University of Macau

Faculty of Science and Technology

澳門大學

UNIVERSIDADE DE MACAU

UNIVERSITY OF MACAU

2024-2025-CISC3003-001 Web Programming

Final Project

By

Team09

Kuan Hou In DC227476

Mak Weng Hou DC227551

Lei Ka Chon DC207583

Chong Chi Hoi DC226952

Abstract

This report details the development of the "Chatbot Website Project," a conversational chatbot website. Inspired by ChatGPT, the application offers a simplified yet functional interface with real-time chat capabilities, user authentication, conversation history management, and model selection features. The technology stack includes FastAPI for the backend, PostgreSQL for persistent data storage, and vanilla JavaScript for the frontend, ensuring a modern and efficient architecture. Key functionalities include WebSocket-based real-time messaging, secure authentication via Supabase (supporting Google and email logins), and a responsive multi-page design compatible across devices. The project comprises five main pages: chat interface, conversation history, user profile, subscription simulation, and settings. This report outlines the project's architecture, implementation details, and deployment process, while reflecting on the team's accomplishments and lessons learned. The development process emphasized practical application of web programming concepts, resulting in a robust, user-centric application that meets course requirements and showcases the team's collaborative efforts.

List of Service

The primary web application developed is the Chatbot Website, encompassing the following integrated services:

1. Real-Time Chat Interface – Enables seamless, interactive conversations with streaming responses.
2. User Authentication System – Provides secure login options using Supabase.
3. Conversation History Management – Allows storage, retrieval, and management of chat logs.
4. Model Selection Feature – Permits users to choose from available language models.
5. Multi-Page Website Layout – Delivers a navigable structure with five distinct pages.

List of Tasks (Group + Individual Assigned)

The project was a collaborative effort with tasks distributed as follows:

Mak Weng Hou [DC227551] - Backend development with FastAPI and WebSocket integration. Database design and management with SQLModel and PostgreSQL. Deployment setup with Docker and GitHub Actions.

Lei Ka Chon [DC207583] - Frontend development using HTML, CSS, and JavaScript. Authentication implementation using Supabase.

Kuan Hou In [DC227476] - Database design and management with SQLModel and PostgreSQL. Deployment setup with GitHub Actions.

Chong Chi Hoi [DC226952] - Frontend development using HTML, CSS, and JavaScript. Authentication implementation using Supabase.

Group efforts included planning, integration testing, and documentation.

Project Accomplishments

Front-End Development

The frontend was crafted with a mobile-first, responsive design using HTML, CSS, and vanilla JavaScript. Key pages include:

1. Chat Interface: Real-time messaging with markdown-rendered responses.
2. Conversation History: Infinite scrolling, search, and editable titles.
3. User Profile: Displays and updates user details and model preferences.
4. Subscription Simulation: Static plan details and status display.
5. Settings: Basic configuration options to meet multi-page requirements.

Back-End Development

The backend leverages Python with FastAPI and PostgreSQL:

1. Database Connectivity: SQLModel facilitates ORM for PostgreSQL, managing users and conversations.
2. Data Population: User profiles and chat logs are created dynamically.
3. Data Retrieval: Queries support real-time chat and history features.

Webpage and Coding:

Webpage Structure

• Landing Page (index.html): A marketing hub with sections for features, project purpose, team details, and calls-to-action. It uses modern CSS shapes and lazy loaded images for visual appeal and performance.

• Chat Page (home.html): The core interaction interface, featuring a responsive chat layout with distinct user and bot message styling.

• History Page (history.html): Displays a searchable, editable conversation list, with infinite scrolling for performance.

• Profile Page (profile.html): A form-based interface for managing user details and theme preferences, with a responsive card layout.

• Authentication Pages (login.html, register.html, ch_pw.html, reset.html): Secure interfaces for account management, with client-side validation and error handling. The login page is integrated with sign-in-with-Google modern method.

Coding Practices

• Frontend:

JS: VanillaJavaScript with ES modules (script.js, app.js, profile.js, etc.) ensures modular functionality. Event listeners handle user interactions such as form submissions and theme switching.

CSS: We designed custom CSS stylesheets for various page properties and implemented responsive design through media queries. A separate RL_styles.css style sheet is provided to match the authentication page.

• Backend:

WebSocket: WebSocket integration supports instant messaging. Python's FastAPI provides a clean API structure with SQLModel for database interaction.

Supabase: Provides third-party authentication and database services to complement PostgreSQL's user management.

Error handling: Server-side checks ensure strong input handling. Success and error messages use aria-live for accessibility.

• Third-party component integration

Google Sign-in: Implemented login.html using OAuth 2.0 to simplify authentication.

Cloudflare Security: An embedded script in the HTML archive integrates Cloudflare’s challenge platform for chatbot protection.

• Optimization strategy:

Delay loading of images to reduce initial page load time.

Minify CSS and modularize JavaScript to minimize resource usage

Efficient database query with indexing to support real-time functions.

Project Insight and Learning

The team gained expertise in:

- Real-time communication using WebSockets.
- Third-party authentication integration with Supabase.
- Responsive design principles for cross-device compatibility.
- Challenges like WebSocket stability were overcome through reconnection logic and rigorous testing.

Installation Instructions

1. Prerequisites

Ensure you have the following installed:

- **Docker** and **Docker Compose** (recommended for deployment)
- **Git**

You'll also need accounts for:

- **Supabase**: For authentication and database.
- **OpenAI** (or another API provider, if applicable)

2. Set Up Your Supabase Instance

Follow these steps to create and configure your own Supabase project.

2.1 Create a Supabase Account

- Visit Supabase and sign up for an account.

2.2 Create a New Project

1. Log in to your Supabase dashboard.
2. Click **New Project** and provide:
    a. A project name.

b. A region for hosting.

c. A secure database password (save this).

2.3 Obtain Supabase Credentials

1. In the dashboard, go to **Settings** > **API**:
    a. **Project URL**: Copy this (your SUPABASE_URL).
    b. **Anon Key**: Copy this (your SUPABASE_ANON_KEY, for frontend use).
2. Go to **Settings** > **Database**:
    a. Copy the **Connection String** (your DATABASE_URL).
3. Go to **Settings** > **API** > **JWT Settings**:
    a. Copy the **JWT Secret** (your SUPABASE_JWT_SECRET).

3. Configure Environment Variables

3.1 Backend Configuration

1. *Copy the example environment file:*

Copy **.env.cloud-db.example** and rename to **.env**

2. *Edit .env with your Supabase credentials:*

```
DATABASE_URL=postgresql://[user]:[password]@[host]:[port]/[dbname]
SUPABASE_URL=https://your-supabase-project.supabase.co
SUPABASE_JWT_SECRET=your-jwt-secret
OPENAI_API_KEY=your-openai-api-key
```

3.2 Frontend Configuration

The frontend needs these variables:

- SUPABASE_URL: Your Supabase project URL.

- SUPABASE_ANON_KEY: Your Supabase anon key.

1. *Go to /frontend/public/js/config.js*
2. *Edit config.js with your Supabase credentials:*

SUPABASE_URL=https://your-supabase-project.supabase.co

SUPABASE_ANON_KEY=your-supabase-anon-key

3.3 Docker Configuration

1. *Go to /docker/production/.env*
2. *Copy the example environment file:*

   Copy .env.cloud-db.example and rename to .env

3. *Edit .env with your Supabase credentials:*

   DATABASE_URL=postgresql://[user]:[password]@[host]:[port]/[dbname]
   SUPABASE_URL=https://your-supabase-project.supabase.co
   SUPABASE_JWT_SECRET=your-jwt-secret
   OPENAI_API_KEY=your-openai-api-key

4. Run the Project

Using Docker (Recommended)

Start the application by running,

you may choose to replace ‘docker/development/docker-compose.yml’ with ‘docker/production/docker-compose.yml’ if you want to run the production version of the application.

‘BASH’
docker compose **--project-directory** ./ -f docker/development/docker-compose.yml **--env-file** docker/development/.env up **--build --remove-orphans**

Add a **--detach** flag at the end to run it in detached mode.

Database Initialization

- When the app runs, it should automatically create database tables using SQLMode.
- The table is already write in /backend/src/chatbot_api/database.py

Project Incomplete

Payment method implementation

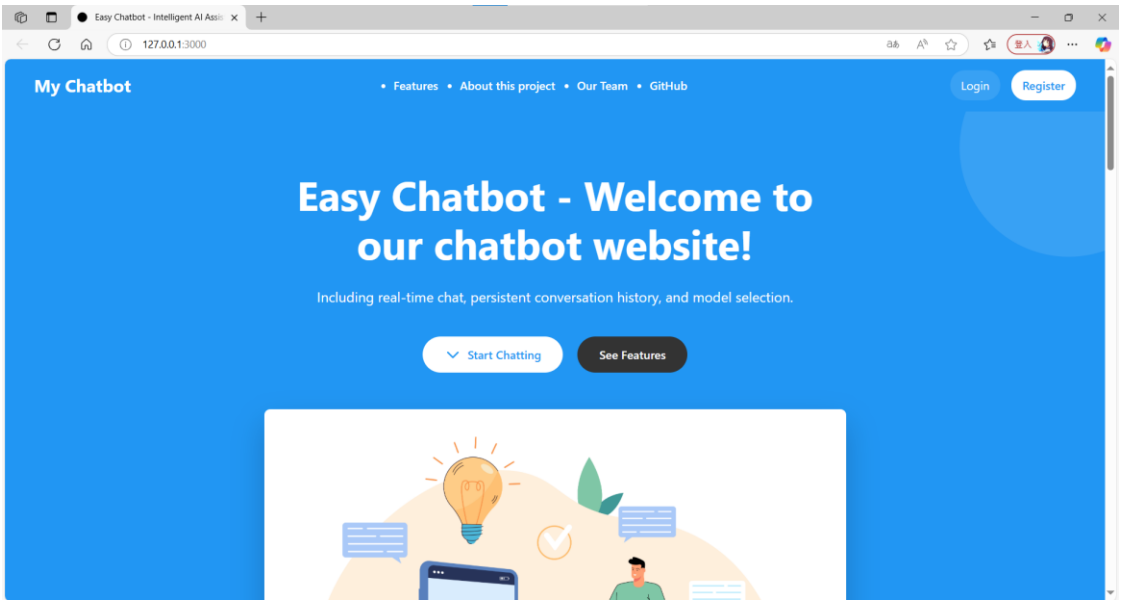
Subscription Payment simulation

Model Selection

Workthrough (frontend and Backend)

Welcome Page:

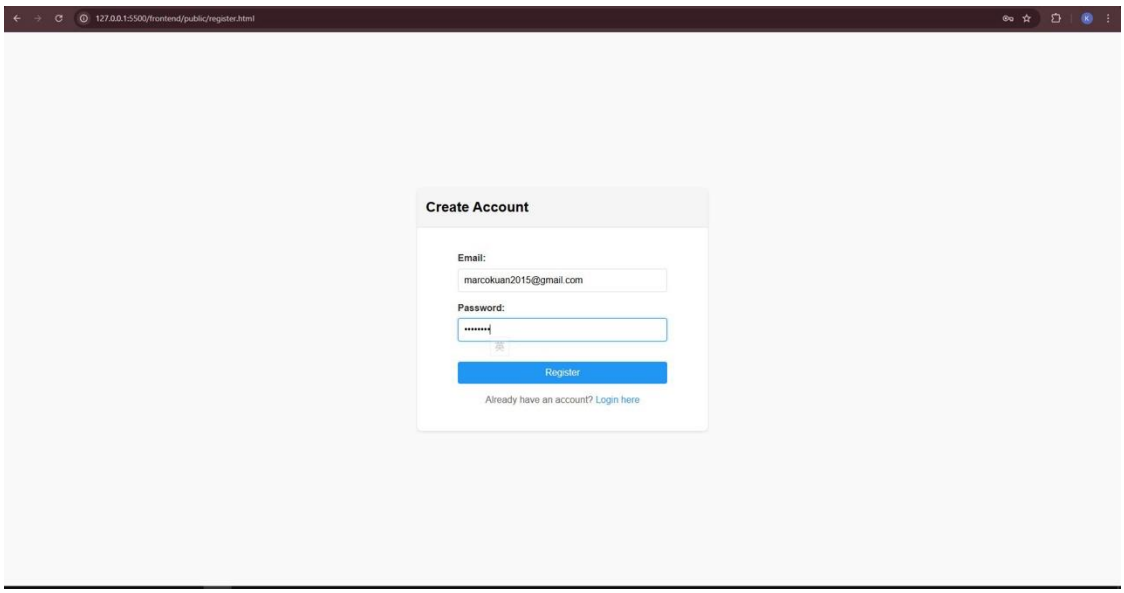
I enter the website; I go to the welcome page.



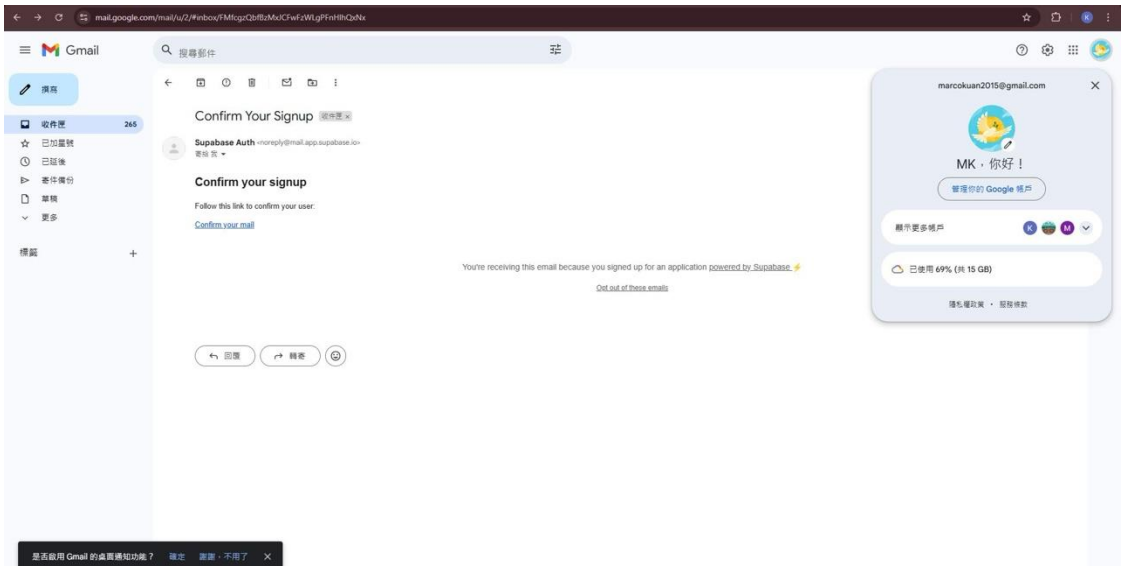
I click the Register button and then I am directed to the register page.

Reister Page:

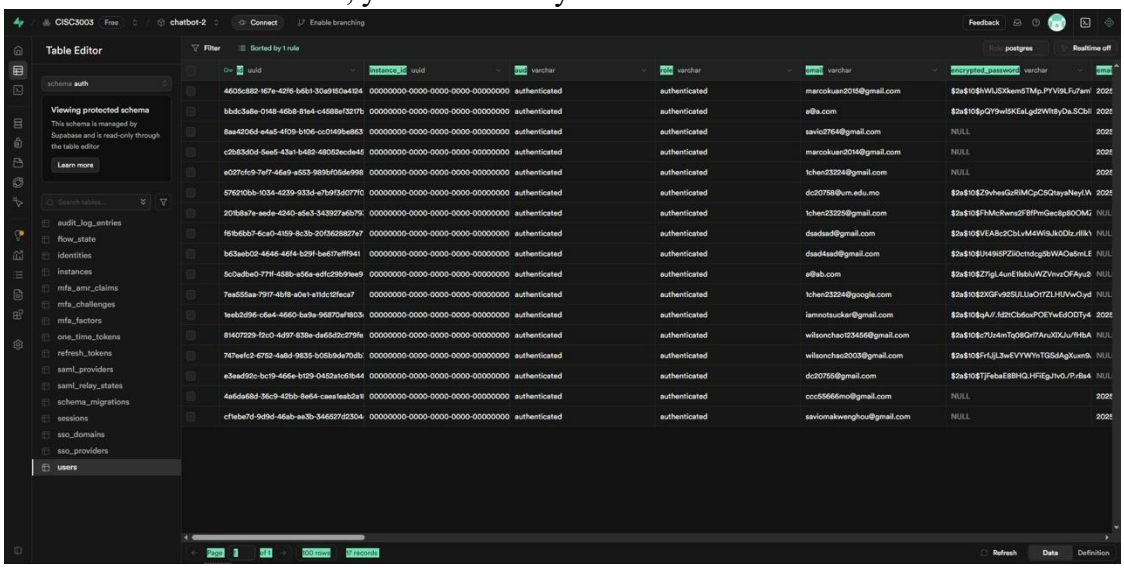
(I am going to Register a new Account on it)



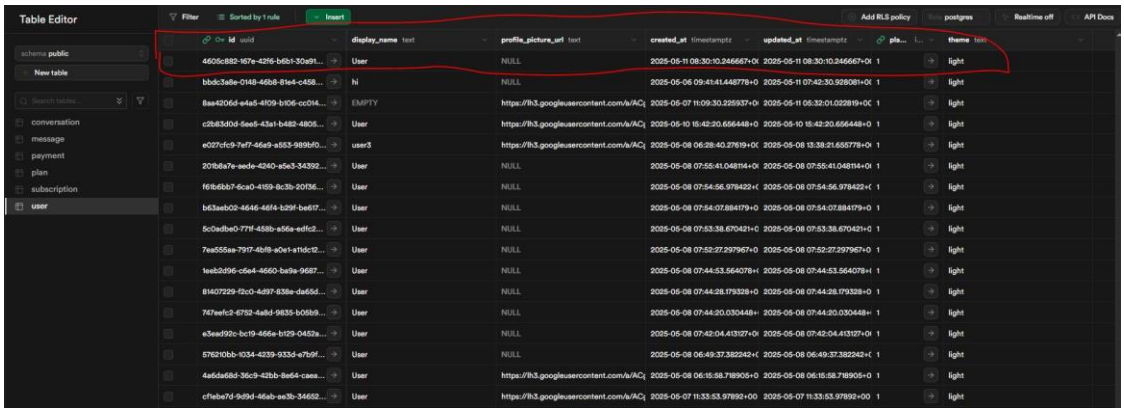
Validation Email Received (send by Supabase), then I click the Confirmed your email button.



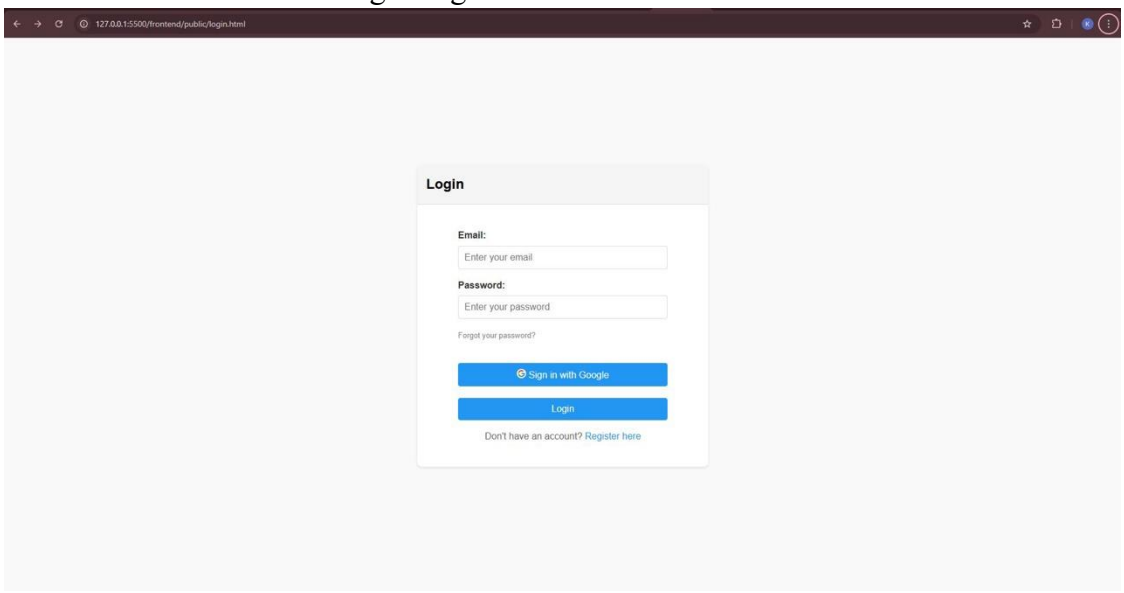
Meanwhile in the database, you can find my email in the first row.



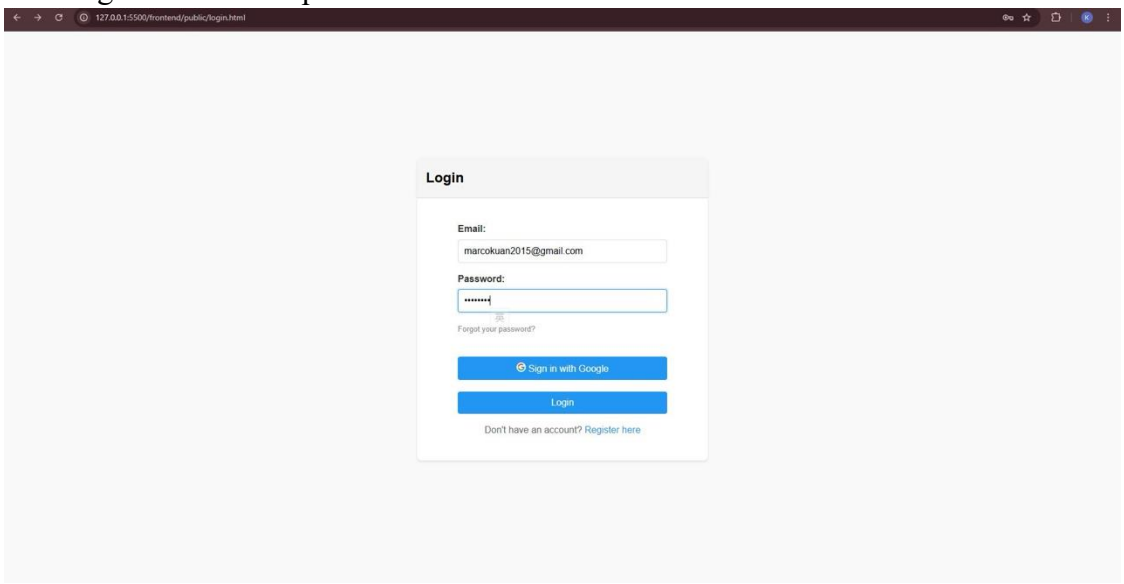
On the public user table(the first row is my new account)



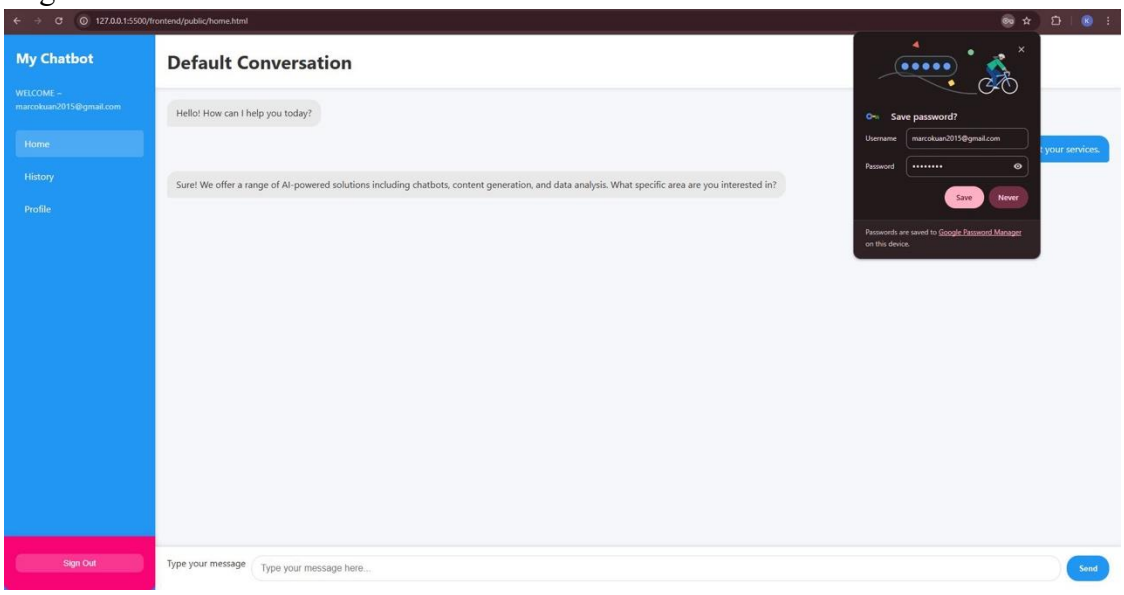
Then I am directed to the Login Page



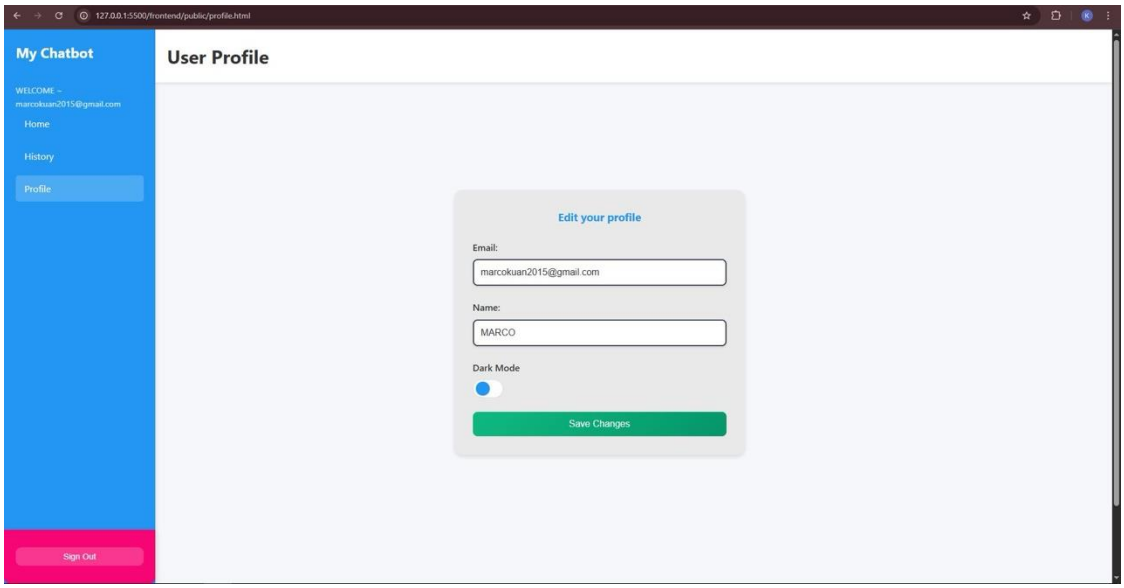
Entering the email and password



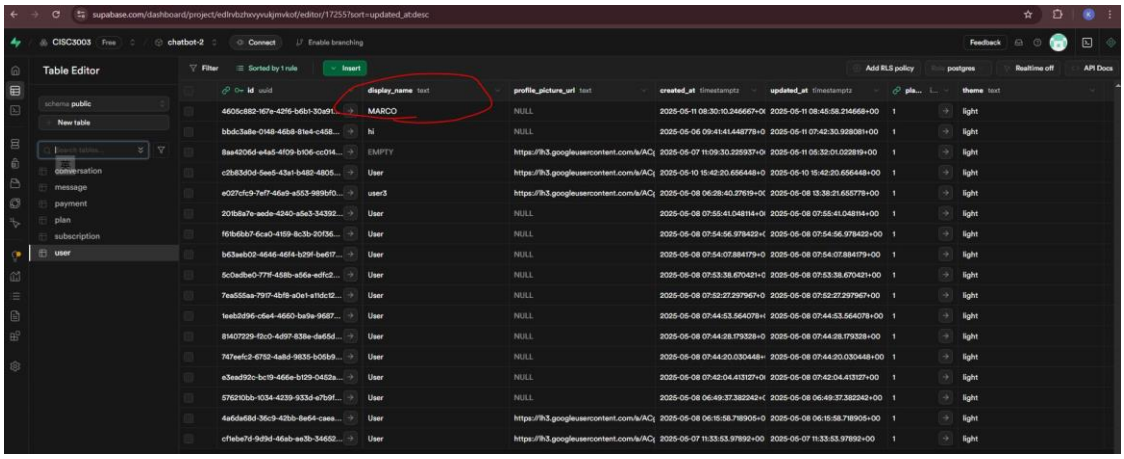
Login successful



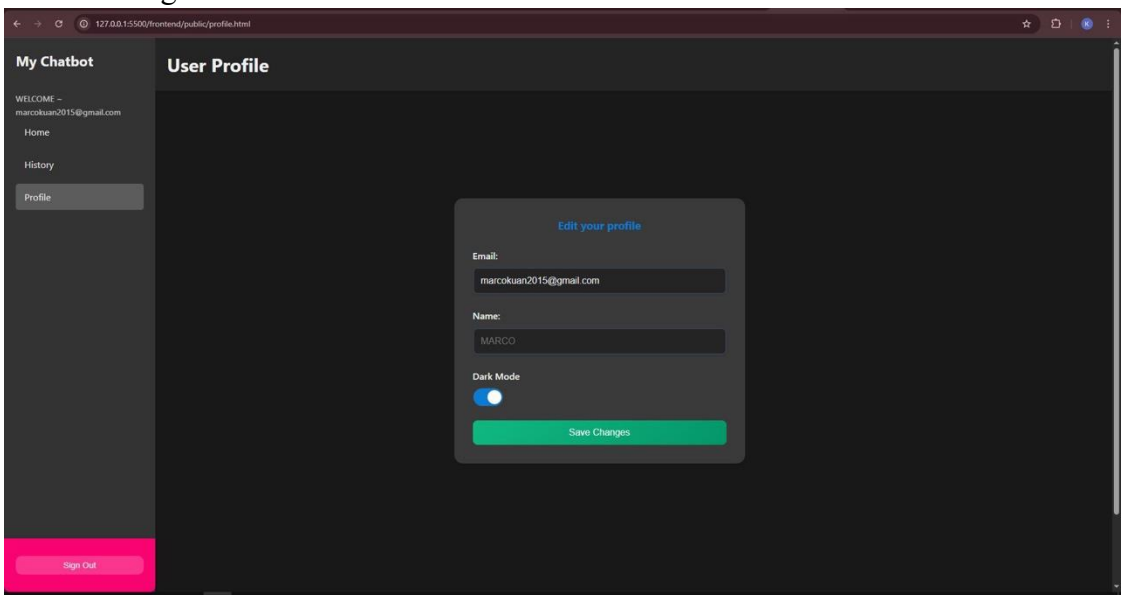
Change Name in the Profile page



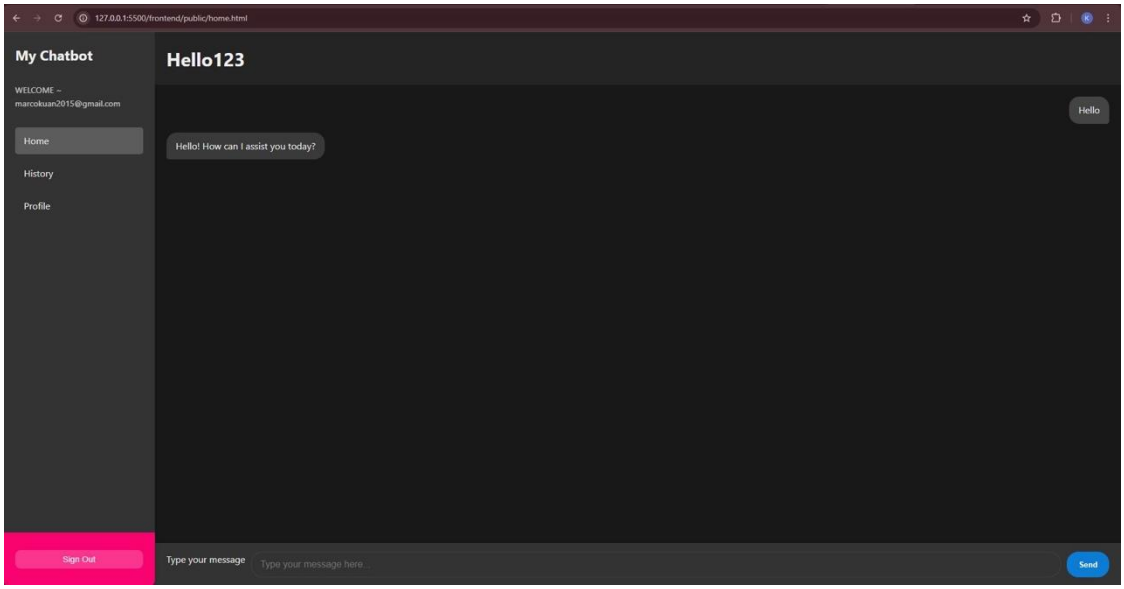
Meanwhile on the database:

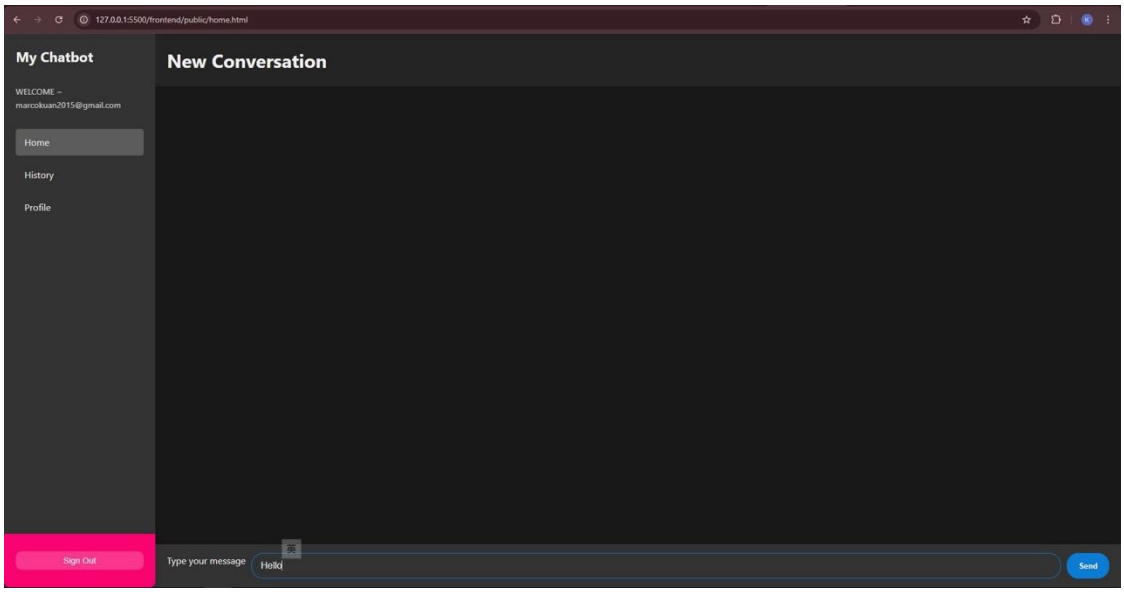


Theme change :

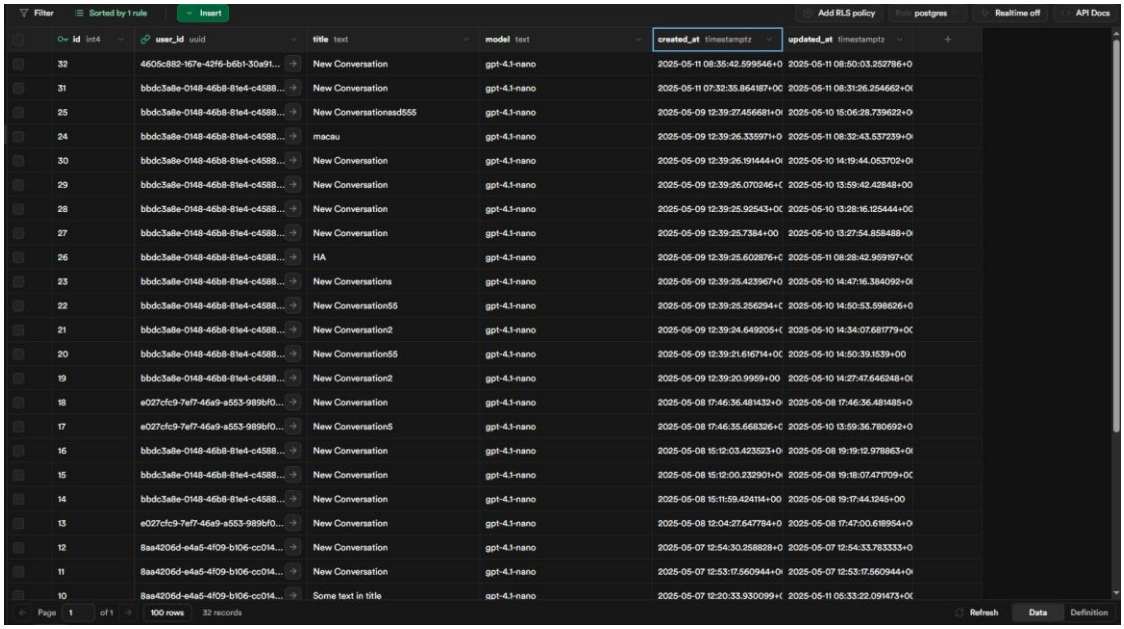


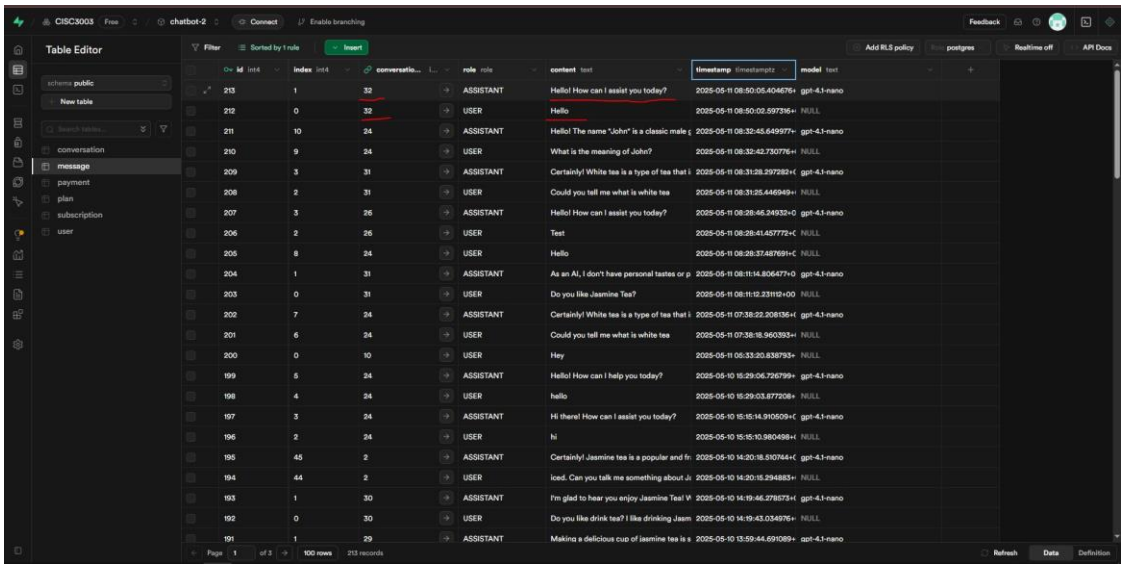
Start a New Conversation:



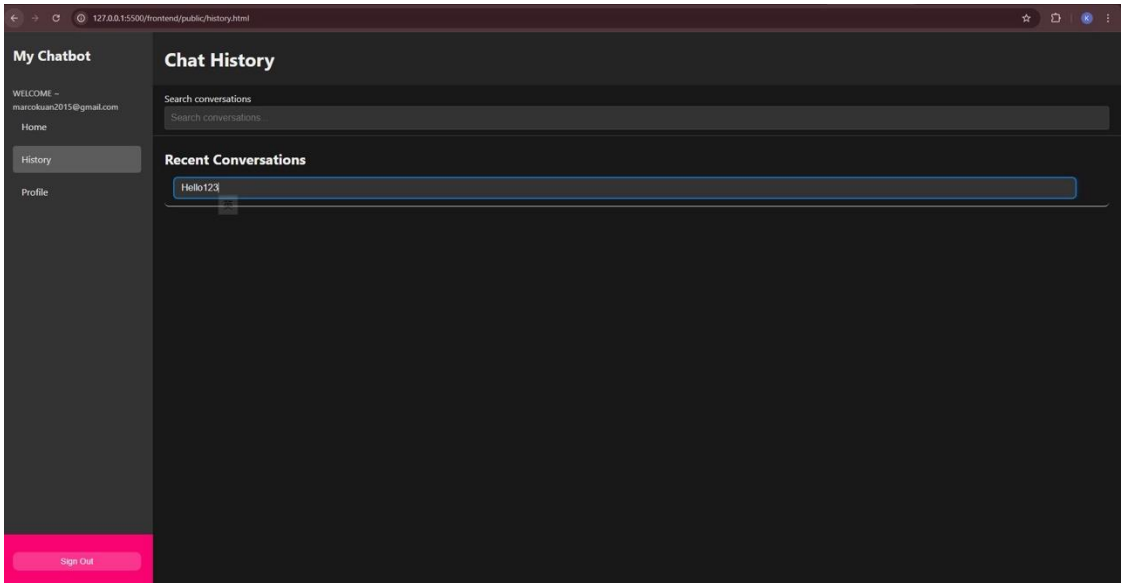


Meanwhile on the database:





Change the conversation Name:



Meanwhile on the database: