RANCANG BANGUN *FRONTEND* ALEA (AKSARA *LEARNING APP)* BERBASIS *MOBILE*

TUGAS AKHIR

Diajukan Oleh:

M DOLYANDA HARIALDY

NIM. 210705112

Mahasiswa Fakultas Sains dan Teknologi

Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI AR-RANIRY

BANDA ACEH

2025 M / 1447 H

LEMBAR PERSETUJUAN

RANCANG BANGUN *FRONTEND* ALEA (AKSARA *LEARNING APP)* 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 Prodi Teknologi Informasi

Oleh:
M DOLYANDA HARIALDY
210705112
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui untuk Munaqasyah Oleh:

Pembimbing I,	Pembimbing II,
Dr. Hendri Ahmadian, M.I.M	**Mulkan Fadhli, S.T., M.T.**
NIP. 198301042014031002	**NIP. 198811282020121006**

Mengetahui,

Ketua Program Studi Teknologi Informasi

Malahayati, M.T
NIP. 198301272015032003

LEMBAR PENGESAHAN

RANCANG BANGUN *FRONTEND* ALEA (AKSARA *LEARNING APP)* BERBASIS *MOBILE*

TUGAS AKHIR

Telah Diuji Oleh Panitia Ujian Munaqasyah Tugas Akhir
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus
Serta Diterima Sebagai Salah Satu Beban Studi Program Sarjana (S1)
Dalam Program Studi Teknologi Informasi

Pada Hari/Tanggal: Senin, 03 Juli 2025
7 Muharram 1447 H
Di Darussalam, Banda Aceh

Panitia Ujian Munaqasyah Tugas Akhir:

Ketua,	Sekretaris,
Dr. Hendri Ahmadian, S.Si, M.I.M	**Mulkan Fadhli, S.T., M.T.**
NIP. 198301042014031002	**NIP. 198811282020121006**
Penguji I,	Penguji II,
Ghufran Ibnu Yasa, M.T	**Aulia Syarif Aziz, M.Sc**
NIP.198409262014031005	**NIP. 199305212022031001**

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

Prof. Dr. Ir. Muhammad Dirhamsyah, M.T., IPU
NIP. 19620021988111001

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	: M Dolyanda Harialdy
Nim	: 210705112
Program Studi	: Teknologi Informasi
Fakultas	: Sains dan Teknologi
Judul	: Rancang Bangun *Frontend* ALEA (Aksara *Learning App*) Berbasis *Mobile*

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

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggung jawabkan;
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 bertanggung jawab 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,

Yang Menyatakan

M Dolyanda Harialdy

ABSTRAK

Nama	: M Dolyanda Harialdy
Nim	: 210705112
Program Studi	: Teknologi Informasi
Judul	: Rancang Bangun *Frontend* ALEA (Aksara *Learning App*) Berbasis *Mobile*
Tanggal Sidang	: 30 Juni 2025 / 4 Muharram 1447 H
Jumlah Halaman	: 95 Halaman
Pembimbing I	: Dr. Hendri Ahmadian, S.Si., M.I.M
Pembimbing II	: Mulkan Fadhli, S.T., M.T.
Kata Kunci	: Aksara, Aplikasi Belajar, Kotlin, CameraX, *Personal Extreme Programming*

Indonesia memiliki beragam aksara tradisional sebagai bagian dari kekayaan budaya daerah. Namun, minat generasi muda untuk mempelajari aksara tersebut semakin menurun, sehingga dibutuhkan media pembelajaran yang menarik dan mudah diakses salah satunya menggunakan aplikasi *mobile*. Penelitian ini bertujuan untuk merancang dan membangun antarmuka pengguna (*frontend*) dari aplikasi ALEA (Aksara *Learning App*) berbasis *mobile*, yang menyediakan fitur belajar aksara Sunda, Bali, Arab Jawi, dan Lampung secara interaktif. Aplikasi dikembangkan dengan menggunakan bahasa pemrograman Kotlin pada Android Studio, dan metode pengembangan yang digunakan adalah Personal *Extreme Programming* (PXP). Fitur utama yang disediakan dalam aplikasi antara lain kamus aksara, kuis, materi pembelajaran, *scanner* aksara menggunakan gambar, serta fitur penggunaan secara *offline*. Hasil akhir menunjukkan bahwa seluruh fitur berhasil diimplementasikan dan terintegrasi dengan baik dengan layanan API *backend server*. Aplikasi ini diharapkan dapat menjadi sarana pembelajaran modern yang efektif sekaligus berkontribusi dalam pelestarian aksara tradisional di Indonesia.

ABSTRACT

Name	: M Dolyanda Harialdy
Nim	: 210705112
Department	: *Information Technology*
Title	: *Frontend Development of ALEA (Aksara Learning App) Mobile-Based Application*
Date	: 30 *June* 2025 / 4 *Muharram* 1447 H
Thesis Pages	: *95 pages*
Supervisor I	: Dr. Hendri Ahmadian, S.Si., M.I.M
Supervisor II	: Mulkan Fadhli, S.T., M.T.
Keywords	: *Aksara, Learning App, Kotlin, CameraX, Personal Extreme Programming*

Indonesia has a diverse range of aksara tradisional as part of its rich regional cultural heritage. However, the interest of younger generations in learning these aksara is steadily declining, creating the need for engaging and accessible learning media, such as mobile applications. This study aims to design and develop the user interface (frontend) of ALEA (Aksara Learning App), a mobile-based application that provides interactive learning features for aksara Sunda, aksara Bali, aksara Arab Jawi, and aksara Lampung. The application was developed using the Kotlin programming language in Android Studio, and the development methodology employed was Personal Extreme Programming (PXP). The main features offered in this application include a script dictionary, quizzes, learning materials, a script scanner using image input, and offline usage capability. The final result shows that all frontend features were successfully implemented and integrated with backend server API services. This application is expected to serve as an effective modern learning tool while contributing to the preservation of traditional aksara in Indonesia.

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr. Wb

Alhamdulillahirabbil'alamin, puja dan puji serta syukur kita ucapkan kehadirat Allah SWT, berkat nikmat dan karunia-Nya yang indah yang masih kita rasakan sampai pada saat ini, nikmat berupa iman, Islam, kesehatan, kesempatan, pengetahuan yang tentunya masih banyak lagi nikmat yang tidak dapat dijabar di atas seluruh kertas ini.

Dalam kesempatan ini penulis bersyukur kepada Allah SWT, berkat Ridho-Nya penulis mampu merampungkan proposal skripsi yang berjudul *"*Rancang Bangun *Frontend* ALEA (Aksara *Learning App*) Berbasis *Mobile"*. Proposal skripsi ini disusun sebagai kewajiban penulis guna melengkapi tugas dan syarat untuk menyelesaikan pendidikan Strata-I (S1) Program Studi Teknologi Informasi Universitas Islam Negeri Ar-Raniry Banda Aceh, serta memperoleh gelar Sarjana Komputer Universitas Islam Negeri Ar-Raniry Banda Aceh.

Dalam proses penyusunan proposal ini, penulis telah mendapatkan banyak bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, dengan segala hormat dan rasa Syukur, penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT yang telah memberikan nikmat, rahmat dan kemudahan dalam pengerjaan Tugas Akhir ini.
2. Yang teristimewa, Ayahanda Ali Zamzami dan Ibunda Nova Kasmarianti, serta keluarga besar penulis, khususnya saudara Apriliya Familiari dan Jaisyi Islami, yang senantiasa menjadi sumber kekuatan, cinta, dan doa yang tiada henti. Semoga Allah senantiasa melimpahkan rahmat, kesehatan, dan kebahagiaan kepada mereka, serta menjadikan segala amal mereka sebagai jalan menuju ridha dan cinta-Nya.
3. Bapak Dr. Hendri Ahmadian, M.I.M Selaku dosen pembimbing I Tugas Akhir yang telah dengan sabar membimbing, memberikan saran, kritik, dan masukan berharga selama proses penyusunan tugas akhir ini.
4. Bapak Mulkan Fadhli, S.T., M.T. selaku pembimbing II Tugas Akhir yang telah banyak meluangkan waktu serta fikiran dalam memberikan bantuan, kritik, dan masukan berharga untuk membimbing penulis demi kesempurnaan Tugas Akhir ini.

5. Ketua dan Sekretaris Program Studi Teknologi Informasi, Ibu Malahayati, M.T dan Bapak Khairan Ar, M.Kom, serta Bapak dan Ibu dosen Program Studi Teknologi Informasi yang telah memberikan ilmu pengetahuan dalam bidang Teknologi Informasi.
6. Pembimbing Akademik, bapak Mulkan Fadhli, S.T., M.T. yang telah membimbing dan memberikan saran selama masa perkuliahan.
7. Staf Prodi Ibu Cut Ida Rahmadiana, S.Si., yang telah membantu penulis dalam pengurusan administrasi dan surat-menyurat.
8. Ida Yanti, terima kasih karena sudah setia menemani, memberikan dukungan, dan motivasi kepada penulis sehingga penelitian Tugas Akhir ini dapat diselesaikan dengan baik.
9. Rekan satu tim dalam pengembangan ALEA, Arief Fathin Abrar dan Hanafi Akbar. Terima kasih atas kolaborasi yang solid, kerja keras tanpa lelah, serta semangat yang tak pernah surut dalam mewujudkan aplikasi ini.
10. Teman-teman seperjuangan saya, khususnya Dani Rizqullah, Mindarina, Anggil Maulana, Maudy Akmal, Muhammad Arif Rama, dan Yudi Arami terima kasih atas kebersamaan, dukungan, dan semangat yang telah kalian berikan sepanjang perjalanan perkuliahan ini.

Peneliti menyadari bahwa Tugas Akhir ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun selalu peneliti harapkan, demi penyusunan Tugas Akhir yang baik. Peneliti berharap semoga tulisan ini dapat bermanfaat bagi perkembangan ilmu pengetahuan.

Banda Aceh, 03 Juli 2025

M Dolyanda Harialdy

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
BEAS	Belajar Aksara Sunda	3
ALEA	Aksara *Learning App*	3
IOS	*IPhone Operating System*	8
OHA	Open Handset Alliance	8
JVM	*Java Virtual Machine*	10
OOP	*Object Orirented Programming*	10
API	*Application Programming Interface*	11
XML	*eXtensible Markup Language*	11
JSON	*JavaScript Object Notation*	12
REST	*Representational State Transfer*	12
HTTP	*Hypertext Transfer Protocol*	12
PXP	*Personal Extreme Programming*	12
PSP	*Personal Software Process*	12
XP	*Extreme Programming*	12
MVP	*Minimum Viable Product*	14
SQL	*Structured Query Language*	26
URL	*Uniform Resource Locator*	52
HTML	*Hypertext Markup Language*	73

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia dikenal sebagai negara yang kaya akan keragaman budaya dan bahasa. Negara ini memiliki lebih dari 700 bahasa daerah dan berbagai aksara tradisional yang menjadi identitas setiap suku dan daerah (Hardyanto, 2023). Namun seiring dengan perkembangan zaman dan pengaruh globalisasi, keberadaan aksara tradisional semakin terpinggirkan dan terancam punah. Minat generasi muda dalam mempelajari aksara daerah terus menurun, bahkan beberapa aksara hampir hilang. Menurut Erniati (2020) fenomena ini terjadi karena beberapa faktor, salah satunya adalah tidak adanya pewarisan bahasa secara terstruktur dari orang tua kepada generasi penerus, yang seharusnya menjadi pelestari bahasa daerah. Situasi ini menunjukkan urgensi untuk melestarikan aksara tersebut sebelum hilang ditelan waktu.

Menghadapi tantangan ini, media digital, khususnya aplikasi *mobile*, menjadi pilihan efektif sebagai sarana pembelajaran. Menurut Riyan (2021) aplikasi *mobile* menawarkan berbagai macam kelebihan, seperti akses yang cepat baik secara *online* maupun *offline* dan dapat diakses dimana saja dan kapan saja. Selain itu, aplikasi *mobile* juga dapat mengintegrasikan berbagai metode belajar, seperti kuis interaktif, serta menyediakan informasi yang dapat diperbarui secara berkala. Hal ini memungkinkan penyampaian informasi terbaru mengenai aksara tradisional kepada pengguna secara efektif dan efisien. Keunggulan-keunggulan tersebut menjadikan media digital, khususnya aplikasi *mobile*, lebih fleksibel dan lebih diminati dibandingkan dengan media konvensional seperti buku dan majalah. Dengan demikian, aplikasi *mobile* tidak hanya memenuhi kebutuhan belajar yang dinamis tetapi juga mendukung pelestarian aksara tradisional melalui pendekatan yang inovatif dan menarik bagi generasi muda.

Upaya pelestarian aksara daerah secara digital juga masih sangat minim. Jika kondisi ini dibiarkan, bukan tidak mungkin aksara daerah di Indonesia akan punah. Bahkan, kurang dari lima persen bahasa di dunia yang tersedia dapat diakses secara *online*, yang menunjukkan betapa pentingnya digitalisasi dalam upaya

pelestarian bahasa dan aksara tradisional. Kurangnya akses digital tidak hanya membatasi penyebaran pengetahuan tentang aksara tersebut, tetapi juga menghambat generasi muda untuk belajar dan mengapresiasi warisan budaya mereka melalui *platform modern* yang mereka gunakan sehari-hari. Oleh karena itu, integrasi aksara tradisional ke dalam media digital menjadi langkah strategis yang esensial untuk memastikan keberlanjutan dan relevansi aksara tersebut di era digital (Nur Islamia et al., 2024) .

Penulis disini mengambil 4 jenis aksara untuk diintegrasikan ke dalam aplikasi *mobile* yang akan dikembangkan, yaitu aksara Sunda, aksara Bali, aksara , dan juga aksara Lampung. Keempat aksara ini diambil dengan beberapa pertimbangan, mulai dari ketersediaan data untuk diimplementasikan, sebaran wilayah, dan juga relevansi budaya dalam penggunaan sehari-hari. Saat proposal ini ditulis, diketahui telah tersedia beberapa aplikasi belajar aksara berbasis aplikasi *mobile*. Di sini penulis menggunakan kata kunci "belajar aksara" dan "Aksara" pada toko aplikasi Google Play Store, beberapa diantara aplikasi tersebut yaitu "Pegon-Gliph: Belajar Pegon", "Tuntunan Baca Tulis Pegon Jawa", "Baja – Belajar Aksara Jawa", "Belajar Aksara Sunda (BEAS)", dan juga "Majalah Aksara Bali". Selain itu terdapat juga beberapa penelitian yang melakukan perancangan aplikasi belajar aksara, diantaranya yaitu penelitian yang dilakukan oleh Fatah et al., (2020) dengan judul "*Rancang Bangun Program Aplikasi Pembelajaran Aksara Sunda Berbasis Android*" dan "*Aplikasi Multimedia Pembelajaran Aksara Jawa Berbasis Android Untuk Siswa Sekolah Dasar*" oleh Hartiyani et al., (2023). Namun, aplikasi yang dihasilkan dari penelitian dan juga aplikasi yang terdapat toko aplikasi tersebut masih terdapat berbagai kekurangan, seperti hanya tersedia untuk satu bahasa, tidak memiliki fitur *scanner* untuk menerjemahkan gambar aksara menjadi teks bahasa Indonesia, dan lingkungan belajar yang kurang lengkap karena tidak terintegrasinya fitur seperti kamus, *quiz* dan juga fitur belajar sejarah, dan belajar aksara yang lengkap di dalam satu aplikasi. Tentunya kekurangan ini merupakan suatu hal yang tidak boleh untuk terus diabaikan.

Oleh karena itu demi menyempurnakan aplikasi belajar aksara yang telah ada, penelitian ini akan berfokus pada pengembangan aplikasi belajar aksara berbasis *mobile* dengan penambahan berbagai fitur yang dirasa perlu seperti fitur

untuk menerjemahkan gambar aksara menjadi teks bahasa Indonesia, menyediakan 4 kamus aksara di dalam satu aplikasi, dan juga pengintegrasian lingkungan belajar yang mencakup belajar aksara, belajar sejarah, kamus, dan fitur *quiz*. Aplikasi keluaran penelitian ini bernama "Aksara *Learning App*" yang disingkat menjadi "ALEA". Pengembangan aplikasi ALEA dibagi menjadi tiga bagian yaitu *frontend, backend,* dan juga *machine learning*, adapun penulis bertanggung jawab untuk mengembangkan antarmuka sisi *client* aplikasi. Adapun untuk *backend* aplikasi akan dikembangkan oleh Hanafi Akbar (2025) dalam penelitiannya yang berjudul "*Desain dan Implementasi Arsitektur Backend Berbasis Google Cloud Pada Aplikasi ALEA Untuk Pembelajaran Aksara*", dan *machine learning* yang digunakan oleh aplikasi ALEA akan dikembangkan oleh Arief Fathin Abrar (2025) di dalam penelitiannya yang berjudul "*Implementasi Arsitektur YOLOv8 Untuk Mengklasifikasikan Aksara Pada Aksara Learning App*", aplikasi ini diharapkan dapat memperbaiki dan menutup kekurangan dari aplikasi yang telah ada sebelumnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan oleh penulis maka rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana membangun antarmuka fitur *scanner* aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung menggunakan *input* gambar berbasis aplikasi *mobile*?
2. Bagaimana membangun antarmuka fitur *quiz* aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung berbasis aplikasi *mobile*?
3. Bagaimana membangun antarmuka fitur *dictionary, history dan* materi aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung berbasis aplikasi *mobile?*
4. Bagaimana membangun aplikasi *mobile* aksara Sunda, aksara Bali, aksara Arab Jawi, dan aksara Lampung yang dapat digunakan secara *online* dan *offline* dengan data yang dinamis*?*

1.3 Tujuan Penelitian

Berdasarkan latar belakang dan juga rumusan masalah yang telah diuraikan oleh penulis maka tujuan dari penelitian ini adalah sebagai berikut:

1. Membangun antarmuka fitur *scanner* aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung menggunakan *input* gambar berbasis aplikasi *mobile*.
2. Membangun antarmuka fitur *quiz* aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung berbasis aplikasi *mobile*.
3. Membangun antarmuka fitur antarmuka fitur *dictionary, history dan* materi aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung berbasis aplikasi *mobile.*
4. Membangun aplikasi mobile aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung yang dapat digunakan secara *online* dan *offline* dengan data dinamis yang dapat di *update*.

1.4 Manfaat penelitian

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

1. Memberikan kontribusi dalam menjaga dan melestarikan sejarah dan juga aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung di Indonesia dengan menggunakan Teknologi Informasi.
2. Aplikasi belajar aksara yang dirancang dan dibangun pada penelitian ini dapat menjadi media belajar dan menjadi referensi untuk menulis dalam bahasa aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung.
3. Aplikasi belajar aksara yang dirancang dan dibangun dalam penelitian ini dapat mengatasi keterbatasan membaca aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung dengan memanfaatkan fitur *scanner* yang tersedia.

1.5 Batasan Penelitian

Untuk memastikan kesesuaian pembahasan penelitian ini dengan judul dan latar belakang yang telah diuraikan, penulis memutuskan untuk mempersempit cakupan masalah yang akan dibahas dalam penelitian ini dengan menetapkan batasan sebagai berikut:

1. Aksara yang akan digunakan dan implementasikan terbatas pada aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung.
2. Aplikasi yang dibangun terbatas pada aplikasi berbasis *mobile* yaitu pada *platform* android.
3. Bahasa pemrograman yang digunakan untuk membangun aplikasi berbasis *mobile* adalah bahasa pemrograman Kotlin.
4. Fitur *scanner* aksara Sunda, aksara Bali, aksara Arab Jawi, dan juga aksara Lampung. hanya akan menerima berkas berupa gambar, yang berasal dari hasil *capture* secara langsung dan dari galeri.
5. Model *machine learning* yang digunakan berasal dari penelitian Arief Fathin Abrar (2025) berjudul "Implementasi Arsitektur YOLOv8 Untuk Mengklasifikasikan Aksara Pada Aksara *Learning App"*.
6. *Server backend* yang digunakan berasal dari penelitian Hanafi Akbar (2025) berjudul "Desain dan Implementasi Arsitektur *Backend* Berbasis *Google Cloud* Pada Aplikasi ALEA Untuk Pembelajaran Aksara".

BAB II
TINJAUAN PUSTAKA

2.1 Aksara

Menurut Zahratul & Sarwadan (2024) Aksara merupakan lambang yang mewakili bunyi atau fonem. Dalam Kamus Besar Bahasa Indonesia aksara didefinisikan sebagai sistem tanda grafis yang digunakan oleh manusia untuk berkomunikasi, yang sebagian besar mewakili ujaran. Aksara sering disebut juga sebagai huruf atau abjad dan dipahami sebagai simbol bunyi .Selain itu, aksara dikenal sebagai "sistem tulisan". Dalam perkembangannya, aksara dianggap sebagai sistem simbol visual yang dapat dituliskan pada media seperti kertas, batu, kayu, kain, atau pohon, dan digunakan untuk mengungkapkan elemen ekspresif dalam suatu bahasa. Secara etimologis, kata "aksara" berasal dari bahasa Sanskerta, yaitu "a-" berarti "tidak", dan "kshara" berarti "musnah".

Aksara Lampung, atau dikenal juga sebagai Had Lampung, merupakan sistem tulisan tradisional yang terbagi menjadi dua bentuk, yaitu aksara kuno dan aksara yang telah disederhanakan. Bentuk aksara yang digunakan secara resmi saat ini adalah hasil penyederhanaan yang disahkan melalui musyawarah para tokoh adat Lampung pada 23 Februari 1985. Secara struktural, aksara Lampung terdiri atas kelebai surat (huruf induk) dan benah surat (huruf turunan), serta memiliki kemiripan dengan aksara Rejang Bengkulu, aksara Rencong, aksara Sunda, dan aksara Lontara. Sistem ini mencakup huruf pokok, huruf turunan, huruf ganda, gugus konsonan, serta simbol, angka, dan tanda baca. Aksara Lampung, yang juga dikenal dengan istilah Kaganga, ditulis dari kiri ke kanan dan terdiri atas dua puluh huruf induk sebagai dasar penulisan (Kholidah et al., 2023).

bahasa Bali dalam bentuk simbol-simbol grafis. Aksara ini memiliki karakteristik unik berupa bentuk huruf yang melengkung dan struktur yang bersifat alfa-silabis, yaitu setiap huruf konsonan secara bawaan mengandung vokal ‘a’. Dalam penggunaannya, aksara Bali terbagi menjadi dua kategori utama berdasarkan fungsi dan konteks pemakaiannya, yaitu aksara biasa dan aksara suci. Aksara biasa mencakup wreastra, yang digunakan dalam penulisan bahasa Bali sehari-hari, serta swalalita, yang digunakan untuk menuliskan kata-kata dari bahasa

serapan seperti Sanskerta dan Kawi. Aksara dasar dalam sistem ini berjumlah 18 karakter utama, yang menjadi fondasi dalam pembelajaran dan penulisan. Sementara itu, aksara suci terdiri dari wijaksara dan modre, yang memiliki nilai spiritual dan digunakan dalam konteks keagamaan Hindu, terutama pada naskah-naskah suci dan ritual magis. Aksara modre dikenal karena bentuknya yang menyerupai simbol mistik dan tidak digunakan dalam komunikasi umum, melainkan dalam praktik spiritual yang bersifat sakral (Gunada et al., 2022).

Aksara Pegon atau yang sering disebut dengan nama Arab-Melayu adalah sistem penulisan yang menggunakan aksara Arab, tetapi ejaan dan pelafalannya menggunakan bahasa Melayu. Jenis tulisan ini juga dikenal dengan nama Jawi, Arab Jawi, Pegon atau Harah-Jawoe di Aceh. Ketika Islam mulai dianut dan menyebar di Nusantara, penulisan bahasa Melayu mulai menggunakan huruf Arab. Pada era tersebut, Arab-Melayu menjadi bahasa tulis resmi di kalangan masyarakat dan kerajaan Islam. Penulisan huruf Arab lainnya umumnya tidak menggunakan harakat sehingga tampak seperti Arab gundul, sedangkan huruf illat yaitu alif, waw, dan ya digunakan untuk menandakan vokal A, I, dan U. Akan tetapi, tidak semua kalimat vokal di dalam bahasa melayu penulisannya dapat di bantu oleh huruf illat (Jannah, 2023).

Aksara Sunda merupakan salah satu sistem tulisan tradisional yang menjadi bagian dari warisan budaya masyarakat Sunda di Jawa Barat. Sebagai media komunikasi, aksara ini digunakan untuk menyampaikan pesan, baik dalam interaksi antar individu maupun antar kelompok. Secara historis, aksara Sunda berasal dari turunan aksara Brahmi, yang berkembang menjadi dua bentuk utama, yaitu aksara Sunda Kuno dan aksara Sunda Baku. Aksara Sunda Baku sendiri terbagi menjadi tiga jenis, yakni aksara swara (huruf vokal), aksara ngalagena (huruf konsonan), dan aksara rarangkén (tanda diakritik), yang masing-masing memiliki fungsi dan bentuk yang berbeda (Sari, 2023).

2.2 Perancangan Aplikasi *Mobile*

Perancangan adalah suatu sekumpulan aktivitas yang menggambarkan secara rinci bagaimana sistem akan berjalan (Fauzi et al., 2023). Perancangan juga dapat diartikan sebagai proses menggambarkan rencana dan membuat sketsa atau

menyusun berbagai elemen terpisah menjadi satu kesatuan yang berfungsi dengan baik. Sementara itu, aplikasi adalah kumpulan perintah yang dibuat untuk menjalankan tugas-tugas tertentu. Menurut Waruwu et al., (2023) Aplikasi *mobile*, atau yang sering disebut *mobile apps*, merupakan jenis perangkat lunak yang dirancang untuk dijalankan pada perangkat bergerak seperti *smartphone* maupun tablet, dan beroperasi secara mandiri melalui sistem operasi yang mendukungnya. Pengguna dapat memperoleh aplikasi-aplikasi tersebut dengan mengunduhnya melalui platform distribusi resmi yang sesuai dengan sistem operasi perangkat, seperti Google Play Store untuk Android atau App Store untuk *Iphone Operating System* (IOS). Sedangkan aplikasi pembelajaran adalah media yang dapat digunakan perangkat *mobile* untuk menyampaikan isi materi yang melibatkan perangkat bergerak seperti ponsel berbasis android.

2.3 Android

Menurut Ditha et al., (2023) Android merupakan sistem operasi berbasis Linux yang dirancang khusus untuk perangkat bergerak, seperti *smartphone* dan tablet, dengan mengintegrasikan sistem operasi, *middleware*, serta berbagai aplikasi pendukung. Sistem ini memberikan platform terbuka bagi para pengembang untuk merancang dan mengembangkan aplikasi secara bebas. Android sendiri menggunakan versi modifikasi dari kernel Linux serta komponen perangkat lunak sumber terbuka lainnya.

2.3.1 Sejarah Singkat Android

Android awalnya dikembangkan oleh Android Inc., sebuah perusahaan yang kemudian diakuisisi oleh Google pada tahun 2005. Pada tahun 2007, Google bersama dengan beberapa perusahaan teknologi terkemuka membentuk Open Handset Alliance (OHA), sebuah konsorsium yang bertujuan untuk mengembangkan standar terbuka bagi perangkat *mobile*. Anggota awal OHA termasuk perusahaan-perusahaan besar seperti Texas Instruments, Intel, LG, Motorola, Samsung Electronics, Qualcomm, dan HTC. Pada tanggal 9 Desember 2008, proyek Android semakin berkembang dengan bergabungnya 14 anggota baru, termasuk Sony Ericsson, Toshiba, Vodafone, dan Garmin Ltd.. Dengan dukungan

dari berbagai perusahaan teknologi, Android berkembang menjadi sistem operasi *mobile* yang dominan di dunia, dikenal karena sifatnya yang terbuka dan fleksibel bagi pengembang serta produsen perangkat (Ditha et al., 2023).

2.3.2 Kelebihan Android

Menurut Hartono & Fauzi (2021) Android mempunyai berbagai macam keunggulan yaitu :

a. Open Source

Android merupakan sistem operasi berbasis Linux yang berarti android menyediakan sumber daya yang dapat dikembangkan bagi siapa saja yang ingin mengembangkannya, hal dapat kita lihat dengan banyaknya versi android yang di modifikasi tersebar di internet.

b. Dukungan Google

Dikarenakan android merupakan salah satu aset yang di miliki oleh Google, hal ini membuat semua hal yang berhubungan dengan Google dengan otomatis tersinkronisasi *device* android contohnya seperti Youtube, Chrome, Google Maps dan lain-lain.

c. Modifikasi

Karakteristik *open-source* yang dimiliki oleh sistem operasi Android memberikan keleluasaan bagi para pengembang dalam melakukan kustomisasi dan pengembangan lanjutan. Hal ini memungkinkan Android untuk diadaptasi sesuai dengan kebutuhan spesifik.

d. User Friendly

Sistem operasi Android dikenal memiliki antarmuka yang intuitif dan *user-friendly*, sehingga memudahkan proses adaptasi bahkan bagi pengguna yang belum memiliki pengalaman sebelumnya. Kemudahan navigasi serta desain yang responsif memungkinkan pengguna baru untuk memahami dan mengoperasikan perangkat berbasis Android dalam waktu yang relatif singkat.

2.4 Kotlin

Kotlin adalah bahasa pemrograman yang berjalan di atas Java Virtual Machine (JVM). Dan bersifat statis artinya bahasa pemrograman kotlin dapat menggunakan paradigma *Object Orirented Programming* (OOP) dan Fungsional. Kotlin sendiri merupakan bahasa pemrograman *open-source* yang dikembangkan oleh Jet Brains untuk berbagai macam platform akan tetapi bahasa pemrograman kotlin sendiri saat ini sangat populer digunakan untuk membangun aplikasi Android (Siswanto, 2022).

Bahasa pemrograman kotlin merupakan bahasa pemrograman yang tergolong relatif mudah untuk dipelajari, hal ini karena sebagian besar kodenya mirip dengan java, bahkan lebih ringkas, bahasa pemrograman kotlin juga dapat dengan mudah diintegrasikan dengan IDE seperti NetBeans, IntelliJ, Visual Studio Code, dan juga Android Studio (Dzulqarnain & Tukino, 2023).

2.5 Jetpack

Jetpack adalah kumpulan *library*, *tools*, dan panduan pengembangan yang disediakan oleh Google untuk membantu pengembang membangun aplikasi Android berkualitas tinggi dengan lebih mudah dan efisien. Jetpack dirancang untuk mengurangi kode *boilerplate* yang berlebihan, menyederhanakan pekerjaan kompleks, dan memungkinkan pengembang untuk lebih fokus pada logika bisnis aplikasi (Sudiatmika et al., 2022).

2.6 Android Studio

Android studio *Integrated Development Environment* yang dibuat dan dikembangkan oleh Google khusus untuk pengembangan aplikasi berbasis Android. Android Studio menggabungkan berbagai macam fitur dan komponen-komponen yang mendukung pengembangan, pengujian, dan penyebaran aplikasi android (Sulfani, 2023). Android studio dirilis secara resmi pada tahun 2013 dan langsung menarik banyak perhatian dikarenakan memiliki fitur yang lengkap.

2.7 CameraX

CameraX adalah salah satu *library* jetpack, yang diciptakan untuk mempermudah developer dalam hal pengembangan aplikasi yang menggunakan kamera. Konsistensi antarmuka API yang dimiliki CameraX menjadikannya dapat mudah digunakan pada sebagian besar perangkat android, CameraX memiliki kompatibilitas hingga Android versi 5 Lollipop (Evelyn et al., 2022).

2.8 *Application Programming Interface* (API)

Application Programming Interface (API) adalah sistem yang terdiri dari dokumentasi berisi antarmuka, fungsi, kelas, struktur, dan komponen lainnya yang dirancang untuk membangun perangkat lunak. Dengan API, pengembang dapat dengan mudah "membongkar" suatu perangkat lunak dan mengembangkannya lebih lanjut atau mengintegrasikannya dengan sistem lain. API bertindak sebagai penghubung antara satu aplikasi dengan aplikasi lainnya, memungkinkan pengembang menggunakan fungsi-fungsi yang telah disediakan tanpa perlu membangun semuanya dari awal (Yudhistira, 2021).

2.8.1 Cara Kerja API

Arsitektur API umumnya dijelaskan dalam konteks hubungan antara klien dan *server*. Aplikasi yang mengirimkan permintaan disebut sebagai klien, sedangkan aplikasi yang memberikan respons disebut sebagai *server*. Berdasarkan perkembangan dan kebutuhan tertentu,

2.8.2 Jenis-Jenis API

Secara umum, terdapat empat jenis API yang sering digunakan dalam pengembangan perangkat lunak, yaitu:

1. API *Simple Object Access Protocol*

API ini menggunakan protokol berbasis *eXtensible Markup Language (*XML) untuk pertukaran pesan antara klien dan *server* serta kurang fleksibel dibandingkan dengan API *modern* lainnya.

2. API *Remote Procedure Call*

API ini memungkinkan klien untuk mengeksekusi fungsi atau prosedur pada *server*, kemudian *server* akan mengirimkan hasil atau *output* kembali kepada klien.

3. API *WebSocket*

API ini merupakan pengembangan *modern* yang mendukung komunikasi dua arah antara aplikasi klien dan *server* menggunakan format *JavaScript Object Notation* (JSON). *WebSocket* memberikan kemampuan bagi *server* untuk mengirimkan pesan secara *real-time* ke klien yang terhubung, sehingga lebih efisien dibandingkan API *Representational State Transfer* (REST) dalam skenario tertentu.

4. API *Representational State Transfer*

Merupakan API yang paling banyak digunakan saat ini karena fleksibilitasnya. Klien mengirimkan permintaan kepada *server* melalui *Hypertext Transfer Protocol* (HTTP), dan *server* akan memproses permintaan tersebut serta mengembalikan data *output* dalam format yang disederhanakan, tanpa perlu *rendering* grafis.

2.9 *Personal Extreme Programming* (PXP)

Menurut Firdaus (2023) *Personal Extreme Programming* (PXP) merupakan salah satu pendekatan pengembangan perangkat lunak berbasis *Agile* yang disesuaikan agar seorang pengembang individu dapat menggunakannya dalam proyek pengembangan aplikasi. PXP merupakan kombinasi dari metode *Personal Software Process* (PSP) yang telah disederhanakan dengan penerapan praktik dari *Extreme Programming* (XP), dan disesuaikan agar lebih mudah diterapkan oleh pengembang yang bekerja sendiri. Salah satu konsep penting dalam metode ini adalah *user story*, yaitu deskripsi singkat yang menggambarkan fitur yang diinginkan oleh pengguna akhir. *User story* biasanya ditulis dalam bentuk sederhana yang menjelaskan kebutuhan pengguna, tujuan dari penggunaan fitur tersebut, dan manfaat yang ingin dicapai. Metode PXP mencakup beberapa tahapan, yaitu: *Requirements, Planning, Iteration Initialization, Design, Implementation, System Testing,* dan *Retrospective.* Dengan rincian sebagai berikut:

1. ***Requirements***

Pada tahap ini, kebutuhan aplikasi diidentifikasi, meliputi kebutuhan fungsional dan non-fungsional yang harus dipenuhi.

2. ***Planning***

Tahap ini melibatkan pengembang dalam menyusun berbagai *user story* dan merinci tugas-tugas yang perlu diselesaikan selama setiap iterasi, berdasarkan kebutuhan yang telah diidentifikasi sebelumnya. Setelah itu, skala prioritas dan estimasi waktu ditentukan untuk setiap *user story* dan tugas.

3. ***Iteration Initialization***

Iterasi dimulai dengan mengerjakan *user story* yang memiliki prioritas tinggi. Iterasi berfokus pada pengerjaan fitur tertentu, dengan durasi antara satu hingga tiga minggu tergantung pada tingkat kompleksitasnya. Setiap iterasi diakhiri dengan rilis versi dari fitur yang telah diselesaikan.

4. ***Implementation***

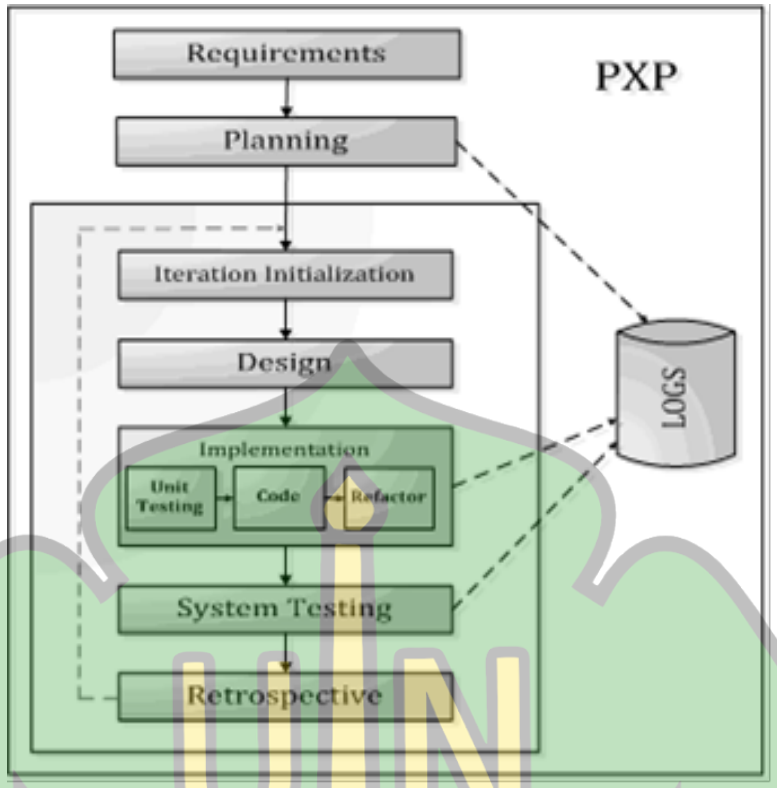
Tahap ini melibatkan penulisan kode aplikasi berdasarkan desain yang telah dibuat, serta pengujian terhadap kode tersebut. Tahap ini terdiri dari tiga sub-tahap, yaitu: pengujian unit, penulisan kode, dan *refactoring*. Implementasi dinyatakan selesai ketika kode berjalan tanpa *error* dan pengujian unit berhasil.

5. ***System Testing***

Pada tahap ini, dilakukan pengecekan untuk memastikan hasil implementasi sesuai dengan kebutuhan yang telah ditetapkan, serta memperbaiki setiap cacat yang ditemukan. *System testing* dilakukan terhadap perilaku aplikasi secara keseluruhan setelah seluruh bagian sistem telah terintegrasi.

6. ***Retrospective***

Tahap ini dilakukan setelah iterasi selesai. Pada tahap ini, dilakukan evaluasi terhadap proses pengembangan yang berlangsung selama iterasi, termasuk menganalisis penyebab keterlambatan atau hambatan yang muncul.



Gambar 2. 1 Tahapan Proses PXP (Firdaus, 2023)

2.10 *Minimum Viable Product* (MVP)

Minimum Viable Product (MVP) adalah pendekatan dalam pengembangan produk yang berfokus pada pembuatan versi awal dengan fitur paling esensial, cukup untuk digunakan oleh pengguna awal serta memberikan dasar untuk evaluasi dan pengembangan selanjutnya.

Menurut Azzahra et al., (2024) MVP dirancang bukan untuk menyajikan produk yang sempurna, melainkan untuk menguji asumsi dasar mengenai kebutuhan pengguna melalui penggunaan nyata. Dengan menyediakan versi minimum dari sebuah produk, pengembang dapat memperoleh umpan balik secara cepat dan langsung dari pengguna, yang kemudian digunakan untuk menyempurnakan aplikasi pada tahap pengembangan berikutnya.

Penerapan MVP bertujuan untuk memastikan bahwa produk awal yang dikembangkan benar-benar memiliki nilai guna dan relevansi bagi pengguna. Fokus utamanya bukan pada kelengkapan fitur, melainkan pada kemampuan produk

dalam menyelesaikan permasalahan utama pengguna melalui fitur inti yang telah disediakan. Dengan demikian, MVP berfungsi sebagai representasi awal dari sistem yang telah siap diuji secara nyata, dan menjadi fondasi awal sebelum pengembangan skala penuh dilakukan di tahap selanjutnya.

Minimum Viable Product (MVP) dinyatakan berhasil jika fitur-fitur utama yang dibangun dapat digunakan oleh pengguna sesuai fungsinya, memberikan manfaat nyata, serta mendapat tanggapan positif dari pengguna awal, meskipun produk belum sempurna atau final (Lortie et al., 2024)

2.11 Penelitian terdahulu

Penelitian ini mengintegrasikan berbagai referensi terkait dengan metode dan objek penelitian untuk memberikan batasan yang jelas pada metode dan sistem yang akan dikembangkan lebih lanjut. Hasil penelitian sebelumnya yang menjadi dasar pengembangan penelitian ini adalah sebagai berikut.

Tabel 2. 1 Penelitian terdahulu

No	Peneliti	Judul	Hasil
1	Mutammimah et al., (2024)	Pengembangan Aplikasi KAGANGA Berbasis Android untuk Mengenalkan Aksara Sunda di Sekolah Dasar	Aplikasi KAGANGA yang dibuat sebagai media pembelajaran aksara Sunda di sekolah dasar, efektif meningkatkan pemahaman siswa dengan peningkatan nilai dari 69,1 ke 88,9.
2	Wijaya et al., (2021)	Pegon-Gliph *Game* Pengenalan Dan Pembelajaran Arab Pegon Berbasis Android	Penelitian ini menghasilkan sebuah *game* Pegon-Gliph berbasis Android untuk pengenalan dan pembelajaran Arab Pegon. Hasil analisis menunjukkan bahwa *game* ini mampu menyediakan media pembelajaran yang menarik dan interaktif. *Game* ini memiliki empat fitur utama yaitu: materi, latihan, permainan, dan motivasi. Hasil pengujian *usability*

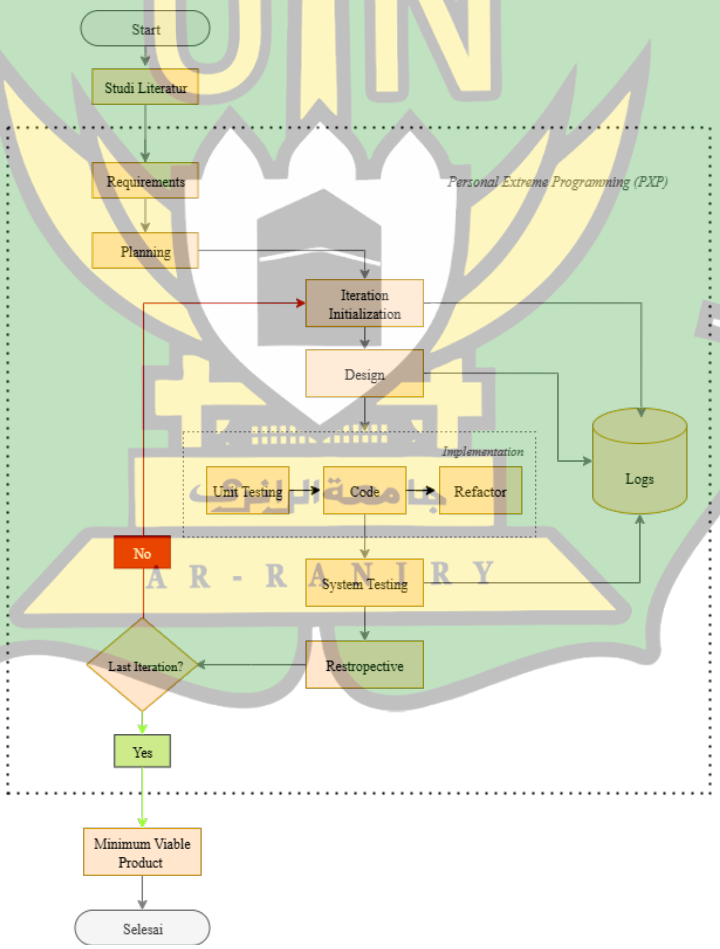
No	Peneliti	Judul	Hasil
			menunjukkan bahwa *game* ini memperoleh hasil "*excellent*" pada kategori *attractiveness*, *perspicuity*, *efficiency*, *dependability*, dan *stimulation*, serta "*good*" pada kategori *novelty*. *User* performa menunjukkan nilai rata-rata 95% pada pengujian *usability* menggunakan *task scenarios*.
3	Ardhy & Syahrobi (2021)	Rancang Bangun Aplikasi Pembelajaran Aksara Lampung Berbasis Android	Berdasarkan penelitian yang diuraikan dalam artikel, aplikasi Pembelajaran Aksara Lampung Berbasis Android ini dirancang untuk membantu siswa dan generasi muda masyarakat Lampung dalam mempelajari bentuk aksara Lampung. Aplikasi ini mempermudah pengguna untuk mengenali bentuk induk huruf, anak huruf, dan cara penulisannya dengan dilengkapi suara pengucapan.
4	Indah Puji Astuti et al., (2020)	Rancang Bangun Aplikasi *Mobile* Pengenalan Huruf Jawa (Aksara Jawa) Berbasis Android	Penelitian ini menghasilkan sebuah aplikasi *mobile* berbasis Android yang berfungsi sebagai media pembelajaran interaktif untuk mengenalkan aksara Jawa. Aplikasi ini memuat materi aksara lengkap beserta suara pelafalan dan kuis berbentuk *game* yang menarik. Pengembangan dilakukan dengan model *waterfall* dan diuji menggunakan metode *black box*, dengan hasil menunjukkan seluruh fitur berjalan sesuai rencana. Aplikasi ini telah tersedia di Play Store dan diharapkan dapat memudahkan pengguna dalam mempelajari

No	Peneliti	Judul	Hasil
			aksara Jawa secara menyenangkan.
5	Pujianti et al., (2019)	Perancangan Aplikasi Pembelajaran Arab Melayu Berbasis *Augmented Reality* Untuk Siswa Tingkat Sekolah Dasar	Penelitian ini menghasilkan sebuah aplikasi pembelajaran Arab Melayu berbasis *Augmented Reality* untuk siswa tingkat sekolah dasar. Aplikasi ini dirancang untuk membantu siswa mempelajari aksara Arab Melayu dengan cara yang lebih interaktif dan menarik, menggunakan teknologi *Augmented Reality* yang menampilkan objek 3D melalui *smartphone*.

BAB III
METODE PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini dilakukan dengan mengikuti tahapan yang diadaptasi dari proses pengembangan perangkat lunak menggunakan metode *Personal Extreme Programming* (PXP). Metode ini dipilih karena pengembangan aplikasi dilakukan oleh satu pengembang, sesuai dengan kondisi penulis yang bekerja secara individu. Selain itu, pendekatan *PXP* yang bersifat iteratif memberikan fleksibilitas yang lebih besar bagi pengembang untuk menyesuaikan diri dengan perubahan kebutuhan selama proses pengembangan aplikasi. Tahapan penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3. 1 Diagram Alur Penelitian

3.2 Studi Literatur

Pada tahap ini, penulis mengumpulkan berbagai data atau informasi yang terkait dengan bahasa aksara dan perancangan aplikasi yang relevan dengan penelitian ini. Sumber data mencakup jurnal, buku, penelitian terdahulu, artikel, dan literatur lainnya.

Penulis juga melakukan eksplorasi terhadap aplikasi-aplikasi belajar aksara yang telah dikembangkan dalam penelitian sebelumnya dan aplikasi serupa yang tersedia di toko aplikasi saat penelitian ini dilakukan, khususnya aksara Sunda, Bali, Arab Jawi, dan Lampung. Penulis meneliti apakah aplikasi tersebut memiliki penulisan yang sesuai kaidah, serta apakah aplikasi-aplikasi tersebut memiliki fitur seperti kamus, *scanner* aksara, *quiz,* belajar dasar aksara dan juga fitur belajar sejarah.

Melalui studi literatur ini, penulis memperoleh pemahaman yang lebih mendalam mengenai kebutuhan pengguna, kekuatan dan kelemahan dari aplikasi pembelajaran aksara yang telah ada, serta peluang inovasi yang dapat diimplementasikan dalam pengembangan aplikasi ALEA.

Analisis terhadap kekurangan fungsional dan kurangnya integrasi fitur pada aplikasi yang telah tersedia menjadi dasar pertimbangan dalam merancang solusi yang lebih komprehensif dan responsif terhadap kebutuhan pelestarian aksara daerah. Dengan demikian, studi literatur tidak hanya berfungsi sebagai landasan teoretis, tetapi juga sebagai pijakan strategis dalam menentukan arah pengembangan aplikasi yang tepat guna.

3.3 Metode Pengembangan Aplikasi

Metode *Personal Extreme Programming* (PXP) mencakup beberapa tahapan yang telah dijelaskan dalam BAB II. Sebagian besar tahapan bersifat iteratif, kecuali tahap *requirements* dan *planning*, yang ditetapkan untuk keseluruhan proses pengembangan. Jika ada perubahan pada *requirements*, maka *planning* juga harus di ubah sesuai kebutuhan. Proses iterasi dimulai dengan *iteration initialization* dan berakhir pada *retrospective*. Selama siklus PXP, pengembang juga mendokumentasikan *log file* yang berisi informasi seperti perencanaan tugas, durasi pengerjaan, serta detail kesalahan yang ditemukan.

3.3.1 *Requirements*

Penulis pertama mengidentifikasi kebutuhan atau persyaratan yang diperlukan dalam pengembangan aplikasi sesuai dengan langkah pengembangan dalam metode *Personal Extreme Programming* (PXP), termasuk kebutuhan fungsional dan non-fungsional.

1. Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fungsi dan fitur yang akan disediakan oleh aplikasi. Untuk aplikasi ALEA berbasis *mobile* ini, sebagian besar kebutuhan fungsional diadopsi dari fitur aplikasi belajar aksara yang sudah ada di toko aplikasi, serta dari hasil penelitian sebelumnya. Namun, dalam penelitian ini dilakukan beberapa peningkatan, seperti peningkatan antarmuka pengguna dan penambahan fitur baru. 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 dijelaskan fitur-fitur yang akan dikembangkan beserta rincian *user stories* dari masing-masing fitur.

Tabel 3. 1 Fitur Aplikasi

No	Fitur	Deskripsi
1	*Main*	Fitur ini berfungsi sebagai tampilan awal yang menyediakan *shortcut* langsung menuju fitur utama lainnya pada aplikasi, seperti *Scanner* Aksara, Kamus, Kuis, dan Sejarah dan Materi Aksara
2	*Dictionary*	Fitur ini akan menangani kebutuhan terkait menampilkan kamus aksara Sunda, Bali, Arab Jawi, dan Lampung, serta Detail dari setiap aksara pada aplikasi ALEA.
3	*Quiz*	Fitur ini memungkinkan pengguna menguji pemahaman mereka tentang aksara Sunda, Bali, Arab Jawi, dan Lampung. Pengguna akan diberikan pertanyaan terkait aksara, dan setelah menjawab, mereka akan menerima umpan balik langsung mengenai jawaban yang benar atau salah.

No	Fitur	Deskripsi
4	*History*	Fitur ini akan menampilkan informasi tentang sejarah aksara Sunda, Bali, Arab Jawi, dan Lampung. Pengguna dapat mempelajari asal-usul dan perkembangan setiap aksara serta perannya dalam budaya.
5	*Scanner*	Fitur ini memungkinkan pengguna mengambil gambar aksara Sunda, Bali, Arab Jawi, dan Lampung, lalu menerjemahkannya secara otomatis ke dalam teks bahasa Indonesia. Pengguna dapat memanfaatkan gambar dari kamera atau galeri perangkat untuk memindai aksara dan mendapatkan hasil terjemahan dengan bahasa Indonesia.
6	Materi	Fitur ini memungkinkan pengguna mempelajari dasar-dasar penggunaan aksara, bentuk aksara, cara penulisan, pelafalan, serta aturan penggunaan masing-masing aksara.
7	*Offline Mode*	Fitur ini memungkinkan pengguna untuk dapat menggunakan aplikasi walaupun sedang *offline*.
8	*Search*	Fitur ini akan menangani kebutuhan terkait pencarian konten dalam aplikasi, mencakup pencarian entri pada fitur kamus, pencarian artikel sejarah, serta pencarian materi pembelajaran aksara. Dengan fitur ini, pengguna dapat dengan mudah menemukan data yang dibutuhkan tanpa harus menelusuri satu per satu daftar yang tersedia.
9	Filter	Fitur ini dirancang untuk menyaring item pada Kamus Aksara berdasarkan klasifikasi struktural dari masing-masing aksara yang ditampilkan dalam aplikasi, seperti huruf dasar abjad, kata pasangan dan lainnya.

Berikut *user stories* dari kebutuhan fitur yang telah dikategorikan dapat dilihat pada Tabel 3.2. *User stories* ini merupakan hasil identifikasi kebutuhan fungsional pengguna terhadap aplikasi ALEA, yang selanjutnya menjadi dasar penentuan prioritas pengembangan.

Tabel 3. 2 *User Story*

No	Fitur	Judul *User Story*	*User Story*
1	*Main*	Beranda	Sebagai *user*, saya ingin melihat tampilan utama yang berisi *shortcut* ke fitur-fitur utama aplikasi.
2	*Dictionary*	Kamus Aksara	saya ingin mengakses kamus aksara Sunda, Bali, Arab Jawi, dan Lampung, sehingga saya dapat mengetahui arti dan cara pengucapan dari setiap aksara yang tersedia dalam aplikasi ALEA.
3	*Quiz*	Kuis Aksara	Sebagai *user*, saya ingin menguji pengetahuan saya tentang aksara dan mengetahui apakah jawaban saya benar atau salah.
4	*History*	Sejarah Aksara	Sebagai *user*, saya ingin mempelajari sejarah aksara yang tersedia dan juga relevansi buayanya, sehingga saya dapat memahami asal-usul dan perkembangannya.
5	*Scanner*	*Scanner* Melalui Kamera	Sebagai *user*, saya ingin memindai aksara secara langsung dari kamera agar aplikasi dapat menerjemahkannya ke teks bahasa Indonesia.
		Scanner Melalui Galeri	Sebagai *user*, saya ingin memindai aksara dari gambar di galeri agar aplikasi dapat menerjemahkannya ke teks bahasa Indonesia.
6	Materi	Materi Aksara	Sebagai *user*, saya ingin hal-hal seputar aksara seperti aturan penulisan aksara Sunda, Bali, Arab Jawi, dan Lampung, sehingga saya dapat memahami dasar penggunaannya.
7	*Offline Mode*	*Offline Mode*	Sebagai *user*, saya ingin agar aplikasi yang saya gunakan dapat digunakan jika tidak ada internet
8	*Search*	Pencarian Konten	Sebagai *user*, saya ingin mencari konten kamus, sejarah, atau materi belajar aksara menggunakan kata kunci tertentu, agar dapat menemukan informasi yang saya butuhkan dengan cepat.
9	Filter	Filter	Sebagai pengguna, saya ingin dapat menyaring kamus dan *quiz* berdasarkan klasifikasi struktural, sehingga saya dapat mempelajari jenis aksara tertentu secara lebih terfokus dan terstruktur.

2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menjelaskan batasan terkait fungsi atau fitur yang disediakan oleh aplikasi. Batasan-batasan ini mencakup perangkat lunak dan perangkat keras yang digunakan dalam pengembangan aplikasi. Dalam penelitian ini, perangkat lunak yang digunakan untuk mengembangkan aplikasi Aksara *Learning App* (ALEA) dapat dilihat pada Tabel 3.3.

Tabel 3. 3 Perangkat Lunak

No	Nama	Versi
1	Android Studio	2024.1.1
2	Kotlin	2.0.21
3	Postman	11.17.0
5	Github	-
6	CameraX	1.5.0-alpha01
7	Figma	124.4.7

Perangkat keras yang digunakan dalam pengembangan aplikasi Aksara *Learning App* (ALEA) pada penelitian ini dapat dilihat pada Tabel 3.4.

Tabel 3. 4 Perangkat Keras

No	Nama	Versi
1	Laptop	Acer Nitro 5
2	*Operation System*	Windows 11
3	CPU	Ryzen 7 5800H
5	GPU	RTX 3060
6	RAM	16 GB

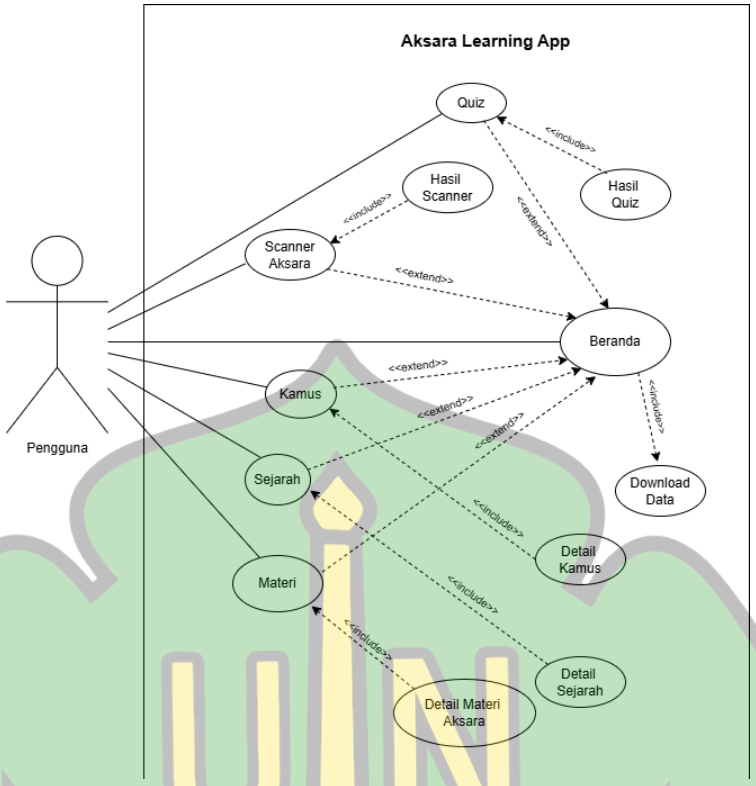
3.3.2 *Planning*

Pada tahap ini, penulis menetapkan estimasi waktu pengerjaan dan prioritas dari setiap *user story* dengan nilai prioritas dibagi menjadi tiga kategori yaitu:

1. Nilai 1, Produk tidak dapat diluncurkan tanpa fitur ini.
2. Nilai 2, Produk sebaiknya memiliki fitur ini, tetapi tidak harus segera diselesaikan.
3. Nilai 3, Fitur bersifat opsional dan bergantung pada sumber daya, waktu, dan risiko.

Selanjutnya, penulis menyusun rencana iterasi dengan menentukan *user story* yang akan dikerjakan pada setiap iterasi dan berapa estimasi waktu yang dibutuhkan. Estimasi waktu pengerjaan diberikan dalam bentuk *story point*, 1 *story point* setara dengan 4 jam. Penulis juga selanjutnya menentukan risiko *user story* yang dibagi dalam tiga kategori yaitu: risiko tinggi, risiko sedang, dan risiko rendah. Setelah menentukan estimasi waktu dalam *story point*, perencanaan iterasi dilakukan dengan menentukan *velocity*, yaitu jumlah *story point* yang dapat diselesaikan dalam satu iterasi. Penulis menentukan maksimum *velocity* sebesar 25 *story point* per iterasi. Dengan demikian, setiap iterasi dapat diselesaikan dalam 12 s.d. 13 hari dengan 8 jam kerja per hari. *User story* dengan prioritas tertinggi ditempatkan pada iterasi awal.

Prioritas suatu *user story*, beserta jumlah *story point* dan tingkat risikonya, ditentukan secara subjektif oleh penulis berdasarkan *use case* diagram yang dapat dilihat pada Gambar 3.2.



Gambar 3. 2 *Use Case* Diagram ALEA

Dalam diagram tersebut, terdapat satu aktor utama, yaitu *user* atau pengguna dari aplikasi. Aktor ini berinteraksi langsung dengan sejumlah fitur utama, yaitu:

a. Beranda, yaitu fitur *landing page* ketika pengguna pertama kali membuka aplikasi.
b. *Scanner*, yaitu fitur untuk memindai gambar aksara dari kamera atau galeri.
c. Kamus, untuk menampilkan daftar aksara kamus bahasa aksara yang tersedia.
d. Kuis, sebagai sarana pembelajaran dan evaluasi pengetahuan pengguna.
e. Sejarah, untuk yaitu fitur untuk mempelajari hal-hal seputar latar belakang budaya dan sejarah setiap aksara
f. Materi, yaitu fitur untuk mempelajari hal-hal seputar aksara seperti cara baca dan aturan penulisan aksara secara mandiri melalui materi pembelajaran yang tersedia dalam aplikasi.

Pada Tabel 3.5 disajikan perencanaan iterasi dalam penelitian ini. Berdasarkan tabel tersebut, dibutuhkan 5 iterasi dengan jumlah total 118 *story point*, untuk menyelesaikan seluruh *user story* yang tersedia.

Tabel 3. 5 Perencanaan iterasi

Iterasi	***User Story***	**Prioritas**	**Risiko**	***Story Point***
Iterasi 1	Beranda	1	Tinggi	20
	Velocity			**20**
Iterasi 2	Kuis Aksara	1	Tinggi	15
	Materi Aksara	1	Tinggi	10
	Velocity			**25**
Iterasi 3	*Scanner* Melalui Kamera	1	Tinggi	12
	Scanner Melalui Galeri	1	Tinggi	12
	Velocity			**24**
Iterasi 4	Kamus Aksara	1	Tinggi	15
	Pencarian Konten	3	Rendah	10
	Velocity			**25**
Iterasi 5	Sejarah Aksara	1	Sedang	10
	Offline Mode	3	Rendah	7
	Filter	3	Rendah	7
	Velocity			**24**

3.3.3 *Iteration Inialization*

Tahap *Iteration Initialization* menandai permulaan dari setiap siklus iterasi dalam proses pengembangan sistem. Lamanya tiap iterasi dapat bervariasi, tergantung pada tingkat kompleksitas dari fitur dan *user story* yang sedang dikerjakan. Pada tahap ini, proses dimulai dengan mengerjakan iterasi sesuai urutan yang telah ditetapkan berdasarkan prioritasnya. Pendekatan ini bertujuan untuk

memastikan bahwa fitur dan *user story* yang paling krusial dan mendesak akan diselesaikan terlebih dahulu. Dengan memulai iterasi berdasarkan urutan prioritas tersebut, fokus pengembangan diarahkan pada penyelesaian komponen-komponen yang memiliki relevansi tinggi dan kontribusi besar terhadap kemajuan sistem.

Setiap kali iterasi dilakukan, perubahan dan penyesuaian yang diterapkan pada aplikasi akan disimpan ke dalam *logs*. Setiap log mencatat Detail iterasi, termasuk perubahan yang dilakukan, serta hasil dari pengujian yang berkaitan dengan iterasi tersebut. Dengan mencatat setiap iterasi ke dalam *logs*, pengembang dapat melacak perkembangan aplikasi, serta memudahkan identifikasi dan penyelesaian masalah jika terjadi kendala pada iterasi berikutnya.

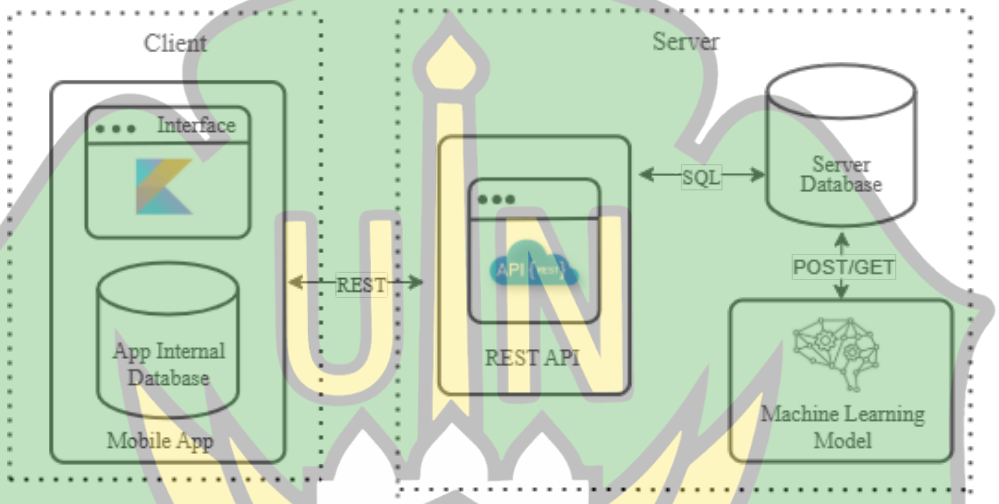
3.3.4 *Design*

Pada tahap ini, penulis merancang desain antarmuka pengguna aplikasi serta memodelkan sistem yang akan diimplementasikan selama proses iterasi. Desain antarmuka dan pemodelan sistem dibuat dengan tujuan untuk memenuhi *requirements* yang berasal dari *user story*, tanpa mencoba memprediksi kebutuhan masa depan. Pendekatan ini memastikan fokus pengembangan tetap pada kebutuhan yang telah ditentukan, sesuai dengan prinsip iteratif yang digunakan.

1. Arsitektur Aplikasi

Aplikasi yang dirancang pada penelitian ini menerapkan gaya arsitektur *Representational State Transfer* (REST). Salah satu prinsip utama REST adalah penerapan pola desain *client-server*, yang berarti tugas pada *client* dan *server* dipisahkan. Pada sisi aplikasi, ALEA menggunakan Kotlin sebagai bahasa pemrograman untuk menangani antarmuka pengguna pada sisi *client*. Data diproses melalui API REST yang berperan sebagai penghubung antara aplikasi *mobile* dan *server*. Pada sisi *server*, API berinteraksi dengan *database* melalui perintah SQL untuk penyimpanan dan pengolahan data. Selain itu, terdapat integrasi antara *server* dengan model *Machine Learning* yang berinteraksi dengan *database* untuk melakukan operasi POST/GET sesuai kebutuhan. Pada penelitian ini penulis hanya akan mengembangkan aplikasi dari sisi *client* atau *user interface* serta penyimpanan internal, dan tidak mengembangkan bagian *server* atau sisi *backend*.

Server atau *backend*, yang digunakan berasal dari penelitian Hanafi Akbar (2025) yang berjudul "*Desain dan Implementasi Arsitektur Backend Berbasis Google Cloud Pada Aplikasi ALEA Untuk Pembelajaran Aksara*", sedangkan *machine learning* yang digunakan oleh aplikasi ini berasal dari penelitian Arief Fathin Abrar (2025) yang berjudul "*Implementasi Arsitektur YOLOv8 Untuk Mengklasifikasikan Aksara Pada Aksara Learning App*". Pada Gambar 3.3 dapat dilihat gambaran dari arsitektur aplikasi yang akan dirancang.



Gambar 3. 3 Arsitektur Aplikasi

2. *Mid-fidelity Wireframe*

Mid-fidelity wireframe adalah sketsa atau kerangka gambar yang menampilkan struktur desain aplikasi secara lebih rinci, termasuk letak dan interaksi antar fitur utama. Pada *wireframe* jenis ini, elemen pada aplikasi sudah lebih jelas dibandingkan *low-fidelity wireframe*, tetapi belum melibatkan elemen visual penuh seperti warna dan gambar yang kompleks dan final.

a. *Main Screen*

Main Screen adalah halaman utama yang ditampilkan ketika pengguna pertama kali membuka aplikasi dan akan berfungsi sebagai pusat navigasi utama aplikasi. Pengguna dapat memilih fitur utama yang ingin mereka akses menggunakan *bottom navigation* dan *shortcut* yang terdapat pada halaman ini.

Gambar 3. 4 *Main Screen – Wireframe*

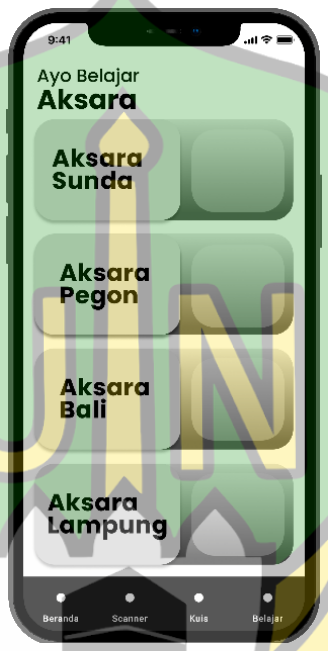
b. Dialog Status *Screen*

Halaman ini digunakan untuk menampilkan dialog ketika aplikasi mencoba mengambil data dari *server*. Halaman ini bertujuan untuk menampilkan status terkini aplikasi ketika sedang mengambil data dari *server*

Gambar 3. 5 Dialog Status *Screen* – *Wireframe*

c. Belajar *Screen*

Halaman ini berfungsi untuk mengategorikan fitur sejarah, materi, dan juga kamus ke masing-masing jenis aksara yang tersedia. Pengguna dapat memilih aksara dari daftar untuk melihat fitur sejarah, kamus, dan materi dari masing-masing aksara tersebut .



Gambar 3. 6 Belajar *Screen – Wireframe*

d. *Choose* Belajar *Feature Screen*

Halaman ini menampilkan daftar fitur sejarah, materi, dan juga kamus pada masing-masing jenis aksara yang tersedia. Pengguna dapat memilih fitur belajar mana yang akan digunakan untuk melihat Detail dari fitur tersebut.

Gambar 3. 7 *Choose* Belajar *Feature Screen– Wireframe*

e. *History Screen*

Halaman ini menampilkan daftar sejarah aksara yang tersedia dalam aplikasi berdasarkan kategori yang dipilih pada halaman Belajar *Screen*. Fitur ini berfungsi untuk memberikan informasi mengenai perkembangan historis dan relevansi budaya dari aksara Sunda, Bali, Arab Jawi, dan Lampung.

Gambar 3. 8 *History Screen – Wireframe*

f. Detail *History Screen*

Halaman Detail *History* merupakan antarmuka yang dirancang untuk menampilkan informasi secara menyeluruh mengenai *item* sejarah aksara yang telah dipilih oleh pengguna pada *History Screen*. Halaman ini bertujuan untuk memberikan pemahaman yang lebih mendalam terhadap suatu entri sejarah, dengan menyajikan elemen-elemen informasi penting seperti judul, uraian deskriptif, serta gambar pendukung yang merepresentasikan konten sejarah tersebut.

Gambar 3. 9 Detail *History Screen – Wireframe*

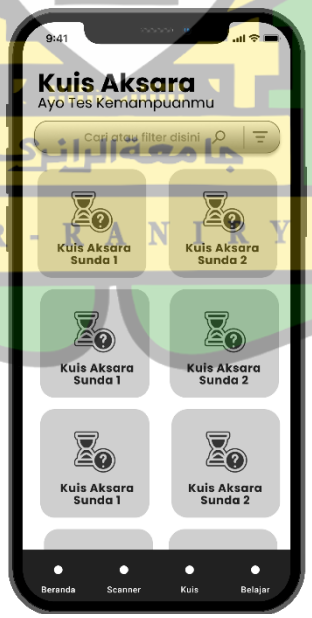
g. *Dictionary Screen*

Halaman *Dictionary* merupakan antarmuka yang menyajikan informasi secara rinci mengenai kamus berdasarkan jenis aksara yang telah dipilih oleh pengguna dari Belajar *Screen*. Tujuan utama dari halaman ini adalah untuk memperluas pemahaman pengguna terhadap aksara tertentu dengan menampilkan data yang lebih lengkap dan terstruktur. Informasi yang disajikan dalam halaman ini meliputi bentuk aksara, arti dari aksara tersebut, cara pengucapan, serta contoh penggunaannya dalam konteks tertentu.

Gambar 3. 10 *Dictionary Screen – Wireframe*

h. *Quiz Screen*

Halaman *Quiz* dirancang untuk menampilkan daftar kuis yang tersedia berdasarkan kategori aksara yang tersedia dan juga kuis campuran antar aksara. Setiap *item* dalam daftar mewakili satu paket kuis yang dapat dipilih oleh pengguna.



Gambar 3. 11 *Quiz Screen – Wireframe*

i. *Play Quiz Screen*

Halaman ini akan menampilkan setiap pertanyaan *Quiz* yang dipilih secara satu per satu, lengkap dengan opsi jawaban pilihan ganda yang dapat dipilih oleh pengguna. Setelah seluruh pertanyaan selesai dijawab, sistem akan menampilkan hasil akhir kuis yang mencakup total skor yang diperoleh pengguna.

Gambar 3. 12 *Play Quiz Screen – Wireframe*

j. *Request Permission* Dialog *Screen*

Halaman ini digunakan untuk memunculkan dialog yang dapat memungkinkan pengguna untuk meminta izin yang diperlukan untuk menjalankan *scanner* seperti izin penggunaan kamera dan juga izin membaca *gallery* dan *file manager*.

Gambar 3. 13 *Request Permission* Dialog *Screen - Wireframe*

k. *Scanner Screen*

Halaman *Scanner* memungkinkan pengguna untuk memindai teks aksara Sunda, Bali, Arab Jawi, dan Lampung menggunakan gambar dari galeri maupun gambar yang diambil langsung menggunakan kamera perangkat mereka.

Gambar 3. 14 *Scanner Screen – Wireframe*

l. *Scanner Result Screen*

Scanner Result Screen berfungsi untuk menampilkan hasil terjemahan aksara yang berhasil dipindai pada *Scanner Screen*.

Gambar 3. 15 *Scanner Result Screen – Wireframe*

m. Materi *Screen*

Halaman ini menampilkan daftar materi aksara yang tersedia dalam aplikasi seperti cara baca dan aturan penulisan aksara sesuai dengan jenis aksara yang dipilih pada Belajar *Screen*.

Gambar 3. 16 Materi *Screen – Wireframe*

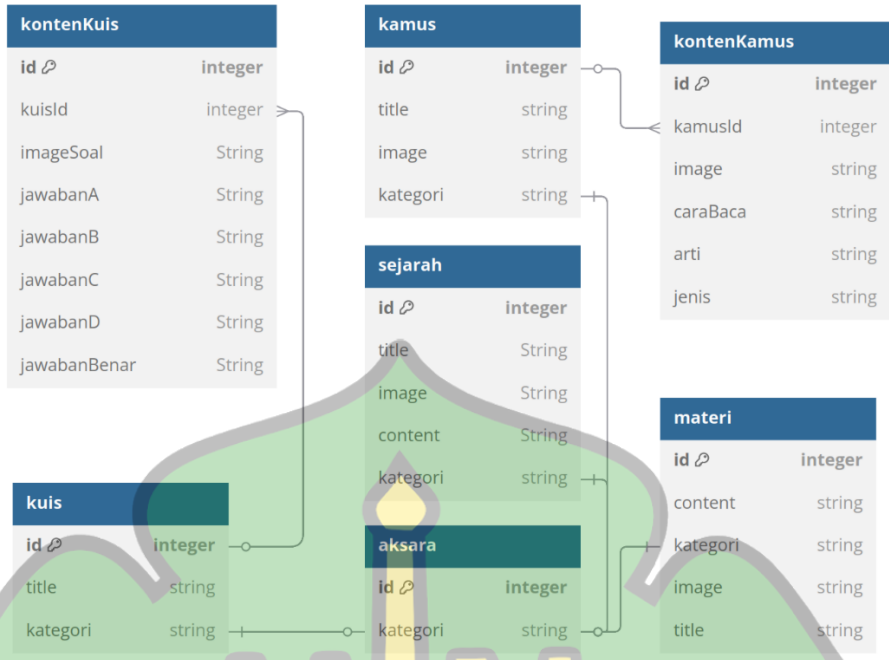
n. Detail Materi *Screen*

Halaman ini merupakan antarmuka yang dirancang untuk menampilkan informasi secara menyeluruh mengenai *item* materi yang telah dipilih oleh pengguna pada Materi *Screen*. Halaman ini bertujuan untuk memberikan pemahaman yang lebih mendalam terhadap materi aksara, dengan menyajikan elemen-elemen informasi penting seperti judul, uraian deskriptif, serta gambar pendukung yang diperlukan.

Gambar 3. 17 Detail Materi *Screen – Wireframe*

3. *App Internal Database Schema*

Database schema adalah rancangan atau cetak biru yang memudahkan pengembang dalam memvisualisasikan struktur data pada *database* relasional. Ini meliputi elemen-elemen utama seperti nama tabel, kolom atau *field*, tipe data untuk setiap kolom, serta hubungan antara tabel. Pada Gambar 3.18 dapat dilihat *database schema* untuk aplikasi Aksara *Learning App*.



Gambar 3. 18 *Internal Database Schema*

Dari hasil perancangan *database schema* didapat tujuh tabel dengan rincian yang dapat dilihat pada tabel dibawah.

Tabel 3. 6 *Table* Kuis

Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
title	string	
kategori	string	

Tabel 3. 7 *Table* aksara

Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
kategori	string	

Tabel 3. 8 *Table* kontenKuis

Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
kuisId	integer	*Foreign Key*
*image*Soal	string	
jawabanA	string	
jawabanB	string	
jawabanC	string	
jawabanD	string	
jawabanBenar	string	

Tabel 3. 9 *Table* Kamus

Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
title	string	
image	string	
kategori	string	

Tabel 3. 10 *Table* kontenKamus

Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
kamusId	integer	*Foreign Key*
image	string	
caraBaca	string	
arti	string	
jenis	string	

Tabel 3. 11 *Table* materi

Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
content	string	
kategori	string	
image	string	
title	string	

Tabel 3. 12 *Table* sejarah

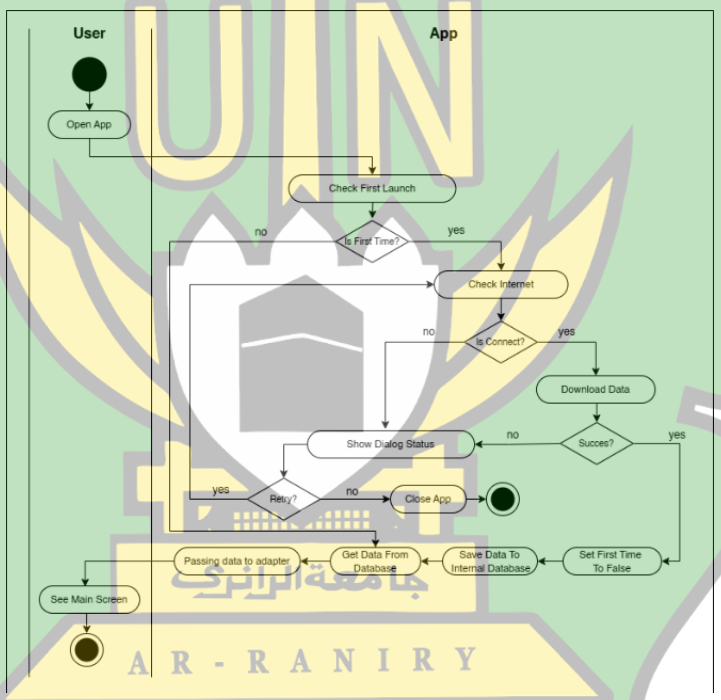
Field Name	***Data Type***	***Description***
id	integer	*Primary Key*
title	string	
image	string	
konten	string	
kategori	string	

4. *Activity* Diagram

Untuk memastikan bahwa aplikasi yang dikembangkan memenuhi *requirement* yang telah ditetapkan sebelumnya dalam bentuk *user story* yang dapat dilihat pada Tabel 3.2 maka penulis akan terlebih dahulu merincikan *Activity* Diagram dari fitur yang akan dibuat. *Activity* Diagram digunakan untuk memvisualisasikan alur proses serta urutan aktivitas yang terjadi ketika pengguna berinteraksi dengan aplikasi. Diagram ini merepresentasikan langkah-langkah operasional dari setiap fitur yang telah dirancang di dalam aplikasi Aksara *Learning App.*

a. *Main Screen*

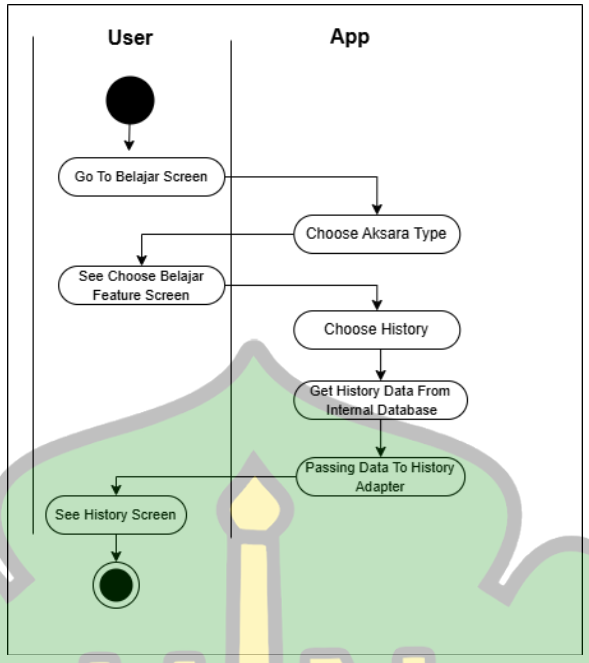
Pada Gambar 3.19 dapat dilihat urutan aktivitas yang terjadi ketika pengguna membuka aplikasi ALEA, aplikasi akan menampilkan halaman *Main Screen* yang dalam prosesnya aplikasi akan melakukan pengecekan dan validasi status pengguna apakah ini merupakan kali pertama aplikasi dijalankan atau bukan. Apabila pengguna telah membuka aplikasi sebelumnya dan seluruh data yang diperlukan telah berhasil diunduh, maka sistem akan langsung menampilkan halaman beranda. Sebaliknya, jika pengguna baru pertama kali membuka aplikasi dan koneksi internet ada, maka aplikasi akan mencoba untuk mengunduh data yang dibutuhkan dari *server*, jika terjadi kesalahan maka akan ditampilkan dialog informasi yang memuat status terkini aplikasi.



Gambar 3. 19 *Activity* Diagram *Main Screen*

b. *History Screen*

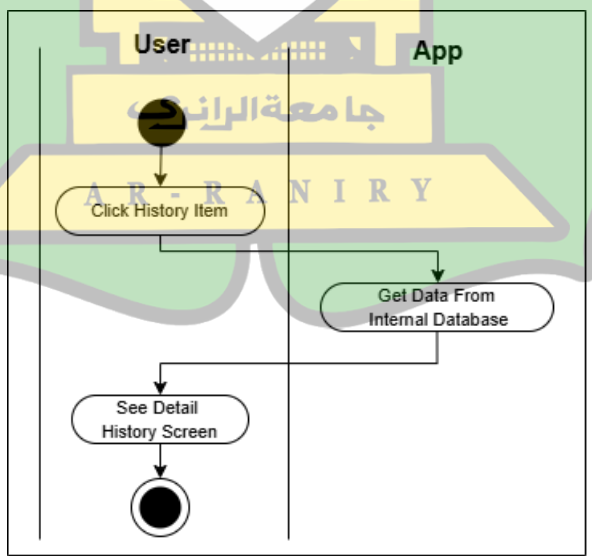
Urutan proses aktivitas ketika pengguna akan membuka *History Screen* dan melihat entri sejarah yang tersedia pada aplikasi melalui Belajar *Screen* yang tersedia dalam aplikasi dapat dilihat pada gambar 3.20.



Gambar 3. 20 *Activity* Diagram *History Screen*

c. Detail *History Screen*

Urutan aktivitas yang terjadi ketika pengguna mengakses Detail *History Screen*, melalui *History Screen* dengan cara memilih salah satu daftar yang tersedia dapat dilihat pada gambar 3.21.

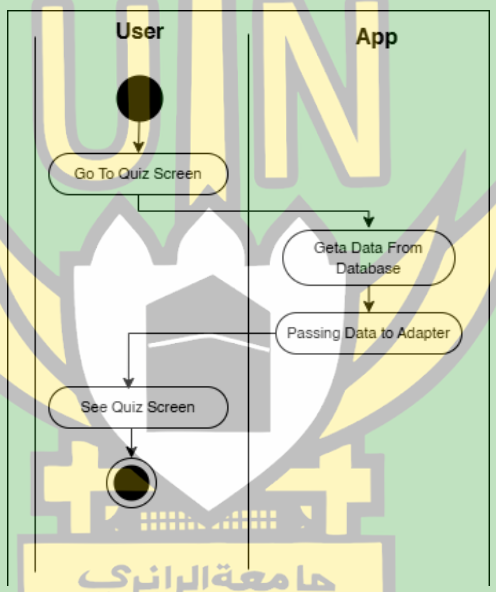


Gambar 3. 21 *Activity* Diagram Detail *History Screen*

Saat pengguna memilih salah satu *item*, sistem akan meneruskan data yang terkait dari *item* tersebut ke Detail *History Screen*. Data yang telah diteruskan kemudian digunakan untuk menampilkan informasi secara lengkap sesuai dengan *item history* yang dipilih oleh pengguna.

d. *Quiz Screen*

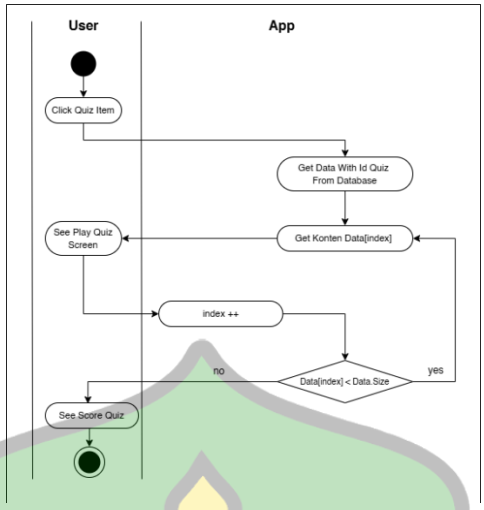
Pada Gambar 3.22 ditunjukkan urutan aktivitas yang terjadi ketika pengguna mengakses halaman kuis melalui menu *bottom navigation*. Ketika pengguna memilih menu tersebut, sistem akan melakukan proses pengambilan data kuis dari basis data internal. Data yang diperoleh kemudian diteruskan ke komponen adapter untuk diproses dan ditampilkan dalam bentuk daftar pada antarmuka halaman kuis.



Gambar 3. 22 *Activity* Diagram *Quiz Screen*

e. *Play Quiz Screen*

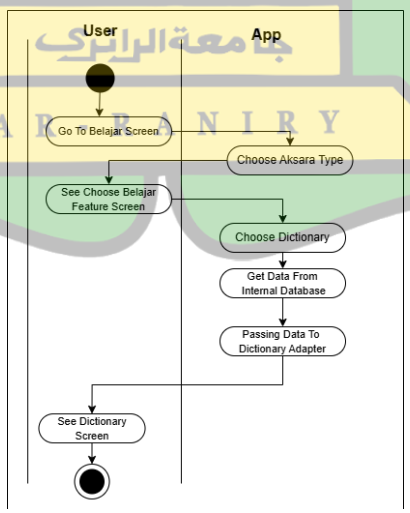
Rincian urutan aktivitas pada *Play Quiz Screen* dimulai ketika pengguna menekan salah satu *item quiz* yang ditampilkan pada *Quiz Screen*, maupun pada *shortcut* yang terletak pada *Main Screen.* Ketika *item* ditekan maka *item* akan mengambil data Konten Kuis di *database* untuk diteruskan ke *Play Quiz Screen* untuk ditampilkan sebagaimana dapat dilihat pada Gambar 3.23.



Gambar 3. 23 *Activity* Diagram *Play Quiz Screen*

f. *Dictionary Screen*

Aktivitas diawali saat pengguna memilih jenis aksara melalui Belajar *Screen*. Setelah memilih jenis aksara maka pengguna akan melihat *Choose* Belajar *Feature Screen*, saat pengguna memilih fitur kamus yang tersedia pada halaman tersebut maka sistem kemudian akan mengambil data dari *database* berdasarkan aksara yang dipilih dan meneruskannya ke adapter untuk ditampilkan dalam bentuk daftar kamus yang tersedia pada aksara tersebut. Detail aktivitas pada *Dictionary Screen* dapat dilihat pada Gambar 3.24.

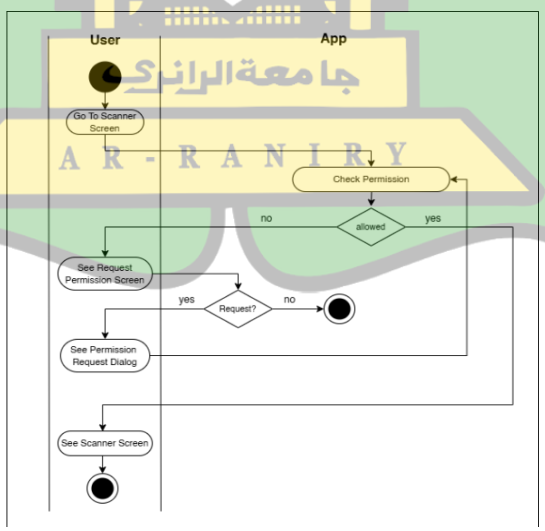


Gambar 3. 24 *Activity* Diagram *Dictionary Screen*

g. *Scanner Screen*

Gambar 3.25. merupakan gambaran urutan aktivitas menggunakan *scanner* aksara untuk menerjemahkan aksara secara otomatis baik itu melalui kamera secara langsung maupun menggunakan gambar dari *gallery photo.* Adapun rincian prosesnya yaitu:

- Pertama, pengguna harus menuju ke halaman *Scanner Screen*, baik menggunakan *bottom navigation* maupun menggunakan *shortcut* dari halaman *Main Screen.*
- Selanjutnya, aplikasi akan mengecek apakah semua izin untuk menggunakan *scanner* aksara telah diberikan atau belum. Jika aplikasi sudah memiliki izin yang memadai untuk menjalankan *scanner* aksara maka *Scanner Screen* akan di tampilkan, akan tetapi jika sebaliknya, maka pengguna akan melihat sebuah halaman yang meminta izin yang diperlukan untuk menggunakan *scanner* aksara.
- Pada halaman *Scanner Screen* pengguna dapat menerjemahkan aksara dengan cara menekan tombol kamera untuk mengambil gambar secara langsung.
- jika pengguna ingin menggunakan gambar yang sudah ada untuk diterjemahkan maka pengguna dapat menggunakan tombol *gallery* yang berada pada samping tombol kamera.

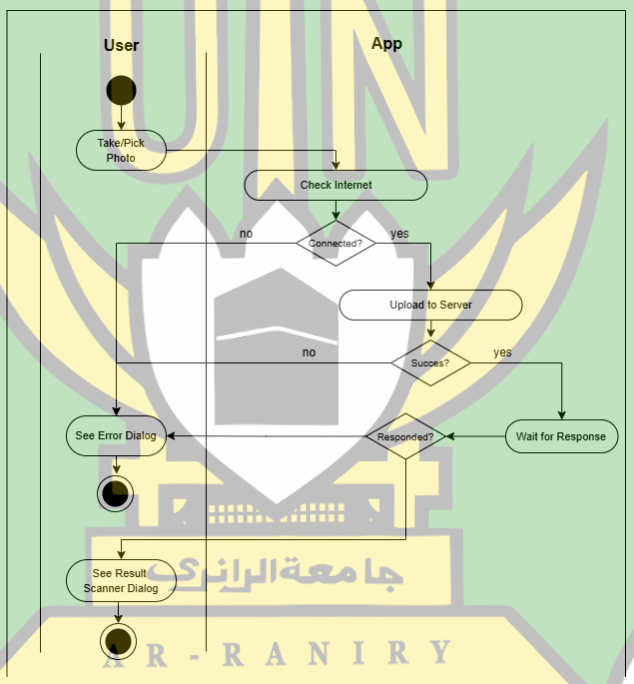


Gambar 3. 25 *Activity* Diagram *Scanner Screen*

h. *Scanner Result Screen*

Gambar 3.26 adalah gambaran dari aktivitas yang terjadi ketika pengguna mengambil gambar baik menggunakan kamera secara langsung maupun menggunakan gambar dari *gallery.* berikut adalah rincian prosesnya:

- Gambar yang di ambil melalui kamera ataupun di pilih pengguna dari *gallery* kemudian akan di *upload* ke *server* untuk diterjemahkan menggunakan *machine learning*.
- *Server* akan mengirimkan respons hasil dari gambar tersebut, respons tersebut bisa berupa respons *succes* maupun respon *error*.
- Aplikasi akan menampilkan dialog yang sesuai dengan respons dari *server*.

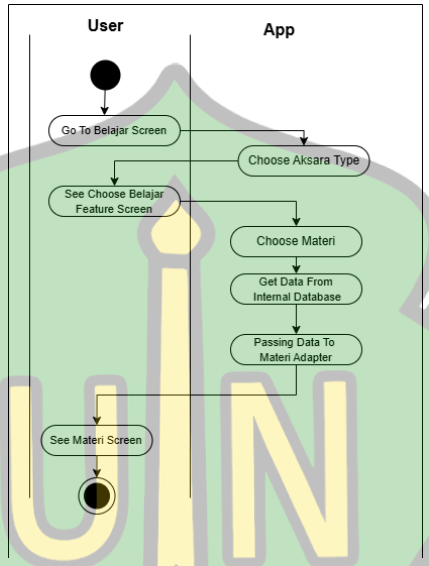


Gambar 3. 26 *Activity* Diagram *Scanner Result Screen*

i. Materi *Screen*

Pada Gambar 3.27 dapat dilihat urutan aktivitas yang terjadi ketika pengguna mengakses halaman Materi, aktivitas diawali saat pengguna memilih jenis aksara melalui Belajar *Screen*. Setelah memilih jenis aksara maka pengguna akan melihat

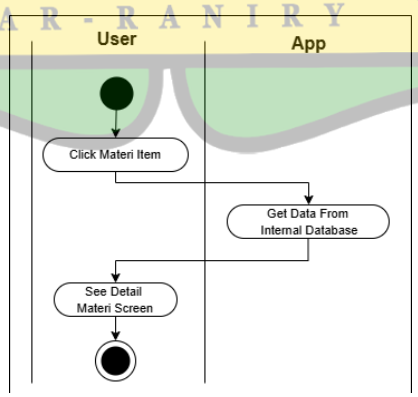
Choose Belajar *Feature Screen* lalu saat pengguna memilih fitur Materi yang tersedia pada halaman tersebut maka sistem kemudian akan mengambil data dari *database* berdasarkan aksara yang dipilih dan meneruskannya ke adapter untuk ditampilkan dalam bentuk daftar Materi Aksara yang tersedia pada aksara tersebut.



Gambar 3. 27 *Activity* Diagram Materi *Screen*

j. Detail Materi *Screen*

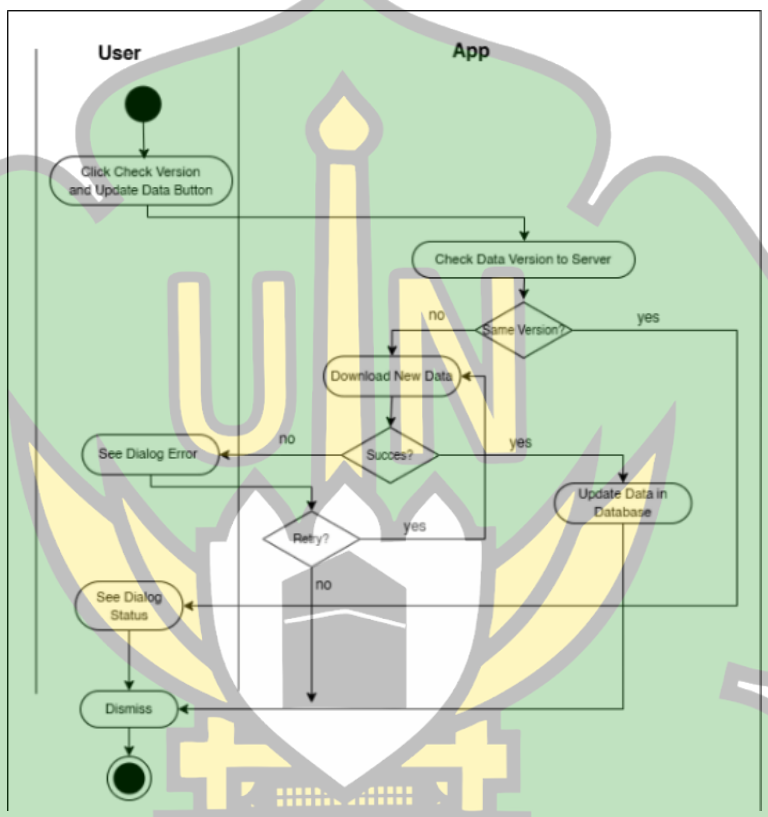
Urutan aktivitas yang terjadi ketika pengguna mengakses Detail Materi *Screen*, melalui Materi *Screen* dengan cara memilih salah satu daftar yang tersedia dapat dilihat pada gambar 3.28.



Gambar 3. 28 *Activity* Diagram Detail Materi *Screen*

k. *Check Version and Update* Data

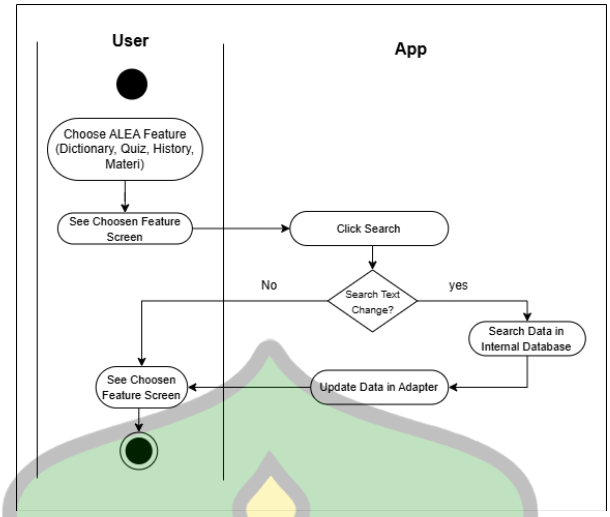
Pada Gambar 3.29 dapat dilihat urutan aktivitas yang terjadi ketika pengguna menekan tombol *update* data, aplikasi akan mengirimkan permintaan untuk memastikan versi data apakah sudah yang terbaru ataupun belum, jika versi data tidak merupakan versi terbaru maka aplikasi akan mengunduh data terkini untuk menggantikan data versi yang lama.



Gambar 3. 29 *Activity* Diagram *Check Version and Update* Data

l. *Search*

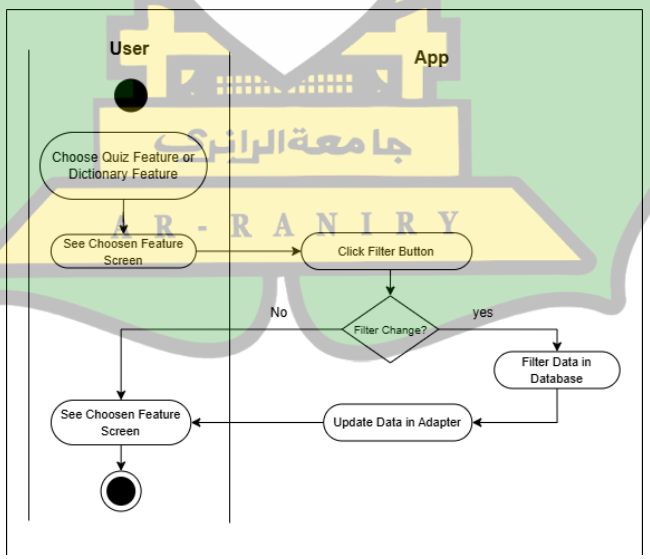
Gambar 3.30 memperlihatkan urutan aktivitas yang terjadi ketika pengguna ingin mencari konten yang tersedia di fitur *Dictionary*, *History*, Materi, dan *Quiz*. Ketika pengguna menekan *search* dan menuliskan kata di *column* pencarian maka aplikasi akan mengirimkan permintaan untuk mengecek apakah ada data yang sama dengan kata pencarian yang dimasukkan oleh pengguna, jika ada yang sama maka aplikasi akan memperbarui data adapter untuk di tampilkan kepada pengguna.



Gambar 3. 30 *Activity* Diagram *Search*

m. Filter

Gambar 3.31 memperlihatkan urutan aktivitas yang terjadi ketika pengguna melakukan filter konten yang tersedia di fitur *Dictionary* dan juga fitur *Quiz*. Ketika pengguna menekan tombol filter dan memilih jenis filter yang tersedia maka aplikasi akan mengirimkan permintaan untuk memperbarui data adapter dengan jenis filter yang dipilih untuk di tampilkan kepada pengguna.



Gambar 3. 31 *Activity* Diagram Filter

3.3.5 *Implementation*

Tahap ini merupakan fase saat kode pemrograman diimplementasikan. Penulis mulai mengimplementasikan antarmuka pengguna yang telah dirancang pada tahap desain. Fase ini terdiri dari tiga sub-tahap, yaitu: *unit testing*, *code generation*, dan *refactoring*. Dalam penelitian ini, karna penulis hanya bertanggung jawab pada sisi *client* maka implementasi hanya akan dilakukan pada sisi *frontend*, tanpa melibatkan pengembangan sisi *backend*.

Proses implementasi akan diawali dengan membuat unit testing, yang bertujuan untuk menguji setiap komponen logika program secara terpisah guna memastikan bahwa fungsionalitas dasar dari aplikasi telah berjalan sesuai dengan rancangan. Pengujian ini dilakukan sebelum penulisan kode program secara keseluruhan agar kesalahan logika dapat dideteksi lebih awal, sehingga proses pengembangan dapat berlangsung secara lebih efisien dan terkontrol.

Tahap selanjutnya adalah *code generation*, yaitu proses saat kode program aktual mulai ditulis berdasarkan rancangan sistem yang telah ditetapkan sebelumnya. Pada tahap ini, tampilan antarmuka aplikasi akan diimplementasikan menggunakan bahasa XML, sedangkan logika aplikasi dikembangkan dalam bahasa pemrograman Kotlin.

Setelah proses *code generation* selesai dilakukan, tahapan berikutnya adalah *refactoring*. *Refactoring* merupakan proses restrukturisasi kode program yang telah ditulis tanpa mengubah perilaku atau fungsionalitas yang sudah ada. Tujuan utama dari *refactoring* adalah meningkatkan kualitas internal kode, seperti keterbacaan, efisiensi, dan kemudahan dalam pemeliharaan serta pengembangan aplikasi di masa depan. Tahap refactoring ini dilakukan hanya jika dianggap perlu, berdasarkan evaluasi terhadap kualitas kode yang telah dihasilkan. Hal ini penting untuk memastikan bahwa kode aplikasi yang dihasilkan memiliki standar yang baik, bersih, efisien, dan mudah dikelola untuk pengembangan selanjutnya.

3.3.6 *System Testing*

System testing adalah tahap pengujian fungsionalitas secara keseluruhan terhadap semua fitur yang telah terintegrasi. *System testing* akan dilakukan oleh pengembang dengan pendekatan yang digunakan adalah *black-box testing*, yang menempatkan penguji dalam sudut pandang pengguna akhir yang fokusnya adalah menguji aplikasi secara *end-to-end*.

Pengujian dilakukan dengan menjalankan aplikasi baik itu pada perangkat fisik maupun emulator, untuk memastikan aliran aplikasi berjalan sesuai harapan tanpa memperhatikan Detail kode program. Pengujian ini mencakup bagaimana aplikasi merespons *input* pengguna, apakah *output* yang dihasilkan sesuai dengan yang diharapkan, serta apakah aplikasi berfungsi dengan baik di lingkungan nyata.

3.3.7 *Retrospective*

Tahap *retrospective* merupakan langkah penutup dalam setiap siklus iterasi yang bertujuan untuk mengevaluasi proses pengembangan yang telah dilaksanakan. Pada tahap ini, pengembang melakukan peninjauan terhadap pekerjaan yang telah diselesaikan, terutama dengan membandingkan antara estimasi waktu pengerjaan dengan waktu aktual yang dibutuhkan. Evaluasi ini mencakup penilaian terhadap kesesuaian hasil implementasi dengan tujuan yang telah dirumuskan dalam *user story*, serta efektivitas proses selama iterasi berlangsung.

Selain itu, tahap ini juga difokuskan pada identifikasi berbagai kendala atau hambatan yang mungkin terjadi selama iterasi, baik yang bersifat teknis maupun non-teknis. Pengembang mencatat dan menganalisis penyebab dari keterlambatan, kesalahan teknis, atau proses yang kurang efisien. Dengan mengetahui akar permasalahan yang dihadapi, pengembang dapat merumuskan solusi atau perbaikan yang tepat untuk menghindari terulangnya masalah yang sama pada iterasi berikutnya.

Manfaat tahap *retrospective* dalam bentuk perbaikan berkelanjutan terhadap proses pengembangan sistem. Evaluasi yang dilakukan secara reflektif ini membantu meningkatkan efisiensi kerja, memperbaiki alur pengembangan, dan memperkuat perencanaan pada siklus iterasi selanjutnya. Dengan demikian, kualitas produk dan proses kerja dapat terus ditingkatkan seiring berjalannya waktu.

3.4 Minimum Viable Product (MVP)

Minimum Viable Product (MVP) merupakan pendekatan yang berfokus pada pengembangan produk dengan menyediakan fitur-fitur paling esensial untuk pengguna awal. Dalam pengembangan aplikasi ALEA pendekatan MVP diterapkan guna memastikan produk keluaran penelitian ini memiliki fungsi utama yang benar-benar mampu memenuhi kebutuhan inti pengguna.

Pendekatan MVP ini memungkinkan aplikasi ALEA diuji langsung oleh pengguna untuk mendapatkan umpan balik yang konstruktif. Uji coba dilakukan untuk memperoleh umpan balik, guna mengevaluasi sejauh mana fitur utama aplikasi telah memenuhi kebutuhan pengguna, baik dari sisi kelayakan, kemudahan penggunaan, maupun kebermanfaatan. Dalam praktiknya, umpan balik terhadap MVP dapat diperoleh melalui berbagai metode, salah satunya adalah pemberian *rating* dari pengguna awal.

Pada penelitian ini, pemberian *rating* dilakukan menggunakan skala Likert. Skala Likert adalah metode penilaian yang digunakan untuk mengukur tingkat kesesuaian atau pendapat responden terhadap suatu pernyataan berdasarkan tingkatan tertentu. Skor diberikan dalam rentang nilai 1 hingga 5, dengan kriteria sebagaimana tercantum pada tabel berikut:

Tabel 3. 13 Skala Penilaian *Rating* Aplikasi

Skor	Kategori Penilaian	Keterangan Penilaian
1	Sangat Tidak Sesuai	Fitur tidak berfungsi, sulit digunakan, dan tidak memberi manfaat sama sekali
2	Tidak Sesuai	Fitur kurang berjalan dengan baik atau masih membingungkan
3	Cukup Sesuai	Fitur berjalan namun masih kurang nyaman atau belum optimal kegunaannya
4	Sesuai	Fitur berjalan baik, mudah digunakan, dan memberi manfaat yang jelas
5	Sangat Sesuai	Fitur sangat membantu, mudah diakses, dan sesuai dengan kebutuhan pengguna

BAB IV
HASIL DAN PEMBAHASAN

4.1 *Implementation*

Hasil dari tahap implementasi akan dituangkan pada poin ini beserta dengan rinciannya. Secara umum, tahapan implementasi dimulai dengan pembuatan *unit testing* yang dibuat berdasarkan *user story* untuk memastikan setiap fitur berfungsi dengan benar tanpa kesalahan, selanjutnya akan dilakukan proses *code generation*, pada tahap ini penulis akan membangun fitur yang telah dirincikan pada bab sebelumnya lalu dilanjutkan dengan mengonversi desain yang telah dirancang sebelumnya ke dalam bentuk kode program menggunakan Android Studio.

Dalam tahapan implementasi ini, proses *refactoring* tidak dilakukan. Hal ini disebabkan karena tidak ditemukan bagian yang perlu disederhanakan atau dioptimalkan ulang. Seluruh komponen logika dan struktur program telah dapat berjalan secara stabil tanpa indikasi duplikasi kode, kompleksitas berlebih, atau potensi kesalahan pemeliharaan di tahap selanjutnya.

4.2 *Unit Testing*

Unit testing yang dibuat difokuskan pada komponen-komponen logika inti yang menjadi penopang utama jalannya fungsionalitas aplikasi. Tujuan utama dari pengujian ini adalah memastikan bahwa setiap fungsi menghasilkan keluaran yang sesuai dengan ekspektasi logika yang telah ditentukan.

Unit testing dilakukan menggunakan bahasa pemrograman Kotlin, dengan memanfaatkan kombinasi *framework* JUnit sebagai *test runner*, dan Mockito sebagai *library* untuk membuat objek tiruan (*mock*). Dengan pendekatan ini, proses pengujian dapat dilakukan secara terisolasi dan dapat mereplikasi berbagai skenario tanpa perlu benar-benar terhubung dengan sumber data atau layanan jaringan secara langsung. Pengujian dilakukan terhadap beberapa fungsi inti berikut:

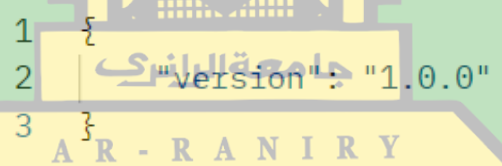
1. Fungsi *search*: memastikan bahwa hasil pencarian sesuai dengan kata kunci yang dimasukkan pengguna.
2. Fungsi filter: memverifikasi bahwa data dapat difilter berdasarkan kategori tertentu secara konsisten dan akurat.

3. Fitur *Quiz*: menguji logika evaluasi jawaban dan perhitungan skor, untuk memastikan bahwa sistem kuis dapat membedakan antara jawaban benar dan salah serta mengakumulasi nilai secara tepat.
4. *Parsing* dan validasi data dari API eksternal: memastikan bahwa data dalam format JSON yang diterima dari *backend* dapat dipetakan dengan benar ke dalam struktur model aplikasi.

Sebelum penulis menuliskan *unit testing parsing* dan validasi data dari API eksternal, penulis terlebih dahulu memastikan respons yang dikembalikan oleh server menggunakan alat bantu Postman. Respons selanjutnya di konversi ke dalam bentuk unit testing. Berikut adalah rincian respons yang di kembalikan oleh API yang digunakan :

1. *Check Version And Update Data*

Gambar 4.1 memperlihatkan respons yang di berikan oleh server saat melakukan proses pengecekan versi data. *Endpoint* API ini dapat diakses melalui *Uniform Resource Locator* (URL) dengan format https://{*domain*}/*version*. Untuk mengirim permintaan ke *endpoint* tersebut kita dapat metode HTTP GET. Saat permintaan berhasil, *server* mengembalikan respons berformat JSON yang berisi sebuah atribut *version* bertipe string yang memuat versi dari data API saat ini.



Gambar 4. 1 Respons API *Check Version And Update* Data

2. *Dictionary*

Gambar 4.2 memperlihatkan respons yang di berikan oleh server saat melakukan proses untuk pengambilan data kamus. *Endpoint* API ini dapat diakses melalui URL dengan format https://{*domain*}/kamus. Untuk mengirim permintaan ke *endpoint* tersebut, digunakan metode HTTP GET.

```
[
    {
        "id": 1,
        "title": "Kamus Aksara Bali",
        "imageUrl": "https://storage.googleapis.com/alea-ta-buckets/kamus/
            eff1ecf0-4784-11f0-977b-0f14c123393a-aksara-bali.jpg",
        "kategori": "bali",
        "kontenKamus": [
            {
                "imageUrl": "https://storage.googleapis.com/alea-ta-buckets/kamus/1/
                    f660a770-4784-11f0-977b-0f14c123393a-abhiseka_penobatan-removebg-preview.png",
                "caraBaca": "abhiseka",
                "arti": "penobatan",
                "jenis": "kata"
            },
```


Gambar 4. 2 Respons API *Dictionary*

Respons yang diterima berupa data dalam format JSON sebagaimana ditunjukkan pada Gambar 4.2. Struktur data ini terdiri atas entri-entri kamus yang masing-masing direpresentasikan oleh sebuah objek dengan lima atribut utama, yaitu: (1) id, bertipe integer sebagai penanda unik;(2) *title,* bertipe string yang memuat nama aksara; (3) *imageUrl*, bertipe string yang memuat url sebagai gambar pendukung visual; (4) kategori, bertipe string yang memuat jenis aksara tiap item; dan (5) kontenKamus, yang bertipe *array of object*. Setiap objek di dalam *array* kontenKamus yang memiliki empat atribut yang semuanya bertipe string, yaitu: (a) *imageUrl*, bertipe string yang memuat *url* gambar item kamus; (b) caraBaca, yang berisi representasi fonetik atau cara pembacaan aksara tersebut; (c) arti, yang menjelaskan makna atau terjemahan dari gambar yang ditampilkan; dan (d) jenis, yang menunjukkan kategori atau tipe dari konten kamus tersebut.

5. Quiz

Gambar 4.3 memperlihatkan respons yang di berikan oleh server saat melakukan proses pengambilan data *Quiz*. *Endpoint* API ini dapat diakses melalui URL dengan format https://{*domain*}/kuis. Untuk mengirim permintaan ke *endpoint* tersebut, digunakan metode HTTP GET.

```
[
    {
        "id": 1,
        "title": "Kuis Aksara Bali",
        "kategori": "bali",
        "konten": [
            {
                "imageSoal": "https://storage.googleapis.com/
                    alea-ta-buckets/kuis/1/
                    a7cc6400-4793-11f0-bcdb-0328a1d0f6dc-sura_dewa-remove
                    bg-preview.png",
                "jawabanA": "Swa",
                "jawabanB": "Lara",
                "jawabanC": "Sura",
                "jawabanD": "Tara",
                "isiJawabanBenar": "Sura"
            },
```


Gambar 4. 3 Respons API *Quiz*

Respons yang diterima berupa data dalam format JSON yang terdiri atas entri-entri kuis yang masing-masing direpresentasikan oleh sebuah objek dengan tiga atribut utama, yaitu: (1) id, bertipe integer sebagai penanda unik; (2) *title,* bertipe string memuat nama atau judul dari kuis tersebut; (3) kategori, bertipe string yang menunjukkan jenis aksara entri tersebut; dan (4) konten yang bertipe *array of object*. *Array* konten ini berisi daftar objek yang masing-masing memiliki enam atribut bertipe string, yaitu: (a) *image*Soal, yang menunjukkan URL atau path gambar soal yang akan ditampilkan dalam antarmuka aplikasi; (b) jawabanA, (c) jawabanB, (d) jawabanC, dan (e) jawabanD, yang masing-masing merepresentasikan pilihan jawaban dalam bentuk teks; serta (f) isiJawabanBenar, yang merupakan penanda jawaban benar dari keempat opsi tersebut.

6. History

Gambar 4.4 memperlihatkan respons yang di berikan oleh server saat melakukan proses pengambilan data *history*. *Endpoint* API ini dapat diakses melalui URL dengan format https://{*domain*}/sejarah. Untuk mengirim permintaan ke *endpoint* tersebut, digunakan metode HTTP GET. Saat permintaan berhasil, *server* mengembalikan respons berformat JSON yang memuat daftar objek *History.* Setiap objek dalam *array* tersebut memiliki empat atribut utama: (1) id, bertipe

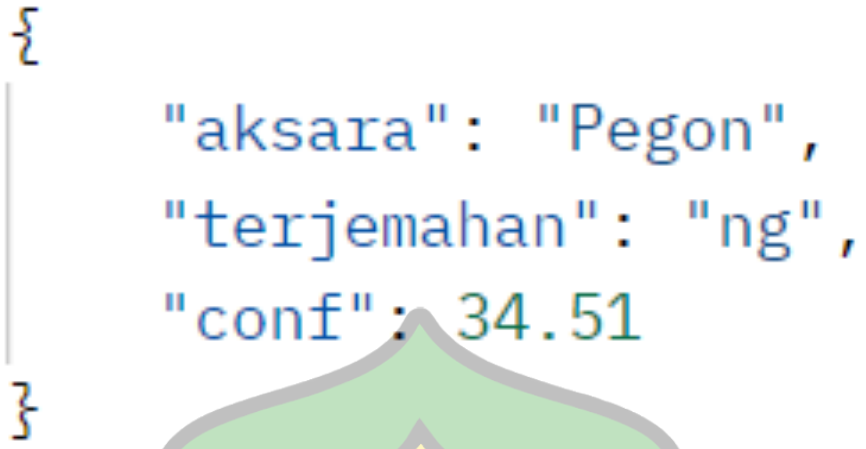
integer sebagai penanda unik; (2) *title,* bertipe string sebagai judul item; (3) *image,* bertipe string sebagai gambar pendukung visual; (4) kategori, bertipe string sebagai penanda jenis sejarah aksara; dan (5) *content,* bertipe string yang memuat isi detail item sejarah tersebut;


```
[
    {
        "id": 1,
        "title": "Sejarah Aksara Lampung dan Perkembangannya",
        "image": "https://storage.googleapis.com/alea-ta-buckets/sejarah/
            4f21e4e0-4781-11f0-977b-0f14c123393a-lampung.jpg",
        "content": "Aksara Lampung, dikenal juga sebagai Had Lampung
            atau Surat Lampung, merupakan sistem tulisan tradisional
            masyarakat Lampung yang berasal dari turunan aksara Brahmi
            melalui aksara Pallawa dan Kawi. Aksara ini terdiri dari 20
            huruf dasar dan 12 diakritik, digunakan untuk menulis bahasa
            Lampung dalam berbagai naskah adat, hukum, dan sastra.
            Penggunaannya mencapai puncak pada abad ke-17 hingga awal
            abad ke-20 sebelum tergantikan oleh huruf Latin. Sayangnya,
            saat ini aksara Lampung mulai luntur karena minimnya tenaga
            pendidik dan hilangnya kurikulum di sekolah-sekolah,
            meskipun upaya pelestarian terus dilakukan melalui
            pengajaran muatan lokal dan penggunaan di ruang publik.
            Sumber: rri.co.id/daerah/841188/
            sejarah-aksara-lampung-dan-perkembangannya",
        "kategori": "lampung"
    },
```


Gambar 4. 4 Respons API *History*

7. Scanner

Gambar 4.5 memperlihatkan respons yang di berikan oleh server saat melakukan proses pengambilan data *history*. *Endpoint* API ini dapat diakses melalui URL dengan format https://{*domain*}/*predict*. Untuk mengirim permintaan ke *endpoint* tersebut, digunakan metode HTTP POST dengan format data yang dikirimkan dalam bentuk *form-data* yang mempunyai *field key image* bertipe *file* dengan *value* merupakan *file* gambar aksara yang ingin di terjemahkan.



Gambar 4. 5 Respons API *Predict*

Respons yang dikembalikan berformat JSON yang berisi atribut: (1) aksara, bertipe string sebagai respons yang menampilkan asal aksara tersebut; (2) terjemahan, bertipe string yang memuat hasil *translate* gambar yang di *upload*; dan (3) conf, bertipe float yang menunjukkan skor kepercayaan hasil terjemahan;

8. Materi

Gambar 4.6 memperlihatkan respons yang di berikan oleh server saat melakukan proses pengambilan data materi aksara. *Endpoint* API ini dapat diakses melalui URL dengan format https://{*domain*}/materi. Untuk mengirim permintaan ke *endpoint* tersebut, digunakan metode HTTP GET. Saat permintaan berhasil, *server* mengembalikan respons *array of object* berformat JSON yang masing-masing objek tersebut memiliki atribut: (1) id, bertipe integer sebagai penanda unik; (2) *title*, bertipe string yang memuat judul untuk setiap objek; (3) *content*, bertipe string sebagai respons yang memuat hasil konten yang akan di tampilkan; (4) kategori, bertipe string yang memuat informasi kategori aksara mana entri tersebut, dan (5) *image*, bertipe string yang mengirimkan *url* dari gambar pendukung visual dari setiap objek.

```
[
    {
        "id": 1,
        "title": "Pengantar Aksara Sunda",
        "kategori": "sunda",
        "content": "<!DOCTYPE html><html><head><meta
            charset=\"UTF-8\"><title>Materi Aksara Sunda</title><style>body {
            font-family: 'Segoe UI', sans-serif; padding: 16px; line-height: 1.
            6; color: #333; } h2 { color: #2c3e50; } img { max-width: 100%;
            height: auto; margin: 12px 0; border-radius: 8px; box-shadow: 0 2px
            8px rgba(0,0,0,0.1); }</style></head><body><h2>Pengenalan Aksara
            Sunda</h2><p>Aksara Sunda adalah salah satu sistem penulisan
            tradisional...</p><img src=\"https://i.imgur.com/7FZ2ZjF.png\"
            alt=\"Contoh Aksara Sunda\"/><p>Belajar mengenal aksara ini membantu
            menjaga identitas dan kekayaan budaya daerah.</p></body></html>",
        "image": "https://storage.googleapis.com/alea-ta-buckets/materi/
            f4adf170-4910-11f0-a6d8-ff3574ac8671-materi-aksara-sunda.jpg"
    }
```


Gambar 4. 6 Respons API Materi

Setelah dilakukan pengujian respons API menggunakan Postman, penulis dapat memastikan bahwa struktur data yang dikembalikan oleh server telah sesuai dengan format yang dibutuhkan oleh aplikasi. Langkah ini menjadi dasar penting dalam merancang unit testing, khususnya untuk memvalidasi proses parsing dan pemetaan data ke dalam model internal aplikasi.

Dalam proses pengujian, objek *mock* digunakan untuk menggantikan *repository*, dan modul API. Hasil pengujian menunjukkan bahwa seluruh fungsi yang diuji telah berjalan sesuai dengan ekspektasi. Logika pencarian dan penyaringan data bekerja dengan benar berdasarkan *input* pengguna, fitur kuis berhasil menghitung skor dengan akurat sesuai jumlah jawaban yang benar, dan proses *parsing* data dari API berhasil memetakan struktur JSON ke dalam model aplikasi tanpa kesalahan, Dengan demikian, unit testing ini memberikan validasi kuat bahwa komponen-komponen logika inti dalam aplikasi ALEA telah berhasil diimplementasikan, hasil pengujian di sajikan dalam bentuk tabel yang dapat dilihat pada Tabel 4.1.

Tabel 4. 1 *Unit Testing*

No	*Unit Test Name*	*Test Criteria*	*Result*
1	*Dictionary Search And* Filter *Test*	Sistem menampilkan hasil pencarian dan penyaringan entri kamus sesuai dengan kata kunci/kategori	*passed*
2	Materi *Search Test*	Sistem menampilkan daftar materi yang relevan berdasarkan kata kunci yang dimasukkan pengguna	*passed*
3	*History Search Test*	Sistem menampilkan riwayat klasifikasi berdasarkan pencarian teks oleh pengguna	*passed*
4	*Quiz Search And* Filter *Test*	Sistem menampilkan daftar kuis sesuai dengan kata kunci dan/atau kategori yang dipilih	*passed*
5	*Quiz Update Score Test*	Sistem menambahkan skor kuis dengan benar sesuai jumlah jawaban benar yang diberikan pengguna	*passed*
6	*Quiz Answer Validation Test*	Sistem memvalidasi apakah jawaban pengguna sesuai dengan jawaban yang benar dari data soal	*passed*
7	API *Parsing Test*	Sistem berhasil memetakan respons JSON dari API menjadi struktur data internal aplikasi	*passed*

4.3 *Code Generation*

Proses implementasi pada tahap ini meliputi tiga komponen utama, yaitu: antarmuka pengguna, *internal database*, dan integrasi API. Antarmuka dikembangkan berdasarkan desain *wireframe* menggunakan XML dan Kotlin. *internal database* akan menggunakan *library* Room dengan entitas yang merepresentasikan struktur data yang telah di rancang sebelumnya di tahap *design*. Sementara itu, integrasi API dilakukan untuk mengambil dan memperbarui data serta menghubungkan fitur *scanner* dengan *backend* melalui protokol REST, memungkinkan pengiriman gambar dan penerimaan hasil prediksi secara *real-time.*

1. Implementasi *Design*

Berikut ini merupakan hasil konversi dari desain *wireframe* menjadi antarmuka aplikasi menggunakan bahasa XML yang telah terintegrasi dalam aplikasi ALEA dengan alur aktivitas yang mengacu pada *Activity* Diagram yang telah di buat, sedangkan logika aplikasi dikembangkan dengan menggunakan bahasa pemrograman Kotlin. Setiap elemen pada antarmuka ini dihubungkan dengan logika program menggunakan bahasa pemrograman Kotlin. Masing-masing hasil implementasi *design* aplikasi ALEA akan dijelaskan lebih lanjut pada subbab berikut, yang dibagi berdasarkan fitur dan halaman yang telah di implementasikan.

a. *Main Screen*

Tampilan awal atau *Main Screen* adalah halaman pertama yang akan dilihat pengguna ketika membuka aplikasi ALEA. Antarmuka ini menyajikan beberapa *shortcut* berupa tombol navigasi menuju fitur-fitur utama, yakni *Scanner* Aksara, Kuis Aksara, dan Belajar Aksara yang mencakup Kamus Aksara, dan Sejarah Aksara dan Materi Aksara. Implementasi tampilan *Main Screen* dapat dilihat pada Gambar 4.7.

Gambar 4. 7 Implementasi *Main Screen*

b. Belajar *Screen*

Gambar 4.8 menampilkan jenis aksara yang dapat dipilih pengguna untuk di pelajari yaitu: Aksara Sunda, Aksara Bali, Aksara Lampung, dan Arab Jawi yang ketika di klik pengguna akan di bawa ke halaman *Choose* Belajar *Type Screen*.

Gambar 4. 8 Implementasi Belajar *Screen*

c. *Choose* Belajar *Type Screen*

Gambar 4.9 menampilkan fitur belajar yang tersedia di ALEA berdasarkan kategori aksara yang dipilih pada Belajar *Screen*, yaitu fitur *Dictionary*, Materi, dan *History*.

Gambar 4. 9 Implementasi *Choose* Belajar *Type Screen*

d. *Dictionary Screen*

Gambar 4.10 menampilkan daftar isi kamus yang tersedia berdasarkan kategori aksara yang dipilih pada Belajar *Screen*. Informasi yang disajikan mencakup gambar aksara, cara pembacaan dalam Bahasa Indonesia, serta arti atau makna dari aksara tersebut.

Gambar 4. 10 Implementasi *Dictionary Screen*

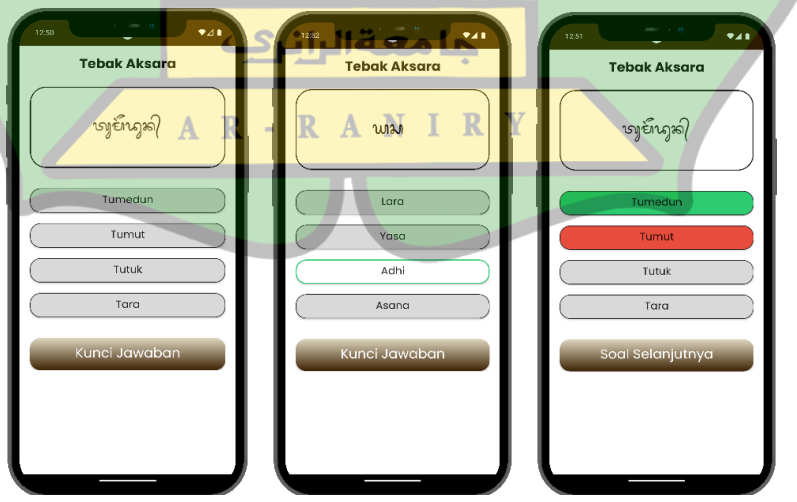
e. *Quiz Screen*

Gambar 4.11 menunjukkan bahwa daftar kuis berhasil ditampilkan berdasarkan jenis aksara yang tersedia, yaitu aksara Sunda, Bali, Arab Jawi, dan Lampung dan kuis campuran antara ke empatnya. Setiap kuis ditampilkan dalam bentuk daftar dengan judul dan deskripsi singkat yang mewakili konten dari masing-masing kuis. Pengguna dapat memilih salah satu kuis dari daftar untuk diarahkan ke halaman *Play Quiz*.

Gambar 4. 11 Implementasi *Quiz Screen*

f. *Play Quiz Screen*

Tampilan *Play Quiz* menyajikan soal-soal kuis secara satu per satu dalam bentuk gambar aksara, yang disertai dengan empat opsi jawaban berbentuk teks. Setelah pengguna memilih salah satu jawaban, sistem secara otomatis menampilkan umpan balik benar atau salah, kemudian melanjutkan ke soal berikutnya. Setelah seluruh soal selesai dijawab, skor akhir ditampilkan secara otomatis berdasarkan jumlah jawaban benar dan salah yang diperoleh pengguna sebagaimana terlihat pada Gambar 4.12 dan Gambar 4.13.



Gambar 4. 12 Implementasi *Play Quiz Screen*

Gambar 4. 13 Implementasi *Play Quiz Screen – Result*

g. *History Screen*

Gambar 4.14 menunjukkan tampilan *history* menyajikan daftar informasi sejarah dari jenis aksara yang telah dipilih sebelumnya di Belajar *Screen* . Data sejarah ditampilkan dalam bentuk daftar judul dan gambar pendukung. Ketika salah satu entri dipilih, pengguna akan diarahkan ke halaman Detail *History* yang menampilkan isi narasi secara lengkap, termasuk ilustrasi visual untuk memperjelas konteks sejarah aksara tersebut.

Gambar 4. 14 Implementasi *History Screen*

h. Detail *History Screen*

Halaman Detail *History* seperti terlihat pada Gambar 4.15 menampilkan informasi secara menyeluruh mengenai sejarah aksara yang telah dipilih oleh pengguna dari *History Screen*. Informasi yang ditampilkan mencakup judul, narasi teks sejarah secara lengkap, serta gambar pendukung yang merepresentasikan konteks budaya dari masing-masing aksara seperti aksara Sunda, Bali, Arab Jawi, dan Lampung. Untuk mengakomodasi panjangnya konten teks sejarah, halaman ini menggunakan komponen *ScrollView* agar seluruh informasi dapat ditampilkan secara vertikal tanpa terpotong.

Gambar 4. 15 Implementasi Detail *History Screen*

i. Dialog Status *Screen*

Dialog Status *Screen* berhasil menampilkan informasi status aplikasi saat proses pengambilan data dari *server* sedang berlangsung. Dialog muncul secara otomatis ketika sistem menjalankan permintaan data, seperti pada saat memuat konten dari fitur kamus, kuis, sejarah dan saat pertama kali membuka aplikasi, maupun pengecekan versi data. Indikator visual yang ditampilkan memberikan umpan balik kepada pengguna mengenai proses yang sedang berjalan.

Gambar 4. 16 Implementasi Dialog Status *Screen* – Gagal *Update*

Gambar 4. 17 Implementasi Dialog Status *Screen* – Memuat Data

Gambar 4. 18 implementasi Dialog Status *Screen* – Info *Update*

Gambar 4. 19 Implementasi Dialog Status *Screen* – Gagal Mengunduh Data

j. *Request Permission* Dialog *Screen*

Pada tampilan *Request Permission* Dialog *Screen*, sistem menampilkan antarmuka permintaan izin saat pengguna mengakses fitur *scanner* yang membutuhkan izin seperti kamera dan penyimpanan. Layar ini muncul ketika aplikasi mendeteksi bahwa izin yang dibutuhkan belum tersedia. Apabila pengguna menyetujui permintaan izin, sistem akan melanjutkan ke tampilan utama fitur terkait. Sebaliknya, jika izin ditolak, pengguna tetap berada pada halaman ini. Dapat dilihat pada Gambar 4.20 merupakan hasil implementasi *Request Permission* Dialog *Screen.*

Gambar 4. 20 Implementasi *Request Permission* Dialog *Screen*

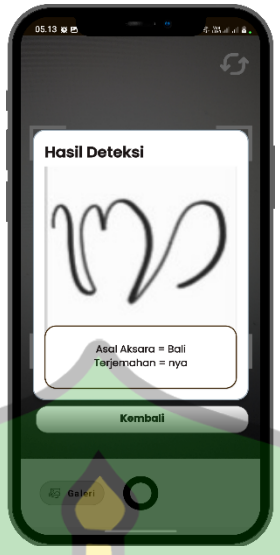
k. *Scanner Screen*

Hasil implementasi *Scanner Screen* seperti yang terlihat pada Gambar 4.21 menunjukkan bahwa pengguna dapat mengakses dua opsi *input* gambar, yaitu melalui kamera dan galeri. Setelah memasuki halaman ini, tombol kamera dan tombol galeri ditampilkan dengan fungsionalitas yang dapat digunakan untuk memilih atau mengambil gambar aksara. Jika izin yang dibutuhkan telah diberikan sebelumnya, sistem langsung menampilkan tampilan *scanner* tanpa hambatan.

Gambar 4. 21 Implementasi *Scanner Screen*

l. *Scanner Result*

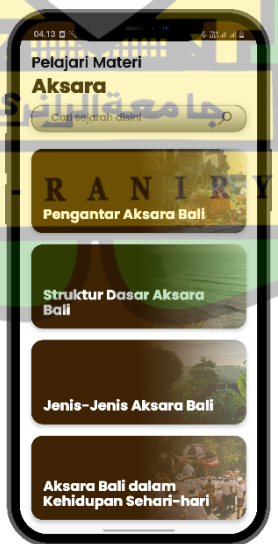
Gambar 4.22 menunjukkan hasil implementasi dari proses pemindaian gambar aksara yang dipilih oleh pengguna. Gambar tersebut dikirim ke *server* untuk diproses dan diterjemahkan ke dalam teks bahasa Indonesia. Setelah proses pemrosesan selesai, hasil terjemahan ditampilkan secara otomatis dalam antarmuka aplikasi. Hasil ditampilkan dalam bentuk teks yang merepresentasikan asal daerah aksara tersebut disertai arti dari aksara yang telah dipindai, sehingga pengguna dapat langsung memahami makna dari aksara yang di terjemahkan.



Gambar 4. 22 Implementasi *Scanner Result Screen*

m. Materi *Screen*

Gambar 4.23 menunjukkan materi *screen* yang menyajikan daftar materi yang tersedia berdasarkan jenis aksara yang telah dipilih sebelumnya di Belajar *Screen* . Data materi ditampilkan dalam bentuk daftar judul dan gambar pendukung. Ketika salah satu entri dipilih, pengguna akan diarahkan ke halaman Detail Materi *Screen* yang menampilkan isi konten entri yang dipilih secara lengkap.



Gambar 4. 23 Implementasi Materi *Screen*

n. Detail Materi *Screen*

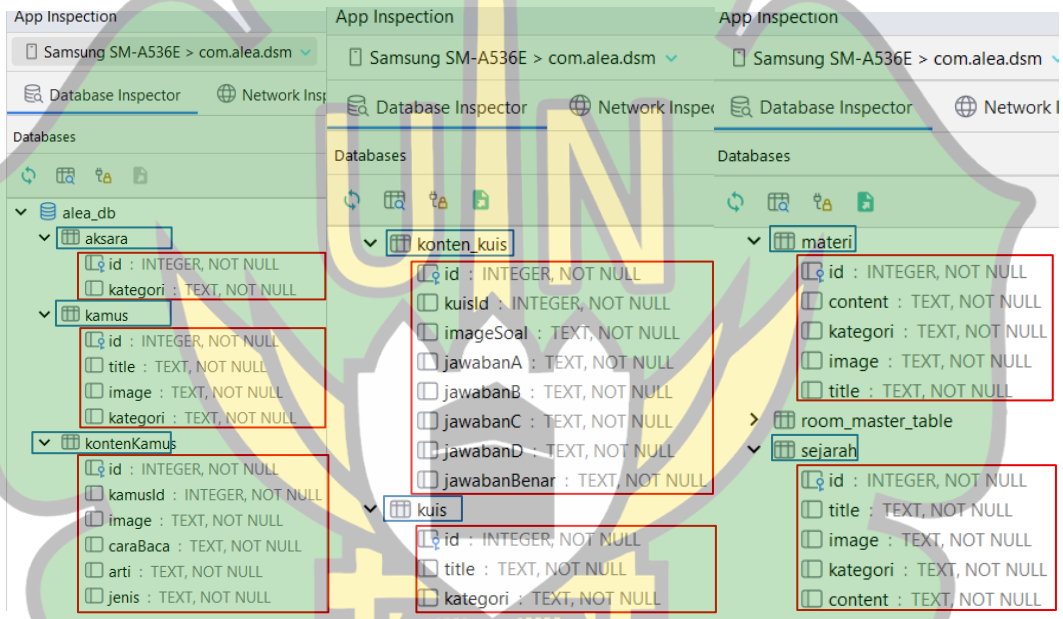
Halaman Detail Materi seperti terlihat pada Gambar 4.24 menampilkan informasi secara menyeluruh mengenai materi aksara yang telah dipilih oleh pengguna dari Materi *Screen*. Informasi yang ditampilkan mencakup judul, isi konten, serta gambar pendukung yang merepresentasikan konteks budaya dari masing-masing aksara seperti aksara Sunda, Bali, Arab Jawi, dan Lampung. Untuk mengakomodasi panjangnya konten teks sejarah, halaman ini menggunakan komponen *ScrollView* agar seluruh informasi dapat ditampilkan secara vertikal tanpa terpotong.

Gambar 4. 24 Implementasi Detail Materi *Screen*

2. *App Internal Database Schema*

Database internal aplikasi ALEA diimplementasikan menggunakan *library* Room. Room merupakan *library* dari Android Jetpack yang berfungsi sebagai lapisan abstraksi di atas SQLite, untuk mempermudah akses dan pengelolaan data lokal dalam aplikasi. Pemanfaatan *database* lokal ini merupakan fondasi utama yang memungkinkan aplikasi dapat beroperasi secara *offline*, sehingga pengguna tetap bisa mengakses aplikasi tanpa bergantung pada koneksi internet.

Skema *database* mencakup beberapa entitas yaitu kuis, kontenKuis, kamus, kontenKamus, materi, aksara, dan sejarah. Room *database* diinisialisasi menggunakan pendekatan *singleton* agar dapat diakses secara global di seluruh bagian aplikasi tanpa menyebabkan inisialisasi berulang. Visualisasi dari hasil implementasi struktur basis data internal menggunakan fitur *Database Inspector* dari android studio dapat dilihat pada Gambar 4.25, kolom dengan warna biru menunjukkan representasi entitas atau tabel sedangkan kolom warna merah menunjukkan atribut yang digunakan dalam penyimpanan data aplikasi secara lokal.

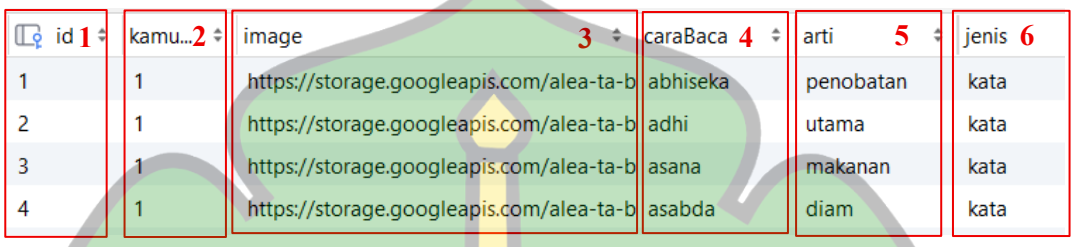


Gambar 4. 25 Implementasi *App Internal Database Schema*

id 1	title 2	image 3	kategori 4
1	Kamus Aksara Bali	https://storage.googleapis.com/alea-ta-buckets/kamus/eff1ecf0-47	bali
2	Kamus Aksara Lampung	https://storage.googleapis.com/alea-ta-buckets/kamus/0f71aaf0-4	lampung
3	Kamus Aksara Pegon	https://storage.googleapis.com/alea-ta-buckets/kamus/94a92800-4	pegon
4	Kamus Aksara Sunda	https://storage.googleapis.com/alea-ta-buckets/kamus/b0cc2670-4	sunda

Gambar 4. 26 Implementasi Kamus *Table*

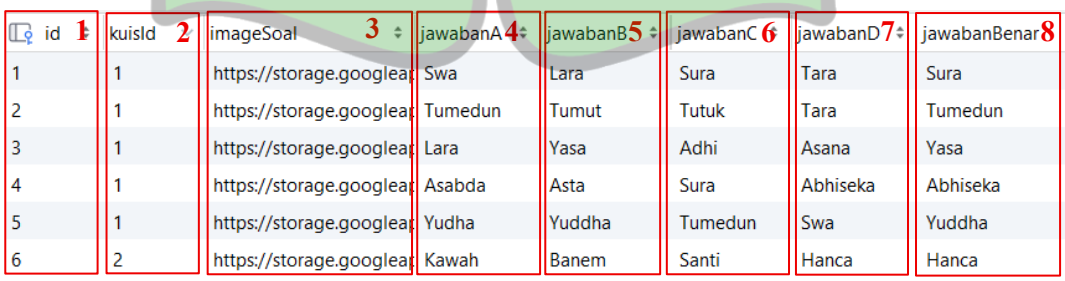
Gambar 4.26 menunjukkan isi tabel kamus yang menyimpan data daftar kamus aksara dengan empat atribut utama, yaitu: (1) id, bertipe *integer*, berfungsi sebagai *primary key* yang mengidentifikasi setiap entri secara unik; (2) *title*, bertipe string, menyimpan judul kamus; (3) *image*, bertipe string, berisi URL gambar sampul setiap kamus sebagai pendukung visual; dan (4) kategori, bertipe string, menunjukkan jenis aksara entri tersebut.



id 1	kamu...2	image 3	caraBaca 4	arti 5	jenis 6
1	1	https://storage.googleapis.com/alea-ta-b	abhiseka	penobatan	kata
2	1	https://storage.googleapis.com/alea-ta-b	adhi	utama	kata
3	1	https://storage.googleapis.com/alea-ta-b	asana	makanan	kata
4	1	https://storage.googleapis.com/alea-ta-b	asabda	diam	kata

Gambar 4. 27 Implementasi kontenKamus *Table*

Gambar 4.27 menunjukkan isi tabel kontenKamus yang merupakan bagian dari basis data lokal aplikasi ALEA. Tabel ini menyimpan data entri kosakata yang termasuk dalam masing-masing kamus aksara dengan enam atribut utama, yaitu: (1) id, bertipe integer, berfungsi sebagai *primary key* yang membedakan setiap entri secara unik; (2) kamusId, bertipe integer, merupakan *foreign key* digunakan untuk mengaitkan setiap entri dengan kamus tertentu, satu kamus dapat memiliki banyak entri kontenKamus; (3) *image*, bertipe string, berisi URL gambar aksara yang ditampilkan sebagai ilustrasi visual; (4) carabaca, bertipe string, menunjukkan cara baca atau pelafalan dari aksara tersebut; (5) arti, bertipe string, menyimpan arti atau makna dari aksara; dan (6) jenis, bertipe string menyimpan jenis item tersebut.



id 1	kuisId 2	imageSoal 3	jawabanA 4	jawabanB 5	jawabanC 6	jawabanD 7	jawabanBenar 8
1	1	https://storage.googlea	Swa	Lara	Sura	Tara	Sura
2	1	https://storage.googlea	Tumedun	Tumut	Tutuk	Tara	Tumedun
3	1	https://storage.googlea	Lara	Yasa	Adhi	Asana	Yasa
4	1	https://storage.googlea	Asabda	Asta	Sura	Abhiseka	Abhiseka
5	1	https://storage.googlea	Yudha	Yuddha	Tumedun	Swa	Yuddha
6	2	https://storage.googlea	Kawah	Banem	Santi	Hanca	Hanca

Gambar 4. 28 Implementasi kontenKuis *Table*

Gambar 4.28 menunjukkan isi tabel kontenKuis yang merupakan bagian dari basis data lokal aplikasi ALEA. Tabel ini menyimpan data soal-soal kuis aksara dengan delapan atribut utama, yaitu: (1) id, bertipe *integer*, berfungsi sebagai *primary key* yang mengidentifikasi setiap soal secara unik; (2) kuisId, bertipe integer, merupakan *foreign key* yang merujuk pada entitas kuis, untuk mengelompokkan soal berdasarkan jenis atau kategori kuis tertentu; (3) *image*Soal, bertipe string, berisi URL gambar aksara yang digunakan sebagai bahan soal kuis; (4) jawabanA, bertipe string, merupakan opsi jawaban pertama; (5) jawabanB, bertipe string, merupakan opsi jawaban kedua; (6) jawabanC, bertipe string, merupakan opsi jawaban ketiga; (7) jawabanD, bertipe string, merupakan opsi jawaban keempat; dan (8) jawabanBenar, bertipe string, menunjukkan jawaban yang benar dari keempat pilihan yang tersedia.

Gambar 4. 29 Implementasi kuis *Table*

Gambar 4.29 menunjukkan isi tabel kuis yang telah diimplementasikan. Tabel ini mempunyai tiga atribut utama, yaitu: (1) id, bertipe integer, berfungsi sebagai *primary key* yang mengidentifikasi setiap kuis secara unik; (2) *title*, bertipe string, menyimpan judul atau nama dari kuis yang akan ditampilkan kepada pengguna; dan (3) kategori, bertipe string, menunjukkan jenis aksara yang digunakan dalam kuis tersebut, seperti bali, sunda, lampung, Arab Jawi, maupun campuran.

Gambar 4. 30 Implementasi materi *Table*

Gambar 4.30 menunjukkan isi tabel materi yang merupakan bagian dari basis data lokal aplikasi ALEA dengan lima atribut utama, yaitu: (1) id, bertipe integer, berfungsi sebagai primary key yang mengidentifikasi setiap entri materi secara unik; (2) *content*, bertipe string, berisi konten materi yang disimpan dalam format HTML untuk mendukung tampilan yang terstruktur dan kaya informasi; (3) kategori, bertipe string, menunjukkan jenis aksara dari entri tersebut; (4) *image*, bertipe string, menyimpan URL gambar ilustratif untuk melengkapi penyajian materi; dan (5) *title*, bertipe string, berisi judul dari materi yang ditampilkan pada antarmuka pengguna.

Gambar 4. 31 Implementasi Sejarah *Table*

Gambar 4.31 menunjukkan isi tabel sejarah yang merupakan bagian dari basis data lokal aplikasi ALEA. Tabel ini mempunyai lima atribut utama, yaitu: (1) id, bertipe *integer*, berfungsi sebagai *primary key* yang mengidentifikasi setiap entri secara unik; (2) *title*, bertipe string, berisi judul dari artikel sejarah aksara yang akan ditampilkan kepada pengguna; (3) *image*, bertipe string, menyimpan URL gambar pendukung yang digunakan untuk memperkaya tampilan visual konten sejarah; (4) kategori, bertipe string, menunjukkan jenis aksara konten sejarah tersebut; dan (5) *content*, bertipe string, berisi narasi sejarah yang menjelaskan asal-usul, perkembangan, serta karakteristik dari masing-masing aksara daerah.

id 1	kategori 2
1	lampung
2	bali
3	pegon
4	sunda

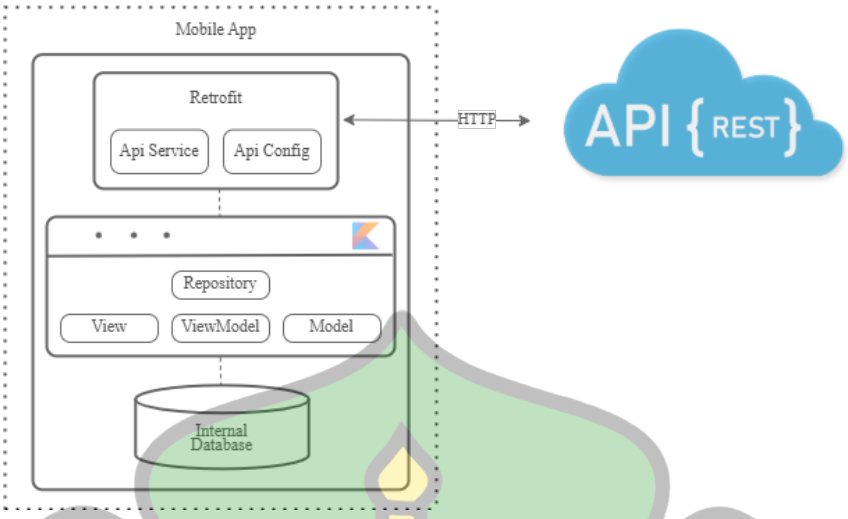
Gambar 4. 32 Implementasi Aksara *Table*

Gambar 4.32 menunjukkan isi tabel kategori yang merupakan bagian dari basis data lokal aplikasi ALEA. Tabel ini menyimpan data jenis aksara yang digunakan sebagai acuan klasifikasi dalam berbagai fitur aplikasi, seperti kamus, sejarah, dan kuis. Tabel ini terdiri dari dua atribut utama, yaitu: (1) *id*, bertipe *integer*, berfungsi sebagai *primary key* yang mengidentifikasi setiap entri kategori secara unik; dan (2) *kategori*, bertipe *string*, menyimpan nama aksara daerah yang tersedia dalam sistem yaitu lampung, bali, pegon, dan sunda.

3. Integrasi API

Integrasi *Application Programming Interface* (API) dalam aplikasi ALEA bertujuan untuk menghubungkan antarmuka pengguna dengan sistem *backend* yang menangani proses klasifikasi aksara secara otomatis. Komunikasi dilakukan menggunakan protokol HTTP dengan pendekatan REST, yang memungkinkan aplikasi mengirimkan data berupa gambar aksara serta menerima hasil klasifikasi dalam format JSON secara *real-time*.

Implementasi integrasi API dilakukan menggunakan *library* Retrofit. Retrofit dipilih karena kemampuannya dalam menyederhanakan proses permintaan HTTP dan konversi data JSON ke dalam objek model Kotlin. Setiap permintaan API dikirim melalui metode yang sesuai dengan kebutuhan masing-masing fitur. visualisasi gambaran proses integrasi API dapat dilihat pada gambar 4.32.



Gambar 4. 33 Integrasi API ALEA

Terdapat dua komponen utama dalam proses integrasi API yang diterapkan pada aplikasi ALEA, yaitu API *Config* dan API *Service*. Komponen ini bekerja sama untuk memungkinkan komunikasi antara aplikasi dan *backend server* secara efisien dan terstruktur.

API *Config* berfungsi sebagai konfigurasi utama dalam membangun objek Retrofit yang akan digunakan untuk semua permintaan HTTP. Komponen ini bertanggung jawab untuk menentukan *base* URL dari *server backend*, mengatur konverter JSON, serta mengelola aspek penting lain seperti *timeout*, *interceptor*, dan *logging* bila diperlukan. Dengan kata lain, API *Config* adalah fondasi yang memastikan bahwa semua permintaan API dilakukan secara konsisten, aman, dan sesuai standar yang diterapkan oleh pengembang. Detail struktur API *Config* dapat dilihat pada Gambar 4.33.

```
class ApiConfig {
    companion object {
        fun getApiService(): ApiService {
            val loggingInterceptor =
                HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY)
            val authInterceptor = Interceptor { user ->
                val requestUser = user.request()
                val requestHeaders = requestUser.newBuilder()
                    .addHeader("Accept", "application/json")
                    .build()
                user.proceed(requestHeaders)
            }
            val client = OkHttpClient.Builder()
                .connectTimeout(1, TimeUnit.SECONDS)
                .readTimeout(120, TimeUnit.SECONDS)
                .writeTimeout(120, TimeUnit.SECONDS)
                .addInterceptor(loggingInterceptor)
                .addInterceptor(authInterceptor)
                .build()
            val retrofit = Retrofit.Builder()
                .baseUrl("https://alea-api-rtidgzjhqa-et.a.run.app")
                .addConverterFactory(GsonConverterFactory.create())
                .client(client)
                .build()
            return retrofit.create(ApiService::class.java)
        }
    }
}
```

Gambar 4. 34 API *Config* ALEA

Selain API *Config*, komponen penting lainnya dalam proses integrasi API pada aplikasi ALEA adalah API *Service*. Komponen ini berfungsi sebagai *interface* yang mendefinisikan seluruh *endpoint* yang dapat diakses oleh aplikasi. API *Service* menentukan struktur permintaan HTTP seperti metode yang akan digunakan contohnya seperti GET dan POST, format parameter, jenis *payload*, serta tipe data yang diharapkan sebagai respons. Sedangkan rincian implementasi integrasi *endpoint* API *Service* yang digunakan dengan aplikasi dapat dilihat pada tabel 4.2.

Tabel 4. 2 Integrasi API *Service* ALEA

No	*Endpoint*	API *Service*	Hasil

1	https://{*domain*}/ *predict*	`@Multipart`
		`@POST("predict")`
		`suspend fun uploadImage(`
		`  @Part file: MultipartBody.Part,`
		`): UploadResponse`
			Terintegrasi

2	https://{*domain*}/ kuis	`@GET("kuis")`
		`suspend fun getAllKuis()`
		`: List<KuisResponse>`
			Terintegrasi

3	https://{*domain*}/ kamus	`@GET("kamus")`
		`suspend fun getAllKamus()`
		`: List<KamusResponse>`
			Terintegrasi

4	https://{*domain*}/ materi	`@GET("materi")`
		`suspend fun getAllMateri()`
		`: List<MateriResponse>`
			Terintegrasi

5	https://{*domain*}/ sejarah	`@GET("sejarah")`
		`suspend fun getAllSejarah()`
		`: List<SejarahResponse>`
			Terintegrasi

6	https://{*domain*}/ *version*	`@GET("version")`
		`suspend fun getDataVersion()`
		`: VersionResponse`
			Terintegrasi

4.4 *System Testing*

Setelah seluruh proses implementasi kode program pada aplikasi ALEA selesai dilakukan, tahapan selanjutnya adalah *system testing*. *System testing* dilakukan berdasarkan dua skenario utama, yaitu: (1) Pengguna belum memenuhi persyaratan sistem, skenario ini menguji perilaku sistem terhadap pengguna yang belum memberikan izin akses tertentu yang dibutuhkan oleh aplikasi, seperti izin kamera dan penyimpanan; (2) Skenario pengguna telah memenuhi seluruh persyaratan sistem, skenario ini menguji alur aplikasi terhadap pengguna yang telah memberikan semua izin akses dan telah memiliki data lokal aplikasi.

Skenario pengujian mencakup seluruh *test case* yang tertera pada Tabel 4.1. *System testing* dilakukan oleh pengembang dengan menguji aplikasi dari sudut pandang pengguna secara *end-to-end* menggunakan metode *blackbox*. Pengujian *Black Box* merupakan metode pengujian perangkat lunak yang berfokus pada evaluasi fungsionalitas sistem tanpa memerlukan pemahaman terhadap struktur internal atau detail implementasi kode program.

Tujuan utama dari pengujian ini adalah untuk memastikan bahwa seluruh fitur, fungsi, serta interaksi yang tersedia dalam perangkat lunak telah berjalan sesuai dengan kebutuhan dan spesifikasi fungsional yang telah ditetapkan sebelumnya. Salah satu keunggulan dari pendekatan ini adalah tidak adanya keharusan bagi penguji untuk memiliki pengetahuan teknis mengenai bahasa pemrograman atau arsitektur sistem yang diuji, sehingga proses pengujian dapat dilakukan secara lebih independen dari perspektif pengguna akhir. Pengujian akan dilakukan dengan cara menjalankan aplikasi baik itu pada perangkat fisik maupun emulator terhadap sistem yang telah terintegrasi secara menyeluruh.

Tabel 4. 3 *System Testing*

Feature	***User Story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
Main	Beranda	Belum memenuhi persyaratan sistem	Membuka aplikasi pertama kali	1. Pengguna membuka aplikasi untuk pertama kalinya		1. Aplikasi melakukan *download* data secara otomatis dari server jika internet tersedia 2. Menampilkan dialog status yang sesuai 3. Menampilkan halaman beranda ketika data terlah selesai di *download*	Berhasil
Main	Beranda	Sudah memenuhi persyaratan sistem	Menampilkan halaman beranda	1. Membuka aplikasi		1. Pengguna melihat halaman beranda	Berhasil
Dictionary	Kamus Aksara	Sudah memenuhi persyaratan sistem	Menampilkan halaman *dictionary*	1. Memilih jenis aksara pada halaman belajar 2. Memilih fitur kamus		2. Menampilkan daftar kamus yang di lengkapi dengan gambar, cara baca, dan arti dari jenis aksara yang dipilih	Berhasil

Feature	*User Story*	*Pre-Condition*	*Test Case*	*Test Steps*	*Test Data*	*Expected Result*	*Result*
Quiz	Kuis Aksara	Sudah memenuhi persyaratan sistem	Menampilkan halaman *quiz*	1. Memilih jenis aksara pada halaman belajar 2. Memilih fitur sejarah 3. memilih item sejarah		1. Menampilkan daftar *quiz* yang tersedia	Berhasil
Quiz	Kuis Aksara	Sudah memenuhi persyaratan sistem	Menampilkan halaman *play quiz*	1. Memilih Item *quiz* yang tersedia		1. Menampilkan soal *quiz* yang dipilih 2. Menampilkan skor benar dan salah setelah seluruh pertanyaan di jawab	Berhasil
History	Sejarah Aksara	Sudah memenuhi persyaratan sistem	Menampilkan halaman *history*	1. Memilih jenis aksara pada halaman belajar 2. Memilih fitur sejarah 3. memilih item sejarah		2. Menampilkan daftar sejarah dari jenis aksara yang dipilih 3. Menampilkan detail item sejarah yang dipilih	Berhasil

Feature	*User Story*	*Pre-Condition*	*Test Case*	*Test Steps*	*Test Data*	*Expected Result*	*Result*
Scanner	*Scanner*	Belum memenuhi persyaratan sistem	Menampilkan halaman *scanner*	1. Membuka halaman *scanner*		1. Menampilkan halaman *request permission* Dialog	Berhasil
Scanner	*Scanner* Melalui Kamera	Sudah memenuhi persyaratan sistem	Mengambil gambar menggunakan kamera	1. Membuka halaman *scanner* 2. Menekan tombol pengambil gambar		1. Gambar berhasil di ambil dan dikirim ke *server* 2. Menampilkan hasil *scan* aksara	Berhasil
Scanner	*Scanner* Melalui Galeri	Sudah memenuhi persyaratan sistem	Mengambil gambar menggunakan galeri	1. Membuka halaman *scanner* 2. Menekan tombol galeri		1. Gambar berhasil di ambil dan dikirim ke *server* 2. Menampilkan hasil *scan* aksara	Berhasil
Materi	Materi Aksara	Sudah memenuhi persyaratan sistem	Menampilkan halaman materi	1. Memilih jenis aksara pada halaman belajar 2. Memilih fitur materi 3. memilih item materi		1. Menampilkan daftar sejarah dari jenis aksara yang dipilih 2. Menampilkan detail item sejarah yang dipilih	Berhasil

Feature	*User Story*	*Pre-Condition*	*Test Case*	*Test Steps*	*Test Data*	*Expected Result*	*Result*
Filter	Filter	Sudah memenuhi persyaratan sistem	Melakukan filter kamus	1. *Tap* filter *button* di bagian atas halaman *quiz* di samping *search* 2. Pilih filter yang akan diterapkan	Filter = Hanya Kata	1. Menampilkan daftar item kamus berdasarkan filter yang dipilih	Berhasil
Filter	Filter	Sudah memenuhi persyaratan sistem	Melakukan filter *quiz*	1. *Tap* filter *button* di bagian atas halaman *quiz* di samping *search* 2. Pilih filter yang akan diterapkan	Filter = Aksara Sunda	1. Menampilkan daftar item *quiz* berdasarkan filter yang dipilih	Berhasil
Search	Pencarian Konten	Sudah memenuhi persyaratan sistem	Melakukan pencarian konten sejarah	1. *Tap search view* di bagian atas halaman *history* 2. Masukkan *keyword* di *form* pencarian	*Keyword* = lampung	1. Menampilkan daftar item sejarah berdasarkan kata pencarian yang dimasukkan	Berhasil
Search	Pencarian Konten	Sudah memenuhi persyaratan sistem	Melakukan pencarian konten materi	1. *Tap search view* di bagian atas halaman materi 2. Masukkan *keyword* di *form* pencarian	*Keyword* = penggunaan	1. Menampilkan daftar item materi berdasarkan kata pencarian yang dimasukkan	Berhasil

Feature	***User Story***	***Pre-Condition***	***Test Case***	***Test Steps***	***Test Data***	***Expected Result***	***Result***
Search	Pencarian Konten	Sudah memenuhi persyaratan sistem	Melakukan pencarian konten *quiz*	1. *Tap search view* di bagian atas halaman *quiz* 2. Masukkan *keyword* di *form* pencarian	*Keyword* = campuran	1. Menampilkan daftar item quiz berdasarkan kata pencarian yang dimasukkan	Berhasil
Search	Pencarian Konten	Sudah memenuhi persyaratan sistem	Melakukan pencarian konten kamus	1. *Tap search view* di bagian atas halaman *dictionary* 2. Masukkan *keyword* di *form* pencarian	*keyword* = utama	1. Menampilkan daftar item kamus berdasarkan kata pencarian yang dimasukkan	Berhasil
Offline Mode	*Offline Mode*	Sudah memenuhi persyaratan sistem	Menampilkan data tanpa internet	1. Membuka aplikasi tanpa akses internet		1. Menampilkan data tanpa akses internet 2. Menampilkan gambar berdasarkan *cache*	Berhasil

4.5 *Retrospective*

Tahap *retrospective* merupakan langkah akhir dalam setiap siklus iterasi, di tahap ini pengembang melakukan evaluasi menyeluruh terhadap proses kerja yang telah dilakukan, termasuk analisis terhadap estimasi waktu pengerjaan, dan realisasi waktu aktual. Detail *retrospective* dapat dilihat pada Tabel 4.4 dengan estimasi *story point* yang mengacu pada Tabel 3.5.

Tabel 4. 4 Hasil *Retrospective*

Iterasi	***User Story***	**Estimasi (*Story Point*)**	**Realisasi (*Story Point*)**	**Evaluasi**
1	Beranda	20	20	Selesai tepat waktu, tidak ada kendala berarti
2	Kuis Aksara	15	18	Terdapat deviasi waktu karena penyesuaian logika skor dan validasi jawaban
3	Materi Aksara	10	10	Implementasi berjalan lancar dan sesuai perencanaan
4	*Scanner* Melalui Kamera	12	12	Implementasi API berjalan sesuai target dan pengujian
5	*Scanner* Melalui Galeri	12	12	Tidak ada kendala teknis yang berarti
6	Kamus Aksara	15	15	Data dan tampilan dapat diintegrasikan dengan baik
7	Pencarian Konten	10	10	Pencarian berdasarkan *keyword* berjalan sesuai rencana
8	Sejarah Aksara	14	14	Penyelesaian sesuai estimasi
9	*Offline Mode*	7	7	Tidak ditemukan hambatan berarti dalam implementasi
10	Filter	7	7	Sesuai estimasi, *filtering* berjalan baik pada kamus dan kuis

4.6 *Minimim Viable Product* (MVP)

Setelah seluruh fitur utama yang dirancang pada BAB III berhasil diimplementasikan, aplikasi ALEA kemudian diuji secara langsung oleh pengguna awal. Tujuan dari pengujian ini adalah untuk mengevaluasi sejauh mana fitur-fitur yang telah dikembangkan mampu memenuhi kebutuhan dasar pengguna dalam mempelajari aksara daerah. Sebagaimana dijelaskan dalam Bab III, pendekatan MVP memungkinkan pengujian aplikasi sejak tahap awal guna memperoleh umpan balik langsung terhadap fitur paling esensial.

Pengujian dilakukan terhadap lima orang responden yang diminta untuk memberikan penilaian terhadap tiga aspek utama, yaitu: (1) kelayakan produk, (2) kemudahan penggunaan, dan (3) manfaat fitur yang tersedia. Penilaian diberikan menggunakan skala Likert 1 hingga 5, dengan ketentuan sebagaimana tercantum pada Tabel 3.13. Tabel berikut menyajikan hasil *rating* dari masing-masing pengguna:

Tabel 4. 5 Tabel Hasil Penilaian *Rating* ALEA

No	Nama Responden	Kelayakan Produk	Kemudahan Penggunaan	Manfaat Fitur	Rata-Rata
1	Responden A	5	4	4	4.33
2	Responden B	4	5	5	4.67
3	Responden C	4	4	4	4.00
4	Responden D	5	5	5	5.00
5	Responden E	4	4	5	4.33
Rata-rata		**4.4**	**4.4**	**4.6**	**4.46**

Berdasarkan data di atas, dapat disimpulkan bahwa aplikasi ALEA telah berhasil memenuhi kriteria sebagai MVP. Rata-rata skor di atas 4 untuk seluruh aspek menunjukkan bahwa fitur utama aplikasi tidak hanya berfungsi dengan baik, tetapi juga memberikan manfaat nyata dan dapat diakses dengan mudah oleh pengguna.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan uraian pada bab sebelumnya, maka dapat diambil kesimpulan sebagai berikut:

1. Antarmuka fitur *scanner* berhasil dikembangkan dan mampu memproses *input* gambar baik dari kamera maupun galeri. Gambar tersebut berhasil dikirim ke *backend server* untuk diterjemahkan ke bahasa Indonesia, respons hasil terjemahan yang dikirim server berhasil diterima dan ditampilkan.
2. Fitur *quiz* berhasil dikembangkan untuk menguji pemahaman pengguna terhadap aksara yang sedang dipelajari. Soal-soal kuis ditampilkan dalam bentuk pilihan ganda yang dikategorikan berdasarkan jenis aksara. Pengguna dapat berpindah antar soal, menjawab, dan melihat skor akhir berdasarkan jawaban yang benar.
3. Antarmuka fitur *dictionary*, materi, dan *history* berhasil dibangun untuk melengkapi proses belajar aksara. Ketiganya memberikan informasi yang dibutuhkan pengguna untuk mengenal aksara secara lebih menyeluruh, mulai dari bentuk, arti, hingga latar belakangnya. Kehadiran fitur ini membuat proses belajar menjadi lebih terarah, mudah dipahami, dan tidak terbatas hanya pada hafalan semata.
4. Fitur penggunaan *offline* pada aplikasi ALEA berhasil dikembangkan dengan mengandalkan penyimpanan data ke basis data lokal menggunakan *library* Room. Pengguna tetap dapat mengakses konten seperti kamus, materi, kuis, dan sejarah aksara tanpa koneksi internet.

5.2 Saran

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

1. Jenis aksara yang tersedia dalam aplikasi saat ini masih terbatas pada empat aksara, yaitu Sunda, Bali, Arab Jawi, dan Lampung. Akan lebih baik jika cakupan aksara diperluas agar semakin banyak pengguna dari berbagai daerah yang dapat belajar dan mengenal aksara daerahnya masing-masing.
2. Menambahkan fitur pencarian menggunakan suara untuk memudahkan pengguna dalam mencari kosakata atau materi tanpa harus mengetik secara manual.
3. Disarankan agar aplikasi menambahkan fitur pembelajaran berbasis audio, seperti *listening test*, juga dapat menjadi pelengkap pembelajaran aksara. Fitur ini memungkinkan pengguna mengenal bunyi aksara secara langsung dan dapat meningkatkan pemahaman melalui latihan mendengar.

DAFTAR PUSTAKA

Ardhy, F., & Syahrobi, H. (2021). Rancang Bangun Aplikasi Pembelajaran Aksara Lampung Berbasis Android. *Jurnal Informasi Dan Komputer*, *9*(2).

Azzahra, A. G., Andhara, R., Dewi, E. P., & Sujatini, [4]siti. (2024). Penerapan Metode Bisnis MVP Pada Ide Bisnis Gentle Glow: sabun wajah dapat dikustomisasi. *Jurnal Inovasi Kewirausahaan*, *2*. https://doi.org/10.37817/jurnalinovasikewirausahaan.v1i2

Ditha, R. L., Faulina, S. T., & Wisnumurti. (2023). Rancang Bangun Aplikasi Layanan Pengaduan Pada Dinas Pendidikan Kabupaten Oku Berbasis Android Menggunakan Android Studio. *Jurnal Informatika Dan Komputer (JIK)*, *14*(2), 25–35.

Dzulqarnain, F., & Tukino. (2023). Rancang Bangun Aplikasi Belajar Arab Untuk Android Menggunakan Jetpack Compose Dan Kotlin. *CBIS JOURNAL*, *11*(01). https://ejournal.upbatam.ac.id/index.php/cbis/article/view/6666

Erniati, S. S. (2020, October). *Keengganan Generasi Milenial pada Bahasa Daerah*. https://kantorbahasamaluku.kemdikbud.go.id/2020/10/kengganan-generasi-milenial-pada-bahasa-daerah/

Evelyn, Adipranata, R., & Gunadi, K. (2022). Sistem Presensi Mahasiswa Menggunakan Face Recognition Dengan Metode Facenet Pada Android. *Jurnal Infra*, *10*(2).

Fatah, H., Ichsan, N., Wahyuni, T., & Ernawati, E. (2020). Rancang Bangun Program Aplikasi Pembelajaran Aksara Sunda Berbasis Android. *Jurnal Sistem Informasi*, *9*(2), 304–320.

Fauzi, R., Nasution, H. N., Hastini, F., Zainy, A., & Lumban Tobing, Y. R. (2023). Perancangan Aplikasi Pariwisata Berbasis Android Di Kota Padang Sidempuan. *Jurnal Education And Development*, *11*(1), 437–442. https://doi.org/10.37081/ed.v11i1.2687

Firdaus, S. N. (2023). *Pengembangan Aplikasi Kamus Bahasa Aceh Berbasis Mobile*. Universitas Islam Negeri Ar-Raniry.

Gunada, W. A., Wiguna, I. B. A. A., & Yasa, M. A. (2022). Pengenalan Aksara Bali Pada Anak Usia Dini Melalui Media Gambar dan Mewarnai. *Dharma Sevanam: Jurnal Pengabdian Masyarakat*, *01*(02).

Hardyanto. (2023, April 16). *Merdeka Belajar untuk Revitalisasi Bahasa Daerah yang Terancam*. https://setkab.go.id/merdeka-belajar-untuk-revitalisasi-bahasa-daerah-yang-terancam/

Hartiyani, S. D., Prayogo, A., & Erlinawati, E. (2023). Aplikasi Multimedia Pembelajaran Aksara Jawa Berbasis Android Untuk Siswa Sekolah Dasar. *Jurnal Pendidikan, Sains Dan Teknologi*, *10*(2), 679–693. https://doi.org/10.47668/edusaintek.v10i2.805

Hartono, E., & Fauzi, R. (2021). Rancangan Aplikasi Pencarian Toko Handphone Murah Dan Terdekat Di Kota Batam Berbasis Android. *Jurnal Comasie*, *04*(05).

Indah Puji Astuti, Ega Feri Romawati, & Ida Widaningrum. (2020). Rancang Bangun Aplikasi Mobile Pengenalan Huruf Jawa (Aksara Jawa) Berbasis Android. *Jurnal CoSciTech (Computer Science and Information Technology)*, *1*(2), 93–100. https://doi.org/10.37859/coscitech.v1i2.2185

Jannah, M. (2023). *Analisis Kesulitan Menulis Huruf Arab Melayu Pada Mahasiswa PAI Tahun 2022/2023* [Universitas Islam Negeri Ar-Raniry]. https://repository.ar-raniry.ac.id/id/eprint/35685/

Kholidah, U., Fitriyani, D., Tussoleha, R., & Monica, L. (2023). Kesamaan Bunyi Bahasa Tulis Aksara Lampung Dan Aksara Batak Toba Sumatera Utara. *Bahtera Indonesia; Jurnal Penelitian Bahasa Dan Sastra Indonesia*, *8*(2), 414–422. https://doi.org/10.31943/bi.v8i2.429

Lortie, J., Cox, K., DeRosset, S., Thompson, R., & Kelly, S. (2024). Unpacking the minimum viable product (MVP): a framework for use, goals and essential elements. *Journal of Small Business and Enterprise Development*, *32*(1), 212–235. https://doi.org/10.1108/JSBED-02-2024-0075

Mutammimah, F. S., Aeni, A. N., & Nugraha, R. G. (2024). Pengembangan Aplikasi KAGANGA Berbasis Android untuk Mengenalkan Aksara Sunda di Sekolah Dasar. *Jurnal Karya Ilmiah Guru*, *9*(2). https://doi.org/10.51169/ideguru.v9i2.858

Nur Islamia, .W, Y., & Hamdani, F. (2024). Rancang Bangun Aplikasi Mobile Pembelajaran Aksara Bima Untuk Mempertahankan Kebudayaan Melalui Teknologi. *Management of Information System Journal*, *2*(3), 49–54.

https://doi.org/10.47065/mis.v2i3.1426

Pujianti, W., Nasir, M., & Mawarni, S. (2019). Perancangan Aplikasi Pembelajaran Arab Melayu Berbasis Augmented Reality Untuk Siswa Tingkat Sekolah Dasar. *Seminar Nasional Industri Dan Teknologi (SNIT)*, 37–44.

Riyan, M. (2021). Penggunaan Media Pembelajaran Berbasis Aplikasi Android Pada Pembelajaran Teks Eksposisi. *Diksi*, *29*(2).

Sari, C. N. (2023). *Perancangan Buku Ilustrasi Interaktif Pembelajaran Aksara Sunda Ngalagena Pada Murid Sekolah Dasar Negeri Baros Kencana CBMTugas Akhir*. Institut Teknologi Nasional Bandung.

Siswanto, E. (2022, April 18). *Mengenal Kotlin, Bahasa Pemrograman Kotlin*. https://teknik-informatika-s1.stekom.ac.id/informasi/baca/Mengenal-Kotlin-Bahasa-Pemrograman-Kotlin/31bbe8b6473b6f9ed5b03f8d8d1994a77a36d355

Sudiatmika, I. P. G. A., Hari Santhi Dewi, K., Raka Jayaningsih, A. A., & Widya Artana, W. (2022). *Application Using Android-Based Firebase And Jetpack Services For Thesis Guidance* [ITB Stikom Bali]. https://sion.stikom-bali.ac.id,

Sulfani, A. (2023, August 22). *Memahami tentang Android Studio, beserta keunggulan dan kekurangannya*. https://blog.unmaha.ac.id/memahami-tentang-android-studio-beserta-keunggulan-dan-kekurangannya

Waruwu, D. A. F., Himamunanto A. Rudatyo, & Setyawan, G. C. (2023). Image Tracking Berbasis AR Untuk Peningkatan Pembelajaran Buah Pada Pendidikan Anak Usia Dini (PAUD). *Infotek : Jurnal Informatika Dan Teknologi*, *6*(2), 381–389.

Wijaya, L. H., Zulkarnain, I. A., & Nurfitri, K. (2021). Pegon-Gliph Game Pengenalan Dan Pembelajaran Arab Pegon Berbasis Android. *KOMPUTEK*, *5*(1), 77. https://doi.org/10.24269/jkt.v5i1.685

Yudhistira, Y. (2021). Implementasi Application Programming Interface (API) Kawal Corona Sebagai Media Informasi Pandemik Covid-19 Berbasis Android. *Jurnal Sistem Informasi Dan Teknologi Peradaban (JSITP)*, *2*(1). www.journal.peradaban.ac.id

Zahratul Laila, S. (2024). *Skripsi Meningkatkan Kemampuan Keaksaraan Awal*

Melalui Media Kartu Huruf Pada Anak Kelompok B TK Yapis Fakfak. Universitas Pendidikan Muhammadiyah Sorong.

LAMPIRAN

Lampiran 1. Room *Database*

a. AleaDatabase

```
@Database(
    entities = [
        SejarahEntity::class,
        KamusEntity::class,
        KontenKamusEntity::class,
        KuisEntity::class,
        KontenKuisEntity::class,
        MateriEntity::class,
        AksaraEntity::class
                ],
    version = 1,
    exportSchema = true
)
abstract class AleaDatabase : RoomDatabase() {

    abstract fun aleaDao(): AleaDao

    companion object {
        @Volatile
        private var INSTANCE: AleaDatabase? = null

        fun getDatabase(context: Context): AleaDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AleaDatabase::class.java,
                    "alea_db"
                ).fallbackToDestructiveMigration().build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

b. AleaDao

```
@Dao
interface AleaDao {

    //Sejarah Dao Section
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertSejarah(sejarah: List<SejarahEntity>)
```

```
    @Query("SELECT * FROM sejarah")
    fun getAllSejarah(): LiveData<List<SejarahEntity>>

    @Query("SELECT * FROM sejarah where kategori = :kategori")
    fun getAllSejarahWithKategori(kategori: String):
LiveData<List<SejarahEntity>>

    @Query("SELECT * FROM sejarah WHERE title LIKE '%' || :query
|| '%'")
    fun searchSejarah(query: String):
LiveData<List<SejarahEntity>>

    //Kuis Dao Section
    @Transaction
    @Query("SELECT * FROM kuis")
    fun getAllKuisWithKonten(): LiveData<List<KuisWithKonten>>

    @Query("SELECT * FROM kuis where kategori = :kategori")
    fun getAllKuisWithKategori(kategori: String):
LiveData<List<KuisEntity>>

    @Query("SELECT * FROM konten_kuis WHERE kuisId = :id")
    fun getKontenKuisByKuisId(id: Int):
LiveData<List<KontenKuisEntity>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertKuis(kuis: List<KuisEntity>)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertKontenKuis(konten: List<KontenKuisEntity>)


    //Kamus Dao Section
    @Transaction
    @Query("SELECT * FROM kamus")
    fun getAllKamusWithKonten(): LiveData<List<KamusWithKonten>>

    @Query("SELECT * FROM kamus where kategori = :kategori")
    fun getAllKamusWithKategori(kategori: String):
LiveData<List<KamusEntity>>

    @Query("SELECT * FROM kontenKamus WHERE kamusId = :id")
    fun getKontenKamusByKamusId(id: Int):
LiveData<List<KontenKamusEntity>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertKamus(kamus: List<KamusEntity>)

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertKonten(konten: List<KontenKamusEntity>)


    //Materi Section
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertMateri(materi: List<MateriEntity>)

    @Query("SELECT * FROM materi where kategori = :kategori")
    fun getAllMateriWithKategori(kategori: String):
LiveData<List<MateriEntity>>
```

```
    //Aksara Section
    @Insert(onConflict = OnConflictStrategy.IGNORE)
    suspend fun insertAllAksara(aksaraList: List<AksaraEntity>)

    @Query("SELECT COUNT(*) FROM aksara")
    suspend fun getAksaraCount(): Int
}
```

c. AleaEntity

```
//SejarahSection
@Parcelize
@Entity(tableName = "sejarah")
data class SejarahEntity(
    @PrimaryKey
    val id: Int,
    val title: String,
    val image: String,
    val kategori: String,
    val content: String
): Parcelable


//KamusSection
@Parcelize
data class KamusWithKonten(
    @Embedded val kamus: KamusEntity,
    @Relation(
        parentColumn = "id",
        entityColumn = "kamusId"
    )
    val kontenKamus: List<KontenKamusEntity>
): Parcelable

@Entity(tableName = "kamus")
@Parcelize
data class KamusEntity(
    @PrimaryKey val id: Int,
    val title: String,
    val image: String,
    val kategori: String
): Parcelable

@Entity(
    tableName = "kontenKamus",
    foreignKeys = [
        ForeignKey(
            entity = KamusEntity::class,
            parentColumns = ["id"],
            childColumns = ["kamusId"],
            onDelete = ForeignKey.CASCADE
        )
    ],
    indices = [Index(value = ["kamusId"])]
)
```

```
@Parcelize
data class KontenKamusEntity(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val kamusId: Int,
    val image: String,
    val caraBaca: String,
    val arti: String,
    val jenis: String
): Parcelable


//Kuis Section
@Entity(tableName = "kuis")
data class KuisEntity(
    @PrimaryKey val id: Int,
    val title: String,
    val kategori: String
)

@Entity(
    tableName = "konten_kuis",
    foreignKeys = [
        ForeignKey(
            entity = KuisEntity::class,
            parentColumns = ["id"],
            childColumns = ["kuisId"],
            onDelete = ForeignKey.CASCADE
        )
    ],
    indices = [Index(value = ["kuisId"])]
)

data class KontenKuisEntity(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val kuisId: Int,
    val imageSoal: String,
    val jawabanA: String,
    val jawabanB: String,
    val jawabanC: String,
    val jawabanD: String,
    val jawabanBenar: String
)

data class KuisWithKonten(
    @Embedded val kuis: KuisEntity,
    @Relation(
        parentColumn = "id",
        entityColumn = "kuisId"
    )
    val konten: List<KontenKuisEntity>
)
```

```
//Materi Section
@Parcelize
@Entity(tableName = "materi")
data class MateriEntity(
    @PrimaryKey val id: Int,
    val content: String,
    val kategori: String,
    val image: String,
    val title: String
): Parcelable

@Entity(tableName = "aksara")
data class AksaraEntity(
    @PrimaryKey val id: Int,
    val kategori: String
)
```

Lampiran 2. CameraX *Function*

```
private fun startCamera() {
    val cameraProviderFuture =
ProcessCameraProvider.getInstance(requireContext())

    cameraProviderFuture.addListener({
        val cameraProvider: ProcessCameraProvider =
cameraProviderFuture.get()
        val preview = Preview.Builder()
            .build()
            .also {

it.setSurfaceProvider(binding.viewFinder.surfaceProvider)
            }

        imageCapture = ImageCapture.Builder().build()

        try {
            cameraProvider.unbindAll()
            cameraProvider.bindToLifecycle(
                this,
                cameraSelector,
                preview,
                imageCapture
            )
        } catch (exc: Exception) {
            showToast(requireContext(), "Gagal Membuka Kamera")
            Log.d("Scanner Fragment, Start Camera Function",
"Exception: $exc")
        }
    }, ContextCompat.getMainExecutor(requireContext()))
}
```

Lampiran 3. ***Upload Function***

```
private fun uploadImage(image: File) {
    viewModel.resetUploadResult()
    currentImageUri?.let { uri ->
        Log.d("Image File", "showImage: ${image.path}")
        val requestImageFile =
image.asRequestBody("image/jpeg".toMediaType())
        val multipartBody = MultipartBody.Part.createFormData(
            "image",
            image.name,
            requestImageFile
        )
        try {
            viewModel.uploadImage(multipartBody)
            Log.i("uploadImage", "file diupload")
        }catch (e: Exception){
            Log.d("uploadImage", "gagal mengupload file karena
$e")
        }
        Log.i("uploadImage", "file diupload")
    }
}
```

Lampiran 4. ***Upload With Camera Function***

```
private fun takePhoto() {
    val imageCapture = imageCapture ?: return
    val photoFile = createCustomTempFile(requireContext())
    val outputOptions =
ImageCapture.OutputFileOptions.Builder(photoFile).build()

    imageCapture.takePicture(
        outputOptions,
        ContextCompat.getMainExecutor(requireContext()),
        object : ImageCapture.OnImageSavedCallback {
            override fun onImageSaved(output:
ImageCapture.OutputFileResults) {
                currentImageUri = output.savedUri
                isCameraImage = true
                val imageFile = uriToFile(currentImageUri!!,
requireContext())
                uploadImage(imageFile)

            }
            override fun onError(exc: ImageCaptureException) {
                showToast(requireContext(), "Gagal Mengambil
Gambar")
                Log.e("Scanner Fragment, Start Camera Function",
"exception: ${exc.message}")
            }
        }
    )
}
```

Lampiran 5. ***Upload With Gallery Function***

```
private fun startGallery() {

launchGallery.launch(PickVisualMediaRequest(ActivityResultContract
s.PickVisualMedia.ImageOnly))
}
private val launchGallery = registerForActivityResult(
    ActivityResultContracts.PickVisualMedia()
) { uri: Uri? ->
    if (uri != null) {
        binding.captureImage.imageTintList =
ContextCompat.getColorStateList(requireContext(),
R.color.tint_capture_image)
        binding.constraintLayoutScannerHolder.apply {
            // Tetap tampil, tapi tidak bisa diklik
            isEnabled = false
            isClickable = false
            isFocusable = false
            alpha = 0.3f
        }
        binding.progressBarScanner.visibility = View.VISIBLE
        currentImageUri = uri
        isGalleryImage = true
        val imageFile = uriToFile(currentImageUri!!,
requireContext())
        uploadImage(imageFile)
    } else {
        showToast(requireContext(), "Tidak Ada Gambar Yang
Dipilih")
        Log.d("Photo Picker", "No media selected")
    }
}
```