Maintenance manual VoiceCoach

Introduction

The VoiceCoach system has 3 main parts, the client side that was made using TS and exported using expo to be available to both Android and IOS , the server side made using python that is located on an AWS EC2 server (part of the amazon system) and the DB that was made for Pgadmin4, in python, and is located on an AWS RDS server, which is a DB server (part of the amazon system).

To see our github and the latest updates go to this link:

https://github.com/Naveavra/VoiceCoach

Client side

As mentioned before, the client side was created in TS, you can see the full code on our Github (link is in the introduction).

In the frontend, the requests are sent using http to different routes on the server, if you want to update the frontend, we recommend to firstly run the code locally.

How to run the frontend locally

You will need to use expo to run the code on your computer, after you downloaded the packages for expo(just use “npm install in the code and all the needed packages will be downloaded), download expo-go on your cellphone, then simply run “npm run start” and you will be able to scan a barcode that is your app running on your local device, be sure your computer and cellphone are connected to the same Wi-Fi otherwise the barcode will not work.

After you test your additions to the app and you want to allow other people to use them, you will need to update the project on the expo servers.

How to create a new frontend version

You will need to create an expo account, after that you will need to update the project to be connected to your expo account, then simply run “eas update” and, if no errors occurred, you would receive a link to your release.

This release is the client side you created; you can share its barcode with anyone who has “expo-go” installed on their device.

Server side

As mentioned before, the client side was created in python, you can see the full code on our Github (link is in the introduction).

The server side was made using different routes and receiving http requests, if you want to update the code first you recommend you run the code locally.

*notice that the instructions are written for using AWS EC2 on an ubuntu machine

How to run the backend locally

After cloning the code, run “pip install -r requirements.txt”, this command will install all the needed packages for running the backend, after that run “flask run” in the “app” folder (Important!), than a local flask server will start to run, if you want to connect it to a local frontend run you will need to update the backend location in ‘config.ts’ on the client side, and then follow the instructions we wrote in “how to run the frontend locally”.

After checking everything works, if you want to run the code on a server for users to use, you will need to find a suitable location for the code to run, we used AWS EC2 but you can use another service.

Configuring the server for the backend to run

After finding a suitable server with all the resources you need, you will need to configure it to run the python flask server in a production environment so that it will be good for users to connect and use, to do that we recommend using “gunicorn” to handle the run of the code and “nginx” to handle the requests entering the server, after you configured both of them, use a service for the nginx to run the code, here is an example of the service we ran:

[Unit]

Description=Gunicorn instance for voiceCoach server

After=network.target

[Service]

User=ubuntu

Group=www-data

WorkingDirectory=/home/ubuntu/your_folder_name/app

ExecStart=/home/ubuntu/ your_folder_name /venv/bin/gunicorn -b localhost:8000 --timeout 120 main:app

Restart=always

[Install]

WantedBy=multi-user.target

*notice we wrote ‘your_folder_name’, this is the name of the folder that you will put all the code into.

To add the code to the server, is using EC2 then use scp request to add all the code, we ran:

scp -i your_pem_file.pem -r app ubuntu@13.60.77.150:/home/ubuntu/your_folder_name/ - command to add all the code to the server

*notice we wrote ‘your_pem_file’, this is the name of the pem file you will create for scp requests in AWS.

After all of that is configured, you will need to start the service using nginx, to do that run:

sudo systemctl daemon-reload

sudo systemctl start service_name

sudo systemctl enable service_name

*notice we wrote ‘service_name’, this is the name of the service file you created.

How to update code on the server

If after you already have a server, you would like to make changes to the code, you can do that by firsly stop the service run, using:

sudo systemctl disable service_name

sudo systemctl stop service_name

sudo systemctl daemon-reload

then add your new code using again the scp command, and lastly start the service again.