**Health Care Translation Web App Prototype**

Code Documentation

By: Sheikh Alyan

Date: November 2024

# Code Documentation:

## Overview:

The Translation Web App is designed to provide real-time translation services using both text input and speech recognition. It supports various languages and uses a socket communication model for efficient processing.

## Code Structure:

The app is built using Flask (backend), Socket.IO (real-time communication), and

JavaScript (front-end logic). Below is an outline of its structure:

* **Backend(app.py):**
* **Purpose:** Manages the server-side logic, including translation requests, WebSocket communication, and interaction with OpenAI API.
* **Key Components:**

1. **Flask app setup:** Initializes Flask, Socket.IO, and environment variables.
2. **Translation Function:** Uses OpenAI’s GPT-3.5-turbo model to provide medical translations while preserving accuracy and cultural appropriateness.
3. R**outes:**
   * /: Serves the front-end HTML.
   * WebSocket Event: translate\_text handles real-time translation requests.
4. **Security:**
   * API keys loaded securely using dotenv.
   * CORS settings configured to prevent unauthorized cross-origin requests.

* **Frontend:**

1. **Translation.js:**

**Purpose:** Sends translation requests to the server via WebSocket and handles responses.

**Key Features:**

* Formats translated text.
* Displays errors and manages the UI state (loading spinners, etc.).

**WebSocket Events:**

* Emit: translate\_text (sends text and language codes).
* Listen: translation\_response and translation\_error.

1. **Speech.js:**

**Purpose:** Provides speech-to-text and text-to-speech functionalities.

**Key Features:**

* Uses webkitSpeechRecognition for capturing spoken input.
* Converts translated text to speech using SpeechSynthesisUtterance.

1. **HTML TEMPLATE(INDEX.html):**

**Purpose:** The user interface with input fields for text, source, and target languages.

**Key Elements:**

1. Buttons for recording, translating, and speaking.
2. Drop-down menus for language selection.
3. Dynamic error and success message displays.
4. **CSS:**

**Purpose:** Styles the interface for better user experience

* **Supporting Files:**
* . env: Contains sensitive information like OPENAI\_API\_KEY and SECRET\_KEY.
* requirements.txt: Specifies Python dependencies (e.g., Flask, python-dotenv, flask-socketio).
* Vercel.json: For deployment purpose on versel
* **AI TOOL:**
* **OpenAI API:**

**Model:** gpt-3.5-turbo

R**ole:** Performs the core translation task, ensuring medical terminology accuracy, cultural appropriateness, and formality.

* **Security Considerations:**

1. **Environment Variables:**
   * Sensitive keys are stored securely using. env.
   * Ensure .env is excluded from version control (via .gitignore).
2. **CORS Settings:**
   * Configured for Socket.IO to prevent unauthorized cross-origin communication.
3. **Input Validation:**
   * Validates user input on the backend (e.g., checks for empty text) to prevent abuse or errors.
4. **Error Handling:**
   * Graceful handling of translation or connection errors to avoid application crashes.
5. **Rate Limiting:**
   * Consider adding rate limiting to protect the OpenAI API from excessive requests.

# NOTE:

The OpenAI API requires a paid subscription for efficient and unrestricted use. It cannot be utilized effectively during the free trial period.

To ensure secure and seamless functionality:

1. Purchase an OpenAI subscription to unlock full API capabilities.
2. Store the API key securely in a .env file.