NIT3004 – IT Capstone Project 2

Semester 2, Block 4 (H2B4) – Group 14

Progress Report for TranslateAI

Student Name and ID: Nachiket Patel (s8082887)

Submission Date: 31st/10/2025

Lecturer: Dr Gongqi (Joseph) Lin

Unit Convenor: Csaba Veres

Table of Contents

Introduction

This is my completed progress report for the product "TranslateAI". The project began as a group effort with Lachlan Brassington (s8082733) and Charlene Custodio (s8059240) during the IT Capstone 1 unit at Victoria University. The original proposal and timeline were finalised in Capstone 1; however, major changes occurred at the beginning of IT Capstone 2 due to limited collaboration and contribution from other group members. Evidence regarding this matter has been shown to the Lecturer and the Unit Convenor.

After two months of minimal progress and no completed tasks, I took the initiative to lead and assign responsibilities to the team. Despite this, there was still no participation from others. With approval from the lecturer and unit convenor, I decided to continue the project independently and began developing TranslateAI on my own. The entire codebase, including both the frontend and backend systems, was designed, implemented, and tested within four full nights of focused development.

This document is the revised and completed version of the earlier proposal and timeline, representing the final deliverable for the IT Capstone 2 unit.

TranslateAI was developed entirely locally using modern frameworks and APIs, with Supabase used as the secure online database for authentication and data storage. Supabase access can be provided to the lecturer upon request. A duplicate version of the source code was maintained for backup purposes. Local development was chosen to prevent the need to repeatedly remove the OpenAI API key during testing, ensuring a smooth and efficient development process. The complete project has been placed in a private repository on GitHub where access will be provided upon request.

Original Proposal and Timeline (NIT3003 - IT Capstone 1) – Moved here and discontinued:

https://drive.google.com/drive/folders/1Z1xOB-lHiLIcI19e2JRmhWNONdoXNlK4?usp=sharing

GitHub Repository (Private – Access Provided upon request)
https://github.com/Plasmorix/Official-TranslateAI
Please refer to the "readME.md" file situated within the repository for more information on how to use the application.

Domain and Hosting:
https://translateai.nachiketp.me
(Secured via Cloudflare DNS and Tunnel, run from local host)

App Progress Report

The development of TranslateAI was completed over three consecutive nights (26th – 28th October 2025), with testing continuing the 29th. Each day involved full-stack progress including backend, frontend, and security configurations while using the local and also cloud-based tools. The following sections document the daily progress and subsystem implementation details.

Frontend Sub-System:

The frontend utilised the Next.js 16 framework for development overall, providing a fast and efficient foundation for building a modern web application. TailWindCSS was used for styling, enabling for a responsive and customisable interface through simple utility classes. The user interface was enhanced further using ShadCN and TweakCN, ensuring consistent and visually appealing components across the platform (for example, maintaining a black-and-white theme with a matching registration and login page). ReactQuery handled the data management, increasing the rate of performance while keeping the client and the server data in sync.

Moreover, authentication was implemented with Supabase, supporting secure user login via Email/Password, Google and GitHub oAuth (To access in Supabase: Authentication→Sign In / Providers→Auth Providers). Additionally, API communication with the backend occurred through both the HTTP and WebSocket protocols, powered by a Uvicorn Server to enable real-time functionality.

The frontend was hosted on Cloudflare Pages, linked to a custom sub-domain translateai.nachiketp.me using Cloudflare DNS and Cloudflare Tunnel. This setup ensures a globally accessible, fast, and secure content delivery network (CDN) for the web interface. A Cloudflare-managed Full SSL certificate was applied to all incoming traffic and outgoing traffic, guaranteeing encrypted communication between the client and server. The frontend was tested across multiple devices and browsers, including Chrome and Edge on laptops while Safari on iPhones, to ensure consistent responsiveness and performance. Additional protection was provided by Cloudflare's DDoS mitigation and Web Application Firewall (WAF) rules, preventing malicious traffic and unauthorised access attempts. The frontend originally ran on localhost:3000; however, since the Cloudflare Tunnel Free Tier allows only one persistent tunnel, it was permanently assigned to the frontend. The frontend originally ran on localhost:3000, but the Cloudflare Tunnel Free Tier supports only one persistent tunnel. As a result, it was assigned to the frontend, so all local requests to port 3000 now redirect securely to translateai.nachiketp.me, while the backend tunnel must be started manually each time via the Cloudflare extension in Visual Studio Code (VSC).

Translation Interface

Files Used:
app/page.tsx
components/TranslationBox.tsx
components/VoiceControls.tsx
services/api.ts

Developer: Nachiket	
Date	**Notes**
26th/10/2025	Initialised frontend using Next.js 16 and TailWindCSS Set up ShadCN UI components for modern, accessible styling. Configured page routing and environment variables for API communication. Designed minimal two-panel translation (input → output). Implemented translation form, voice input buttons, and output box. Added responsive design for desktop. Presented demo to the Lecturer (Joseph) – Core translation not functioning
27th/10/2025	Resolved API key issue – added OpenAI credit for backend requests. Translation (text-to-text, real-time voice-to-text, audio-to-text, image-to-text, and document-to-text) fully operational. Demonstrated functional application to the Lecturer. Teacher feedback: deepen understanding of backend→frontend API flow for text, image, audio, document input. Teacher suggested: “translate-back” feature... Not implementing it because I aim to prioritise core functions due to limited time. Intentionally kept UI simple for clarity and performance. Demonstrated real-time translation working on both laptop and phone – unlike Google Translate, which limits real-time voice translation to mobile devices only.
28th/10/2025	Cloudflare Tunnel configured for frontend hosting at translateai.nachiketp.me Verified DNS migration from Namecheap to Cloudflare; Adjusted records and SSL Implemented HTTPS redirect and verified front-end connection to backend API (port 8000) Successfully integrated Auth0 with Supabase as Third-Party Auth. Configured Auth0 security protocols and etc. Auth0 integration to the code failed due to Auth0 documentation using outdated Next.js v13 methods. Final visual refinements completed; confirmed responsive layout and app notifications.
29th/10/2025	Tested TranslateAI from an iPhone 14 – frontend loaded, but backend communication failed – In depth, Log-in/Registration = Success, Translations = Fail Identified missing backend tunnel; resolved by mapping FastAPI port 8000 to Cloudflare via secure tunnel (Utilised the Cloudflare tunnel extension in VSC). Post-tunnel setup, full translation (text, voice, image, document) worked on iPhone 14 and laptop.

Authentication and Account Management

Files Used:		
lib/supabaseClient.ts	app/(auth)/forgot-password/page.tsx	
app/(auth)/login/page.tsx	components/SocialButtons.tsx	app/dashboard/page.tsx
app/(auth)/register/page.tsx	components/HistoryList.tsx	components/AuthForm.tsx
app/settings/page.tsx	components/PasswordStrength.tsx	services/auth.ts

Developer: Nachiket	
Date	Notes
26th/10/2025	Built registration/login pages. Wired Supabase client. Added password-strength indicator and social (Google/GitHub) buttons.
27th/10/2025	Enabled oAuth providers, translate history, forgot-password flow. Implemented ProtectedRoute guards.
28th/10/2025	Added settings page to change password, improved form validation, loading states, and responsive layouts.

Backend Sub-System:

The backend was developed using the FastAPI framework in Python, running a Uvicorn server to handle both standard and real-time API requests efficiently. It integrated several key libraries, including LangChain, the OpenAI API, and PyPDF2, to enable advanced language processing, document handling, and intelligent translation workflows. Real-time communication was supported via the WebSocket connections, ensuring smooth interactions with the frontend.

The application utilised Supabase with a PostgreSQL database for secure data storage and management. Brevo SMTP was configured to handle automated email services such as verification notifications. To ensure strong protection and safe user access, the backend was secured through a variety of Cloudflare Tunnel, SSL encryption, oAuth authentication, and password validation.

Now, the backend was deployed through a Cloudflare Tunne1 (using its authorised extension from VSC), securely exposing the local uviorn server (running on port 8000) to the web via the /api subdomain endpoint. This method eliminated the need for direct port forwarding or external hosting, while maintaining encrypted and authenticated connections. The backend communicated with the frontend using HTTPS and secure WebSocket protocols, both operating over TLS 1.3 for maximum security. The database and authentication services were managed externally through Supabase, decreasing operational burdens (Smoother processes with fewer effort/time/resources) and simplifying long term maintenance (Allows for easier and less complex tasks such as updates, debugging, etc.). This entire deployment approach by utilising cloud.

Authentication and Account Management

Files Used:		
Main.py	app/dependencies/auth.py	
app/config.py	app/utils/security.py	app/routes/history.py
app/routes/user.py	app/services/database.py	app/schemas/history.py

Developer: Nachiket	
Date	Notes
26th/10/2025	Implemented Supabase JWT verification (security.py), added auth dependency, stubbed /history routes.
27th/10/2025	Completed POST/GET /history endpoints; connected Supabase client; validated token scope.
28th/10/2025	Added /me route; confirmed RLS enforcement; ensured HTTPS + Bearer token only access.

Core Translation Engine

Files Used:
main.py
app/services/translator.py
app/services/audio.py
app/services/database.py

Developer: Nachiket	
Date	Notes
26th/10/2025	Created FastAPI backend with modular structure under "app/services/". Implemented translation endpoints using LangChain and OpenAI GPT-4 API. Configured Supabase SDK for authentication and data persistence. Connected Brevo SMTP for confirmation emails (account verification). Integrated Auth0 with Supabase at the account level for external login handling (not active in local code). Tested text translation endpoint — request passed but failed due to OpenAI credit issue (fixed next day).
27th/10/2025	Added working translation endpoints for text, voice, and document types. Implemented asynchronous WebSocket connections for live voice input/output translation. Added `translate_document` route using PyPDF2 to extract and translate text from uploaded files. Verified OpenAI API key connection and output streaming to frontend. Implemented structured error handling and logging. Conducted demo with lecturer — all translation modes working correctly.
28th/10/2025	Configured Cloudflare Tunnel for backend (port 8000). Secured backend access using Cloudflare SSL and DDoS protection. Implemented password strength validation in Supabase for email signups. Confirmed multi-provider security: Google and GitHub oAuth via Supabase, Brevo email confirmation for registered users. Verified secure communication between frontend and backend using HTTPS and token-based API calls.

Deployment and Cloudflare Tunnel

Files Used:		
supabase/config.toml	package.json	app/config.py
.env	main.py	

Developer: Nachiket	
Date	Notes
26th/10/2025	Completed Namecheap domain setup and linked to Cloudflare by updating nameservers. Verified domain ownership and initiated DNS transfer. Configured A and CNAME records in Cloudflare to point to translateai.nachiketp.me. Created Cloudflare project with default SSL (Full/Strict mode).
27th/10/2025	Tested Cloudflare DNS propagation. Configured frontend deployment for temporary public access via Cloudflare Pages preview. Verified HTTPS access and tested translation routes through the web proxy.
28th/10/2025	Installed and configured Cloudflare Tunnel locally (VSC extension) to securely expose backend (Uvicorn on port 8000) to the public without port forwarding. linking localhost:8000 to translate-api.nachiketp.me. Verified connectivity and SSL handshake. Enabled WebSocket support for realtime translation endpoints.
29th/10/2025	Performed full mobile test. Identified missing backend tunnel from earlier setup — API unreachable externally. Re-ran tunnel configuration linking backend service via extension. Tested end-to-end translation (text, voice, and document) from phone and laptop successfully. Verified tunnel stability and latency.

Miscellaneous Features

Developer: Nachiket	
Enhancements	Notes
Domain Migration	Transferred the official project domain from Namecheap to Cloudflare by updating the assigned nameservers. DNS records were manually rewritten to point the A and CNAME records toward translateai.nachiketp.me.
Email Delivery	Configured Brevo SMTP for transactional email delivery. This enabled verification links, password reset notifications, and user registration confirmations to be sent securely using TLS-encrypted communication.
Auth0 Integration Attempt	Auth0 was integrated at the Supabase level for user mapping but was not connected to the local codebase due to incompatibility with the new Next.js 16 App Router. Future releases may revisit this integration using the Auth0 SDK for React Server Components. Auth0 is not being used for User Registration/Log-In
Version Control	A complete GitHub repository was created at the end of the build cycle. Commits were delayed intentionally to avoid leaking the OpenAI API key during development. A local Git backup and an additional ZIP archive were also maintained as a crash-recovery plan.
Testing	Cross-browser testing performed on Chrome, Edge, and Safari verified consistent translation accuracy and real-time voice functionality. Mobile and desktop both achieved identical response times after Cloudflare Tunnel integration.
Performance	Average API response latency measured through terminal: • Text translation: ~1.4s • Voice translation (real-time): ~2.2s • Document translation (PDF): ~3.1s Image and Audio translation average API have not been measured yet.

Design and Style Constants

Colours:

Primary colour	oklch(0.205 0 0)	bpdPrimaryColour
Secondary colour	oklch(0.97 0 0)	bpdSecondaryColour
Accent colour	oklch(0.97 0 0)	bpdAccentColour
Muted colour	oklch(0.97 0 0)	bpdMutedColour
Card background	oklch(1 0 0)	bpdCardBgColour
Border colour	oklch(0.922 0 0)	bpdBorderColour
Colour	oklch(0.922 0 0)	bpdInputColour
Ring colour	oklch(0.708 0 0)	bpdRingColour
Destructive colour	oklch(0.577 0.245 27.325)	bpdDestructiveColour
White colour	oklch(1 0 0)	bpdWhiteColour
Dark colour	oklch(0.145 0 0)	bpdDarkColour

All colours are stored as OKLCH colour space values for improved contrast consistency across the devices (View each colour at https://oklch.com/).

Images:

Located in: translator-frontend/

Fonts:

Primary Font	Geist Sans (Google Font)
Secondary Font	Geist Mono (Google Font)
Import in layout.tsx using:	(1) --font-geist-sans (2) --font-geist-mono

- Sans-serif used for all general UI Elements
- Mono font used for code display, API keys, and backend logs.

Text Styling:

headlineLarge	font: Geist Sans	size: 32px	weight: 700
headlineMedium	font: Geist Sans	size: 24px	weight: 600
headlineSmall	font: Geist Sans	size: 20px	weight: 600
titleLarge	font: Geist Sans	size: 18px	weight: 500
titleMedium	font: Geist Sans	size: 16px	weight: 500
titleSmall	font: Geist Sans	size: 14px	weight: 500
bodyLarge	font: Geist Sans	size: 16px	weight: 400
bodyMedium	font: Geist Sans	size: 14px	weight: 400
bodySmall	font: Geist Sans	size: 12px	weight: 400
monoText	font: Geist Sans	size: 14px	weight: 400

Text:

APP:	
• App name:	TranslateAI
• App tagline:	Created by Nachiket Patel.
• Footer notice:	© 2025 TranslateAI — Educational project only. Not for commercial use.
Text Translation Module:	
• Section title (Input):	Input
• Section title (Output):	Translation
• Button (Main):	Translate
• Button (Copy):	Copy
• Placeholder (Input):	Enter text to translate
• Placeholder (Output):	Translation will appear here…
• Toast (Empty input):	Please enter text to translate.
• Toast (Success):	Your text has been translated successfully.
• Toast (Copied):	Translation copied to clipboard.
• Toast (Error):	Translation failed.
Image Translation Module	
• Section title:	Image Translation
• Label:	From: / To:
• Placeholder (select):	Auto-detect
• Instruction (drag):	Drag and drop
• Button:	Browser your files
• Button:	Paste from clipboard
• Hint:	Supported file types: .jpg, .jpeg, .png, .webp
• Progress Message:	Processing image…
• Toast Errors:	- Please upload an image. - Maximum file size is 5MB. - No image found in clipboard. - Please allow clipboard access or use the browse button.
• Toast (Success):	Image translation completed successfully
• • Section Labels:	- Uploaded Image: - Extracted Text: - Translation:
Voice Translation Module	
• Section title:	Voice Translation
• Tabs:	Real-time / Upload File
• Realtime Section:	- Start Recording - Stop Recording - 🎤 Recording… Speak naturally and pause for translation - Click to start recording - ✅ Connected to real-time service

	- ⚠️ Disconnected – Connecting to real-time service - ⚠️ Note: This must be served via http://localhost or https:// for microphone access - You said: - Translation:
• Upload Section	- Click to upload audio file - Supported formats: MP3, WAV, M4A. FLAC (Max 25MB) - Processing… - Transcript: - Translation: - No transcript available - No translation available
• Toasts (Feedback)	- Connected to real-time translation - Authentication required for real-time translation - Connection error – check if backend is running - Failed to connect - Translation saved to history - Translation complete! - Recording started - Recording stopped - Failed to access microphone
Password Strength Meter	
Section Sub-headers:	- ✓ Weak / Medium / Strong
	- ✓ Minimum 12 characters
	- ✓ At least one uppercase letter
	- ✓ At least one lowercase letter
	- ✓ At least one number
	- ✓ At least one special character (!@#$%^&*)
	- ✓ Must not contain your email
	- ✓ Must not contain your name

More added during creation of the app. A full listing can be found in:

- /translator-frontend/lib
- /translator-frontend/app