

Healthcare Translation Web App with Generative AI

Link:

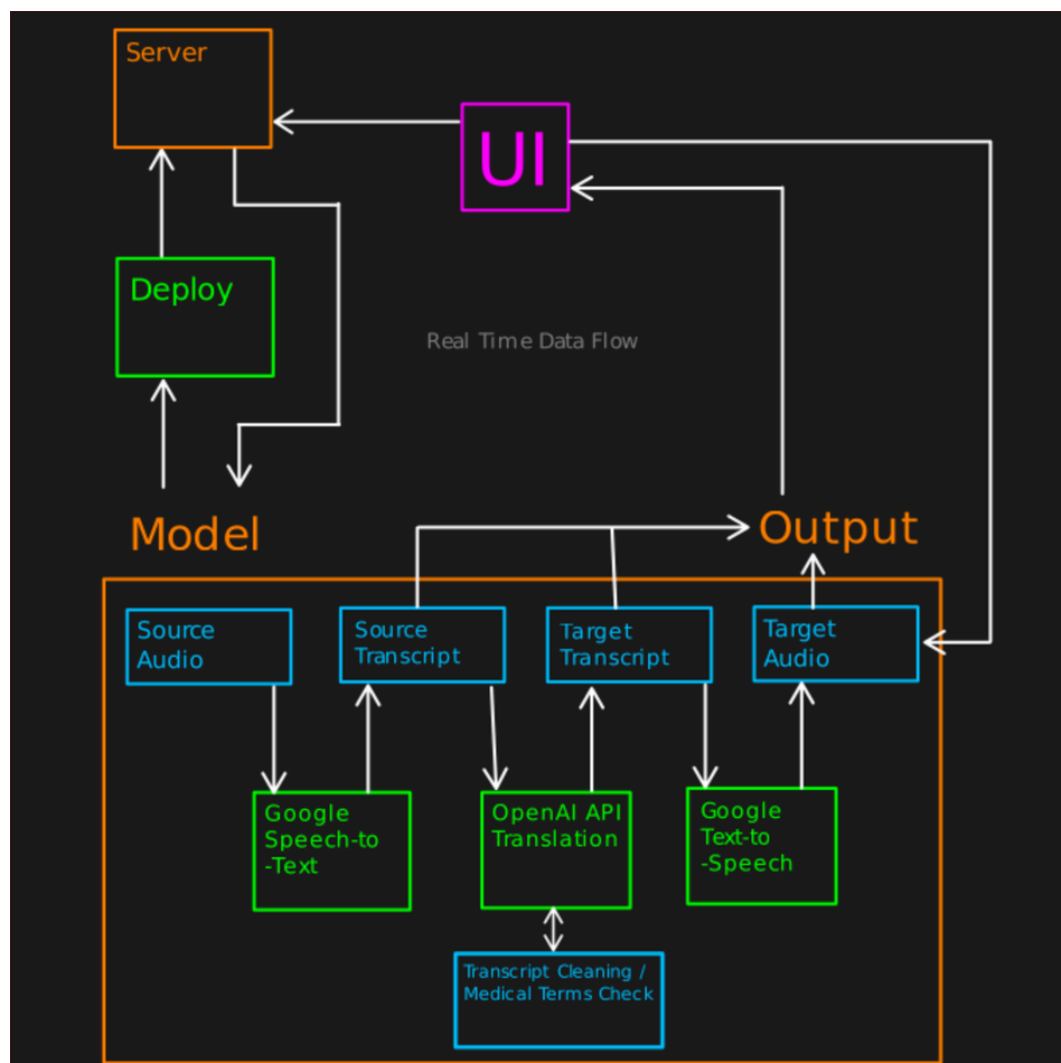
GitHub: [GitHub - ammar20112001/Generative-AI--Healthcare-Translation-Web-App](https://github.com/amm20112001/Generative-AI--Healthcare-Translation-Web-App)

Hosted Version: <https://huggingface.co/spaces/amm20112001/Medical-Translator>

Architecture

The Healthcare application has 2 important components. The data flow between front-end and back-end, and the other most important part, the AI model APIs implementation.

The data-flow: I built the data flow of this application with high focus on modularity and redundancy. That's why I started the project with a DFD diagram, which shows how the data is following within the application, from a high level.

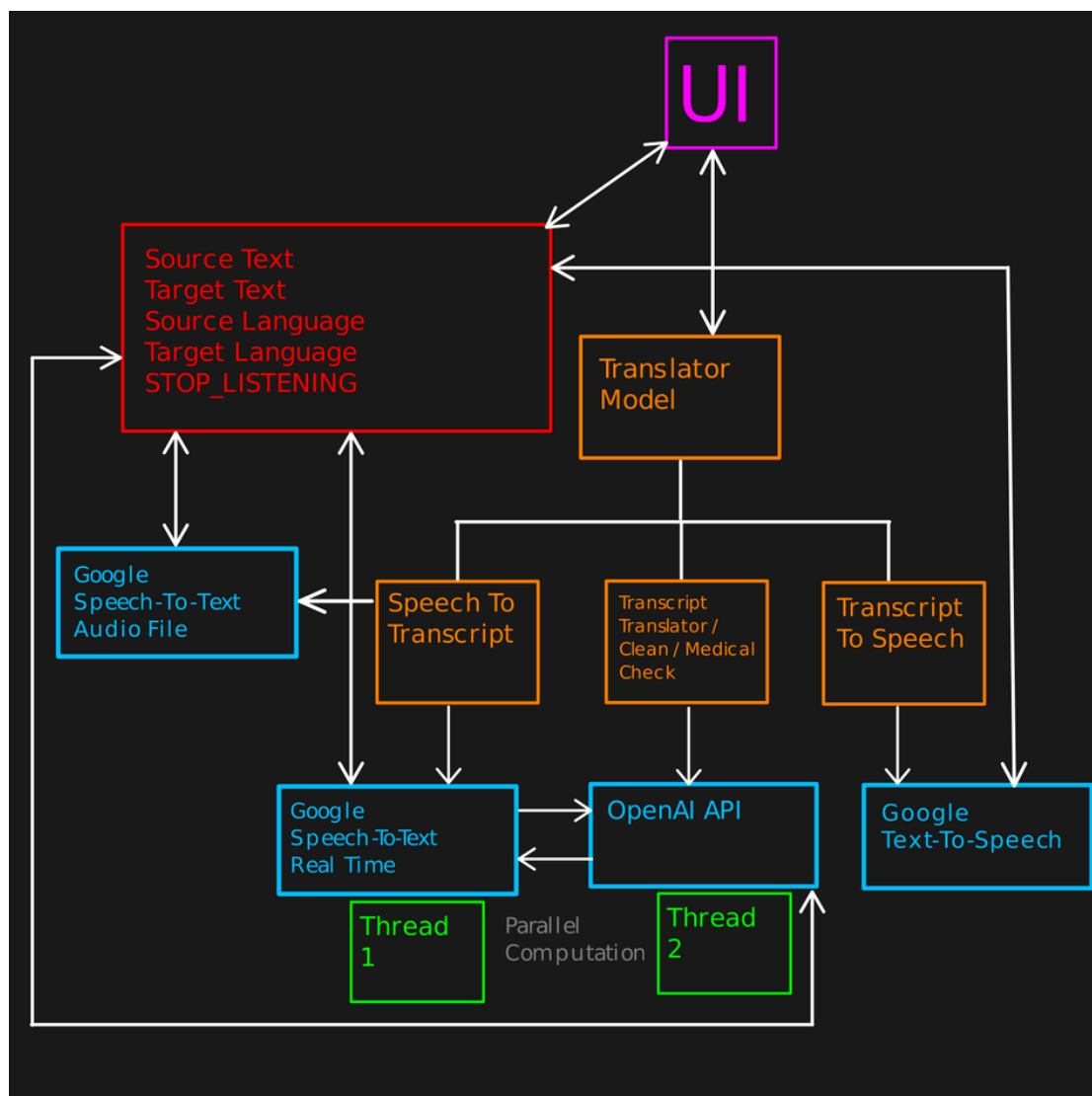


This diagram perfectly demonstrates, how the data flows within the application, and also how the UI interacts with the model. I built the entire program based on this architecture.

The most effective part of this is that the data flows independently and in a sequential manner. This is important for scalability and modularity.

Objects and APIs: The most important part of this program, which builds on top of the data flow, is how the objects interact with each other, and how the APIs are designed to make calls.

To make this process seamless, I created objects, and defined object data variables within the parent object, which inherits all other objects. All of the other objects offer methods of API calls to the services that are responsible to run our application, for transcription and translation purposes.



This object diagram clearly shows how the relationship is created between different objects. The inheritance was necessary, so that every object shares the same data. This was also, necessary to maintain chains between theses models, that run sequentially.

A benefit of using this architecture was that the UI interacted with the object data and parent model only. I believe this is important to not introduce complexity later in the program.

Threading: Threading is a process to run a program, independent of another program, simultaneously. To implement real time transcription and real time translation, it was very important to thread these 2 functions and let them run independently.

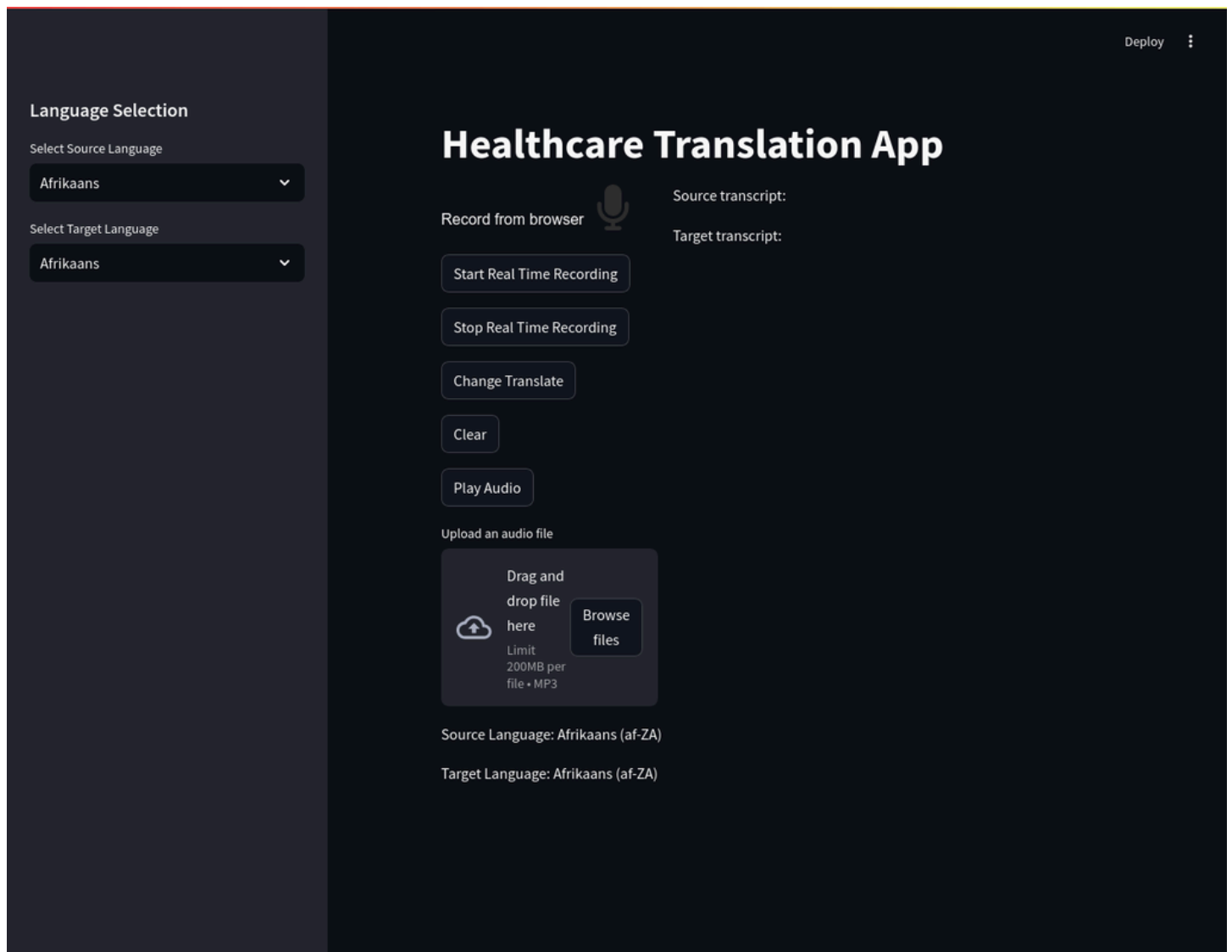
Application deployment:

The application has been deployed on huggingface space and can be visited [here](#).

Note: Due to cloud server hosting, we do not have access to server's microphone, therefore, real time functionality cannot be tested. For accessing every feature of the program, create a local environment and run locally. The program is already configured to build, all you need is to `pip install requirements.txt` and run streamlit with `streamlit run app.py`.

You will also have to setup ENV variables. You can find all implementation details on the GitHub page [here](#).

Here's the look of our web application after implementation:



The screenshot displays the 'Healthcare Translation App' interface. On the left, a sidebar titled 'Language Selection' contains two dropdown menus, both set to 'Afrikaans'. The main area features the app title 'Healthcare Translation App' and a 'Deploy' button in the top right. Below the title, there are sections for 'Record from browser' (with a microphone icon) and 'Upload an audio file' (with a cloud icon). The 'Record from browser' section includes buttons for 'Start Real Time Recording', 'Stop Real Time Recording', 'Change Translate', 'Clear', and 'Play Audio'. The 'Upload an audio file' section includes a 'Browse files' button. Below these sections, the source and target languages are both set to 'Afrikaans (af-ZA)'. The interface is dark-themed with light-colored text and buttons.

Deploy

Healthcare Translation App

Record from browser

Source transcript:

Target transcript:

Start Real Time Recording

Stop Real Time Recording

Change Translate

Clear

Play Audio

Upload an audio file

Drag and drop file here

Limit 200MB per file • MP3

Browse files

Source Language: Afrikaans (af-ZA)

Target Language: Afrikaans (af-ZA)

Technologies used:

APIs

- Google Speech-To-Text: Audio Transcription
- OpenAI API: Transcript Translation / Cleaning / Medical Terms Check
- Google Text-To-Speech: Transcription to Audio

Web Service

- Streamlit

Hosting Service

- Huggingface Space
- Streamlit Cloud

Version Control

- Git
- GitHub

Real-time Transcription/Translation

- Multi-threading

USER GUIDE:

1. The Healthcare Translation App allows you to:
2. Select source and target languages.
3. Record audio in real time and receive transcription and translation results, also in real time.
4. Upload audio files for transcription and translation.
5. Re-translate as per your need.
6. Play back the translated audio.
7. Clear transcripts and restart.

Features:

Language Selection:

- Select Source Language: Choose the language you want to translate from. For example, "Dutch (Belgium)".
- Select Target Language: Choose the language to translate to.

Start Real-Time Recording:

- Click 'Start Real Time Recording' button to begin capturing audio. The app will transcribe the spoken words into text in the Source Transcript area.
- Stop Real-Time Recording:
- Click 'Stop Real Time Recording' button to end the recording.

Change Translation:

- If you're not comfortable with the translated text, press 'Change Translation' button to make a new translation.

Clear:

- Use the Clear button to reset transcripts and prepare for a new session.

Play Audio:

- Click on 'Play Audio' anytime for the program to convert and run audio of the translated text.

Upload an Audio File:

- Drag and drop or browse to upload an audio file (supports up to 200MB in MP3 format, and up to 60 seconds in length). The app will automatically process the audio and display the transcript in the selected languages.