada *platform* Android dan iOS.
2. Bahasa pemograman yang digunakan untuk membangun aplikasi *mobile* adalah Dart dengan *framework* Flutter dan Golang untuk membuat API (*Application Programming Interface*).
3. Data yang digunakan adalah Kamus Bahasa Aceh Inggreh yang disusun oleh Drs. Abdullah Fardan dan Drs. Zulkarnaini yang diterbitkan oleh CV. Boebon Jaya dengan jumlah kata sebanyak 6220 (Fardan & Zulkarnaini, 2019).
4. Fitur pencarian terbatas pada pencarian kosakata bahasa Aceh saja, yang berarti aplikasi hanya menampilkan kata bahasa Aceh dalam rekomendasi kata yang ditampilkan dari hasil pencarian.

BAB II
TINJAUAN PUSTAKA

2.1. Aplikasi *Mobile*

Aplikasi yang dibuat khusus untuk *platform mobile* seperti iOS, Android, atau Windows Mobile disebut aplikasi *mobile* atau lebih dikenal dengan sebutan *mobile apps*. *Mobile apps* adalah aplikasi internet yang dapat berjalan pada *smartphone* atau perangkat *mobile* lainnya (Novianti, 2022). Dalam jurnal (Pebrijayanti & Ardian, 2018), mengutip pendapat Jogiyanto tahun 1999, aplikasi adalah sekumpulan instruksi atau pernyataan di dalam komputer yang disusun sedemikian rupa sehingga komputer tersebut dapat mengolah *input* menjadi *output*. Sedangkan *mobile* dalam konteks Teknologi Informasi merupakan suatu teknologi yang memungkinkan pengguna untuk menggunakan teknologi tersebut dengan berpindah dari suatu tempat ke tempat yang lain. Jadi, Aplikasi *mobile* (*Mobile App* atau *Mobile Application*) merupakan perangkat lunak yang di program untuk perangkat *mobile* seperti telepon genggam, *smartphone*, *smartwatch*, dan *tablet.*

2.2. Kamus

Dalam skripsi (Faathir, 2018), mengutip pengertian KBBI (Kamus Besar Bahasa Indonesia) menyatakan, Kamus adalah bahan referensi yang berisi kata-kata dan ungkapan, biasanya disusun menurut abjad, diikuti dengan penjelasan arti, penggunaan, atau terjemahannya. Kamus praktis membantu pengguna mempelajari kata-kata baru dan artinya. Selain menjelaskan arti kata, juga mencakup penjelasan cara pengucapan kata, penjelasan etimologi kata, dan contoh penggunaannya di masyarakat.

2.3. Bahasa Aceh

Bahasa Aceh merupakan bahasa yang sebagian besar digunakan untuk berkomunikasi dalam kehidupan sehari-hari oleh masyarakat Aceh yang tinggal di

daerah pesisir, pedalaman dan sebagian kepulauan Aceh (Rizal, 2015). Bahasa Aceh tergolong dalam rumpun bahasa *Chamic*, yang merupakan bagian dari rumpun Melayu Polinesia dan Austronesia. Karena itu, bahasa Aceh sangat mirip dengan bahasa Melayu dan Minangkabau, sebab ketiganya masih berada dalam satu rumpun yaitu *Chamic* (Subhayni dkk., 2020). Bahasa Aceh memiliki sistem atau kaidah-kaidah dalam penulisan katanya, dimana terdapat tanda-tanda diakritik pada huruf tertentu. Tanda-tanda diakritik dalam bahasa Aceh diantaranya ialah *grave* (è), *aigu* (é), *trema* (ö), *makron* (ô) dan *apostrof* (´) (Idham & Azwardi, 2008; portalsatu.com, 2016).

Bahasa Aceh ialah bahasa yang unik dengan adanya beberapa dialek. Dalam jurnal (Nurpita dkk., 2021), mengutip pendapat Asyik tahun 2010, menyatakan bahwa bahasa Aceh terdiri dari beberapa dialek diantaranya ialah dialek Banda Aceh, dialek Meulaboh, dialek Pidie, dan dialek Pase. Kemudian pada tahun 1978, penamaan dialek berubah menjadi dialek Aceh Pidie, dialek Aceh Barat, dialek, Aceh Besar, dan dialek Aceh Utara. Keunikan lain dari bahasa Aceh adalah terlihat pada pelafalan. Sebagaimana jika suatu dialek dibandingkan dengan dialek yang lain, maka ditemui dimana pelafalan kata yang berbeda, tetapi memiliki makna yang sama. Dan sebaliknya, dimana pelafalan kata sama, tetapi berbeda maknanya. Seperti contoh kata yang pelafalannya sama, tetapi maknanya berbeda ialah kata *singoh beungöh*. Kata *singoh beungöh* bermakna "besok" di daerah Aceh Selatan, sedangkan di daerah Aceh Utara bermakna "besok pagi". Kemudian kata yang pelafalannya berbeda, tetapi maknanya sama ialah kata *lôn* dan *lông* yang sama-sama bermakna "saya" (Nurpita dkk., 2021).

Berdasarkan bunyi, bahasa Aceh dibagi menjadi dua yaitu vokal dan konsonan. Vokal sendiri terbagi atas dua macam, diantaranya ialah vokal tunggal dan vokal rangkap. Konsonan juga terbagi atas dua macam, yaitu konsonan tunggal dan konsonan rangkap. Bahasa Aceh memiliki tujuh belas vokal tunggal, terdiri dari sepuluh yang dihasilkan melalui mulut, yaitu *a, i, e, è, é, eu, o, ô, ö* dan *u*; dan tujuh yang dihasilkan melalui hidung, yaitu *'a, 'i, 'e, 'eu, 'o, 'ö,* dan *'u*. Bahasa Aceh juga memiliki tujuh belas vokal rangkap, terdiri dari sepuluh berakhiran e yaitu *ie, èe, eue*,

oe, öe, ue, ‘ie, ‘ère, ‘eue, dan *‘ue*; dan tujuh vokal rangkap berakhiran i yaitu *ai, ‘ai, ei, oi, ôi, öi,* dan *ui*. Kemudian terdapat 24 konsonan tunggal yang terdiri dari *p, t, c, k, b, d, j, g, f, s, sy, h, m, n, ny, ng, mb, nd, nj, ngg, l, r, w,* dan *y*. Untuk konsonan rangkap terdapat 23 yang terbagi dengan akhiran h yaitu *ph, th, ch, kh, bh, dh, jh, gh, lh* dan *rh*; kemudian akhiran l yaitu *pl, cl, kl, bl,* dan *gl*; dan akhiran r yaitu *pr, tr, cr, kr, br, dr, jr,* dan *gr* (Rizal, 2015).

2.4. *Personal Extreame Programming* (PXP)

Personal Extreame Programming (PXP) merupakan salah satu model dari metode pengembangan *agile* yang dipersonalisasi untuk pengembangan aplikasi dapat dilakukan oleh seorang *software enginner* secara individu (Dzhurov dkk., 2009). PXP adalah kombinasi dari metode PSP (*Personal Software Process*) yang disederhanakan dengan penerapan praktik metode XP (*Extreame Programming*) yang kemudian disesuaikan untuk mudah diterapkan oleh pengembang aplikasi secara individu. PXP memiliki tahapan proses yang terdiri dari *Requirements, Planning, Iteration Inialization, Design, Implementation, System Testing* dan *Retrospective*. Gambar 2.1 dikutip dari jurnal (Dzhurov dkk., 2009), menunjukkan tahapan proses pada metode PXP dengan rincian sebagai berikut.

1. ***Requirements***

   Tahap *requirements* merupakan fase dimana informasi terkait kebutuhan aplikasi ditentukan, seperti kebutuhan fungsional dan non-fungsional dari aplikasi yang dikembangkan.

2. ***Planning***

   Pada tahap *planning*, pengembang menyusun serangkaian *user story* beserta rincian *task* yang harus dilakukan dalam mengerjakan suatu *user story* pada tiap iterasi berdasarkan *requirements* yang telah ditentukan dan kemudian membuat skala prioritas serta estimasi waktu terhadap *user story* dan *task* tersebut.

3. *Iteration Initialization*

Iteration inialization merupakan awal dari suatu iterasi (*sprint*). Iterasi dimulai dengan pengerjaan *user story* berdasarkan nilai prioritas yang telah ditentukan pada tahap *planning*, yang mana akan menjadi fokus sepanjang iterasi berlangsung. Panjang atau lama suatu iterasi dapat menjadi antara satu hingga tiga minggu bergantung pada tingkat kesulitan dari *user story* yang dikerjakan. Setiap iterasi berakhir dapat menghasilkan versi rilis dari *user story* atau fitur yang telah dibuat.

4. *Design*

Pada tahap ini, pengembang melakukan perancangan desain aplikasi dan pemodelan untuk *user story* yang sedang berlangsung dalam suatu iterasi. Desain aplikasi dan pemodelan yang dibuat hanya bertujuan untuk memenuhi *requirements* terhadap *user story* yang sedang dikerjakan tanpa mencoba menebak apa yang akan dibutuhkan di masa depan.

5. *Implementation*

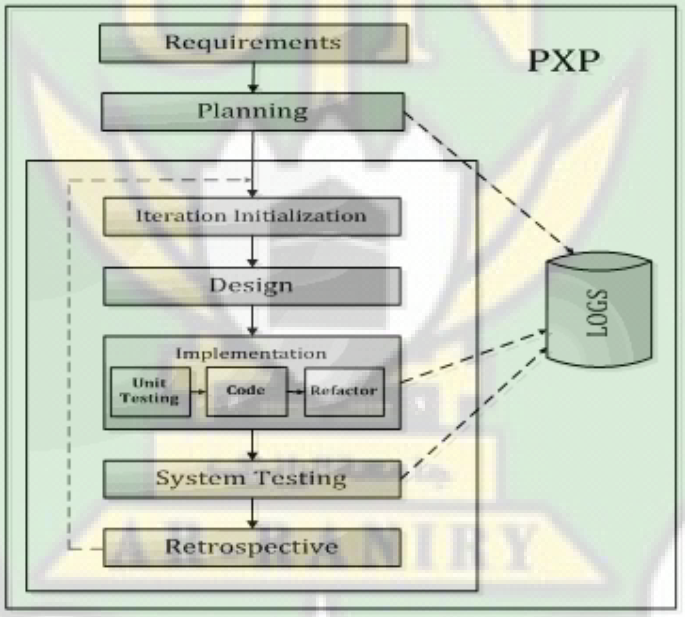
Tahap ini merupakan fase dimana kode program untuk pengembangan aplikasi dibuat berdasarkan desain yang telah dirancang dan selanjutnya menguji kode program tersebut. Tahap ini terdiri dari tiga sub-tahap, yaitu *unit testing*, *code*, dan *refactor*. Tahap *implementation* berakhir dengan kode di-*running* tanpa adanya *error* dan *unit testing* berhasil.

6. *System Testing*

System testing adalah jenis pengujian perangkat lunak yang mana proses pengujiannya dilakukan terhadap prilaku keseluruhan sistem yang terintegrasi (Girindra, t.t.; Suhartono, 2016). Pada fase ini, dilakukan pengecekan secara menyeluruh dan memastikan hasil implementasi yang telah diterapkan sesuai dengan *requirements* yang telah ditentukan. Setiap cacat yang ditemukan pada tahap ini dicatat dan diperbaiki.

7. *Retrospective*

Retrospective merupakan tahapan terakhir yang dilakukan pada suatu iterasi. Pada tahap ini dilakukan analisis terhadap proses pengembangan selama iterasi berlangsung, seperti kesesuaian estimasi waktu *planning* dengan waktu yang sebenarnya dihabiskan, dan alasan penyebab keterlambatan dalam proses pengembangan. Hal tersebut berguna untuk mencegah pengulangan kesalahan yang sama, sehingga pengerjaan pada iterasi berikutnya dapat dilakukan dengan lebih effektif. Tahap ini dapat selesai setelah melakukan perilisan aplikasi. Selanjutnya, iterasi atau iterasi berikutnya dapat dimulai kembali dari tahap *iteration inialization* atau proses pengembangan aplikasi dapat berakhir jika semua *requirements* telah terpenuhi tanpa adanya cacat yang tersisa.



Gambar 2.1 Tahapan Proses PXP (Dzhurov dkk., 2009)

2.5. REST API

REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000 dalam disertasinya. REST (*REpresentational State Transfer*) adalah gaya arsitektur pengembangan API yang menggunakan protokol HTTP (*Hypertext Transfer*

Protocol) sebagai media komunikasi antara *client* dan *server* untuk pertukaran data. API (*Application Programming Interface*) sendiri merupakan mekanisme yang memungkinkan dua komponen perangkat lunak dapat saling terhubung dengan serangkaian definisi dan protokol (aws.amazon.com, t.t.). Jadi bisa disebut, REST API adalah layanan *web* atau *web* API yang menerapkan gaya arsitektur REST dalam pembuatannya (Gupta, 2022).

Dalam praktiknya, *client* dan *server* saling bertukar data dengan menggunakan protokol HTTP. Tepatnya dengan HTTP *Method* seperti GET, POST, PUT, DELETE dan lainnya. Untuk format data yang biasa digunakan dalam berkomunikasi antar *client* dan *server* pada arsitektur REST ialah JSON. JSON (*Javascript Object Notation*) merupakan format data yang mudah dibaca oleh manusia dan juga komputer. Selain itu, JSON memiliki penulisan *code* yang lebih mudah dibandingkan format data lain seperti XML.

2.6. Flutter & Dart

Flutter merupakan kerangka kerja berupa UI (*User Interface*) *toolkit* sumber terbuka yang dikembangkan Google dengan kontributor dari seluruh dunia untuk membangun aplikasi *multi-platform* yang indah dengan peforma *native,* mulai dari aplikasi *mobile, web, desktop,* hingga *embedded device* dengan *single codebase*. Flutter mendukung produktifitas pengembang saat membangun aplikasi dengan adanya fitur *hot reload*. Dengan fitur tersebut *coding* yang diterapkan dalam pengembangan dapat dengan sekejap untuk melihat perubahan pada tampilan UI aplikasi (flutter.dev, t.t.).

Bahasa pemrograman yang didukung dalam pengembangan aplikasi dengan Flutter adalah Dart. Dart adalah bahasa pemrograman sumber terbuka yang *scalable* dengan *runtime* dan dukungan *library* yang kuat untuk membangun aplikasi di berbagai *platform* seperti *web, server,* dan *mobile app*. Bahasa pemrograman Dart sendiri dirancang sebagai bahasa yang dioptimalkan untuk kebutuhan pembuatan UI aplikasi yang berjalan cepat di semua *platform* (dart.dev, t.t.).

2.7. Golang & Gin *Web Framework*

Golang (Go *Language*) atau juga disebut Go merupakan bahasa pemrograman sumber terbuka yang dibuat di Google pada bulan September 2007, dan kemudian diperkenalkan ke publik pada November 2009. Golang mulai didiskusikan oleh Robert Griesemer, Ken Thompson dan Rob Pike untuk dikembangkan dengan tujuan untuk meningkatkan produktifitas pemrograman di era mesin jaringan *multi-core* dan basis kode besar yang menjadi tantangan teknis dalam pekerjaan sehari-hari mereka dan rekan kerja mereka di Google. Bahasa pemrograman Golang dapat digunakan untuk berbagai tujuan pengembangan perangkat lunak, seperti *Cloud & Network Services, Command-line Interface* (*CLIs*), *Web Development* dan *Development Operations* (*DevOps*) *& Site Reliability Engineering* (go.dev, t.t.).

Gin adalah *framework* untuk pengembangan aplikasi *web* yang ditulis dengan bahasa pemrograman Go (Golang). Disebutkan dalam halaman *web* dokumentasi Gin, *framework* ini dapat bekerja 40 kali lebih cepat dibandingkan Go *web framework* lain yang memiliki fitur API yang mirip seperti Martini. Gin menyediakan berbagai fitur untuk mendukung kebutuhan pengembangan aplikasi seperti kecepatan, dukungan *middleware, crash-free,* JSON *validation, routes grouping, error management, rendering built-in,* dan kemudahan dalam modifikasi pembuatan *middleware* sesuai kebutuhan (gin-gonic.com, t.t.).

2.8. Algoritma Jaro-Winkler

Algoritma Jaro-Winkler merupakan algoritma varian dari Jaro Distance metrik untuk mencocokkan atau mengukur tingkat kesamaan antara dua buah *string.* Semakin tinggi nilai dari dua buah *string*, maka persentase kesamaan atau kemiripan antara dua *string* semakin tinggi (Syahputra & Syakti, 2022). Nilai jarak Jaro berkisar antara 0 sampai 1. Dimana nilai 1 berarti kedua *string* adalah sama, sedangkan nilai 0 berarti tidak ada kemiripan antara kedua *string*.

Terdapat tiga langkah yang menjadi dasar dari algoritma Jaro-Winkler dalam menghitung kemiripan atau kesamaan antara dua buah *string,* yaitu:

1. Menghitung panjang kedua *string* yang dibandingkan.

2. Menghitung jumlah karakter yang sama antara kedua *string* yang dibandingkan.
3. Menghitung jumlah transposisi.

Rumus untuk menghitung jarak (d_j) antara dua *string* pada algoritma Jaro Distance yaitu:

$$d_j = \frac{1}{3}\left(\frac{m}{S_1} + \frac{m}{S_2} + \frac{m - t}{m}\right) \quad (2.1)$$

Keterangan:

- S_1 = Panjang *string* 1
- S_2 = Panjang *string* 2
- m = Jumlah karakter yang sama antara S_1 dan S_2
- t = Jumlah transposisi

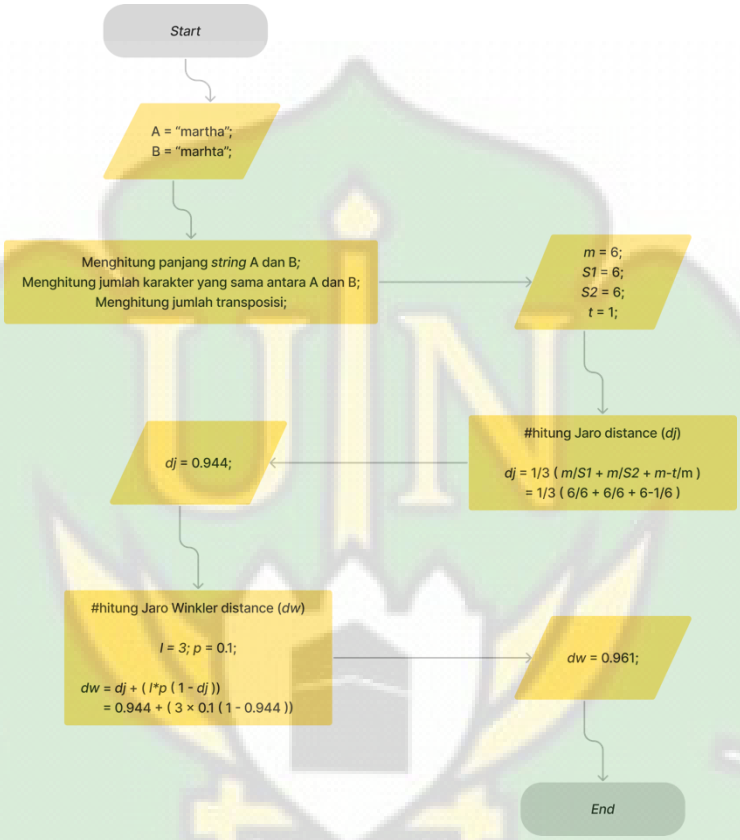
Jaro-Winkler menggunakan *prefix scale* (*p*) untuk memberikan tingkat penilaian yang lebih baik, dan *prefix length* (*l*) yang menyatakan panjang karakter yang sama dari *string* yang dibandingkan sebelum ditemukan ketidaksamaan karakter (maksimal empat karakter). Menurut Winkler, nilai standar konstanta *prefix scale* (*p*) yaitu p = 0.1 (Novantara & Pasruli Opin, 2018). Jaro-Winkler Distance dihitung dengan rumus sebagai berikut:

$$d_w = d_j + \left(l * p\left(1 - d_j\right)\right) \quad (2.2)$$

Keterangan:

- d_j = Jaro Distance untuk kedua *string* yang dibandingkan
- l = *Prefix length* dengan maksimal empat karakter
- p = Konstanta *scaling factor* dengan nilai 0.1

Berikut ada Gambar 2.2 dapat dilihat contoh perhitungan algoritma Jaro-Winkler untuk mencari kemiripan atau kesamaan antara *string* "martha" dan "marhta".



Gambar 2.2 Diagram Alur Contoh Perhitungan Algoritma Jaro-Winkler

Sumber: Data diolah (2023)

2.9. *Mean Average Precision* (MAP)

Mean Average Precision (MAP) adalah salah satu metode perhitungan untuk mengukur kinerja ataupun mengevaluasi peforma dari suatu sistem rekomendasi. Metode ini cocok digunakan untuk algoritma yang mengembalikan urutan peringkat item dalam daftar, di mana setiap item dapat dianggap relevan atau tidak relevan. Dengan menghitung MAP, kita dapat mengetahui seberapa baik sistem rekomendasi dalam menentukan item yang relevan untuk setiap *query*. Nilai MAP dihitung dengan

menggunakan rata-rata dari nilai *Average Precision* (AP) untuk setiap *query* yang diberikan kepada sistem (Aditya Widjaja & Novianus Palit, 2022; Arfisko & Wibowo, 2022).

2.10. Penelitian Terdahulu

Adapun penelitian terdahulu yang relevan untuk menjadi acuan penulis dalam meneliti dan menghindari kesamaan hasil penelitian adalah sebagai berikut.

1. **Penelitian Algoritma Jaro-Winkler**

   Penelitian sebelumnya oleh Yulianingsih (2017) telah menunjukkan bahwa algoritma Jaro-Winkler memiliki kompleksitas waktu *quadratic runtime complexity* yang sangat efektif pada *string* pendek dan dapat memperoleh hasil lebih cepat dibandingkan dengan algoritma Levenshtein Distance. Selain itu, algoritma Jaro-Winkler juga mudah untuk diimplementasikan dan efektif dalam hasil yang dicapai (Yulianingsih, 2017). Penelitian ini menambahkan kepada pengetahuan yang ada dengan menunjukkan bahwa algoritma Jaro-Winkler juga dapat digunakan sebagai dasar menentukan penggunaan algoritma pencarian data untuk beberapa jenis kegiatan lainnya yang serupa.

   Penelitian lainnya oleh Kristien Margi S, dan Agus T (2016) telah menunjukkan bahwa algoritma Jaro-Winkler dapat diimplementasikan untuk pengkoreksian pada sebuah *keyword* bahan pustaka dan juga dapat diterapkan untuk memberikan *suggestion word* jika ada salah pengetikan atau *typo* pada saat pencarian bahan pustaka. Hasil dari penelitian ini menunjukkan bahwa *suggestion word* yang diberikan dapat langsung menghubungkan dengan judul buku yang dimaksud (Kristien Margi S & Agus T, 2016). Penelitian ini memperkuat temuan terdahulu bahwa algoritma Jaro-Winkler dapat digunakan dalam sistem pencarian bahan pustaka untuk mempermudah pengguna perpustakaan dalam mencari buku.

   Selain itu penelitian yang telah dilakukan oleh Glenn Tambahani dkk., (2021) menunjukkan bahwa metode Jaro-Winkler dapat digunakan untuk

mencari kemiripan kata dalam Bahasa Inggris dengan baik. Penelitian ini menemukan bahwa dalam mengenali kemiripan antara kata kunci dengan inputan suara, metode Jaro-Winkler memperoleh skor terbaiknya dengan nilai 100% dan nilai terendahnya 45.83% (Glenn Tambahani dkk., 2021). Hasil ini menunjukkan bahwa aplikasi English Pronunciation Test yang dibangun dengan menggunakan metode Jaro-Winkler dapat membantu para pengajar dalam melatih pengucapan (*pronunciation*) dari peserta didiknya.

Pada penelitian yang telah dilakukan oleh Agung Prasetyo dkk., tahun 2021 mengenai penerapan algoritma Jaro-Winkler Distance untuk fitur *autocorrect* dan *spelling suggestion* pada penulisan naskah bahasa Indonesia di BMS TV, dapat disimpulkan bahwa fitur tersebut dapat menangani kesalahan penulisan ejaan kata pada 49 kata dari 60 kata yang diuji dengan baik. Penelitian tersebut juga menunjukkan bahwa fitur *autocorrect* dan *spelling suggestion* dapat membantu News Director BMS TV dalam memeriksa dan memperbaiki kesalahan penulisan kata secara otomatis serta memberikan saran penulisan ejaan kata yang benar dalam bahasa Indonesia (Prasetyo dkk., 2018).

2. Penelitian Aplikasi Kamus Bahasa Aceh

Dalam penelitian yang dilakukan oleh Agil M. Caesar pada tahun 2018, mereka mengembangkan aplikasi Kamus Bahasa Aceh berbasis Android yang sangat praktis dan mudah digunakan sebagai pedoman awal untuk menerjemahkan kata-kata yang umum dipakai di masyarakat. Aplikasi ini dibangun dengan menggunakan teknologi Android Studio dengan bahasa pemrograman Java dan hanya dapat diimplementasikan pada *smartphone* android. Namun demikian, aplikasi ini hanya bisa menerjemahkan dari bahasa Indonesia ke bahasa Aceh, tidak bisa sebaliknya (Agil M. Caesar, 2018).

Pada penelitian lain yang dilakukan oleh Pebrijayanti dan Ardian pada tahun 2018 dengan judul “Rancang Bangun Aplikasi Kamus Bahasa Indonesia - Bahasa Aceh Menggunakan Metode *Rule Based* Berbasis Android”, dapat ditarik kesimpulan bahwa aplikasi kamus bahasa Indonesia - bahasa Aceh yang

dirancang menggunakan metode *Rule Based* berbasis Android dapat membantu pengguna dalam mempermudah pencarian kosakata dalam bahasa Aceh, serta berjalan secara sempurna di *smartphone* android dan secara fungsional mengeluarkan hasil yang sesuai dengan yang diharapkan (Pebrijayanti & Ardian, 2018).

Selanjutnya, hasil penelitian yang dilakukan oleh Faathir tahun 2018 yang berjudul "Perbandingan Algoritma Hosrpool Dan Not So Naive Dalam Pembuatan Kamus Bahasa Indonesia – Bahasa Aceh Beerbasis Android", dapat disimpulkan bahwa algoritma *Not So Naïve* lebih cepat daripada algoritma *Horspool* dalam proses pencocokan *string* pada aplikasi Kamus Bahasa Indonesia – Bahasa Aceh. Algoritma *Not So Naïve* memiliki rata-rata *running time* 18,9 ms sedangkan algoritma *Horspool* memiliki rata-rata *running time* 26,2 ms. Selain itu, algoritma *Horspool* membutuhkan waktu lebih lama untuk pencarian kata pendek dibandingkan dengan kata panjang. Hasil kompleksitas algoritma *Horspool* adalah *O(n)* sedangkan hasil kompleksitas algoritma *Not So Naïve* adalah *O(mn)*. Namun, algoritma *Not So Naïve* tidak dapat mencari 1 huruf saja karena membutuhkan minimal 2 huruf awal yang dibandingkan untuk menentukan nilai variabel *k* dan *ell* pada proses pencarian (Faathir, 2018).

Penelitian sebelumnya yang disebutkan diatas telah menunjukkan bahwa algoritma Jaro-Winkler dapat digunakan untuk pengkoreksian pada sebuah *keyword*, memberikan *suggestion word* pada saat pencarian bahan pustaka, mencari kemiripan kata dalam Bahasa Inggris, dan menangani kesalahan penulisan ejaan kata pada penulisan naskah bahasa Indonesia. Namun, penelitian sebelumnya belum mengeksplorasi kemampuan algoritma Jaro-Winkler dalam melakukan pencarian kosakata bahasa Aceh atau menangani kesalahan pengetikan kata pada saat melakukan pencarian dalam bahasa Aceh. Dimana diketahui bahwa bahasa Aceh memiliki keunikan penulisan kosakata yang disebut tanda diakritik. Oleh karena itu, penelitian ini akan mencoba mengembangkan aplikasi kamus bahasa Aceh dengan menerapkan fitur *recommendation list* kata menggunakan algoritma Jaro-Winkler pada fitur pencarian dan menguji kemampuannya dalam memberikan rekomendasi

kata yang sesuai dengan kata kunci yang dimasukkan oleh pengguna. Hal ini akan memberikan tambahan manfaat bagi pengguna aplikasi yang ingin mencari kata yang tidak tepat atau tidak tahu cara mengetik kata yang sesuai dengan penulisan bahasa Aceh.

Adapun pada Tabel 2.1 dapat dilihat perbedaan dari hasil penelitian terdahulu yang menunjukkan kesamaan dan kekurangan dengan penelitian ini.

Tabel 2.1 Kesamaan dan Kekurangan dengan Penelitian Terdahulu

No	Peneliti	Judul	Hasil	Kesamaan	Kekurangan
1	Agil M. Caesar, 2018	Perancangan Aplikasi Kamus Bahasa Aceh Berbasis Android	Pada penelitian tersebut, peneliti menghasilkan aplikasi kamus bahasa Aceh berbasis Android menggunakan Android Studio dan bahasa Pemrograman Java. Aplikasi yang dihasilkan bersifat *offline* dan hanya mendukung *platform* Android.	Menggunakan sumber data fisik yang sama, yaitu Kamus Bahasa Aceh Indonesia Inggris disusun oleh Drs. Abdullah Fardan dan Drs. Zulkarnaini yang diterbitkan oleh CV Boebon Jaya (Fardan & Zulkarnaini, 2019).	1. Hanya mendukung satu *platform*, yaitu Android. 2. Data yang ditampilkan pada aplikasi tidak mengandung tanda diakritik sebagaimana yang terdapat pada sumber data. 3. Aplikasi belum menerapkan algoritma pencocokan *string* pada fitur pencarian.

Tabel 2.1 Kesamaan dan Kekurangan dengan Penelitian Terdahulu

No	Peneliti	Judul	Hasil	Kesamaan	Kekurangan
					4. Jumlah kata yang dimasukkan ke *database* sebanyak 300 kata dan berfokus pada data bahasa Aceh dan bahasa Indonesia.
2	Pebrijayanti dkk., 2018	Rancang Bangun Aplikasi Kamus Bahasa Indonesia - Bahasa Aceh Menggunakan Metode Rule Based Berbasis Android	Penelitian tersebut menghasilkan aplikasi kamus bahasa Aceh berbasis Android yang dirancang menggunakan Eclipse dan bahasa pemrograman Java. Aplikasi bersifat *offline* dan hanya mendukung *platform* Android.		1. Aplikasi tidak tersedia di Google Play Store. 2. Android versi 2.1 (Éclair) tidak didukung lagi oleh Google sejak 30 Juni 2017.
3	Faathir M, 2018	Perbandingan Algoritma *Hosrpool* dan *Not So Naïve* dalam	Hasil penelitian perbandingan kompleksitas waktu dan *runtime* dari	Menggunakan sumber data fisik yang sama, yaitu Kamus Bahasa	1. Hanya mendukung satu *platform,* yaitu Android.

Tabel 2.1 Kesamaan dan Kekurangan dengan Penelitian Terdahulu

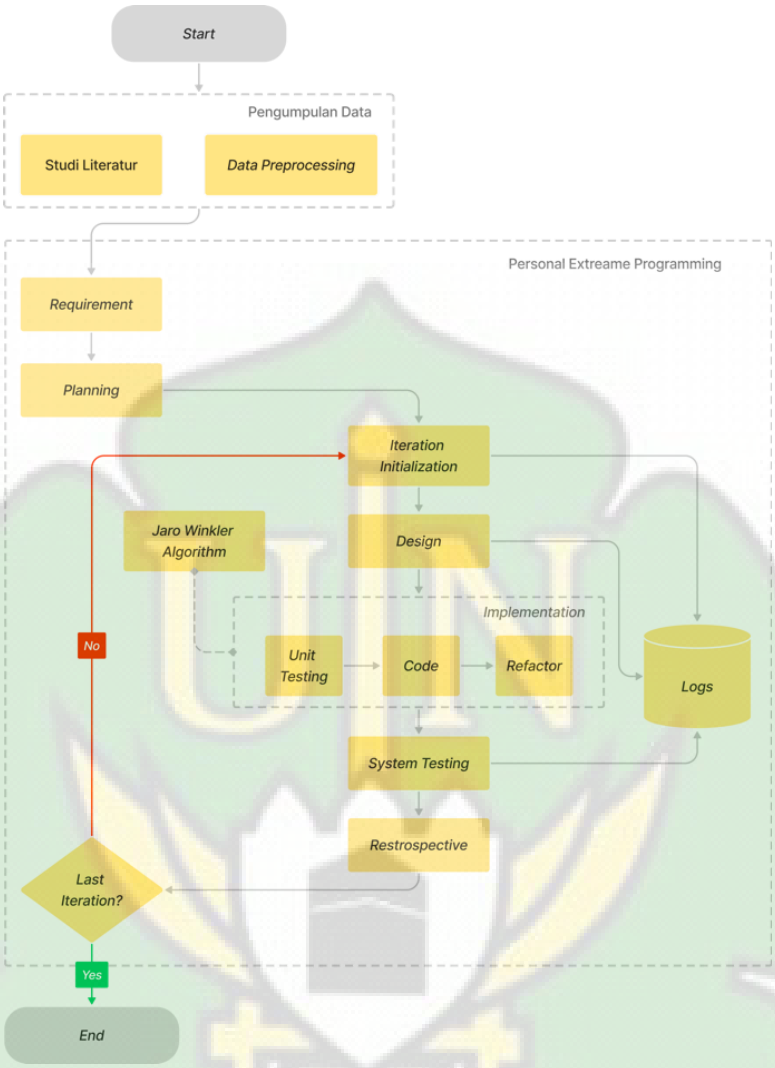
No	Peneliti	Judul	Hasil	Kesamaan	Kekurangan
		Pembuatan Kamus Bahasa Indonesia - Bahasa Aceh Berbasis Android	algoritma *Hosrpool* dan algoritma *Not So Naïve* menggunakan Android Studio dan bahasa pemrograman Java dalam pembuatan aplikasi Kamus Bahasa Indonesia – Bahasa Aceh menunjukkan bahwa, algoritma *Horspool* lebih lambat dibandingkan algoritma *Not So Naïve*. Dimana hasil rata-rata *runtime* algoritma *Hosrpool* adalah 26,6 ms, sedangkan algoritma *Not So Naive* adalah 18,9 ms.	Aceh Indonesia Inggris disusun oleh Drs. Abdullah Fardan dan Drs. Zulkarnaini yang diterbitkan oleh CV Boebon Jaya (Fardan & Zulkarnaini, 2019).	2. Data yang ditampilkan pada aplikasi tidak mengandung tanda diakritik sebagaimana yang terdapat pada sumber data.
					3. Jumlah kata yang dimasukkan ke *database* sebanyak 500 kata dan berfokus pada data bahasa Aceh dan bahasa Indonesia.

Sumber: Data diolah (2023)

BAB III
METODE PENELITIAN

3.1. Tahapan Penelitian

Penelitian ini dikerjakan dengan tahapan-tahapan yang diadaptasi berdasarkan langkah dalam melakukan pengembangan perangkat lunak yang mengacu pada metode *Personal Extreame Programming* (PXP). Metode PXP dipilih karena pengembangan aplikasi dilakukan oleh pengembang tunggal, hal ini sesuai dengan kondisi penulis yang melakukan pengembangan aplikasi secara individu. Selain itu, berdasarkan proses pengembangan PXP yang bersifat iteratif dalam praktiknya memungkinkan pengembang lebih mudah dalam menghadapi perubahan kebutuhan yang terjadi saat proses pengembangan aplikasi berlangsung. Adapun tahapan penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Penelitian

Sumber: Data diolah (2023)

Pada subbab berikut akan dijelaskan rincian terkait diagram alur penelitian yang digambarkan pada Gambar 3.1.

3.2. Metode Pengumpulan Data

Pada bagian ini akan diuraikan tentang dua poin dalam metode pengumpulan data yang dilakukan dalam penelitian ini, yaitu terkait studi literatur dan data *preprocessing.*

3.2.1. Studi Literatur

Pada tahap ini, penulis mengumpulkan berbagai data atau informasi yang terkait dengan bahasa Aceh dan perancangan aplikasi yang dapat mendukung menyelesaikan penelitian ini. Data atau informasi tersebut didapatkan dari berbagai sumber seperti jurnal, buku, penelitian terdahulu, artikel dan literatur lainnya.

Selain itu, penulis juga melakukan eksplorasi terhadap aplikasi kamus bahasa Aceh dari hasil penelitian terdahulu dan pada aplikasi yang tersedia di toko aplikasi saat penelitian ini dilakukan. Apakah aplikasi tersebut menerapkan atau menyediakan data kamus bahasa Aceh dengan penulisan katanya dibubuhi tanda diakritik pada huruf vokal tertentu sesuai dengan kaidah penulisan bahasa Aceh atau tidak.

3.2.2. Data *Preprocessing*

Data *preprocessing* yang dilakukan pada penelitian ini terdiri dari beberapa tahapan yaitu, tahap *Input*, *Processing*, dan *Storage*. Gambar 3.2 menjelaskan alur pemrosesan data dengan rincian sebagai berikut.

1. ***Input***

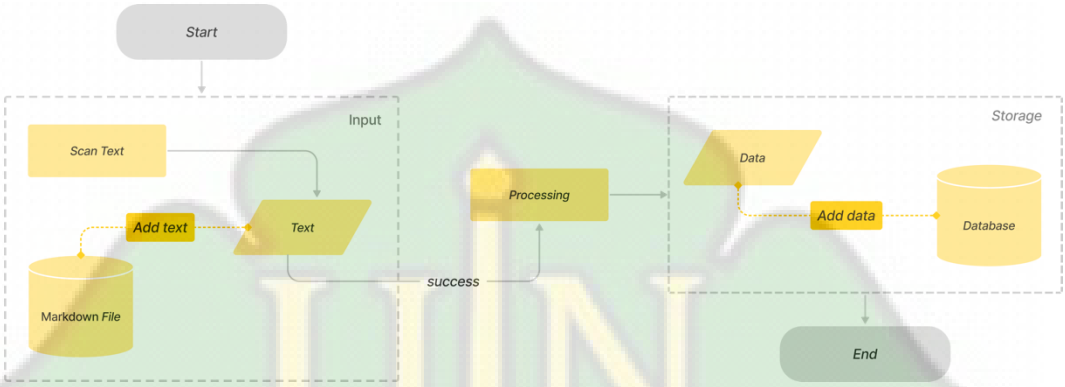
   Pada tahap ini, dilakukan pemindaian teks kosakata bahasa Aceh yang terdapat didalam kamus dengan menggunakan aplikasi Google Lens. Selanjutnya, teks yang telah dipindai tersebut kemudian disimpan ke dalam *file* dengan format Markdown.

2. ***Processing***

   Data teks yang telah disimpan sebelumnya, kemudian dilakukan pemrosesan terhadap data tersebut. Pada tahap ini, dilakukan perbaikan dan penyusuaian antara data yang terdapat di kamus dengan data yang telah dipindai dan disimpan sebelumnya.

3. ***Storage***

Tahap selanjutnya setelah pemrosesan data yaitu menyimpan data hasil pemrosesan untuk disimpan ke dalam *database*. Sehingga dengan begitu dapat digunakan untuk pengembangan aplikasi kamus bahasa Aceh.



Gambar 3.2 Diagram Alur Data *Preprocessing*

Sumber: Data diolah (2023)

3.3. Metode Pengembangan Aplikasi

Metode *Personal Extreame Programming* (PXP) dalam prosesnya terdiri dari beberapa tahapan yang dapat dilihat pada BAB II TINJAUAN PUSTAKA. Tahapan tersebut bersifat iteratif kecuali tahap *requirements dan planning*, karena kedua tahapan ini dilakukan dan ditentukan untuk keseluruhan pengembangan aplikasi. *Planning* dapat direvisi jika terjadi perubahan *requirements.* Dan proses iterasi dilakukan mulai dari *iteration inialization* dan diakhiri dengan iterasi *retrospective.* Selama proses PXP, pengembang menyimpan *file logs* dengan informasi terkait perencanaan tugas, durasi aktual, saran peningkatan, serta jumlah dan detail cacat (Dzhurov dkk., 2009). Berikut rincian tahap-tahap pengembangan aplikasi dengan metode PXP.

3.3.1. *Requirements*

Tahapan pertama dalam *Personal Extreame Programming* (PXP) adalah menentukan kebutuhan atau persyaratan dalam pengembangan aplikasi seperti kebutuhan fungsional dan non-fungsional.

1. Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fungsi-fungsi dan fitur yang disediakan pada aplikasi (Melinda dkk., 2017). Adapun kebutuhan fungsional aplikasi kamus bahasa Aceh berbasis *mobile* ini adalah sebagian besar diadopsi atau diperoleh dari fitur aplikasi kamus bahasa Aceh yang sudah tersedia di toko aplikasi dan dari hasil penelitian terdahulu. Namun, pada penelitian ini dilakukan peningkatan seperti pada antarmuka pengguna aplikasi dan beberapa fitur tambahan. Kebutuhan fitur tersebut ditulis dalam bentuk *user story* dengan format "Sebagai <jenis pengguna>, saya ingin <melakukan tindakan tertentu> sehingga <mendapatkan manfaat dari tindakan tersebut>". Pada Tabel 3.1 dapat dilihat penjelasan fitur-fitur yang akan dibuat, kemudian juga dirincikan *user stories* dari fitur-fitur tersebut.

Tabel 3.1 Fitur Aplikasi

No	Fitur	Deskripsi
1	*Authentication*	Fitur ini akan menangani kebutuhan proses otentikasi pengguna seperti *login, logout,* dan *register*.
2	*User Profile*	Fitur ini akan menangani kebutuhan terkait menampilkan informasi pengguna yang telah terdaftar.
3	*Search*	Fitur ini akan menangani kebutuhan terkait pencarian kata bahasa Aceh pada aplikasi, termasuk dengan *recommendation list*.

Tabel 3.1 Fitur Aplikasi

No	Fitur	Deskripsi
4	*Dictionary*	Fitur ini akan menangani kebutuhan terkait menampilkan daftar kata bahasa Aceh dan detail dari suatu kata pada aplikasi.
5	*Bookmark*	Fitur ini menangani kebutuhan terkait menandai atau menyimpan suatu kata yang diperlukan pengguna.

Sumber: Data diolah (2023)

Berikut *user stories* dari kebutuhan fitur yang telah dikategorikan dapat dilihat pada Tabel 3.2.

Tabel 3.2 *User Story*

Fitur	Judul *User Story*	*User Story*
Authentication	*Sign Up & Create an Account*	Sebagai *user*, saya ingin mendaftar atau membuat sebuah akun sehingga saya dapat *login* ke aplikasi.
	Sign In	Sebagai *user* yang terdaftar, saya ingin *login* ke aplikasi dengan akun yang terdaftar, sehingga saya bisa akses fitur yang terkunci.
	Splace Screen	Sebagai *user* yang sudah melakukan proses *login*, saya ingin tidak melakukan *login* kembali ketika saya membuka aplikasi di waktu yang lain, sehingga saya dapat mengakses fitur terkunci lebih cepat.

Tabel 3.2 *User Story*

Fitur	**Judul *User Story***	***User Story***
User Profile	*User Profile Detail*	Sebagai *user* yang sudah terotentikasi, saya ingin melihat informasi data saya, sehingga saya dapat mengetahui jika saya sudah terotentikasi pada aplikasi.
Dictionary	*Show All Word*	Sebagai *user*, saya ingin melihat daftar kata kamus pada aplikasi, sehingga saya dapat mengetahui kata apa saja yang tersedia.
	Word Detail	Sebagai *user*, saya ingin melihat informasi rinci terkait sebuah kata, sehingga saya dapat mengetahui dengan baik terkait suatu kata tersebut seperti makna dan gambaran objek nya.
Search	*Recommendation List*	Sebagai *user*, saya ingin aplikasi dapat memberikan rekomendasi kata yang berdekatan dengan kata kunci pencarian yang saya input, sehingga saya dapat menemukan kata tanpa kesulitan jika saya melakukan kesalahan pengetikan pada kata kunci pencarian.
Bookmark	*Mark & Unmark Word*	Sebagai *user*, saya ingin bisa menandai dan menyimpan kata yang saya temui atau saya cari di aplikasi, sehingga saya dapat mengakses kata tersebut dengan cepat.
	Bookmark List	Sebagai *user*, saya ingin melihat daftar kata yang telah saya tandai, sehingga saya dapat memastikan jika kata yang ditandai telah disimpan dengan baik dan dapat saya akses lebih cepat.

Tabel 3.2 *User Story*

Fitur	Judul *User Story*	*User Story*
	Remove Bookmark Item	Sebagai *user*, saya ingin memiliki kemampuan menghapus kata yang saya tandai dari daftar kata yang tertanda, sehingga saya dapat menghapus kata yang tidak perlu atau tidak penting untuk saya tandai lagi.
	Remove All Bookmark	Sebagai *user*, saya ingin menghapus semua kata yang ada didaftar *bookmark* sekaligus, sehingga saya dapat menghapus lebih cepat.

Sumber: Data diolah (2023)

2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menjelaskan batasan dari fungsi atau fitur yang disediakan aplikasi. Batasan kebutuhan tersebut termasuk perangkat lunak dan perangkat keras yang digunakan dalam pengembangan aplikasi (Melinda dkk., 2017). Perangkat lunak yang digunakan dalam pengembangan aplikasi kamus bahasa Aceh pada penelitian ini dapat dilihat pada Tabel 3.3.

Tabel 3.3 Perangkat Lunak

No	Nama	Versi
1	Visual Studio Code	1.73.0
2	Android Stuido	2021.1
3	Xcode	14.0.1
4	Flutter	3.3.4-0.0.pre.1 (*stable*)

Tabel 3.3 Perangkat Lunak

No	Nama	Versi
5	Dart	2.18.2
6	Go	go1.17.5
7	Gin *Web Framework*	1.8.1
8	PostgreSQL	-
9	Postman	10.4.6
10	Git	2.37.0
11	Github	-
12	Figma	116.4.4
13	Google Lens	-
14	Supabase Schema	-

Sumber: Data diolah (2023)

Adapun perangkat keras yang digunakan dalam membangun aplikasi kamus bahasa Aceh pada penelitian ini dapat dilihat pada Tabel 3.4.

Tabel 3.4 Perangkat Keras

No	Nama	Versi
1	MacBook Air	M1, 2020
2	macOS Monterey	12.4
3	RAM 8 GB	-
4	Minimal *Disk Space* 5 GB	-

Sumber: Data diolah (2023)

3.3.2. *Planning*

Adapun pada tahap ini, penulis menentukan estimasi waktu pengerjaan dan prioritas pengerjaan dari setiap *user story*. Selanjutnya penulis melakukan perencanaan iterasi dengan menyusun *user story* yang akan dikerjakan pada setiap iterasi.

Estimasi waktu pengerjaan diberikan dalam bentuk *story point.* Nilai dari *story point* ini dapat dikonversi menjadi satuan jam atau hari tergantung kebutuhan. Pada penelitian ini penulis menggunakan satuan jam sebagai nilai dari *story point*, 1 *story point* adalah 4 jam. Prioritas suatu *user story* ditentukan berdasarkan nilai bisnis yang diperingkatkan secara subjektif dari *user story* tersebut. Nilai bisnis tersebut terbagi tiga, yaitu 1 yang berarti produk tidak dapat diluncurkan tanpa fitur; 2 berarti produk tidak dapat diluncurkan tanpa fitur, tapi tidak harus segera ditangani; 3 berarti fitur bersifat opsional berdasarkan sumber daya, waktu, dan resiko. Selanjutnya, untuk *risk* dari *user story* ditentukan berdasarkan peringkat subjektif dari ketidakpastian relatif terkait keberhasilan penyelesaian *user story*. Nilai peringkat dibagi dalam tiga kategori, yaitu 1 adalah tinggi; 2 adalah sedang; dan 3 adalah rendah (KathrynEE dkk., 2022).

Setelah mendapatkan daftar *user story* beserta estimasi pengerjaannya dalam bentuk *story point,* selanjutnya dilakukan perencanaan iterasi. Pada perencanaan iterasi, penulis menentukan *velocity* untuk mengetahui jumlah *user story* yang dapat dikerjakan dalam suatu iterasi. Penulis menentukan nilai maksimum *velocity* yaitu, 12 *story point* untuk setiap iterasi. Dengan begitu penulis dapat menyelesaikan suatu iterasi dalam waktu 6-7 hari dengan upaya (*effort*) pengerjaan per harinya adalah 8 jam. *User story* dengan nilai prioritas lebih tinggi dimasukkan ke dalam iterasi yang lebih awal untuk dikerjakan.

Pada Tabel 3.5 dapat dilihat perencanaan iterasi pada penelitian ini. Diketahui pada tabel tersebut terdapat 34 jumlah *story point*, sehingga dibutuhkan 3 kali iterasi untuk menyelesaikan *user story* yang tersedia.

Tabel 3.5 Perencanaan Iterasi

Iterasi	*User Story*	Prioritas	Resiko	*Story Point*
Iterasi 1	*Show All Word*	1	Rendah	3
	Word Detail	1	Sedang	4
	Recommendation List	1	Tinggi	4
Velocity				**11**
Iterasi 2	*Create an Account*	2	Sedang	4
	Sign In	2	Sedang	3
	User Profile Detail	2	Rendah	3
	Splace Screen	2	Rendah	1
Velocity				**11**
Iterasi 3	*Mark & Unmark Word*	2	Rendah	3
	Bookmark List	2	Rendah	3
	Remove Bookmark Item	2	Rendah	3
	Remove All Bookmark	2	Rendah	3
Velocity				**12**

Sumber: Data diolah (2023)

3.3.3. *Iteration Inialization*

Tahap ini menunjukkan awal dari setiap iterasi yang akan dilakukan selama pengembangan sistem. Proses iterasi dilakukan berdasarkan hasil perencanaan iterasi yang telah tersusun pada tahap *planning.*

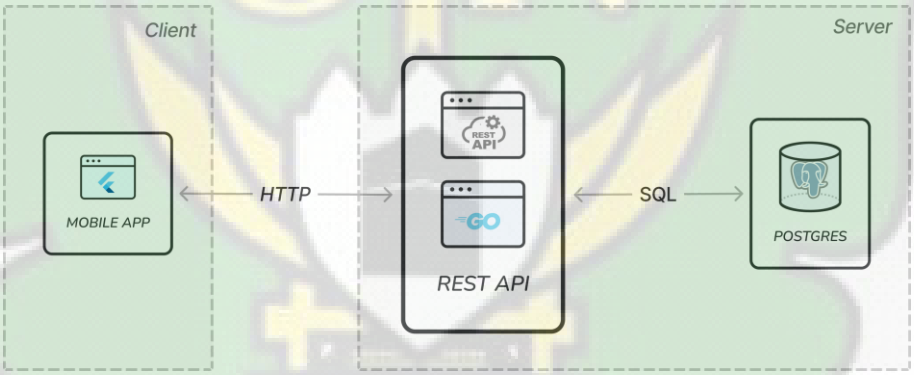
3.3.4. *Design*

Pada tahap ini, penulis melakukan perancangan desain antarmuka pengguna aplikasi dan pemodelan sistem yang akan diimplementasikan selama proses iterasi berlangsung. Desain antarmuka pengguna dan pemodelan sistem yang dibuat hanya

bertujuan untuk memenuhi *requirements* dari *user story* tanpa mencoba menebak apa yang akan dibutuhkan di masa depan.

1. Arsitektur Aplikasi

Aplikasi yang dirancang pada penelitian ini menerapkan gaya arsitektur REST (*REpresentational State Transfer*). Salah satu dari prinsip REST adalah penerapan pola desain *client-server*, dimana masalah disisi *client* dan masalah disisi *server* dipisah. Flutter dan bahasa pemrograman Dart digunakan untuk menangani masalah antarmuka pengguna disisi *client*, sedangkan disisi *server* menggunakan bahasa pemrograman Golang, Gin *framework* dan *database* PostgreSQL untuk menangani masalah pengolahan dan penyimpanan data. Pada Gambar 3.3 dapat dilihat gambaran dari arsitektur aplikasi yang akan dirancang.



Gambar 3.3 Arsitektur Aplikasi

Sumber: Data diolah (2023)

2. Mid-fidelity Wireframe

Mid-fidelity wireframe merupakan sketsa atau kerangka gambar yang menampilkan struktur desain dari aplikasi dengan detail fitur yang cukup jelas. *Mid-fidelity wireframe* dibuat dengan warna hitam putih atau sedikit abu-abu untuk memperjelas elemen-elemen yang ada pada suatu layer aplikasi (Costa,

2020; Handayani, 2022). Berikut sketsa atau desain *mid-fidelity wireframe* pada penelitian perancangan aplikasi kamus bahasa Aceh.

a. *Splace Screen*

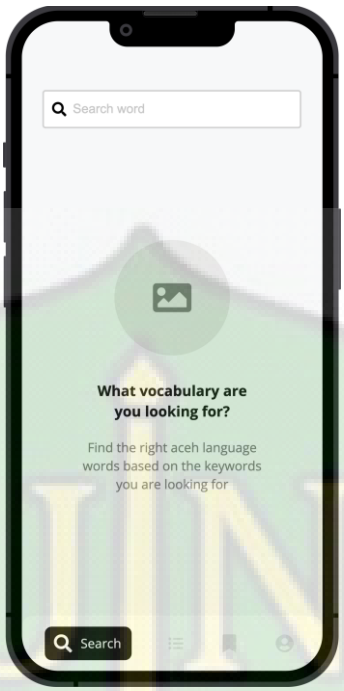
Splace Screen merupakan halaman yang muncul saat pertama kali aplikasi dibuka. Halaman ini akan tampil di layar selama beberapa detik sebelum halaman utama aplikasi ditampilkan ke layar perangkat pengguna. Pada Gambar 3.4 dapat dilihat *wireframe* dari halaman *Splace Screen.*

Gambar 3.4 *Splace Screen – Wireframe*

Sumber: Data diolah (2023)

b. *Search Screen*

Search Screen adalah halaman untuk fitur pencarian kata pada aplikasi kamus bahasa Aceh. Fitur ini dapat diakses dengan menekan tombol *icon search* dibagian *bottom navigation bar.* Adapun *wireframe* halaman *Search Screen* dapat dilihat pada Gambar 3.5.

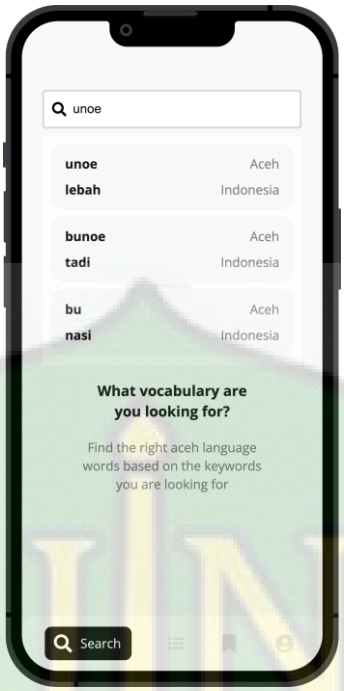


Gambar 3.5 *Search Screen - Wireframe*

Sumber: Data diolah (2023)

c. *Search Screen (State on searching)*

Ini adalah halaman *Search Screen* dengan *state* dimana sedang melakukan pencarian kata. Fitur *recommendation list* ditampilkan setelah memasukkan kata kunci yang ingin dicari. *Wireframe* halaman *Search Screen* dengan *state on searching* dapat dilihat pada Gambar 3.6.

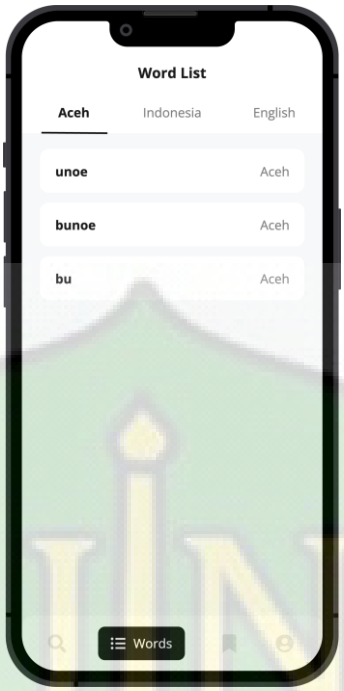


Gambar 3.6 *Search Screen (State on searching) – Wireframe*

Sumber: Data diolah (2023)

d. *Word List Screen*

Halaman ini merupakan halaman yang menampilkan daftar kosakata *(word list)* yang tersedia pada *database*. Pada Gambar 3.7 dapat dilihat *wireframe* dari halaman *Word List Screen.*

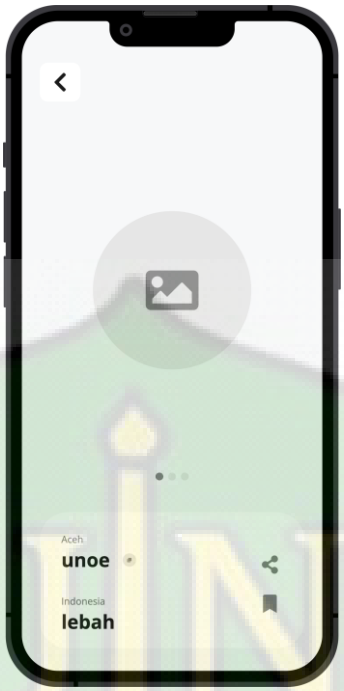


Gambar 3.7 *Word List Screen – Wireframe*

Sumber: Data diolah (2023)

e. *Detail Screen*

Detail Screen adalah halaman yang menampilkan detail dari suatu kata. Halaman ini akan muncul ketika *user* menekan salah satu *item* pada daftar kata yang tampil di halaman *Word List Screen* dan pada *item* yang muncul di fitur *recommendation list* saat melakukan pencarian kata. Pada halaman ini tersedia *bookmark icon button* yang memungkinkan *user* untuk menyimpan atau memberi tanda suatu kata, dimana nantinya *user* dapat melihat kata yang disimpam atau diberi tanda tersebut di halaman *Bookmark Screen*. *Wireframe* dari halaman *Detail Screen* dapat dilihat pada Gambar 3.8.

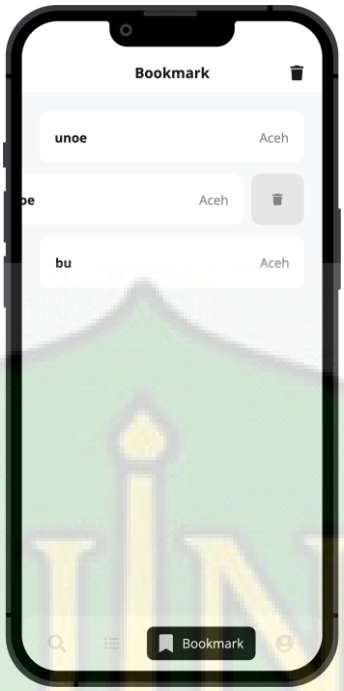


Gambar 3.8 *Detail Screen – Wireframe*

Sumber: Data diolah (2023)

f. *Bookmark Screen*

Bookmark Screen merupakan halaman yang menampilkan daftar kata yang telah diberi tanda oleh pengguna, untuk kemudian bisa diakses lebih cepat tanpa perlu lagi melakukan proses pencarian. Pada Gambar 3.9 dapat dilihat *wireframe* dari halaman *Bookmark Screen*.

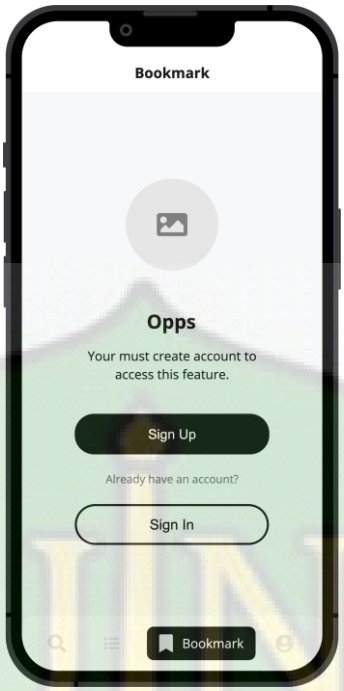


Gambar 3.9 *Bookmark Screen – Wireframe*

Sumber: Data diolah (2023)

g. *Bookmark Screen (State while no loggedin yet)*

Tampilan ini muncul ketika pengguna mengakses halaman *Bookmark Screen* dengan menekan tombol *icon bookmark* yang terdapat di *bottom navigation bar,* namun pengguna belum melakukan proses *login* dengan akun yang sudah terdaftar. Pada halaman ini tersedia tombol untuk melakukan *sign up* dan *sign in,* untuk memungkinkan pengguna mendaftar dan masuk ke aplikasi dengan akun yang sudah terdaftar. Pada Gambar 3.10 dapat dilihat tampilan *wireframe* dari halaman *Bookmark Screen* dengan situasi dimana pengguna belum melakukan *login.*

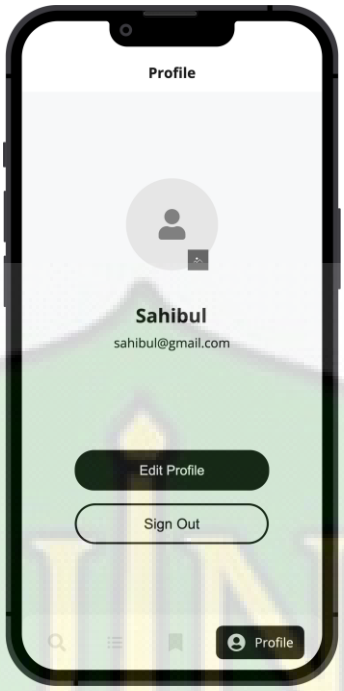


Gambar 3.10 *Bookmark Screen (State while no loggedin yet) – Wireframe*

Sumber: Data diolah (2023)

h. *Profile Screen*

Halaman ini merupakan halaman yang menampilkan informasi tentang pengguna seperti nama, *email,* dan foto profil pengguna. Pada halaman ini juga tersedia tombol untuk melakukan *sign out* dan *edit profile.* Pada Gambar 3.11 dapat dilihat tampilan *wireframe* halaman *Profile Screen*.

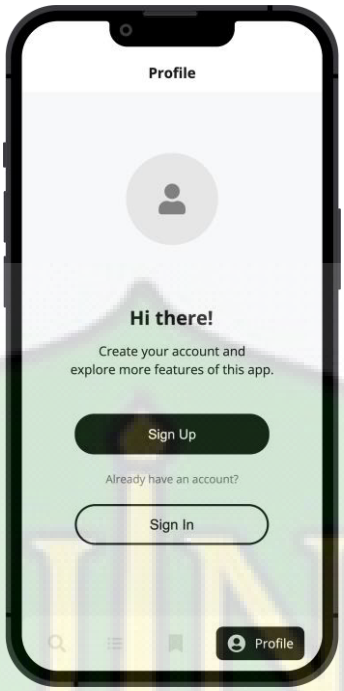


Gambar 3.11 *Profile Screen – Wireframe*

Sumber: Data diolah (2023)

i. *Profile Screen (State while no loggedin yet)*

Ini adalah tampilan halaman *Profile Screen* ketika pengguna belum melakukan *login* dengan akun yang terdaftar. Pada halaman ini disediakan tombol untuk melakukan *sign up* dan tombol *sign in.* Pada Gambar 3.12 dapat dilihat tampilan *wireframe* halaman *Profile Screen* ketika pengguna belum melakukan *login.*

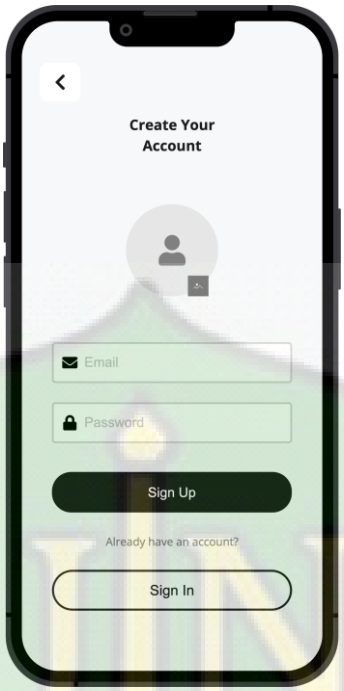


Gambar 3.12 *Profile Screen (State while no loggedin yet) – Wireframe*

Sumber: Data diolah (2023)

j. *Sign Up Screen*

Sign Up Screen adalah halaman untuk melakukan pendaftaran pada aplikasi. Di halaman ini tersedia *form* pendaftaran seperti *email, password* dan fitur unggah foto profil. Pada Gambar 3.13 dapat dilihat *wireframe* halaman *Sign Up Screen.*

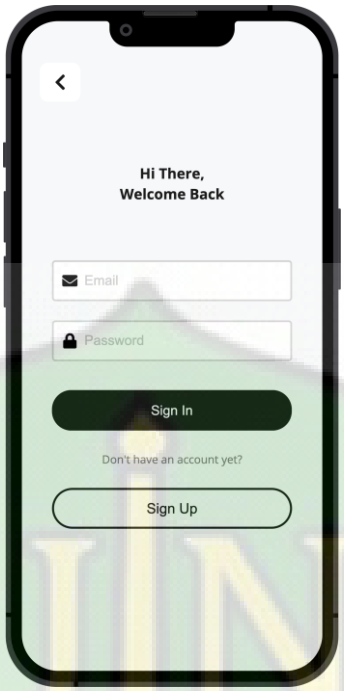


Gambar 3.13 *Sign Up Screen – Wireframe*

Sumber: Data diolah (2023)

k. *Sign In Screen*

Sign In Screen adalah halaman untuk masuk ke aplikasi. Pada halaman ini tersedia *form* yang harus diisi oleh pengguna seperti *email*, dan *password*. Pada Gambar 3.14 dapat dilihat *wireframe* halaman *Sign In Screen.*

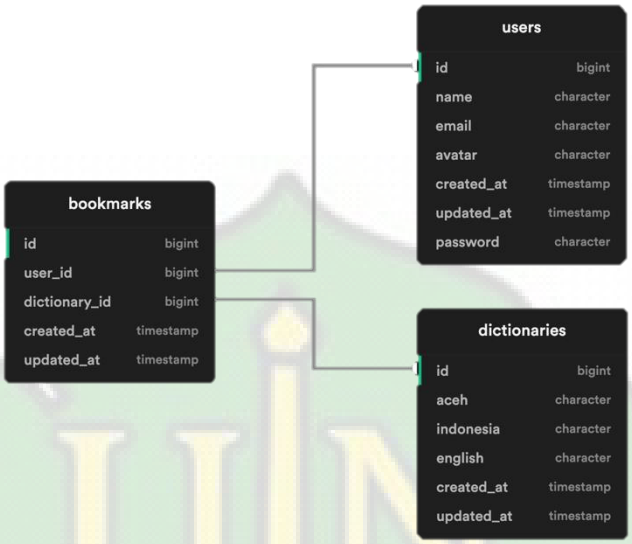


Gambar 3.14 *Sign In Screen – Wireframe*

Sumber: Data diolah (2023)

3. Database Schema

Database Schema merupakan *blueprint* yang dapat membantu pengembang memvisualisasikan bagaimana data diatur dalam *database* relasional, seperti nama tabel, *field*, tipe data dan hubungan antar entitas. Namun, *database schema* tidak berisi data (ibm.com, 2021). Pada Gambar 3.15 dapat dilihat *database schema* untuk aplikasi kamus bahasa Aceh.



Gambar 3.15 *Database Schema*

Sumber: Data diolah (2023)

Dari hasil perancangan *database schema* didapat tiga tabel, yaitu tabel *users*, *dictionaries* dan *bookmarks* dengan rincian sebagai berikut:

Tabel 3.6 *Table Users*

Field Name	***Data Type***	***Description***
id	*bigint*	*Primary Key*
name	*character*	
email	*character*	
password	*character*	
created_at	*timestamp*	

Tabel 3.6 *Table Users*

Field Name	***Data Type***	***Description***
updated_at	*timestamp*	

Sumber: Data diolah (2023)

Tabel 3.7 *Table Dictionaries*

Field Name	***Data Type***	***Description***
id	*bigint*	*Primary Key*
aceh	*character*	
indonesia	*character*	
english	*character*	
created_at	*timestamp*	
updated_at	*timestamp*	

Sumber: Data diolah (2023)

Tabel 3.8 *Table Bookmarks*

Field Name	***Data Type***	***Description***
id	*bigint*	*Primary Key*
user_id	*character*	*Foreign Key*
dictionary_id	*character*	*Foreign Key*
created_at	*timestamp*	
updated_at	*timestamp*	

Sumber: Data diolah (2023)

3.3.5. *Implementation*

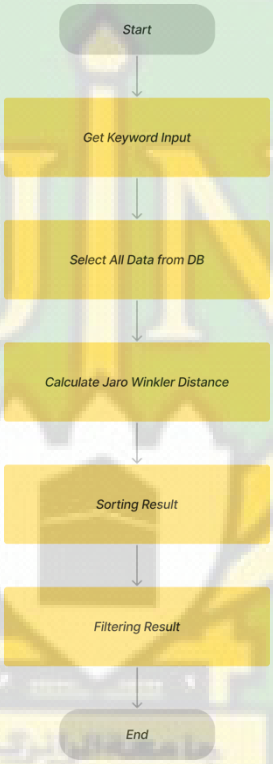
Tahap ini merupakan fase dimana kode pemrograman dibuat. Penulis mengimplementasikan antarmuka pengguna yang telah dirancang pada tahap *design*. Tahap ini sendiri terbagi menjadi tiga tahap, yaitu *unit testing*, *code generation*, dan *refactor*. Dalam praktik yang dilakukan pada penelitian ini, fase *implementation* dibagi menjadi dua sisi yaitu *backend* dan *frontend mobile* mengikuti prinsip penerapan gaya arsitektur REST (*REpresentational State Transfer*). Masing-masing sisi akan dimulai dengan tahap *unit testing*, dimana dilakukan pembuatan *code test* yang kemudian di uji berdasarkan *acceptance criteria* yang diperoleh bersamaan dengan *user story*. Setelah proses pengujian unit berhasil, selanjutnya akan dilakukan tahap *code generation* dimana kode program yang sebenarnya dibuat. Setelah kode program dibuat, selanjutnya adalah *code refactor* jika terdapat kode yang berantakan atau sulit untuk dibaca.

- **Fitur *Recommendation List***

  Metode rekomendasi atau algortima yang digunakan untuk fitur *recommendation list* adalah algoritma Jaro-Winkler. Algoritma Jaro- Winkler digunakan karena terbukti lebih akurat dan tepat dalam pencarian *string* untuk mengidentifikasi kesalahan pengetikan seperti yang dijelaskan pada penelitian studi perbandingan terkait algoritma pencarian *string* dalam metode Approximate String Matching untuk identifikasi kesalahan pengetikan teks oleh Rochmawati Y. dkk. (Rochmawati & Kusumaningrum, 2016). Dalam penelitian lain yang dilakukan oleh (Julian Tannga dkk., 2017), menyatakan bahwa algoritma Jaro-Winkler terbukti memiliki tingkat kesamaan atau kemiripan (*similiarity*) lebih tinggi dan waktu proses lebih cepat dibandingkan algoritma Levenshtein Distance.

  Penulis menggunakan algoritma Jaro-Winkler ini dalam bentuk *package* yang telah dibuat fungsinya dan siap untuk diimplementasikan, sehingga penulis dapat berfokus menulis kode program pengembangan aplikasi tanpa harus menghabiskan waktu untuk membuat fungsi algoritma Jaro-Winkler sendiri.

Package yang digunakan adalah *go-jaro-winkler-distance* yang ditulis dalam bahasa pemrograman Golang oleh Juuso Haavisto dan dapat diakses pada *repository* akun Github miliknya (Haavisto, 2022). *Package go-jaro-winkler-distance* ini diimplementasikan pada sisi *backend*, yang mana nantinya akan digabungkan ke dalam kode program pembuatan API untuk fitur *recommendation list.*



Gambar 3.16 Diagram Alur Algoritma *Recommendation List*

Sumber: Data diolah (2023)

Gambar 3.16 menggambarkan terkait langkah-langkah yang diterapkan dalam pembuatan algoritma *recommendation list* pada sisi *backend* dengan menggunakan algoritma Jaro-Winkler. Adapun rincian yang menjelaskan langkah-langkah yang terdapat pada Gambar 3.16 adalah sebagai berikut.

1. ***Get Keyword Input***

   Pertama adalah mendapatkan input kata kunci pencarian yang dimasukkan pada parameter URL (*Uniform Resource Locator*) ketika melakukan *request* API.

2. ***Select All Data from DB***

   Langkah selanjutnya adalah melakukan SQL (*Standard Query Language*) *query* pada *database* untuk mendapatkan semua data kamus yang tersedia dalam tabel.

3. ***Calculate* Jaro-Winkler Distance**

   Setelah mendapatkan data kamus pada *database*, selanjutnya adalah melakukan kalkulasi atau menghitung kemiripan (*similiarity*) antara inputan kata kunci dengan setiap *item* kata yang tersedia di *database*. Perhitungan *similiariry* dilakukan dengan memanfaatkan *package go-jaro-winkler-distance*.

4. ***Sorting Result***

   Langkah yang dilakukan selanjutnya adalah melakukan pengurutan terhadap hasil perhitungan *similiarity* yang telah didapatkan. Pengurutan dilakukan berdasarkan hasil nilai *similiarity* secara *descending* (besar ke kecil). Jika terdapat nilai *similiarity* sama, maka akan diurutkan berdasarkan alfabet.

5. ***Filtering Result***

   Terakhir adalah melakukan proses *filtering* dari hasil pengurutan yang didapatkan, dimana akan diambil beberapa data saja sesuai kebutuhan untuk di-*return* sebagai hasil dari fitur atau sistem rekomendasi (misal 5 sampai 10 data).

3.3.6. *System Testing*

System testing yang juga disebut dengan *integration testing* merupakan pengujian fungsionalitas kesuluruhan fitur hasil implementasi yang telah terintegrasi. *System testing* termasuk kategori teknik pengujian *black box. System testing* bertujuan untuk menguji semua fitur yang telah diimplementasikan pada suatu iterasi dari sudut pandang pengguna secara *end-to-end.* Artinya pengujian dilakukan secara menyeluruh dengan menjalankan aplikasi pada lingkungan yang sesungguhnya (perangkat nyata atau *emulator*), dimana pengujian berfokus pada seluruh aliran aplikasi tanpa perlu memperhatikan bagaimana kode program bekerja. Hal ini mencakup bagaimana sistem atau aplikasi bereaksi terhadap *input* yang diberikan, apakah *output* yang dihasilkan sesuai dengan yang diharapkan, dan apakah sistem atau aplikasi tersebut dapat berfungsi dengan baik di lingkungan yang sesungguhnya.

3.3.7. *Retrospective*

Tahap terakhir dari proses iterasi adalah *retrospective*. Pengembang menganalisis jalannya setiap pengerjaan *user story*, kesesuaian estimasi waktu pengerjaan, penyebab terjadinya keterlambatan ketika proses pengembangan dan mencegah hal tersebut terulang kembali di iterasi selanjutnya.

3.4. Sumber Data

Sumber data yang digunakan pada penelitian ini adalah Kamus Bahasa Aceh Indonesia Inggris disusun oleh Drs. Abdullah Fardan dan Drs. Zulkarnaini yang diterbitkan oleh CV Boebon Jaya (Fardan & Zulkarnaini, 2019). Total jumlah data yang didapatkan dan dimasukkan ke *database* dari kamus tersebut adalah sebanyak 6220 data kosakata bahasa Aceh yang terdiri dari dua terjemahan yaitu, bahasa Indonesia dan bahasa Inggris.

3.5. Evaluasi Algoritma Fitur *Recommendation List*

Algoritma fitur *recommendation list* yang dibuat pada aplikasi ini dievaluasi untuk mengetahui akurasi atau performanya dalam memberikan rekomendasi kata.

Dalam prosesnya, terlebih dahulu ditentukan apakah kata yang direkomendasikan oleh sistem dari hasil kalkulasi algoritma Jaro-Winkler relevan atau tidak. Relevansi tersebut diukur dengan memberikan nilai ambang batas (*threshold*) tertentu berdasarkan nilai *similiarity* dari kalkulasi algoritma Jaro-Winkler. Kata yang dianggap relevan akan diberi label 1 dan yang tidak relevan diberi label 0.

Proses evaluasi dilakukan dengan menggunakan metrik *Average Precision* (AP) dan *Mean Average Precision* (MAP). AP akan mengukur seberapa baik sistem dalam memberikan rekomendasi kata yang relevan untuk setiap kata kunci, sementara MAP akan mengukur seberapa baik sistem dalam memberikan rekomendasi kata yang relevan untuk seluruh kata kunci secara keseluruhan. Selain itu, evaluasi juga akan dilakukan terhadap algoritma Levenshtein Distance (sebagai pembanding) dengan metode evaluasi yang sama, yaitu menggunakan metrik AP dan MAP.

AP dihitung dengan menjumlahkan item relevan yang ditemukan (R) dan menghitung presisi di setiap urutan ($Precision@i$) pada saat item relevan ditemukan. Kemudian, mengalikan jumlah presisi dengan ($1/R$). Secara matematis, *Average Precision* (AP) dihitung dengan persamaan (3.1).

$$AP = (1/R) \times \sum_{i=1}^{n} Precision@i \tag{3.1}$$

Dimana:

- R adalah jumlah kata atau item relevan yang ditemukan.
- $Precision@i$ adalah presisi pada urutan ke-i.

Untuk mencari dan mendapatkan nilai dari $Precision@i$ pada urutan ke-i digunakan persamaan (3.2).

$$Precision@i = Rn/n \tag{3.2}$$

Dimana:

- Rn adalah jumlah kata atau item relevan yang ditemukan sistem rekomendasi pada urutan ke-i.
- n adalah jumlah item yang dikeluarkan oleh sistem rekomendasi pada urutan ke-i.

Sedangkan, rumus untuk menghitung nilai *Mean Average Precision* (MAP) adalah menggunakan persamaan (3.3).

$$MAP = (1/N) \times \sum_{i=1}^{n} (AP(i)) \tag{3.3}$$

Dimana:

- N adalah jumlah kata kunci yang dicari.
- $AP(i)$ adalah nilai *Average Precision* untuk setiap kata kunci i.

BAB IV
HASIL DAN PEMBAHASAN

4.1. *Design*

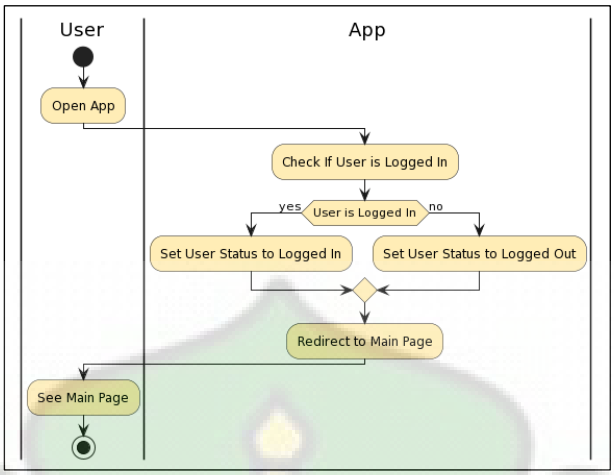
Dalam subbab ini, akan diuraikan terkait desain ataupun pemodelan dari pengembangan aplikasi kamus bahasa Aceh berbasis *mobile*. Dalam hal ini, tujuan utama dari desain adalah untuk memastikan bahwa aplikasi yang dikembangkan memenuhi *requirement* yang telah dirincikan dalam bentuk *user story* pada bab sebelumnya seperti yang terlihat pada Tabel 3.2.

4.1.1. *Activity Diagram*

Activity Diagram digunakan untuk menggambarkan proses dan urutan aktifitas yang terjadi saat pengguna berinteraksi dengan aplikasi. Berikut dapat dilihat *activity diagram* untuk fitur-fitur yang dikembangkan atau dirancang pada aplikasi kamus bahasa Aceh berbasis *mobile*.

1. *Splace Screen*

Pada Gambar 4.1 dapat dilihat urutan proses yang terjadi saat aplikasi pertama kali dibuka oleh pengguna dan aplikasi menampilkan halaman *Splace Screen*. Dalam proses ini, saat pengguna membuka aplikasi maka kemudian aplikasi akan melakukan pengecekan dan validasi status pengguna apakah telah melakukan *login* sebelumnya atau tidak. Selanjutnya pengguna diarahkan ke halaman utama aplikasi.



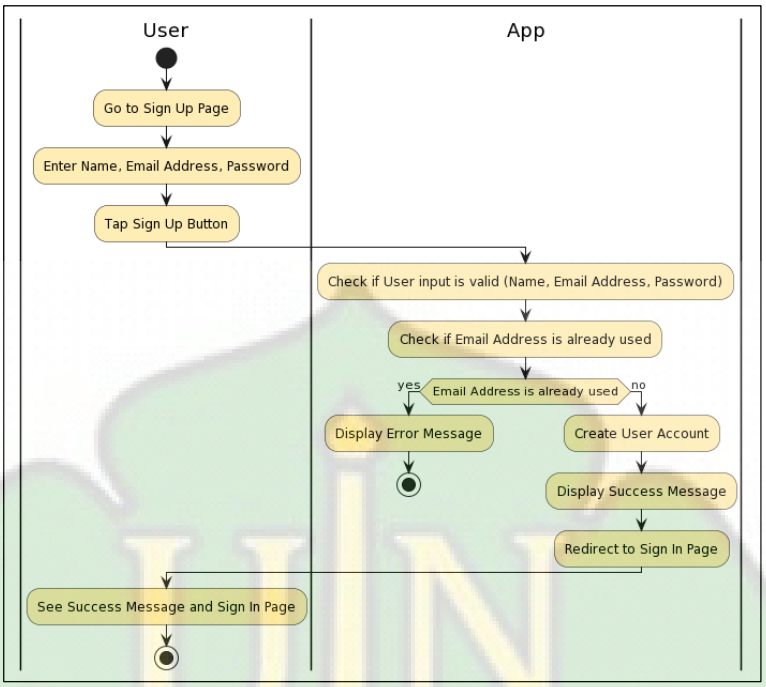
Gambar 4.1 *Activity Diagram Splace Screen*

Sumber: Data diolah (2023)

2. *Sign Up/Create Account*

Gambar 4.2 menggambarkan urutan proses pendaftaran akun pengguna di aplikasi. Adapun rinciannya yaitu:

- Pertama, pengguna masuk ke halaman *Sign Up* aplikasi dan melakukan proses pembuatan akun dengan mengisi data *name, email* dan *password.* Selanjutnya menekan tombol *Sign Up*.
- Kemudian, aplikasi memproses pembuatan akun pengguna berdasarkan data yang telah dimasukkan. Setelah itu, aplikasi memberikan respon hasil pembuatan akun dengan menampilkan notifikasi berhasil atau gagal membuat akun kepada pengguna.
- Jika proses pembuatan akun berhasil, maka pengguna diarahkan ke halaman *Sign In* untuk melakukan proses *login* dengan akun yang telah terdaftar.



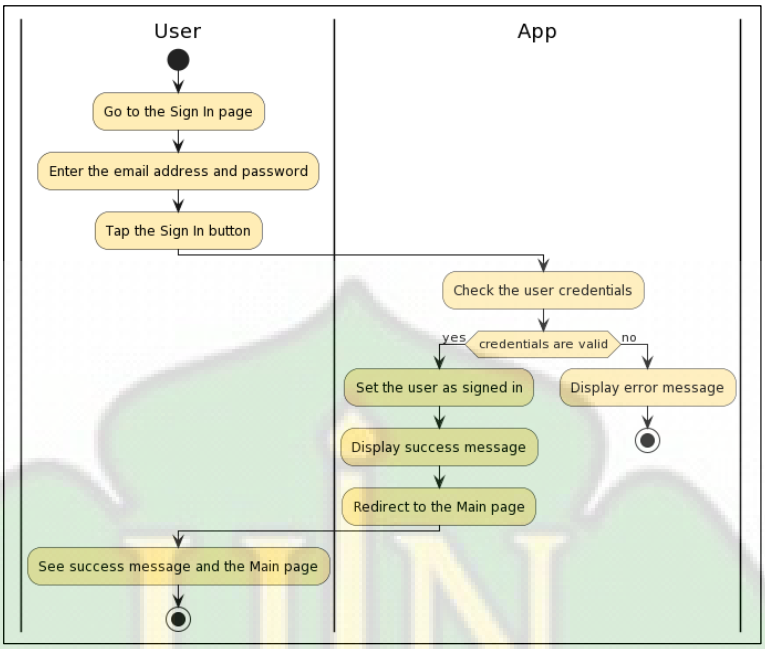
Gambar 4.2 *Activity Diagram Sign Up*

Sumber: Data diolah (2023)

3. *Sign In*

Gambar 4.3 menggambarkan rangkaian proses *login* yang dilakukan pengguna di aplikasi kamus bahasa Aceh berbasis *mobile*. Adapun rinciannya ialah sebagai berikut:

- Pertama, pengguna masuk ke halaman *Sign In* aplikasi dan melakukan proses *login* dengan mengisi data *email* dan *password* yang terdaftar pada aplikasi. Selanjutnya menekan tombol *Sign In*.
- Kemudian, aplikasi memproses dan memvalidasi *credential* data pengguna yang telah dimasukkan. Setelah itu, aplikasi memberikan respon hasil proses *login* dengan menampilkan notifikasi berhasil atau gagal kepada pengguna.
- Jika berhasil, maka pengguna akan diarahkan ke halaman utama aplikasi dan pengguna bisa akses fitur yang dibatasi ketika tidak melakukan *login*.



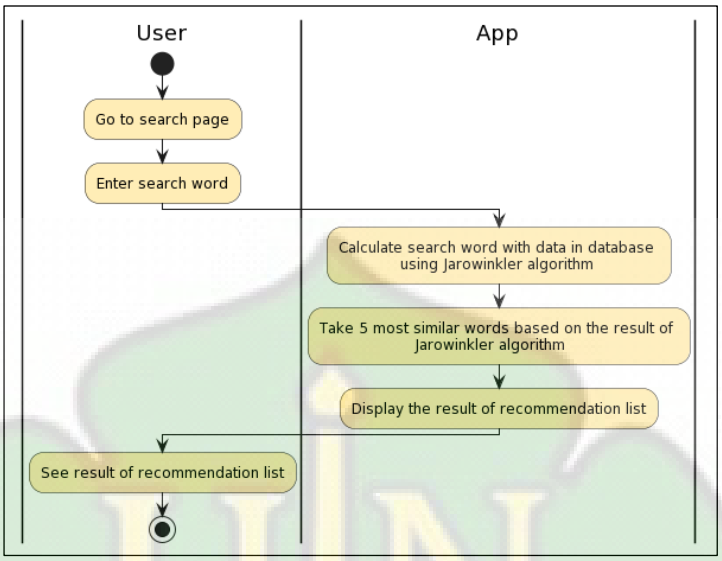
Gambar 4.3 *Activity Diagram Sign In*

Sumber: Data diolah (2023)

4. *Search – Recommendation List*

Gambar 4.4 merupakan gambaran urutan aktifitas untuk melakukan pencarian kata di aplikasi kamus bahasa Aceh berbasis *mobile*. Adapun penjelasan rinciannya adalah sebagai berikut:

- Pertama, masuk ke halaman *Search* dan memasukkan kata kunci bahasa Aceh yang ingin dicari.
- Kemudian, aplikasi akan memproses kata kunci yang dimasukkan pengguna dengan mencari data yang memiliki kemiripan dengan data yang tersedia di database menggunakan algoritma Jaro-Winkler. Proses tersebut dapat dilihat pada Gambar 3.16 yang telah dijelaskan pada subbab *Implementation* di bab 3.
- Selanjutnya diambil maksimal lima data untuk dimasukkan kedalam daftar rekomendasi kepada pengguna. Data yang diambil merupakan data dengan tingkat kemiripan paling tinggi berdasarkan hasil proses atau kalkulasi dari algoritma rekomendasi yang diterapkan menggunakan algoritma Jaro-Winkler.

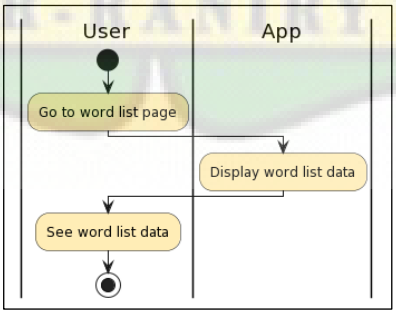


Gambar 4.4 *Activity Diagram Search – Recommendation List*

Sumber: Data diolah (2023)

5. ***Show All Word***

Gambar 4.5 menggambarkan urutan aktifitas untuk melihat daftar kosakata yang tersedia pada halaman *Word List*. Untuk mengakses fitur ini, pengguna cukup melakukan navigasi ke halaman *Word List* di aplikasi kamus bahasa Aceh berbasis *mobile*. Kemudian aplikasi akan menampilkan daftar kosakata yang tersedia di *database* pada halaman tersebut.

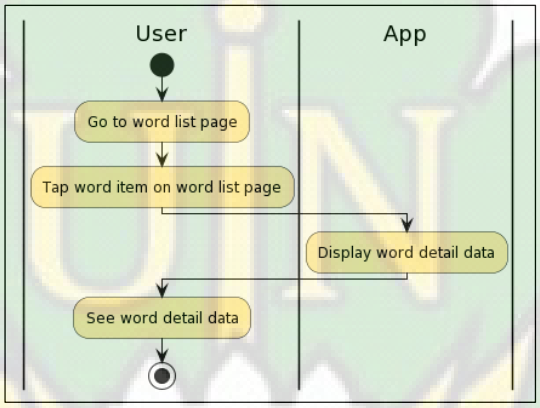


Gambar 4.5 *Activity Diagram Show All Word*

Sumber: Data diolah (2023)

6. *Word Detail*

Gambar 4.6 merupakan gambaran urutan aktifitas untuk melihat rincian atau detail dari suatu kosakata yang ditampilkan pada aplikasi. Untuk melakukannya, pengguna masuk ke halaman *Word List* di aplikasi, kemudian tekan salah satu kata yang ditampilkan di halaman *Word List*. Setelah itu tampilan aplikasi akan di navigasi ke halaman *Word Detail*, dimana pada halaman tersebut akan ditampilkan terkait rincian dari suatu kosakata.



Gambar 4.6 *Activity Diagram Word Detail*

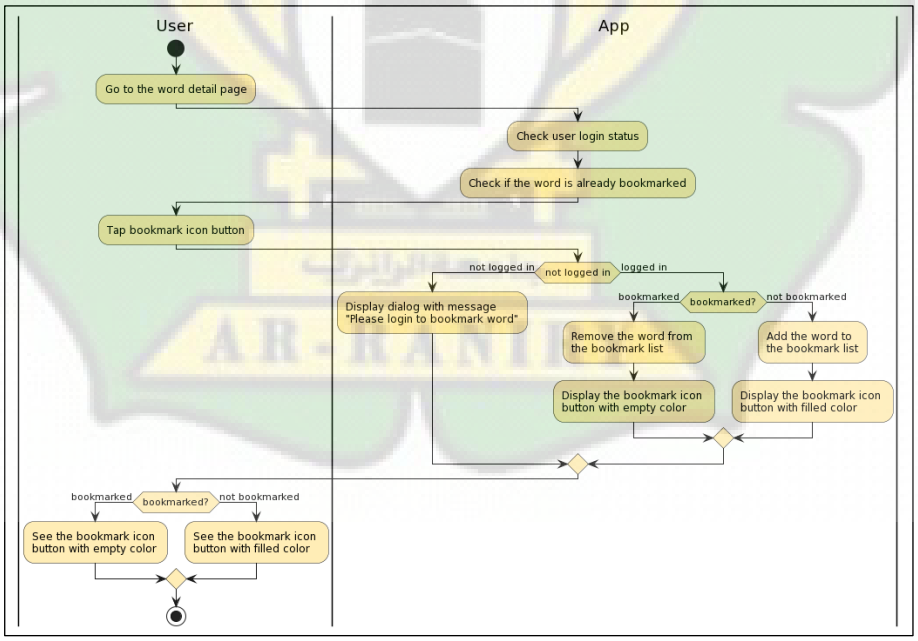
Sumber: Data diolah (2023)

7. *Mark & Unmark Word*

Gambar 4.7 merupakan gambaran urutan aktifitas untuk menandai dan menghapus kosakata yang ditandai. Adapun rincian prosesnya yaitu:

- Pertama, masuk ke halaman *Word Detail* sebagimana petunjuk yang dijelaskan pada Gambar 4.6.
- Selanjutnya, aplikasi akan mengecek status *login* pengguna. Karena fitur *Bookmark* hanya bisa diakses oleh pengguna yang terdaftar pada aplikasi kamus bahasa Aceh berbasis *mobile*.
- Kemudian dilakukan pengecekan apakah kosakata yang ditampilkan pada halaman *Word Detail* telah ditandai atau belum.

- Seteleh itu pengguna menekan tombol ikon *bookmark* untuk menandai kosakata atau menghapus kosakata tersebut dari daftar *bookmark*.
- Ketika pengguna menekan tombol, aplikasi memvalidasi status *login* pengguna untuk kemudian jika pengguna belum *login*, maka akan diarahkan agar melakukan proses *login* terdahulu.
- Jika pengguna sudah *login* dan jika belum menandai kosakata tersebut maka ketika pengguna menekan tombol ikon *bookmark*, kosakata akan dimasukkan kedalam daftar *bookmark* pengguna. Lalu, jika sudah menandai kosakata tersebut maka kosakata akan dihapus dari daftar *bookmark*.
- Kosakata yang telah ditandai dapat diketahui dengan tombol ikon *bookmark* berubah menjadi *full* bewarna dan kosakata tersebut tersedia dalam daftar *bookmark* yang bisa dilihat pada halaman *Bookmark List*. Sedangkan, jika kosakata tidak ditandai maka tombol ikon *bookmark* di halaman *Word Detail* bewarna putih dengan warna *border* hitam dan kosakata tersebut tidak tersedia dalam daftar *bookmark* yang dapat dilihat pada halaman *Bookmark List.*



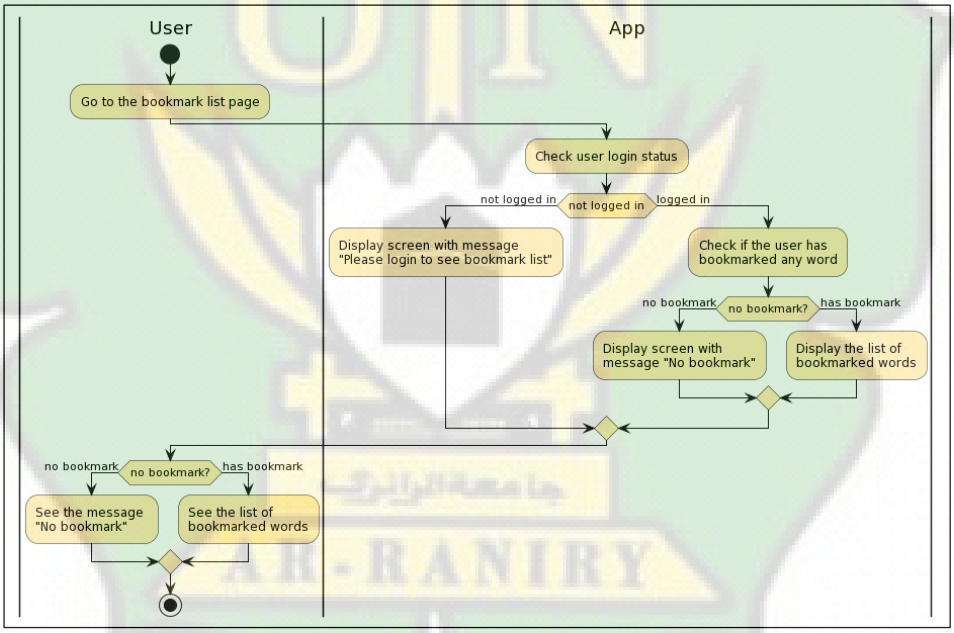
Gambar 4.7 *Activity Diagram Mark & Unmark Word*

Sumber: Data diolah (2023)

8. *Bookmark List*

Gambar 4.8 adalah gambaran urutan proses saat pengguna melihat daftar kosakata yang telah ditandai pada halaman *Bookmark*. Adapun rincian prosesnya yaitu:

- Pengguna masuk ke halaman *Bookmark*
- Kemudian sistem akan melakukan pengecekan status *login* pengguna. Jika pengguna belum *login* maka pada halaman *Bookmark* akan ditampilkan pesan dan tombol *login* agar pengguna melakukan proses *login* terlebih dahulu untuk akses fitur.
- Lalu, jika pengguna sudah *login* maka pengguna dapat melihat daftar kosakata yang telah ditandai pada halaman *Bookmark*.



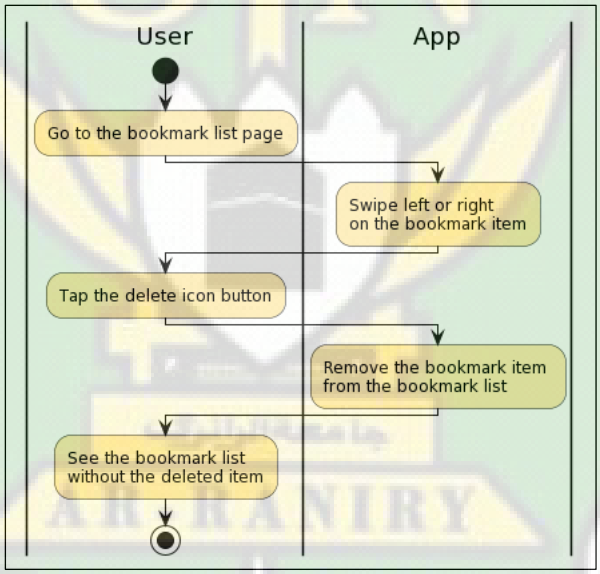
Gambar 4.8 *Activity Diagram Bookmark List*

Sumber: Data diolah (2023)

9. *Remove Bookmark Item*

Gambar 4.9 merupakan gambaran urutan proses untuk menghapus kosakata dari daftar *bookmark* yang ditampilkan pada halaman *Bookmark*. Adapun rincian prosesnya yaitu:

- Pertama masuk ke halaman *Bookmark*
- Kemudian dari geser kesamping kiri atau kanan salah satu *item* kosakata. Lalu akan terlihat tombol berwarna merah dengan ikon *trash*.
- Selanjutnya, pengguna menekan *tombol* berwarna merah tersebut untuk menghapus kosakata dari daftar *bookmark*.
- Aplikasi akan memproses penghapusan kosakata dari daftar *bookmark*. Jika berhasil maka pengguna akan melihat kosakata yang dihapus tersebut tidak tersedia pada daftar *bookmark* yang ditampilkan di halaman *Bookmark*.



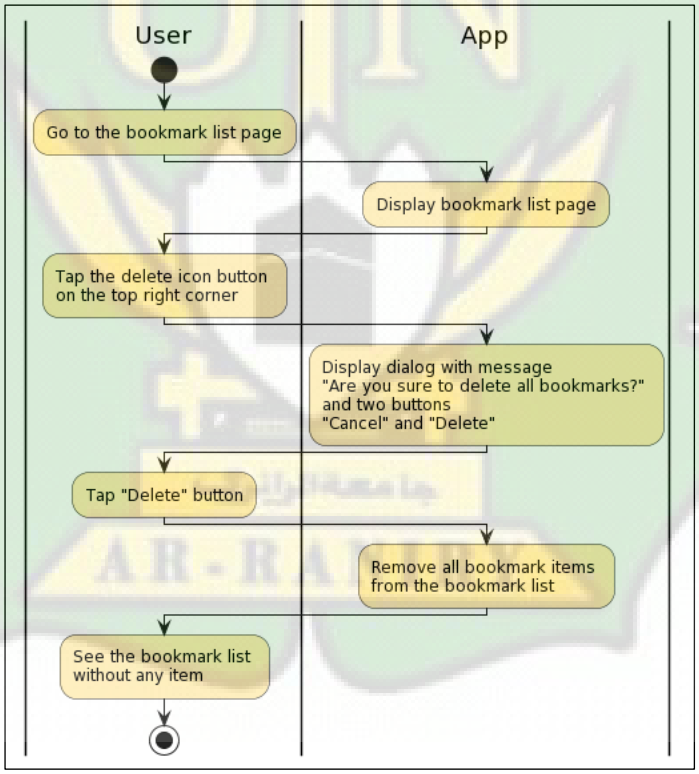
Gambar 4.9 *Activity Diagram Remove Bookmark Item*

Sumber: Data diolah (2023)

10. *Remove All Bookmark*

Gambar 4.10 menggambarkan urutan proses untuk menghapus semua kosakata yang tersedia dalam daftar *bookmark* sekaligus. Adapun rinciannya adalah sebagai berikut:

- Pertama masuk ke halaman *Bookmark*
- Kemudian tekan tombol ikon *trash* yang berada disebelah kanan atas atau disamping kanan pada *AppBar*. Selanjutnya tekan tombol *Delete* pada *dialog* yang muncul untuk mengkonfirmasi penghapusan semua kosakata yang ada dalam daftar *bookmark* sekaligus.
- Jika berhasil, maka akan terlihat illustrasi dan pesan pada halaman *Bookmark* bahwa tidak ada kosakata yang ditandai dalam daftar *bookmark.*



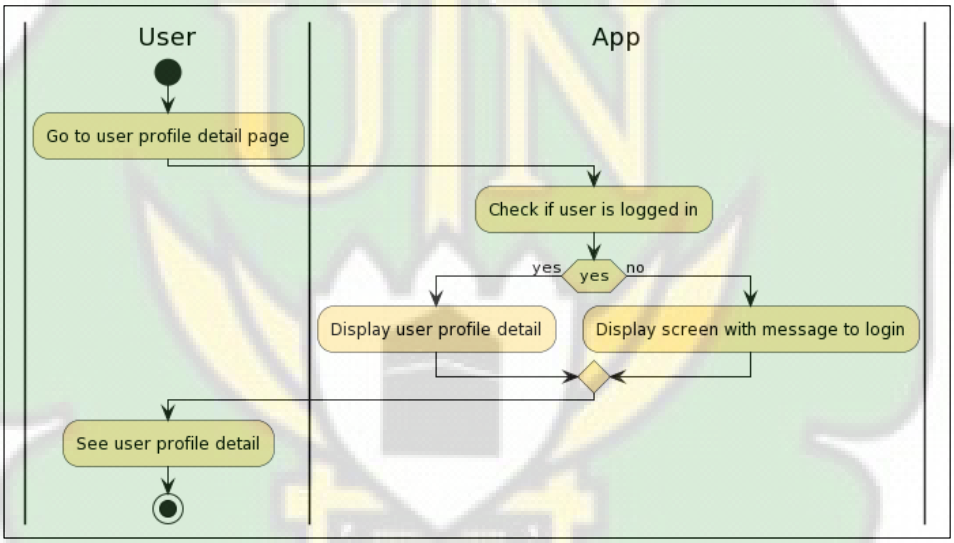
Gambar 4.10 *Activity Diagram Remove All Bookmark*

Sumber: Data diolah (2023)

11. *User Profile*

Gambar 4.11 merupakan gambaran proses untuk menampilkan detail pengguna yang telah *login*. Adapun rinciannya adalah sebagai berikut:

- Pertama, masuk ke halaman *Profile*
- Kemudian, jika pengguna belum melakukan proses *login* maka akan ditampilkan layar dengan pesan untuk mengarahkan pengguna melakukan *login* terlebih dahulu.
- Lalu, jika pengguna telah *login* maka pengguna dapat melihat data mereka yang telah terdaftar pada aplikasi di halaman *Profile*.



Gambar 4.11 *Activity Diagram User Profile*

Sumber: Data diolah (2023)

4.2. *Implementation*

Hasil dari tahap implementasi dituangkan pada poin ini beserta rinciannya. Secara garis besar, proses implementasi dimulai dengan pembuatan API (*Application Programming Interface*) disisi *backend* menggunakan bahasa pemrograman Go dan melakukan pengujian terhadap API yang dibuat menggunakan Postman sebagai salah

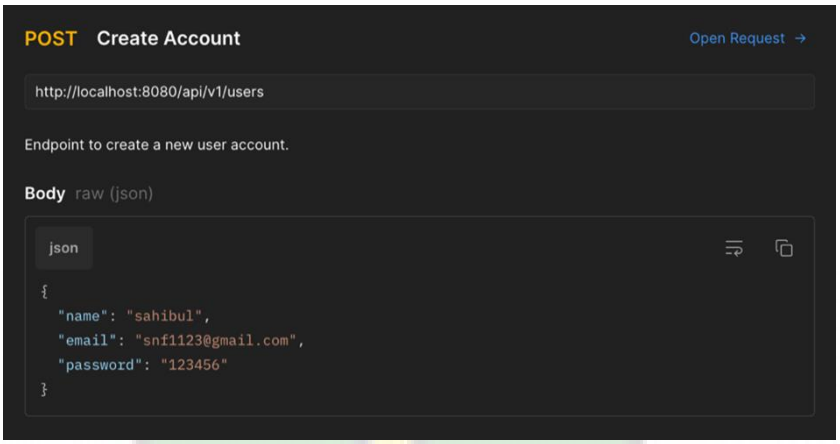
satu alat penguji atau simulasi API. Kemudian dilanjutkan disisi *frontend* dengan pembuatan tampilan aplikasi dan melakukan integrasi API pada aplikasi menggunakan bahasa pemrograman Dart dan Flutter. Adapun hasil implementasi pada pengembangan aplikasi kamus bahasa Aceh berbasis *mobile* dapat dilihat pada subbab berikut beserta rinciannya dalam bentuk tampilan aplikasi dan dokumentasi API. Dokumentasi API berfungsi untuk mengetahui bagaimana API dapat digunakan atau diterapkan pada aplikasi disisi *client*, hal ini mencakup cara mengirim *request* dan mendapatkan *response* yang dikembalikan dari *server*.

4.2.1. Fitur *Authentication*

Adapun hasil API untuk fitur *Authentication* dihasilkan dua *endpoint* yaitu, *endpoint Sign Up/Create Account* dan *endpoint Sign In*. Pada poin-poin berikut dapat dilihat rincian dokumentasinya.

1. **API *Sign Up/Create Account***

   Gambar 4.12 adalah *endpoint* atau titik akhir untuk membuat akun pengguna baru di aplikasi. *Endpoint* ini dapat diakses melalui URL dengan format https://{domain}/api/v1/users. Adapun rincian yang harus diterapkan untuk mengirim *request* melalui *endpoint* ini yaitu dengan menggunakan *method* POST dan menyertakan data *name*, *email*, dan *password* dengan format JSON (*JavaScript Object Notation*) pada bagian *body*.

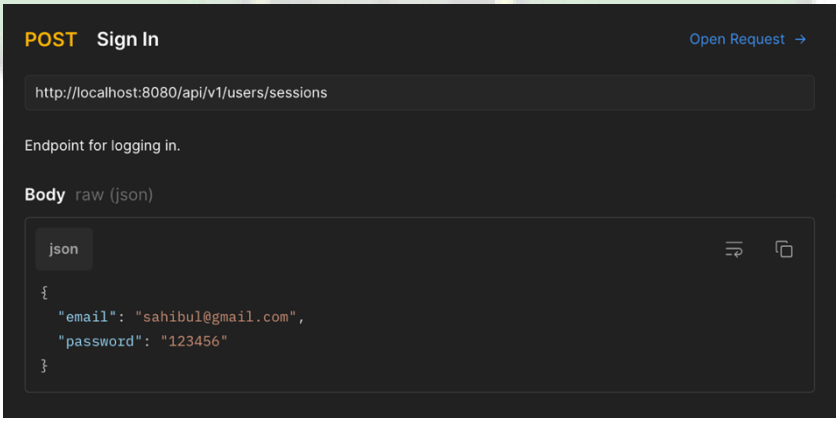


Gambar 4.12 Dokumentasi API *Create Account*

Sumber: Data diolah (2023)

2. API *Sign In*

Gambar 4.13 adalah *endpoint* atau titik akhir untuk melakukan proses *login* di aplikasi. *Endpoint* ini dapat diakses melalui URL dengan format https://{domain}/api/v1/users/sessions. Adapun rincian yang harus diterapkan untuk mengirim *request* melalui *endpoint* ini yaitu dengan menggunakan *method* POST dan menyertakan data *name*, dan *password* dengan format JSON (*JavaScript Object Notation*) pada bagian *body*.



Gambar 4.13 Dokumentasi API *Sign In*

Sumber: Data diolah (2023)

Adapun hasil dari tampilan aplikasi untuk fitur *Authentication* terdapat tiga halaman, diantaranya adalah halaman *Splace Screen*, halaman *Sign Up,* dan halaman *Sign In*. Penjelasan lebih lanjut dikemas pada poin-poin berikut.

1. Halaman *Splace Screen*

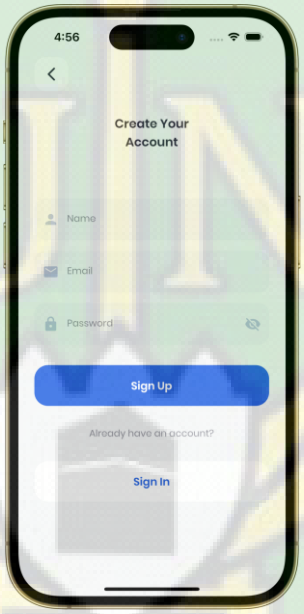
Tampilan pada Gambar 4.14 adalah halaman dari hasil *user story Splace Screen* yang merupakan bagian dari fitur atau modul *Authentication.* Halaman ini merupakan halaman yang pertama kali muncul ketika aplikasi dibuka. Halaman ini berfungsi sebagai halaman yang akan mengecek status pengguna apakah sudah melakukan proses *login* atau tidak. Pengecekan status tersebut berfungsi untuk memberi dan membatasi hak akses terkait fitur yang tersedia dalam aplikasi kamus bahasa Aceh berbasis *mobile.*

Gambar 4.14 Hasil *Splace Screen*

Sumber: Data diolah (2023)

2. **Halaman *Sign Up/Create Account***

Gambar 4.15 merupakan tampilan halaman *Sign Up*. Pada halaman ini pengguna dapat membuat akun untuk terdaftar dan bisa akses fitur yang dibatasi pada aplikasi kamus bahasa Aceh berbasis *mobile*. Halaman ini adalah hasil dari pengerjaan *user story Sign Up & Create an Account* yang merupakan bagian dari fitur atau modul *Authentication*. Untuk membuat akun pengguna diharuskan memasukkan data seperti *name*, *email* dan *password* pada *form* yang tersedia.

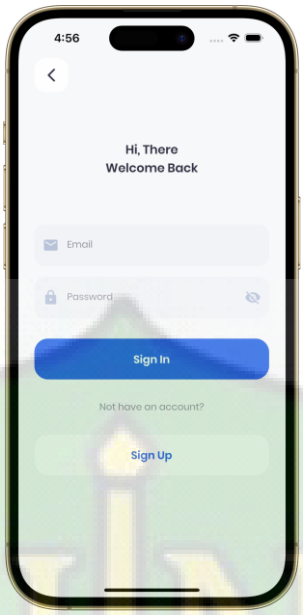


Gambar 4.15 Hasil Halaman *Sign Up*

Sumber: Data diolah (2023)

3. **Halaman *Sign In***

Berikut dapat dilihat tampilan halaman *Sign In* pada Gambar 4.16. Halaman ini tentunya berfungsi untuk proses *login* pengguna pada aplikasi. Pada halaman ini, tersedia dua data yang harus diisi oleh pengguna yaitu *email*, dan *password* sebagai *credential* untuk kebutuhan sistem mengotentikasi pengguna yang melakukan proses *login* pada aplikasi.

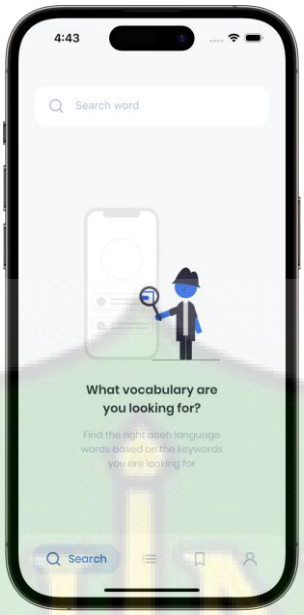


Gambar 4.16 Hasil Halaman *Sign In*

Sumber: Data diolah (2023)

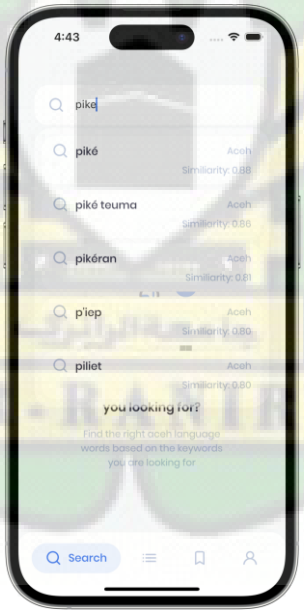
4.2.2. Fitur *Search – Recommendation List*

Tampilan pada Gambar 4.17 dan Gambar 4.18 adalah halaman dari hasil *user story Recommendation List* yang merupakan bagian dari fitur atau modul *Search*. Pada halaman ini memungkinkan pengguna untuk melakukan pencarian kata bahasa Aceh dan aplikasi akan mengembalikan hasil pencarian dalam bentuk *recommendation list* kata. Jumlah kata yang dikembalikan dalam *recommendation list* adalah maksimal sebanyak lima kata. Kelima kata yang menjadi daftar rekomendasi tersebut merupakan kata dengan tingkat kemiripan paling tinggi dengan kata kunci yang dicari berdasarkan perhitungan dari algoritma Jaro-Winkler.



Gambar 4.17 Hasil Halaman *Search*

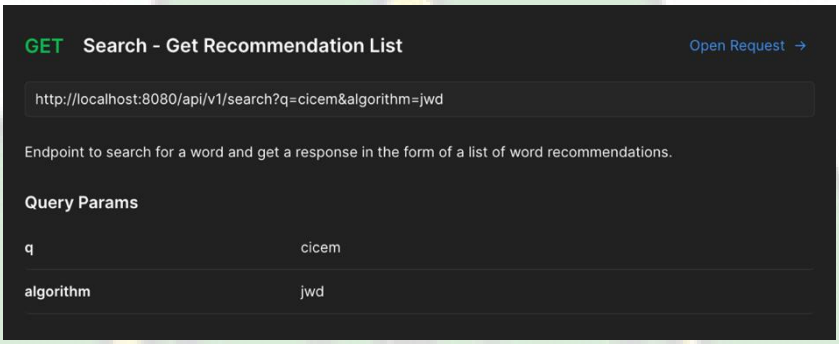
Sumber: Data diolah (2023)



Gambar 4.18 Hasil *Output Recommendation List*

Sumber: Data diolah (2023)

Adapun API endpoint untuk fitur *Search – Recommendation List* dapat dilihat dokumentasinya pada Gambar 4.19 dibawah. *Endpoint* fitur pencarian dapat diakses melalui URL (*Uniform Resource Locator*) dengan format berikut https://{domain}/api/v1/search?q={keyword}&algorithm=jwd. *Endpoint* ini harus diakses menggunakan *method* GET serta menyertakan dua parameter yaitu (*q*) untuk memasukkan kata kunci yang dicari dan (*algorithm*) dengan nilai "*jwd"* yang menyatakan penggunaan algoritma Jaro-Winkler saat memproses pencarian.



Gambar 4.19 Dokumentasi API *Search*

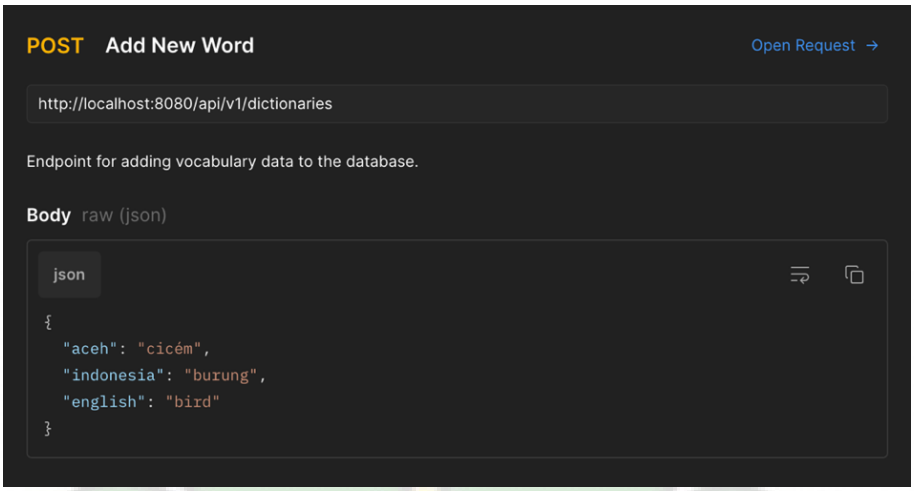
Sumber: Data diolah (2023)

4.2.3. Fitur *Dictionary*

Adapun *endpoint* API untuk fitur *Dictionary* dapat dilihat dokumentasinya sebagai berikut.

1. API *Add New Word*

Gambar 4.20 merupakan *endpoint* atau titik akhir untuk menambahkan data kosakata ke *database*. *Endpoint* ini dapat diakses melalui URL dengan format berikut https://{domain}/api/v1/dictionaries. *Endpoint* harus melakukan *request* dengan menggunakan *method* POST dan menyertakan data kosakata bahasa *aceh*, *indonesia* dan *english* pada bagian *body* dengan format JSON.

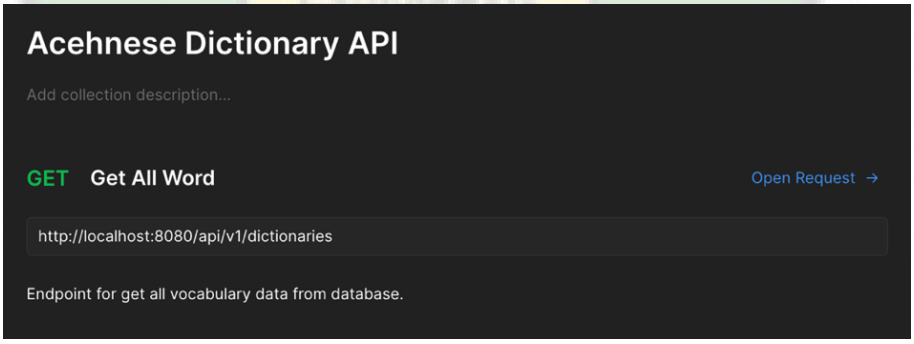


Gambar 4.20 Dokumentasi API *Add New Word*

Sumber: Data diolah (2023)

2. API *Get All Words*

Endpoint ini berfungsi untuk mendapatkan semua data kosakata yang tersedia di *database*. *Endpoint* dapat diakses melalui URL dengan format yang sama dengan *endpoint* untuk menambah kosakata baru seperti yang terlihat pada Gambar 4.20. Namun yang membedakannya adalah *endpoint* ini menggunakan *method* GET saat melakukan *request*, dan tanpa perlu menyertakan data apapun pada bagian *body*. Pada Gambar 4.21 dapat dilihat rinciannya.

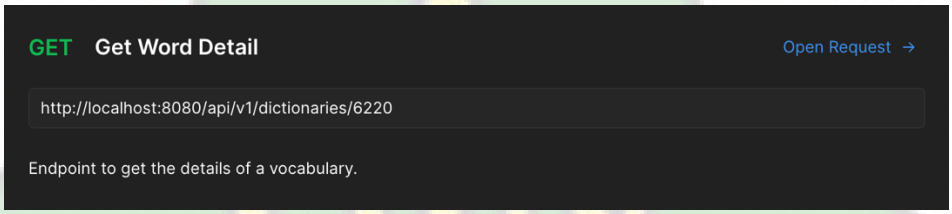


Gambar 4.21 Dokumentasi API *Get All Words*

Sumber: Data diolah (2023)

3. API *Get Word Details*

Gambar 4.22 adalah endpoint yang berfungsi untuk mendapatkan detail dari suatu kosakata. *Endpoint* ini dapat diakses dengan skema *request* yang sama seperti API *endpoint Get All Words*. Namun, pada *endpoint* ini diakhir URL ditambahkan (*id*) yang menyatakan sebagai indeks data yang akan diakses dari *database.* Sebagai contoh pada gambar tersebut yang berarti menyatakan akan mengakses atau mengambil data kosakata dengan (*id* = 6220) dari *database*.



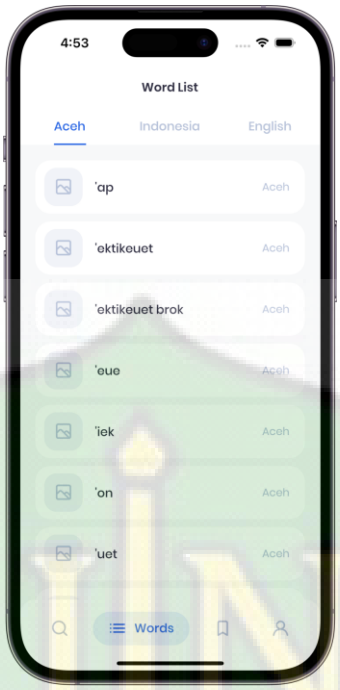
Gambar 4.22 Dokumentasi API *Get Word Details*

Sumber: Data diolah (2023)

Adapun hasil dari tampilan aplikasi kamus bahasa Aceh berbasis *mobile* untuk fitur *Dictionary* adalah sebagai berikut.

1. Tampilan Halaman *Word List*

Tampilan pada Gambar 4.23 adalah halaman *Word List* dari hasil *user story Show All Word* yang merupakan bagian dari fitur atau modul *Dictionary*. Halaman ini menampilkan daftar kosakata yang tersedia di *database* yang terdiri dari tiga jenis bahasa yaitu bahasa Aceh, Indonesia, dan Inggris.



Gambar 4.23 Hasil Halaman *Word List*

Sumber: Data diolah (2023)

2. Tampilan Halaman *Word Detail*

Gambar 4.24 adalah halaman yang menampilkan detail suatu kata dari data yang tersedia. Halaman tersebut merupakan hasil dari pengerjaan *user story Word Detail*. Pada halaman ini, detail kata yang ditampilkan yaitu berupa terjemahan kata, contoh gambar dan tombol untuk menyimpan kata tersebut ke dalam daftar kata yang ditandai (*bookmark*).

Gambar 4.24 Hasil Halaman *Word Detail*

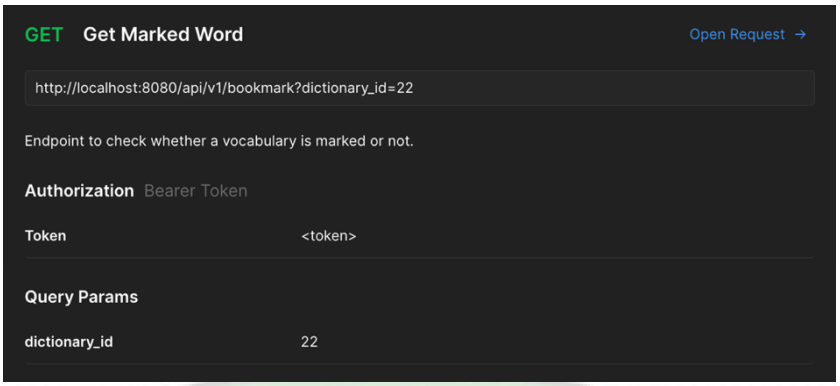
Sumber: Data diolah (2023)

4.2.4. Fitur *Bookmark*

Adapun *endpoint* API untuk fitur *Bookmark* aplikasi kamus bahasa Aceh berbasis *mobile* dapat dilihat dokumentasinya pada poin – poin berikut.

1. API *Get Marked Word*

Gambar 4.25 adalah *endpoint* API yang berfungsi berfungsi untuk mengecek apakah suatu kosakata ditandai atau tidak. *Endpoint* ini dapat diakses menggunakan *method* GET dan melalui URL dengan format berikut yaitu, https://{domain}/api/v1/bookmark?dictionary_id={id}. Selain itu, juga harus menyertakan parameter (*id*) yang mengacu kepada indeks data di table *dictionaries* dan juga token yang didapatkan dari *response* API *Sign In* untuk disematkan pada bagian *Authorization header.*

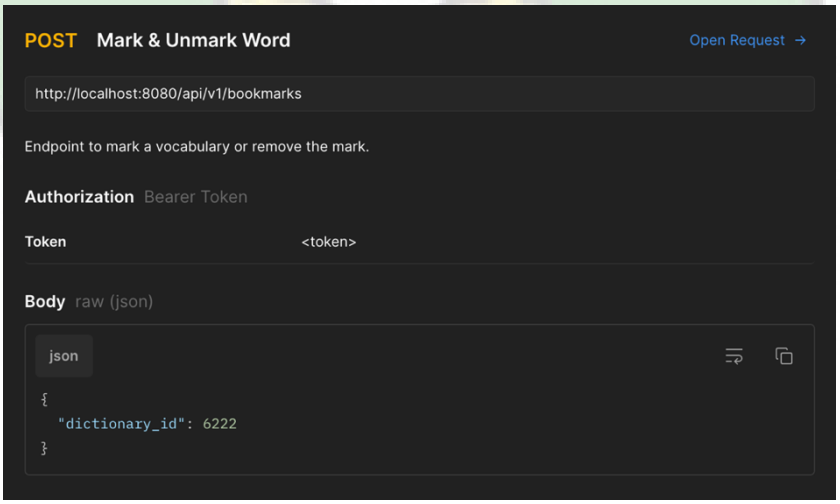


Gambar 4.25 Dokumentasi API *Get Marked Word*

Sumber: Data diolah (2023)

2. API *Mark & Unmark Word*

Gambar 4.26 merupakan dokumentasi *endpoint* API yang berfungsi untuk menandai dan menghapus penanda pada suatu kosakata. *Endpoint* ini dapat diakses dengan format URL https://{domain}/api/v1/bookmarks dan menggunakan *method* POST. Selain itu, juga harus menyertakan *Authorization header* dan data (*dictionary_id*) pada bagian *body request* dengan format JSON yang menyatakan indeks dari data pada tabel *dictionaries* di *database*.

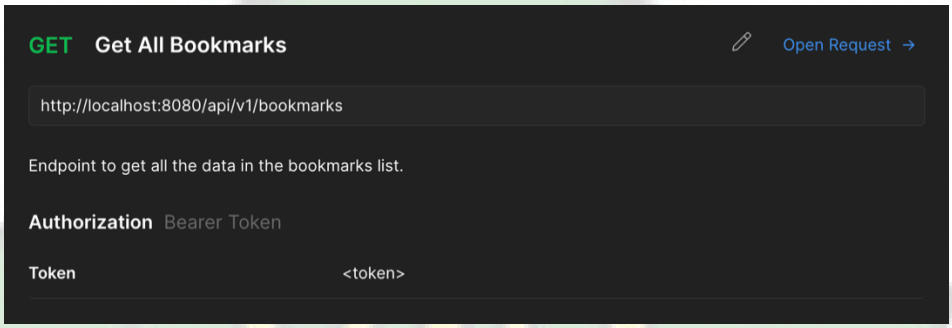


Gambar 4.26 Dokumentasi API *Mark & Unmark Word*

Sumber: Data diolah (2023)

3. API *Get All Bookmarks*

Gambar 4.27 adalah *endpoint* API yang berfungsi untuk mendapatkan daftar kosakata yang diberi penanda. *Endpoint* ini dapat diakses dengan format URL yang sama dengan *endpoint* API *Mark & Unmark Word*. Namun *endpoint* ini diakses menggunakan *method* GET dan perlu menyertakan token pada bagian *Authorization header*.

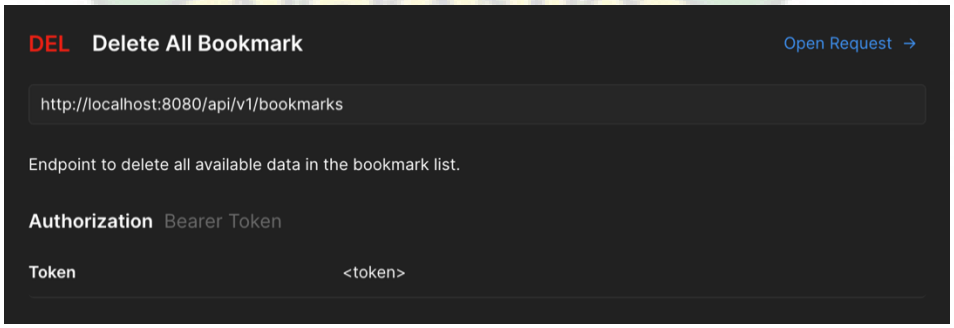


Gambar 4.27 Dokumentasi API *Get All Bookmarks*

Sumber: Data diolah (2023)

4. API *Delete All Bookmarks*

Gambar 4.28 adalah *endpoint* API yang berfungsi untuk menghapus semua kosakata yang telah diberi penanda. *Endpoint* ini dapat diakses dengan skema *request* yang sama dengan *endpoint* API *Get All Bookmarks*. Perbedaannya adalah *endpoint* ini diakses menggunakan *method* DELETE.

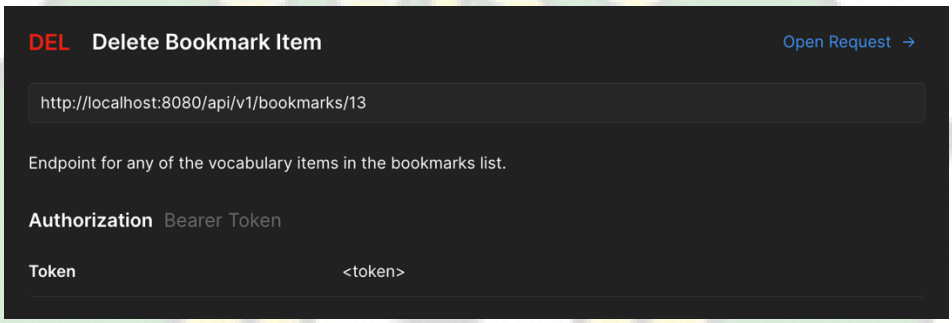


Gambar 4.28 Dokumentasi API *Delete All Bookmarks*

Sumber: Data diolah (2023)

5. API *Delete Bookmark Item*

Gambar 4.29 adalah *endpoint* API yang berfungsi untuk menghapus salah satu kosakata yang telah diberi penanda. *Endpoint* ini diimplementasikan pada halaman *Bookmark List* di aplikasi untuk menghapus salah satu kosakata dari daftar bookmark. *Endpoint* ini dapat diakses dengan skema *request* yang sama dengan *endpoint* API *Delete All Bookmarks*. Perbedaannya adalah terdapat pada akhir URL (*Uniform Resource Locator*) *endpoint* ini ditambahkan (*bookmark_id*) yang menyatakan indeks data pada tabel *bookmarks* di *database*. Sehingga URL menjadi seperti ini, https://{domain}/api/v1/bookmarks/{bookmark_id}.



Gambar 4.29 Dokumentasi API *Delete Bookmark Item*

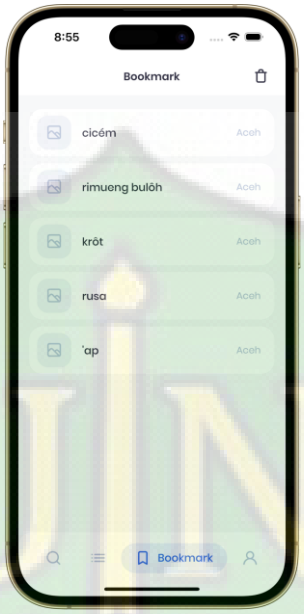
Sumber: Data diolah (2023)

Adapun hasil tampilan aplikasi dari pengerjaan user *stories* untuk kategori fitur *Bookmark* adalah sebagai berikut.

1. Halaman *Bookmark* dengan Data Tersedia

Gambar 4.30 adalah tampilan halaman *Bookmark.* Halaman ini merupakan hasil dari pengerjaan *user story Bookmark List, Remove All Bookmark* dan *Mark & Unmark Word.* Pada halaman ini, pengguna dapat melihat daftar kata yang ditandai ketika menekan tombol *bookmark icon* pada halaman *Word Detail.* Daftar kata yang ditandai ditampilkan dalam bentuk *ListView* pada halaman ini.

Selain itu, pada halaman ini juga disediakan tombol *trash icon* yang berfungsi untuk menghapus semua daftar kata yang ditandai sekaligus.

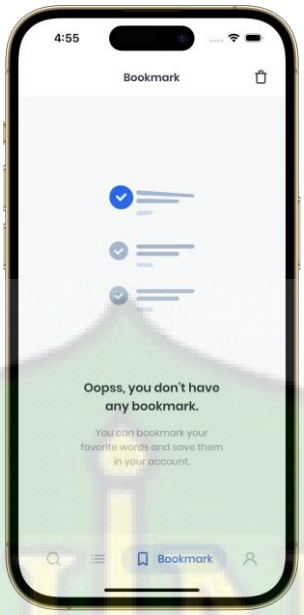


Gambar 4.30 Hasil Halaman *Bookmark List*

Sumber: Data diolah (2023)

2. Halaman *Bookmark* dengan Data Kosong

Gambar 4.31 adalah tampilan halaman *Bookmark* ketika pengguna tidak memiliki kata yang ditandai. Pada kasus tersebut, maka pada halaman *Bookmark* akan menampilkan gambar illustrasi dan pesan teks yang memberi tahu pengguna bahwa daftar *bookmark* masih kosong, karena tidak ada kata yang telah ditandai oleh pengguna.

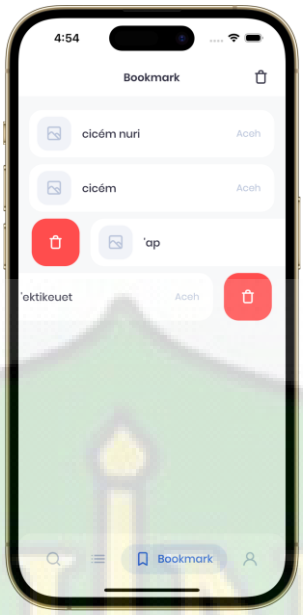


Gambar 4.31 Hasil Halaman *Bookmark* (*empty state*)

Sumber: Data diolah (2023)

3. Tampilan *Remove Bookmark Item*

Gambar 4.32 adalah tampilan halaman *Bookmark* ketika ingin menghapus salah satu dari daftar kata yang ditampilkan. Untuk menghapus salah satu dari daftar kata yang tersedia, pengguna cukup menggeserkan salah satu *item* kesamping kiri ataupun kanan. Setelah itu, maka akan terlihat tombol bewarna merah dengan *trash icon* yang berfungsi untuk menghapus *item* tersebut. Tampilan pada Gambar 4.32 tersebut merupakan hasil dari pengerjaan *user story Remove Bookmark Item*.

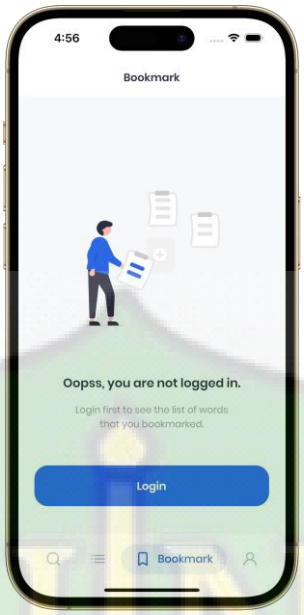


Gambar 4.32 Hasil Tampilan *Remove Bookmark Item*

Sumber: Data diolah (2023)

4. Halaman *Bookmark* saat Pengguna Belum *Login*

Gambar 4.33 adalah tampilan yang muncul ketika status pengguna belum melakukan *login* saat membuka halaman *Bookmark*. Sehingga pengguna tidak memiliki akses untuk menambah, menghapus, dan melihat kata yang ditandai pada halaman ini. Maka pada kasus ini, aplikasi menampilkan gambar illustrasi dan pesan teks yang memberi tahu pengguna bahwa harus melakukan proses *login* terlebih dahulu di aplikasi untuk akses fitur *Bookmark*. Selain itu, terdapat tombol *Login* untuk membantu pengguna melakukan proses *login* di aplikasi.

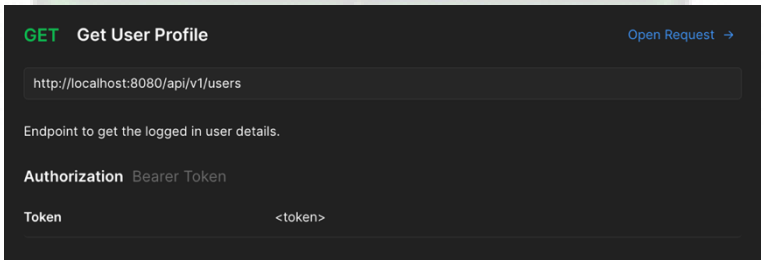


Gambar 4.33 Hasil Halaman *Bookmark* (*not logged in*)

Sumber: Data diolah (2023)

4.2.5. Fitur *User Profile*

Adapun *endpoint* API untuk fitur *User Profile* aplikasi kamus bahasa Aceh berbasis *mobile* dapat dilihat dokumentasinya pada Gambar 4.34 dibawah. *Endpoint* ini dapat diakses melalui URL dengan format https://{domain}/api/v1/users. Lalu, *endpoint* ini harus diakses dengan *method* GET dan menyertakan token pada *Authorization Header* saat melakukan *request* ke server dengan protokol HTTP.



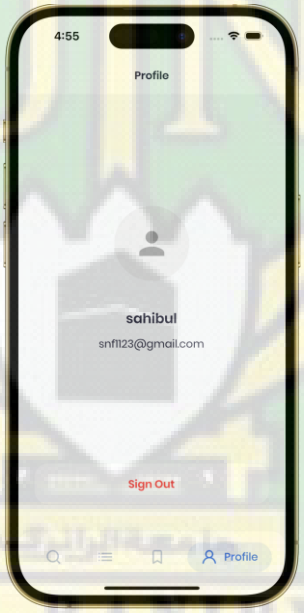
Gambar 4.34 Dokumentasi API *User Profile*

Sumber: Data diolah (2023)

Adapun hasil tampilan aplikasi untuk fitur *User Profile* dapat dilihat pada poin – poin berikut.

1. Halaman *Profile*

Pada halaman ini, data pengguna yang telah melakukan proses *login* ke aplikasi ditampilkan. Data pengguna yang ditampilkan diantaranya yaitu nama, alamat email, dan foto profil. Selain itu, pada halaman ini juga tersedia tombol *Sign Out* yang berfungsi untuk menghapus sesi pengguna yang sedang *login* pada aplikasi. Adapun tampilan dari halaman *Profile* dapat dilihat pada Gambar 4.35.



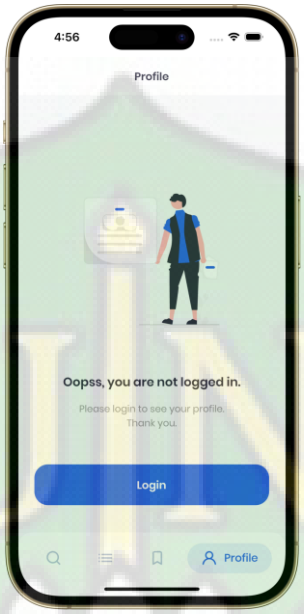
Gambar 4.35 Hasil Halaman *User Profile*

Sumber: Data diolah (2023)

2. Halaman *Profile* saat Pengguna Belum *Login*

Gambar 4.36 merupakan halaman dengan tampilan yang mirip seperti pada halaman *Bookmark* saat pengguna belum *login* ke aplikasi, seperti yang terlihat pada Gambar 4.33. Pada halaman ini juga terdapat gambar illustrasi, pesan

teks, dan tombol *Login*. Tentunya halaman ini muncul saat keadaan yang sama seperti pada halaman *Bookmark,* yaitu muncul ketika status pengguna belum melakukan proses *login* di aplikasi.



Gambar 4.36 Hasil Halaman *Profile* (*not logged in*)

Sumber: Data diolah (2023)

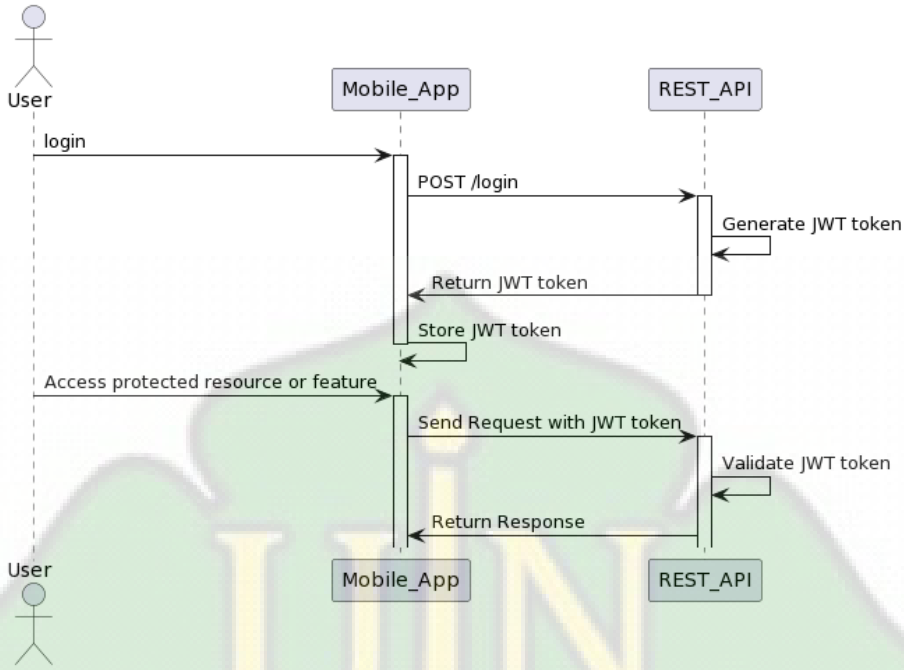
4.3. Keamanan Sistem

Dari segi keamanan sistem, aplikasi kamus bahasa Aceh berbasis *mobile* menerapkan penggunaan JSON *Web Token* (JWT) untuk mengamankan pertukaran informasi antara aplikasi *mobile* dengan REST API atau *web service*. JSON *Web Token* (JWT) merupakan standar terbuka (RFC 7591) yang mendefinisikan cara ringkas dan aman untuk mengirim informasi antar pihak dalam bentuk JSON. JWT dikatakan aman karena informasi yang dikirim dapat diverifikasi dan dipercaya karena ditandatangani secara digital menggunakan algoritma HMAC (*Hash Based Message Authentication Code*) atau pasangan kunci *public*/*privat key* menggunakan RSA atau ECDSA (jwt.io, t.t.).

JSON *Web Token* (JWT) terdiri dari tiga bagian utama, yaitu *Header*, *Payload*, dan *Signature*. *Header* biasanya berfungsi untuk menyimpan dua informasi terkait algoritma yang digunakan untuk membuat tanda tangan (*signature*) seperti HMAC SHA256 atau RSA, dan jenis *token* (JWT) dalam bentuk JSON. *Payload* berfungsi untuk menyimpan data utama dalam *token*, biasanya adalah informasi terkait *user* seperti data *email* atau *id* pengguna dan data lain sesuai kebutuhan untuk dikirim ke penerima dalam format JSON. *Signature* merupakan hasil dari proses enkripsi *header* dan *payload* menggunakan algoritma yang ditentukan dalam *header*. *Signature* ini berguna untuk memverifikasi bahwa data dalam *payload* yang dikirim ke penerima tidak dimodifikasi selama perjalanannya (jwt.io, t.t.).

Adapun mekanisme proses pengamanan sistem menggunakan JWT pada aplikasi kamus bahasa Aceh berbasis *mobile* dapat dilihat pada desain arsitektur Gambar 4.37. Adapun rincian dari gambar tersebut adalah sebagai berikut:

- Pada aplikasi ini, pengamanan sistem dituju untuk membatasi akses *resources* atau fitur tertentu pada aplikasi *mobile* dan REST API, seperti untuk akses *resources* atau fitur *Bookmark*.
- Saat *user* ingin mengakses fitur *Bookmark*, pengguna diharuskan terlebih dahulu untuk melakukan proses *Authentication* atau *login* di aplikasi. Sehingga aplikasi (*client*) mendapatkan *token* (JWT) dari *response* yang dihasilkan oleh REST API (*server*). Kemudian *token* yang didapatkan tersebut disimpan di *local storage*.
- Setelah melakukan proses *login* dan mendapatkan *token* (JWT) yang di-*generate* disisi *backend* dengan REST API. Kemudian *user* bisa mengakses fitur atau *resources* yang dibatasi seperti fitur *Bookmark* dengan adanya *token* (JWT) yang telah didapatkan.
- Secara teknis, aplikasi *mobile* (*client*) akan mengirim *request* ke REST API (*server*) bersamaan dengan *token* (JWT) yang disematkan dibagian *Authorization Header*. Selanjutnya, REST API akan menvalidasi *token* yang dikirim sesuai prosedur apakah *user* atau aplikasi *mobile* (*client*) memiliki akses atau tidak terhadap *resources* yang diakses.



Gambar 4.37 *Sequence Diagram* Arsitektur JWT

Sumber: Data diolah (2023)

4.4. *System Testing*

Setelah pengerjaan kode program yang dilakukan pada tahapan implementasi, maka selanjutnya dilakukan tahapan pengujian sistem (*system testing*). Secara garis besar, pengujian sistem dilakukan dengan dua skenario pengujian yang diantaranya yaitu *user* yang belum melakukan pendaftaran akun dan *user* yang telah memiliki akun terdaftar. Dimana setiap skenario mencakup seluruh pengujian yang menerapkan kasus uji (*test case*) yang tertera pada Tabel 4.1. Pengujian sistem dilakukan sebagaimana yang telah dijabarkan pada bab sebelumnya yaitu, dengan melakukan pengujian dari ujung ke ujung (*end to end*) terhadap sistem yang telah terintegrasi secara menyeluruh. Pengujian sistem diterapkan dengan menggunakan *package* atau *library* bernama *integration_test* yang tersedia pada bahasa pemrograman Dart dan Flutter.

Tabel 4.1 *Test Case*

Feature	***User story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
Authentication	*Sign Up & Create an Account*	Belum punya akun	Membuat akun pengguna	1. Masuk ke halaman *Sign Up* 2. Masukkan *name, email* dan *password* 3. *Tap Sign Up button*	*email* = *snf@gmail.com* *password* = *pass123*	1. *User* mendapatkan pesan notifikasi berhasil buat akun 2. Masuk ke halaman *Sign In*	Berhasil
	Sign In	Sudah punya akun terdaftar	Melakukan *Sign In* pada aplikasi	1. Masuk ke halaman *Sign In* 2. Masukkan *email* dan *password* 3. *Tap Sign In button*	*email* = *snf@gmail.com* *password* = *pass123*	1. *User* mendapatkan pesan notifikasi berhasil *login* 2. *User m*asuk ke halaman utama aplikasi	Berhasil
	Splace Screen		Menampilkan halaman *Splace*	1. Buka aplikasi		1. Logo dan nama aplikasi muncul di halaman *Splace Screen*	Berhasil

Tabel 4.1 *Test Case*

Feature	***User story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
			Screen			2. *User* masuk ke halaman utama aplikasi	
Dictionary	*Show All Word*		Menampilkan daftar kosakata	1. *Tap word list icon button* di *Bottom Navigation Bar*		1. *User* masuk ke halaman *Word List* 2. *User* dapat melihat daftar kosakata kamus	Berhasil
	Word Detail		Menampilkan detail suatu kosakata	1. *Tap* salah satu item dari daftar kosakata kamus di halaman *Word List* atau di hasil pencarian dengan *Recommendation List*		1. *User* masuk ke halaman *Word Detail* 2. *User* dapat melihat detail suatu kosakata	Berhasil
Search	*Recommendation List*		Melakukan	1. *Tap search icon button* di *Bottom*	*keyword = cicem*	1. *User* masuk ke halaman *Search*	Berhasil

Tabel 4.1 *Test Case*

Feature	***User story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
			pencarian dan mendapatkan daftar rekomendasi kata	*Navigation Bar* 2. Masukkan *keyword* di *form* pencarian		2. *User* dapat melihat daftar rekomendasi kata dari hasil pencarian	
Bookmark	*Mark & Unmark Word*	*User* sudah *login*	Menandai suatu kosakata atau menghapusnya	1. Masuk ke halaman *Word Detail* 2. *Tap bookmark icon button*		1. *Bookmark icon button* di halaman *Word Detail* berubah 2. Kata yang diberi tanda muncul atau terhapus dalam daftar penanda pada halaman *Bookmark*	Berhasil
		User belum *login*		1. Masuk ke halaman *Word Detail* 2. *Tap bookmark icon button*		*User* mendapatkan pesan kesalahan, dimana *user* harus *login* untuk akses fitur	Berhasil

Tabel 4.1 *Test Case*

Feature	***User story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
						ini.	
	Bookmark List	*User* sudah *login*	Menampilkan daftar *bookmark*	1. *Tap bookmark icon button* di *Bottom Navigation Bar*		1. *User* masuk ke halaman *Bookmark* 2. *User* dapat melihat daftar kata yang ditandai	Berhasil
		User belum *login*		1. *Tap bookmark icon button* di *Bottom Navigation Bar*		*User* mendapatkan pesan kesalahan, dimana *user* harus *login* untuk akses fitur ini.	Berhasil
	Remove Bookmark Item	*User* sudah *login*	Menghapus salah satu kosakata dari daftar *bookmark*	1. *Tap bookmark icon button* di *Bottom Navigation Bar* 2. Geser ke kiri atau ke kanan salah satu kosakata pada		1. *User* masuk ke halaman *Bookmark* 2. *User* mendapatkan pesan konfirmasi 3. *Bookmark item* terhapus dari	Berhasil

Tabel 4.1 *Test Case*

Feature	***User story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
				daftar *bookmark* 3. *Tap trash_delete icon button* 4. *Tap Delete button*		daftar *bookmark*	
	Remove All Bookmark	*User* sudah *login*	Menghapus semua kosakata yang tersedia dalam daftar *bookmark*	1. *Tap bookmark icon button* di *Bottom Navigation Bar* 2. *Tap trash_delete icon button* di *AppBar* 3. *Tap Delete button*		1. *User* masuk ke halaman *Bookmark* 2. *User* mendapatkan pesan konfirmasi 3. Semua *bookmark item* terhapus dari daftar *bookmark*	Berhasil

Sumber: Data diolah (2023)

4.5. Hasil Evaluasi Algoritma Fitur *Recommendation List*

Pada Tabel 4.2 dapat dilihat daftar 20 yang mewakili 1000 data testing hasil sistem rekomendasi kata menggunakan algoritma Jaro-Winkler dan algoritma Leveinshtein. Tabel tersebut menunjukkan hasil *similiarity* kata antara kata kunci (*input*) dan kata yang diharapkan (*output*) dari kalkulasi kedua algoritma. Selain itu, ditunjukkan juga *priority number* dari kata yang diharapkan muncul pada urutan keberapa dalam daftar rekomendasi. Dalam proses evaluasi, daftar rekomendasi kata yang dikembalikan oleh sistem adalah 10 kata. Dan data yang digunakan untuk diuji adalah data kata bahasa Aceh yang tidak diketik tanda diakritik pada kata kunci. Kemudian, nilai *threshold* yang digunakan dalam proses evaluasi untuk menentukan kata yang dianggap relevan dari daftar rekomendasi kata yang dikembalikan oleh sistem adalah (0,9), (0,8), dan (0,7). Nilai *threshold* tersebut mengacu kepada nilai hasil *similiarity* dari kalkulasi algoritma Jaro-Winkler dan Levenshtein.

Tabel 4.2 Daftar 20 Data Testing Algoritma Jaro-Winkler dan Levenshtein

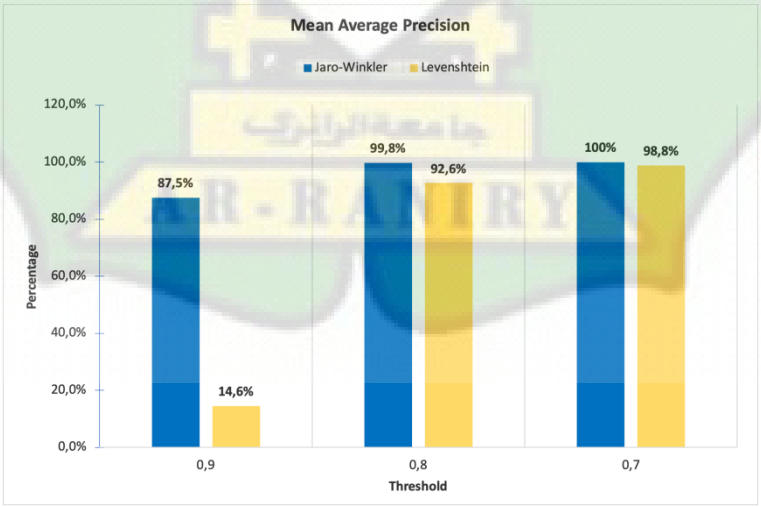
No	*Keyword*	*Expectation*	Jaro-Winkler		Levenshtein	
			Similiarity	*Priority Number*	*Similiarity*	*Priority Number*
1	yok	yôk	0,8	4	0,75	1
2	yoh	yôh	0,8	2	0,75	1
3	weng	wéng	0,85	1	0,8	1
4	weh	wèh	0,8	2	0,75	1
5	wase	wasé	0,88	3	0,8	1
6	wareh	waréh	0,91	3	0,83	1
7	wajek	wajék	0,91	1	0,83	1
8	wajeb	wajéb	0,91	1	0,83	1
9	utom	utôm	0,87	2	0,8	1
10	utoh	utôh	0,87	1	0,8	1
11	usos	usôs	0,87	8	0,8	1
12	use	usé	0,82	4	0,75	1
13	urotan	urôtan	0,91	1	0,86	1
14	urosan	urôsan	0,91	1	0,86	1

Tabel 4.2 Daftar 20 Data Testing Algoritma Jaro-Winkler dan Levenshtein

No	*Keyword*	*Expectation*	Jaro-Winkler		Levenshtein	
			Similiarity	*Priority Number*	*Similiarity*	*Priority Number*
15	urong	urông	0,89	1	0,83	1
16	seukreng	seukréng	0,96	2	0,89	1
17	uree	urèe	0,87	5	0,8	1
18	unyet	unyèt	0,91	1	0,83	1
19	untong	untông	0,92	1	0,86	1
20	unggoi	unggôi	0,93	1	0,86	1

Sumber: Data diolah (2023)

Berdasarkan hasil evaluasi yang telah dilakukan terhadap algoritma fitur *recommendation list* dengan menggunakan 1000 data testing kosakata bahasa Aceh yang tidak diketik tanda diakritik, terlihat bahwa algoritma Jaro-Winkler memberikan hasil yang lebih baik dibandingkan dengan algoritma Levenshtein. Hal ini dapat dilihat pada grafik Gambar 4.38, dimana algoritma Jaro-Winkler menunjukkan nilai *Mean Average Precision* (MAP) yang lebih tinggi di setiap *threshold* yang diuji.



Gambar 4.38 Grafik MAP Evaluasi Algoritma *Recommendation List*

Sumber: Data diolah (2023)

Pada *threshold* (0,9), algoritma Jaro-Winkler memperoleh nilai MAP sebesar 87,5%, sedangkan algoritma Levenshtein hanya memperoleh nilai sebesar 14,6%. Pada *threshold* (0,8), algoritma Jaro-Winkler memperoleh nilai MAP sebesar 99,8%, sedangkan algoritma Levenshtein memperoleh nilai sebesar 92,6%. Dan pada *threshold* (0,7), algoritma Jaro-Winkler memperoleh nilai MAP sebesar 100%, sedangkan algoritma Levenshtein memperoleh nilai sebesar 98,8%.

Perbedaan hasil MAP kedua algoritma yang diuji pada setiap *threshold*, dikarenakan terdapat item relevan yang muncul dalam hasil rekomendasi kata dari satu kali pencarian. Sehingga hal tersebut mempengaruhi hasil perhitungan *Average Precision* (AP) yang dicari dengan persamaan (3.1). Item relevan sendiri diukur dengan menggunakan nilai *threshold* yang telah ditentukan dan membandingkannya dengan nilai *similiarity* dari hasil kalkulasi algoritma Jaro-Winkler dan Levenshtein.

Tabel 4.3, Tabel 4.4, dan Tabel 4.5 menunjukkan contoh hasil rekomendasi dari pencarian kata kunci *yok* dan *weng* yang diuji pada ketiga *threshold*. Pada ketiga tabel tersebut dapat dilihat, dengan kata kunci yang sama terdapat perbedaan kemunculan item relevan dalam daftar rekomendasi yang ditandai dengan kolom berwarna kuning. Pada kasus pengujian dengan *threshold* (0,9), item relevan lebih banyak muncul dalam daftar rekomendasi yang menggunakan algoritma Jaro-Winkler dibandingkan dengan algoritma Levenshtein pada satu kata kunci yang sama dari 1000 data testing yang diuji. Oleh sebab itu, nilai MAP algoritma Jaro-Winkler dan algoritma Levenshtein pada pengujian dengan *threshold* (0,9) memiliki perbedaan yang sangat jauh. Sedangkan pada pengujian dengan *threshold* (0,8) dan (0,7), kemunculan item relevan pada kedua algoritma tidak berbeda jauh atau lebih sering sama-sama muncul item relevan saat diuji dengan kata yang sama dari 1000 data testing.

Tabel 4.3 Contoh Hasil Rekomendasi dengan *Threshold* 0,9

Keyword	Jaro-Winkler				Levenshtein			
	Priority Number - Text	*Similiarity*	*Relevant Items*	*AP*	*Priority Number - Text*	*Similiarity*	*Relevant Items*	*AP*
yok	1 - syok	0,92	2	1	1 - yôk	0,75	0	0
	2 - yo	0,91			2 - syok	0,75		
	3 - yokyok	0,88			3 - yo	0,67		
	4 - yôk	0,8			4 - sok	0,67		
	5 - sok	0,78			5 - rok	0,67		
	6 - rok	0,78			6 - pok	0,67		
	7 - proyèktor	0,78			7 - cok	0,67		
	8 - pok	0,78			8 - yôh	0,5		
	9 - cok	0,78			9 - yokyok	0,5		
	10 - meuyok-yok	0,77			10 - tunyok	0,5		
weng	1 - wéng	0,85	0	0	1 - wéng	0,8	0	0
	2 - beng	0,83			2 - beng	0,75		
	3 - wangi	0,81			3 - wangi	0,6		
	4 - tueng	0,78			4 - tông	0,6		
	5 - rueng	0,78			5 - tueng	0,6		
	6 - mieng	0,78			6 - sèng	0,6		
	7 - lueng	0,78			7 - rueng	0,6		
	8 - bueng	0,78			8 - pèng	0,6		
	9 - bieng	0,78			9 - mieng	0,6		
	10 - aseng	0,78			10 - lông	0,6		

Sumber: Data diolah (2023)

Tabel 4.4 Contoh Hasil Rekomendasi dengan *Threshold* 0,8

Keyword	Jaro-Winkler				Levenshtein			
	Priority Number - Text	*Similiarity*	*Relevant Items*	*AP*	*Priority Number - Text*	*Similiarity*	*Relevant Items*	*AP*
yok	1 - syok	0,92	4	1	1 - yôk	0,75	0	0
	2 - yo	0,91			2 - syok	0,75		
	3 - yokyok	0,88			3 - yo	0,67		
	4 - yôk	0,8			4 - sok	0,67		
	5 - sok	0,78			5 - rok	0,67		
	6 - rok	0,78			6 - pok	0,67		
	7 - proyèktor	0,78			7 - cok	0,67		
	8 - pok	0,78			8 - yôh	0,5		
	9 - cok	0,78			9 - yokyok	0,5		
	10 - meuyok-yok	0,77			10 - tunyok	0,5		
weng	1 - wéng	0,85	3	1	1 - wéng	0,8	1	1
	2 - beng	0,83			2 - beng	0,75		
	3 - wangi	0,81			3 - wangi	0,6		
	4 - tueng	0,78			4 - tông	0,6		
	5 - rueng	0,78			5 - tueng	0,6		
	6 - mieng	0,78			6 - sèng	0,6		
	7 - lueng	0,78			7 - rueng	0,6		
	8 - bueng	0,78			8 - pèng	0,6		
	9 - bieng	0,78			9 - mieng	0,6		
	10 - aseng	0,78			10 - lông	0,6		

Sumber: Data diolah (2023)

Tabel 4.5 Contoh Hasil Rekomendasi dengan *Threshold* 0,7

Keyword	Jaro-Winkler				Levenshtein			
	Priority Number - Text	*Similiarity*	*Relevant Items*	*AP*	*Priority Number - Text*	*Similiarity*	*Relevant Items*	*AP*
yok	1 - syok	0,92	10	1	1 - yôk	0,75	2	1
	2 - yo	0,91			2 - syok	0,75		
	3 - yokyok	0,88			3 - yo	0,67		
	4 - yôk	0,8			4 - sok	0,67		
	5 - sok	0,78			5 - rok	0,67		
	6 - rok	0,78			6 - pok	0,67		
	7 - proyèktor	0,78			7 - cok	0,67		
	8 - pok	0,78			8 - yôh	0,5		
	9 - cok	0,78			9 - yokyok	0,5		
	10 - meuyok-yok	0,77			10 - tunyok	0,5		
weng	1 - wéng	0,85	10	1	1 - wéng	0,8	2	1
	2 - beng	0,83			2 - beng	0,75		
	3 - wangi	0,81			3 - wangi	0,6		
	4 - tueng	0,78			4 - tông	0,6		
	5 - rueng	0,78			5 - tueng	0,6		
	6 - mieng	0,78			6 - sèng	0,6		
	7 - lueng	0,78			7 - rueng	0,6		
	8 - bueng	0,78			8 - pèng	0,6		
	9 - bieng	0,78			9 - mieng	0,6		
	10 - aseng	0,78			10 - lông	0,6		

Sumber: Data diolah (2023)

Dari hasil evaluasi ini, dapat disimpulkan bahwa algoritma Jaro-Winkler lebih unggul dan tepat digunakan dalam fitur *recommendation list* atau sistem rekomendasi kata pada aplikasi kamus bahasa Aceh dalam kasus dimana kata kunci bahasa Aceh yang diketik tidak mengandung tanda diakritik. Hal ini dikarenakan algoritma tersebut mampu memberikan nilai MAP yang tinggi, terutama pada *threshold* yang lebih tinggi. Namun, meskipun algoritma Levenshtein memberikan hasil yang kurang baik dibandingkan dengan algoritma Jaro-Winkler, algoritma tersebut masih memberikan nilai MAP yang cukup tinggi pada *threshold* yang lebih rendah. Oleh karena itu, fitur *recommendation list* atau sistem rekomendasi kata pada bahasa Aceh dapat menggunakan kedua algoritma tersebut sesuai dengan kebutuhan dan kondisi yang diinginkan.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan uraian pada bab sebelumnya, maka dapat disimpulkan hal-hal sebagai berikut:

1. Aplikasi kamus bahasa Aceh berbasis *mobile* dapat dikembangkan dengan menggunakan metode pengembangan aplikasi *Personal Extreame Programming* (PXP).
2. Aplikasi kamus bahasa Aceh berbasis *mobile* yang dilengkapi dengan fitur *recommendation list* dapat menampilkan kosakata yang mengandung tanda diakritik pada hasil pencarian dengan adanya penerapan algoritma Jaro-Winkler pada algoritma fitur rekomendasi.
3. Aplikasi kamus bahasa Aceh berbasis *mobile* yang telah dikembangkan ini merupakan perbaikan dari aplikasi kamus bahasa Aceh yang sudah ada sebelumnya dengan jumlah data yang juga sudah bertambah menjadi 6220 kosakata yang terdiri dari bahasa Aceh, bahasa Indonesia, dan bahasa Inggris.
4. Algoritma Jaro-Winkler tepat untuk diterapkan dalam fitur *recommendation list* atau sistem rekomendasi kata pada aplikasi kamus bahasa Aceh. Hal tersebut dibuktikan dengan hasil evaluasi yang menunjukkan nilai *Mean Average Precision* (MAP) yang tinggi di setiap *threshold* yang diuji.

5.2. Saran

Setelah melakukan penelitian ini, terdapat beberapa saran yang dapat dilakukan pada penelitian selanjutnya. Adapun saran tersebut adalah sebagai berikut:

1. Aplikasi kamus bahasa Aceh berbasis *mobile* ini dikembangkan menggunakan metode pengembangan *Personal Extreame Programming* (PXP), pada penelitian

selanjutnya dapat mencoba menerapkan metode pengembangan perangkat lunak yang lain.

2. Data yang terdapat pada *database* aplikasi kamus bahasa Aceh berbasis *mobile* ini adalah sebanyak 6220 kosakata yang terdiri dari tiga jenis bahasa. Pada penelitian selanjutnya, dapat memperkaya *database* dengan menambahkan jumlah data, jenis bahasa terjemahan, dan contoh penggunaan kata pada kalimat.
3. Mengembangkan dan memperkaya fitur aplikasi seperti:
   - Menyediakan fitur pencarian menggunakan suara (*voice search*).
   - Fitur pembelajaran bahasa Aceh dengan menambahkan modul-modul pelajaran atau menambahkan latihan soal seperti *quiz*, dsb.
   - Fitur terjemahan bahasa Aceh ke bahasa lain atau sebaliknya.
   - Fitur lokasisasi bahasa, agar dapat menjangkau dan memudahkan pengguna yang memiliki latar belakang bahasa ibu yang berbeda untuk mempelajari bahasa Aceh.
4. Selain itu, dapat juga menganalisis dan memperbaiki kesalahan penulisan tanda diakritik yang mungkin masih terdapat pada aplikasi ini.

DAFTAR PUSTAKA

Aditya Widjaja, A., & Novianus Palit, H. (2022). Hybrid Recommendation System untuk Peminjaman Buku Perpustakaan dengan Collaborative dan Content-Based Filtering. *JURNAL INFRA*, *10*(2), 1–6. https://publication.petra.ac.id/index.php/teknik-informatika/article/view/12512

Agil M. Caesar. (2018). *Perancangan Aplikasi Kamus Bahasa Aceh Berbasis Android*. Universitas Sumatera Utara.

Aguswandi. (2021). *Kamus Aceh Lengkap* (2.0). Google Play Store. https://play.google.com/store/apps/details?id=com.agw.aceh

Arfisko, H. H., & Wibowo, A. T. (2022). Sistem Rekomendasi Film Menggunakan Metode Hybrid Collaborative Filtering Dan Content-Based Filtering. *e-Proceeding of Engineering*, *9*(3), 2149–2159.

aws.amazon.com. (t.t.). *Apa itu API?* Diambil 13 September 2022, dari https://aws.amazon.com/id/what-is/api/

Azklabs Mobile. (2020). *Kamus Bahasa Aceh Lengkap* (13.0). Google Play Store. https://play.google.com/store/apps/details?id=com.idkamus.acehoffline

Costa, R. (2020, Januari 16). *Low fidelity vs high fidelity wireframes: what's the difference?* https://www.justinmind.com/wireframe/low-fidelity-vs-high-fidelity-wireframing-is-paper-dead

dart.dev. (t.t.). *Dart*. Diambil 15 September 2022, dari https://dart.dev/

Dzhurov, Y., Krasteva, I., & Ilieva, S. (2009). *Personal Extreme Programming-An Agile Process for Autonomous Developers*. https://core.ac.uk/display/213561553?utm_source=pdf&utm_medium=banner&utm_campaign=pdf-decoration-v1

Faathir, M. W. (2018). *Perbandingan Algoritma Hosrpool Dan Not So Naive Dalam Pembuatan Kamus Bahasa Indonesia – Bahasa Aceh Beerbasis Android*. Universitas Sumatera Utara.

Fardan, A., & Zulkarnaini. (2019). *Kamus Aceh Indonesia Inggris* (Bahri Rajab & Eriyanti, Ed.; 2019 ed.). CV. Boebon Jaya.

flutter.dev. (t.t.). *Flutter*. Diambil 15 September 2022, dari https://flutter.dev/

gin-gonic.com. (t.t.). *Gin Web Framework*. Diambil 17 September 2022, dari https://gin-gonic.com

Girindra, P. (t.t.). *Jenis-Jenis Software Testing*. Diambil 12 September 2022, dari https://qatros.com/blog/blog-technology-1/post/jenis-jenis-software-testing-54

Glenn Tambahani, A., Sanjaya, A., & Widodo, D. W. (2021). Penerapan Metode Jaro Winkler Untuk Mencari Kemiripan Kata Pada Aplikasi English Pronunciation Test. *Seminar Nasional Inovasi Teknologi UN PGRI Kediri*.

go.dev. (t.t.). *The Go Programming Language*. Diambil 16 September 2022, dari https://go.dev/

Gupta, L. (2022, April 7). *What is REST*. https://restfulapi.net/

Haavisto, J. (2022). *go-jaro-winkler-distance* (v0.0.0-20220308004700-32671379e37c). https://pkg.go.dev/. https://pkg.go.dev/github.com/jhvst/go-jaro-winkler-distance

Handayani, M. T. (2022, Januari 11). *Wireframe: 3 Elemen dan Bedanya dengan Mockup Dan Prototype*. https://www.ekrut.com/media/wireframe-adalah

ibm.com. (2021, Agustus 23). *Database Schema*. https://www.ibm.com/cloud/learn/database-schema

Idham, M., & Azwardi. (2008). Analisis kesalahan penulisan bahasa Aceh pada media Luar Ruang di Kota Banda Aceh. *Langgam Bahasa*, *2*(2), 74–82. https://www.onesearch.id/Record/IOS3239.slims-4880

Julian Tannga, M., Rahman, S., & Hasniati. (2017). Analisi Perbandingan Algoritma Levenshtein Distance Dan Jaro Winkler Untuk Aplikasi Deteksi Plagiarisme Dokumen Teks. *JTRISTE*, *4*(1), 44–54.

jwt.io. (t.t.). *Introduction to JSON Web Tokens*. Diambil 17 Februari 2023, dari https://jwt.io/introduction

KathrynEE, Mabee, D., Casey, L., v-chmccl, v-rajagt-zz, Coulter, D., Petersen, T., Jacobs, M., chcomley, & Danielson, S. (2022, Agustus 10). *Agile process work item types for workflow management in Azure Boards*. Microsoft. https://learn.microsoft.com/en-us/azure/devops/boards/work-items/guidance/agile-process-workflow?view=azure-devops

Khaidir, M. (2018). *Kamus Inggris Aceh* (2.0). Google Play Store. https://play.google.com/store/apps/details?id=id.kamus.splashscreen

Kristien Margi S, & Agus T. (2016). Pengkoreksian Dan Suggestion Word Pada Keyword Menggunakan Algoritma Jaro Winkler. *Jurnal Teknologi Informasi-Aiti*, *13*(2), 169–181. https://ejournal.uksw.edu/aiti/article/view/1271/615

Melinda, M., Indra Borman, R., & Redy Susanto, E. (2017). *Rancang Bangun Sistem Informasi Publik Berbasis Web (Studi Kasus : Desa Durian Kecamatan Padang Cermin Kabupaten Pesawaran)* (Vol. 11, Nomor 1).

Nimrod Studio. (2019). *Kamus Resmi Bahasa Aceh Offlin* (5.0). Google Play Store. https://play.google.com/store/apps/details?id=com.nimrod.kamusbahasaaceh

Novantara, P., & Pasruli Opin. (2018). Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi. *Jurnal Buffer Informatika*, *3*(2). https://www.researchgate.net/publication/338423894_Implementasi_Algoritma_Jaro-Winkler_Distance_Untuk_Sistem_Pendeteksi_Plagiarisme_Pada_Dokumen_Skripsi

Novianti, A. (2022). *Literature Review : Analisis Metodologi Dan Bidang Penerarapan Dalam Perancangan Aplikasi Mobile*. https://www.researchgate.net/publication/359809801_LITERATURE_REVIEW_ANALISIS_METODOLOGI_DAN_BIDANG_PENERARAPAN_DALAM_PERANCANGAN_APLIKASI_MOBILE?enrichId=rgreq-da51bb96b595e2c504913592210fa7c2-

XXX&enrichSource=Y292ZXJQYWdlOzM1OTgwOTgwMTtBUzoxMTQyMzk2NTY1ODkzMTIwQDE2NDkzNzk5NzQ0MTM%3D&el=1_x_2&_esc=publicationCoverPdf

Nurpita, R., Iba, H., & Safriandi. (2021). Analisis Persamaan Dan Perbedaan Pelafalan Dalam Bahasa Aceh Antara Dialek Aceh Selatan Dan Dialek Aceh Utara. *Jurnal Dedikasi Pendidikan*, *5*(2), 417–430.

Pebrijayanti, R., & Ardian, Z. (2018). Rancang Bangun Aplikasi Kamusbahasaindonesia-bahasa Aceh Menggunakan Metode Rule Based Berbasis Android. *Journal of Informatics and Computer Science*, *4*(1).

portalsatu.com. (2016, Mei 4). *Cara Menulis Tanda Diakritik dalam Bahasa Aceh*. https://portalsatu.com/cara-menulis-tanda-diakritik-dalam-bahasa-aceh/#:~:text=Tanda%2Dtanda%20diakritik%20yang%20sebutkan,%20dan%20apostrof%20%2F'%2F

Prasetyo, A., Baihaqi, W. M., & Had, I. S. (2018). Algoritma Jaro-Winkler Distance: Fitur Autocorrect dan Spelling Suggestion pada Penulisan Naskah Bahasa Indonesia di BMS TV. *Jurnal Teknologi Informasi dan Ilmu Komputer*, *5*(4), 435. https://doi.org/10.25126/jtiik.201854780

Rizal. (2015). Permainan Tebak Kata Bahasa Aceh Menggunakan Algoritma. *Techsi*, *6*(1), 170–188.

Rochmawati, Y., & Kusumaningrum, R. (2016). Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching untuk Identifikasi Kesalahan Pengetikan Teks. *Jurnal Buana Informatika*, *7*(2), 125–134.

Subhayni, Armia, & Nurrahmah. (2020). Restrukturisasi Sapaan Kekerabatan Bahasa Aceh Sebagai Pendidikan Strategi Tutur Sapa Bagi Kaum Muda Aceh. *Jurnal Serambi Ilmu*, *21*(1), 118–130.

Suhartono, J. (2016, Desember 16). *System Testing*. https://sis.binus.ac.id/2016/12/16/system-testing/

Syahputra, I., & Syakti, F. (2022). Perbandingan Algoritma Levenshtein dan Jaro Winkler Pada Sistem Informasi Pencarian Dokumen Perundang-Undangan

(Studi Kasus : Diskominfo Lahat). *SMATIKA JURNAL*, *12*(02), 176–186. https://doi.org/10.32664/smatika.v12i02.696

techniqueapp. (2022). *Kamus Indonesia - Aceh* (1.3). techniqueapp. https://play.google.com/store/apps/details?id=appinventor.ai_zamronizam.bahasa

Yulianingsih, Y. (2017). Implementasi Algoritma Jaro-Winkler dan Levenstein Distance dalam Pencarian Data pada Database. *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, *2*(1), 18. https://doi.org/10.30998/string.v2i1.1720

LAMPIRAN

Adapun lampiran dari penelitian Pengembangan Aplikasi Kamus Bahasa Aceh Berbasis *Mobile* dapat diakses pada link berikut:

https://1drv.ms/b/s!ApFzxrNKsaNahS2Uiuws1gFZNbK6?e=vYo77d