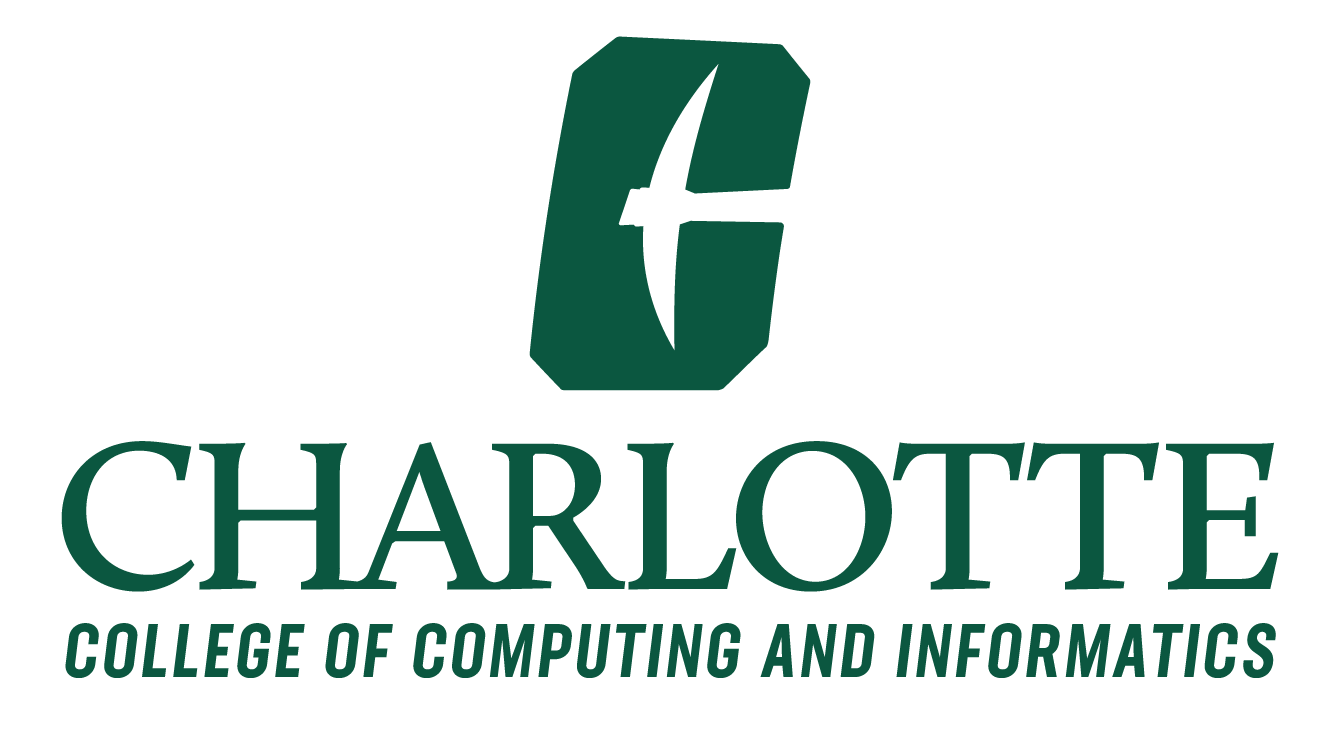
**ITCS 6166 - Computer Communications and Networks**

****

**Final Report**

**Text to Voice Conversion**

**Group 18:**

**Sumanth Dodda**

**Hemanth Kumar Anumolu**

**Ramya Jampani**

**Bhavya Sri Kili**

**Abstract:**

This project aims to provide developers with a high-performance, Text-to-Voice (TTS) solution that allows them to create high-quality voices which can be used across a wide range of applications. By leveraging deep learning models, Speaker Encoder, and Vocoder models, developers can easily generate voices with fast and efficient model training, testing and using the models, and a modular code base for easy customization. The architecture of the application is designed to use a front-end web application as the user interface with the back-end TTS API hosted on a cloud-based provider. As an additional feature we are using the googletrans library to translate the text to the language model input language that we are using. Googletrans library also has a feature to detect the laguage of the entered text. The project will go through several iterations, from familiarizing with the TTS library and testing the web application, deploying the project, and finally documenting and presenting the project. The project's success is demonstrated through the deployment of the TTS API on Digital Ocean and hosting it on a domain.

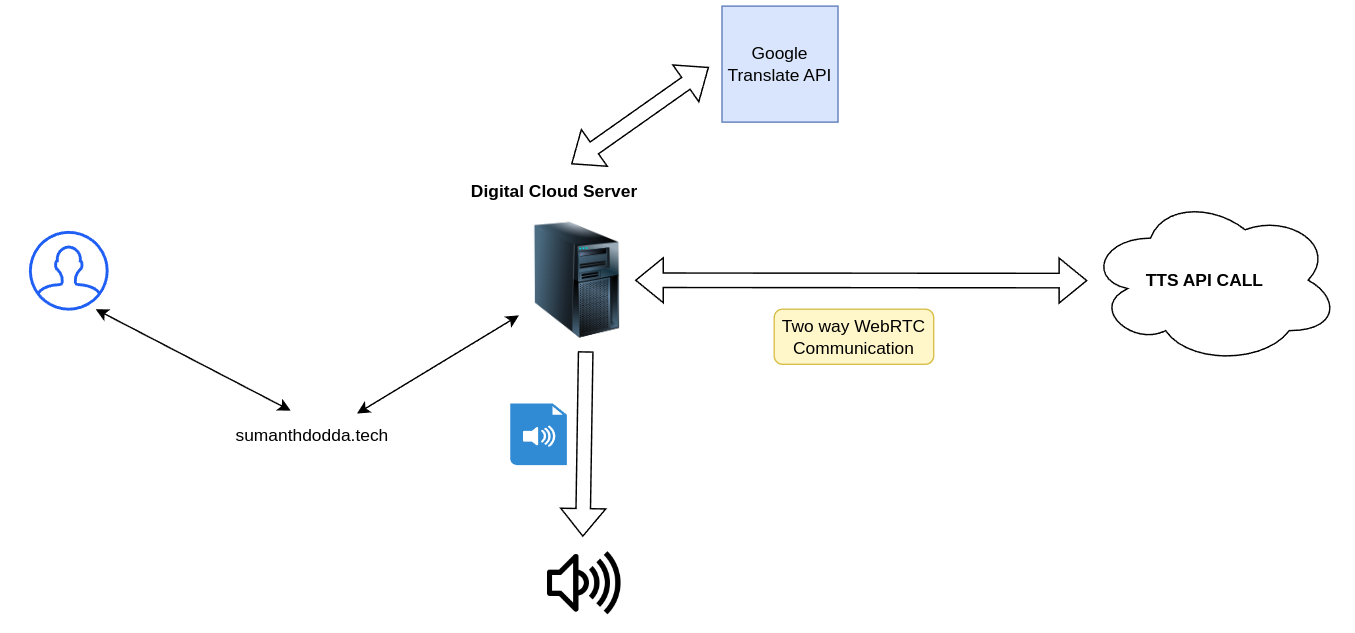
**Introduction:**

The main goal of Coqui-TTS is to provide high-quality speech that can be used for a wide range of applications, such as voice assistants. It is designed to support multiple languages, and can even be trained on new data sets to make it perform better for specific purposes.

The technology behind TTS is based on two models, Tacotron 2 and WaveGlow, these models use deep neural networks to convert written text into sound. They were trained on a large number of recordings of human speech, allowing them to produce high-quality synthetic speech.

Coqui-TTS is written in Python and uses the PyTorch deep learning library, The tool also comes with pre-trained models for several languages, including English, French, German, Spanish, and other languages. In general the TTS library supports 20+ languages and the pre-trained models are available through the TTS library api. These models can be used as-is to synthesize speech from text, or they can be fine-tuned on new data sets to make them more accurate for specific purposes, it is open-source in nature and the community is active.

**System Architecture:**



**Implementation:  
System configuration and project setup: (Ubuntu or any Debian distro)**

**Step 1: Update your System to the latest**

*sudo apt update -y && sudo apt full-upgrade -y*

**Step 2: Download the supporting python Version for TTS (Python>=3.7 <3.11)**

*wget https://www.python.org/ftp/python/3.10.0/Python-3.10.0.tgz*

**Step 3: Unzip it and Change into the directory**

*tar zxvf Python-3.10.0.tgz && cd Python-3.10.0/*

**Step 4: Install Python 3.10**

*./configure --enable-optimizations*

**Step 5: Clone the Project and Change into the directory**

*git clone* [*https://github.com/sumo10451/group14.git*](https://github.com/sumo10451/group14.git) *&& cd group14/*

**Step 6: Install espeak manually**

*sudo apt install espeak-ng -y*

**Step 7: Install Python Dependencies for TTS, (streamlit included)**

*python3.10 -m pip install -r requirements.txt*

**Step 8: Run the Application:**

*streamlit run streamlit\_audio\_converter.py --server-port 80*

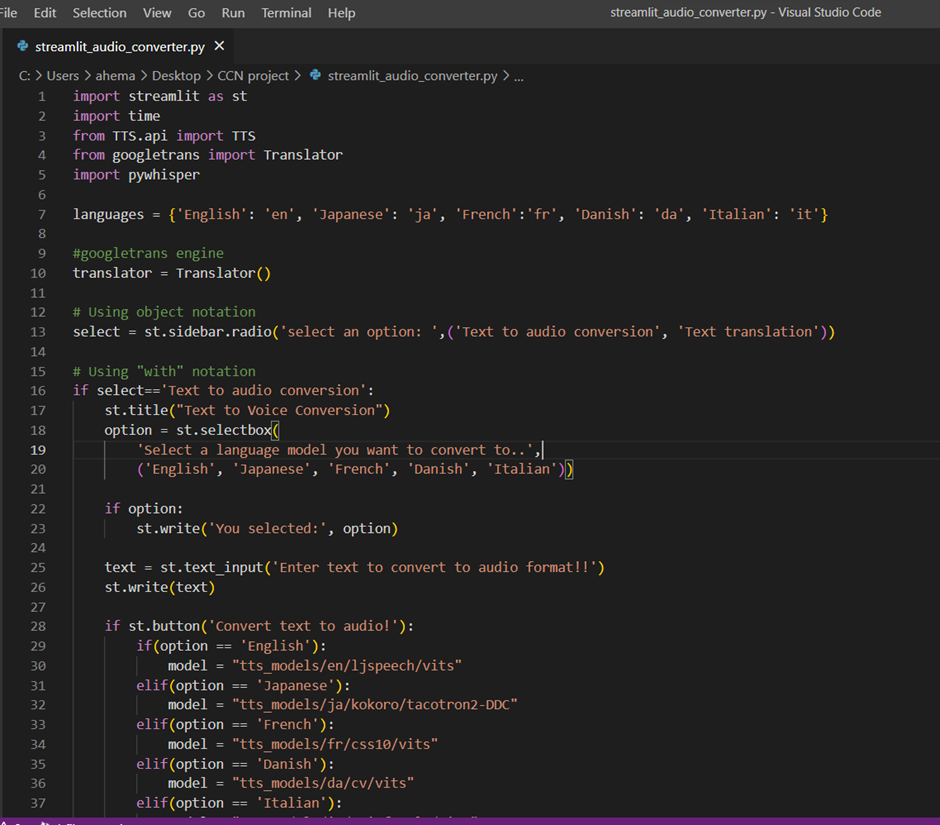
**Step 9: Access the Application:**

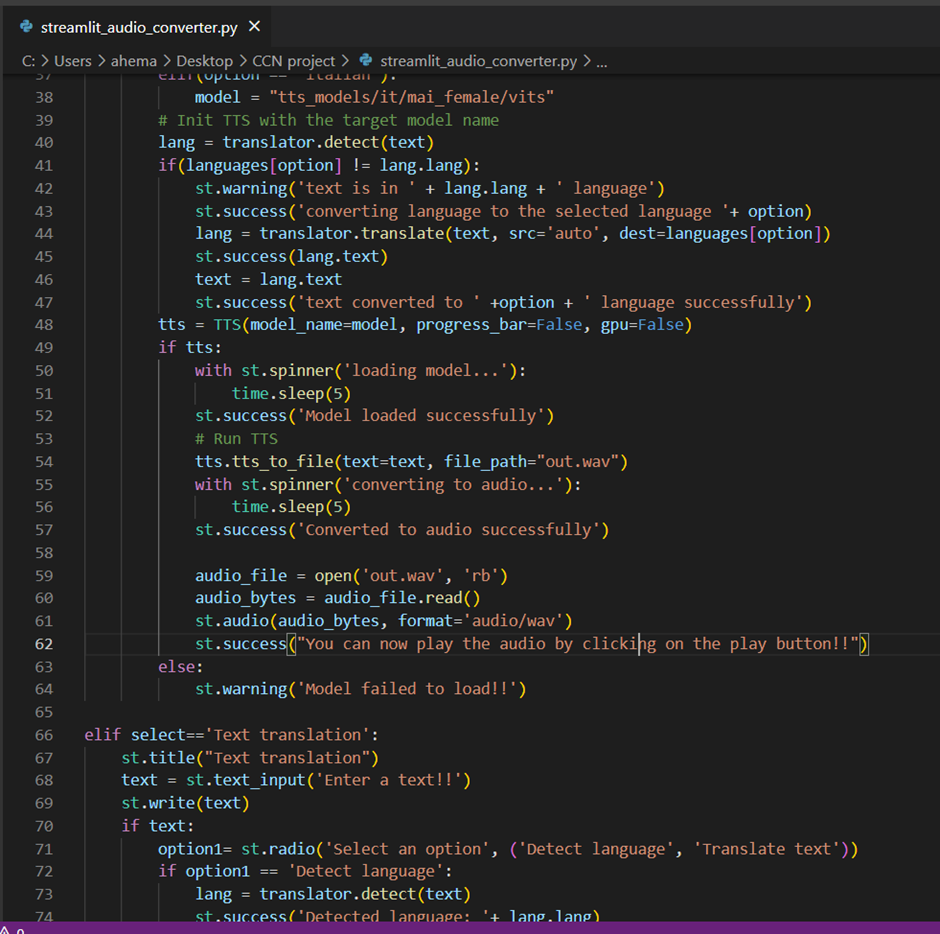
*Locally:* [*http://127.0.0.1*](http://127.0.0.1)

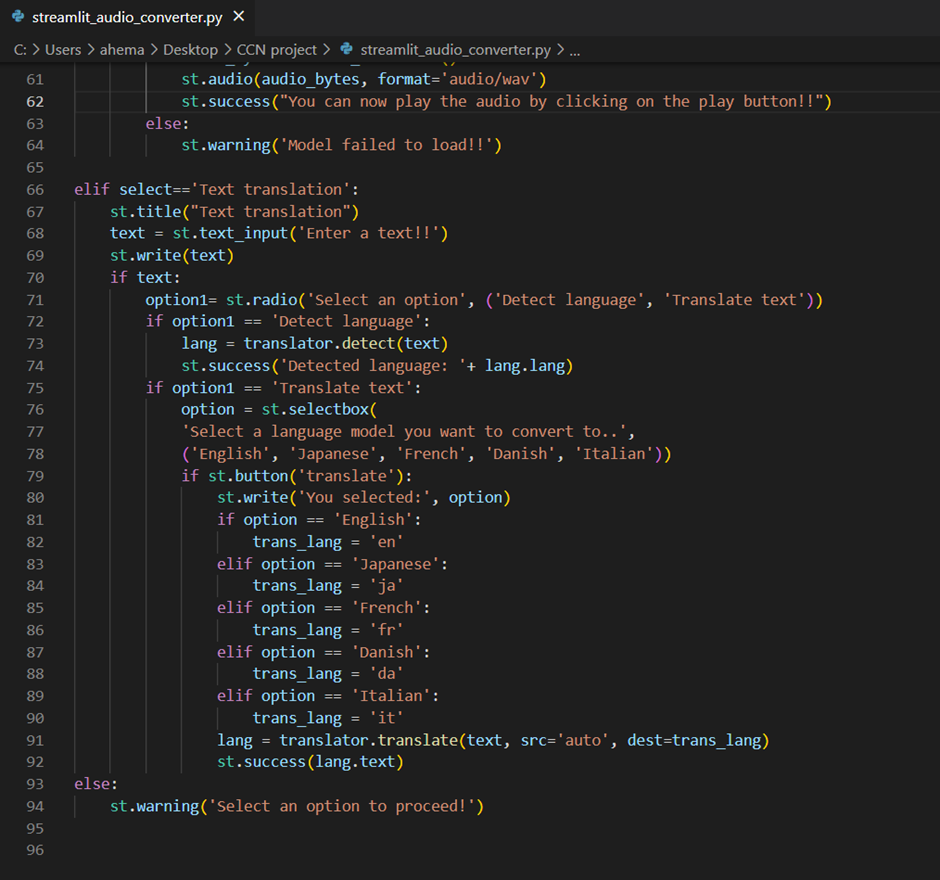
*Cloud Based Deployment: http://<public ip>*

* *\*\*Note: You need to add inbound port 80 to firewall rules\*\**

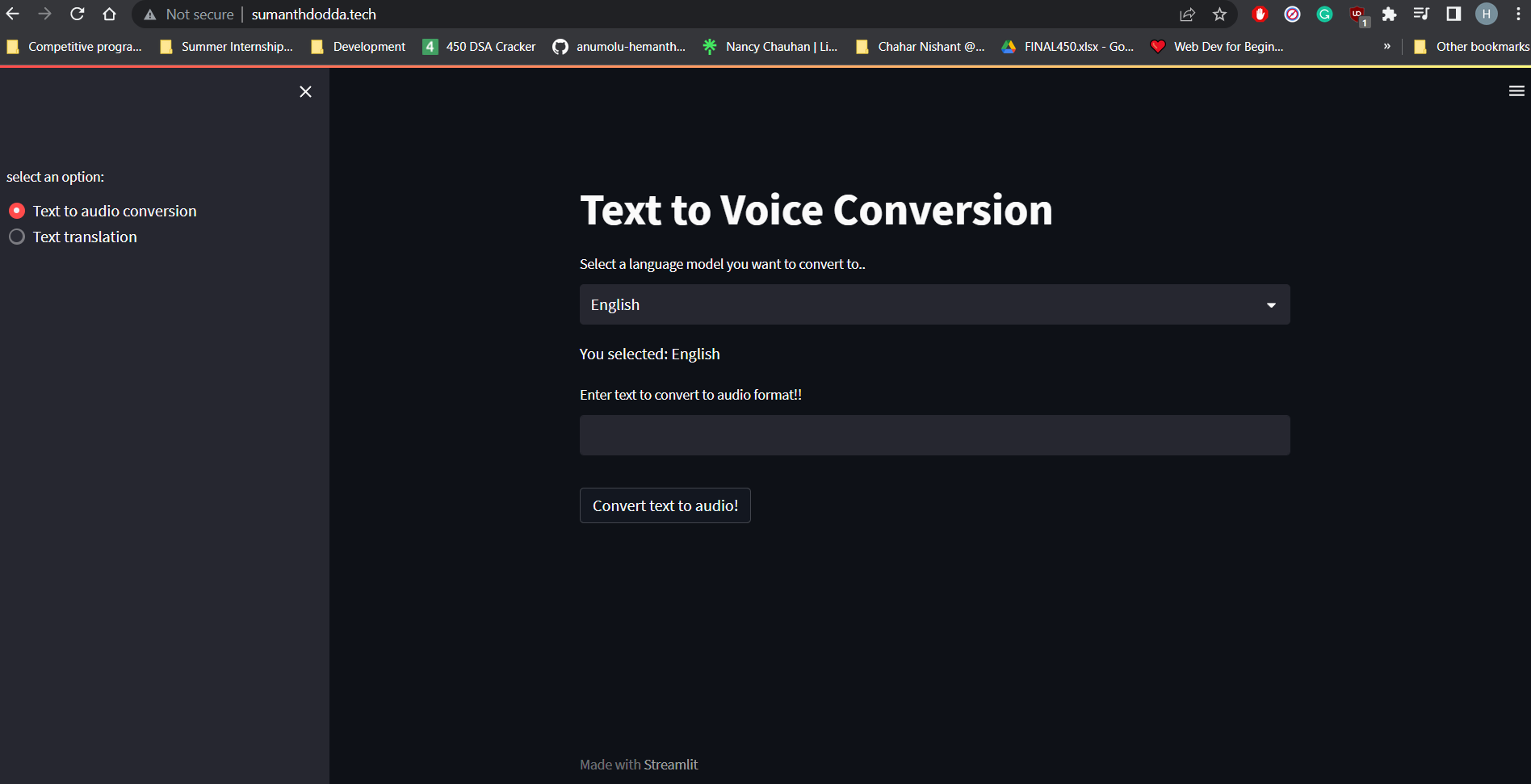
**Code Snippets:**

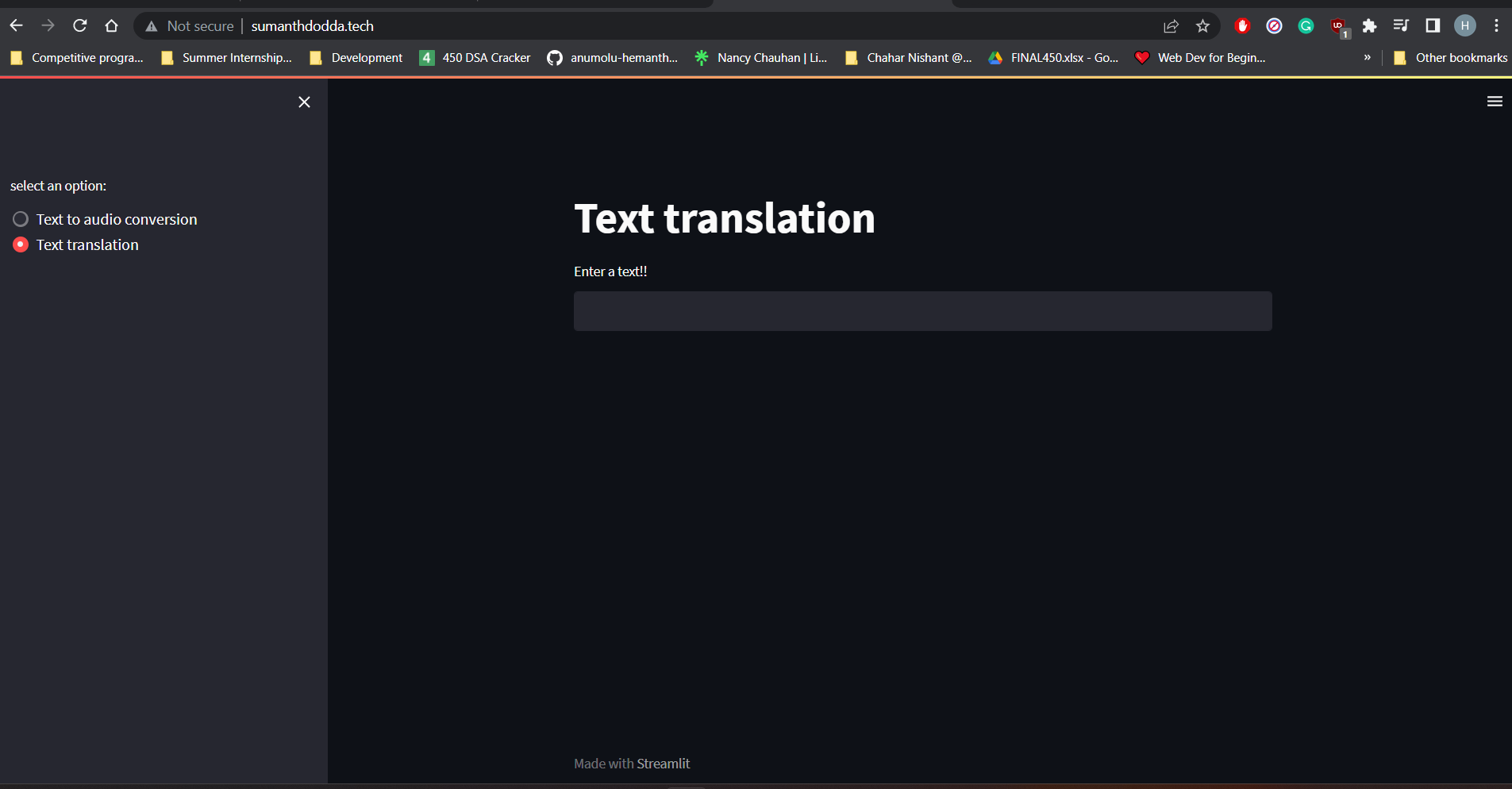


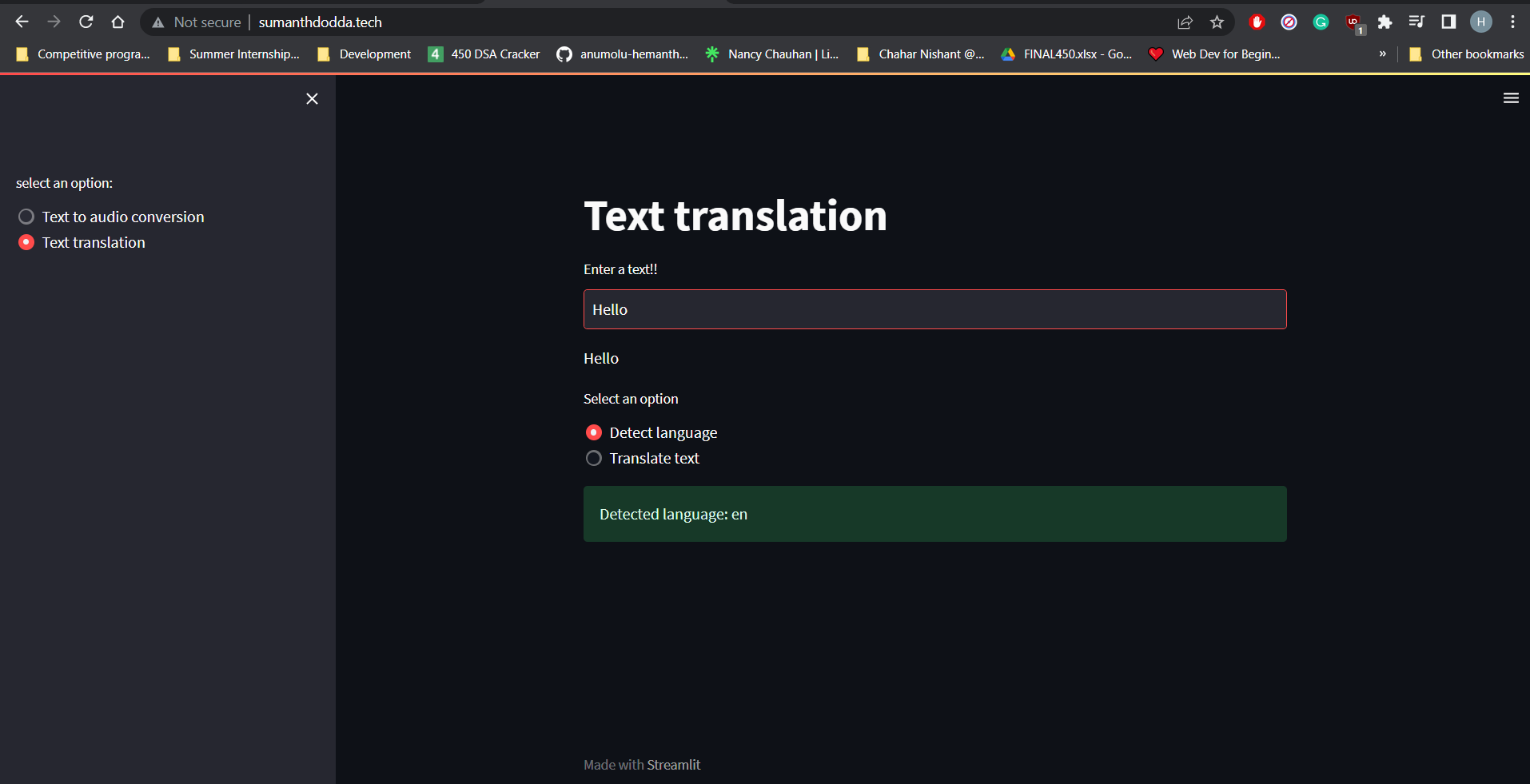


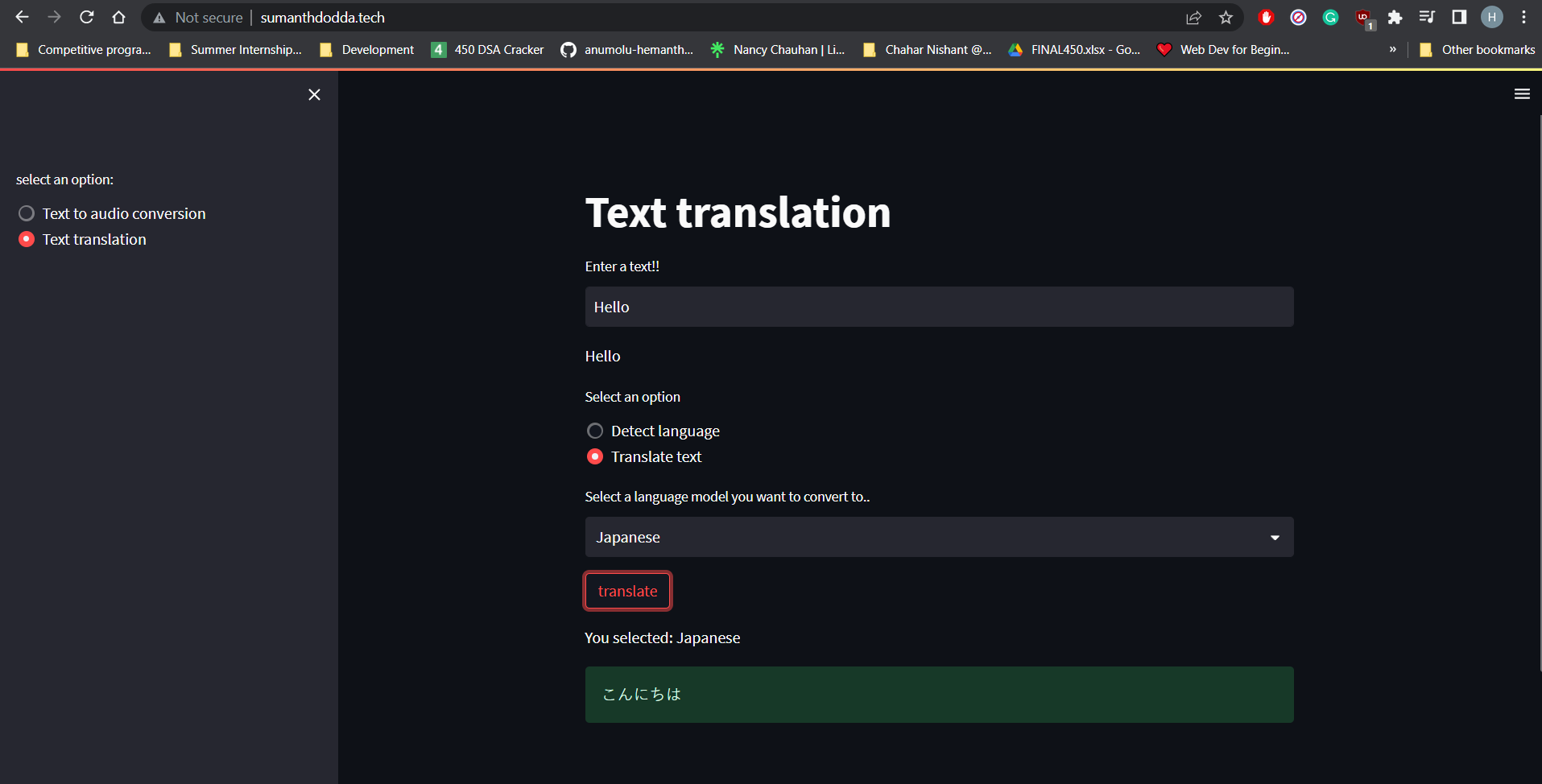


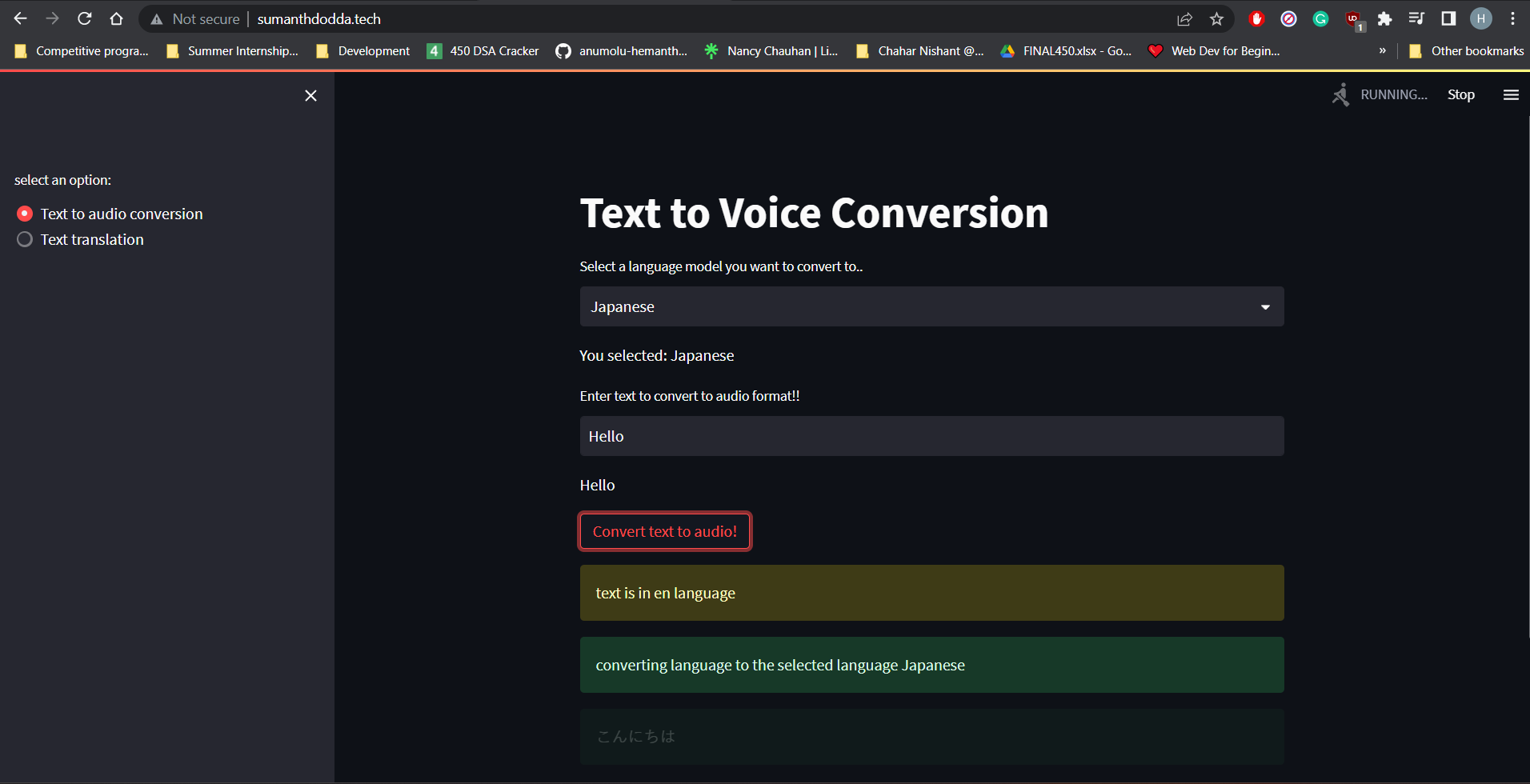
**Project Demo:**

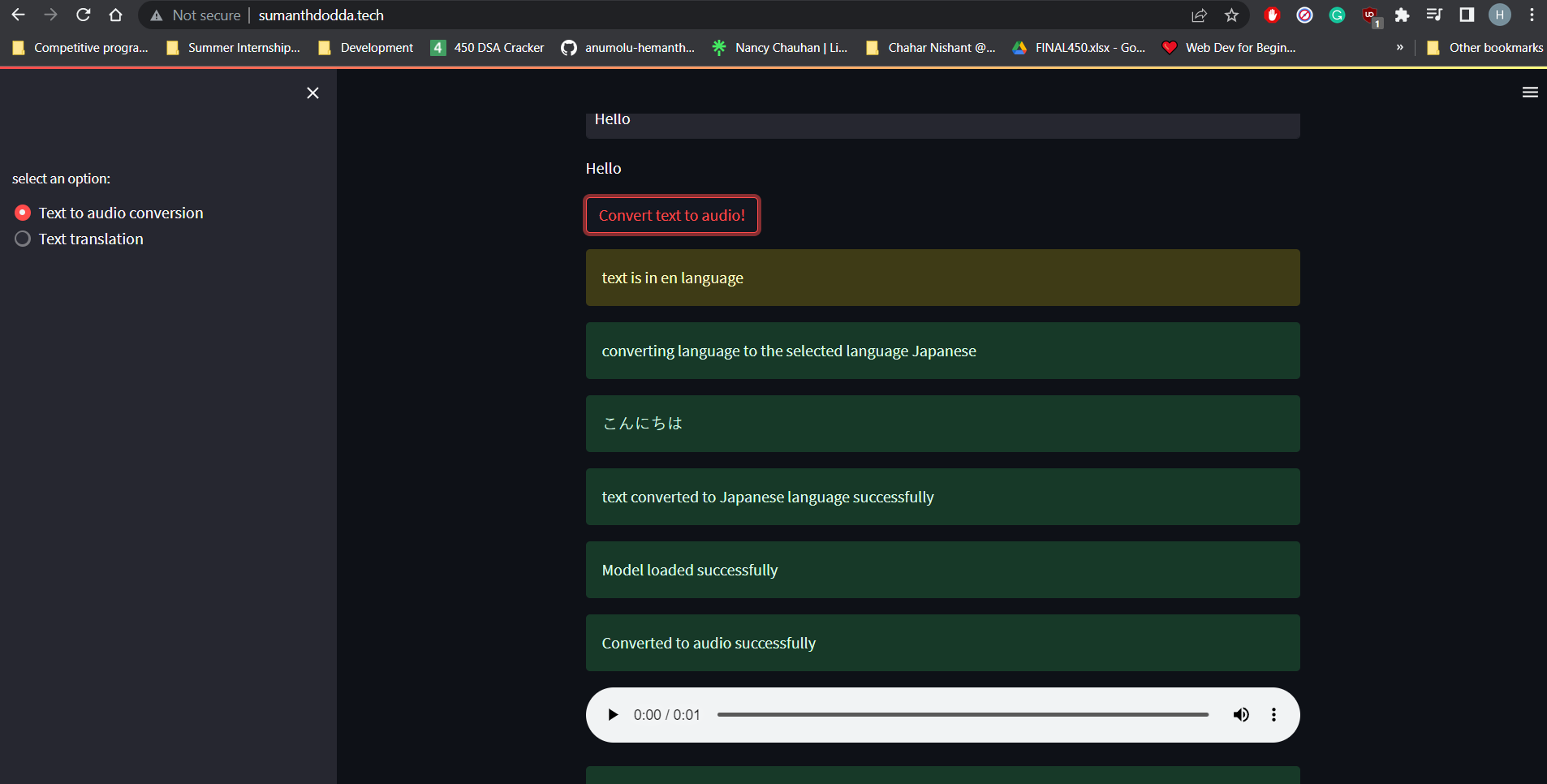
****

****

****

****

****

****

**Challenges Faced:**

* Using different python version than the host installed python version and manually calling specific python version.
* Integrating the TTS model with the web application and deploying it on a cloud-based provider.

**Conclusion:**

The project's success depends on the careful execution of several iterations. In the first iteration, the developers familiarize themselves with the TTS library, its input and output formats, and the required software packages. The second iteration involves working with the model and testing it for any issues. If necessary, the developers make changes to the model to improve its accuracy. The third iteration focuses on developing the web application and integrating the text-to-voice model with the application. The developers test the application for any issues and fix them if necessary. The fourth iteration involves creating a deployment pipeline and deploying the project on a cloud-based provider. The developers test the project end-to-end, fix any issues if found, and create a basic webpage to post input to the server backend API and display the output given.

The team had to understand how to pass on manual input to the API using a static webpage and display the output on a webpage to make it accessible over the internet. Additionally, the team had to figure out the best technologies, tools, pipelines, and programming languages to build the application.

The project's architecture was designed to use a front-end web application as the user interface, with the back-end TTS API hosted on a cloud-based provider. This design provides flexibility making it easy to integrate the TTS API into different applications. The team chose to host the TTS API on Digital Ocean and host it on a domain to demonstrate the project's success.

In conclusion, the TTS AI project is a valuable tool for developers who want to create high-quality voices for their applications. The project's success depends on careful execution of the different iterations, including familiarizing with the TTS library, testing the model, developing the web application, and creating a deployment pipeline.