NIT3004 – IT Capstone Project 2

Semester 2, Block 4 (H2B4) – Group 14

80% Completion Report for TranslateAI

Student Name and ID: Nachiket Patel (s8082887)

Date of Submission: 31st October 2025

Lecturer: Dr Gongqi (Joseph) Lin

Unit Convenor: Csaba Veres

Table of Contents:

Executive Summary

TranslateAI is a multilingual translation platform that utilises artificial intelligence to translate text, voice, image, and document inputs within countless seconds. It has been created and developed for the IT Capstone Project at Victoria University to demonstrate full-stack development skills. The application merges a Next.js frontend with a FastAPI backend, which is integrated with Supabase for authentication and database management. Cloudflare is used to provide a secure deployment environment within SSL encryption and domain-level protection.

This report presents TranslateAI's current progress, which now stands at approximately eighty per cent completion. The project code is completely functional with core and enhanced functionalities including text-to-text, audio-to-text, and document-to text working as intended. The remaining tasks include creating a technical implementation report, a user manual, a poster, and preparation for the oral presentation.

1.0 Introduction

TranslateAI is designed to optimise cross-language communication by offering real-time translation through a web application. The platform uses OpenAI's large language model API to process and translate inputs in multiple languages. User can enter text, upload documents, or speak directly into the system to receive instant text-based translations in a chosen language. The application is aimed at tourists, and organisations seeking a free, privacy-conscious alternative to commercial translation services.

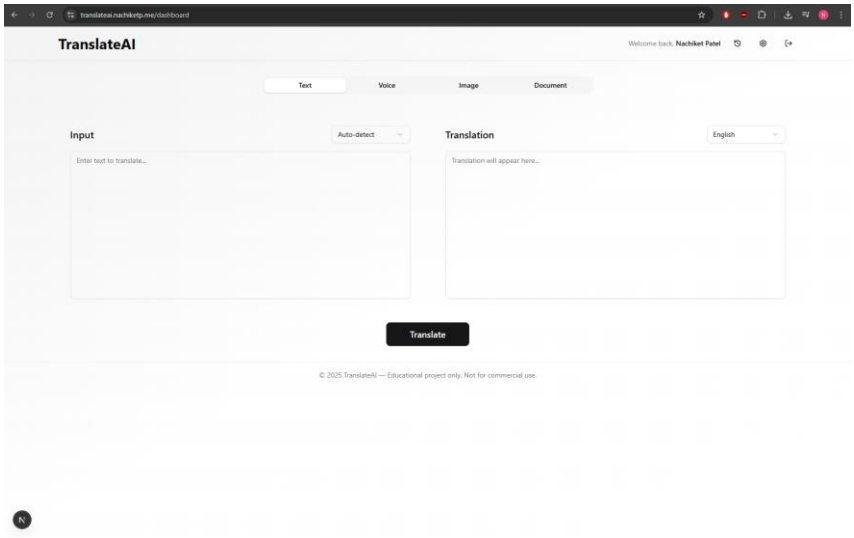

Figure 1. TranslateAI homepage showing the two-panel translation layout with input and output areas.

1.1 Background and Context

Traditionally, language translation has relied on private software which restricts data privacy and requires subscription fees. TranslateAI was conceived as a free, secure, and educational alternative built entirely with open-source technologies. The project also serves as a proof of concept for AI-assisted translation deployed within a cloud security framework.

Development began during the IT Capstone 1 unit, where the initial proposal and timeline were drafted. Due to limited to zero group collaboration following IT Capstone 1, the project officially transitioned to an independent development model on 28th October 2025 under my leadership. All coding, testing, and documentation have been produced independently, ensuring a consistent vision and quality standard throughout.

1. Introduction

Context

In today's globalised digital economy, language remains one of the most persistent barriers to equitable access, collaboration, and innovation. With over 7,000 languages spoken worldwide, the digital divide is exacerbated by the underrepresentation of low-resource languages in AI systems (Ta & Lee, 2023). As cross-border communication becomes essential in education, business, and public services, the need for intelligent, real-time translation tools has grown exponentially.

Purpose

This project proposes the development of a multimodal translation tool that leverages the OpenAI ChatGPT API (OpenAI, 2025) to deliver context-aware, real-time translations across text, voice, image, and document inputs. Unlike traditional word-for-word converters, this tool will preserve tone, intent, and nuance—addressing the limitations of current mainstream solutions. It will also support user-specific translation history and playback features, enhancing both accessibility and user engagement.

Scope

The application will be built using Python (FastAPI) (FastAPI, 2023) for backend services and JavaScript (React or Vue) for the frontend. It will integrate:

- ChatGPT for natural language translation (OpenAI, 2025)
- Google Speech-to-Text for voice input (Google, 2023)
- Tesseract OCR for image-based text extraction (tesseract-ocr, 2019)
- gTTS or Amazon Polly for speech synthesis (AWS, 2025; gTTS, 2018)

The system will support secure user authentication, translation history management, and exportable outputs (text/audio). It will be designed for scalability, accessibility, and extensibility.

Figure 2. Extract from initial TranslateAI project proposal

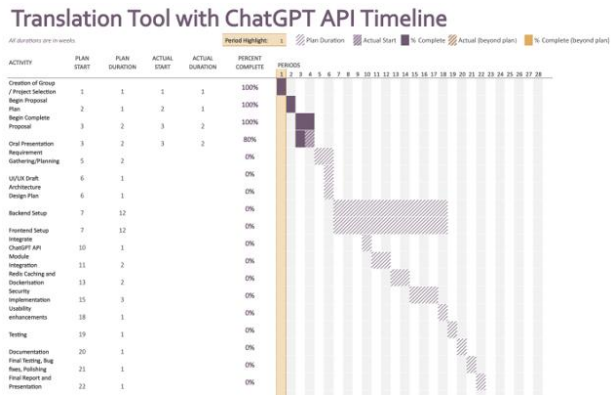

Figure 3. Timeline developed in IT Capstone 1.

1.2 Project Objectives

The primary aim of TranslateAI is to build a fully functional multilingual translation platform that operated securely within a web environment. It involves specific objectives such as:

- To develop a modern, responsive web interface for real-time translation of text, image, voice, and documents.
- To implement secure user authentication through Supabase with email and social login.
- To enable cloud-based data storage for translation history while preserving use privacy through row-level security rules.
- To deploy the application on Cloudflare to ensure encrypted connections and protection from DDoS attacks.
- To portray technical competence in full-stack web development and AI integration

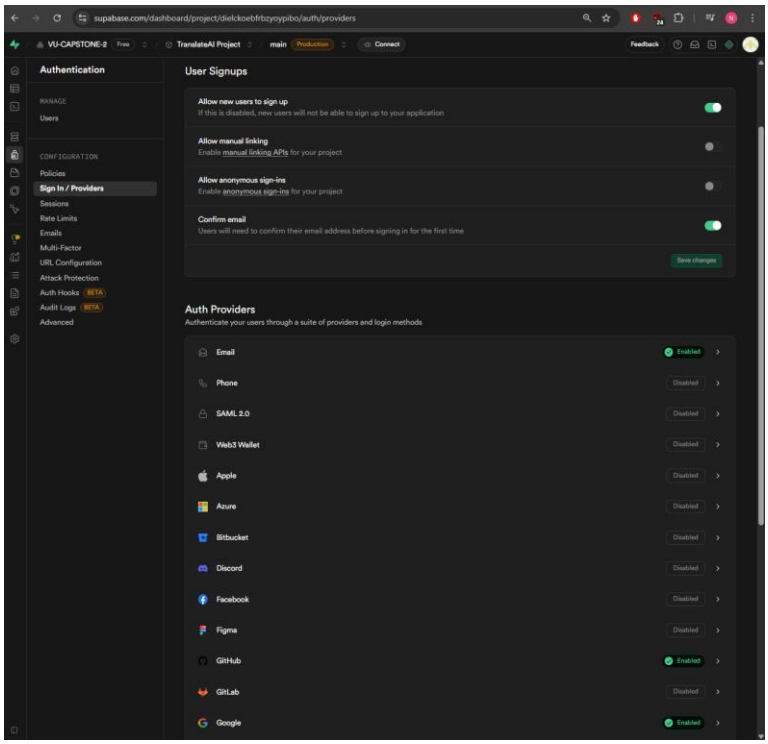

Figure 4. Supabase authentication console showing configured email and social logins

.

1.3 Significance of the Project

TranslateAI holds academic and social significance. From an academic perspective, it demonstrates how AI models can be embedded into secure applications to achieve real-world utility. For users, it represents a privacy-respecting alternative to normalised translation services that often collect and monetise user data for profits. The platform's open-architecture design makes it adaptable for educational institutions and non-profit organisations seeking affordable language tools.

Furthermore, TranslateAI aligns with Victoria University's emphasis on innovation and ethical technology development within the Cyber Security discipline. By building a full functional AI-powered translator from scratch, the project demonstrates skills in secure competencies for graduate cyber professionals.

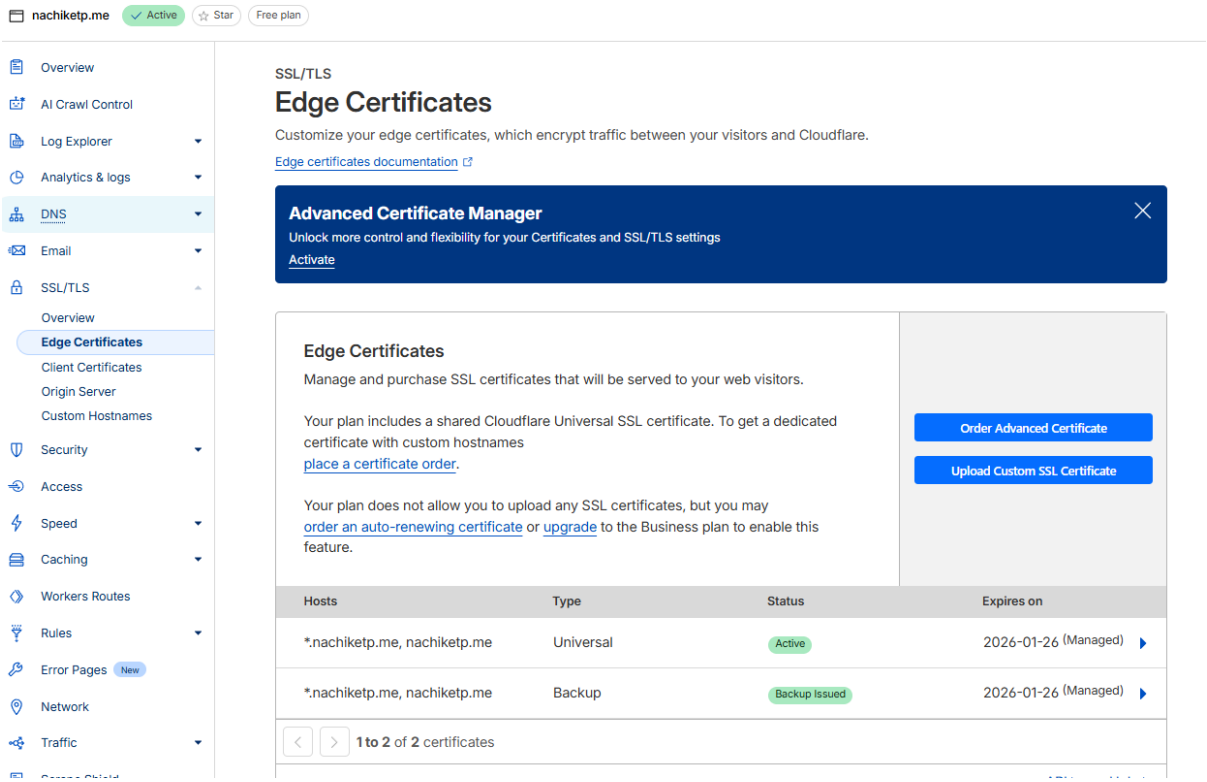

Figure 5. Cloudflare configuration displaying SSL certificate status

1.4 Current Progress Status

At the time of writing this report, TranslateAI is eighty per cent complete. The core features include a translation interface, an authentication system, and a backend API. All 3 core features are operational and have been tested across multiple devices. A functioning deployment pipeline via Cloudflare Tunnel ensures secure access to the local backend server.

The frontend includes all user interaction modules: registration, login, translation, history view and settings. The backend provides a structured API that handles translation requests and interacts with Supabase for user data storage. The testing phase has confirmed that translations are accurate and delivered within a short latency period of approximately 1-2 seconds for text inputs.

The project code is one hundred percent complete however, there are remaining tasks regarding the documentation including the Technical Implementation Document, User Manual, Poster, along with the presentation.

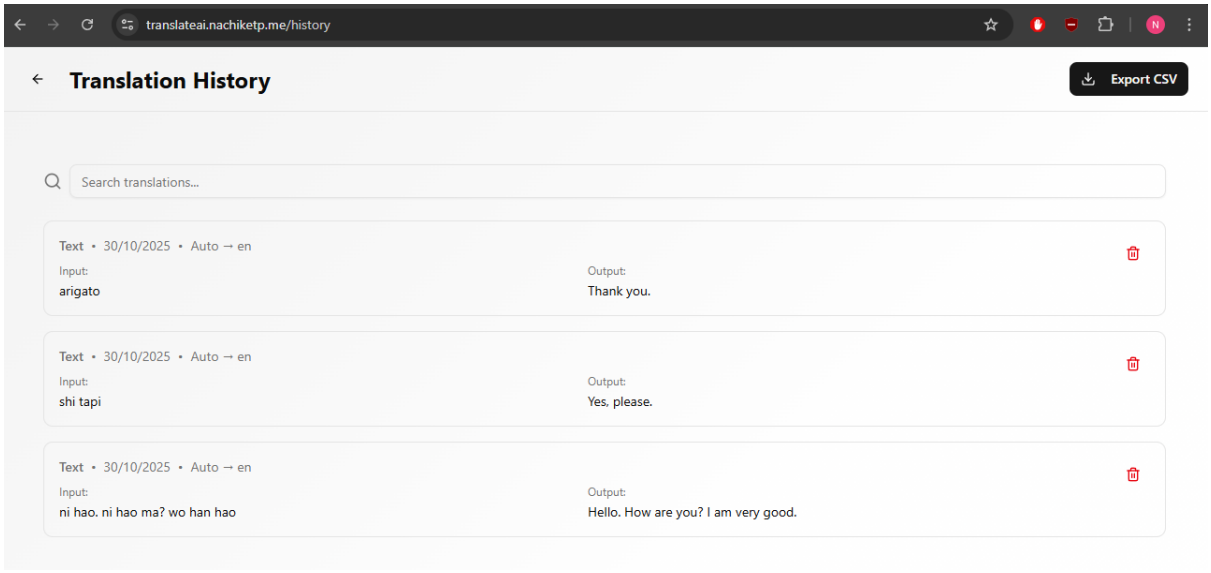

Figure 6. TranslateAI dashboard page displaying stored translation history records.

2.0 System Development and Progress Overview

TranslateAI was built using a full architecture combining Next.js for the frontend, FastAPI for the backend, Supabase for authentication and database management, and Cloudflare for secure deployment. The system was designed to support development in multiple areas, ensuring that each core component as translation, authentication, and storage could operate independently and be debugged efficiently.

The entire system was developed over an intensive three-day period. Each day focused on specific milestones, including frontend design, backend integration, and deployment setup.

2.1 Development Timeline Summary

Day 1 – Backend Setup and API Integration

The initial day focused on setting up the FastAPI server, connecting it with OpenAI's API and testing responses. The backend was structured modular services:

- `translator.py` handled text and document translation through the OpenAI API
- `audio.py` implemented voice translation by converting audio input into text using speech recognition and then re-synthesising the translated output.
- `database.py` connected the backend to Supabase via RESTful endpoints to store translation history.
- `schemas/translate/py` defined structured response models for consistent API returns.

All environment variables, including the OpenAI API key and Supabase credentials, were configured via a `.env` file to ensure there was a secure separation of sensitive data.

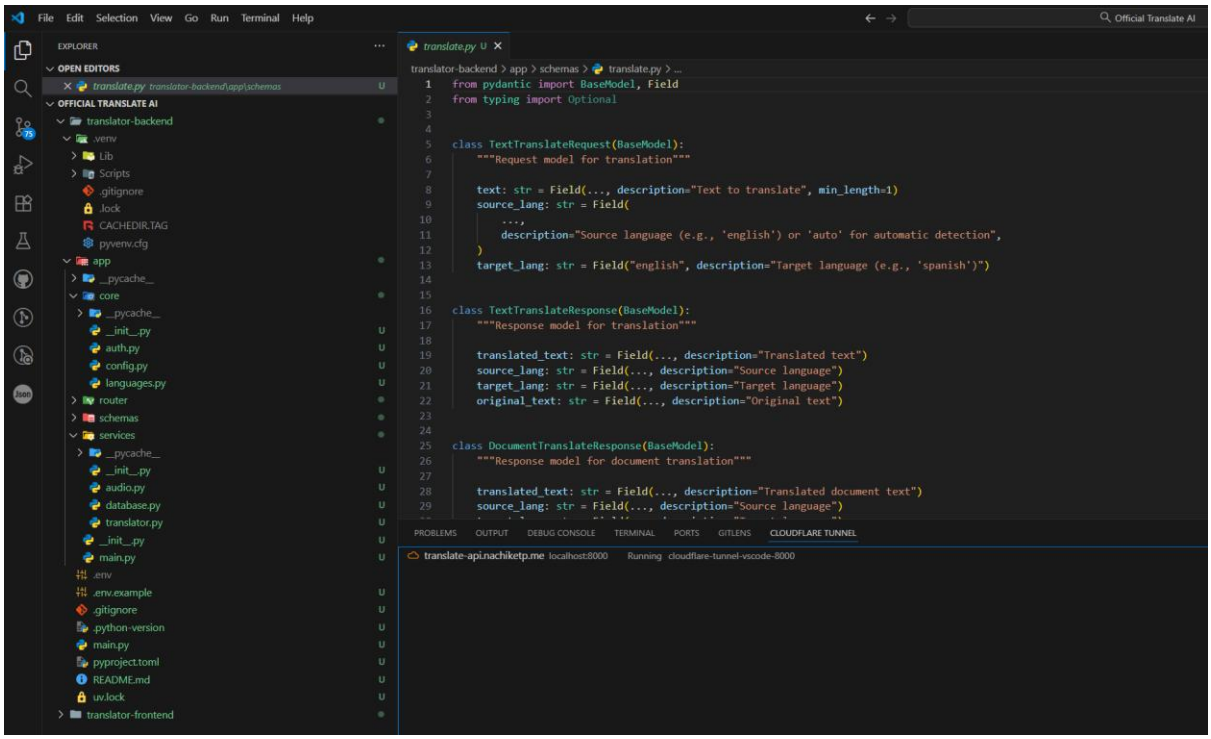

Figure 7. Folder hierarchy of backend files including translator.py, database.py, and FastAPI main application

Day 2 – Frontend Implementation and Authentication

The second day focused on building the user interface using Next.js with TailwindCSS and integrating authentication using Supabase. Features added include:

- Registration and login pages for both email/password and social logins (Google and GitHub).
- Password strength validation during registration.
- “Forgot Password” and “Reset Password” flows linked to Brevo SMTP email verification
- Translation dashboard with input/output texts areas, history panel, and real-time feedback

UI components were developed using ShadCN and TweakCN for consistency and accessibility. React Query was utilised for efficient backend communication and real-time updates.

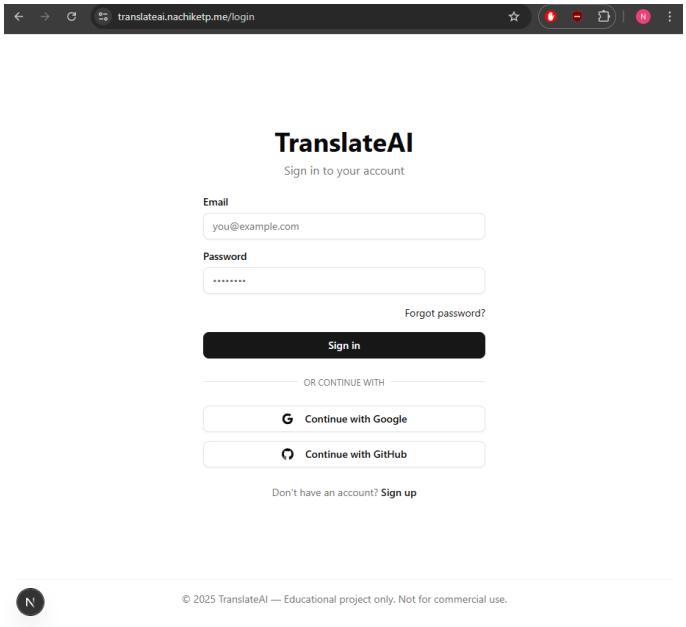

Figure 8. Login and registration pages integrated with Supabase authentication.

Day 3 – Deployment and Security Hardening

The final day involved deploying the project through Cloudflare Tunnel. Namecheap DNS records were transferred and updated in Cloudflare, enabling HTTPS access via `translateai.nachiketp.me`. Cloudflare SSL certificates were configured for automatic encryption and DDoS protection. A Cloudflare Tunnel was created to securely expose the frontend service from the local environment to the internet without public port forwarding.

Security implementations included:

- SSL-encrypted data transmission.
- Hidden local ports through Cloudflare Tunnel.
- Supabase row-level security enforcing user isolation.
- OAuth 2.0 login mechanisms via trusted providers

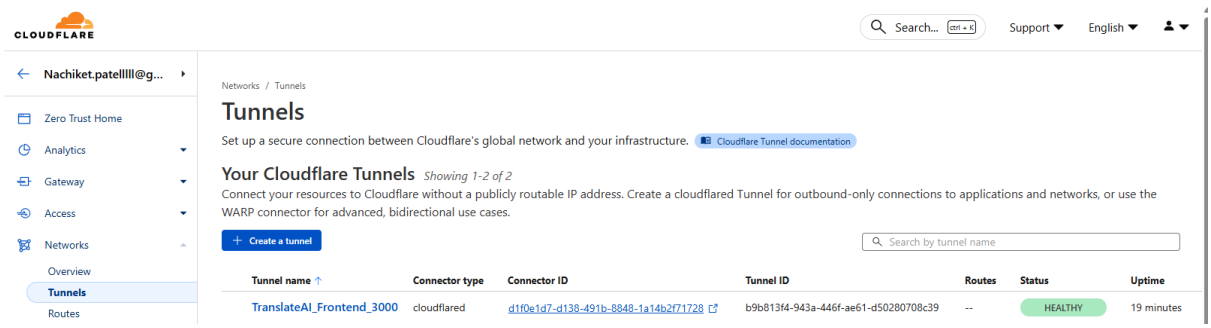

Figure 9. Cloudflare dashboard showing active tunnel connected to the local frontend.

2.2 System Architecture

TranslateAI follows a client-server model with clear separation between the presentation layer (frontend), logic layer (backend), and storage layer (database).

Frontend (Next.js + TailwindCSS):

- User interface built with Next.js 16, TailwindCSS, ShadCN, and React Query. Handles user input, authentication, and real-time translation display.
- Provides the user interface, enabling users to log in, submit text or files, and view translations

Backend (FastAPI + LangChain):

- FastAPI server handles API routing, LangChain handles the OpenAI integration
- Uvicorn runs the API server locally.
- Py2PDF handles the document parsing, and translation processing.
- Communicates securely with Supabase via HTTPS and WebSocket.
- Processes translation requests using OpenAI APIs and manages session history.

Database (Supabase):

- Handles user authentication, translation records
- Secure data storage via row-level security.

Cloudflare Deployment:

- Manages DNS routing and SSL encryption
- Tunnel connections for secure external access (no IP leaks or public port forwarding)
- Provides DDoS protection through Cloudflare Tunnel and subdomain configuration.

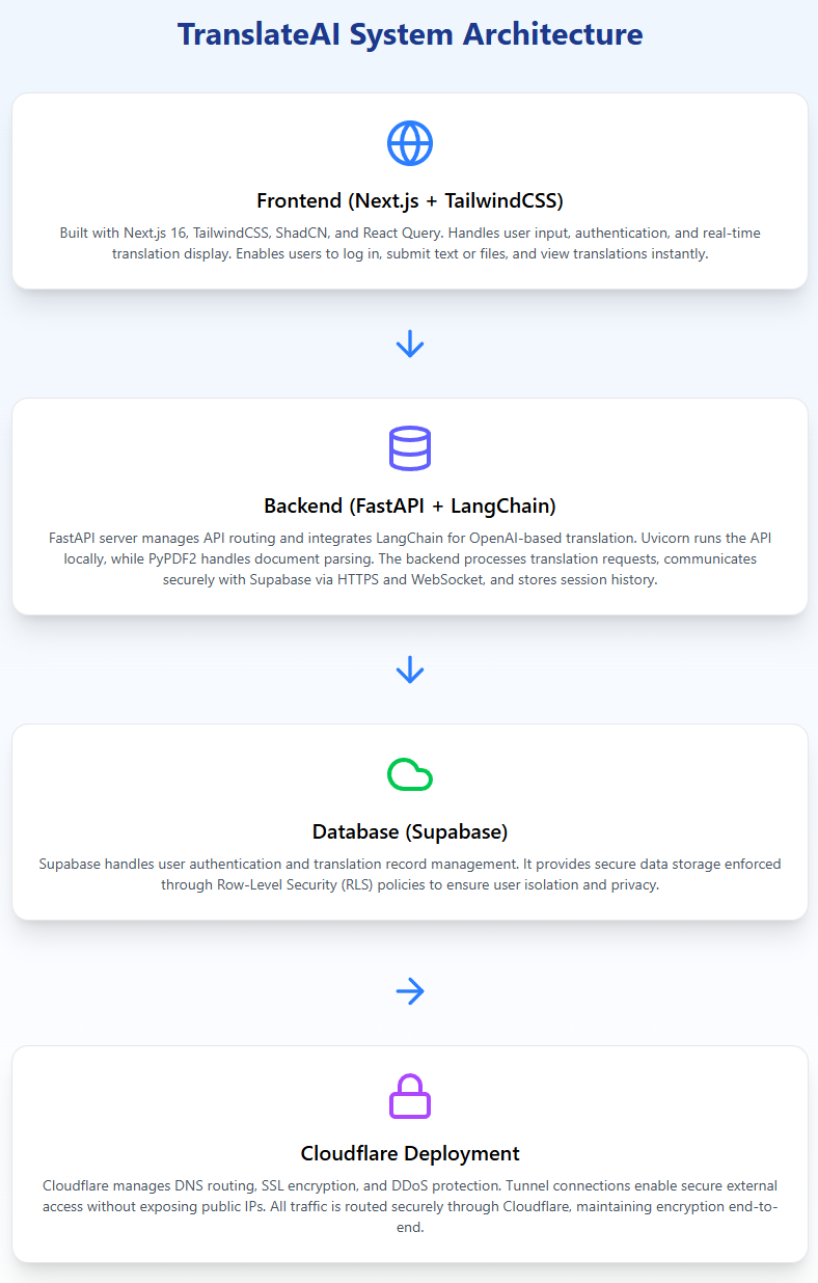

Figure 10. TranslateAI system architecture illustrating the connection between the frontend, backend, Supabase, and Cloudflare.

3.0 Demonstration of Completed Work

This section presents the current features of TranslateAI that have been fully implemented and tested. Each feature was validated through manual testing on both mobile and desktop browsers.

3.1 User Interface Overview

The interface utilises a minimalist design, prioritising clarity and accessibility. The landing page includes input and output boxes for translation, a language selector and voice input buttons. Users can access navigation options for History, Settings, and Logout through the menu, located in the top-right of the page.

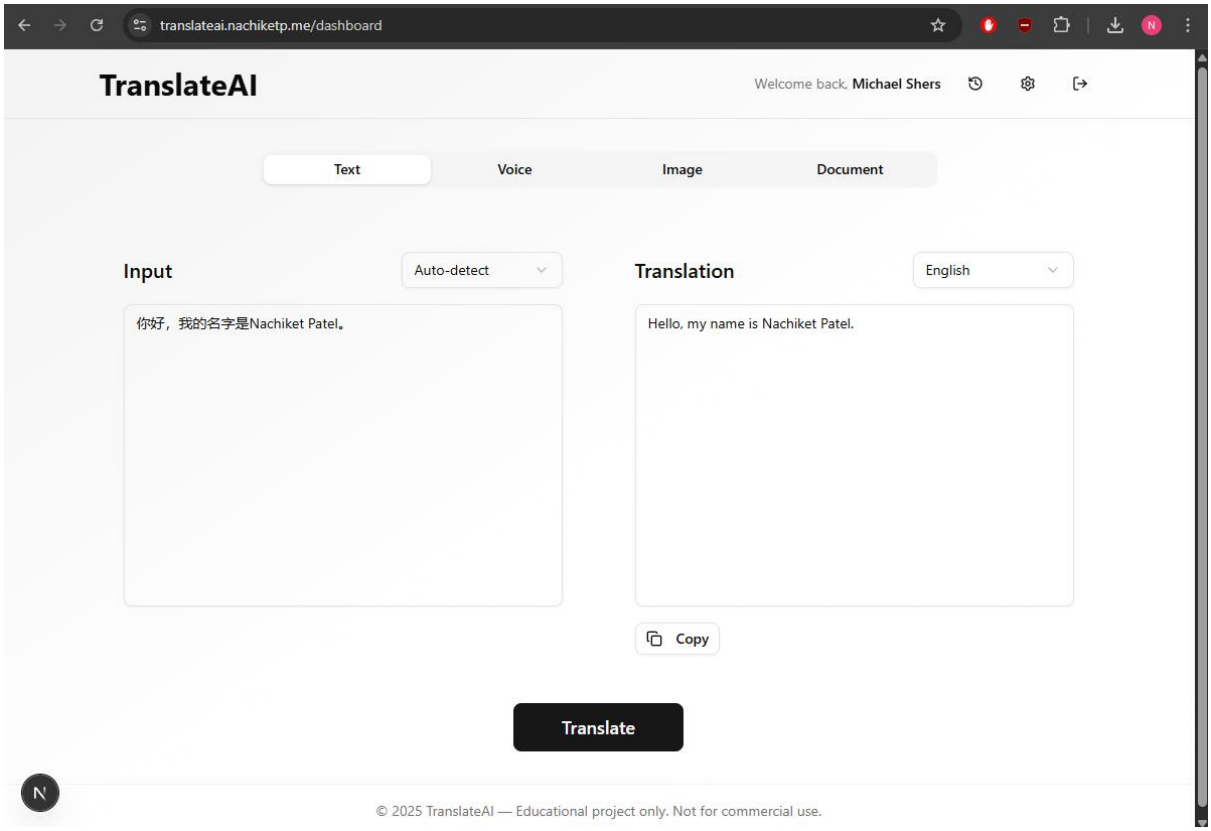

Figure 11. Main translation interface displaying input and translated output sections.

3.2 Core Functionalities

- **Registration / Log in:** Users must register or log in to access the platform. This feature has been added so translation history can be saved automatically and accessed at any time within a certain account.

- **Text-to-Text Translations:** Users can input text in any supported language and receive translations in another.
- **Real-time Voice-to-Text:** Enables spoken input through a microphone, translating, and outputting as text.
- **Audio-to-Text Translations:** Users can upload an audio file and receive translations in a text output.
- **Image-to-Text Translation**: Allows upload of .jpg, and .png files for transcribing and text-based translation output.
- **Document-to-Text Translation:** Allows upload of PDF or DOCX files for full-text translation.
- **Translation History:** Automatically stores each translation with date, time, source, and target language metadata.
- **Account Management:** Users can update their email, reset passwords, and manage credentials via the settings page.

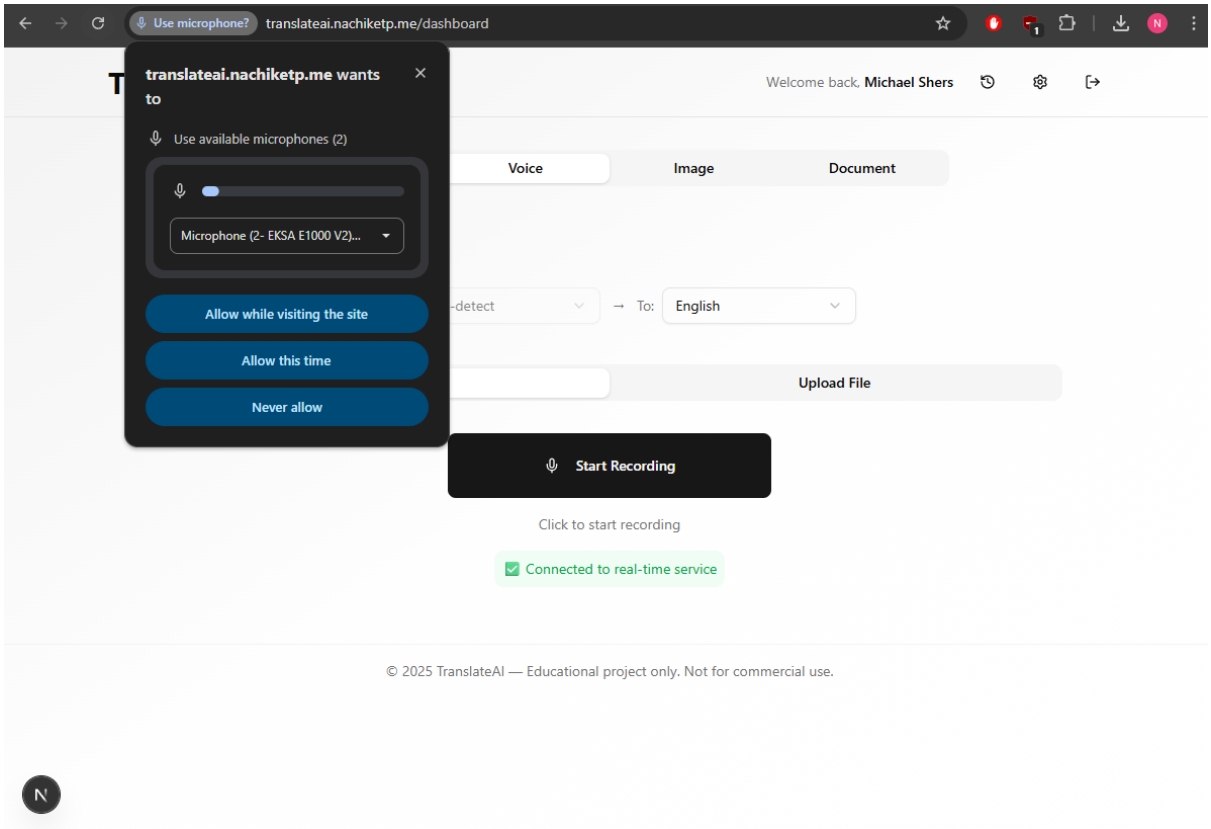

Figure 12. TranslateAI real-time voice-to-text translation interface with microphone activation prompt.

3.3 Testing and Performance Analysis

Functional Testing

The system was manually tested throughout development to ensure smooth performance. Translation accuracy was checked against Google Translate and DeepL to confirm similar quality.

Performance Testing

Latency measurements recorded:

- Text translation with an average of 2.17 seconds.

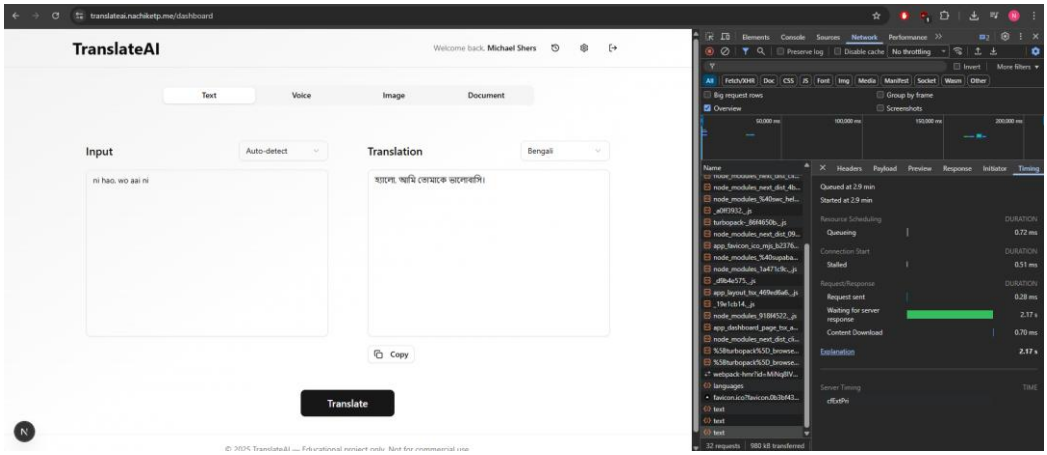

Figure 13. Text translation latency measured via Devtools

- Real-time voice translation with an average of 45 seconds
- Image translation with an average of 5.47 seconds

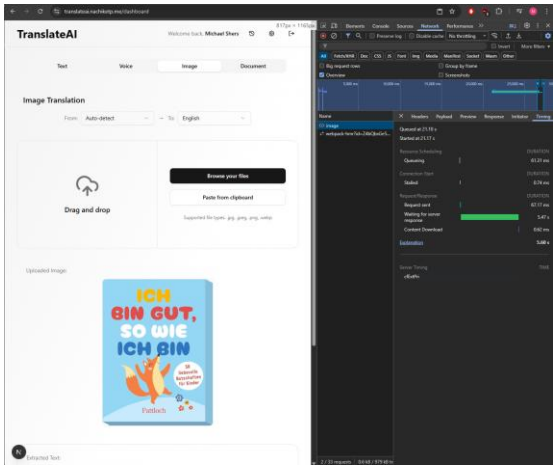

Figure 14. Image translation latency measured via Devtools

- Document translation with an average of 32 seconds (Depends on file size)

Cross-browser testing confirmed full compatibility on Chrome, Edge, and Safari.

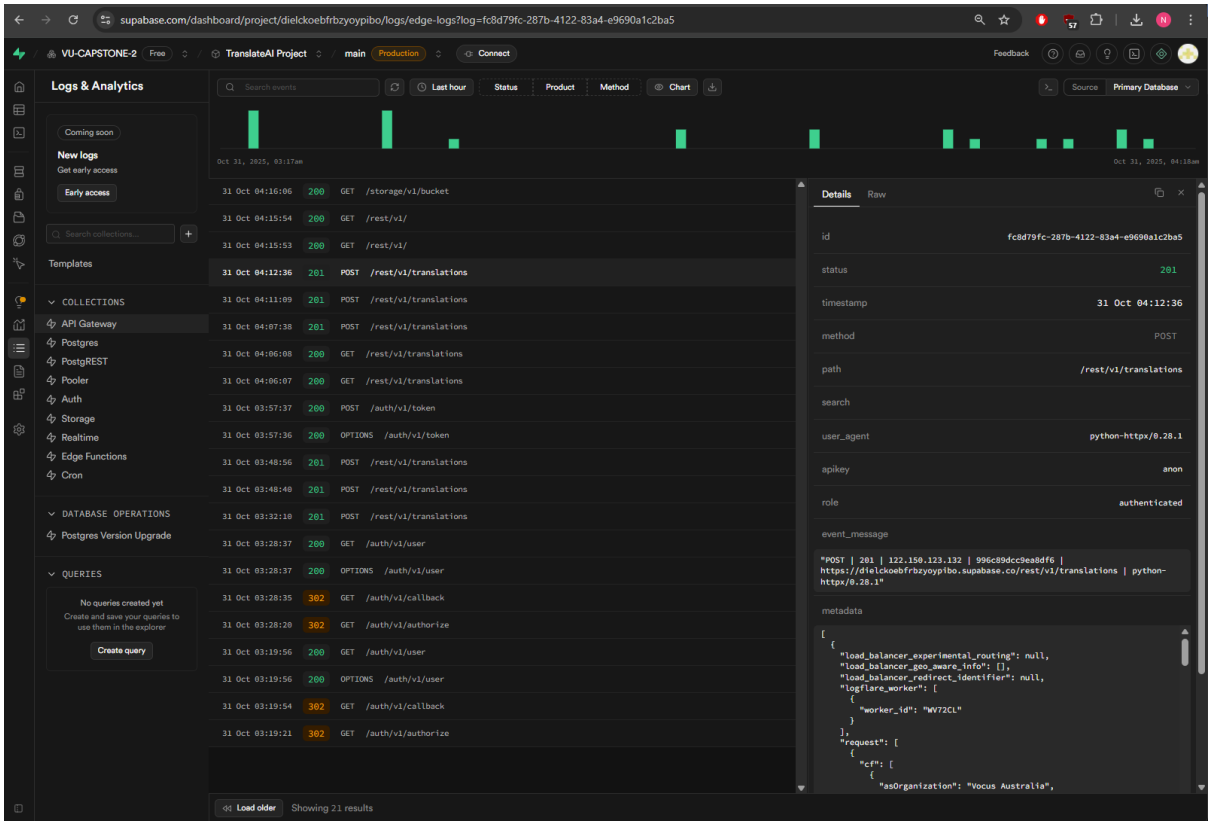

Figure 15. Supabase dashboard showing stored translation history records linked to user accounts.

4.0 Documentation Status

4.1 User Manual Draft

The user manual is under active preparation and currently includes a section on “Getting Started” located in the `readME.md` file in the root folder of TranslateAI. It will be utilised to create the User Manual later on.

The user manual will contain the following as well:

- Account creation and login process
- Performing text, voice, image, and document translations.
- Managing history and resetting passwords.
- Exporting translations and troubleshooting common errors.

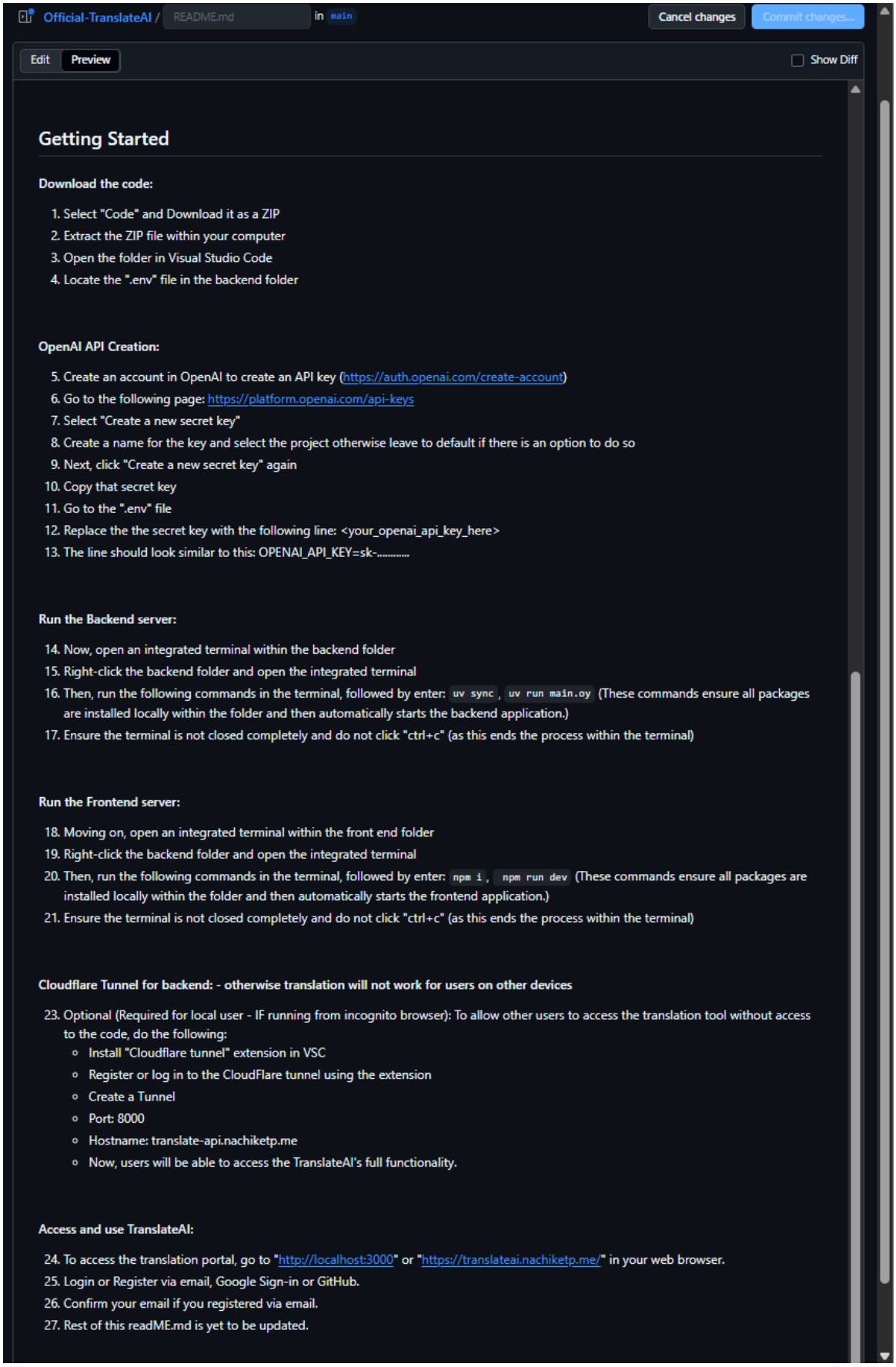

Figure 16. `readME.md` draft of the "Getting Started" section for the upcoming TranslateAI User Manual.

4.2 Technical Documentation

The technical documentation will outline the software architecture, database schema, environment configurations, and code deployment procedures. It will be structured into the following components:

- Overview of application layers (frontend, backend, database)
- List of dependencies and framework versions
- Security implementation details (Supabase RLS, Cloudflare SSL, OAuth Providers)
- Version control summary and GitHub repository reference

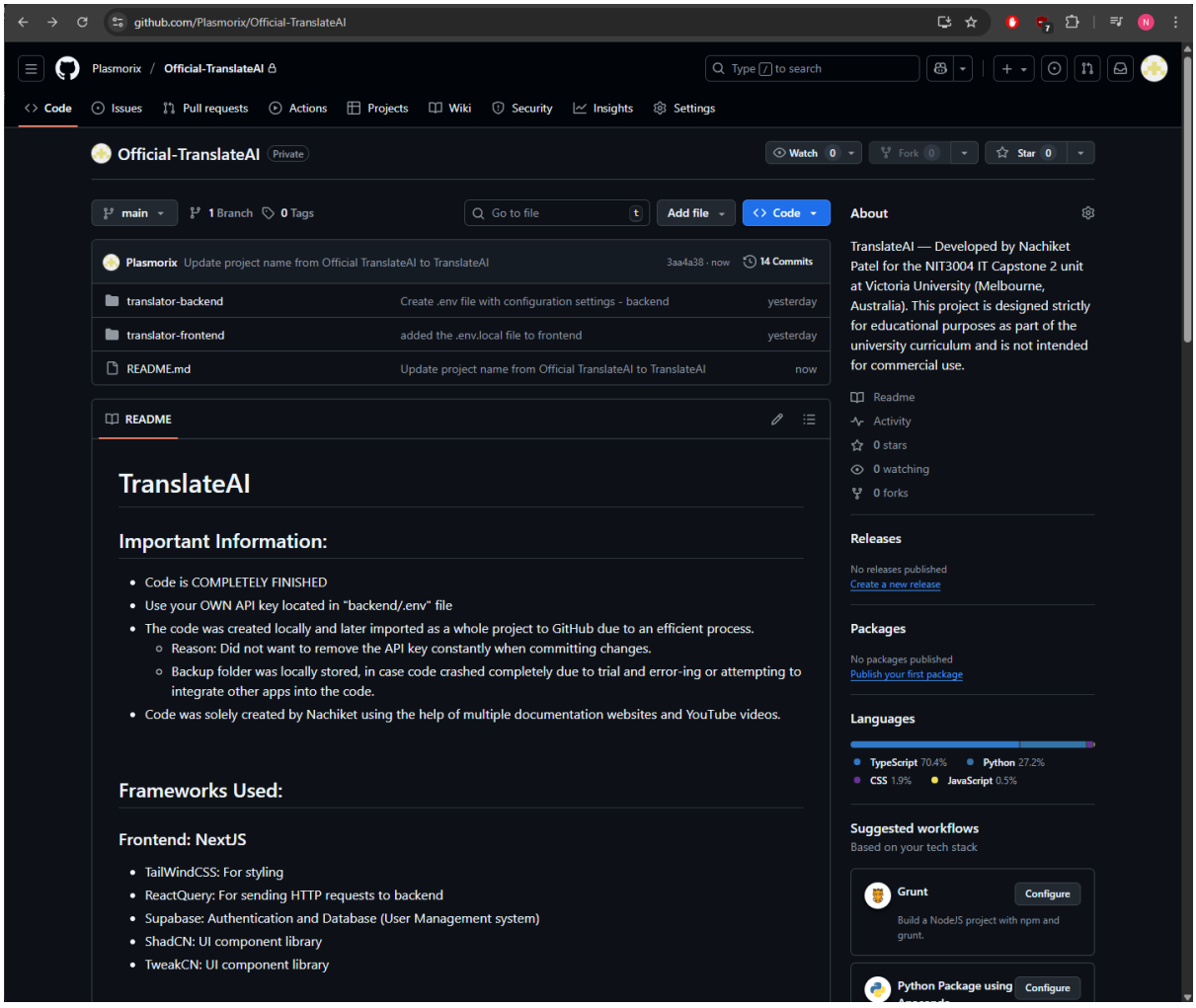

Figure 17. GitHub Repository of TranslateAI, showing project structure and commits.

5.0 Timeline and Next Steps:

The TranslateAI project followed an accelerated but highly structured three-day development timeline. Each phase was designed to deliver functional milestones with direct integration into the overall architecture.

5.1 Development Timeline Overview

Phase 1 – Planning and Environment Setup (Preliminary Stage)

Before commencing full development, the environment was prepared with the following configurations:

- Installed FastAPI, Uvicorn, Supabase, LangChain, Python, Next.js, and OpenAI dependencies.
- Registered the subdomain `translateai.nachiketp.me` through Namecheap and configured it with Cloudflare nameservers.
- Created a Git Repository called "Official TranslateAI"

This phase established a stable foundation for streamlined coding and deployment.

Phase 2 – Backend Development (Day 1)

Key tasks completed:

- Created all FastAPI service modules (translation, audio, database, and schema).
- Connected backend to Supabase via RESTful APIs.
- Integrated OpenAI API for real-time translation and PDF parsing.
- Tested endpoint responses in VS Code terminal and confirmed API calls.
- Stored backup versions locally to prevent data loss during trials.

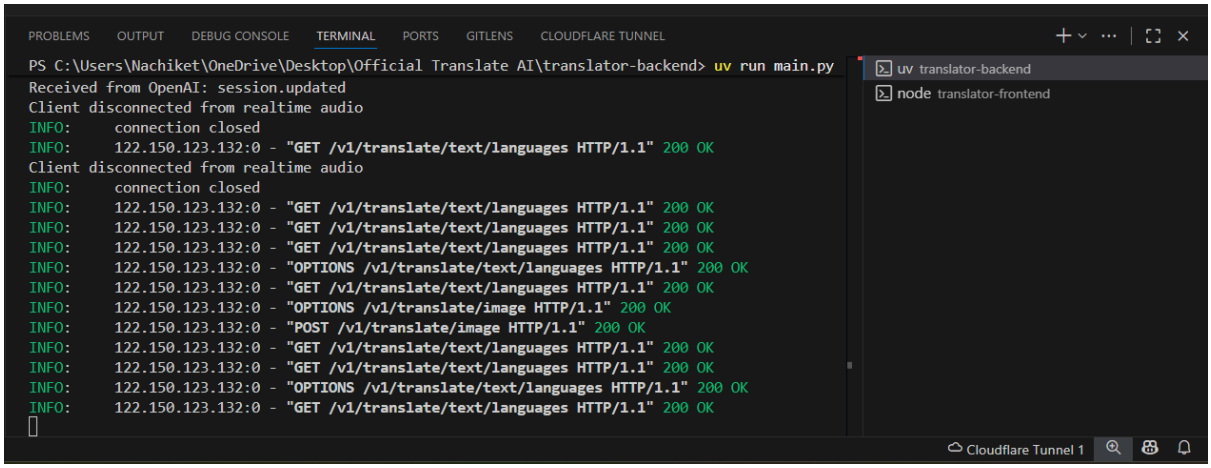

Figure 18. Backend console output showing successful API response (shown by "200 OK").

Phase 3 – Frontend Development (Day 2)

Key tasks completed:

- Created login, registration, and settings pages using Next.js and TailwindCSS.
- Linked frontend to Supabase for authentication and session management.
- Built history page to display previous translations dynamically.
- Added password validation logic and implemented secure sign-up flow.

- Tested functionality on Chrome and Edge browsers.

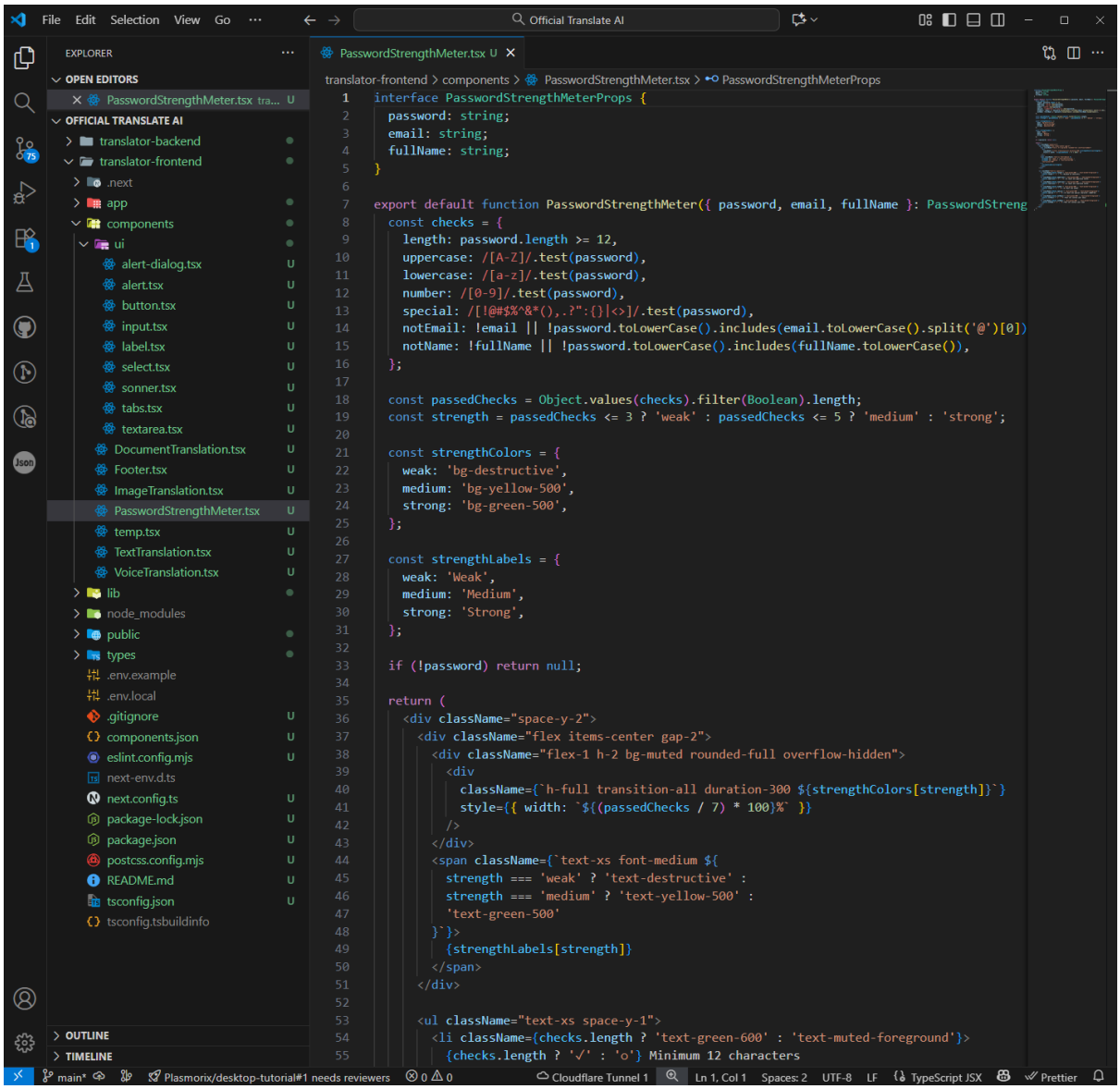

Figure 19. Development of TranslateAI frontend using Next.js and TailwindCSS.

Phase 4 – Cloudflare Deployment and Security Integration:

Key tasks completed:

- Migrated DNS from Namecheap to Cloudflare and configured tunnel access.
- Enabled HTTPS with free SSL certificate and tested secure routing.
- Masked backend port 8000 using Cloudflare Tunnel to prevent IP exposure.
- Performed live translation demo for lecture (Joseph); translation successfully returned accurate results.

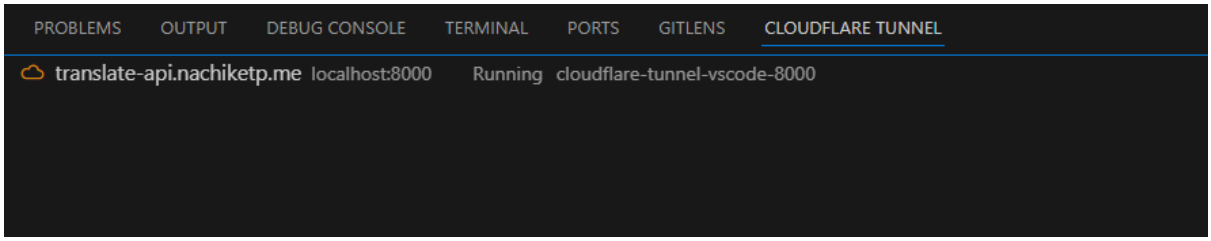

Figure 20. Successful backend deployment via Cloudflare extension in VSC.

5.2 Remaining Work

Despite reaching eight per cent completion, several key components remain under development to ensure complete functionality:

Bug Fixes:

- The real-time voice-to-text translation has slower performance as it is limited by GPT-4o's speed but retained due to budget constraints.

Documentation and Delivery:

- Finalise user manual with walkthroughs and screenshots.
- Complete technical documentation.
- Create the poster
- Prepare final presentations slides for submission.

5.3 Projected Completion Timeline

The following table outlines the projected timeline for completing the remaining tasks leading to 100% project completion.

Task Description	Start Date	End Date
Technical Demonstration	2nd/11/2025	5th/11/2025
User Manual	6th/11/2025	7th/11/2025
Poster	8th/11/2025	9th/11/2025
Final Presentation	10th/11/2025	13th/11/2025

6.0 Conclusion

TranslateAI showcases a successful integration of artificial intelligence, secure cloud infrastructure, and modern web development. The platform demonstrates how natural language processing can be incorporated into a real-world application with user-friendly design and enterprise-level security.

At this eighty per cent completion stage, the system already provides end-to-end translation across multiple languages using text, voice, image, and document inputs. It has been deployed securely under Cloudflare, with authentication and storage handled through Supabase.

The remaining work primarily involves optimisation and documentation rather than core functionality, highlighting the project's overall readiness. TranslateAI effectively meets the intended learning outcomes for the IT Capstone Project by demonstrating practical implementation of security, API integration, and system design principles within a real-world AI application.

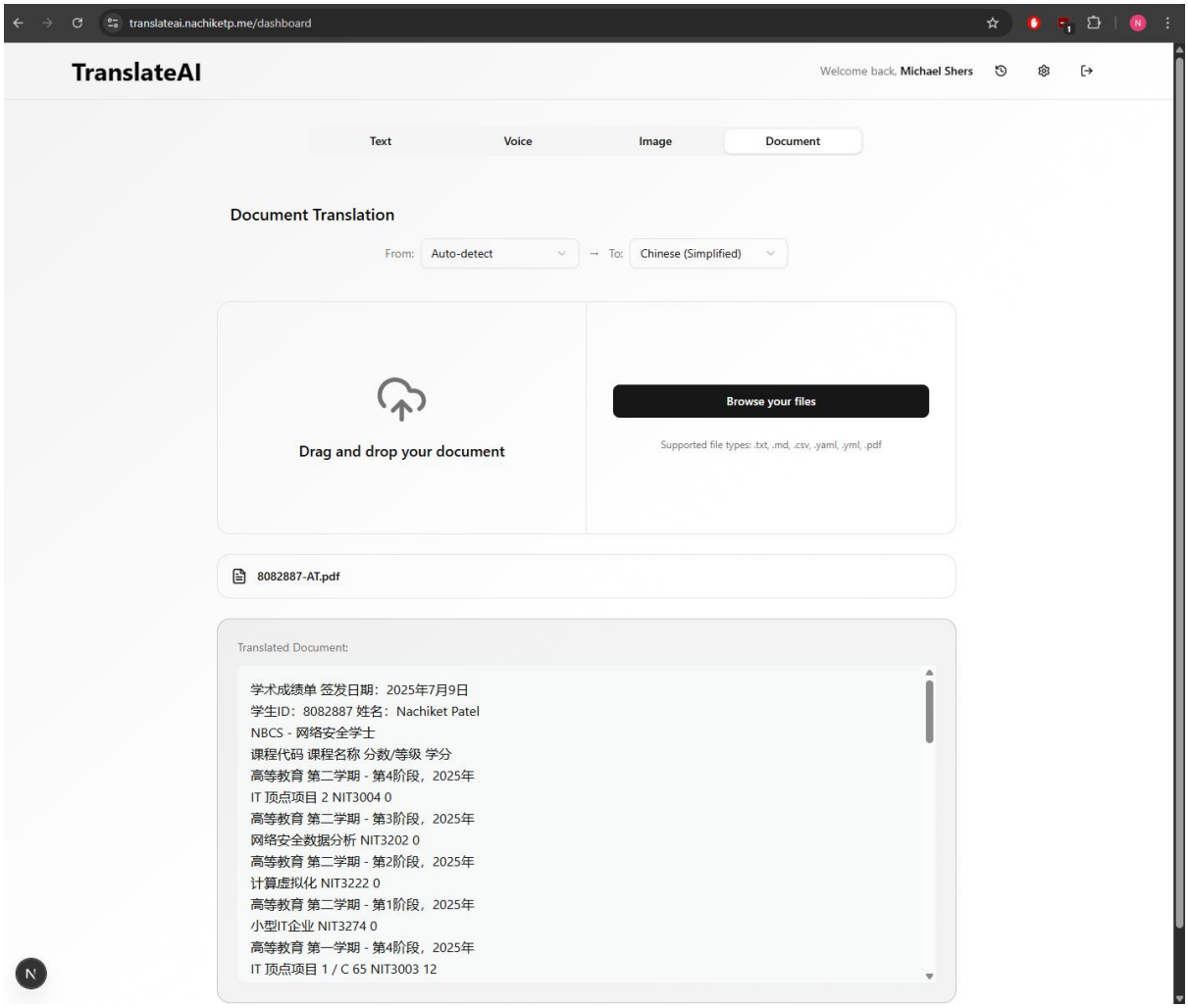

Figure 21. Example of a successful document translation from Auto-Detect to Chinese (Simplified) displayed on the TranslateAI interface.

7.0 Reflection

The development of TranslateAI was deeply educational that strengthened both technical and personal skills. Working independently required effective time management, strong problem-solving, and consistent research. The process provided a comprehensive understanding of how multiple frameworks such as Next.js, FastAPI, Supabase, and Cloudflare, can operate together within a single secure ecosystem.

Throughout this project, I learned how to design and implement authentication system, manage databases with role-based security, and handle deployment through encrypted tunnels. More importantly, it emphasised the values of persistence, iteration, and adaptability when developing under time constraints.

The project also reinforced the importance of documenting each stage carefully. Writing detailed project notes not only facilitated debugging but also reflected the logical progression of the system's development. TranslateAI's progress from a concept to a nearly complete working prototype reflects the ability to translate theory into practice, which is an essential skill for a career in cyber security and technology innovation.

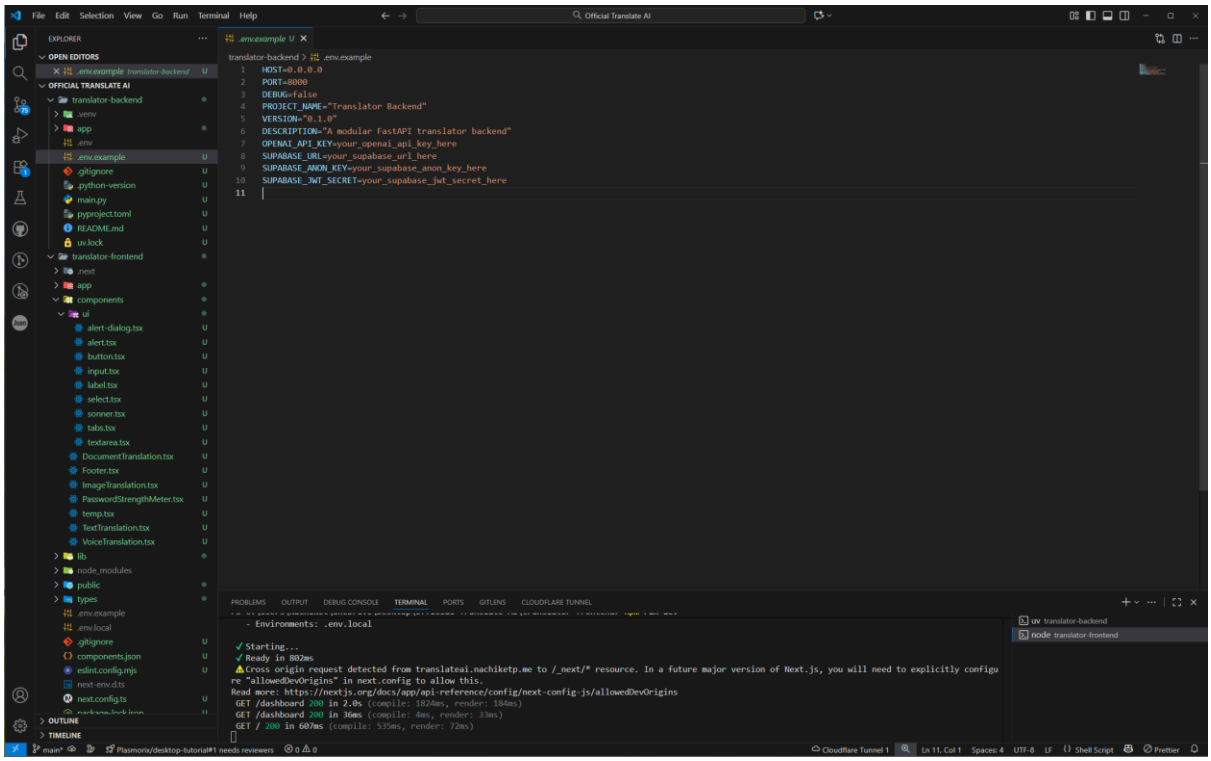

Figure 22. Development workspace used during the creation of TranslateAI.