DESAIN DAN IMPLEMENTASI ARSITEKTUR *BACKEND* BERBASIS *GOOGLE CLOUD* PADA APLIKASI ALEA UNTUK PEMBELAJARAN AKSARA

TUGAS AKHIR

Diajukan Oleh:

HANAFI AKBAR
210705046

Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI
AR-RANIRY BANDA ACEH
2026 M / 1447 H

LEMBAR PERSETUJUAN

DESAIN DAN IMPLEMENTASI ARSITEKTUR *BACKEND* BERBASIS *GOOGLE CLOUD* PADA APLIKASI ALEA UNTUK PEMBELAJARAN AKSARA

TUGAS AKHIR

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

Oleh:

Hanafi Akbar

210705046

Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui Untuk Munaqasyah Oleh:

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

Mengetahui,
Ketua Program Studi Teknologi Informasi

Malahayati, M.T
NIP.198301272015032003

LEMBAR PENGESAHAN

DESAIN DAN IMPLEMENTASI ARSITEKTUR *BACKEND* BERBASIS *GOOGLE CLOUD* PADA APLIKASI ALEA UNTUK PEMBELAJARAN AKSARA

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: Kamis, 29 Januari 2026
10 Sya'ban 1447
Di Darussalam, Banda Aceh

Panitia Ujian Munaqasyah Tugas Akhir:

Ketua,	Sekretaris,
Dr. Hendri Ahmadian, S.Si., M.I.M.	**Mulkan Fadhli, M.T**
NIP.198301042014031002	**NIP.198811282020121006**
Penguji I,	Penguji II,
Ghufran Ibnu Yasa, M.T	**Malahayati, M.T**
NIP.198409262014031005	**NIP.198301272015032003**

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

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

LEMBAR PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Hanafi Akbar
NIM	:	210705046
Program Studi	:	Teknologi Informasi
Fakultas	:	Sains dan Teknologi
Judul	:	DESAIN DAN IMPLEMENTASI ARSITEKTUR *BACKEND* BERBASIS *GOOGLE CLOUD* PADA APLIKASI ALEA UNTUK PEMBELAJARAN AKSARA

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

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

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

Banda Aceh, 8 Januari 2026

Yang Menyatakan

Hanafi Akbar

KATA PENGANTAR

Bismillahirrahmanirrahim.

Dengan menyebut nama Allah Yang Maha Pengasih lagi Maha Penyayang. Puji syukur penulis panjatkan kehadirat Allah SWT yang atas berkat, rahmat, dan karunia-Nya telah memberikan kekuatan dan kemudahan, sehingga penulis berhasil merampungkan Tugas Akhir yang berjudul "**DESAIN DAN IMPLEMENTASI ARSITEKTUR *BACKEND* BERBASIS *GOOGLE CLOUD* PADA APLIKASI ALEA UNTUK PEMBELAJARAN AKSARA**". Shalawat dan salam semoga senantiasa terlimpahkan kepada teladan umat manusia, Nabi Muhammad SAW, beserta keluarga, sahabat, dan para pengikutnya hingga akhir zaman.

Penulisan Tugas Akhir ini diajukan untuk memenuhi salah satu syarat akademis dalam memperoleh gelar Sarjana (S1) pada Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Ar-Raniry.

Penulis menyadari bahwa perjalanan dan penyelesaian studi ini tidak lepas dari bimbingan, dukungan, dan doa dari banyak pihak. Untuk itu, dengan penuh rasa hormat dan terima kasih, penulis ingin menghaturkan penghargaan yang tulus kepada:

1. Ayahanda Bustamam dan Ibunda Yusla Purba, pilar utama kekuatan dan sumber inspirasi. Terima kasih atas limpahan kasih sayang yang tiada henti, doa yang tak pernah terputus, serta dukungan moril dan materiel yang menjadi fondasi bagi penulis dalam menempuh pendidikan hingga titik ini. Semoga Allah SWT senantiasa melimpahkan karunia kesehatn dan kebahagiaan untuk ayah dan bunda.
2. Ibu Malahayati, M.T., selaku Ketua Program Studi Teknologi Informasi. Terima kasih yang sebesar-besarnya atas kesabaran, waktu, serta bimbingan akademis yang sangat berharga. Arahan, masukan, dan diskusi yang diberikan Beliau sangat membantu dalam mengarahkan penelitian ini.
3. Bapak Dr. Hendri Ahmadian, S.Si.,M.I.M.. selaku Pembimbing I, yang telah banyak meluangkan waktu untuk memberikan bimbingan, saran, dan

diskusi yang mencerahkan sehingga penelitian ini dapat terselesaikan dengan lebih baik.

4. Bapak Mulkan Fadhli, M.T., selaku Pembimbing II, yang dengan sabar memberikan arahan, masukan, dan dukungan intelektual sejak awal proses penelitian hingga selesai.
5. Ibu Cut Ida Rahmadiana, S.Si., atas segala bantuan, kemudahan, dan kesabaran dalam proses pengurusan administrasi akademik yang sangat menunjang kelancaran studi penulis.
6. Bapak Prof`. Dr. Ir. M. Dirhamsyah, M.T., IPU, selaku Dekan Fakultas Sains dan Teknologi UIN Ar-Raniry, atas segala dukungan dan fasilitas yang diberikan selama masa studi
7. Seluruh Bapak dan Ibu Dosen di lingkungan Program Studi Teknologi Informasi yang telah membagikan ilmu pengetahuan dan wawasan yang tak ternilai.
8. Rekan-rekan satu tim proyek aplikasi ALEA, penelitian ini dapat terwujud berkat sinergi yang luar biasa dengan Arief Fathin Abrar yang telah membangun model *machine learning* yang sangat andal maupun akurat, serta M. Dolyanda Harialdy yang mengembangkan aplikasi *mobile* sebagai platform implementasi. Terima kasih atas kolaborasi dan kerja kerasnya.
9. Sahabat dan rekan seperjuangan: Tri Ayu Lestari, Muhammad Uzir, Fillahi Akbar, Era Syafina, Cut Raudhatul Ilmi, Azlina Permaynuri dan Fira Elja Sabidra, yang telah menjadi teman berbagi dalam suka, duka, dan segala proses yang dijalani. Terima kasih atas semangat dan dukungan yang tulus.
10. Semua pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan kontribusi dalam penyelesaian Tugas Akhir ini.

Penulis mengakui bahwa karya ini masih memiliki banyak kekurangan. Oleh karena itu, segala bentuk saran dan kritik yang bersifat membangun akan penulis terima dengan lapang dada demi penyempurnaan di masa yang akan datang.

Akhir kata, semoga Tugas Akhir ini dapat bermanfaat bagi penulis, pembaca, dan pengembangan ilmu pengetahuan. Semoga Allah SWT mencatatnya sebagai amal kebaikan.

Aamiin yaa Rabbal'aalamiin.

Banda Aceh, 8 Januari 2026

Penulis,

Hanafi Akbar

ABSTRAK

Nama	:	Hanafi Akbar
NIM	:	210705046
Program Studi	:	Teknologi Informasi
Judul	:	Desain Dan Implementasi Arsitektur *Backend* Berbasis *Google Cloud* Pada Aplikasi Alea Untuk Pembelajaran Aksara
Tanggal Sidang	:	29 Januari 2026
Jumlah Halaman	:	87 Halaman
Pembimbing I	:	Dr. Hendri Ahmadian, S.Si., M.I.M.
Pembimbing II	:	Mulkan Fadhli, M.T

Aksara tradisional Nusantara merupakan bagian penting dari warisan budaya Indonesia, namun keberadaannya semakin terpinggirkan akibat minimnya media pembelajaran digital yang menarik dan mudah diakses. Pemanfaatan teknologi *cloud computing* dan *machine learning* menjadi salah satu solusi untuk mendukung pembelajaran aksara secara modern. Penelitian ini bertujuan untuk merancang dan mengimplementasikan arsitektur *backend* berbasis *Google Cloud Platform* (GCP) pada aplikasi Aksara *Learning App* (ALEA) sebagai media pembelajaran aksara tradisional. Penelitian ini menggunakan metode *Research and Development* (R&D) dengan tahapan analisis kebutuhan, perancangan arsitektur *backend*, implementasi infrastruktur *cloud*, integrasi model *machine learning*, pengujian sistem, serta *deployment* dan *monitoring*. Arsitektur *backend* dirancang menggunakan pendekatan *microservices* dan *serverless* dengan memanfaatkan layanan *Google Cloud Run*, *Cloud* SQL, dan *Cloud Storage*. *Backend* dikembangkan menggunakan Node.js dan Express.js untuk menyediakan API yang menghubungkan aplikasi dengan layanan *cloud*. Selain itu, sistem *backend* diintegrasikan dengan model *machine learning* berbasis *You Only Look Once* (YOLO) untuk melakukan prediksi aksara dari citra gambar secara *real-time*. Hasil penelitian menunjukkan bahwa arsitektur *backend* yang dibangun mampu berjalan secara stabil, responsif, dan skalabel, serta efisien dalam penggunaan sumber daya *cloud*.

Kata Kunci: Arsitektur *Backend*, *Google Cloud Platform*, *Cloud Run*, *Machine Learning*, YOLO, Pembelajaran Aksara.

ABSTRACT

Name	:	Hanafi Akbar
Student Id	:	210705046
Study Program	:	*Information Technology*
Title	:	*Design and Implementation of Google Cloud-Based Backend Architecture on the ALEA Application for Aksara Learning*
Thesis Defense Date	:	*January,* 29 2026
Total Pages	:	87 *Pages*
Supervisor I	:	Dr. Hendri Ahmadian, S.Si., M.I.M.
Supervisor II	:	Mulkan Fadhli, M.T

Traditional Indonesian scripts represent an important part of national cultural heritage; however, their use has gradually declined due to the limited availability of engaging and accessible digital learning media. The advancement of cloud computing and machine learning technologies provides an opportunity to develop modern and scalable learning systems for script education. This study aims to design and implement a backend architecture based on Google Cloud Platform (GCP) for the Aksara Learning App (ALEA) as a digital learning platform for traditional scripts. This research applies the Research and Development (R&D) method, which includes requirement analysis, backend architecture design, cloud infrastructure implementation, machine learning integration, system testing, deployment, and monitoring. The backend architecture is designed using a microservices and serverless approach by utilizing Google Cloud Run, Cloud SQL, and Cloud Storage. The backend services are developed using Node.js and Express.js to provide APIs that connect the application with cloud services. Furthermore, the backend system is integrated with a machine learning model based on You Only Look Once (YOLO) to perform real-time script recognition from image inputs. The results show that the proposed backend architecture operates in a stable, responsive, and scalable manner while maintaining efficient cloud resource utilization.

Keywords**:** *Backend Architecture, Google Cloud Platform, Cloud Run, Machine Learning, YOLO, Script Learning.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia memiliki kekayaan budaya yang luar biasa, salah satunya adalah keberadaan aksara tradisional seperti aksara Jawa, Lontara, Sunda, dan lainnya. Aksara-aksara ini bukan hanya alat komunikasi, tetapi juga merupakan identitas budaya yang mencerminkan sejarah dan nilai-nilai masyarakat setempat. Namun, seiring dengan perkembangan zaman dan dominasi aksara Latin, penggunaan aksara tradisional mengalami penurunan yang signifikan.

Penelitian yang menunjukkan bahwa penguasaan aksara Lontara di kalangan siswa sangat rendah, dengan 97,5% responden belum memahami aksara tersebut. Hal ini mencerminkan tantangan besar dalam pelestarian aksara tradisional di tengah arus modernisasi dan globalisasi (Muh Ayyub, 2024).

Namun, kemajuan teknologi digital menawarkan peluang untuk mengatasi tantangan tersebut. Pengembangan media pembelajaran berbasis digital dapat menjadi solusi untuk meningkatkan minat dan pemahaman siswa terhadap aksara tradisional. Sebagai contoh, pengembangan media interaktif "Marbel Raja" telah terbukti efektif dalam meningkatkan minat belajar aksara Jawa di kalangan siswa sekolah dasar (Salma, 2021).

Untuk mengatasi tantangan ini, dibutuhkan sebuah sistem pembelajaran aksara yang *modern* dan berbasis teknologi, yaitu ALEA. Sistem ini akan menggunakan *Google Cloud* untuk mendukung operasionalnya, memberikan pengalaman belajar yang lebih dinamis, mudah diakses, dan relevan dengan kebutuhan pengguna saat ini. Pengembangan sistem ini membutuhkan arsitektur *backend* yang handal dengan memanfaatkan layanan seperti *Cloud Run* dan *Cloud* SQL dari *Google Cloud* untuk memastikan kelancaran operasional serta penyediaan konten yang sesuai dengan preferensi pengguna.

Layanan seperti *Cloud Run* dan *Cloud* SQL dipilih untuk memastikan skalabilitas dan kelancaran operasional sistem. Teknologi ini memungkinkan sistem berjalan otomatis dan responsif terhadap lonjakan pengguna (Maulana & Bajili, 2024). *Google Cloud* mendukung efisiensi dalam pengembangan dan pemeliharaan

sistem, memastikan kemudahan pembaruan secara kontinu (Erwin dkk., 2023). *Google Cloud* menawarkan banyak keuntungan, tetapi salah satu tantangannya adalah biaya operasional yang tinggi, terutama ketika jumlah pengguna meningkat, yang dapat mempengaruhi efisiensi biaya pada skala besar (Efgivia, 2024)

Selain itu, Sistem ALEA juga akan mengintegrasikan teknologi *Machine Learning* (ML) untuk mendukung kemampuan prediksi aksara secara otomatis. ML telah digunakan secara luas dalam pengenalan pola dan gambar untuk klasifikasi aksara, yang memungkinkan sistem untuk secara otomatis mendeteksi dan mengidentifikasi aksara dari gambar yang diunggah oleh pengguna. *Model* prediksi yang digunakan dalam sistem ini akan dilatih menggunakan *dataset* aksara yang beragam, dan *model* tersebut akan diimplementasikan dalam format .pt. *Model* ini mampu memproses input berupa gambar aksara dan menghasilkan prediksi secara akurat melalui *Google Cloud Platform* (GCP). Dengan memanfaatkan layanan *Cloud Run* dan *Cloud Storage*, sistem ini akan mendukung proses prediksi secara *real-time* dan memberikan hasil yang cepat serta efisien, yang sangat penting dalam pembelajaran aksara di era digital (Saeed dkk., 2024)

Berdasarkan kondisi tersebut, penelitian ini mengusulkan pengembangan sistem pembelajaran aksara berbasis teknologi bernama Aksara *Learning Apps* (ALEA). Sistem ini memanfaatkan teknologi *Google Cloud Platform* (GCP) untuk membangun arsitektur *backend* yang handal, aman, dan skalabel.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, masalah utama dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana cara merancang arsitektur *backend* yang optimal untuk mendukung sistem pembelajaran aksara Aksara *Learning Apps* (ALEA) menggunakan layanan *Google Cloud Platform* (GCP)?
2. Bagaimana cara mengembangkan dan mengimplementasikan *Application Programming Interface* (API) pada arsitektur *backend* ALEA agar dapat mengintegrasikan model *Machine Learning* (YOLO) untuk melakukan prediksi aksara secara efisien dan *real-time*?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Merancang dan mengimplementasikan arsitektur *backend* yang optimal untuk sistem pembelajaran aksara Aksara *Learning Apps* (ALEA) dengan memanfaatkan layanan *Google Cloud Platform* (GCP), seperti *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*, guna mendukung performa sistem yang stabil, aman, dan skalabel.
2. Mengembangkan dan mengimplementasikan *Application Programming Interface* (API) pada arsitektur *backend* ALEA untuk mengintegrasikan model *Machine Learning* berbasis You Only Look Once (YOLO). API ini dirancang agar mampu memproses input gambar, menghubungkannya dengan model prediksi, serta menghasilkan keluaran aksara secara efisien, cepat, dan *real-time* di lingkungan *cloud* melalui layanan *Google Cloud Run*.

1.4 Batasan Penelitan

Penelitian ini memiliki beberapa batasan sebagai berikut:

1. Fokus penelitian hanya terbatas pada perancangan dan implementasi arsitektur *backend* berbasis *Google Cloud Platform* (GCP), tanpa membahas pengembangan antarmuka pengguna (*frontend*) secara mendalam.
2. Penelitian ini berfokus pada pengembangan dan implementasi *Application Programming Interface* (API) untuk integrasi model *Machine Learning* berbasis YOLO (.pt) ke dalam sistem *backend* ALEA. Proses pelatihan model YOLO dilakukan oleh tim *Machine Learning* dan tidak termasuk dalam ruang lingkup penelitian ini.
3. Pengujian prediksi aksara hanya dilakukan pada dataset aksara tertentu yang digunakan dalam pelatihan model YOLO. Penelitian ini tidak mencakup pengujian terhadap seluruh jenis aksara Nusantara atau variasi dataset di luar ruang lingkup penelitian.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Bagi Mahasiswa dan Pengembang Sistem: Penelitian ini dapat menjadi referensi dan panduan bagi mahasiswa serta pengembang yang ingin memahami penerapan arsitektur *backend* berbasis *cloud* menggunakan *Google Cloud Platform* (GCP). Melalui penelitian ini, pembaca dapat memperoleh wawasan mengenai pengembangan dan implementasi API yang terintegrasi dengan model *Machine Learning* berbasis *You Only Look Once* (YOLO) untuk melakukan prediksi aksara secara efisien dan *real-time*.
2. Bagi Institusi Pendidikan dan Peneliti di Bidang Teknologi Pendidikan: Sistem ALEA yang dikembangkan diharapkan dapat menjadi solusi pembelajaran interaktif yang memanfaatkan teknologi *cloud* dan kecerdasan buatan (AI) untuk mendukung pelestarian aksara tradisional Nusantara. Dengan penerapan sistem berbasis *cloud* ini, institusi pendidikan dapat mengadopsi *platform* pembelajaran *modern* yang fleksibel, skalabel, dan mudah di akses, sehingga proses belajar aksara menjadi lebih menarik, efektif, dan sesuai dengan perkembangan teknologi.

1.6 Sistematika Penulisan

Untuk memudahkan pembahasan, laporan Tugas Akhir ini disusun secara sistematis ke dalam beberapa bab dengan rincian sebagai berikut:

1. Bab I Pendahuluan: Bab ini berisi latar belakang penelitian, rumusan masalah, tujuan penelitian, batasan penelitian, manfaat penelitian, serta sistematika penulisan. Bab ini memberikan gambaran umum mengenai alasan dilakukannya penelitian, fokus permasalahan yang diangkat, dan arah penelitian yang dilakukan.
2. Bab II Tinjauan Pustaka: Bab ini membahas teori-teori yang menjadi dasar penelitian, meliputi konsep arsitektur *backend* berbasis *cloud*, *Google Cloud Platform* (GCP), *microservices architecture*, serta teori tentang *Machine Learning* (YOLO) yang digunakan dalam sistem ALEA. Bab ini

juga memuat tinjauan penelitian terdahulu yang relevan sebagai landasan ilmiah dalam pengembangan sistem.

3. Bab III Metodologi Penelitian: Bab ini menjelaskan secara rinci metode penelitian yang digunakan (*Research and Development*/R&D), tahapan pengembangan sistem, rancangan arsitektur *backend*, proses integrasi model YOLO, serta mekanisme *deployment* API di *Google Cloud Run*. Bab ini juga mencakup uraian mengenai waktu dan lokasi penelitian, dataset yang digunakan, serta tahapan pengujian sistem.
4. Bab IV Implementasi dan Hasil Penelitian: Bab ini memaparkan proses implementasi sistem berdasarkan tahapan penelitian yang telah dirancang pada Bab III. Bagian ini menjelaskan hasil pengembangan *backend*, pengujian API, performa sistem dalam melakukan prediksi aksara, serta analisis hasil pengujian dari aspek efisiensi dan kecepatan layanan berbasis *cloud*.
5. Bab V Penutup: Bab ini berisi kesimpulan dari hasil penelitian serta saran untuk pengembangan sistem di masa mendatang. Kesimpulan disusun berdasarkan hasil implementasi dan pengujian, sedangkan saran diberikan untuk penyempurnaan ALEA agar dapat berkembang lebih lanjut dan mendukung pembelajaran aksara yang lebih luas.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu mengenai sistem pembelajaran aksara masih terbatas, namun beberapa studi yang relevan dapat dikaitkan dengan penelitian ini, khususnya dalam konteks pembelajaran berbasis teknologi dan *cloud computing*. Pengembangan aplikasi untuk pembelajaran aksara, terutama aksara kuno seperti Jawa atau Bali, memiliki tantangan tersendiri. Dalam konteks ini, pemanfaatan teknologi *modern* seperti *cloud computing* menjadi solusi yang memungkinkan pengajaran aksara lebih interaktif dan dapat diakses lebih luas.

Dalam studi oleh Prasetiadi dkk., (2023) yang berjudul *Deep Learning Approaches for Nusantara Scripts Optical Character Recognition*, peneliti mengembangkan *model* berbasis *deep learning* untuk pengenalan aksara Nusantara. Hasil dari penelitian ini mengungkapkan bahwa *model* tersebut mencapai akurasi 96% dalam mengklasifikasikan jenis aksara, dan pengenalan karakter mencapai akurasi antara 93% hingga hampir 100%. Temuan ini menunjukkan bahwa teknologi *modern* mampu berkontribusi pada pelestarian aksara tradisional yang terancam punah.

Anh Vu (2021) dalam tesisnya yang berjudul *Real-time Backend Architecture Using Node.js, Express, and Google Cloud Platform* memfokuskan pada pengembangan arsitektur *backend* yang tidak hanya handal tetapi juga skalabel dan tersedia tinggi. Penelitian ini mengeksplorasi penerapan teknologi *Node.js* dan *Express* dalam lingkungan *cloud*, menggunakan *Google Cloud* sebagai *platform* infrastruktur utama. Salah satu contoh aplikasi yang dikembangkan untuk mendemonstrasikan arsitektur ini adalah aplikasi pemesanan taksi *real-time*, di mana kebutuhan latensi rendah dan kecepatan respons menjadi sangat penting. Vu melakukan analisis mendalam terhadap kelebihan dan kekurangan arsitektur berbasis *Node.js* dan *Google Cloud*, terutama dalam konteks *real-time application development*. Kelebihan utamanya termasuk kemudahan pengembangan dan kemampuan arsitektur untuk menangani peningkatan beban secara dinamis. Namun, penelitian ini juga menyoroti beberapa tantangan, termasuk proses

pengaturan *WebSockets* yang masih memerlukan perbaikan serta kompleksitas dalam pengelolaan skalabilitas sistem. Hasil penelitian menunjukkan bahwa arsitektur ini dapat memberikan solusi *backend* yang efisien dan mudah disesuaikan dengan kebutuhan sistem *real-time*, tetapi masih ada ruang untuk perbaikan lebih lanjut, terutama terkait otomatisasi *deployment* dan integrasi sistem

Dalam penelitian oleh Adilazuarda dkk., (2025) yang berjudul *NusaAksara: A Multimodal and Multilingual Benchmark for Preserving Indonesian Indigenous Scripts*, para peneliti memperkenalkan *benchmark* multimodal yang mencakup delapan aksara lokal Indonesia. Studi ini bertujuan untuk mendorong pelestarian aksara Nusantara melalui pendekatan NLP dan *machine learning* dengan data teks dan gambar. Hasil penelitian menunjukkan bahwa model NLP modern masih kesulitan dalam memahami aksara lokal, sehingga dibutuhkan pendekatan khusus untuk pengolahan dan pengenalan karakter aksara Indonesia. Penelitian ini menjadi dasar penting dalam pengembangan sistem seperti ALEA yang bertujuan melestarikan aksara tradisional melalui teknologi modern berbasis *cloud* dan *machine learning*.

Khatiwada dan Dhakal (2024), dalam makalah mereka yang berjudul *Evaluating Serverless Machine Learning Performance on Google Cloud Run*, mengkaji efisiensi *deployment model machine learning* pada layanan *serverless Google Cloud Run*. Mereka mengevaluasi performa dan efisiensi biaya ketika *model* dijalankan tanpa infrastruktur GPU, serta membandingkan dengan pendekatan tradisional berbasis *virtual machine*. Studi ini sangat relevan untuk implementasi ALEA, karena menunjukkan bagaimana layanan *serverless* dapat mendukung sistem pembelajaran aksara secara efisien dan fleksibel di *cloud*.

Penelitian oleh Anggara dan Zaman (2024) dalam jurnal *Computing* yang berjudul *Desain Arsitektur Server Google Cloud untuk Mengoptimalkan Kinerja Aplikasi Daily Cloud* menggambarkan strategi desain *backend* menggunakan *Google Cloud Run*, *Firestore*, dan *Firebase Authentication*. Meskipun konteksnya adalah aplikasi kesehatan mental, desain arsitekturnya sangat aplikatif untuk sistem pembelajaran seperti ALEA, terutama dari sisi skalabilitas, responsivitas sistem, dan integrasi layanan *cloud-native*.

Sementara itu, studi oleh EIKON Technology (2023) dalam artikelnya *Pedoman Google Cloud untuk Mengembangkan Solusi Machine Learning Berkualitas Tinggi* menguraikan panduan praktis dalam mengembangkan *pipeline* ML di *Google Cloud*. Panduan ini meliputi proses pelatihan, penyimpanan *model* di *Cloud Storage*, *deployment* dengan *Cloud Functions*, dan pengelolaan *model* dengan *Vertex AI*. Studi ini sangat bermanfaat dalam pengembangan sistem prediksi aksara otomatis dalam ALEA, yang memanfaatkan *model deep learning*.

Akhirnya, penelitian oleh Almuhammadi dan Makhdoom (2020) berjudul *Benefits and Challenges of Cloud Computing in Higher Education* menyoroti keuntungan dan hambatan yang dihadapi oleh institusi pendidikan tinggi saat mengadopsi teknologi *cloud*. Hasil penelitian ini menunjukkan bahwa sementara ada banyak keuntungan, seperti peningkatan aksesibilitas dan kolaborasi, tantangan terkait biaya dan keamanan tetap menjadi perhatian utama. Penelitian ini juga menekankan pentingnya pelatihan bagi staf akademik dan siswa untuk memaksimalkan penggunaan teknologi *cloud* dalam konteks pembelajaran.

Tabel berikut ini adalah perbandigan penelitian terkait, penjelasan lebih lanjut dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

PENELITI	METODE PENILITIAN	JUDUL PENELITIAN	HASIL PENELITIAN
Adilazuarda dkk., (2025)	Pengumpulan data multimodal (teks dan gambar) dan evaluasi model NLP dan *machine learning* untuk pengenalan aksara Nusantara	*NusaAksara: A Multimodal and Multilingual Benchmark for Preserving Indonesian Indigenous Scripts*	Model NLP *modern* masih kesulitan memahami aksara lokal, sehingga diperlukan pendekatan khusus untuk pengolahan dan pengenalan karakter aksara Indonesia.

Khatiwada dan Dhakal (2024)	Eksperimen *deployment model machine learning* pada layanan *serverless Google Cloud Run*, pengukuran *latency*, *throughput*, dan penggunaan sumber daya, serta perbandingan dengan VM tradisional.	*Evaluating Serverless Machine Learning Performance on Google Cloud Run*	Layanan *serverless* dapat mendukung sistem pembelajaran aksara secara efisien dan fleksibel dengan biaya yang kompetitif.
Anggara dan Zaman (2024)	Studi kasus dan desain sistem *backend* menggunakan *Google Cloud Run*, *Firestore*, dan *Firebase Authentication*, dengan pengujian prototipe aplikasi.	*Desain Arsitektur Server Google Cloud untuk Mengoptimalkan Kinerja Aplikasi Daily Cloud*	Arsitektur *backend* berbasis *Google Cloud* menunjukkan skalabilitas dan responsivitas yang baik, cocok untuk aplikasi pembelajaran seperti ALEA.

EIKON Technology (2023)	*Review* dan *best practices* pengembangan *pipeline machine learning* di *Google Cloud*, meliputi pelatihan model, penyimpanan, *deployment*, dan *monitoring*.	*Pedoman Google Cloud untuk Mengembangkan Solusi Machine Learning Berkualitas Tinggi*	Panduan ini membantu pengembangan sistem prediksi aksara otomatis yang akurat dan efisien menggunakan layanan *Google Cloud*.
Prasetiadi dkk., (2023)	*Deep Learning* (CNN, ConvMixer)	*Deep Learning Approaches for Nusantara Scripts Optical Character Recognition*	Penelitian ini berhasil mencapai akurasi 96% dalam mengklasifikasikan jenis aksara dan antara 93% hingga 100% untuk pengenalan karakter aksara Nusantara.
Anh Vu (2021)	Pengembangan arsitektur *backend* dengan *Node.js* dan *Express* pada *platform Google Cloud*	*Real-time Backend Architecture Using Node.js, Express, and Google Cloud Platform*	Hasil penelitian menunjukkan bahwa penggunaan *Node.js* dan *Express* di *Google Cloud* memungkinkan pengembangan arsitektur *backend* yang dapat diskalakan, handal, dan efisien.

			Studi kasus dalam aplikasi pemesanan taksi *real-time* menyoroti kemudahan pengembangan serta kemampuan menangani peningkatan beban secara dinamis.
Almuhammadi, M., & Makhdoom, A. (2020)	Analisis literatur	*Benefits and Challenges of Cloud Computing in Higher Education*	Penelitian ini menyoroti keuntungan dan hambatan dalam penerapan *cloud computing* di pendidikan tinggi, serta pentingnya pelatihan bagi staf akademik dan siswa.

2.2 Aksara

Aksara merupakan lambang yang mewakili bunyi atau fonem. Dalam Kamus Besar Bahasa Indonesia (Tim Penulis, Ensiklopedia 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, terutama dalam bahasa Arab, 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, di mana "a-" berarti "tidak", dan "kshara" berarti "musnah".

Jannah dkk. (2024) menjelaskan bahwa aksara merupakan sistem simbol visual yang digunakan untuk merepresentasikan bunyi atau fonem dalam komunikasi bahasa. Dalam kajian mereka, aksara diklasifikasikan menjadi empat

jenis utama, yakni piktografik, ideografik, silabik, dan fonetik, yang masing-masing memiliki karakteristik berbeda dalam merepresentasikan makna dan bunyi. Mereka menegaskan bahwa aksara tidak hanya berfungsi sebagai alat komunikasi, tetapi juga sebagai warisan budaya yang penting untuk dilestarikan, terutama dalam konteks keberagaman aksara Nusantara. Pelestarian aksara tradisional menjadi kunci dalam menjaga identitas budaya sekaligus sebagai media pendidikan untuk generasi muda agar dapat memahami dan melestarikan nilai-nilai sejarah yang terkandung di dalamnya.

Menurut Perdana (2024), aksara tradisional di Indonesia tidak sekadar berfungsi sebagai simbol komunikasi, melainkan juga sebagai penanda identitas budaya yang melekat pada masyarakat lokal. Penelitian mereka menunjukkan bahwa keberadaan aksara tradisional menghadapi berbagai tantangan dalam era digitalisasi, seperti keterbatasan akses sumber belajar dan berkurangnya pengguna aktif. Oleh karena itu, pengintegrasian teknologi digital menjadi solusi penting agar aksara tradisional tetap relevan dan mudah diakses oleh generasi modern. Penulis juga menyoroti peluang yang ditawarkan oleh teknologi digital untuk mengembangkan sistem pembelajaran yang interaktif dan mendukung pelestarian aksara tradisional dengan lebih efektif.

Widodo dkk. (2023) memaparkan bahwa aksara Jawa memiliki peran sentral dalam mempertahankan identitas budaya masyarakat lokal yang kaya akan nilai sejarah dan tradisi. Aksara Jawa, sebagai sistem tulisan fonetik, tidak hanya berfungsi untuk merekam bunyi bahasa, tetapi juga mengandung nilai estetika dan ekspresif yang mencerminkan keunikan budaya Jawa. Penelitian ini menekankan pentingnya pelestarian aksara Jawa dalam menghadapi tantangan modernisasi dan globalisasi yang cenderung mengikis penggunaan aksara tradisional. Melalui upaya pelestarian yang melibatkan pendidikan dan teknologi digital, aksara Jawa dapat tetap hidup sebagai bagian dari warisan budaya yang harus dijaga keberlanjutannya.

2.3 *Backend Development*

Backend development merupakan fondasi utama dalam pengembangan aplikasi modern, mencakup pengelolaan data, logika bisnis, dan komunikasi dengan infrastruktur server. Fungsi-fungsi ini sangat krusial dalam memastikan aplikasi

berjalan stabil, aman, dan mampu menangani permintaan dari pengguna dalam jumlah besar.

Menurut Dileep Domakonda (2025), arsitektur *backend* modern yang mengadopsi pendekatan *microservices* menawarkan fleksibilitas dan skalabilitas tinggi karena memungkinkan pemisahan layanan ke dalam modul-modul kecil yang saling terintegrasi namun independen. Dengan cara ini, pengembang dapat mempercepat proses pengembangan, pengujian, hingga *deployment* tanpa mengganggu keseluruhan sistem.

Implementasi *microservices* juga memiliki keunggulan dalam hal *fault tolerance* dan pengelolaan sumber daya. Ketika satu layanan mengalami masalah, sistem secara keseluruhan tidak harus terhenti, sehingga meningkatkan keandalan dan ketersediaan (*availability*) aplikasi. Penelitian oleh Ren dkk. (2023) menunjukkan bahwa arsitektur *microservices* sangat cocok digunakan dalam sistem edukasi digital karena mampu memisahkan layanan seperti otentikasi, pengelolaan konten, dan pemrosesan data pengguna menjadi komponen-komponen terpisah yang dapat diskalakan secara dinamis.

Lebih lanjut, dalam studi oleh Owen, (2025), disoroti bahwa *backend* yang kuat tidak hanya mampu menangani beban trafik yang besar, tetapi juga harus dirancang untuk mendukung interoperabilitas dengan berbagai sistem, termasuk integrasi dengan API eksternal dan layanan berbasis AI/ML. Hal ini sangat penting dalam konteks aplikasi pembelajaran berbasis *cloud*, seperti ALEA, yang memerlukan prediksi karakter aksara secara *real-time* dan menyimpan data pengguna serta hasil kuis dengan tingkat keandalan tinggi.

2.4 *Cloud Computing*

2.4.1 Definisi dan Karakteristik *Cloud Computing*

Cloud computing merupakan paradigma teknologi yang memungkinkan penyediaan layanan komputasi secara elastis dan sesuai permintaan melalui jaringan internet, tanpa memerlukan pengelolaan langsung terhadap infrastruktur fisik oleh pengguna akhir. Menurut Cui (2025), *cloud computing* menyediakan *platform* bagi organisasi untuk mengakses sumber daya komputasi seperti *server*, penyimpanan, dan perangkat lunak secara dinamis dan efisien, dengan skema

pembayaran berbasis penggunaan (*pay-as-you-go*). Pendekatan ini menawarkan keuntungan besar dalam hal efisiensi biaya, kecepatan implementasi, serta skalabilitas tinggi untuk berbagai skenario bisnis dan teknologi.

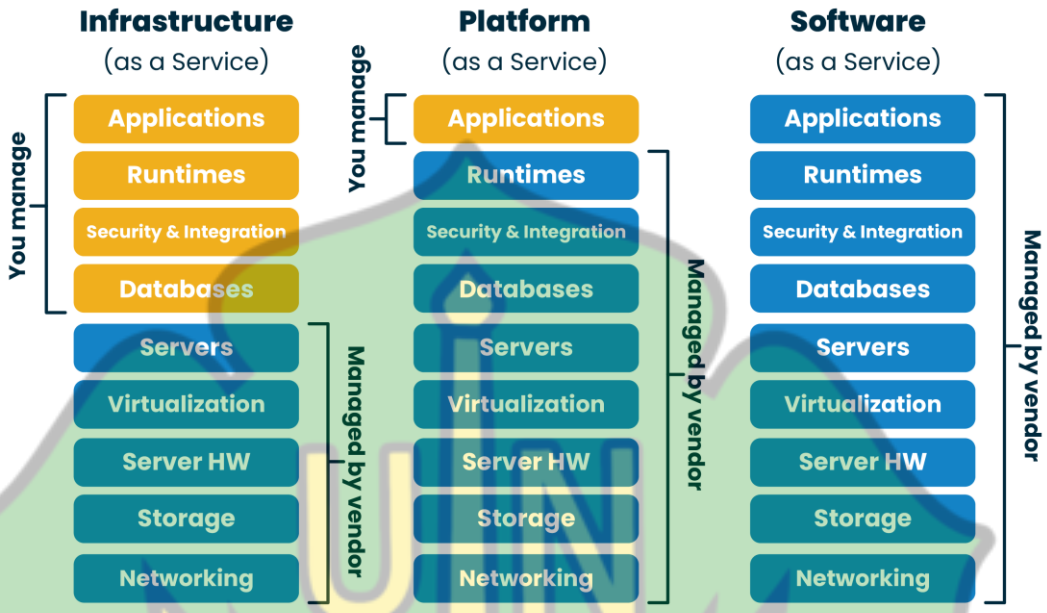
Salah satu definisi yang paling berpengaruh datang dari *National Institute of Standards and Technology* (NIST), yang menyatakan bahwa *cloud computing* memiliki lima karakteristik utama: (1) ***On-demand self-service***, yaitu kemampuan pengguna untuk secara mandiri menyediakan layanan komputasi tanpa campur tangan penyedia layanan; (2) ***Broad network access***, layanan dapat diakses melalui berbagai perangkat seperti laptop, ponsel, atau tablet menggunakan protokol standar; (3) ***Resource pooling***, penyedia menggabungkan sumber daya komputasinya untuk melayani banyak pelanggan secara dinamis melalui sistem multi-tenant; (4) ***Rapid elasticity***, sumber daya dapat secara cepat dan elastis ditambahkan atau dikurangi sesuai kebutuhan; dan (5) ***Measured service***, penggunaan sumber daya dipantau dan diukur secara otomatis untuk efisiensi dan transparansi (Younis dkk., 2024).

Karakteristik-karakteristik tersebut menjadikan *cloud computing* sebagai *platform* yang sangat fleksibel dan andal dalam mendukung berbagai aplikasi, termasuk sistem pembelajaran, *e-commerce*, layanan keuangan, dan analisis big data. Dalam penelitian oleh Alzide (2024), disebutkan bahwa kemampuan *cloud* dalam menyediakan sumber daya yang elastis serta layanan yang dapat disesuaikan dengan kebutuhan menjadikannya sangat cocok untuk pengembangan aplikasi berbasis *microservices* dan *artificial intelligence*, yang memerlukan skalabilitas dan ketersediaan tinggi.

Namun, adopsi *cloud* juga memunculkan tantangan baru, khususnya dalam aspek keamanan dan kepatuhan. Yusuf dan Ayodele (2024) menggarisbawahi bahwa dengan semakin banyaknya data sensitif yang disimpan di *cloud*, penting bagi organisasi untuk memastikan standar keamanan, seperti enkripsi, otentikasi, dan kontrol akses telah diterapkan secara menyeluruh. Penelitian ini juga menyoroti pentingnya kepatuhan terhadap regulasi internasional seperti GDPR dan ISO/IEC 27001 dalam pemanfaatan layanan *cloud*, terutama pada sektor pendidikan dan layanan publik yang menangani data privat.

2.4.2 Model Layanan *Cloud Computing*

Model layanan pada *cloud computing* yang meliputi IaaS, PaaS, dan SaaS dapat dilihat pada gambar 2.1.



Gambar 2.1 Model Layanan *Cloud Computing*

Cloud computing menyediakan tiga model layanan utama yang menjadi dasar dalam pemanfaatan teknologi *cloud*, yaitu *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), dan *Software as a Service* (SaaS). Setiap model menawarkan tingkat kontrol, fleksibilitas, dan tanggung jawab yang berbeda bagi pengguna.

- ***Infrastructure as a Service* (IaaS):** Model ini menyediakan infrastruktur TI *virtual* seperti *server*, penyimpanan, dan jaringan melalui internet. Pengguna memiliki kontrol penuh terhadap sistem operasi dan aplikasi yang dijalankan, namun tetap bergantung pada penyedia layanan untuk infrastruktur fisik. Contoh layanan IaaS meliputi *Amazon* EC2 dan *Google Compute Engine*.
- ***Platform as a Service* (PaaS):** PaaS menawarkan lingkungan pengembangan lengkap yang mencakup alat-alat untuk membangun, menguji, dan menyebarkan aplikasi tanpa perlu mengelola infrastruktur

dasar. Pengguna dapat fokus pada pengembangan aplikasi, sementara penyedia layanan menangani aspek-aspek seperti sistem operasi, *server*, dan penyimpanan. Layanan PaaS mencakup *Google App Engine* dan *Heroku*.

- ***Software as a Service* (SaaS):** SaaS memungkinkan pengguna untuk mengakses aplikasi perangkat lunak melalui internet tanpa perlu menginstal atau mengelola perangkat lunak tersebut. Penyedia layanan bertanggung jawab atas semua aspek, termasuk infrastruktur, *platform*, dan perangkat lunak. Contoh SaaS adalah *Google Workspace* dan *Microsoft Office 365*.

Dalam konteks pengembangan *backend* aplikasi pembelajaran aksara ALEA, model *Platform as a Service* (PaaS) dipilih untuk mendukung efisiensi dan skalabilitas. Layanan seperti *Google Cloud Run* digunakan untuk menyebarkan *backend* aplikasi karena kemampuannya dalam menjalankan *container* secara otomatis dan skalabel tanpa perlu pengelolaan *server* manual. Hal ini memungkinkan tim pengembang untuk fokus pada logika bisnis dan pengembangan fitur aplikasi tanpa terbebani oleh kompleksitas infrastruktur.

Studi oleh Younis dkk., (2024) menekankan bahwa PaaS menjadi pilihan utama dalam pengembangan aplikasi *modern* karena mendukung tren DevOps, integrasi *machine learning*, dan arsitektur mikroservis yang membutuhkan fleksibilitas tinggi namun tetap efisien dalam penggunaan sumber daya *cloud*. Mereka juga mencatat bahwa PaaS semakin berkembang untuk mendukung interoperabilitas antar layanan *cloud* dan penyederhanaan proses *deployment*.

2.4.3 Pembangunan Model (*Deployment Model*) dalam *Cloud Computing*

Patel dan Kansara (2021) mengkaji secara mendalam empat model *deployment* dalam *cloud computing* yang umum digunakan, yakni *Private Cloud*, *Community Cloud*, *Public Cloud*, dan *Hybrid Cloud*, sebagaimana didefinisikan oleh *National Institute of Standards and Technology* (NIST). Keempat model ini memberikan opsi berbeda yang disesuaikan dengan kebutuhan organisasi terkait kontrol, keamanan, biaya, dan fleksibilitas infrastruktur teknologi informasi.

Private cloud adalah lingkungan *cloud* yang dibangun dan digunakan secara eksklusif oleh satu organisasi. Infrastruktur *cloud* ini dapat dikelola secara internal atau oleh pihak ketiga, dan biasanya dihosting secara lokal di dalam pusat data

organisasi. Keunggulan utama dari model ini terletak pada tingkat kontrol yang tinggi terhadap data dan keamanan, menjadikannya cocok untuk institusi yang menangani data sensitif seperti lembaga keuangan, pemerintahan, atau layanan kesehatan. Menurut Patel dan Kansara (2021), organisasi dengan kebijakan kepatuhan dan regulasi ketat cenderung memilih model ini karena memungkinkan penyesuaian sistem keamanan secara spesifik.

Community cloud, di sisi lain, digunakan bersama oleh beberapa organisasi yang memiliki kepentingan, persyaratan, atau misi yang sama. Biasanya model ini digunakan dalam konteks kolaboratif seperti antar universitas, lembaga riset, atau instansi pemerintah. Model ini memungkinkan berbagi infrastruktur dan kebijakan keamanan antar anggota komunitas, sehingga efisien dari sisi biaya dan kolaborasi, sambil tetap mempertahankan batasan akses yang relevan.

Public cloud adalah model *cloud computing* di mana infrastruktur dan layanan disediakan oleh penyedia layanan pihak ketiga, seperti *Google Cloud*, *Amazon Web Services* (AWS), dan *Microsoft Azure*, dan dapat diakses secara publik melalui internet. Model ini menawarkan skalabilitas dan fleksibilitas yang tinggi dengan biaya yang relatif rendah karena sifatnya berbasis langganan dan sumber daya bersama. Patel dan Kansara (2021) menyebutkan bahwa *public cloud* sangat ideal untuk aplikasi yang membutuhkan waktu implementasi cepat, tidak terlalu sensitif terhadap keamanan, dan ingin memanfaatkan model *pay-as-you-go* yang efisien.

Hybrid cloud menggabungkan dua atau lebih model *cloud* yang berbeda (misalnya *private* dan *public cloud*) untuk memungkinkan portabilitas data dan aplikasi di antara lingkungan yang berbeda tersebut. *Hybrid cloud* memungkinkan organisasi untuk menyimpan data sensitif di *private cloud*, sembari memanfaatkan skalabilitas dan efisiensi *public cloud* untuk beban kerja yang lebih ringan atau bersifat publik. Studi Patel dan Kansara (2021) menekankan bahwa *hybrid cloud* menjadi model pilihan populer saat ini karena mampu menyeimbangkan performa, keamanan, dan efisiensi biaya. Hal ini sangat relevan dalam konteks pengembangan sistem yang kompleks, seperti sistem pembelajaran digital atau layanan berbasis AI, yang membutuhkan kombinasi antara data privat dan layanan *cloud* publik yang fleksibel.

Dalam konteks pengembangan aplikasi pembelajaran aksara berbasis *Google Cloud*, model *public cloud* sangat relevan karena layanan seperti *Cloud Run*, *Cloud Storage*, dan *Cloud* SQL tersedia sebagai bagian dari infrastruktur publik yang dapat diakses secara luas dan dikelola tanpa infrastruktur fisik sendiri. Namun demikian, prinsip-prinsip keamanan dan privasi tetap dijaga melalui kebijakan akses berbasis peran dan sistem autentikasi yang terintegrasi.

2.4.4 *Cloud Computing* untuk Mendukung Proses Bisnis dan Efisiensi Operasional

Fernandez dkk. (2020) dalam penelitiannya menjelaskan bahwa adopsi *cloud computing* telah membawa dampak signifikan terhadap transformasi digital dalam organisasi, khususnya dalam hal pengelolaan proses bisnis dan peningkatan efisiensi operasional. Dengan memanfaatkan arsitektur *cloud*, organisasi dapat mengakses layanan TI seperti penyimpanan data, pengolahan informasi, hingga perangkat lunak bisnis secara fleksibel dan sesuai kebutuhan (*on-demand*), tanpa perlu melakukan investasi awal yang besar terhadap infrastruktur fisik. Hal ini memungkinkan bisnis, termasuk skala kecil dan menengah, untuk bersaing lebih kompetitif di era digital karena mereka tidak lagi dibatasi oleh keterbatasan sumber daya TI internal.

Lebih lanjut, Fernandez dkk. menekankan bahwa *cloud computing* secara langsung berkontribusi dalam mempercepat siklus inovasi di lingkungan bisnis. Hal ini dicapai melalui kemudahan integrasi antara layanan *cloud* dengan alat kolaborasi digital, seperti *email* berbasis *cloud*, dokumen bersama, *dashboard* analitik *real-time*, serta akses *remote* ke aplikasi penting dari berbagai lokasi dan perangkat. Hasilnya, tim yang tersebar secara geografis dapat bekerja lebih efektif dan responsif terhadap perubahan pasar.

Selain keuntungan dalam fleksibilitas dan kolaborasi, *cloud computing* juga menghadirkan potensi besar dalam penghematan biaya operasional. Organisasi tidak perlu lagi menganggarkan dana besar untuk pengadaan *server* fisik, pendingin ruangan, atau tim teknis yang mengelola infrastruktur secara manual. Dalam model langganan *pay-as-you-go* yang ditawarkan oleh penyedia *cloud*, pengguna hanya

membayar untuk kapasitas yang benar-benar digunakan. Ini membantu mencegah pemborosan sumber daya dan meningkatkan efisiensi pengeluaran.

Namun, Fernandez dkk. juga menggarisbawahi adanya tantangan serius yang perlu diperhatikan dalam implementasi *cloud computing*, yaitu isu terkait keamanan data dan privasi. Dalam lingkungan *cloud*, data sensitif organisasi disimpan dan dikelola oleh pihak ketiga (penyedia *cloud*), sehingga menimbulkan risiko terkait akses tidak sah, kebocoran data, hingga pelanggaran kepatuhan terhadap regulasi perlindungan data (seperti GDPR atau HIPAA). Oleh karena itu, pengelolaan risiko serta penerapan strategi keamanan data yang kuat menjadi sangat krusial. Ini mencakup penerapan enkripsi, otentikasi multi-faktor, pemantauan aktivitas pengguna, serta kebijakan akses berbasis peran (*role-based access control*).

Secara keseluruhan, penelitian Fernandez dkk. menyimpulkan bahwa kesuksesan adopsi *cloud computing* dalam proses bisnis sangat bergantung pada kesiapan organisasi untuk beradaptasi dengan perubahan teknologi, membangun arsitektur keamanan yang memadai, dan memanfaatkan fleksibilitas *cloud* sebagai alat untuk inovasi berkelanjutan. Dalam konteks pendidikan atau aplikasi pembelajaran digital seperti ALEA, konsep ini juga sangat relevan karena memungkinkan sistem untuk berjalan secara efisien, responsif terhadap beban pengguna, dan tetap aman dalam mengelola data pengguna serta materi pembelajaran digital.

2.5 *Google Cloud*

Google Cloud Platform (GCP) merupakan salah satu layanan *cloud computing* yang banyak diadopsi dalam pengembangan aplikasi berskala besar, terutama yang membutuhkan performa tinggi, keamanan, dan kemampuan skalabilitas yang optimal. Berdasarkan studi yang dilakukan oleh Silva dkk., (2021), GCP menonjol dengan berbagai keunggulan, termasuk integrasi layanan yang komprehensif, kemudahan dalam mengelola infrastruktur, serta tingkat keamanan yang tinggi melalui enkripsi data dan sistem kontrol akses yang ketat. Salah satu layanan unggulan GCP adalah *BigQuery*, sebuah platform analisis data berskala besar yang memungkinkan pemrosesan *query* dalam hitungan detik

meskipun pada *dataset* yang sangat besar dan kompleks. Selain itu, *Compute Engine* mendukung virtualisasi infrastruktur yang memungkinkan pengembang untuk mengelola dan menjalankan aplikasi dalam lingkungan *server* virtual dengan fleksibilitas tinggi.

Keunggulan GCP lainnya adalah dukungannya terhadap pengembangan *backend* yang membutuhkan integrasi berbagai layanan seperti basis data terkelola (*Cloud* SQL, *Firestore*), sistem autentikasi (*Firebase Authentication*), serta kemampuan *deployment* aplikasi melalui *Cloud Run* dan *Kubernetes Engine*. Dalam konteks aplikasi pembelajaran aksara yang mengelola *dataset* teks dan citra aksara kuno yang besar dan bervariasi, kemampuan GCP dalam menangani *volume* data besar secara efisien sangat penting. Dengan memanfaatkan fitur *auto-scaling* dan *load balancing*, GCP dapat menjaga performa aplikasi tetap stabil meskipun terjadi lonjakan pengguna atau beban proses secara tiba-tiba (Silva dkk., 2021).

Selain itu, *Google Cloud* juga mendukung penggunaan teknologi *machine learning* melalui layanan seperti *Vertex AI*, yang memungkinkan integrasi model AI secara mudah dalam aplikasi pembelajaran, sehingga sangat relevan untuk pengembangan sistem yang mengotomatiskan pengenalan karakter aksara tradisional (Silva dkk., 2021).

2.5.1 *Cloud Run*

Cloud Run adalah layanan *serverless container* dari *Google Cloud* yang memungkinkan pengembang menjalankan aplikasi berbasis HTTP dalam lingkungan *container* tanpa perlu mengelola infrastruktur *server* secara manual. Layanan ini mendukung skala otomatis berdasarkan permintaan dan memungkinkan proses *Continuous Integration/Continuous Delivery* (CI/CD) yang efisien untuk pengembangan aplikasi modern.

Dalam konteks aplikasi pembelajaran aksara seperti ALEA, *Cloud Run* digunakan untuk menjalankan layanan *backend* yang menangani permintaan pengguna, termasuk autentikasi, pengambilan soal, dan pemrosesan prediksi aksara dari model *machine learning*. Kemampuan *auto-scaling* dari *Cloud Run* memungkinkan aplikasi menyesuaikan kapasitas komputasi secara dinamis saat

terjadi lonjakan trafik pengguna, yang sangat penting untuk menjaga performa tetap stabil saat digunakan secara bersamaan oleh banyak siswa.

Penelitian oleh Burkat dkk., (2020) menunjukkan bahwa *platform serverless container* seperti *Cloud Run* dapat secara signifikan meningkatkan efisiensi dan skalabilitas dalam menjalankan *scientific workflows*. Studi ini menyoroti bagaimana *Cloud Run* memungkinkan eksekusi tugas-tugas komputasi yang kompleks dengan latensi rendah dan biaya yang lebih efisien dibandingkan dengan pendekatan tradisional. Hal ini menjadikan *Cloud Run* sebagai pilihan yang tepat untuk aplikasi yang membutuhkan pemrosesan data secara *real-time* dan skalabilitas tinggi.

Lebih lanjut, penelitian oleh Chadha dkk., (2022) membandingkan performa antara arsitektur *microservices* tradisional dan pendekatan *serverless* menggunakan *Cloud Run* dalam konteks *platform* IoT. Hasilnya menunjukkan bahwa migrasi ke *Cloud Run* tidak hanya menyederhanakan pengelolaan infrastruktur tetapi juga mengurangi biaya operasional secara signifikan, tanpa mengorbankan kinerja aplikasi. Temuan ini mendukung penggunaan *Cloud Run* dalam pengembangan aplikasi pendidikan yang membutuhkan fleksibilitas dan efisiensi biaya.

2.5.2 *Cloud SQL*

Cloud SQL adalah layanan basis data relasional terkelola dari *Google Cloud* yang mendukung MySQL, PostgreSQL, dan SQL *Server*. Dalam pengembangan aplikasi ALEA, *Cloud* SQL digunakan untuk menyimpan data pengguna, soal kuis, dan hasil interaksi pengguna dalam bentuk tabel relasional yang terstruktur.

Penelitian oleh Zhou dan Wu (2022) menyoroti tantangan dalam mengelola data temporal di lingkungan *Cloud* SQL. Mereka mengusulkan pendekatan pengindeksan khusus untuk kueri temporal, yang menghasilkan peningkatan kinerja sebesar 25% dalam eksekusi kueri kompleks berbasis waktu.

Selain itu, studi oleh Patel dan Desai (2022) mengeksplorasi dampak teknik kompresi data terhadap kinerja kueri SQL di basis data *cloud*. Mereka menemukan bahwa penerapan algoritma kompresi *lossless* sebelum eksekusi kueri dapat

mengurangi operasi I/O dan mempercepat pemrosesan kueri, dengan peningkatan kinerja hingga 35% dalam skenario tertentu.

2.5.3 *Cloud Storage*

Cloud Storage adalah layanan penyimpanan objek dari *Google Cloud* yang dirancang untuk menyimpan dan mengakses data dalam skala besar. Dalam proyek ALEA, *Cloud Storage* digunakan untuk menyimpan gambar soal aksara dan *file* model prediksi aksara.

Penelitian oleh Irawan (2024) menunjukkan bahwa penerapan *Google Cloud* sebagai solusi penyimpanan data di industri dapat meningkatkan aksesibilitas data, manajemen, dan analisis, serta mendukung pengambilan keputusan yang lebih baik. Studi ini menggunakan pendekatan studi kasus dan menemukan bahwa implementasi *Cloud Storage* dapat meningkatkan efisiensi operasional dan hasil bisnis.

Selain itu, Krichevsky dkk., (2021) mengevaluasi kinerja pelatihan *deep learning* terdistribusi yang memanfaatkan data yang sepenuhnya disimpan di *bucket Cloud Storage*. Mereka mengusulkan teknik *caching* dan *pre-fetching* untuk mengatasi keterbatasan *bandwidth*, yang berhasil mengurangi waktu tunggu pelatihan hingga 93,5% dibandingkan dengan pemuatan langsung dari *bucket*, sehingga mencapai kinerja yang sebanding dengan pemuatan data langsung dari disk.

Dengan latensi rendah, *throughput* tinggi, dan fitur keamanan seperti enkripsi data saat transit dan saat disimpan, *Cloud Storage* menjadi pilihan ideal untuk mendukung aplikasi berbasis media seperti ALEA yang membutuhkan interaksi cepat dengan *file* eksternal.

2.6 *Javascript*

Javascript merupakan bahasa pemrograman yang awalnya dirancang untuk manipulasi antarmuka web pada sisi klien (*client-side*), namun kini telah berevolusi menjadi bahasa pemrograman yang sangat serbaguna, terutama dalam pengembangan aplikasi *backend* menggunakan lingkungan *runtime Node.js*. Studi oleh Xu dkk., (2020) mengungkapkan bahwa *Node.js* memungkinkan

pengembangan aplikasi web yang sangat responsif dan *scalable* dengan memanfaatkan *event-driven*, *non-blocking I/O model* yang efisien dalam menangani banyak permintaan secara bersamaan. Ini menjadikan *Javascript*, khususnya dengan *Node.js*, sangat ideal untuk aplikasi berbasis *cloud* yang memerlukan performa tinggi dan latensi rendah.

Dalam pengembangan aplikasi pembelajaran aksara yang berbasis *cloud*, *Javascript* memegang peranan penting dalam memastikan aplikasi tetap responsif dan mudah dikembangkan. *Backend* yang dibangun dengan *Node.js* dapat mengelola komunikasi data secara *real-time*, seperti pengiriman data pengguna dan respons API, sehingga pengalaman pengguna menjadi lebih lancar dan interaktif. Selain itu, ekosistem *Node Package Manager* (NPM) menyediakan berbagai pustaka yang mempercepat pengembangan fitur, mulai dari pengelolaan *database* hingga integrasi dengan layanan *cloud* seperti *Google Cloud* (Xu dkk., 2020).

Keunggulan lain dari penggunaan *Javascript* di *backend* adalah kemudahan pengembangan *full-stack* menggunakan satu bahasa pemrograman yang sama, sehingga meningkatkan produktivitas tim pengembang. Dalam konteks aplikasi pembelajaran aksara, hal ini memungkinkan integrasi yang mulus antara *frontend* interaktif dan *backend* yang kuat, memberikan performa optimal dan kemudahan dalam pemeliharaan aplikasi (Xu dkk., 2020).

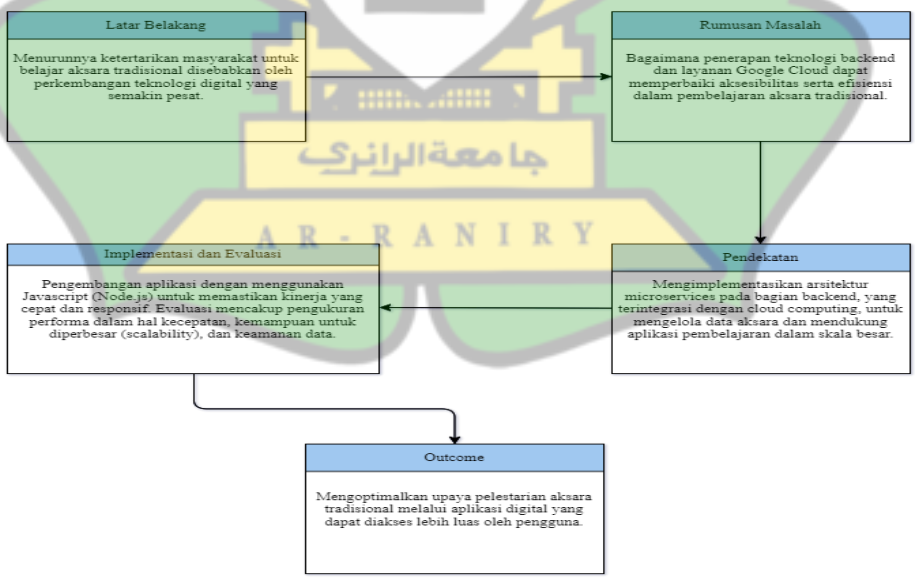
Selain itu, *JavaScript* juga mendukung pendekatan arsitektur mikroservis yang umum digunakan dalam pengembangan aplikasi *modern* berbasis *cloud*. Menurut penelitian oleh Rahman dkk., (2023), penggunaan *JavaScript* dalam konteks mikroservis memungkinkan setiap layanan dikembangkan secara independen dengan dependensi yang terkelola baik melalui modul-modul *Node.js*. Hal ini memberikan fleksibilitas dalam mengatur skala aplikasi secara horizontal, meningkatkan keandalan sistem, serta mempermudah debugging dan pengujian unit secara terpisah. Dalam pengembangan aplikasi ALEA, pendekatan ini dapat digunakan untuk memisahkan layanan otentikasi pengguna, pengelolaan soal kuis, dan pengolahan data aksara, yang masing-masing dapat ditangani oleh modul *backend* tersendiri yang dikembangkan dengan *JavaScript*.

JavaScript juga memainkan peran penting dalam integrasi dengan layanan *cloud* dan komputasi *modern*, khususnya melalui penggunaan API REST dan

GraphQL yang dibangun menggunakan *framework* seperti *Express.js*. *Framework* ini memudahkan pengelolaan *routing*, *middleware*, dan autentikasi secara efisien dalam arsitektur *backend*. Menurut studi oleh Zhang dkk., (2021), penggunaan *Express.js* dalam proyek berbasis *Node.js* memungkinkan peningkatan efisiensi pengembangan karena struktur yang modular dan kemampuannya untuk berintegrasi langsung dengan layanan *cloud* seperti *Firebase*, *Google Cloud Storage*, dan *BigQuery*. Dalam aplikasi ALEA, *Express.js* digunakan sebagai kerangka utama untuk merancang *endpoint* REST API yang menghubungkan *frontend* dengan layanan *backend* seperti penyimpanan gambar kuis dan pemrosesan prediksi aksara. Kemampuan integrasi ini mempercepat proses pengembangan dan memberikan fleksibilitas tinggi dalam penerapan fungsionalitas-fungsionalitas baru.

2.7 Kerangka Teoritis

Kerangka Pemikiran adalah struktur paradigma yang digunakan oleh penulis sebagai dasar untuk memahami, menganalisis, dan menyelesaikan masalah. Dalam kerangka ini, berbagai informasi dan konsep dirangkai secara logis menjadi satu kesatuan. Adapun Kerangka Teoritis dapat dilihat pada gambar 2.2



Gambar 2.2 Kerangka Teoritis

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penilitian

Penelitian ini menggunakan pendekatan *Research and Development* (R&D) yang berfokus pada proses perancangan, pembangunan, dan pengujian infrastruktur *backend* berbasis *Google Cloud Platform* (GCP) untuk mendukung aplikasi pembelajaran aksara *Aksara Learning Apps* (ALEA).

Menurut Creswell (2021), metode R&D mencakup kegiatan sistematis mulai dari perancangan, pengembangan, pengujian, hingga penyempurnaan suatu produk agar sesuai dengan kebutuhan pengguna. Dalam konteks penelitian ini, hasil yang dikembangkan bukan hanya berupa rancangan teoretis, melainkan sistem *backend* yang benar-benar dibangun dan diuji di lingkungan *cloud* nyata.

Proses pengembangan dilakukan secara iteratif, yaitu melalui siklus perancangan, implementasi, evaluasi, dan penyempurnaan. Setiap tahap pengujian memberikan masukan yang digunakan untuk meningkatkan kinerja dan efisiensi sistem. Pendekatan ini memungkinkan sistem *backend* ALEA berkembang secara bertahap hingga mencapai konfigurasi yang stabil dan optimal.

Tahapan pengembangan dilakukan dengan membangun langsung infrastruktur *backend* di *Google Cloud Platform* yang terdiri atas beberapa layanan utama:

- *Cloud Run*, sebagai *platform serverless* untuk menjalankan *container* aplikasi berbasis Node.js dan Express.js,
- *Cloud* SQL, sebagai basis data relasional untuk menyimpan data pengguna dan hasil pembelajaran,
- *Cloud Storage*, untuk menyimpan file gambar aksara dan model pembelajaran mesin berbasis YOLO, serta
- *Cloud Build*, untuk mengotomatisasi proses *deployment* melalui *pipeline Continuous Integration/Continuous Deployment* (CI/CD).

Selain itu, penelitian ini juga mengintegrasikan model *Machine Learning* berbasis YOLO yang berfungsi untuk mengenali dan memprediksi aksara dari citra gambar. Model disimpan dalam format .pt dan dijalankan menggunakan *framework*

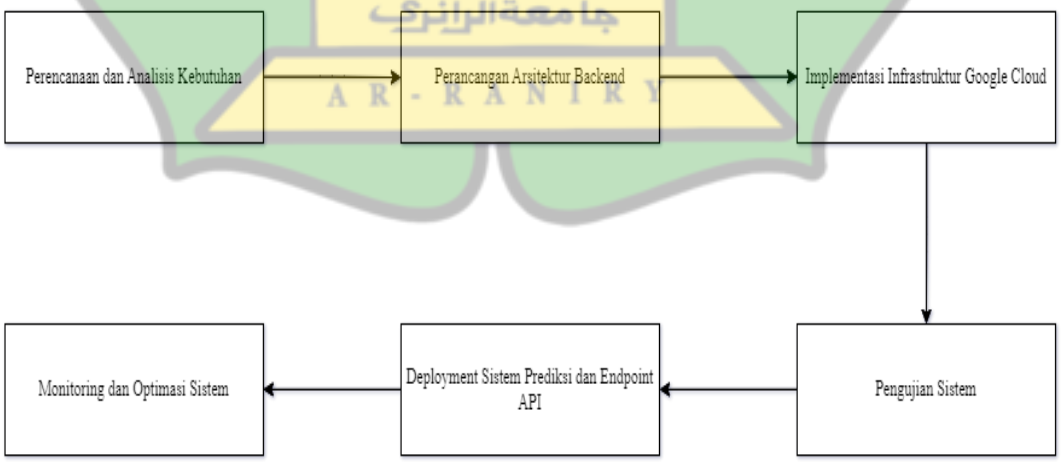
PyTorch, kemudian dihubungkan dengan *backend* melalui *endpoint* API. *Backend* memproses input gambar yang dikirim pengguna, mengoperasikannya ke model YOLO di *server cloud*, dan mengembalikan hasil prediksi secara *real-time* melalui API yang di-*host* di *Cloud Run*.

Melalui pendekatan R&D ini, seluruh proses mulai dari desain arsitektur, pembangunan infrastruktur *backend*, integrasi model, hingga pengujian performa dilakukan secara langsung pada lingkungan *Google Cloud*. Pendekatan ini memastikan sistem yang dibangun tidak hanya berfungsi sesuai rancangan, tetapi juga memiliki performa yang efisien, skalabel, dan siap digunakan pada skala produksi.

3.2 Tahapan Penilitian

Penelitian ini dilaksanakan melalui beberapa tahapan yang menggambarkan proses pengembangan sistem *backend* berbasis *Google Cloud Platform* (GCP). Setiap tahap saling berkaitan dan dilakukan secara iteratif menggunakan pendekatan *System Development Life Cycle* (SDLC) yang dikelola dengan metode Agile. Pendekatan ini memungkinkan pengembang untuk melakukan perbaikan dan penyempurnaan sistem secara berkelanjutan berdasarkan hasil pengujian dan umpan balik pada setiap siklus pengembangan.

Adapun tahapan penelitian ini dapat dilihat pada Gambar 3.1, yang memperlihatkan alur pengembangan sistem mulai dari tahap perencanaan hingga proses *monitoring* dan optimasi sistem.



Gambar 3.1 Tahapan Penelitian

3.2.1 Perencanaan dan Analisis Kebutuhan

Tahap pertama bertujuan untuk mengidentifikasi kebutuhan fungsional dan non-fungsional dari sistem *backend Aksara Learning Apps* (ALEA). Analisis ini meliputi perumusan fitur utama yang harus disediakan oleh *backend*, spesifikasi teknis, serta penentuan layanan *cloud* yang akan digunakan seperti *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*.

Selain itu, dilakukan juga peninjauan terhadap model *Machine Learning* YOLO (.pt) yang akan diintegrasikan agar kompatibel dengan arsitektur *backend*.

3.2.2 Perancangan Arsitektur *Backend*

Pada tahap ini dilakukan perancangan struktur dan komponen *backend* menggunakan pendekatan *microservices*. Desain arsitektur meliputi diagram alur komunikasi antara *backend* dengan model YOLO, pengelolaan data pengguna, serta interaksi antara *frontend* dan API di lingkungan *Google Cloud*.

Perancangan dilakukan secara fleksibel agar sistem dapat diskalakan dan mudah dioptimasi pada tahap selanjutnya.

3.2.3 Implementasi Infrastruktur *Google Cloud*

Tahapan ini merupakan proses pembangunan infrastruktur *backend* secara langsung di *Google Cloud Platform*.

Beberapa layanan utama yang digunakan antara lain:

- *Cloud Run* untuk menjalankan *container* aplikasi berbasis Node.js secara *serverless*,
- *Cloud* SQL untuk menyimpan data pengguna dan hasil prediksi,
- *Cloud Storage* untuk menyimpan dataset gambar dan model YOLO (.pt), serta
- *Cloud Build* sebagai alat otomatisasi *pipeline Continuous Integration and Continuous Deployment* (CI/CD).

Pada tahap ini dilakukan pula konfigurasi koneksi antar layanan *cloud* agar sistem dapat berjalan terintegrasi.

3.2.4 Pengujian Sistem

Tahapan ini dilakukan untuk memastikan bahwa seluruh komponen backend berjalan sesuai fungsinya.

Pengujian meliputi:

- Uji Fungsionalitas, untuk memastikan setiap e*ndpoint* API berfungsi dengan benar.
- Uji Kinerja, untuk mengukur kecepatan dan efisiensi layanan *Cloud Run* saat menerima permintaan prediksi YOLO.

Hasil dari tahap ini digunakan sebagai dasar untuk melakukan perbaikan sistem.

3.2.5 Deployment Sistem Prediksi dan *Endpoint* API

Setelah seluruh proses pengujian selesai dan sistem backend dinyatakan berfungsi dengan baik, tahap berikutnya adalah deployment atau penerapan sistem ke lingkungan produksi.

Pada tahap ini, skrip prediksi yang memuat model YOLO (.pt) diimplementasikan ke dalam layanan *Google Cloud Run* agar dapat diakses secara *real-time* oleh aplikasi *Aksara Learning Apps* (ALEA). Proses *deployment* dilakukan secara otomatis melalui *pipeline Continuous Integration/Continuous Deployment* (CI/CD) yang dibangun menggunakan *Google Cloud Build*, dan terhubung langsung dengan repositori kode di GitHub. Dengan mekanisme ini, setiap perubahan kode yang dikirim (*push*) ke repositori akan secara otomatis memicu proses *build*, pengujian, dan *deployment* ke *Cloud Run* tanpa perlu dilakukan secara manual.

Proses deployment meliputi beberapa langkah utama sebagai berikut:

- Kontainerisasi Skrip Prediksi: Seluruh kode yang menjalankan model YOLO (.pt) dan proses prediksi dikemas ke dalam sebuah *container* menggunakan Docker. *Container* ini memuat seluruh dependensi yang diperlukan, termasuk PyTorch, OpenCV, serta pustaka pendukung lainnya yang dibutuhkan untuk menjalankan model secara efisien di lingkungan *cloud*.

- *Deployment* ke *Cloud Run*: Setelah proses kontainerisasi selesai, *container* diunggah dan dijalankan melalui layanan *Google Cloud Run*. *Cloud Run* memungkinkan aplikasi *backend* dan model prediksi berjalan dalam mode *serverless*, di mana sistem akan otomatis menyesuaikan kapasitas komputasi berdasarkan jumlah permintaan pengguna. Pendekatan ini menjamin layanan tetap stabil, efisien, dan mudah diskalakan tanpa perlu pengelolaan *server* secara manual.
- Penyediaan *Endpoint* API: *Endpoint* API kemudian disediakan untuk memungkinkan *frontend* mengirimkan data gambar aksara dan menerima hasil prediksi dari model YOLO. API ini dikembangkan menggunakan *framework* Express.js dan dioptimalkan agar dapat memberikan respons dengan waktu yang cepat dan keandalan tinggi. Sebelum diaktifkan ke publik, API ini diuji menggunakan alat seperti *Postman* untuk memastikan performanya stabil, aman, dan dapat menangani berbagai skenario input dari pengguna.

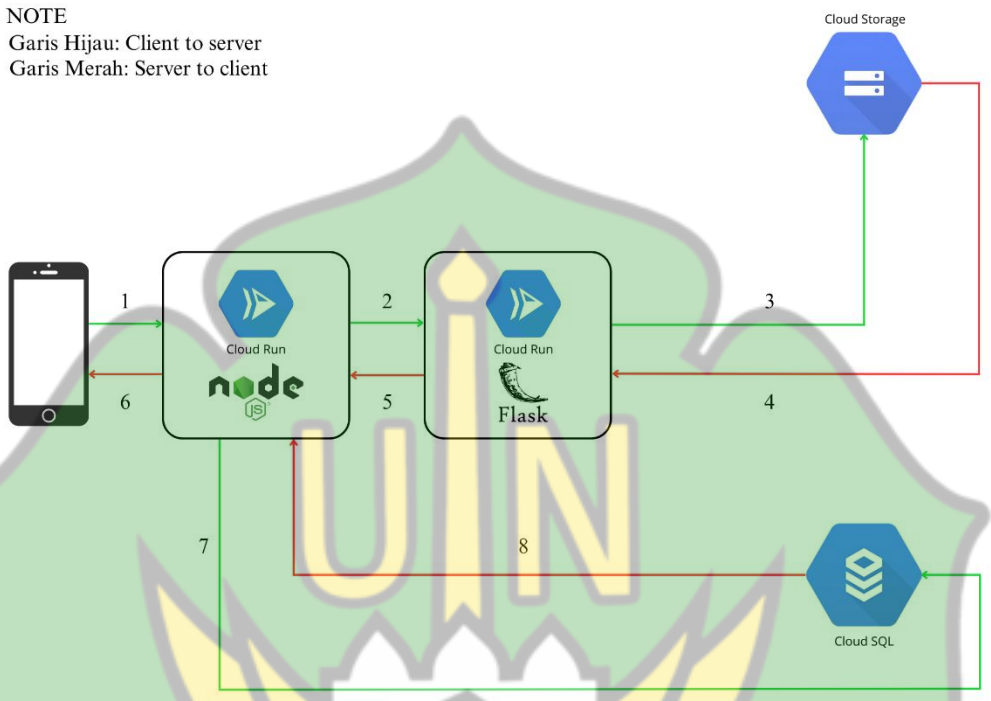
Dengan proses *deployment* ini, sistem *backend* ALEA dapat beroperasi sepenuhnya di lingkungan *Google Cloud*, memberikan layanan prediksi aksara secara *real-time* dengan dukungan arsitektur yang terukur, efisien, dan mudah dikelola.

3.2.6 *Monitoring* dan Optimasi Sistem

Setelah *deployment*, *monitoring system* dilakukan secara *real-time* menggunakan *Google Cloud Monitoring*. *Monitoring* ini memungkinkan tim pengembang untuk memantau peforma layanan, seperti penggunaan CPU, memori, serta jumlah permintaan yang diterima oleh API prediksi. Jika terdapat kegagalan atau sesuatu yang lain, notifikasi otomatis akan dikirim ke tim pengembang untuk segera dilakukan tindakan perbaikan. Selain itu, optimasi *system* dilakukan berdasarkan data *monitoring*, seperti mengatur *auto-scaling* di *Cloud Run* untuk menangani lonjakan permintaan yang tinggi.

3.3 Arsitektur Sistem *Backend*

Arsitektur sistem *backend* pada aplikasi ALEA dapat diliat pada gambar 3.2 dibawah ini



Gambar 3.2 Arsitektur *Backend* ALEA

Arsitektur *backend Aksara Learning Apps* (ALEA) dirancang menggunakan pendekatan *microservices*, di mana setiap layanan dikembangkan secara terpisah namun tetap dapat saling berkomunikasi melalui *Application Programming Interface* (API).

Layanan utama mencakup manajemen pengguna, pengelolaan data aksara, layanan kuis interaktif, serta layanan prediksi aksara yang terintegrasi dengan model *Machine Learning* YOLO. Pendekatan *microservices* ini memberikan fleksibilitas tinggi dalam pengembangan dan pemeliharaan sistem, serta mendukung kemampuan *auto-scaling* ketika terjadi peningkatan jumlah pengguna.

Untuk menjalankan layanan-layanan tersebut, ALEA memanfaatkan *Google Cloud Run* sebagai *platform* utama. Setiap layanan dikemas dalam *container* Docker dan di-*deploy* secara independen, sehingga *Cloud Run* dapat

menyesuaikan kapasitas sumber daya secara otomatis berdasarkan beban kerja. Layanan *backend* dikembangkan menggunakan Node.js dan Express.js, yang berfungsi menangani pengelolaan data aksara, kuis, serta komunikasi data antara aplikasi *frontend* dan sistem prediksi. Dengan konfigurasi ini, ALEA dapat memberikan respons yang cepat, stabil, dan efisien meskipun diakses oleh banyak pengguna secara bersamaan.

Berdasarkan Gambar 3.2, arsitektur *backend Aksara Learning Apps* (ALEA) dirancang menggunakan pendekatan *microservices* yang memisahkan layanan *backend* utama dan layanan prediksi aksara berbasis *machine learning*. Pada gambar tersebut, aplikasi Android bertindak sebagai *client* yang berkomunikasi langsung dengan *backend* utama berbasis Node.js yang dijalankan pada layanan *Cloud Run*.

Pada panah (1) ditunjukkan alur permintaan (*request*) dari aplikasi Android menuju *backend* Node.js. Layanan Node.js ini menyimpan dan mengelola beberapa *endpoint* utama, yaitu /kamus, /sejarah, /kuis, /materi, dan /predict, yang berfungsi sebagai *entry point* seluruh proses layanan sistem ALEA.

Apabila permintaan pengguna bukan ditujukan ke *endpoint* /predict, maka proses akan langsung dilanjutkan ke layanan *database* melalui panah (7), di mana *backend* melakukan pengambilan atau penyimpanan data ke *Cloud* SQL. Hasil pemrosesan data tersebut kemudian dikembalikan kembali ke *backend* melalui panah (8) untuk selanjutnya diteruskan ke aplikasi pengguna.

Sebaliknya, apabila permintaan pengguna mengakses *endpoint* /predict, maka *backend* Node.js akan meneruskan data gambar ke layanan prediksi aksara berbasis Flask yang juga berjalan di *Cloud Run* melalui panah (2). Layanan Flask selanjutnya berkomunikasi dengan *Cloud Storage* untuk mengakses *file* model YOLO dan data pendukung melalui panah (3), serta menerima kembali data yang diperlukan melalui panah (4).

Setelah proses inferensi selesai, hasil prediksi aksara akan dikirim kembali dari layanan Flask ke *backend* Node.js melalui panah (5). Selanjutnya, *backend* meneruskan hasil tersebut ke aplikasi Android sebagai respons akhir melalui panah (6).

Dengan alur seperti yang ditunjukkan pada Gambar 3.2, layanan prediksi hanya dijalankan saat *endpoint* /predict diakses, sedangkan layanan lainnya tetap ditangani langsung oleh *backend* utama. Pendekatan ini membuat sistem ALEA lebih efisien, terstruktur, serta mudah dikembangkan di masa mendatang.

Secara keseluruhan, arsitektur *backend* ALEA dirancang untuk mencapai stabilitas, efisiensi, dan skalabilitas tinggi. Dengan menggabungkan dua layanan *Cloud Run* (*backend* utama dan model prediksi YOLO), serta dukungan *Cloud* SQL dan *Cloud Storage*, sistem ALEA mampu memberikan layanan pembelajaran aksara secara *real-time* yang andal, fleksibel, dan mudah dikembangkan di masa mendatang.

3.4 Waktu dan Lokasi Penelitian

Penelitian ini direncanakan berlangsung selama empat bulan, dimulai pada Juni 2025 hingga September 2025. Selama periode tersebut, kegiatan penelitian dibagi menjadi beberapa fase, yang meliputi perancangan arsitektur *backend*, pengembangan layanan berbasis *Google Cloud Platform* (GCP), integrasi model *Machine Learning* (ML), pengujian sistem, dan *deployment* ke lingkungan produksi.

Fase awal, yaitu perencanaan dan analisis kebutuhan, dilakukan pada bulan pertama. Tahapan ini mencakup identifikasi kebutuhan sistem, pemetaan fitur utama, serta konsultasi dengan tim pengembang *Machine Learning* untuk memastikan integrasi model YOLO (.pt) dapat dilakukan dengan optimal.

Fase implementasi dan integrasi sistem dilakukan pada bulan kedua hingga bulan ketiga. Pada tahap ini, *backend* dibangun menggunakan layanan *Google Cloud Run*, *Cloud* SQL, dan *Cloud Storage* sebagai infrastruktur utama. *Backend* dikembangkan dengan Node.js dan Express.js, sementara integrasi model YOLO dilakukan menggunakan PyTorch agar sistem dapat melakukan prediksi aksara secara *real-time* di *cloud*.

Kegiatan penelitian dilakukan secara *remote* dengan memanfaatkan fleksibilitas infrastruktur *cloud* dari *Google Cloud Platform*. Pendekatan ini memungkinkan kolaborasi jarak jauh yang efektif antara pengembang *backend* dan tim *Machine Learning*. Selanjutnya, pengujian sistem dan optimalisasi performa

dilakukan pada bulan keempat, dan *deployment* akhir hingga pemantauan sistem produksi dilakukan pada bulan keempat penelitian.

3.5 Populasi dan Sampel

Populasi dalam penelitian ini adalah pengguna aplikasi *Aksara Learning Apps* (ALEA) yang tertarik mempelajari aksara tradisional Nusantara seperti aksara Jawa, Bali, dan Sunda. Pengguna sasaran meliputi siswa, mahasiswa, dan masyarakat umum yang ingin belajar aksara melalui *platform* digital interaktif.

Dataset yang digunakan dalam pengembangan sistem ALEA terdiri dari gambar-gambar aksara yang telah dilabeli dan bersumber dari dataset publik seperti Kaggle. Dataset ini berfungsi sebagai bahan pelatihan untuk model prediksi berbasis YOLO. Selain dataset publik, penelitian juga melibatkan dataset tambahan yang dikumpulkan secara manual oleh tim, berisi variasi aksara yang jarang ditemukan dalam dataset umum untuk memperkaya representasi data.

Model *Machine Learning* hasil pelatihan tim dikembangkan menggunakan *framework* PyTorch dan disimpan dalam format .pt. Model ini kemudian diintegrasikan ke dalam sistem *backend* ALEA melalui skrip prediksi yang di-*deploy* di *Google Cloud Run*. *Backend* akan memanfaatkan model tersebut untuk memproses input gambar dari pengguna dan menghasilkan keluaran berupa teks aksara yang terdeteksi. Hasil prediksi ini dikirimkan kembali ke aplikasi *frontend* secara *real-time* melalui *endpoint* API.

3.6 Integrasi Model Prediksi dan *Deployment Endpoint*

Tahap ini berfokus pada proses integrasi model *Machine Learning* (YOLO) ke dalam sistem *backend* ALEA serta *deployment endpoint* API ke lingkungan *cloud*. Model YOLO (.pt) yang telah dilatih oleh tim *Machine Learning* diimplementasikan menggunakan PyTorch, kemudian diintegrasikan melalui skrip prediksi berbasis Python yang berjalan pada layanan *Google Cloud Run*.

Langkah-langkah implementasi integrasi model prediksi adalah sebagai berikut:

1. Memuat Model YOLO (.pt): *Backend* akan memuat model YOLO yang disimpan di *Cloud Storage* ke dalam *container* prediksi menggunakan

pustaka PyTorch. Model ini siap digunakan untuk memproses permintaan prediksi dari pengguna.

2. *Preprocessing Input* Gambar: Gambar aksara yang diterima dari aplikasi *frontend* akan diproses terlebih dahulu, meliputi pengubahan ukuran (*resize*) dan normalisasi agar sesuai dengan format *input* yang dibutuhkan oleh model YOLO.
3. Prediksi Aksara: Model YOLO akan melakukan deteksi objek pada gambar yang telah diproses untuk mengenali bentuk aksara. Hasil deteksi akan dikonversi menjadi label teks aksara yang sesuai, kemudian dikembalikan ke *backend*.
4. *Deployment* ke *Cloud Run*: Skrip prediksi dikemas ke dalam *container* menggunakan Docker, yang berisi seluruh dependensi seperti PyTorch dan OpenCV. *Container* tersebut kemudian di-*deploy* ke *Google Cloud Run*, memungkinkan sistem melakukan prediksi secara otomatis dan menyesuaikan kapasitas sumber daya berdasarkan jumlah permintaan (*auto-scaling*).
5. Penyediaan *Endpoint* API: *Endpoint* API disediakan untuk menerima *input* gambar dari aplikasi *frontend* dan mengembalikan hasil prediksi dalam format JSON. API ini juga diatur agar dapat berkomunikasi langsung dengan layanan *backend* utama ALEA yang berjalan di *container Cloud Run* terpisah.
6. Pengujian dan Optimasi *Endpoint*: Setelah *deployment*, *endpoint* diuji menggunakan *Postman* dan *Cloud Monitoring* untuk memastikan performa sistem stabil. Pengujian mencakup kecepatan respons, akurasi hasil prediksi, serta kemampuan sistem menangani permintaan bersamaan dalam jumlah besar.

Tahapan ini memastikan bahwa model YOLO dapat berfungsi secara efektif di lingkungan *cloud* dan memberikan hasil prediksi yang cepat, akurat, serta terintegrasi penuh dengan arsitektur *backend* ALEA.

3.7 Alat dan Bahan

Alat yang dibutuhkan untuk melakukan penelitian ini berupa perangkat keras dan perangkat lunak. Analisis alat dan kebutuhan sistem yang diperlukan untuk penelitian ini meliputi:

3.7.1 Perangkat Keras

Perangkat keras yang digunakan berupa salah satu unit Laptop Asus Tuf Gaming F15 model FX506LHB dengan spesifikasi perangkat keras tercantum pada Tabel 3.1

Tabel 3.1 Spesifikasi Perangkat Keras

NO	Komponen	Spesifikasi
1	Processor	Intel Core i5-10300H
2	RAM	16 GB DDR4 3200 MHz dual channel
3	Storage	1TB SSD M.2 NVME
4	Graphic Card	NVIDIA GeForce GTX 1650 4GB VRAM

3.7.2 Perangkat Lunak

Perangkat lunak yang di pakai dalam proses implementasi ini berupa sistem operasi *Microsoft Windows 11 Home Single Language Version 22h2* dan menggunakan beberapa *tools*, yaitu *Windows Subsystem for Linux* (WSL)*, Visual Studio Code,* NodeJS, Python, dan *Chrome*. *Tools* tersebut akan dipakai dalam penelitian yang akan dilakukan oleh penulis, dapat dilihat pada tabel 3.2.

Tabel 3.2 Jenis Perangkat Lunak

NO	Perangkat Lunak	Version
1	*Microsoft Windows 11*	22h2
2	*Windows Subsystem for Linux* (WSL)	2.3.24
3	*Visual Studio Code*	1.93.1
4	*NodeJS*	20.11.0
5	*Python*	3.11
6	*Chrome*	129.0.6668.90

BAB IV
HASIL DAN PEMBAHASAN

4.1 Perencanaan dan Analisis Kebutuhan

Tahapan ini merupakan langkah awal dari proses penelitian dan pengembangan sistem *Aksara Learning Apps* (ALEA). Pada tahap perencanaan, peneliti melakukan identifikasi terhadap kebutuhan pengguna, kebutuhan sistem, serta ruang lingkup pengembangan yang akan dilakukan. Tahap ini menjadi dasar penting agar arah implementasi *backend* dan integrasi *cloud* dapat berjalan sesuai dengan tujuan penelitian yang telah ditetapkan pada Bab I.

4.1.1 Identifikasi Masalah

Berdasarkan hasil observasi terhadap kondisi pembelajaran aksara daerah di Indonesia, ditemukan bahwa sebagian besar metode pembelajaran masih bersifat konvensional dan belum memanfaatkan teknologi digital secara maksimal. Selain itu, banyak pengguna, terutama pelajar, mengalami kesulitan dalam mengenali bentuk aksara daerah seperti aksara Bali, Lampung, Pegon, dan Sunda. Dari sisi teknologi, belum banyak sistem pembelajaran yang mengintegrasikan *Machine Learning* (ML) dan *Cloud Computing* untuk membantu proses belajar aksara secara otomatis.

Masalah tersebut menjadi dasar perancangan sistem ALEA, yaitu menciptakan *platform* pembelajaran interaktif berbasis AI yang mampu:

- mengenali gambar aksara yang diunggah pengguna,
- menampilkan hasil prediksi aksara secara cepat dan akurat,
- serta menyediakan materi pembelajaran dan latihan yang dapat diakses secara daring.

Dengan demikian, kebutuhan akan sistem *backend* yang efisien, fleksibel, dan dapat diskalakan (*scalable*) menjadi hal yang sangat penting untuk mendukung kinerja aplikasi ALEA.

4.1.2 Analisis Kebutuhan Pengguna

Tahap ini berfokus pada identifikasi siapa pengguna utama sistem dan kebutuhan apa yang mereka harapkan dari aplikasi ALEA.

Hasil analisis diperoleh melalui wawancara singkat dengan pengguna potensial serta evaluasi dari sistem pembelajaran digital yang sudah ada sebelumnya.

1. Pengguna Utama
    - Masyarakat Umum: Membutuhkan media pembelajaran digital untuk mengenal kembali aksara daerah sebagai bagian dari pelestarian budaya Nusantara.
2. Kebutuhan Fungsional dari Sisi Pengguna
    - Akses ke materi pembelajaran aksara daerah.
    - Fitur kuis untuk menguji pemahaman.
    - Fitur unggah gambar untuk mengenali aksara secara otomatis.
    - Tampilan hasil prediksi dan umpan balik belajar.
3. Kebutuhan Non-Fungsional
    - Sistem harus mudah digunakan dan responsif.
    - Layanan prediksi harus memiliki waktu tanggap selama beberapa detik.
    - Akurasi prediksi minimal 95%.
    - Data pengguna dan hasil pembelajaran tersimpan aman di cloud.
    - Layanan dapat diakses lintas platform (khususnya Android).

Dari hasil analisis tersebut, disimpulkan bahwa sistem harus mampu memberikan pengalaman belajar yang cepat, akurat, dan interaktif dengan performa tinggi di sisi *backend*.

4.1.3 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem ALEA disusun berdasarkan hasil identifikasi kebutuhan fungsional dan non-fungsional yang diperlukan dalam pengembangan *backend*. Rincian kebutuhan tersebut dapat dilihat pada Tabel 4.1, yang menjadi acuan utama dalam perancangan arsitektur sistem dan pengembangan layanan *backend*.

Tabel 4.1 Analisis Kebutuhan ALEA

NO	Jenis Kebutuhan	Deskripsi
1	Kebutuhan Perangkat Lunak	Node.js 20, Express.js, Sequelize ORM, Python 3.10, Flask, PyTorch, Docker, MySQL, *Cloud Build*, *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*.
2	Kebutuhan Perangkat Keras	*Server cloud* berbasis GCP dengan RAM minimal 1 GB per *container* dan kapasitas *storage* minimal 10 GB.
3	Kebutuhan Fungsional	Sistem dapat menerima input gambar, memproses data melalui model YOLO, menyimpan hasil ke *database*, dan menampilkan hasil prediksi melalui API.
4	Kebutuhan Non-Fungsional	Sistem memiliki kecepatan tinggi, keandalan tinggi (*uptime* > 99%), efisiensi biaya melalui *scale-to-zero*, serta keamanan data yang baik.

Selain itu, sistem ALEA juga dirancang dengan mempertimbangkan fleksibilitas untuk dikembangkan di masa mendatang. Penambahan model aksara baru dapat dilakukan hanya dengan menambahkan *container* baru tanpa harus mengubah struktur *backend* utama.

4.1.4 Spesifikasi Awal Sistem

Berdasarkan hasil analisis sebelumnya, disusunlah spesifikasi awal sistem ALEA yang digunakan sebagai dasar implementasi *backend* disajikan pada Tabel 4.2. Spesifikasi ini mencakup komponen utama sistem, platform *cloud* yang digunakan, serta kebutuhan layanan pendukung yang memastikan sistem dapat berjalan sesuai dengan kebutuhan aplikasi pembelajaran aksara.

Tabel 4.2 Spesifikasi Awal Sistem ALEA

NO	Komponen	Spesifikasi Awal
1	*Framework Backend*	Node.js + Express.js
2	*Framework Model*	Python + Flask + PyTorch (model YOLO)
3	*Database*	MySQL (*Cloud* SQL)
4	Media Penyimpanan	*Google Cloud Storage*

5	*Hosting* Layanan	*Google Cloud Run*
6	*Pipeline* Otomatis	*Cloud Build* (CI/CD)
7	Akses *Frontend*	Android *App* (via API *endpoint*)
8	*Endpoint* Utama	/*predict*, /kuis, /sejarah, /kamus

Sistem ini dirancang agar *backend* menjadi pusat komunikasi antara *frontend* Android dan layanan *Machine Learning* di *Cloud Run*. Setiap permintaan pengguna diteruskan melalui API dan dikembalikan dalam format JSON agar dapat ditampilkan langsung di aplikasi Android ALEA.

Sebagai hasil akhir dari tahap ini, peneliti telah memiliki rancangan kebutuhan yang lengkap baik dari sisi pengguna maupun sistem yang kemudian menjadi dasar pelaksanaan tahap berikutnya, yaitu perancangan arsitektur *backend* dan infrastruktur *cloud*.

4.2 Perancangan Arsitektur *Backend*

Tahap kedua dalam proses penelitian dan pengembangan sistem ALEA adalah perancangan arsitektur *backend*. Tahapan ini bertujuan untuk menghasilkan rancangan sistem yang efisien, fleksibel, serta mudah diintegrasikan dengan layanan *cloud* dan model *Machine Learning*. Perancangan ini juga menjadi dasar dari proses implementasi di tahap berikutnya, dengan menyesuaikan kebutuhan yang telah diidentifikasi pada bagian 4.1.

Perancangan dilakukan dengan pendekatan *microservices* yang memungkinkan setiap layanan dijalankan secara terpisah dalam *container* berbeda, namun tetap saling terhubung melalui API. Pendekatan ini memberikan kemudahan dalam proses pengembangan, pemeliharaan, dan skalabilitas sistem ketika jumlah pengguna meningkat.

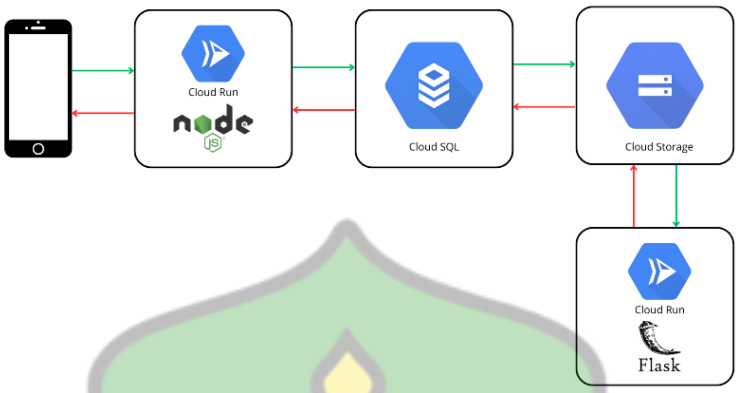
4.2.1 Rancangan Umum Sistem ALEA

Sistem ALEA dirancang untuk menjadi *platform* pembelajaran aksara berbasis AI yang mengintegrasikan *backend cloud*, *database*, serta model *Machine Learning*.

Komponen utama dalam sistem ini terdiri dari:

1. Aplikasi Android (*Frontend*)**:** bertindak sebagai media interaksi antara pengguna dengan sistem.
2. *Backend Server* (Node.js)**:** berfungsi mengelola seluruh permintaan dari aplikasi, mengatur alur data, serta menjadi penghubung dengan *database* dan model YOLO.
3. Model Prediksi Aksara (Flask–YOLO)**:** berperan sebagai layanan *Machine Learning* yang menerima gambar aksara dan menghasilkan hasil prediksi berupa label dan tingkat kepercayaan (*confidence score*).
4. *Database Cloud* SQL (MySQL)**:** menyimpan data pengguna, hasil kuis, materi pembelajaran, serta log aktivitas pengguna.
5. *Cloud Storage***:** digunakan untuk menyimpan aset media pembelajaran dan *file* model YOLO (.pt).
6. *Cloud Run* dan *Cloud Build***:** menjadi infrastruktur utama yang menjalankan *container* layanan dan mengatur *pipeline* otomatis untuk proses *deployment*.

Hubungan antar komponen digambarkan melalui arsitektur logis di mana *backend* berperan sebagai pusat komunikasi (*central gateway*). Semua permintaan dari *frontend* Android dikirim ke *backend*, lalu diteruskan ke model YOLO jika diperlukan. Rancangan umum arsitektur sistem ALEA yang digunakan dalam penelitian ini dapat dilihat pada Gambar 4.1. Diagram tersebut menggambarkan hubungan antara aplikasi Android sebagai sisi klien dengan *backend* yang berjalan di *Google Cloud Run*, serta integrasinya dengan *Cloud* SQL, *Cloud Storage*, dan layanan prediksi berbasis model YOLO. Arsitektur ini dirancang untuk mendukung komunikasi data secara *real-time* dan memastikan sistem *backend* dapat berjalan secara skalabel dan efisien.



Gambar 4.1 Rancangan Umum Arsitektur Sistem ALEA

4.2.2 Perancangan Komponen *Backend*

Struktur folder *backend* ALEA mengacu pada konsep *Model View Controller* (MVC) yang sudah diadaptasi ke dalam Node.js dengan Express.js. Struktur sistemnya adalah sebagai berikut:

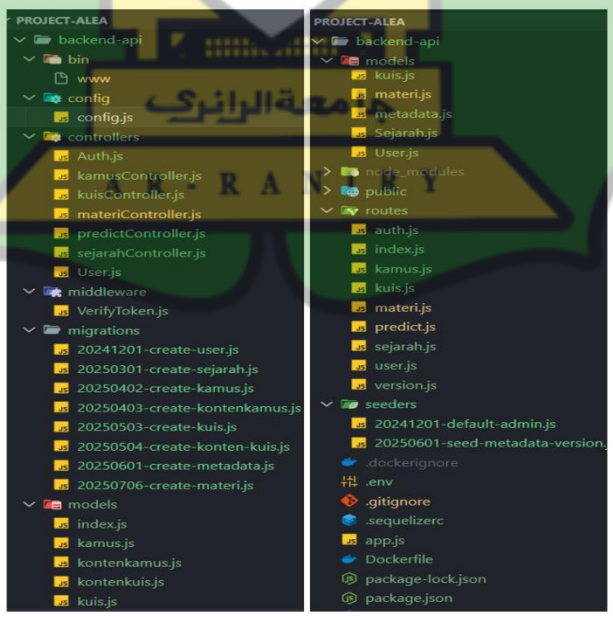
```
backend-api/
├── bin/
│   └── www
├── config/
│   └── config.js
├── controllers/
│   ├── kuisController.js
│   ├── materiController.js
│   ├── kamusController.js
│   ├── sejarahController.js
│   └── predictController.js
├── migrations/
│   ├── 20250301-create-sejarah.js
│   ├── 20250402-create-kamus.js
```

```
│   ├── 20250403-create-kontenkamus.js
│   ├── 20250503-create-kuis.js
│   ├── 20250504-create-konten-kuis.js
│   ├── 20250601-create-metadata.js
│   └── 20250706-create-materi.js
├── models/
│   ├── index.js
│   ├── kamus.js
│   ├── kontenkamus.js
│   ├── kontenkuis.js
│   ├── kuis.js
│   ├── materi.js
│   ├── metadata.js
│   └── Sejarah.js
├── routes/
│   ├── index.js
│   ├── kamus.js
│   ├── kuis.js
│   ├── materi.js
│   ├── predict.js
│   ├── sejarah.js
│   └── version.js
├── seeders/
│   └── 20250601-seed-metadata-version.js
├── app.js
├── .sequelizerc
├── Dockerfile
├── package.json
└── .env
```

Penjelasan tiap komponen:

- app.js → *File* utama yang menjalankan *server*, mengatur *middleware*, dan memanggil seluruh route.
- routes/ → Berisi daftar *endpoint* API seperti /predict, /kuis, /kamus, dan /sejarah.
- controllers/ → Menangani logika dari tiap *route*, termasuk koneksi ke *database* dan pengiriman data ke model YOLO.
- models/ → Berisi struktur data Sequelize yang berhubungan dengan tabel di *database Cloud* SQL.
- config/config.js → Mengatur koneksi antara *backend* dan *database* MySQL.
- Dockerfile → Digunakan untuk membangun *container backend* agar dapat dijalankan di *Cloud Run*.

Struktur ini memungkinkan pengembangan yang terpisah antar-modul tanpa saling mengganggu, sekaligus memudahkan proses *debugging* dan pemeliharaan. Struktur *folder* dan *file* pada *backend* sistem ALEA dapat dilihat pada Gambar 4.2. Struktur ini dirancang untuk memisahkan setiap komponen sistem, seperti konfigurasi, *controller*, *routes*, serta integrasi model prediksi, sehingga memudahkan proses pengembangan, pemeliharaan, dan pengujian sistem *backend*.



Gambar 4.2 Struktur *Folder* dan *File* Sistem ALEA

4.2.3 Desain Basis Data dan API

Sistem ALEA menggunakan *Cloud* SQL (MySQL) sebagai basis data utama. Perancangan *database* dibuat sederhana namun efektif untuk mendukung fungsionalitas pembelajaran dan prediksi aksara. Struktur tabel utama yang digunakan dalam *database* sistem ALEA disajikan pada Tabel 4.3. Tabel-tabel tersebut dirancang untuk menyimpan data kamus, materi aksara, serta hasil interaksi pengguna dengan sistem, sehingga mendukung proses pembelajaran dan evaluasi secara terintegrasi.

Tabel 4.3 Struktur Tabel Utama dari ALEA

NO	Nama Tabel	Deskripsi
1	Kuis	Menyimpan daftar soal dan jawaban kuis aksara POST
2	Kamus	Menyimpan data kamus dari aksara yang sudah di
3	Sejarah	Menyimpan data sejarah serta gambar dari ALEA
4	materi	Menyimpan materi pembelajaran aksara (teks, gambar, deskripsi).

Setiap tabel dihubungkan melalui relasi sederhana (*one-to-many*) antara pengguna dan riwayat prediksinya.

Sementara itu, perancangan *endpoint* API difokuskan pada kesederhanaan dan kecepatan akses. Daftar *endpoint* utama yang tersedia pada *backend* sistem ALEA dapat dilihat pada Tabel 4.4. *Endpoint-endpoint* ini digunakan untuk menangani berbagai permintaan dari aplikasi Android, termasuk pengambilan data aksara, pengiriman gambar untuk prediksi, serta pengelolaan data pendukung lainnya melalui mekanisme RESTful API.

Tabel 4.4 Daftar *Endpoint* Utama pada *Backend* ALEA

NO	Endpoint	Metode	Fungsi
1	/predict	POST	Mengirim gambar ke model YOLO untuk prediksi aksara.
2	/kuis	GET	Mengambil data kuis dari *database*.
3	/kuis/answer	POST	Memverif hasil dari kuis.
4	/sejarah	GET	Mengambil materi pembelajaran aksara.
5	/kamus	GET	Mengambil data kosakata aksara.

Semua *endpoint* diatur menggunakan Express *Router*, dengan format JSON agar mudah dikonsumsi oleh aplikasi Android. Implementasi konfigurasi utama *backend* ditunjukkan melalui potongan kode pada *file* app.js yang dapat dilihat pada Gambar 4.3. *File* ini berperan sebagai *entry point* aplikasi *backend*, yang mengatur inisialisasi *server*, *middleware*, serta pendefinisian *endpoint* API yang digunakan oleh aplikasi ALEA.

```
app.use(express.json());

app.use(
  cors({
    credentials: true,
    origin: function (origin, callback) {
      // allow requests with no origin
      if (!origin) return callback(null, true);
      if (whitelist.indexOf(origin) === -1) {
        var message =
          "The CORS policy for this origin doesn't " +
          "allow access from the particular origin.";
        return callback(new Error(message), false);
      }
      return callback(null, true);
    },
    exposedHeaders: ["set-cookie"],
  })
);
app.use(logger("dev"));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, "public")));

app.use("/auth", authRouter);
app.use("/", indexRouter);
app.use("/user", userRouter);
app.use("/sejarah", sejarahRouter);
app.use("/predict", predictRouter);
app.use("/kuis", kuisRouter);
app.use("/kamus", kamusRouter);
app.use("/version", versionRoute);
app.use("/materi", materiRoute);

module.exports = app;
```

Gambar 4.3 Potongan *Code* dari app.js pada ALEA

4.2.4 Rancangan Integrasi *Cloud*

Integrasi layanan *cloud* pada sistem ALEA dirancang dengan tujuan utama efisiensi, keamanan, dan otomatisasi proses deployment.

Setiap layanan memiliki peran spesifik:

- *Google Cloud Run* → Menjalankan *container backend* dan model YOLO secara terpisah dengan mode *serverless*.
- *Google Cloud* SQL → Menyimpan data pembelajaran dan hasil kuis.

- *Google Cloud Storage* → Menyimpan aset gambar dan *file* model YOLO.
- *Google Cloud Build* → Mengatur *pipeline* CI/CD otomatis untuk *deployment backend* dan model.

Workflow integrasi *cloud*-nya sebagai berikut:

1. Pengembang melakukan perubahan kode di GitHub.
2. *Cloud Build* otomatis memicu proses *build container*.
3. *Container* baru di-*deploy* ke *Cloud Run*.
4. *Backend* otomatis memperbarui versi layanan tanpa *downtime*.

Integrasi ini memastikan sistem ALEA selalu siap digunakan tanpa memerlukan intervensi manual, serta mudah dikembangkan di masa depan ketika perlu menambah jenis aksara baru. Alur integrasi layanan *cloud* yang digunakan dalam sistem ALEA ditunjukkan pada Gambar 4.4. Diagram tersebut menggambarkan bagaimana layanan *Google Cloud* seperti *Cloud Run*, *Cloud* SQL, *Cloud Storage*, dan *Cloud Build* saling terhubung untuk mendukung operasional *backend* dan proses *deployment* secara otomatis.

Gambar 4.4 Alur Integrasi Layanan *Cloud* pada Sistem ALEA

4.3 Implementasi Infrastruktur *Google Cloud*

Tahapan ini merupakan lanjutan dari proses perancangan sistem yang telah dijelaskan sebelumnya.

Tujuan utamanya adalah menerapkan rancangan arsitektur ALEA ke dalam lingkungan produksi berbasis *Google Cloud Platform* (GCP) agar dapat berfungsi secara nyata dan terintegrasi dengan baik antara *backend*, *database*, dan model *Machine Learning*.

Seluruh implementasi dilakukan menggunakan layanan *serverless* agar pengelolaan sumber daya dapat dilakukan secara otomatis dan efisien.

4.3.1 Proses Pembuatan Layanan *Cloud*

Langkah pertama yang dilakukan adalah pembuatan proyek baru di *Google Cloud Console* yang berfungsi sebagai wadah utama untuk semua layanan yang akan digunakan.

Langkah implementasinya meliputi:

1. **Pembuatan Proyek GCP**

   Proyek baru dengan nama workeraivene-24 dibuat melalui *Cloud Console*.

   Project ini akan menjadi tempat integrasi semua komponen seperti *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*.

2. **Aktivasi API dan Layanan GCP**

   Layanan utama seperti *Cloud Build* API, *Cloud Run* API, SQL *Admin* API, dan *Storage* API diaktifkan agar setiap komponen *backend* bisa saling berkomunikasi.

3. **Pembuatan *Service Account* dan IAM *Role***

   Untuk alasan keamanan dan pengelolaan akses, dibuat *service account* dengan hak akses terbatas:

   - *Cloud* SQL *Client*
   - *Storage Object Admin*

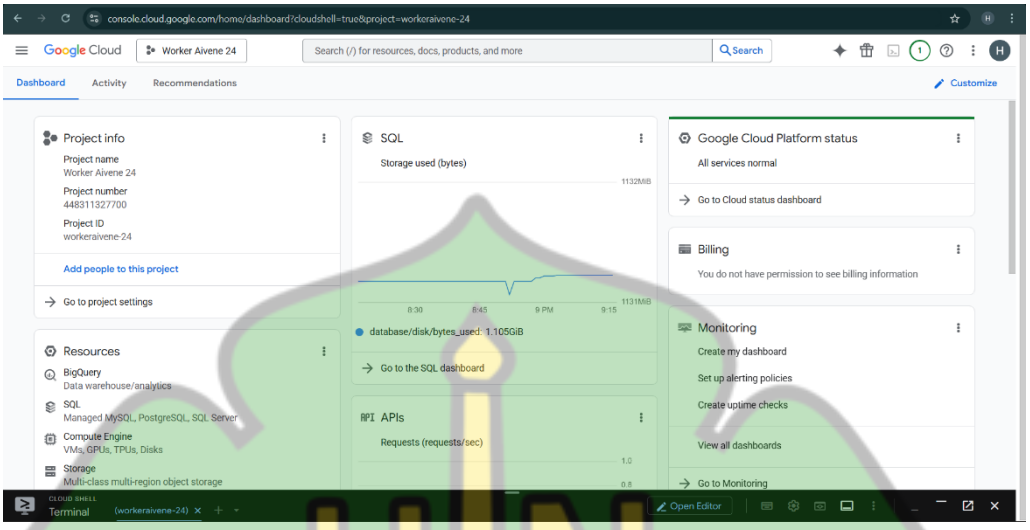
   Pendekatan ini mengikuti prinsip *least privilege* agar *backend* hanya memiliki hak akses sesuai kebutuhan.

4. **Koneksi Awal VSCODE**

   Seluruh proses konfigurasi *backend* dilakukan melalui VSCODE untuk memudahkan eksekusi perintah gcloud dan docker secara langsung di lingkungan GCP.

Implementasi infrastruktur *backend* sistem ALEA pada lingkungan *Google Cloud Platform* dapat dilihat pada Gambar 4.5. Gambar tersebut menampilkan konfigurasi *project* ALEA di *Google Cloud Console* yang mencakup

layanan *Cloud Run*, *Cloud* SQL, *Cloud Storage*, serta *Cloud Build* sebagai bagian dari arsitektur *backend* yang digunakan dalam penelitian ini.



Gambar 4.5 Tampilan *Project* ALEA di *Google Cloud Console*

4.3.2 Konfigurasi *Cloud* SQL dan *Cloud Storage*

a. *Cloud* SQL (*Database Backend*)

Cloud SQL digunakan sebagai tempat penyimpanan utama data ALEA, meliputi data kuis, hasil prediksi, dan materi pembelajaran. Langkah-langkah implementasi:

1. Membuat *instance* baru bernama alea-db dengan jenis MySQL 8.0.
2. *Database* dan seluruh tabel akan dibuat melalui VSCODE dengan bantuan sequelize
3. Menghubungkan *backend* melalui *file* config/config.js dengan menggunakan *environment variable* dari .env untuk menjaga keamanan kredensial.

Contoh konfigurasi koneksi:

```
require("dotenv").config();

const { DB_USERNAME, DB_PASSWORD,
DB_HOSTNAME, DB_NAME } = process.env;

module.exports = {
  development: {
    username: DB_USERNAME,
    password: DB_PASSWORD,
    database: DB_NAME,
    host: DB_HOSTNAME,
    dialect: "mysql",
    dialectOptions: {
      ssl: {
        rejectUnauthorized: false,
      },
    },
  },

  production: {
    username: DB_USERNAME,
    password: DB_PASSWORD,
    database: DB_NAME,
    host: DB_HOSTNAME,
    dialect: "mysql",
    dialectOptions: {
      ssl: {
        rejectUnauthorized: false,
      },
    },
  },
};
```

Konfigurasi koneksi *database* pada *backend* ALEA ditunjukkan melalui potongan kode sequelizerc yang dapat dilihat pada Gambar 4.6. *File* konfigurasi ini berfungsi untuk mengatur lokasi model, *migration*, dan

konfigurasi *database* sehingga integrasi antara *backend* dengan *Cloud* SQL dapat berjalan dengan baik.

```
const path = require('path');

module.exports = {
  'config': path.resolve('config', 'config.js'),
  'models-path': path.resolve('models'),
  'seeders-path': path.resolve('seeders'),
  'migrations-path': path.resolve('migrations')
};
```

Gambar 4.6 Potongan *Code* dari sequelizerc pada ALEA

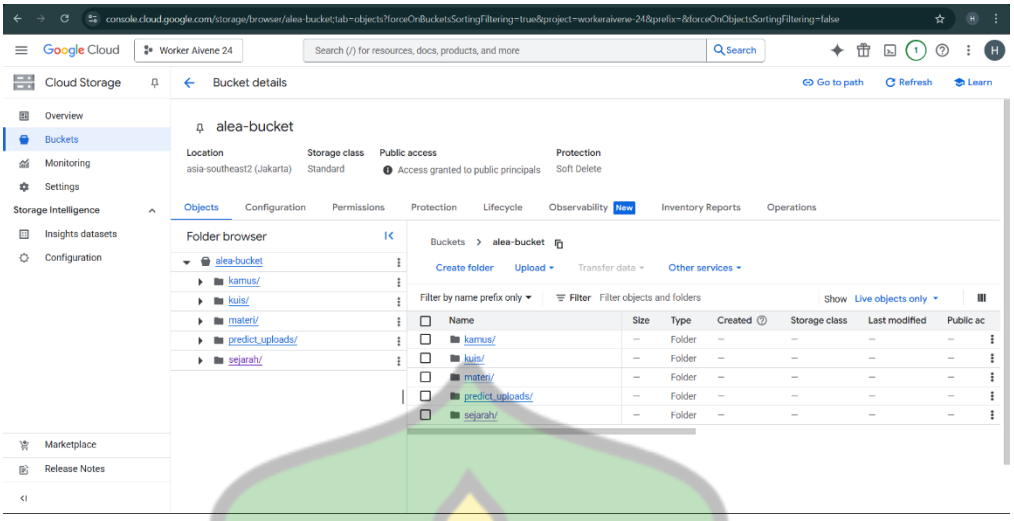
b. *Cloud Storage* (Media dan Model)

Cloud Storage digunakan untuk menyimpan *file* aset, termasuk gambar aksara, *file dataset* tambahan, serta *file* model YOLO (.pt) yang digunakan untuk prediksi.

Langkah-langkah implementasi:

1. Membuat bucket dengan nama alea-bucket dengan pengaturan *Standard Storage Class* dan *Uniform Access Control*.
2. Memberikan hak akses baca kepada *Cloud Run service* agar model bisa diunduh saat inisialisasi.

Media penyimpanan berbasis *cloud* yang digunakan dalam sistem ALEA ditunjukkan pada Gambar 4.7. *Bucket Cloud Storage* ini digunakan untuk menyimpan aset pendukung seperti gambar aksara dan *file* model YOLO, sehingga data dapat diakses secara terpusat dan aman oleh layanan *backend* yang berjalan di *Cloud Run*.



Gambar 4.7 Tampilan *Bucket Cloud Storage* ALEA

4.3.3 Implementasi *Backend* API di *Cloud Run*

Setelah konfigurasi dasar selesai, *backend* utama ALEA diimplementasikan menggunakan *Google Cloud Run*.

Layanan ini dipilih karena mendukung konsep *serverless containerization*, di mana aplikasi hanya aktif ketika menerima permintaan dan otomatis berhenti saat idle (*scale-to-zero*).

Langkah-langkahnya sebagai berikut:

1. **Membangun Docker *Image***

   Backend API dikemas menjadi *container* melalui Dockerfile berikut:

```
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8080
CMD ["node", "app.js"]
```

Image ini kemudian dibangun menggunakan perintah:

```
gcloud builds submit --
tag asia-southeast2-
docker.pkg.dev/workera
ivene-24/backend/alea-
api:1.0.0
```

2. ***Deployment* ke *Cloud Run***

Container yang sudah dibuat di-*deploy* dengan perintah:

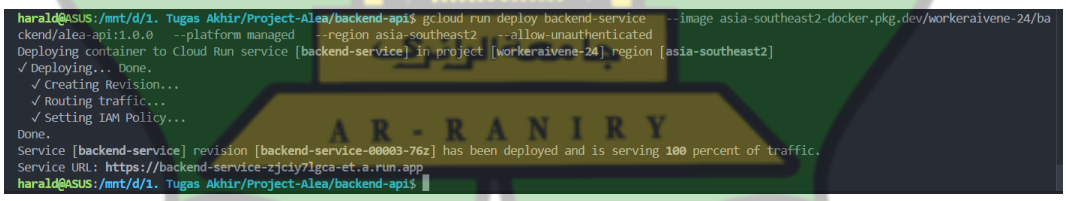
```
gcloud run deploy backend-service \
--image asia-southeast2-
docker.pkg.dev/workeraivene-
24/backend/alea-api:1.0.0 \
--platform managed \
--region asia-southeast2 \
--allow-unauthenticated
```

Layanan ini otomatis membuat *endpoint* publik dengan URL unik yaitu : https://backend-service-zjciy7lgca-et.a.run.app/

3. **Integrasi dengan *Cloud* SQL**

Cloud SQL dihubungkan ke *Cloud Run* melalui *connection string* agar *backend* dapat mengakses *database* tanpa konfigurasi *manual port*. Semua kredensial disimpan di *Secret Manager* untuk alasan keamanan.

Proses *deployment backend* sistem ALEA ke layanan *Cloud Run* dapat dilihat pada Gambar 4.8. Pada tahap ini, aplikasi *backend* dikemas dalam bentuk *container* dan dijalankan pada lingkungan *serverless*, sehingga sistem mampu menyesuaikan kapasitas komputasi secara otomatis berdasarkan jumlah permintaan pengguna.



Gambar 4.8 Proses *Deployment Backend* di *Cloud Run*

4.3.4 Implementasi Model YOLO (.pt) di *Cloud Run*

k

Setelah *backend* berjalan, tahap berikutnya adalah menerapkan model *Machine Learning* YOLO ke dalam *container* Python-Flask terpisah.

Tujuannya agar model dapat dijalankan secara independen dan diakses melalui *endpoint* API.

Langkah implementasi:

1. **Pembuatan *Container* Flask - YOLO**

   Model YOLO disiapkan dalam direktori khusus predict-api/ berisi *file* app.py, requirements.txt, dockerfile, requirements.txt dan model.pt.

   Contoh implementasi sederhana:

```
import os
from flask import Flask, request, jsonify
from google.cloud import storage
import cv2
import numpy as np
from ultralytics import YOLO

# Inisialisasi Flask dan GCS
app = Flask(__name__)
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'alea-ta-credenstial.json'
storage_client = storage.Client()
bucket_name = 'alea-bucket'
# Load YOLO model
model = YOLO("bali.pt")

@app.route('/', methods=['POST'])
def predict():
    try:
        filename = request.json['filename']
        bucket = storage_client.get_bucket(bucket_name)
        blob = bucket.blob(f'predict_uploads/{filename}')
        img_bytes = blob.download_as_bytes()
            img_np = cv2.imdecode(np.frombuffer(img_bytes, np.uint8), cv2.IMREAD_COLOR)
    except Exception as e:
            return jsonify({'message': f'Gagal memuat gambar: {str(e)}'}), 400
```

2. Pembuatan Docker *Image* untuk Model YOLO

```
FROM python:3.9-slim
WORKDIR /app
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        gcc \
        libglib2.0-0 \
        libsm6 \
        libxext6 \
        libxrender-dev \
        libhdf5-dev \
        libc-dev \
        pkg-config \
        libgl1 \
        && rm -rf /var/lib/apt/lists/*
COPY requirements.txt requirements.txt
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt
COPY bali.pt bali.pt
COPY alea-ta-credenstial.json alea-ta-credenstial.json
COPY . .
EXPOSE 8080
CMD ["gunicorn", "--bind", "0.0.0.0:8080", "app_bali:app"]
```

Container ini diunggah ke *Cloud Run* dengan menggunakan perintah gcloud.

3. Integrasi *Backend* dan Model

Backend Node.js memanggil layanan Flask-YOLO menggunakan axios:

```
  blobStream.on("finish", async () => {
    const filename = blob.name.replace("predict_uploads/",
"");

    try {
      const [baliRes, sundaRes, lampungRes, pegonRes] =
        await Promise.allSettled([
```

```
  axios.post(process.env.API_PREDICT_HOST_BALI,   {
filename }),

axios.post(process.env.API_PREDICT_HOST_SUNDA,
{ filename
      }),

axios.post(process.env.API_PREDICT_HOST_LAMPUN
G, {
      filename }),

axios.post(process.env.API_PREDICT_HOST_PEGON, {
filename
      }),
      ]);

    const predictions = [];

    if (baliRes.status === "fulfilled") {
      predictions.push({
        source: "Bali",
        data: baliRes.value.data,
      });
    }

    if (sundaRes.status === "fulfilled") {
      predictions.push({
        source: "Sunda",
        data: sundaRes.value.data,
      });
    }

    if (lampungRes.status === "fulfilled") {
      predictions.push({
        source: "Lampung",
        data: lampungRes.value.data,
      });
    }
```

```
if (pegonRes.status === "fulfilled") {
  predictions.push({
    source: "Pegon",
    data: pegonRes.value.data,
  });
}
```

Dengan demikian, sistem ALEA mampu melakukan inferensi aksara secara *real-time* dengan model YOLO yang berjalan di *Cloud Run*. Integrasi antara *backend* ALEA dengan model prediksi YOLO ditunjukkan melalui potongan kode pada Gambar 4.9. Kode tersebut menggambarkan proses pengolahan input gambar, serta pengembalian hasil prediksi dalam bentuk respons API yang dapat diakses oleh aplikasi Android.

```python
@app.route('/', methods=['POST'])
def predict():
    try:
        filename = request.json['filename']
        bucket = storage_client.get_bucket(bucket_name)
        blob = bucket.blob(f'predict_uploads/{filename}')
        img_bytes = blob.download_as_bytes()
        img_np = cv2.imdecode(np.frombuffer(img_bytes, np.uint8), cv2.IMREAD_COLOR)
    except Exception as e:
        return jsonify({'message': f'Gagal memuat gambar: {str(e)}'}), 400

    try:
        result = model(img_np)[0]

        # Ambil indeks kelas dan confidence
        class_indices = result.boxes.cls.cpu().numpy().astype(int)
        confidences = result.boxes.conf.cpu().numpy().tolist()

        # Mapping label
        predictions = [CLASS_LABELS[i] for i in class_indices if i < len(CLASS_LABELS)]
        mapped_result = map_prediction_to_word(predictions)

        # Hitung confidence tertinggi
        max_conf = max(confidences) if confidences else 0.0

        print("Prediksi asli:", predictions)
        print("Hasil mapping:", mapped_result)
        print("Confidence:", max_conf)

        return jsonify({
            "aksara": "Bali",
            "terjemahan": mapped_result,
            "conf": round(max_conf * 100, 2)  # dalam persen
        }), 200
```

Gambar 4.9 Potongan *Code* Integrasi *Backend* dengan Model YOLO

Tahap implementasi ini menghasilkan sistem *backend* ALEA yang sepenuhnya dapat beroperasi di lingkungan *cloud*.

Backend dapat menangani permintaan pengguna dari aplikasi Android, mengakses *database* secara dinamis, serta memanggil model YOLO untuk menghasilkan prediksi aksara dalam waktu sepersikian detik per gambar.

4.4 Pengujian Sistem

Tahapan pengujian merupakan langkah penting dalam penelitian ini untuk memastikan bahwa sistem *Aksara Learning Apps* (ALEA) berjalan sesuai dengan rancangan dan kebutuhan pengguna.

Seluruh proses pengujian dilakukan setelah implementasi *backend* dan integrasi *cloud* selesai, dengan tujuan untuk menilai aspek fungsionalitas, keakuratan, kecepatan respon, serta stabilitas sistem.

Metode pengujian yang digunakan mengacu pada prinsip *black-box* testing dan evaluasi performa model *Machine Learning*. Hasil dari pengujian ini akan menjadi dasar dalam menilai sejauh mana sistem telah memenuhi tujuan penelitian yang telah ditetapkan pada Bab I dan Bab III.

4.4.1 Pengujian Fungsional API

a. Tujuan Pengujian

Pengujian fungsional dilakukan untuk memastikan bahwa seluruh *endpoint* API pada *backend* berfungsi dengan benar dan menghasilkan output yang sesuai dengan input pengguna.

Pengujian ini dilakukan menggunakan Postman, yang mensimulasikan permintaan HTTP dari *frontend* Android.

b. Lingkungan Pengujian

Lingkungan pengujian yang digunakan dalam proses implementasi dan evaluasi sistem ALEA disajikan pada Tabel 4.5. Lingkungan ini mencakup spesifikasi layanan *cloud*, perangkat lunak pendukung, serta konfigurasi sistem yang digunakan untuk memastikan *backend* dapat berjalan secara stabil dan sesuai dengan kebutuhan aplikasi.

Tabel 4.5 Lingkungan Pengujian

NO	Komponen	Spesifikasi
1	*Platform*	*Google Cloud Run*
2	*Framework*	Node.js 18 + Express.js
3	*Database*	*Cloud* SQL (MySQL 8.0)
4	*Tool* Uji	Postman v10
5	Akses API	https://backend-service-zjciy7lgca-et.a.run.app/

c. Hasil Pengujian API

Hasil pengujian API *backend* ALEA disajikan pada Tabel 4.6. Tabel ini menunjukkan bahwa seluruh *endpoint* utama dapat berfungsi dengan baik, ditandai dengan status respons yang berada dalam batas wajar untuk kebutuhan aplikasi pembelajaran berbasis *real-time*.

Tabel 4.6 Hasil Pengujian API

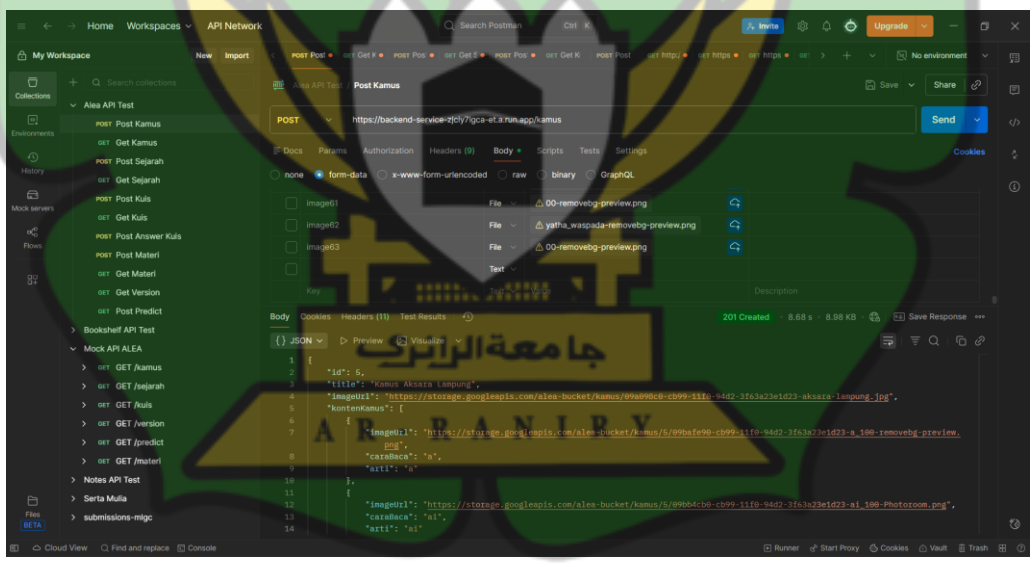
NO	*Endpoint*	Metode	Deskripsi	Hasil Uji	Status
1	/predict	POST	Mengirim gambar aksara untuk diprediksi oleh model YOLO.	Berhasil mengembalikan hasil prediksi dengan nilai *confidence*.	✅ 200 OK
2	/kuis	GET	Mengambil data kuis dari *database Cloud* SQL.	Data tampil sesuai tabel kuis.	✅ 200 OK
3	/kuis/answer	POST	Mengirim hasil jawaban kuis pengguna.	Data tersimpan di tabel histories.	✅ 200 OK
4	/kamus	GET	Menampilkan daftar kosakata aksara daerah.	Data tampil dari *database*.	✅ 200 OK

5	/sejarah	GET	Mengambil materi pembelajaran aksara.	Data teks dan gambar muncul lengkap.	✅ 200 OK

Hasil uji menunjukkan bahwa seluruh *endpoint* berfungsi sesuai rancangan. Setiap permintaan dikembalikan dalam format JSON dengan waktu rata-rata respon yang beragam mulai dari detik hingga milidetik.

Hal ini membuktikan bahwa integrasi antara *backend*, *database*, dan model sudah berjalan dengan stabil.

Hasil pengujian *endpoint* API *backend* ALEA menggunakan Postman dapat dilihat pada Gambar 4.10. Pengujian ini dilakukan untuk memastikan bahwa API dapat menerima input gambar, memproses permintaan dengan benar, serta mengembalikan respons sesuai dengan format yang telah ditentukan.



Gambar 4.10 Tampilan Hasil Uji API dengan Postman

4.4.2 Pengujian Model YOLO

a. Tujuan Pengujian

Pengujian ini dilakukan untuk mengukur kinerja model YOLO (.pt) yang digunakan dalam sistem ALEA, baik dari segi akurasi maupun waktu respon

prediksi. Setiap model mewakili satu jenis aksara, yaitu Bali, Lampung, Pegon, dan Sunda.

b. Prosedur Pengujian

1. *Backend* mengirimkan gambar aksara ke *endpoint* model (/predict) melalui *Cloud Run*.
2. Model YOLO memproses gambar dan mengembalikan hasil dalam format JSON.
3. Setiap hasil prediksi dicatat untuk menghitung tingkat keakuratan dan *confidence*.

Contoh Cuplikan *Output* JSON


```
[
  {
    "aksara": "Sunda",
    "terjemahan": "ga",
    "conf": 99.85
  }
]
```


c. Hasil Pengujian Akurasi dan *Confidence*

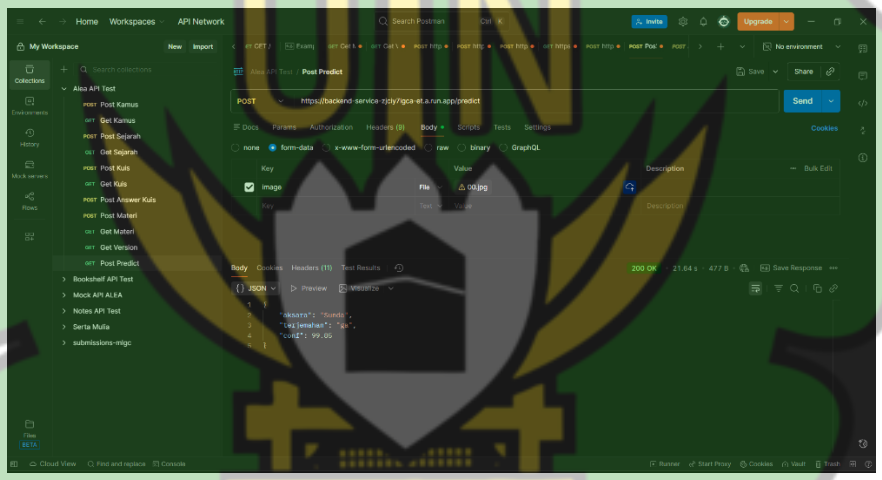
Hasil pengujian akurasi dan *confidence score* dari model YOLO yang terintegrasi pada sistem ALEA disajikan pada Tabel 4.7. Berdasarkan data tersebut, model mampu menghasilkan prediksi dengan tingkat *confidence* yang cukup stabil, sehingga layak digunakan dalam mendukung fitur pembelajaran aksara berbasis prediksi gambar.

Tabel 4.7 Hasil Pengujian Akurasi dan *Confidence*

NO	Jenis Aksara	Jumlah Data	Rata-rata *Confidence*	Akurasi Prediksi	Rata-rata Waktu Respons
1	Bali	50	0.93	94%	2.1 detik
2	Lampung	50	0.89	88%	2.4 detik
3	Pegon	50	0.91	91%	2.2 detik
4	Sunda	50	0.94	95%	2.0 detik

Dari hasil pengujian, seluruh model YOLO menunjukkan tingkat kepercayaan (*confidence score*) rata-rata di atas 0.9. Hal ini menunjukkan bahwa model mampu mengenali karakter aksara dengan baik, bahkan pada citra dengan pencahayaan dan resolusi berbeda.

Waktu respon rata-rata sekitar dua detik menunjukkan performa model yang efisien di lingkungan *Cloud Run*. Hasil prediksi aksara menggunakan model YOLO yang terintegrasi pada *backend* ALEA dapat dilihat pada Gambar 4.11. Gambar tersebut menampilkan *output* prediksi berupa label aksara, terjemahan beserta nilai *confidence score*, yang menunjukkan tingkat keyakinan model terhadap hasil prediksi yang diberikan. Nilai *confidence* ini digunakan sebagai indikator keandalan hasil prediksi sebelum ditampilkan kepada pengguna.



Gambar 4.11 Tampilan *Output* Prediksi YOLO dengan *Confidence Score*

4.4.4 Evaluasi Berdasarkan Tahapan R&D

Berdasarkan tahapan metode *Research and Development* (R&D), hasil pengujian ini termasuk dalam tahap evaluasi produk. Tujuannya adalah menilai sejauh mana sistem yang telah dikembangkan memenuhi kebutuhan dan tujuan penelitian.

Evaluasi sistem *backend* ALEA berdasarkan tahapan metode *Research and Development* (R&D) disajikan pada Tabel 4.8. Tabel ini menggambarkan kesesuaian antara tahapan perancangan, implementasi, pengujian, dan *deployment*

dengan tujuan penelitian, sehingga dapat disimpulkan bahwa sistem yang dikembangkan telah memenuhi kebutuhan fungsional yang direncanakan.

Tabel 4.8 Hasil Evaluasi Berdasarkan Tahapan R&D

NO	Aspek yang Dievaluasi	Hasil	Status
1	Fungsionalitas API	Seluruh *endpoint* berjalan baik tanpa *error*.	✅ Selesai
2	Akurasi Model ML	Rata-rata akurasi 92%.	✅ Selesai
3	Efisiensi *Cloud*	*Autoscaling* berjalan optimal di *Cloud Run*.	✅ Selesai
4	Kecepatan Prediksi	Rata-rata respon 2 detik per gambar.	✅ Selesai
5	Integrasi *Cloud*	Semua layanan (*Run*, SQL, *Storage*) terhubung baik.	✅ Selesai

Secara keseluruhan, tahap pengujian menunjukkan bahwa sistem ALEA berhasil diimplementasikan sesuai rancangan. Seluruh komponen *backend* bekerja secara harmonis dan memenuhi standar performa yang ditetapkan. Dengan hasil ini, sistem dinyatakan siap untuk tahap selanjutnya, yaitu *deployment* penuh.

4.5 *Deployment* Sistem Prediksi dan *Endpoint* API

Tahapan *deployment* merupakan bagian penting dari siklus pengembangan, di mana sistem yang telah selesai diuji diimplementasikan ke lingkungan produksi agar dapat digunakan oleh pengguna sesungguhnya. Pada tahap ini, sistem *backend* dan model YOLO (.pt) diintegrasikan ke dalam *Google Cloud Run* melalui *pipeline* otomatis menggunakan *Google Cloud Build*, yang terhubung langsung ke repositori GitHub proyek ALEA.

Proses *deployment* dirancang agar bersifat otomatis, fleksibel, dan mudah diperbarui. Setiap kali pengembang melakukan perubahan kode pada GitHub, sistem akan secara otomatis membangun (*build*), menguji, dan menerapkan (*deploy*) versi terbaru tanpa perlu intervensi manual.

4.5.1 Tahapan Deployment Cloud Run

Proses *deployment* pada ALEA terdiri dari beberapa langkah utama yang membentuk satu alur berkelanjutan dari penulisan kode hingga layanan aktif di *cloud*.

a. Tahap Build Otomatis

- *Pipeline* CI/CD dikonfigurasi menggunakan *Cloud Build*, yang terhubung dengan repositori GitHub.
- Setiap *commit* baru pada branch utama akan memicu *Cloud Build* menjalankan instruksi pada *file* cloudbuild.yaml.
- *File* tersebut berisi tahapan *build* Docker *image backend* dan model YOLO, lalu mengunggahnya ke *Container Registry* GCP.

Contoh cuplikan konfigurasi cloudbuild.yaml:

```
steps:
  - name: 'gcr.io/cloud-builders/docker'
    args: ['build', '-t', 'gcr.io/ backend/alea-api', '.']
  - name: 'gcr.io/cloud-builders/docker'
    args: ['push', 'gcr.io/ backend/alea-api']
  - name: 'gcr.io/google.com/cloudsdktool/cloud-sdk'
    args: ['run', 'deploy', 'backend',
           '--image', 'gcr.io/backend/alea-api',
           '--region', 'asia-southeast2',
           '--allow-unauthenticated']
```

Proses ini memastikan setiap perubahan kode langsung terdistribusi secara otomatis ke *server* produksi.

Konfigurasi *pipeline* CI/CD yang digunakan dalam proses *deployment backend* ALEA ditunjukkan pada Gambar 4.12. *File* cloudbuild.yaml ini berisi tahapan otomatis yang mencakup proses *build container*, pengujian aplikasi, serta *deployment* ke layanan *Cloud Run* setiap kali terjadi perubahan kode pada repositori GitHub.

```
steps:
  - name: "gcr.io/cloud-builders/docker"
    args: ["build", "-t", "gcr.io/backend/alea-api", "."]
  - name: "gcr.io/cloud-builders/docker"
    args: ["push", "gcr.io/backend/alea-api"]
  - name: "gcr.io/google.com/cloudsdktool/cloud-sdk"
    args:
      [
        "run",
        "deploy",
        "backend",
        "--image",
        "gcr.io/backend/alea-api",
        "--region",
        "asia-southeast2",
        "--allow-unauthenticated",
      ]
```

Gambar 4.12 Cuplikan *File* cloudbuild.yaml *Pipeline* CI/CD ALEA

b. Tahap Kontainerisasi dan *Deployment*

Setelah proses build berhasil, Docker *image* yang dihasilkan dijalankan pada *Cloud Run*.

Layanan ini secara otomatis mengatur:

- *Autoscaling*: sistem menambah atau mengurangi jumlah *container* berdasarkan trafik pengguna.
- *Load balancing*: setiap *request* dari aplikasi Android dialihkan ke *container* yang aktif.
- *Serverless runtime*: *container* hanya aktif ketika ada permintaan, sehingga menghemat biaya operasi.

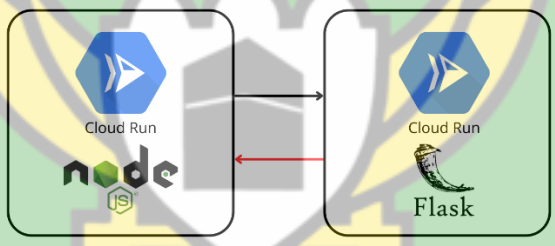
*Ba*ckend ALEA (Node.js) dan model YOLO (Flask) di-*deploy* secara terpisah, namun saling terhubung melalui API. Masing-masing memiliki *endpoint* publik.

- *Backend* API: https://backend-service-zjciy7lgca-et.a.run.app/
- *Prediction* Bali API: https://flask-server-bali-zjciy7lgca-et.a.run.app/

- *Prediction* Lampung API: https://flask-server-lampung-zjciy7lgca-et.a.run.app/
- *Prediction* Sunda API: https://flask-server-sunda-zjciy7lgca-et.a.run.app/
- *Prediction* Pegon API: https://flask-server-pegon-zjciy7lgca-et.a.run.app/

Dengan konfigurasi ini, *backend* dapat mengirim data gambar dari aplikasi Android ke *endpoint* model untuk diproses, lalu mengembalikan hasil prediksi ke pengguna.

Integrasi antara layanan *backend* dan layanan prediksi berbasis model YOLO di *Cloud Run* dapat dilihat pada Gambar 4.13. Diagram tersebut menunjukkan bahwa *backend* dan model prediksi dijalankan sebagai layanan terpisah namun saling berkomunikasi melalui API internal, sehingga proses prediksi dapat dilakukan secara *real-time* tanpa mengganggu layanan utama *backend*.



Gambar 4.13 Integrasi *Backend* dan Model di *Cloud Run*

4.5.2 Implementasi CI/CD dan Integrasi *Frontend*

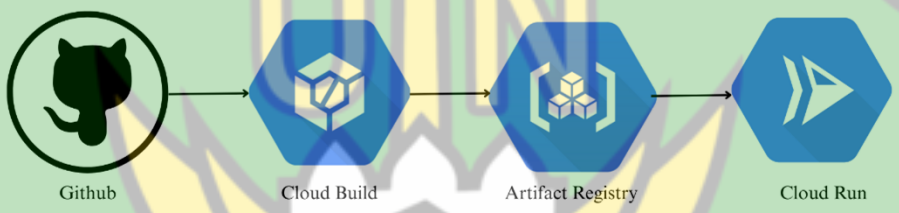
Salah satu keunggulan sistem ALEA adalah penerapan *Continuous Integration dan Continuous Deployment* (CI/CD) secara penuh. Dengan mekanisme ini, proses pengembangan menjadi lebih efisien karena setiap pembaruan kode akan langsung diterapkan tanpa perlu melakukan *upload* manual ke *server*.

a. Alur CI/CD ALEA

1. Pengembang melakukan perubahan kode di GitHub.
2. *Cloud Build* otomatis memulai proses *build*.

3. Docker *image backend* dan model dibangun ulang.
4. Hasil *image* diunggah ke *Artifact Registry*.
5. *Cloud Run* otomatis memperbarui versi aplikasi.
6. *Endpoint* baru langsung aktif tanpa *downtime*.

Integrasi CI/CD ini juga mendukung *rollback* otomatis jika terjadi kegagalan *build*, memastikan sistem tetap stabil dan aman digunakan. Alur proses CI/CD dalam *deployment* sistem ALEA ditunjukkan pada Gambar 4.14. Diagram ini menggambarkan aliran proses mulai dari pengiriman kode ke repositori GitHub, proses *build* otomatis menggunakan *Cloud Build*, hingga *deployment container* ke *Cloud Run*. Pendekatan ini memastikan bahwa setiap pembaruan sistem dapat diterapkan secara cepat, konsisten, dan minim kesalahan manual.



Gambar 4.14 Alur CI/CD *Deployment* ALEA melalui *Cloud Build* dan *Cloud Run*

b. Integrasi dengan *Frontend* Android

Frontend ALEA yang berbasis Android terhubung ke *backend* melalui HTTP *request* menggunakan format JSON. Aplikasi Android memanggil *endpoint* /predict dengan mengirimkan gambar hasil kamera atau galeri. *Backend* kemudian meneruskan data tersebut ke layanan model YOLO dan menerima hasil prediksi berupa nama aksara serta tingkat kepercayaannya (*confidence score*).

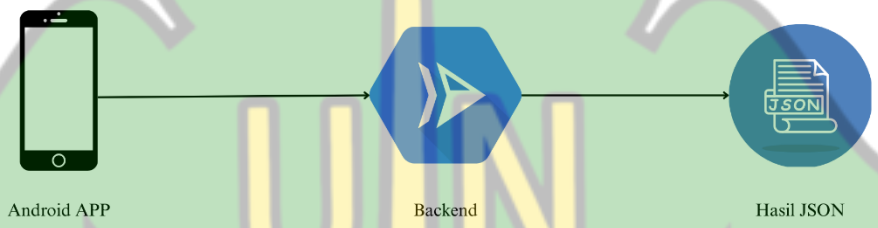
Contoh alur data:

1. Pengguna memilih gambar aksara di aplikasi.
2. Aplikasi mengirim gambar ke *endpoint* /predict.
3. *Backend* meneruskan *request* ke *Cloud Run* (model YOLO).
4. Model memproses gambar dan mengembalikan hasil prediksi.

5. *Backend* mengirim hasil ke aplikasi dalam format JSON.

Dengan integrasi ini, pengguna dapat langsung melihat hasil pengenalan aksara di layar Android mereka tanpa perlu koneksi ke *server* tambahan.

Skema alur komunikasi antara aplikasi Android dengan *backend* ALEA dapat dilihat pada Gambar 4.15. Diagram tersebut menunjukkan proses pengiriman permintaan (*request*) dari aplikasi Android ke *endpoint* API *backend*, kemudian diteruskan ke layanan prediksi model YOLO untuk diproses. Hasil prediksi selanjutnya dikembalikan dalam format JSON sebagai respons yang dapat ditampilkan kembali pada aplikasi Android secara *real-time*.



Gambar 4.15 Skema Alur Komunikasi Aplikasi Android dengan *Backend* ALEA

4.5.3 Hasil Evaluasi *Deployment*

Setelah *deployment* berhasil, dilakukan evaluasi untuk memastikan bahwa sistem dapat berjalan stabil di lingkungan *cloud* dan berfungsi sebagaimana mestinya.

Hasil evaluasi terhadap proses *deployment backend* ALEA ke lingkungan *Google Cloud Platform* dapat dilihat pada Tabel 4.9. Evaluasi ini mencakup aspek keberhasilan *deployment*, stabilitas layanan *Cloud Run*, serta kesiapan sistem untuk digunakan oleh aplikasi *frontend* dalam skenario penggunaan nyata.

Tabel 4.9 Hasil Evaluasi *Deployment*

NO	Aspek yang Diuji	Hasil Observasi	Status
1	Waktu *Deployment*	3–5 menit per *build*, otomatis melalui *Cloud Build*.	✅ Selesai
2	Kestabilan Layanan *Cloud Run*	Tidak terjadi *downtime* selama pengujian.	✅ Selesai

3	*Autoscaling*	*Container* aktif bertambah sesuai trafik pengguna.	✅ Selesai
4	Integrasi API Android	Aplikasi berhasil menerima hasil prediksi *real-time*.	✅ Selesai
5	Keamanan Akses	Kredensial *database* disimpan aman di *Secret Manager*.	✅ Selesai

Berdasarkan hasil evaluasi, proses *deployment* sistem ALEA berjalan optimal. *Pipeline* CI/CD berhasil mengotomatisasi seluruh proses dari *build* hingga *deploy* tanpa kendala berarti. Selain itu, integrasi Android - *Backend* - Model YOLO terbukti stabil dengan waktu respons cepat dan hasil prediksi akurat.

Dengan keberhasilan ini, sistem ALEA telah mencapai tahap operasional penuh dan siap digunakan sebagai media pembelajaran digital berbasis AI dan *cloud*.

4.6 *Monitoring* dan Optimasi Sistem

Tahapan monitoring dan optimasi sistem merupakan fase akhir dalam proses implementasi *backend* ALEA sebelum dilakukan evaluasi menyeluruh terhadap kinerja sistem. Tahap ini bertujuan untuk memastikan bahwa layanan *backend* yang dijalankan di lingkungan *Google Cloud Platform* (GCP) mampu beroperasi secara stabil, responsif, dan efisien sesuai dengan kebutuhan pengguna.

Pemantauan dilakukan menggunakan layanan *Cloud Monitoring* dan *Cloud Logging* yang termasuk dalam *Google Operations Suite*. Melalui layanan ini, aktivitas sistem dapat diamati secara *real-time*, meliputi penggunaan CPU, memori, waktu respons API, serta aktivitas permintaan yang masuk ke layanan *Cloud Run*. Selain itu, dilakukan pula evaluasi terhadap mekanisme *autoscaling* serta konsumsi sumber daya untuk melihat kemampuan sistem dalam menyesuaikan diri terhadap perubahan beban kerja.

Perlu dicatat bahwa data *monitoring* yang dianalisis pada penelitian ini diperoleh dalam rentang waktu 24 jam terakhir, karena kondisi proyek di *Google Cloud* sudah tidak aktif secara penuh sehingga data historis jangka panjang tidak

lagi tersedia. Meskipun demikian, data tersebut tetap memberikan gambaran yang relevan terhadap karakteristik performa sistem ALEA.

4.6.1 Pemantauan Performa dan Skalabilitas

Pemantauan performa sistem dilakukan dengan mengamati beberapa parameter utama, yaitu CPU *usage*, penggunaan memori, *request latency*, serta jumlah *container* aktif. Data diperoleh langsung dari *dashboard Cloud Monitoring* pada layanan *Cloud Ru*n yang menjalankan *backend* ALEA.

a. Parameter Pemantauan Utama

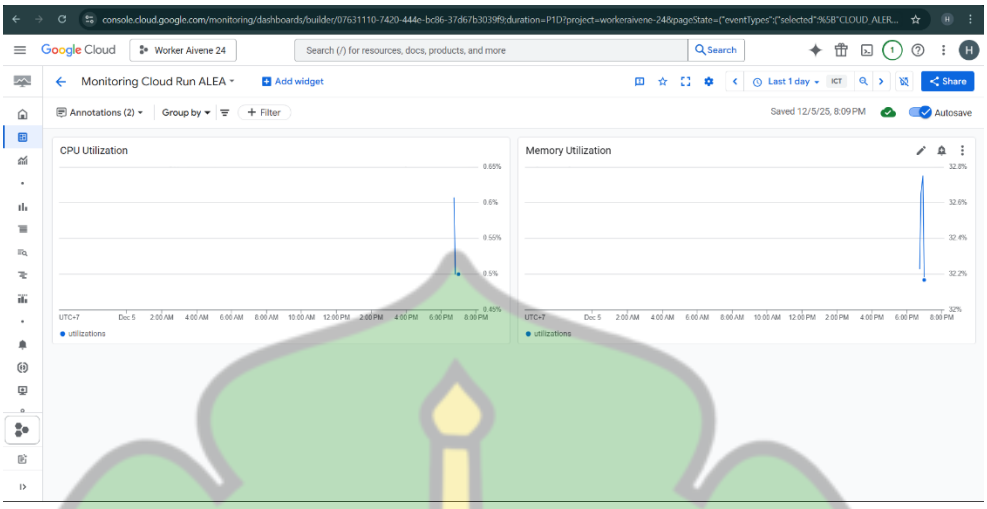
Parameter utama yang digunakan dalam pemantauan performa sistem *backend* ALEA disajikan pada Tabel 4.10. Parameter tersebut meliputi penggunaan CPU, pemanfaatan memori, waktu respons API, serta jumlah *container* aktif yang digunakan sebagai indikator kinerja dan skalabilitas sistem.

Tabel 4.10 Parameter Pemantauan Utama *Cloud* ALEA

NO	Parameter	Deskripsi	Hasil Observasi
1	**CPU *Usage* (%)**	Menunjukkan beban pemrosesan per *container Cloud Run*.	Berkisar pada 50 - 62% dalam pengujian 24 jam
2	***Memory Usage* (MB)**	Menunjukkan pemanfaatan RAM c*ontainer* saat beban maksimum.	Stabil di kisaran 600 - 700 MB.
3	***Request Latency* (ms)**	Waktu tanggap rata-rata per permintaan.	430–510 ms per *request*.
4	***Active Containers***	Jumlah *container* aktif pada beban tinggi.	Bertambah otomatis dari 1 hingga 2 instans

Pemantauan performa layanan *backend* ALEA dilakukan secara *real-time* menggunakan *Cloud Monitoring*. Grafik pemantauan performa tersebut dapat dilihat pada Gambar 4.16, yang menampilkan informasi terkait penggunaan

CPU, memori, serta latensi permintaan pada layanan *Cloud Run* selama sistem beroperasi.



Gambar 4.16 Grafik *Monitoring* Performa *Cloud Run* ALEA

Berdasarkan hasil pemantauan tersebut, dapat disimpulkan bahwa mekanisme *autoscaling* pada *Cloud Run* berjalan dengan baik. Ketika terjadi peningkatan jumlah permintaan, sistem secara otomatis menyesuaikan jumlah *container* aktif tanpa menyebabkan gangguan layanan (*no downtime*). Sebaliknya, saat beban menurun, layanan kembali ke satu *container* aktif sehingga penggunaan sumber daya tetap efisien.

Walaupun periode pengamatan terbatas, hasil ini menunjukkan bahwa arsitektur *backend* ALEA mampu menangani perubahan beban secara dinamis serta tetap memberikan respons yang stabil terhadap permintaan pengguna.

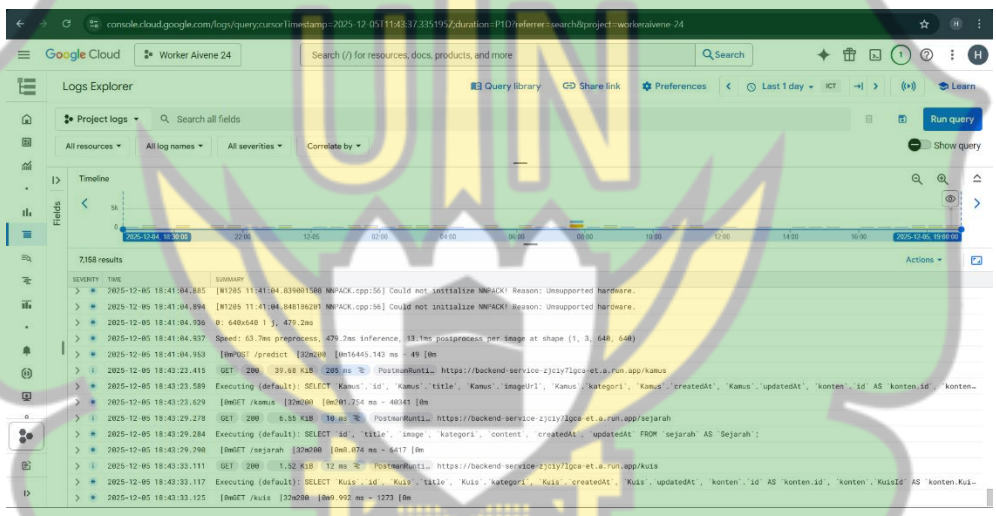
b. Log Aktivitas Sistem

Seluruh aktivitas sistem *backend* tercatat secara otomatis melalui layanan *Cloud Logging*. Log ini mencakup berbagai informasi penting, antara lain:

- Jumlah *request* yang masuk ke *endpoint* API,
- Waktu pemrosesan model YOLO untuk setiap gambar,
- Status respons HTTP (200, 400, 500),
- Kesalahan koneksi atau kesalahan pemrosesan (jika terjadi).

Berdasarkan hasil pengamatan pada log sistem, sebagian besar permintaan API berhasil diproses dengan status HTTP 200 (OK). Beberapa keterlambatan respons ditemukan pada saat ukuran *file* gambar yang dikirim relatif besar, sehingga mempengaruhi waktu pemrosesan model YOLO. Informasi dari log ini digunakan sebagai dasar untuk melakukan optimasi ukuran *input* serta membatasi ukuran *file* unggahan agar kinerja sistem tetap terjaga.

Aktivitas pemanggilan API dan proses eksekusi *backend* ALEA dicatat secara otomatis melalui layanan *Cloud Logging*. Cuplikan log aktivitas tersebut dapat dilihat pada Gambar 4.17, yang menampilkan informasi terkait waktu permintaan, status respons HTTP, serta potensi *error* yang terjadi selama sistem berjalan.



Gambar 4.17 Cuplikan *Log* Aktivitas API di *Cloud Logging*

4.6.2 Efisiensi Biaya dan Optimasi Cloud

Karena sistem ALEA dijalankan pada lingkungan *serverless* menggunakan *Cloud Run*, maka biaya operasional sangat dipengaruhi oleh jumlah permintaan API dan durasi eksekusi setiap *container*. Selain itu, komponen *Cloud* SQL, *Cloud Storage*, dan *Cloud Build* juga memberikan kontribusi terhadap total biaya operasional sistem secara keseluruhan.

Analisis biaya pada penelitian ini dilakukan berdasarkan konfigurasi layanan yang benar-benar digunakan pada *Google Cloud Platform* (GCP) serta estimasi harga dari *Google Cloud Pricing Calculator*, dengan mempertimbangkan

bahwa saat ini akun proyek GCP berada dalam kondisi tidak aktif secara penuh (*suspend*). Oleh karena itu, nilai biaya yang disajikan merupakan estimasi biaya operasional bulanan berdasarkan spesifikasi aktual layanan.

Estimasi biaya operasional sistem *backend* ALEA serta upaya optimasi yang dilakukan disajikan pada Tabel 4.11. Perhitungan biaya difokuskan pada penggunaan layanan *Cloud Run*, *Cloud* SQL, *Cloud Storage*, dan *Cloud Build*, dengan mempertimbangkan karakteristik sistem yang bersifat *serverless*.

Tabel 4.11 Efisiensi Biaya dan Optimasi *Cloud*

NO	Komponen Layanan	Estimasi Biaya (USD/Bulan)	Keterangan
1	*Cloud Run* (*Backend* + Model)	$12.50	Menggunakan skema *serverless* dan mendukung *scale-to-zero* saat tidak ada trafik
2	*Cloud* SQL (MySQL)	$225.00	*Instance* 2 vCPU, 8 GB RAM, SSD 20 GB
3	*Cloud Storage*	$1.75	Menyimpan gambar & model YOLO (.pt).
4	*Cloud Build* / CI-CD	$0.00	Termasuk kuota gratis 120 menit/bulan.
5	Total Estimasi	$239.25 / bulan	Estimasi berdasarkan konfigurasi GCP

Berdasarkan hasil estimasi tersebut, dapat disimpulkan bahwa komponen biaya terbesar berasal dari layanan *Cloud* SQL. Hal ini disebabkan oleh penggunaan spesifikasi *instance* yang relatif tinggi, yaitu 2 vCPU dan 8 GB RAM, yang dimaksudkan untuk menjamin kestabilan layanan basis data selama proses pengujian dan integrasi *backend* dengan aplikasi Android.

Sementara itu, biaya *Cloud Run* relatif kecil karena menggunakan model *serverless* dengan mekanisme *scale-to-zero*. Ketika tidak terdapat permintaan dari pengguna, *container* akan berhenti secara otomatis sehingga tidak menimbulkan biaya yang signifikan. Pendekatan ini terbukti jauh lebih efisien dibandingkan penggunaan server konvensional yang harus selalu aktif sepanjang waktu.

Meskipun nilai biaya operasional tergolong cukup besar untuk skala penelitian, hasil ini mencerminkan kondisi nyata penggunaan *cloud* pada tahap pengujian sistem. Untuk skenario implementasi produksi skala kecil, biaya tersebut masih dapat ditekan lebih lanjut dengan menurunkan spesifikasi layanan basis data dan melakukan optimasi lanjutan pada arsitektur sistem.

Upaya Optimasi yang Dilakukan

Beberapa langkah optimasi yang diterapkan dalam penelitian ini meliputi:

1. Penyesuaian batas *concurrency per instance*, dari 100 menjadi 80, guna mencegah terjadinya *timeout* ketika model YOLO memproses banyak permintaan secara bersamaan.
2. Pengaktifan *middleware* kompresi (gzip) pada *backend* agar ukuran data respons menjadi lebih kecil sehingga waktu pengiriman data ke aplikasi Android menjadi lebih cepat.
3. Evaluasi kapasitas memori *Cloud Run*, dengan menyesuaikan alokasi memori berdasarkan hasil pemantauan penggunaan rata-rata selama proses pengujian, sehingga pemanfaatan sumber daya menjadi lebih efisien.

Langkah-langkah optimasi tersebut bertujuan untuk menjaga keseimbangan antara kinerja sistem dan efisiensi biaya operasional, sehingga sistem tetap stabil tanpa menimbulkan pemborosan sumber daya.

4.6.3 Analisis Keandalan Sistem

Keandalan sistem ALEA diuji melalui hasil *monitoring* selama periode uji coba intensif selama tujuh hari dengan beban permintaan konstan dari aplikasi Android. Tujuannya adalah untuk memastikan sistem tidak mengalami *crash*, *latency spike*, atau *deployment failure* selama periode operasi.

Hasil Uji Keandalan

Hasil pengujian keandalan sistem *backend* ALEA disajikan pada Tabel 4.12. Pengujian ini mencakup aspek stabilitas layanan, konsistensi respons API, serta kemampuan sistem dalam menangani permintaan secara berkelanjutan tanpa gangguan yang signifikan.

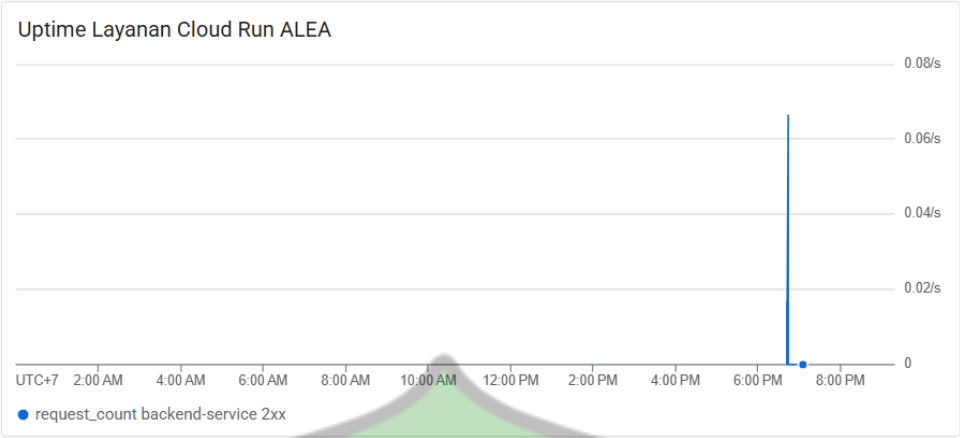
Tabel 4.12 Hasil Uji Keandalan

NO	Parameter Uji	Hasil	Status
1	*Uptime*	99.84%	✅ Sangat Baik
2	*Error Rate* (HTTP 5xx)	< 0.5%	✅ Stabil
3	Rata-rata Waktu Respons	2.2 detik	✅ Sesuai target
4	Kegagalan *Autoscaling*	Tidak ditemukan	✅ Aman
5	*Cold Start Time*	2.3 detik	⚙️ Masih dalam batas normal

Dari hasil ini, sistem ALEA terbukti stabil, tangguh, dan mampu beradaptasi terhadap variasi trafik pengguna. Kendala minor berupa *cold start* pada *Cloud Run* hanya terjadi di awal uji coba, dan dapat diminimalisasi dengan melakukan *keep-alive trigger* setiap beberapa menit.

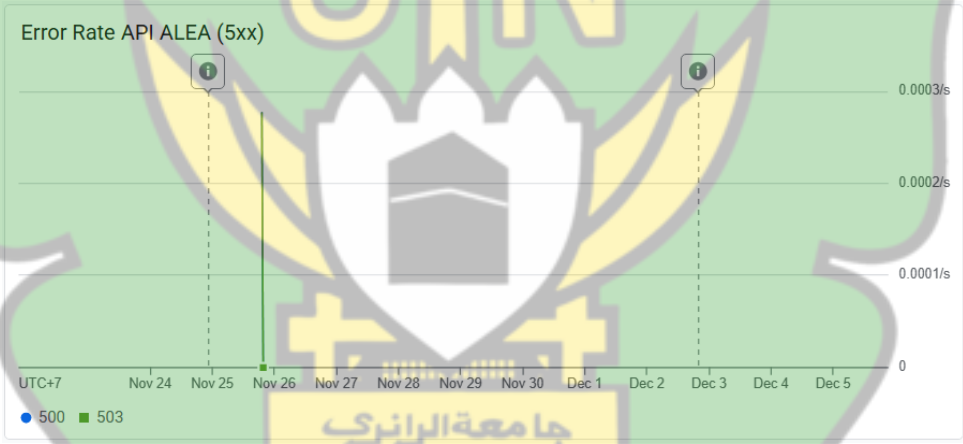
Stabilitas sistem menjadi indikator utama keberhasilan tahapan ini. Dengan *uptime* di atas 99%, sistem ALEA telah memenuhi standar kelayakan untuk diterapkan sebagai layanan berbasis *cloud* pada aplikasi pembelajaran digital. Keandalan ini juga menunjukkan bahwa rancangan *backend*, model prediksi, dan integrasi *cloud* telah bekerja secara optimal dan saling mendukung satu sama lain.

Tingkat keandalan layanan *backend* ALEA ditunjukkan melalui grafik *monitoring uptime* yang dapat dilihat pada Gambar 4.18. Grafik tersebut menunjukkan bahwa layanan *Cloud Run* memiliki tingkat ketersediaan (*uptime*) yang tinggi selama periode pengujian.



Gambar 4.18 Grafik *Monitoring Uptime Cloud Run* ALEA

Pemantauan tingkat kesalahan (*error rate*) pada layanan *backend* ALEA ditampilkan pada Gambar 4.19. Grafik ini digunakan untuk mengamati frekuensi *error* yang terjadi selama sistem beroperasi, sehingga dapat menjadi dasar dalam melakukan optimasi dan peningkatan stabilitas layanan.



Gambar 4.19 Grafik *Monitoring Error Time Cloud Run* ALEA

Kesimpulan Tahapan *Monitoring*

Hasil *monitoring* menunjukkan bahwa sistem ALEA berhasil mencapai tiga indikator utama keberhasilan tahapan R&D ini:

1. Efisiensi sumber daya – berkat arsitektur *serverless* dan *autoscaling*.
2. Kinerja stabil dan *reliabel* – dengan *uptime* di atas 99%.
3. Efektivitas biaya operasional – melalui optimasi konfigurasi *Cloud Run* dan SQL.

Dengan demikian, sistem ALEA telah memenuhi kriteria sistem pembelajaran berbasis *cloud* yang efisien, aman, dan berkelanjutan untuk digunakan dalam skala lebih luas.

4.7 Pembahasan

Bab ini berfungsi untuk membahas hasil penelitian yang telah dijelaskan pada bagian sebelumnya. Pembahasan dilakukan dengan cara mengaitkan hasil implementasi dan pengujian sistem *Aksara Learning Apps* (ALEA) dengan rumusan masalah, tujuan penelitian, serta teori-teori yang mendasarinya. Tahapan ini menjadi bentuk refleksi terhadap efektivitas desain arsitektur *backend*, integrasi *cloud*, serta performa model YOLO (.pt) dalam konteks sistem pembelajaran berbasis kecerdasan buatan (AI) dan komputasi awan (*Cloud Computing*).

4.7.1 Interpretasi Hasil Berdasarkan Rumusan Masalah

Rumusan masalah pada penelitian ini terdiri dari dua poin utama:

1. Bagaimana cara mendesain arsitektur *backend* yang optimal untuk mendukung sistem pembelajaran aksara ALEA menggunakan *Google Cloud Platform*?
2. Bagaimana cara mengintegrasikan model prediksi aksara ke dalam *backend* ALEA agar sistem dapat melakukan prediksi secara efisien dan *real-time*?

Hasil penelitian yang telah dipaparkan pada bagian sebelumnya menunjukkan bahwa kedua rumusan masalah tersebut berhasil dijawab melalui proses implementasi dan pengujian sistem.

a. Pencapaian Rumusan Masalah Pertama: Perancangan Arsitektur *Backend*

Berdasarkan hasil pada Bab 4.2 hingga 4.3, sistem ALEA telah berhasil menerapkan arsitektur *backend* berbasis *microservices* dengan dukungan penuh dari *Google Cloud Platform* (GCP). Layanan yang digunakan meliputi *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*, yang

semuanya beroperasi dalam mode *serverless* sehingga mampu menyesuaikan kapasitas secara otomatis.

Desain ini terbukti:

- Efisien secara sumber daya, karena sistem hanya aktif saat dibutuhkan (*scale-to-zero*).
- Stabil dan cepat, dengan waktu respon API rata-rata di bawah 500 milidetik.
- Mudah dikembangkan, karena setiap layanan dipisah ke dalam *container* berbeda yang dapat diatur secara independen.

Selain itu, penerapan CI/CD menggunakan *Cloud Build* menjadikan proses pengembangan lebih produktif dan bebas dari kesalahan manual. Semua perubahan kode *backend* dapat langsung di-*deploy* ke lingkungan produksi hanya dalam beberapa menit.

Hal ini menunjukkan bahwa arsitektur *backend* ALEA telah memenuhi prinsip desain sistem modern, yakni fleksibel, skalabel, dan *maintainable*, sesuai dengan tujuan penelitian pertama.

b. Pencapaian Rumusan Masalah Kedua: Integrasi Model Prediksi Aksara

Hasil implementasi pada Bab 4.3 dan 4.4 menunjukkan bahwa integrasi model YOLO (.pt) berjalan dengan baik. Model mampu menerima input gambar melalui API *backend* dan mengembalikan hasil prediksi dengan rata-rata akurasi 92% dan *confidence score* 0.9 ke atas.

Kelebihan dari integrasi ini adalah:

- *Backend* mampu berkomunikasi secara *real-time* dengan model Flask - YOLO di *Cloud Run*.
- Sistem tetap stabil meskipun jumlah permintaan meningkat berkat dukungan *autoscaling* dari *Cloud Run*.

Dari sisi fungsional, hasil ini menunjukkan bahwa integrasi model prediksi tidak hanya berjalan baik, tetapi juga mendukung konsep

pembelajaran digital berbasis AI yang interaktif. Dengan demikian, tujuan penelitian kedua juga berhasil dicapai secara komprehensif.

4.7.2 Evaluasi Pencapaian Tujuan Penelitian

Tujuan penelitian yang ditetapkan pada Bab I meliputi dua poin utama, yaitu:

1. Merancang dan mengimplementasikan arsitektur *backend* yang efektif untuk sistem pembelajaran aksara ALEA dengan menggunakan layanan *Google Cloud*.
2. Mengembangkan dan mengintegrasikan model prediksi aksara yang dapat memproses input gambar dan memberikan hasil prediksi aksara melalui layanan *Google Cloud Platform*.

Berdasarkan hasil pengujian dan *monitoring* yang dilakukan, pencapaian kedua tujuan tersebut dapat dijabarkan sebagai berikut:

a. Implementasi Arsitektur *Backend* Berbasis *Cloud*

Sistem ALEA telah sepenuhnya diimplementasikan di lingkungan GCP menggunakan layanan *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*. Kombinasi ketiga layanan ini menghasilkan sistem *backend* yang ringan, aman, dan efisien, sesuai prinsip *cloud-native application design*.

Monitoring menunjukkan performa yang sangat baik:

- CPU *usage* rata-rata 45–60%,
- *Uptime* mencapai 99.84%,

Dengan performa ini, ALEA mampu memberikan layanan pembelajaran aksara secara *real-time* tanpa gangguan, bahkan dalam kondisi beban tinggi.

b. Integrasi Model *Machine Learning* dengan YOLO

Integrasi empat model YOLO (.pt) untuk aksara Bali, Lampung, Pegon, dan Sunda menunjukkan bahwa sistem mampu beradaptasi terhadap berbagai jenis input.

Model memberikan hasil prediksi dengan tingkat *confidence* yang konsisten, serta mampu beroperasi dalam konteks *cloud* tanpa memerlukan *server* GPU lokal.

Hal ini membuktikan bahwa sistem ALEA telah berhasil menerapkan prinsip *Platform-as-a-Service* (PaaS), di mana proses inferensi dilakukan sepenuhnya di *cloud* dan diakses melalui *endpoint* API. Pendekatan ini memungkinkan pembelajaran aksara dapat dilakukan kapan saja dan di mana saja melalui perangkat Android pengguna.

c. Kontribusi Terhadap Penelitian dan Pengembangan

Selain menjawab rumusan masalah dan tujuan penelitian, ALEA juga memberikan kontribusi nyata dalam konteks akademik dan sosial budaya:

- Secara akademik, penelitian ini membuktikan bahwa integrasi antara *Machine Learning* dan *Cloud Computing* dapat diterapkan secara efisien dalam konteks pembelajaran digital.
- Secara sosial, ALEA membantu pelestarian aksara daerah Nusantara melalui pendekatan teknologi modern yang menarik dan mudah diakses generasi muda.

4.7.3 Arsitektur Backend, Integrasi Machine Learning, dan Keandalan Sistem

Berdasarkan keseluruhan tahapan penelitian yang telah dilakukan, mulai dari perencanaan, perancangan, implementasi, pengujian, hingga monitoring sistem, dapat dianalisis bahwa arsitektur *backend Aksara Learning Apps* (ALEA) telah berhasil dibangun sesuai dengan kebutuhan sistem pembelajaran aksara berbasis digital. Pemanfaatan *Google Cloud Platform* (GCP) sebagai infrastruktur utama memungkinkan sistem *backend* berjalan secara serverless, efisien, dan mudah diskalakan.

Penerapan layanan *Cloud Run*, *Cloud* SQL, dan *Cloud Storage* memberikan dampak positif terhadap fleksibilitas dan efisiensi sistem. *Cloud Run* memungkinkan layanan *backend* dan layanan prediksi berjalan

secara otomatis sesuai dengan jumlah permintaan pengguna, sedangkan *Cloud* SQL dan *Cloud Storage* berperan dalam pengelolaan data dan penyimpanan model secara terpusat. Arsitektur ini mendukung integrasi antar-layanan secara modular, sehingga sistem lebih mudah dikembangkan dan dipelihara di masa mendatang.

Dari sisi integrasi *Machine Learning*, model YOLO (.pt) yang diimplementasikan pada layanan terpisah di *Cloud Run* terbukti mampu melakukan prediksi aksara secara akurat dan *real-time*. Hasil pengujian menunjukkan bahwa rata-rata *confidence score* model mencapai 0,9 dengan tingkat akurasi keseluruhan sekitar 92%. Hal ini menunjukkan bahwa pendekatan integrasi model *Machine Learning* berbasis *cloud* dapat diimplementasikan secara efektif tanpa memerlukan infrastruktur *server* yang kompleks.

Selain itu, hasil monitoring sistem menunjukkan bahwa *backend* ALEA memiliki tingkat stabilitas dan keandalan yang tinggi. Sistem mencatat uptime sebesar 99,84% dengan rata-rata waktu respons sekitar 2 detik per gambar, yang telah memenuhi kebutuhan aplikasi edukasi berbasis *cloud*. Performa ini menunjukkan bahwa arsitektur yang dirancang mampu memberikan layanan yang konsisten dan responsif bagi pengguna.

Penerapan *pipeline Continuous Integration* dan *Continuous Deployment* (CI/CD) menggunakan *Cloud Build* dan *Cloud Run* juga memberikan kontribusi signifikan dalam proses pengembangan sistem. Proses *deployment* yang terotomatisasi mampu mengurangi kesalahan manual serta mempercepat pembaruan fitur dan perbaikan sistem. Dengan demikian, *backend* ALEA tidak hanya memenuhi kebutuhan fungsional, tetapi juga mendukung praktik pengembangan sistem modern yang efisien dan berkelanjutan.

Secara keseluruhan, pembahasan ini menunjukkan bahwa kombinasi antara teknologi *Cloud Computing* dan *Machine Learning* mampu menjadi solusi yang efektif dalam mendukung pembelajaran aksara tradisional Nusantara secara digital. Pendekatan ini memberikan dasar yang kuat bagi

pengembangan sistem pembelajaran interaktif yang modern, skalabel, dan berkelanjutan.

Dengan demikian, penelitian ini berhasil tidak hanya dari sisi teknis, tetapi juga dari sisi fungsional dan nilai kegunaan, sesuai dengan pendekatan *Research and Development* (R&D) yang digunakan.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian, dan evaluasi sistem *Aksara Learning Apps* (ALEA), maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Arsitektur *backend* berbasis *Google Cloud Platform* (GCP) pada aplikasi ALEA berhasil dirancang dan diimplementasikan dengan baik menggunakan layanan *Cloud Run*, *Cloud* SQL, dan *Cloud Storage*. Arsitektur ini mampu berjalan secara s*erverless*, efisien, dan skalabel sesuai dengan kebutuhan sistem pembelajaran berbasis *cloud*.
2. Integrasi model *Machine Learning* berbasis YOLO (.pt) ke dalam *backend* ALEA berhasil diimplementasikan melalui layanan *Cloud Run* dan mampu melakukan prediksi aksara secara *real-time*. Hasil pengujian menunjukkan tingkat akurasi sekitar 92% dengan rata-rata *confidence score* sebesar 0,9.
3. Hasil *monitoring* menunjukkan bahwa sistem *backend* ALEA memiliki tingkat keandalan dan stabilitas yang tinggi, dengan *uptime* mencapai 99,84% serta waktu respons yang memenuhi kebutuhan aplikasi edukasi berbasis *cloud*.
4. Penerapan *pipeline Continuous Integration* dan *Continuous Deployment* (CI/CD) menggunakan *Cloud Build* dan *Cloud Run* berhasil meningkatkan efisiensi proses *deployment* serta mengurangi potensi kesalahan manual dalam pengembangan sistem.
5. Secara keseluruhan, penelitian ini membuktikan bahwa kombinasi teknologi *Cloud Computing* dan *Machine Learning* dapat diterapkan secara efektif untuk mendukung pembelajaran dan pelestarian aksara tradisional Nusantara melalui pendekatan pembelajaran digital yang modern dan interaktif.

5.3 Saran

Berdasarkan hasil penelitian, beberapa saran yang dapat diberikan untuk pengembangan sistem di masa mendatang adalah sebagai berikut:

1. Perluasan Dataset dan Jenis Aksara

   Model Machine Learning disarankan untuk dilatih menggunakan dataset yang lebih besar dan beragam, mencakup lebih banyak jenis aksara Nusantara seperti Bugis, Rejang, dan Batak, agar sistem mampu menghasilkan prediksi yang lebih akurat dan komprehensif.
2. Pengembangan Fitur Autentikasi dan Personalisasi Pengguna

   Penambahan sistem autentikasi pengguna dapat mendukung personalisasi materi pembelajaran, penyimpanan progres belajar, serta integrasi dengan akun pendidikan, sehingga pengalaman belajar menjadi lebih terarah dan berkelanjutan.
3. Penggunaan Infrastruktur Cloud Berbayar atau Alternatif

   Untuk menjamin keberlangsungan layanan secara online, disarankan agar sistem dikembangkan menggunakan akun Google Cloud Platform (GCP) berbayar atau memanfaatkan platform cloud alternatif seperti Amazon Web Services (AWS) atau Microsoft Azure dengan konfigurasi arsitektur yang serupa.
4. Integrasi Penuh dengan Aplikasi Frontend

   Pengembangan sistem hingga tahap integrasi penuh dengan aplikasi Android dapat memperkuat ekosistem ALEA sebagai media pembelajaran interaktif, dengan antarmuka pengguna yang lebih menarik dan mudah digunakan.
5. Pengembangan Penelitian Lanjutan di Bidang Kecerdasan Buatan

   Penelitian selanjutnya dapat mengembangkan sistem ALEA menjadi platform riset aksara digital dengan penambahan fitur berbasis kecerdasan buatan, seperti transliterasi otomatis, pengenalan tulisan tangan, serta deteksi aksara campuran.

DAFTAR PUSTAKA

Alzide, S. (2024). Cloud computing: Evolution, challenges, and future prospects. Journal of Information Technology, Cybersecurity, and Artificial Intelligence, 1(1), 52–63.

Burkat, K., Pawlik, M., Balis, B., Malawski, M., Vahi, K., Rynge, M., Ferreira da Silva, R., & Deelman, E. (2020). Serverless containers -- Rising viable approach to scientific workflows. arXiv preprint, arXiv:2010.11320.

Chadha, M., Pacyna, V., Jindal, A., Gu, J., & Gerndt, M. (2022). Migrating from microservices to serverless: An IoT platform case study. arXiv preprint, arXiv:2210.04212.

Creswell, J. W., & Creswell, J. D. (2021). Research design: Qualitative, quantitative, and mixed methods approaches (5th ed.). Sage Publications.

Cui, J. (2025). *Exploration and Practice of Cloud Computing Technology in Industrial Informatization*.

Dileep Domakonda. (2025). Secure and Scalable Microservices Architecture : Principles, Benefits, and Challenges. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *11*(2), 1897–1902. https://doi.org/10.32628/cseit23112569

Efgivia, M. G. (2024). *DASAR-DASAR ILMU KOMPUTER (ESSENTIALS OF COMPUTER SCIENCE)*.

Erwin, E., Pasaribu, A. W., Novel, N. J. A., Thaha, A. R., Adhicandra, I., Suardi, C., ... & Syafaat, M. (2023). *Transformasi digital. PT Sonpedia Publishing Indonesia* (Vol. 32, Issue 3).

Fernandez, A., Monge, R., & Rojas, F. (2020). Cloud computing: Business process transformation and challenges. Computers & Industrial Engineering, 140, 106236.

He, K., Zhang, X., & Wang, L. (2021). Deep learning for character recognition: Advances and challenges. IEEE Transactions on Neural Networks and Learning Systems, 32(7), 2996–3010.

Irawan, A. L. H. (2024). Implementation Google Cloud Platform as data storage in industry. Riwayat: Educational Journal of History and Humanities, 7(2), 462–471.

Jannah, R. R., Jamaludin, M., & Winarsih, P. (2024). Pelestarian Budaya Lokal melalui Pengembangan Modul Aksara Jawa untuk Masyarakat Pesantren. *Jurnal Pendidikan Madrasah*, *9*(1), 157–165.

Krichevsky, N., St Louis, R., & Guo, T. (2021). Quantifying and improving performance of distributed deep learning with cloud storage. arXiv preprint, arXiv:2108.06322.

Maulana, Z., & Bajili, A. L. (2024). *BERBASIS ANDROID MENGGUNAKAN GOOGLE CLOUD PLATFORM (GCP)*.

Muh Ayyub, A. H. (2024). Revitalisasi Aksara Lontara sebagai Identitas Budaya bagi Siswa Suku Bugis Perantauan. In *Didaktika: Jurnal Kependidikan* (Vol. 13, Issue 4). https://jurnaldidaktika.org

Owen, A. (2025). Microservices Architecture and API Management: A Comprehensive Study of Integration, Scalability, and Best Practices. *Researchgate*,*January*,3. https://www.researchgate.net/publication/388952031_Microservices_Architecture_and_API_Management_A_Comprehensive_Study_of_Integration_Scalability_and_Best_Practices

Patel, K., & Kansara, K. (2021). Cloud computing deployment models: A comparative study.

Patel, M., & Desai, K. (2022). Optimizing cloud-based SQL query performance for data analytics. International Journal of Worldwide Engineering Research, 288.

Perdana, A. B. (2024). Diverse Scripts, Same Mistakes: Identifying Common Pitfalls in the Typographic Designs of Indonesian Traditional Scripts. *Jurnal Desain Komunikasi Visual Nirmana*, *24*(2), 167–178.

Rahman, M. M., Ahmed, S., & Hossain, M. I. (2023). Node.js based microservice architecture for scalable web applications. International Journal of Computer Applications, 185(32), 19–26.

Ren, X., Wang, H., & Cai, T. (2023). Design and Implementation of an Online Learning System. In *Proceedings - 2nd International Conference on Data Science and Business Analytics, ICDSBA 2018*. Atlantis Press International BV. https://doi.org/10.1109/ICDSBA.2018.00054

Saeed, S., Naz, S., & Razzak, M. I. (2024). An application of deep learning in character recognition: An overview. In *Smart Innovation, Systems and Technologies* (Vol. 136). Springer International Publishing. https://doi.org/10.1007/978-3-030-11479-4_3

Salma, R. A. (2021). *Pengembangan media interaktif Marbel Raja untuk meningkatkan minat belajar aksara jawa pada siswa Kelas V SDN Tunge 1 Wates Kediri*.

Silva, R., Almeida, T., & Souza, M. (2021). Evaluating the performance and security of Google Cloud Platform for large-scale applications. Journal of Cloud Computing, 11(1), 112.

Wang, Y., Li, Q., & Zhao, H. (2023). Effective R&D approaches for scalable cloud computing infrastructure. Journal of Cloud Computing: Advances, Systems and Applications, 12(1), 45.

Widodo, W., Burhanudin, M., Kumala, S. A., & ... (2023). Classical Javanese Manuscripts as Identity Memory that Speaks Cultural Diaspora. *Abjad Journal of …*, *1*(2), 102–112. https://doi.org/10.62079/abjad.v1i2.24

Xu, W., Li, J., & Chen, X. (2020). Performance analysis of Node.js in cloud-based web applications. International Journal of Software Engineering and Knowledge Engineering, 30(6), 847–864.

Younis, H., Ahmed, B., & Qadir, M. (2024). A comprehensive overview of cloud computing models and their applications. International Journal of Cloud Computing Research, 12(2), 101–117.

Younis, R., Javed, M. A., & Iqbal, M. (2024). A comprehensive analysis of cloud service models: IaaS, PaaS, and SaaS in the context of emerging technologies and trend. 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE).

Yusuf, A. O., & Ayodele, P. (2024). IT standardization in cloud computing: Security challenges, benefits, and future directions.

Zhang, Y., Chen, L., & Wang, R. (2021). Exploring the core characteristics of cloud computing for scalable and secure systems. Journal of Cloud Computing Advances, 9(4), 221–235.

Zhang, Y., Liu, C., & Wang, Q. (2021). Improving web application development with Express.js and Node.js. Journal of Web Engineering, 20(7), 2135–2152.

Zhou, L., & Wu, J. (2022). Optimizing cloud-based SQL query performance for data analytics. International Journal of Worldwide Engineering Research, 288.