# INTEGRASI MODEL BAHASA GEMMA UNTUK SISTEM TERJEMAHAN DUA ARAH ANTARA PSEUDOCODE DAN BAHASA PEMROGRAMAN BERBASIS MOBILE

**Rangga Wahyu Pratama, Paskah Abadi Simanullang,**
**Peter Tymoty Hutabarat, Sirus Daniel H Nababan, Adidtya Perdana**
Ilmu Komputer, Universitas Negeri Medan
*ranggawahyupratama386@gmail.com*

**ABSTRAK**

Perkembangan kecerdasan buatan (Artificial Intelligence) khususnya dalam bidang Natural Language Processing (NLP) telah memungkinkan pengembangan sistem penerjemahan cerdas antara bahasa manusia dan bahasa pemrograman. Penelitian ini bertujuan untuk mengimplementasikan integrasi model bahasa Gemma 3:4B dalam sistem penerjemahan dua arah antara pseudocode dan bahasa pemrograman berbasis mobile. Metode pengembangan yang digunakan adalah Waterfall, dengan tahapan analisis kebutuhan, perancangan, implementasi, pengujian, dan pemeliharaan. Sistem dibangun menggunakan tiga komponen utama, yaitu Flutter sebagai frontend untuk antarmuka pengguna, PHP sebagai backend autentikasi dan manajemen pengguna, serta Python FastAPI sebagai layanan penerjemahan yang terhubung ke model Gemma melalui API Ollama. Pengujian dilakukan dengan metode Black Box Testing terhadap sepuluh fitur utama, dan seluruh fungsi berjalan sesuai harapan tanpa ditemukan kesalahan fatal. Hasil penelitian menunjukkan bahwa sistem mampu menerjemahkan pseudocode ke bahasa pemrograman (C++, Java, Python) maupun sebaliknya dengan akurasi yang baik dan waktu respons yang cepat. Kesimpulannya, integrasi model Gemma pada sistem ini efektif dalam membantu proses pembelajaran algoritma dan pemrograman secara interaktif melalui perangkat mobile.

***Kata kunci :*** *Gemma, Large Language Model, Mobile Application, Pseudocode, Translation System*

## 1. PENDAHULUAN

Perkembangan teknologi kecerdasan buatan (Artificial Inteligence/AI) mengalami kemajuan pesat dalam dua dekade terakhir, terutama pada bidang Natural Language Processing (NLP) yang memungkinkan komputer memahami, menghasilkan, dan memanipulasi bahasa manusia secara alami[1]. Salah satu inovasi yang menarik perhatian dalam ranah ini adalah kemunculan *Large Language Model* (LLM) seperti Gemma, yang mampu memahami konteks semantik dan sintaksis dengan akurasi tinggi[2]. Model semacam ini telah banyak dimanfaatkan dalam berbagai bidang, termasuk pendidikan, linguistik komputasional, dan pengembangan perangkat lunak berbasis teks[3].

Dalam konteks pendidikan dan rekayasa perangkat lunak, kemampuan untuk menjembatani bahasa manusia dan bahasa pemrograman menjadi kebutuhan yang semakin penting. Proses penerjemahan dua arah antara *pseudocode* dan bahasa pemrograman membantu pengguna, khususnya mahasiswa dan pengembang pemula, memahami keterkaitan antara logika algoritmik dengan implementasi nyata di sistem komputer[4]. Proses penerjemahan tidak hanya sebatas pengalihan kata, tetapi juga mempertahankan kesepadanan makna dan struktur logika[3]. Sementara itu, penelitian menunjukkan bahwa sekitar 73% kesalahan penerjemahan otomatis disebabkan oleh ketidaksesuaian sintaksis, dan 27% lainnya oleh kesalahan semantik[2].

Dalam pembelajaran algoritma, *pseudocode* berfungsi sebagai jembatan antara bahasa alami dan bahasa pemrograman formal. Penggunaan *pseudocode* mempermudah pemahaman struktur algoritma karena menyerupai bahasa manusia[5]. Pembelajaran algoritma melalui representasi *pseudocode* dapat meningkatkan kemampuan berpikir komputasional mahasiswa[6]. Algoritma yang ditulis dalam bentuk *pseudocode* dapat membantu mengurangi kesalahan logika serta mempermudah proses *debugging*[7].

Beberapa penelitian terdahulu juga menyoroti hubungan antara linguistik, penerjemahan, dan pembelajaran komputasi. Pendekatan Linguistik Fungsional Sistemik (*Systemic Functional Linguistics*) dapat membantu mengidentifikasi elemen makna dalam teks teknis, yang relevan untuk penerjemahan berbasis AI[1]. Penerapan teknik penerjemahan harus mempertimbangkan keterbacaan dan kesesuaian istilah teknis dalam bidang komputasi. Hal ini penting agar sistem terjemahan berbasis AI tidak hanya akurat secara sintaksis, tetapi juga relevan secara semantik bagi pengguna[8].

Di sisi lain, integrasi teknologi AI pada platform mobile juga menjadi perhatian dalam penelitian terkini. Aspek *usability*—seperti kemudahan, efisiensi, keandalan, dan daya tarik—menjadi faktor penting dalam pengembangan aplikasi mobile[9]. Implementasi algoritma *sequential search* untuk mendukung fitur pencarian dalam aplikasi donasi, yang membuktikan efektivitas algoritma dalam konteks mobile[10].

Tingkat penerimaan teknologi sangat dipengaruhi oleh persepsi kemudahan penggunaan dan kemanfaatan, sebagaimana dijelaskan dalam model TAM (*Technology Acceptance Model*)[11].

Berdasarkan latar belakang tersebut, penelitian ini berfokus pada pengembangan sistem penerjemahan dua arah antara *pseudocode* dan bahasa pemrograman berbasis mobile dengan memanfaatkan model bahasa Gemma. Sistem ini diharapkan dapat membantu pengguna dalam memahami alur logika algoritma sekaligus menghasilkan kode yang dapat dijalankan langsung pada perangkat mobile. Rumusan masalah dalam penelitian ini meliputi: (1) bagaimana memanfaatkan model bahasa Gemma untuk melakukan translasi dua arah antara *pseudocode* dan bahasa pemrograman mobile, (2) bagaimana merancang arsitektur sistem terjemahan yang efektif dan efisien untuk mendukung integrasi model bahasa Gemma, serta (3) sejauh mana sistem ini dapat membantu pengguna dalam memahami dan mengonversi *pseudocode* menjadi kode program yang dapat dieksekusi di platform mobile.

Adapun tujuan penelitian ini adalah untuk: (1) mengimplementasikan integrasi model bahasa Gemma dalam sistem terjemahan dua arah antara *pseudocode* dan bahasa pemrograman mobile, (2) merancang dan mengembangkan prototipe aplikasi mobile yang mendukung proses terjemahan tersebut, dan (3) mengevaluasi kinerja sistem dalam hal akurasi translasi, kecepatan, serta kemudahan penggunaan. Penelitian ini diharapkan dapat memberikan kontribusi terhadap peningkatan efektivitas pembelajaran algoritma, efisiensi proses pengembangan aplikasi mobile, serta menjadi acuan bagi pengembangan sistem penerjemahan kode berbasis AI di masa depan.

## 2. TINJAUAN PUSTAKA

Perkembangan kecerdasan buatan (*Artificial Intelligence* / AI) saat ini telah membawa dampak signifikan dalam berbagai bidang, termasuk pendidikan dan pengembangan perangkat lunak. Salah satu cabang AI yang mengalami kemajuan pesat adalah *Natural Language Processing* (NLP), yaitu bidang yang berfokus pada kemampuan mesin untuk memahami, mengolah, dan menghasilkan bahasa manusia secara alami Penelitian yang dilakukan oleh Indarti dkk. (2024) menunjukkan bahwa AI dapat mendukung pembelajaran bahasa secara adaptif dan personal melalui pelatihan dan integrasi teknologi yang sistematis di ruang kelas. NLP menjadi dasar dari munculnya berbagai model bahasa besar (*Large Language Model* / LLM) seperti Gemma, GPT, dan PaLM, yang memiliki kemampuan memahami konteks semantik dan sintaksis secara mendalam[12].

### 2.1. Pseudocode dalam Pembelajaran Algoritma

Pseudocode berfungsi sebagai representasi logika algoritma menggunakan bahasa semi-formal yang mudah dipahami manusia. Fajri (2022) menyatakan bahwa penggunaan pseudocode dalam pembelajaran algoritma dapat meningkatkan kemampuan berpikir komputasional mahasiswa secara signifikan[6]. Sistem AI mampu mendeteksi pola linguistik non-formal dengan baik, sehingga membantu proses konversi antara representasi linguistik (pseudocode) dan bahasa pemrograman. Pseudocode berperan penting sebagai jembatan antara bahasa alami dan bahasa pemrograman formal, karena dapat mempermudah proses analisis dan *debugging* logika program[13].

Beberapa penelitian juga menunjukkan bahwa pseudocode dapat mengurangi tingkat kesalahan sintaksis pada tahap awal pemrograman . Dengan mengonversi pseudocode ke bahasa pemrograman yang spesifik, pengguna dapat lebih mudah memahami bagaimana logika algoritmik direalisasikan ke dalam kode yang dapat dieksekusi.

### 2.2. Model Bahasa Besar (Large Language Model) dalam Sistem Terjemahan

LLM seperti Gemma 3:4B dirancang untuk memahami pola linguistik dan logika komputasional secara bersamaan. Kesalahan terjemahan otomatis umumnya terjadi akibat ketidaksesuaian antara sintaksis (73%) dan semantik (27%) . Hal ini memperlihatkan pentingnya *prompt engineering* dan *context control* dalam sistem penerjemahan berbasis AI[2].

Dalam sistem terjemahan dua arah antara pseudocode dan bahasa pemrograman, LLM digunakan untuk memahami hubungan logika antara input dan output tanpa kehilangan struktur algoritmik. Pendekatan ini juga selaras dengan metode linguistik fungsional sistemik (Systemic Functional Linguistics) yang berfokus pada pemetaan makna dalam konteks teknis[12]. Dengan demikian, sistem berbasis LLM seperti Gemma dapat berfungsi bukan hanya sebagai penerjemah literal, tetapi juga sebagai mediator semantik antara dua domain bahasa berbeda.

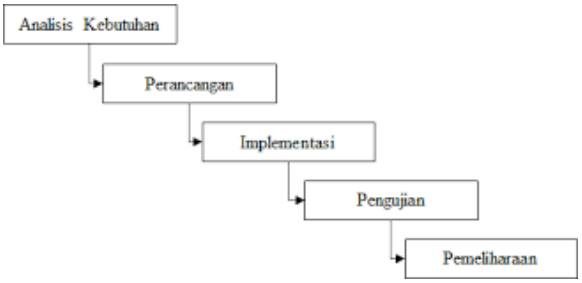
### 2.3. Penelitian Terdahulu

Beberapa penelitian sebelumnya telah membahas penerapan pseudocode dan sistem penerjemahan untuk mendukung pembelajaran algoritma. Pengembangan aplikasi pembelajaran rekonstruksi algoritma berbasis pseudocode yang membantu mahasiswa memahami logika dasar sebelum menulis kode program[14]. Pendekatan ini terbukti efektif dalam mengurangi kesalahan logika dan meningkatkan kemampuan analisis algoritmik. Peneraan teknik *Natural Language Processing* (NLP) dalam sistem penerjemah otomatis untuk meningkatkan akurasi sintaks dan makna pada hasil terjemahan[15].

Selanjutnya, merancang sistem penerjemah dua arah antara bahasa pemrograman dan pseudocode dengan metode *rule-based parsing* untuk mendukung pembelajaran algoritma interaktif[8]. Sistem ini mampu menerjemahkan struktur dasar seperti perulangan dan percabangan, meski masih terbatas pada pola sintaks sederhana. Pentingnya integrasi kecerdasan buatan dalam pembelajaran bahasa formal melalui pendekatan NLP adaptif berbasis konteks[16].

Dari hasil-hasil tersebut dapat disimpulkan bahwa penelitian terdahulu masih berfokus pada pendekatan berbasis aturan dengan keterbatasan dalam memahami konteks semantik. Oleh karena itu, penelitian ini mengusulkan sistem terjemahan dua arah berbasis *Large Language Model* Gemma untuk meningkatkan akurasi dan fleksibilitas penerjemahan pseudocode serta bahasa pemrograman dalam platform mobile.

## 3. METODE PENELITIAN




Gambar 1. Metode Waterfall

Penelitian ini menggunakan metode pengembangan perangkat lunak Waterfall, yang dikenal juga sebagai Software Development Life Cycle (SDLC) model klasik. Metode ini dipilih karena memiliki pendekatan yang sistematis, terstruktur, dan cocok digunakan untuk pengembangan aplikasi dengan kebutuhan yang sudah terdefinisi dengan jelas sejak awal. Model Waterfall terdiri atas tahapan berurutan di mana setiap tahap harus diselesaikan sebelum tahap berikutnya dimulai.

Pendekatan ini memastikan bahwa setiap proses — mulai dari analisis kebutuhan, desain sistem, implementasi, pengujian, hingga pemeliharaan — dilakukan dengan dokumentasi dan validasi yang menyeluruh. Kelebihan utama metode ini terletak pada kejelasan alur kerja, pengendalian proses yang ketat, serta kemudahan pelacakan kemajuan proyek. Karena sistem Translate App ini memiliki cakupan fungsi yang spesifik dan dependensi yang terpisah antara frontend, backend PHP, dan backend Python, model Waterfall dianggap paling sesuai untuk menjaga keteraturan dan keterpaduan antar komponen selama proses pengembangan. Adapun tahapan metode Waterfall yang diterapkan dalam penelitian ini meliputi.

### 3.1. Analisis Kebutuhan

Tahap ini mencakup proses identifikasi kebutuhan fungsional dan non-fungsional sistem. Analisis dilakukan terhadap kebutuhan pengguna (dosen, guru, mahasiswa, dan siswa) yang memerlukan alat bantu untuk menerjemahkan pseudocode ke bahasa pemrograman dan sebaliknya. Kebutuhan sistem juga meliputi autentikasi pengguna, pengelolaan profil, dan batas kuota harian translasi. Selain itu, dilakukan pula analisis teknologi pendukung seperti Flutter untuk frontend, PHP untuk layanan autentikasi, Python (FastAPI) untuk layanan penerjemahan, serta MySQL sebagai basis data utama.

### 3.2. Perancangan Sistem

Pada tahap ini dilakukan desain arsitektur aplikasi menggunakan pendekatan multi-tier client–server. Perancangan meliputi struktur folder proyek (frontend, backend PHP dengan Python), desain basis data, antarmuka pengguna (UI), serta alur komunikasi antar komponen melalui API. Desain sistem juga mencakup, Entity Relationship Diagram (ERD), dan rancangan antar muka (mockup) untuk memastikan kesesuaian antara kebutuhan pengguna dan implementasi teknis.

### 3.3. Implementasi

Tahap implementasi mencakup proses pembangunan sistem sesuai desain yang telah ditetapkan. Aplikasi dikembangkan secara modular — Flutter untuk tampilan dan interaksi pengguna, PHP untuk autentikasi dan manajemen pengguna, serta Python/FastAPI untuk layanan penerjemahan berbasis model bahasa Gemma 3:4B. Setiap komponen diintegrasikan melalui protokol HTTP menggunakan format data JSON agar komunikasi antar lapisan berjalan efisien dan terstandarisasi.

### 3.4. Pengujian

Setelah implementasi selesai, sistem diuji menggunakan metode black-box testing untuk memastikan bahwa setiap fungsi berjalan sesuai dengan kebutuhan pengguna. Pengujian dilakukan pada fitur login, register, update profil, serta proses penerjemahan dua arah (pseudocode ↔ bahasa pemrograman). Aspek yang diuji meliputi akurasi hasil terjemahan, waktu respons, validasi input, dan penanganan error seperti batas kuota harian serta koneksi API.

### 3.5. Pemeliharaan

Tahap ini dilakukan untuk memperbaiki bug, meningkatkan performa, serta menyesuaikan sistem dengan perubahan kebutuhan di masa depan. Pemeliharaan juga mencakup pengoptimalan performa model Gemma, peningkatan sistem antrian dan kuota pengguna, serta perbaikan antarmuka untuk meningkatkan pengalaman pengguna.

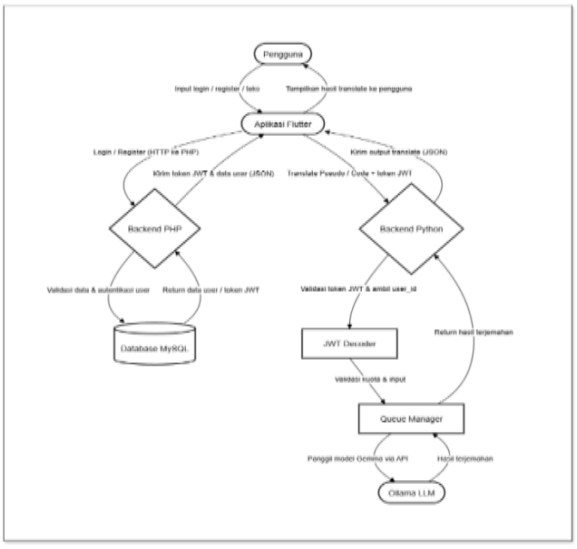
## 4. HASIL DAN PEMBAHASAN

Sistem penerjemah dua arah antara pseudocode dan bahasa pemrograman yang dikembangkan dalam penelitian ini terdiri atas tiga komponen utama, yaitu Frontend Flutter, Backend PHP, dan Backend Python. Arsitektur sistem ini dirancang dengan pendekatan client–server multi-tier agar setiap lapisan memiliki tanggung jawab terpisah, sehingga mudah dikembangkan dan dipelihara. Flutter berfungsi sebagai client-side application yang menyediakan antarmuka interaktif bagi pengguna, sementara PHP bertindak sebagai authentication service layer untuk

menangani registrasi, login, dan manajemen data pengguna. Adapun Python digunakan sebagai processing layer yang menjalankan layanan penerjemahan berbasis large language model (LLM) Gemma3:4b melalui API Ollama.

```
App Translate Dua Arah/                  # Root project utama
├── frontend_flutter/                    # Aplikasi client-side yang dibuat dengan Flutter
│   ├── lib/                             # Folder utama kode Flutter
│   │   ├── main.dart                    # Entry point Flutter app
│   │   ├── config/                      # Konfigurasi global
│   │   │   └── api_config.dart          # Base URL untuk koneksi ke PHP & Python backend
│   │   ├── models/                      # Model data (data structure/POJO)
│   │   │   ├── user.dart                # Model untuk User (id, username, email, dll)
│   │   │   └── translation.dart         # Model hasil translasi (input-output)
│   │   ├── services/                    # Layer untuk API service / business logic
│   │   │   ├── auth_service.dart        # Service login & register ke backend PHP
│   │   │   └── gemma_service.dart       # Service translasipenghubung ke backend Python
│   │   └── screens/                     #UI Screens / Halaman aplikasi
│   │       ├── landing_page.dart        # Halaman awal
│   │       ├── login_screen.dart        # Halaman login user
│   │       ├── register_screen.dart     # Halaman registrasi user baru
│   │       ├── dashboard_screen.dart    # Dashboard utama setelah login
│   │       ├── pl_to_pseudo_screen.dart # Halaman: Translate Code → Pseudocode
│   │       ├── pseudo_to_pl_screen.dart # Halaman: Translate Pseudocode → Code
│   │       └── simple_profile.dart      # Halaman profil user sederhana
│   └── pubspec.yaml                     # File konfigurasi dependencies Flutter
├── backend_php/                         # Backend PHP untuk autentikasi & Database
│   ├── database.php                     # File koneksi ke MySQL
│   ├── register.php                     # Endpoint registrasi user baru
│   ├── login.php                        # Endpoint login user
│   ├── user.php                         # Endpoint ambil data user
│   └── update_profile.php               # Endpoint update data profil user
├── backend_python/                      # Backend Python untuk translasi via Gemma model
│   ├── main.py                          # Entry point FastAPI untuk menjalankan server
│   ├── gemma.py                         # Endpoint translasi
│   ├── config.py                        # Konfigurasi model seperti Ollama URL dan nama model
│   ├── queue_manager.py                 # Manajemen antrean/limit request user
│   └── requirements.txt                 # Dependens seperti fastapi, httpx, uvicorn, dll
└── database/                            # Folder database SQL
    └── myapp_db.sql                     # Script SQL untuk membuat tabel users
```

Gambar 2. Struktur Folder dan File Sistem




Gambar 3. Diagram Alur Sistem Translate App

Alur komunikasi antar komponen dimulai dari pengguna yang berinteraksi melalui aplikasi Flutter. Setiap permintaan login atau pendaftaran dikirim ke Backend PHP melalui protokol HTTP menggunakan metode POST. Setelah proses autentikasi berhasil, backend menghasilkan token JWT yang disimpan secara lokal di aplikasi. Token ini digunakan kembali sebagai header authorization pada setiap permintaan berikutnya. Ketika pengguna melakukan penerjemahan, aplikasi mengirim teks masukan (baik pseudocode maupun kode program) ke Backend Python melalui endpoint FastAPI. Backend Python kemudian memproses permintaan tersebut, memeriksa kuota pengguna, mengakses model Gemma3:4b melalui API lokal Ollama, dan mengembalikan hasil terjemahan ke aplikasi Flutter. Dengan demikian, sistem memiliki alur komunikasi dua arah yang aman dan efisien antara tiga lapisan utama.

Pemilihan teknologi pada sistem ini didasarkan pada pertimbangan performa, kompatibilitas lintas platform, dan skalabilitas. Flutter dipilih karena mampu menghasilkan antarmuka pengguna modern dan reaktif di berbagai platform (Android dan iOS) menggunakan satu basis kode. PHP dipilih sebagai layanan autentikasi karena sifatnya yang ringan, mudah diintegrasikan dengan MySQL, serta memiliki dukungan luas terhadap pustaka keamanan seperti JWT. Sementara itu, Python dipilih karena dukungannya terhadap machine learning framework dan integrasi cepat dengan API LLM menggunakan FastAPI yang dikenal memiliki kinerja tinggi dan latensi rendah. Ketiga teknologi ini saling melengkapi untuk membentuk sistem yang tangguh, modular, dan dapat dikembangkan lebih lanjut

### 4.1. Backend PHP

Backend PHP menyediakan RESTful API dengan tiga endpoint utama (register, login, user profile) yang berkomunikasi melalui JSON dengan CORS header untuk integrasi Flutter. Koneksi database menggunakan PDO dengan prepared statements untuk mencegah SQL injection, sementara fungsi utilitas (validasi input, respons terstruktur, token management) dikemas dalam database.php sebagai lapisan abstraksi. Autentikasi menerapkan JWT berbasis HMAC-SHA256 dengan masa berlaku 30 hari, di mana setiap request terproteksi memerlukan token di header Authorization yang diverifikasi terhadap signature dan expiration time. Endpoint /register.php memvalidasi tujuh field (username minimal 3 karakter alfanumerik, email dengan format valid, password minimal 6 karakter di-hash bcrypt, serta nilai enum untuk gender dan role akademik) dengan pemeriksaan keunikan username/email sebelum insert, mengembalikan HTTP 409 jika duplikasi terdeteksi.

Endpoint /login.php memverifikasi kredensial dengan pesan error generik untuk mencegah user enumeration, sementara /user.php mendukung GET untuk mengambil profil dan PUT untuk update selektif dengan validasi ulang terhadap format dan keunikan data. Seluruh operasi mengembalikan respons JSON dengan HTTP status code standar (200, 400, 401, 409, 500) untuk memastikan error handling yang konsisten.

### 4.2. Backend Python

Layanan translasi dibangun menggunakan FastAPI dengan integrasi ke model Gemma3:4b melalui Ollama API. Sistem menyediakan dua endpoint yaitu /gemma/pseudo-to-code dan /gemma/code-to-pseudo untuk konversi kode dalam tiga bahasa (C++, Java, Python). Setiap request

dikirim ke Ollama dengan prompt yang dirancang untuk menghasilkan output murni tanpa komentar atau penjelasan tambahan. Jika model mendeteksi input tidak valid, sistem mengembalikan string "Error" yang di-handle oleh frontend untuk menampilkan pesan peringatan. Konfigurasi sistem menggunakan Pydantic Settings yang mendukung pengaturan melalui file .env, meliputi URL Ollama (http://localhost:11434/api/generate), nama model (gemma3:4b), batas quota harian (3 translasi per user per hari), dan maksimum concurrent requests (3 per user). Validasi bahasa pemrograman dilakukan pada setiap request untuk mencegah input tidak sah. API dilengkapi CORS middleware untuk integrasi cross-platform dan endpoint /health untuk monitoring status sistem.

Sistem quota management menggunakan dua mekanisme: (1) Daily quota tracking yang disimpan dalam file user_quota.json dengan struktur {"user_id": {"date": "YYYY-MM-DD", "count": n}} yang auto-reset setiap tengah malam, dan (2) Concurrent request queue management menggunakan in-memory dictionary dengan thread-safe locks. Sebelum memproses translasi, sistem memeriksa apakah user masih memiliki quota dan slot concurrent. Jika quota habis atau queue penuh, request ditolak dengan HTTP 429. Setelah translasi selesai, quota di-increment dan slot concurrent di-release melalui blok finally untuk memastikan cleanup bahkan jika terjadi error. Penanganan error mencakup timeout request ke Ollama (60 detik), validasi input (minimal 3 karakter), dan error handling untuk kondisi model crash atau response kosong. Penyimpanan quota menggunakan file JSON dengan thread-safe file I/O.

### 4.3. Database

Database MySQL menyimpan data pengguna dalam tabel users dengan sembilan kolom: id (primary key auto-increment), username dan email (unique constraint untuk mencegah duplikasi), fullname, password (hash bcrypt), gender dan role (ENUM: Laki-laki dan Perempuan, Dosen, Guru, Mahasiswa, dan Siswa untuk integritas data), birth_date (DATE), serta created_at (TIMESTAMP default CURRENT_TIMESTAMP). Desain ini mendukung kebutuhan autentikasi dan manajemen profil secara efisien tanpa redundansi. Pelacakan quota translasi harian menggunakan file JSON (user_quota.json) yang dikelola oleh queue_manager.py dengan struktur {"user_id": {"date": "YYYY-MM-DD", "count": n}}.

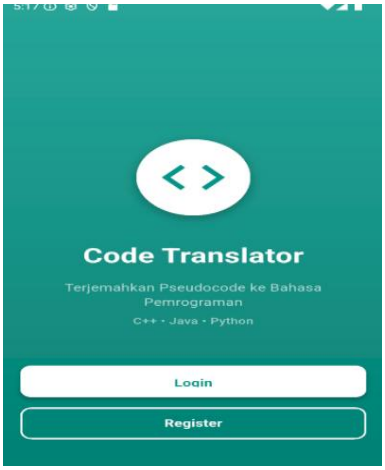
Pendekatan ini dipilih karena data quota bersifat temporary dan tidak memerlukan relasi kompleks. File diakses menggunakan thread-safe locks untuk mencegah race condition dalam concurrent environment. Sistem auto-reset quota setiap tengah malam berdasarkan perbandingan tanggal dan menjalankan cleanup otomatis untuk menghapus data lebih dari 7 hari guna menjaga ukuran file. Strategi ini menyeimbangkan kesederhanaan, performa I/O cepat, dan keandalan untuk skala penggunaan edukasi.


Gambar 4. Entity Relationship Diagram (ERD) Tabel Users

### 4.4. Arsitektur Aplikasi Flutter

Aplikasi mobile dikembangkan menggunakan Flutter dengan struktur modular berbasis separation of concerns: direktori models/ untuk entitas data (User, Translation) dengan metode serialisasi JSON; services/ untuk logika komunikasi eksternal (AuthService ke backend PHP, GemmaService ke backend Python); dan screens/ untuk komponen UI mandiri (LoginScreen, DashboardScreen, LandingPage). Konfigurasi sistem dipusatkan di api_config.dart yang menyimpan base URL kedua backend dan timeout request, sementara dependensi seperti http, shared_preferences, dan logger dideklarasikan di pubspec.yaml. Manajemen state menggunakan StatefulWidget tanpa library tambahan, dengan konversi respons jaringan ke objek model sebelum rendering UI dan navigasi berbasis named routes untuk transisi konsisten.

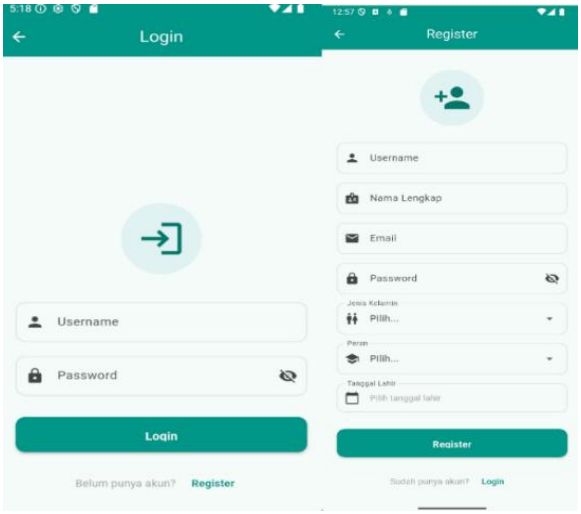
LandingPage berfungsi sebagai splash screen dan gerbang autentikasi yang dijalankan pertama kali saat aplikasi dibuka. Halaman ini memeriksa keberadaan token JWT di SharedPreferences menggunakan AuthService.getToken()—jika token valid ditemukan, sistem otomatis mengarahkan pengguna ke DashboardScreen tanpa login ulang (auto-login); jika tidak ada token atau token expired, ditampilkan halaman sambutan dengan tombol Login dan Register untuk pengguna baru. Mekanisme ini memastikan pengalaman seamless bagi pengguna returning sambil tetap aman dengan validasi token di setiap startup aplikasi.




Gambar 5. Landing Page Aplikasi

### 4.5. Fitur Autentikasi dan Dashboard

Halaman login dirancang sebagai titik masuk utama bagi pengguna terdaftar, dengan antarmuka yang bersih dan validasi real-time. Menggunakan TextFormField dengan form validation, sistem memastikan bahwa username dan password tidak kosong sebelum permintaan dikirim. Saat tombol login ditekan, aplikasi memanggil AuthService.login(), yang mengirim kredensial ke endpoint /login.php di backend PHP dalam format JSON. Jika autentikasi berhasil, respons berisi token JWT disimpan secara lokal menggunakan SharedPreferences—mekanisme penyimpanan persisten ringan yang umum digunakan di aplikasi Flutter untuk data sensitif berukuran kecil. Token ini kemudian digunakan di sesi berikutnya untuk menghindari login berulang. Selama proses berlangsung, antarmuka menampilkan indikator pemuatan (loading spinner), dan setiap kegagalan—baik karena kredensial salah maupun gangguan jaringan—ditampilkan melalui snackbar dengan pesan yang informatif namun tidak membocorkan detail teknis.




Gambar 6. Halaman Login dan Register Aplikasi

Halaman registrasi menyediakan formulir lengkap yang mengumpulkan tujuh atribut pengguna: username, nama lengkap, email, kata sandi, jenis kelamin, peran (Dosen, Guru, Mahasiswa, atau Siswa), serta tanggal lahir. Antarmuka memanfaatkan dropdown untuk memilih nilai kategorikal dan date picker untuk memastikan format tanggal konsisten (ISO 8601), sementara validasi formulir diterapkan secara ketat—baik di sisi klien maupun server.

Input seperti username dan email diverifikasi keunikannya oleh backend PHP, sedangkan kata sandi diwajibkan minimal enam karakter. Saat formulir dikirim, AuthService.register() mengirim data ke endpoint /register.php, dan jika berhasil, token JWT yang dikembalikan langsung disimpan ke SharedPreferences. Mirip dengan alur login, pengguna kemudian dialihkan otomatis ke dashboard, menciptakan pengalaman onboarding yang mulus tanpa perlu login manual setelah registrasi.

Dashboard berfungsi sebagai pusat navigasi utama setelah pengguna berhasil masuk, menampilkan antarmuka yang bersih dan berorientasi tindakan. Saat pertama kali dimuat, layar ini mengambil data profil pengguna—khususnya username—melalui AuthService.getUserData(), yang memanggil endpoint /user.php di backend PHP dengan token JWT yang tersimpan. Data ini digunakan untuk mempersonalisasi sapaan selamat datang, sementara indikator pemuatan ditampilkan selama permintaan berlangsung. Antarmuka utama terdiri dari dua kartu aksi bergradasi: satu untuk terjemahan pseudocode ke kode dan satu lagi untuk arah sebaliknya, masing-masing dikaitkan ke rute yang sesuai. Informasi tambahan—seperti daftar bahasa yang didukung (C++, Java, Python)—ditampilkan dalam kartu notifikasi untuk memandu pengguna. Di app bar, tombol profil dan logout disediakan; yang terakhir memicu dialog konfirmasi sebelum memanggil AuthService.logout(), yang menghapus token dari SharedPreferences dan mengarahkan pengguna kembali ke halaman selamat datang, menutup sesi dengan aman


Gambar 7. Halaman Dashboard Aplikasi

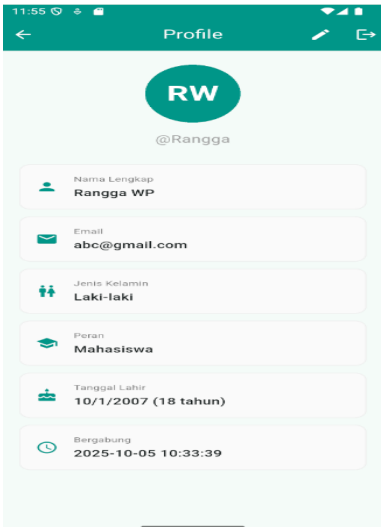
### 4.6. Fitur Manajemen Profil

Fitur manajemen profil memungkinkan pengguna melihat dan memperbarui informasi pribadi mereka melalui antarmuka yang beralih dinamis antara mode tampil (view mode) dan mode edit (edit mode). Saat pertama kali dibuka, layar menampilkan data profil lengkap—termasuk nama lengkap, email, jenis kelamin, peran (Dosen/Guru/Mahasiswa/Siswa), tanggal lahir, dan waktu pendaftaran—dalam bentuk kartu informasi yang rapi, dilengkapi avatar berisi inisial nama pengguna.

Pengguna dapat beralih ke mode edit dengan menekan ikon pensil di app bar, yang mengubah tampilan menjadi formulir interaktif dengan text fields untuk nama dan email, dropdown untuk jenis kelamin dan peran, serta date picker untuk tanggal lahir. Semua perubahan divalidasi secara lokal (format email dan kelengkapan nama), lalu dikirim ke backend PHP melalui AuthService.updateProfile(), yang memperbarui data di basis data setelah memverifikasi keunikan email dan integritas nilai kategorikal. Jika pembaruan berhasil, antarmuka otomatis kembali ke

mode tampil dan menampilkan notifikasi sukses; jika gagal, pengguna diberi umpan balik melalui snackbar.

Fitur ini dirancang untuk menjaga konsistensi data sekaligus memberikan kontrol penuh kepada pengguna atas informasi profil mereka, tanpa mengizinkan perubahan pada username—yang bersifat permanen setelah registrasi—sebagai bagian dari strategi integritas identitas sistem.



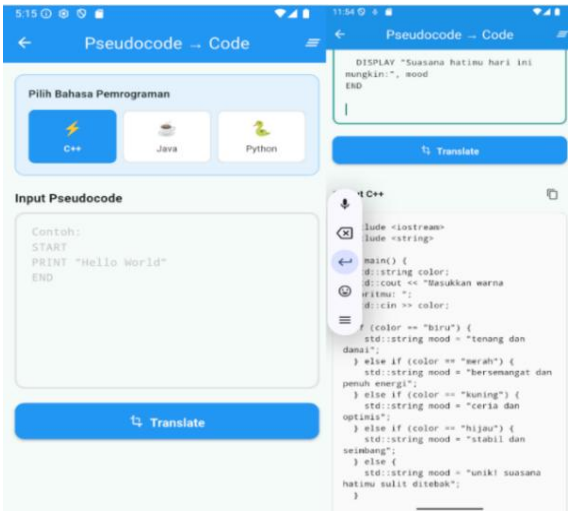

Gambar 8. Halaman Profil Pengguna

### 4.7. Fitur Translation

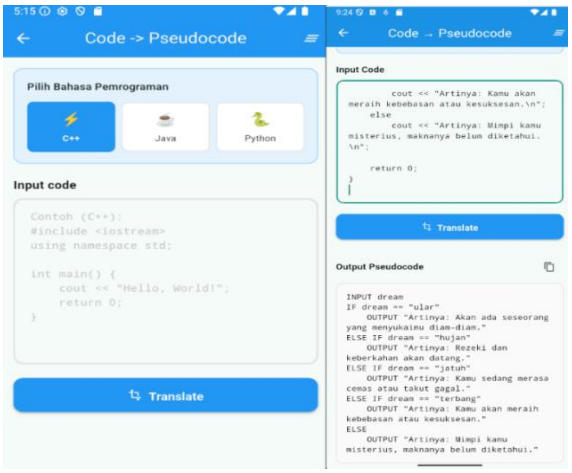
Sistem menyediakan antarmuka terjemahan dua arah yang intuitif untuk konversi antara pseudocode dan tiga bahasa pemrograman (C++, Java, Python). Pada mode Pseudocode ke Code, pengguna memilih bahasa target melalui chip selector berikon, memasukkan pseudocode pada text field multiline berfont monospace, lalu menekan tombol Translate setelah validasi lokal (minimal 3 karakter).

Proses translasi memanggil GemmaService.pseudoToCode() dengan user ID dari token JWT, menampilkan loading indicator selama pemrosesan, dan menghasilkan output pada area terpisah yang dilengkapi fungsi copy-to-clipboard. Jika model mengembalikan string "Error", sistem menampilkan pesan peringatan kontekstual.

Mode Code ke Pseudocode bekerja serupa dengan arah konversi terbalik melalui endpoint /gemma/code-to-pseudo, memvalidasi input kode sebelum dikirim ke backend Python. Kedua antarmuka menerapkan prinsip Material Design dengan skema warna biru konsisten, error handling yang informatif melalui snackbar, serta mekanisme quota management yang membatasi tiga translasi per user per hari untuk mencegah penyalahgunaan layanan.




Gambar 9. Halaman Penerjemah Psudocode ke Bahasa Pemrograman




Gambar 10. Halaman Penerjemah Bahasa Pemrograman ke Pseudocode

### 4.8. Pengujian

Pengujian sistem dilakukan menggunakan metode Black Box Testing, yaitu pendekatan pengujian yang berfokus pada fungsi sistem tanpa melihat struktur internal kode program. Metode ini dipilih karena sesuai untuk menguji kesesuaian antara input dan output pada setiap fitur aplikasi yang telah dikembangkan. Setiap komponen diuji berdasarkan spesifikasi fungsional, dengan tujuan memastikan bahwa sistem berjalan sesuai kebutuhan pengguna. Pengujian dilakukan terhadap seluruh lapisan sistem, meliputi Frontend (Flutter), Backend PHP, Backend Python, dan Database. Hasil pengujian dicatat dalam tabel berikut:

Tabel 1. Hasil BlackBox Testing

| No | Fitur yang Diuji | Hasil yang Diharapkan | Hasil Pengujian |
|---|---|---|---|
| 1 | Login pengguna | Pengguna dapat masuk dengan username dan password yang valid | Berhasil |
| 2 | Registrasi pengguna baru | Akun baru berhasil dibuat dan token JWT dikembalikan | Berhasil |
| 3 | Update profil pengguna | Data profil (nama, email, gender, role) berhasil diperbarui | Berhasil |
| 4 | Translasi pseudocode ke bahasa pemrograman | Sistem menampilkan hasil kode (C++, Java, atau Python) dari input pseudocode | Berhasil |
| 5 | Translasi bahasa pemrograman ke pseudocode | Sistem menampilkan hasil pseudocode yang logis dari input kode | Berhasil |
| 6 | Validasi kuota harian translasi | Setelah 3 kali translasi, sistem menolak permintaan berikutnya dengan pesan batas kuota | Berhasil |
| 7 | Navigasi antarmuka | Setiap tombol dan route aplikasi berfungsi dengan benar tanpa error | Berhasil |
| 8 | Logout pengguna | Token dihapus dari penyimpanan lokal dan pengguna diarahkan ke halaman awal | Berhasil |
| 9 | Penyimpanan data pengguna | Informasi user tersimpan di database MySQL tanpa duplikasi email/username | Berhasil |
| 10 | Koneksi ke backend Python (Gemma API) | Permintaan translasi berhasil diproses dan hasil dikembalikan melalui FastAPI | Berhasil |

### 4.9. Pemeliharaan

Hasil implementasi sistem Translate App menunjukkan bahwa aplikasi telah berfungsi sesuai dengan tujuan awal, yaitu membantu pengguna dalam menerjemahkan pseudocode ke bahasa pemrograman dan sebaliknya secara lebih efektif melalui perangkat mobile. Untuk menjaga kinerja, stabilitas, dan keandalan sistem dalam jangka panjang, dilakukan proses pemeliharaan secara berkala setelah tahap implementasi dan pengujian selesai.Pemeliharaan sistem meliputi beberapa jenis kegiatan, di antaranya pemeliharaan korektif, yaitu perbaikan terhadap kesalahan atau bug yang ditemukan setelah sistem digunakan oleh pengguna, seperti error pada koneksi API atau kegagalan validasi input; pemeliharaan adaptif, yaitu penyesuaian sistem terhadap perubahan lingkungan teknologi seperti pembaruan versi Flutter, FastAPI, atau pustaka dependensi lainnya; serta pemeliharaan preventif, yang bertujuan mencegah potensi gangguan melalui optimalisasi performa backend, pembaruan keamanan token autentikasi, dan perawatan basis data pengguna. Dengan dilakukannya pemeliharaan secara berkelanjutan, sistem diharapkan tetap stabil, aman, dan mampu menyesuaikan diri dengan perkembangan kebutuhan pengguna maupun teknologi di masa mendatang.

## 5. KESIMPULAN DAN SARAN

Sistem Translate App berhasil mengintegrasikan teknologi mobile, web backend, dan large language model menjadi sebuah solusi edukasi yang utuh untuk konversi antara pseudocode dan bahasa pemrograman. Arsitektur tiga lapis—Flutter di sisi klien, PHP untuk autentikasi dan manajemen pengguna, serta Python/FastAPI sebagai layanan terjemahan berbasis model Gemma 3:4B—memungkinkan pemisahan tanggung jawab yang jelas, skalabilitas modul, dan keandalan sistem secara keseluruhan.

Fitur autentikasi berbasis JWT, validasi data ketat, kuota harian, serta antarmuka pengguna yang intuitif memastikan pengalaman yang aman, responsif, dan mudah digunakan, khususnya bagi dosen, guru, mahasiswa, dan siswa yang membutuhkan alat bantu dalam memahami logika pemrograman.Implementasi terjemahan menunjukkan bahwa model bahasa besar dapat diarahkan secara efektif untuk tugas teknis spesifik melalui prompt engineering yang ketat dan pembatasan output. Sistem mampu menghasilkan kode yang valid dalam C++, Java, dan Python dari deskripsi algoritmik, maupun sebaliknya, dengan mempertahankan struktur logika inti. Meskipun keterbatasan kuota dan ketergantungan pada model eksternal (Ollama) menjadi pertimbangan operasional, pendekatan ini terbukti layak untuk skenario penggunaan edukasi skala kecil hingga menengah, di mana akurasi kontekstual dan kemudahan akses lebih diutamakan daripada throughput tinggi. Pengembangan lebih lanjut, sistem dapat ditingkatkan dengan mengganti penyimpanan kuota berbasis file JSON ke basis data relasional atau Redis guna mendukung skalabilitas multi-pengguna secara robust. Selain itu, integrasi model yang lebih ringan atau fine-tuning khusus untuk domain pseudocode dapat meningkatkan akurasi dan kecepatan respons. Fitur tambahan seperti riwayat terjemahan, ekspor kode ke file, atau mode offline parsial juga berpotensi memperkaya pengalaman pengguna tanpa mengorbankan prinsip desain yang sudah ada.

## DAFTAR PUSTAKA

[1] Irawati, Gusnawaty, Tadjuddin Maknun, Muhammad Hasyim, and Asriani Abbas, “Analisis Sirkumstan dalam Teks Terjemahan dengan Pendekatan Sistemic Functional Linguistics (SFL),” *J. Onoma Pendidikan, Bahasa, dan Sastra*, vol. 8, no. 2, pp. 436–455, 2022, doi: 10.30605/onoma.v8i2.1771.

[2] S. Utaminingsih and D. Andriani, "Analisis Kesalahan Linguitik Hasil Terjemahan Google Translate Dari Teks Bahasa Inggris Ke Dalam Bahasa Indonesia," *J. Eduscience*, vol. 9, no. 3, pp. 838–849, 2022, doi: 10.36987/jes.v9i3.3386.

[3] I. Y. Panessai, D. Iskandar, Afriani, Pratiwi, and E. Effendi, "Analisis Teknik Penerjemahan pada Abstrak Jurnal IJAI 6(1)," *J. Humanit. Soc. Sci.*, vol. 3, no. 1, pp. 9–22, 2021, doi: 10.36079/lamintang.jhass-0301.187.

[4] Vina Alliana and Yahfizham Yahfizham, "Analisis Penerapan Dan Fungsi Algoritma Pemrograman Pada Komputer," *J. Ris. Rumpun Mat. Dan Ilmu Pengetah. Alam*, vol. 3, no. 1, pp. 207–218, 2024, doi: 10.55606/jurrimipa.v3i1.2401.

[5] A. Hoyyi and R. Rahmawati, "Prediksi Nilai Aset Menggunakan Model Geometric Brownian Motion Dan Model Variance Gamma," *J. Gaussian*, vol. 13, no. 2, pp. 415–420, 2024, doi: 10.14710/j.gauss.13.2.415-420.

[6] K. Fajri, "Pemodelan Tingkat Inflasi di Sumatra Menggunakan Model Gamma dan Binomial Negatif," *J. Ekon. Bisnis dan Manaj.*, vol. 3, no. 4, pp. 338–349, 2024, doi: 10.58192/ebismen.v3i4.2785.

[7] Ahmad Al-hafiz Sagala and Yahfizham Yahfizham, "Analisis Pengenalan Konsep Algoritma Pemrograman Matematika Pada Kehidupan Sehari Hari," *Morfol. J. Ilmu Pendidikan, Bahasa, Sastra dan Budaya*, vol. 2, no. 1, pp. 01–16, 2024, doi: 10.61132/morfologi.v2i1.267.

[8] S. T. Ndapa Lawa, C. P. Ate, and V. P. Feka, "Penggunaan Google Translate Sebagai Alternatif Media Penerjemah Pada Abstrak Jurnal Mahasiswa," *HINEF J. Rumpun Ilmu Pendidik.*, vol. 1, no. 1, pp. 86–93, 2022, doi: 10.37792/hinef.v1i1.431.

[9] S. Prasetyaningsih and W. P. Ramadhani, "Analisa User Experience pada TFME Interactive Learning Media Menggunakan User Experience Questionnaire," *J. Integr.*, vol. 13, no. 2, pp. 147–157, 2021, doi: 10.30871/ji.v13i2.3180.

[10] A. Sonita and N. Praja, "Anisya Sonita, Nofriansyah Praja," *J. Pseudocode*, vol. 9, no. 1, 2022.

[11] R. Adityawan, A. Gunawan, and G. L. Ginting, "Analisis Kepuasan Penggunaan Aplikasi Pospay Menerapkan Metode TAM," *J. Kaji. Ilm. Teknol. Inf. dan Komput.*, vol. 1, no. 2, pp. 57–62, 2024, doi: 10.62866/jutik.v1i2.85.

[12] T. Indarti, U. Z. Fanani, R. Nasrullah, H. Septiana, and N. Afdholy, "Innovative artificial intelligence (AI) technology learning support for Indonesian language teachers at a junior high school in Tuban," *Transform. J. Pengabdi. Masy.*, vol. 20, no. 2, pp. 438–452, 2024, doi: 10.20414/transformasi.v20i2.11317.

[13] N. Puji *et al.*, "Transformation of Indonesian Language in Social Media Using AI Expert Systems and Machine Learning," vol. 3, no. 2, pp. 130–139, 2025.

[14] V. A. Lestari, R. Arianto, B. S. Anindito, Y. K. Tarmita, L. Amalia, and O. D. Triswidrananta, "Aplikasi Pembelajaran Rekonstruksi Algoritma Pseudocode dengan Pendekatan Element Fill-In-Blank Problems di Pemrograman Java," *J. Tek. Ilmu dan Apl.*, vol. 3, no. 2, pp. 154–162, 2022.

[15] M. Yunus and Y. Anwar, "Aplikasi Penerjemah Bahasa Isyarat Indonesia Ke Dalam Huruf Abjad," *J. Sintaks Log.*, vol. 2, no. 1, pp. 257–262, 2022, doi: 10.31850/jsilog.v2i1.1726.

[16] H. Fatah *et al.*, "Rancang bangun aplikasi kamus bahasa indonesia –bawean menggunakan algoritma," *J. Rekursif*, vol. 3, no. 1, p. 41, 2022.