Translate App

Group 7:

Amir Patel

Caleb Okafor

I. Alejandro Acevedo Aguilera

Kaleab Dubale

Que Thi Nguyen

Prof. Ronaldo Felipe

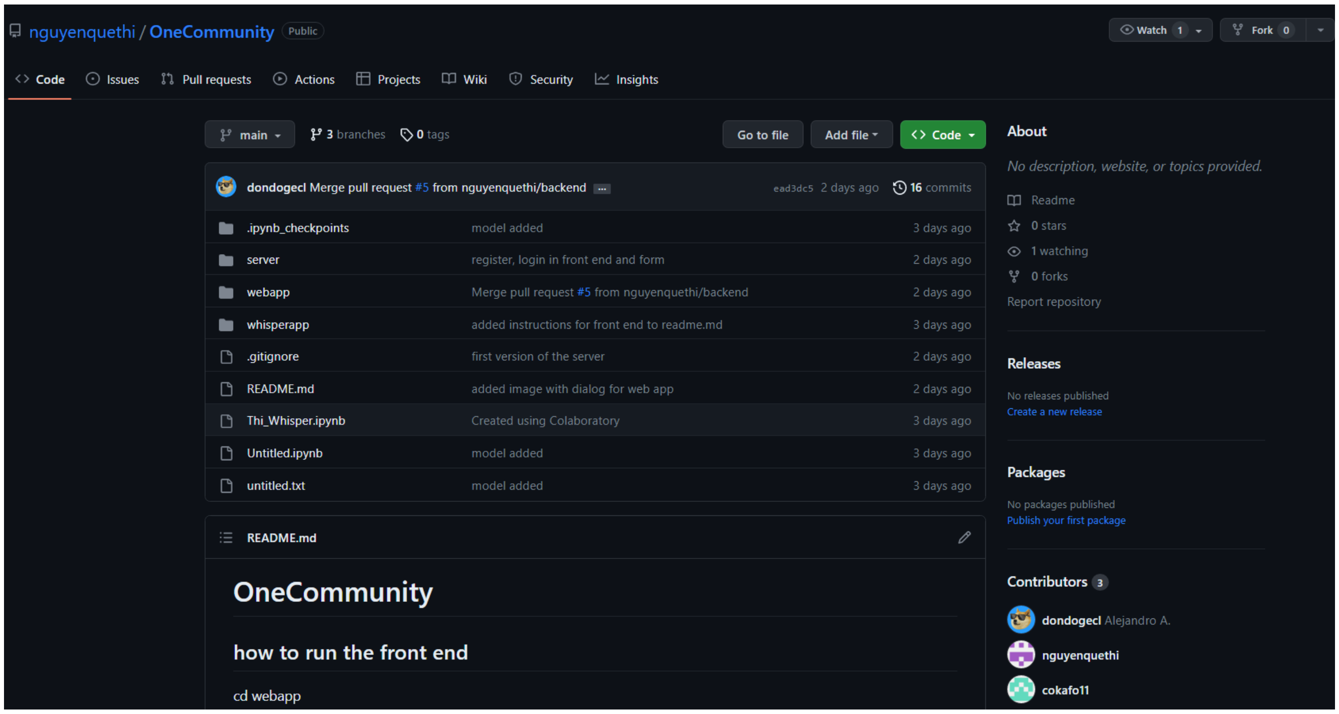
April 21, 2023.

Project Overview:

The ***Translate App*** is an original concept conceived by members of this group in response to probable scenarios in which individuals may need to translate audio inputs (messages, files, voice recordings) from one language to another. The application incorporates the use of existing the OpenAI Whisper model and ChatGPT turbo for the conversion between different languages. Using few common languages, the software is built to receive audio data (speech), process it into text, use a backend API to translate it into the preferred language of the user of the application, send it back to the API and convert the text to speech for the end user. For the purpose of this demonstration, three languages have been chosen namely, English, Spanish, and Vietnamese.

Image is the logo featuring our original character Hana-chan (in the future the desing may change)

Github repository of the project:


nguyenquethi / OneCommunity Public

Watch 1 · Fork 0

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights

main · 3 branches · 0 tags · Go to file · Add file · Code

dondogecl Merge pull request #5 from nguyenquethi/backend ... ead3dc5 2 days ago · 16 commits

.ipynb_checkpoints	model added	3 days ago
server	register, login in front end and form	2 days ago
webapp	Merge pull request #5 from nguyenquethi/backend	2 days ago
whisperapp	added instructions for front end to readme.md	3 days ago
.gitignore	first version of the server	2 days ago
README.md	added image with dialog for web app	2 days ago
Thi_Whisper.ipynb	Created using Colaboratory	3 days ago
Untitled.ipynb	model added	3 days ago
untitled.txt	model added	3 days ago

README.md

OneCommunity

how to run the front end

cd webapp

About

No description, website, or topics provided.

Readme
0 stars
1 watching
0 forks
Report repository

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

dondogecl Alejandro A.
nguyenquethi
cokafo11

Image of the github repository: https://github.com/nguyenquethi/OneCommunity

Requirements:

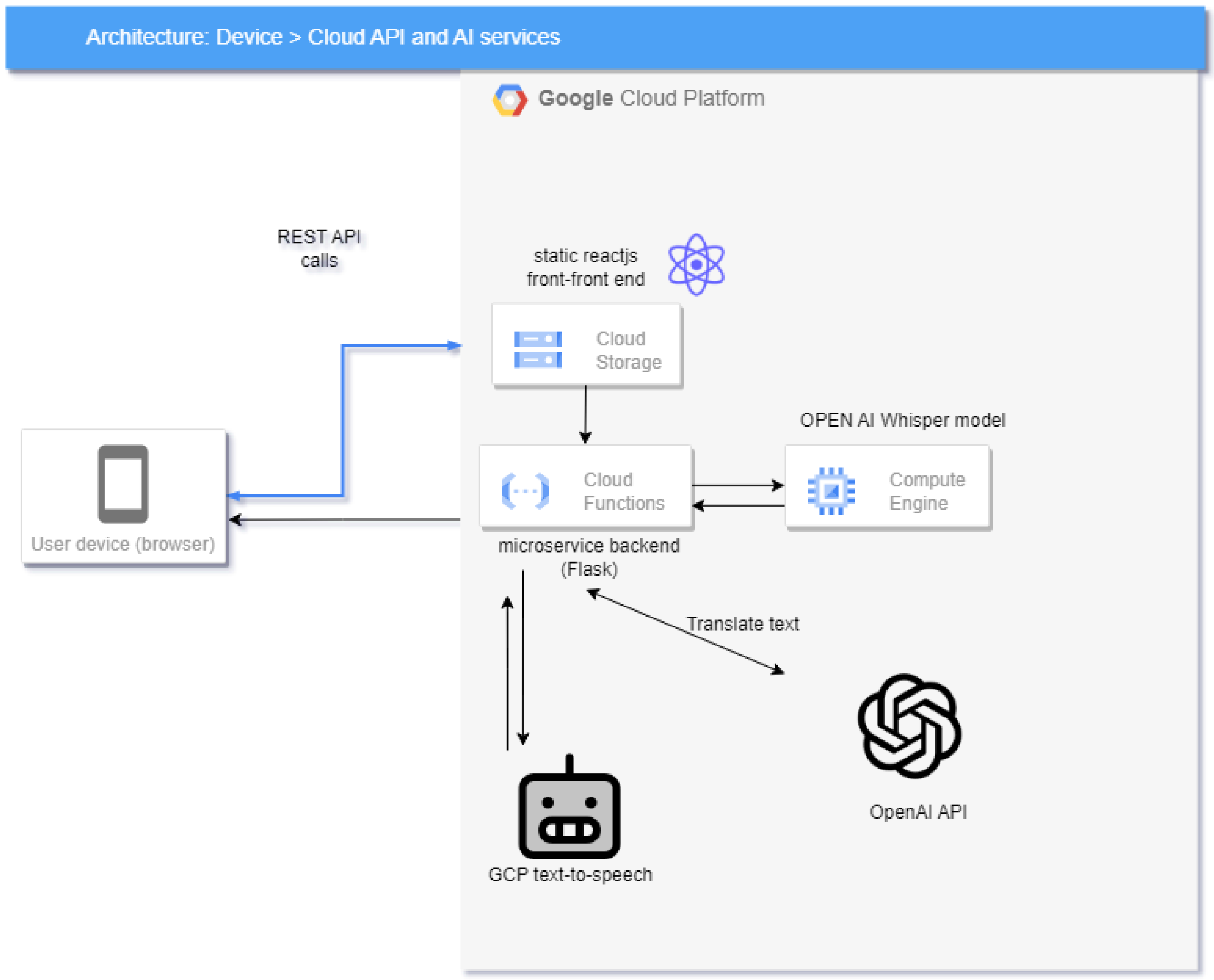
The following are the agreed-upon requirements for the *Translate App:*

1. Conversion among minimum of 5 languages.
2. Functional and accessible GUI for the front-end.
3. Ability to record audio from the front-end and have it converted to another language.
4. Ability to convert audio files and have it converted to another language.
5. Up-and-running back-end server.
6. Implement user registration to manage the usage per user and avoid high costs

Technical Specification

1. Functional backend server developed using Flask (Python), Google Cloud Storage, and Node JS.
2. Front-end developed using ReactJS and Bootstrap 5.
3. Conversion between languages using ChatGPT turbo.
4. Use of OpenAI’s Whisper model to convert speech to text.
5. Use of Google Cloud text to Speech API services (to be changed in the future for a TF lite model or similar)

Project Architecture



Running the Front-End on Your Local Host

The folders for the front-end can be obtained from the following link: https://github.com/nguyenquethi/OneCommunity/tree/front

Type in the following commands in sequence in your terminal to run the application on your localhost:

First time only (installation of packages in local server):

```
cd webapp
npm install
npm install nodemon –save-dev
npm run dev
```

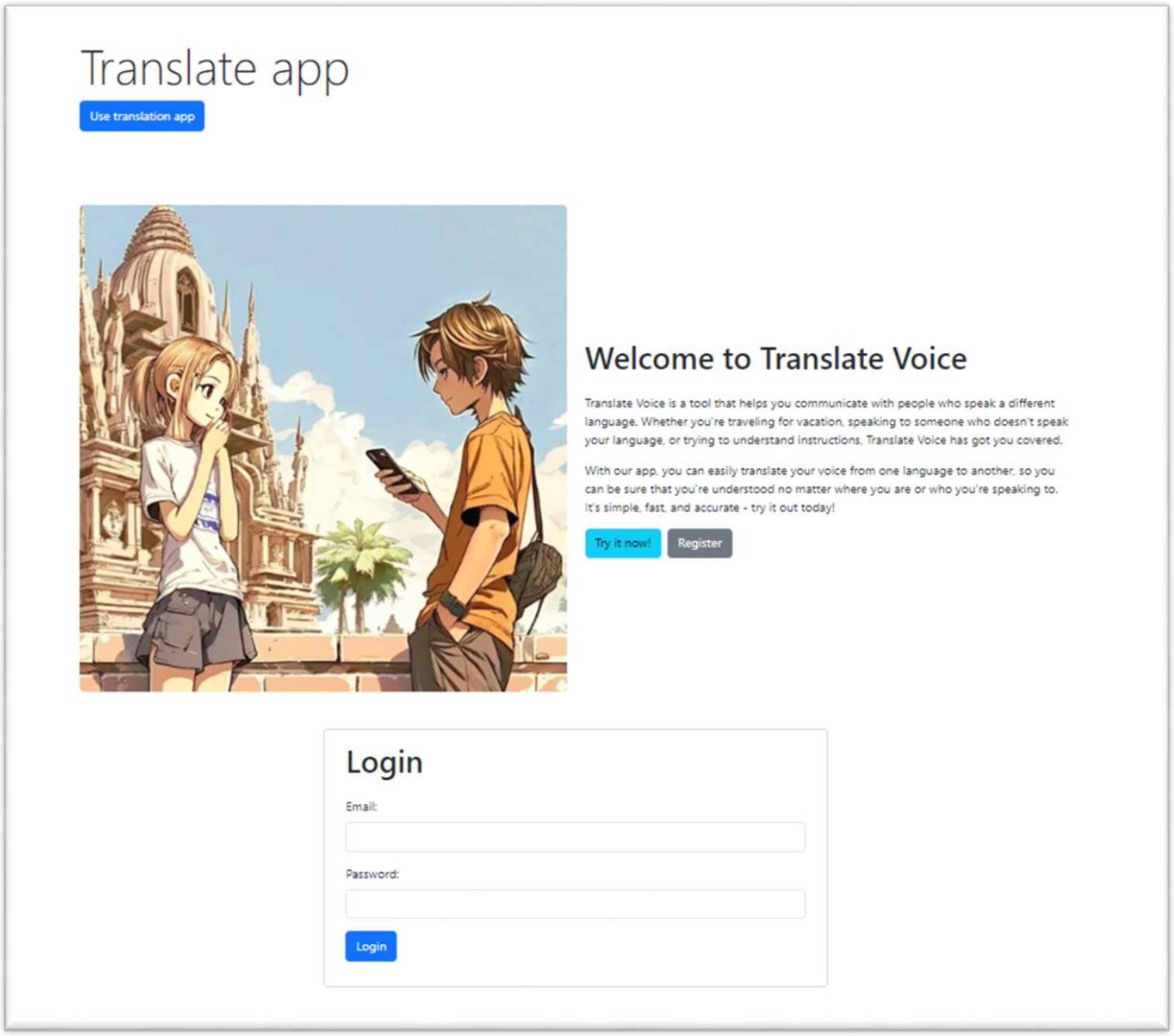
After it has been installed, only run:

```
cd webapp
npm run dev
```

The app will be available at http://127.0.0.1:3001

Breakdown of Front-end Functionality.

As earlier mentioned, the user of the application should have the ability to record themselves in a particular language, and have their words converted to another language, as illustrated below:



Translate app

Welcome to Translate Voice

Translate Voice is a tool that helps you communicate with people who speak a different language. Whether you're traveling for vacation, speaking to someone who doesn't speak your language, or trying to understand instructions, Translate Voice has got you covered.

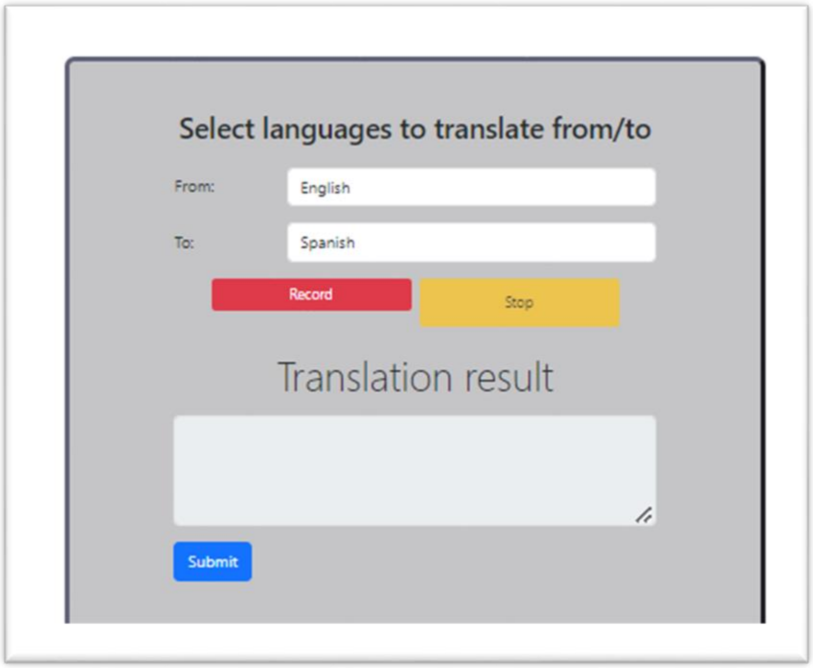
With our app, you can easily translate your voice from one language to another, so you can be sure that you're understood no matter where you are or who you're speaking to. It's simple, fast, and accurate - try it out today!

Login

Email:

Password:

- By login in using a valid username and password, you will be directed to the page below that allows you to convert audio recordings between different languages:



- ➢ The page above allows you to select languages that you want to convert from/to
  - It also allows you to record your voice by clicking on the record button.
  - By default we will record 10 seconds of audio and it will stop (in the future we will allow less or more by using the stop button)
  - The "Translation Result" shows your audio recording in the language you have chosen to convert into.

Running the backend

The folders for the backend server can be obtained from: https://github.com/nguyenquethi/OneCommunity/tree/backend

The first time (installation) is done by running:

```
cd  server
python install –r requirements.txt
```

After installing dependencies

```
cd server
python app.py
```

The server API will be available at: 127.0.0.1:5000/

Running the example Whisper code (ML)

We used Jupyter Notebooks that will be converted into Python scripts, to run the examples, we first need to install the dependencies:

First time:

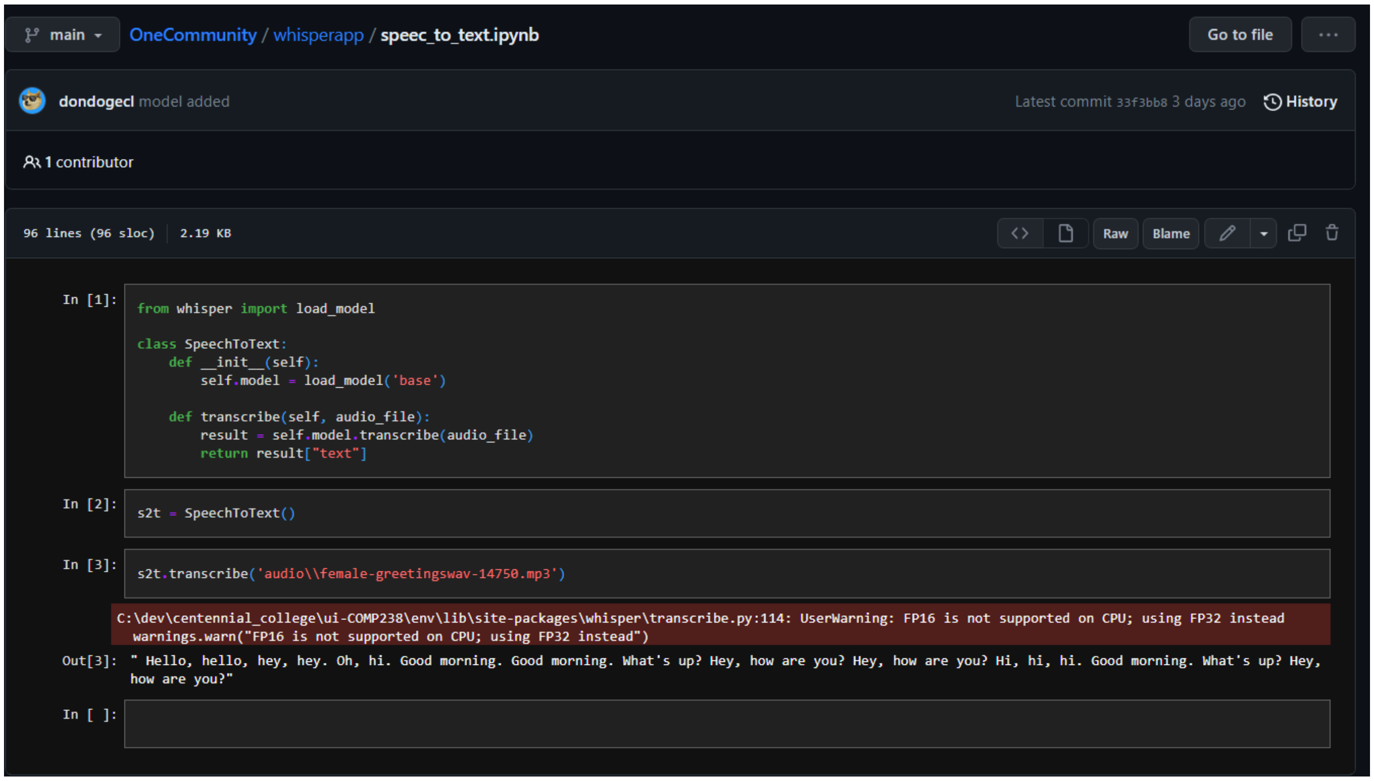
cd whisperapp

Python –r install requirements.txt

Subsequently the notebooks can be run in either Jupyter notebook, jupyper lab or Google colab.

Code examples

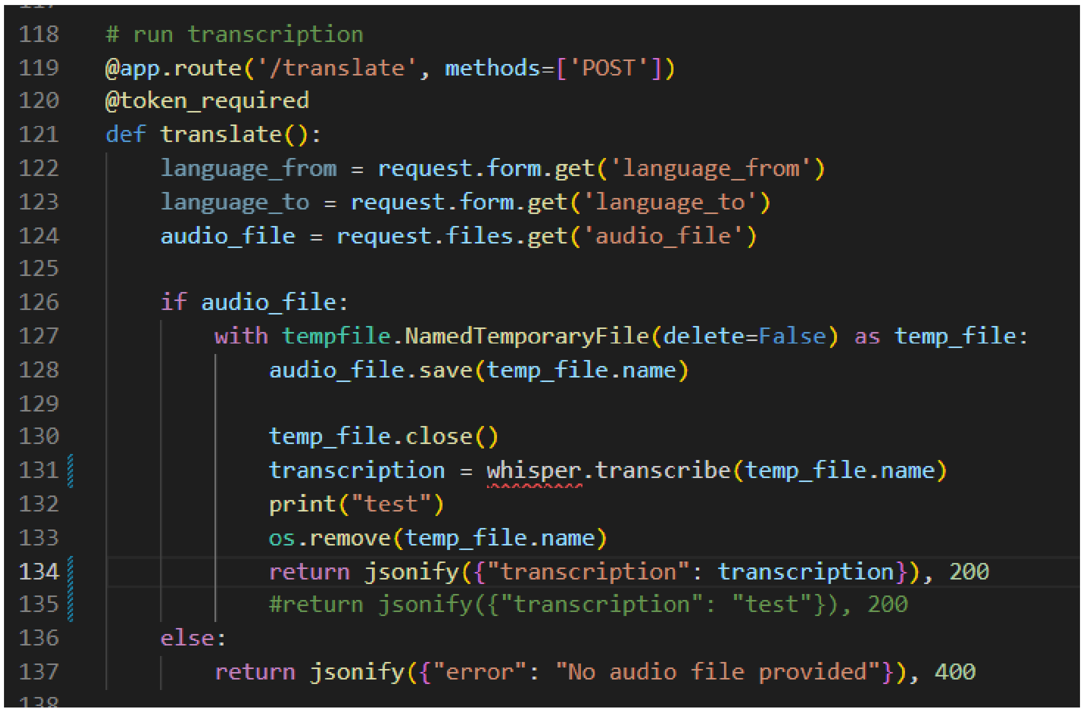
Whisper API examples: Running one sample audio:


main
OneCommunity / whisperapp / speec_to_text.ipynb
Go to file
dondogecl model added
Latest commit 33f3bb8 3 days ago
History
1 contributor
96 lines (96 sloc) | 2.19 KB
Raw
Blame

```
In [1]:  from whisper import load_model

         class SpeechToText:
             def __init__(self):
                 self.model = load_model('base')

             def transcribe(self, audio_file):
                 result = self.model.transcribe(audio_file)
                 return result["text"]

In [2]:  s2t = SpeechToText()

In [3]:  s2t.transcribe('audio\\female-greetingswav-14750.mp3')

C:\dev\centennial_college\ui-COMP238\env\lib\site-packages\whisper\transcribe.py:114: UserWarning: FP16 is not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Out[3]: " Hello, hello, hey, hey. Oh, hi. Good morning. Good morning. What's up? Hey, how are you? Hey, how are you? Hi, hi, hi. Good morning. What's up? Hey,
        how are you?"

In [ ]:
```

Backend:

Example of the whisper endpoint

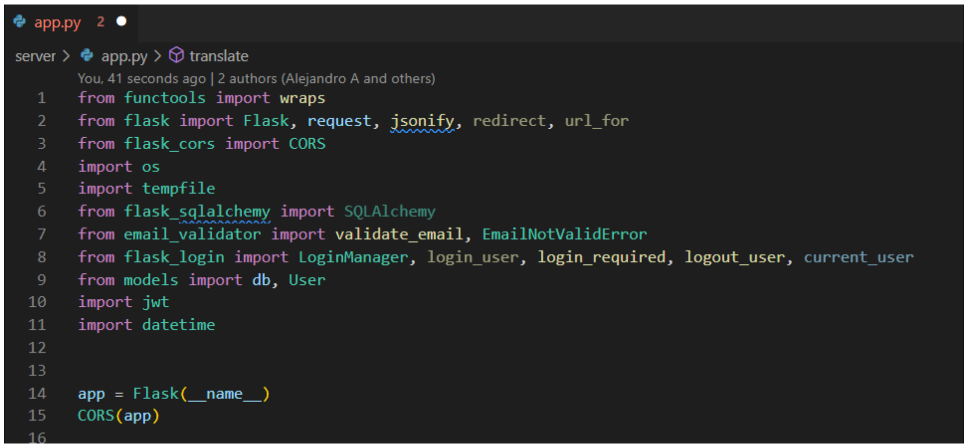


```python
# run transcription
@app.route('/translate', methods=['POST'])
@token_required
def translate():
    language_from = request.form.get('language_from')
    language_to = request.form.get('language_to')
    audio_file = request.files.get('audio_file')

    if audio_file:
        with tempfile.NamedTemporaryFile(delete=False) as temp_file:
            audio_file.save(temp_file.name)

            temp_file.close()
            transcription = whisper.transcribe(temp_file.name)
            print("test")
            os.remove(temp_file.name)
            return jsonify({"transcription": transcription}), 200
            #return jsonify({"transcription": "test"}), 200
    else:
        return jsonify({"error": "No audio file provided"}), 400
```

We also have endpoints for:

- Register (user)
- Login
- Users (returns a list of the users registered)

This can be reviewed in the code repository



app.py 2 ●

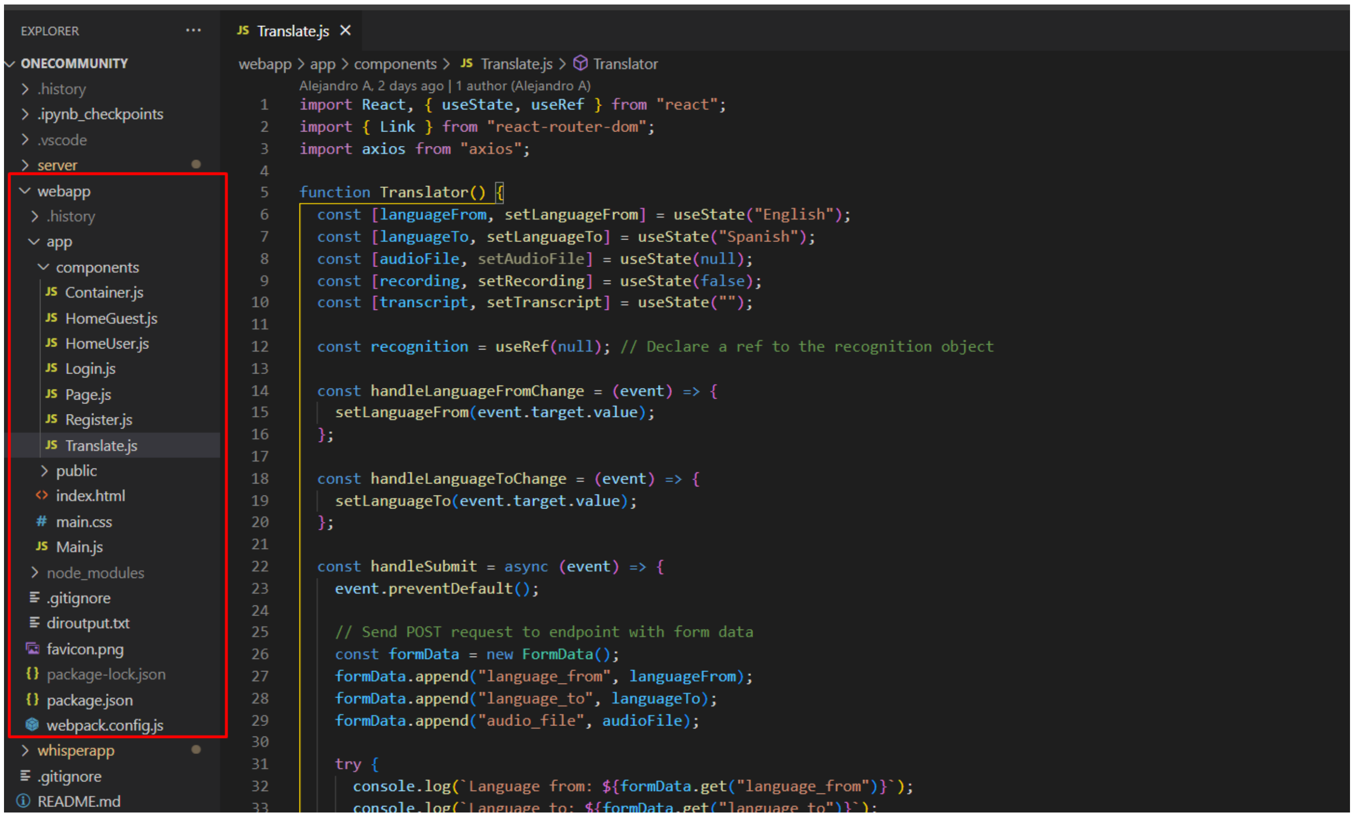
server > app.py > translate

You, 41 seconds ago | 2 authors (Alejandro A and others)

```python
from functools import wraps
from flask import Flask, request, jsonify, redirect, url_for
from flask_cors import CORS
import os
import tempfile
from flask_sqlalchemy import SQLAlchemy
from email_validator import validate_email, EmailNotValidError
from flask_login import LoginManager, login_user, login_required, logout_user, current_user
from models import db, User
import jwt
import datetime


app = Flask(__name__)
CORS(app)
```

For the Front-end:

React.js takes care of the user interactions (GUI) and does some simple validations. We understand that using the cloud has associated costs, that is why we implemented user registration.



```
EXPLORER ...
ONECOMMUNITY
  > .history
  > .ipynb_checkpoints
  > .vscode
  > server
  v webapp
    > .history
    v app
      v components
        Container.js
        HomeGuest.js
        HomeUser.js
        Login.js
        Page.js
        Register.js
        Translate.js
      > public
      index.html
      main.css
      Main.js
    > node_modules
    .gitignore
    diroutput.txt
    favicon.png
    package-lock.json
    package.json
    webpack.config.js
  > whisperapp
  .gitignore
  README.md
```

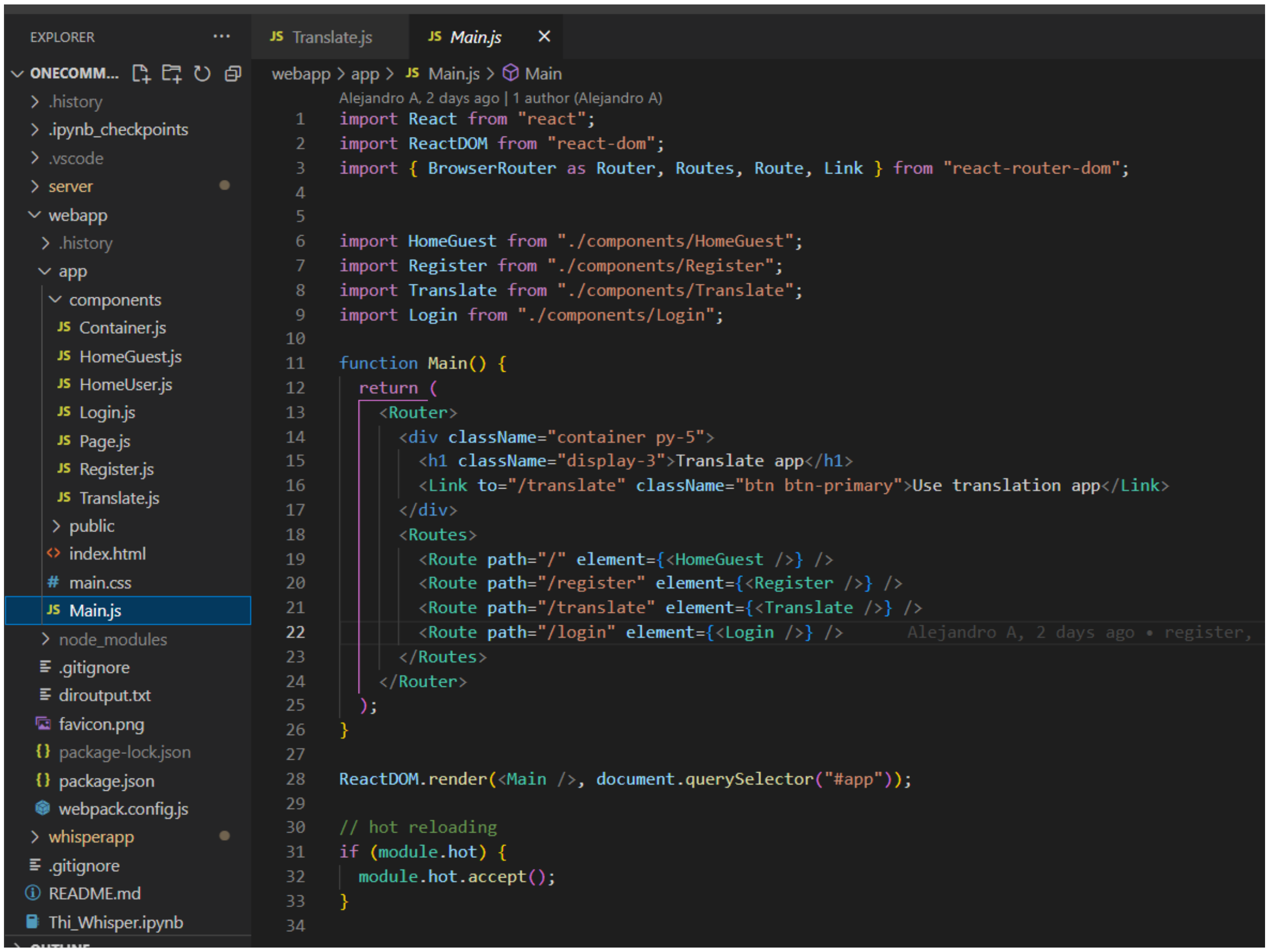
Translate.js

webapp > app > components > Translate.js > Translator

Alejandro A, 2 days ago | 1 author (Alejandro A)

```javascript
import React, { useState, useRef } from "react";
import { Link } from "react-router-dom";
import axios from "axios";

function Translator() {
  const [languageFrom, setLanguageFrom] = useState("English");
  const [languageTo, setLanguageTo] = useState("Spanish");
  const [audioFile, setAudioFile] = useState(null);
  const [recording, setRecording] = useState(false);
  const [transcript, setTranscript] = useState("");

  const recognition = useRef(null); // Declare a ref to the recognition object

  const handleLanguageFromChange = (event) => {
    setLanguageFrom(event.target.value);
  };

  const handleLanguageToChange = (event) => {
    setLanguageTo(event.target.value);
  };

  const handleSubmit = async (event) => {
    event.preventDefault();

    // Send POST request to endpoint with form data
    const formData = new FormData();
    formData.append("language_from", languageFrom);
    formData.append("language_to", languageTo);
    formData.append("audio_file", audioFile);

    try {
      console.log(`Language from: ${formData.get("language_from")}`);
      console.log(`Language to: ${formData.get("language_to")}`);
```

Main.js (most of the components are called from here):



EXPLORER

ONECOMM...

- .history
- .ipynb_checkpoints
- .vscode
- server
- webapp
  - .history
  - app
    - components
      - Container.js
      - HomeGuest.js
      - HomeUser.js
      - Login.js
      - Page.js
      - Register.js
      - Translate.js
    - public
    - index.html
    - main.css
    - Main.js
  - node_modules
  - .gitignore
  - diroutput.txt
  - favicon.png
  - package-lock.json
  - package.json
  - webpack.config.js
- whisperapp
- .gitignore
- README.md
- Thi_Whisper.ipynb

Translate.js | Main.js

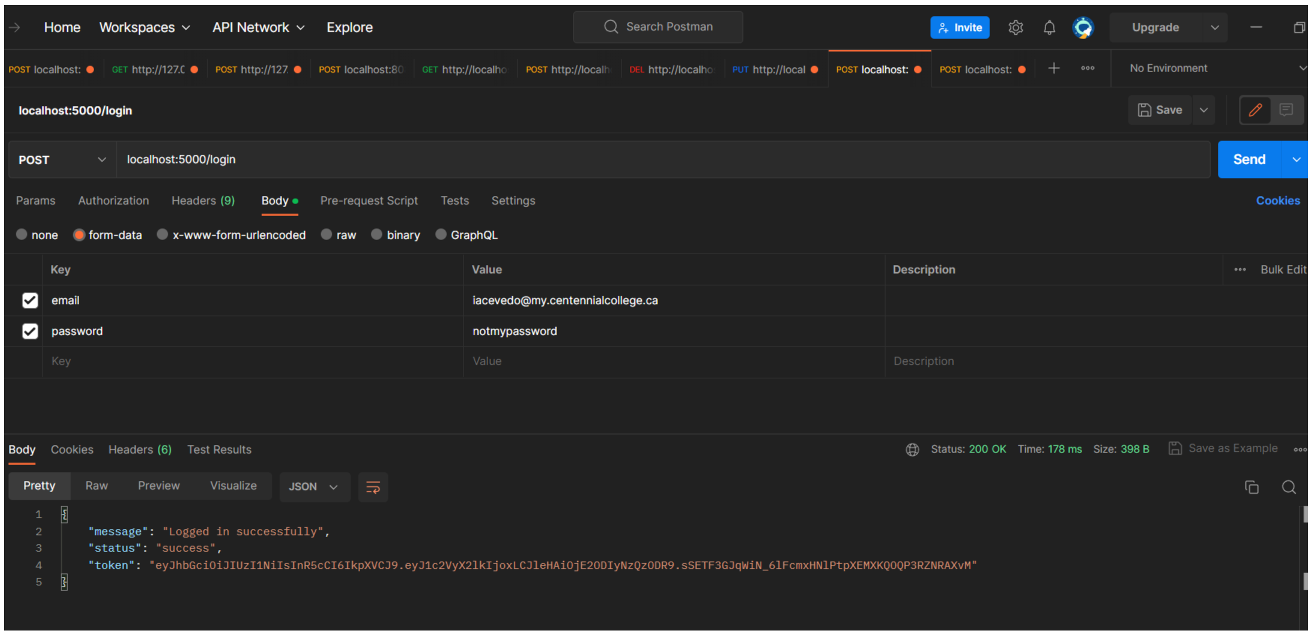
webapp > app > Main.js > Main

Alejandro A, 2 days ago | 1 author (Alejandro A)

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";


import HomeGuest from "./components/HomeGuest";
import Register from "./components/Register";
import Translate from "./components/Translate";
import Login from "./components/Login";

function Main() {
  return (
    <Router>
      <div className="container py-5">
        <h1 className="display-3">Translate app</h1>
        <Link to="/translate" className="btn btn-primary">Use translation app</Link>
      </div>
      <Routes>
        <Route path="/" element={<HomeGuest />} />
        <Route path="/register" element={<Register />} />
        <Route path="/translate" element={<Translate />} />
        <Route path="/login" element={<Login />} />     Alejandro A, 2 days ago • register,
      </Routes>
    </Router>
  );
}

ReactDOM.render(<Main />, document.querySelector("#app"));

// hot reloading
if (module.hot) {
  module.hot.accept();
}
```

Testing

For testing of the API endpoints we used POSTMAN.



localhost:5000/login

POST localhost:5000/login

Body: [✓ form-data]

Key	Value	Description
email	iacevedo@my.centennialcollege.ca	
password	notmypassword	

Status: 200 OK Time: 178 ms Size: 398 B

```
{
    "message": "Logged in successfully",
    "status": "success",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLCJleHAiOjE2ODIyNzQzODR9.sSETF3GJqWiN_6lFcmxHNlPtpXEMXKQOQP3RZNRAXvM"
}
```

As seen in the image above, we sent a POST request with a valid username and password that is already in the database. It sends back a JSON WebToken (JWT) that the front-end will store as LocalStorage in the browser for one hour (duration of the session).