CS-623 Cloud Computing - Final Project

Audio Transcriber & Translator

Using Google Cloud

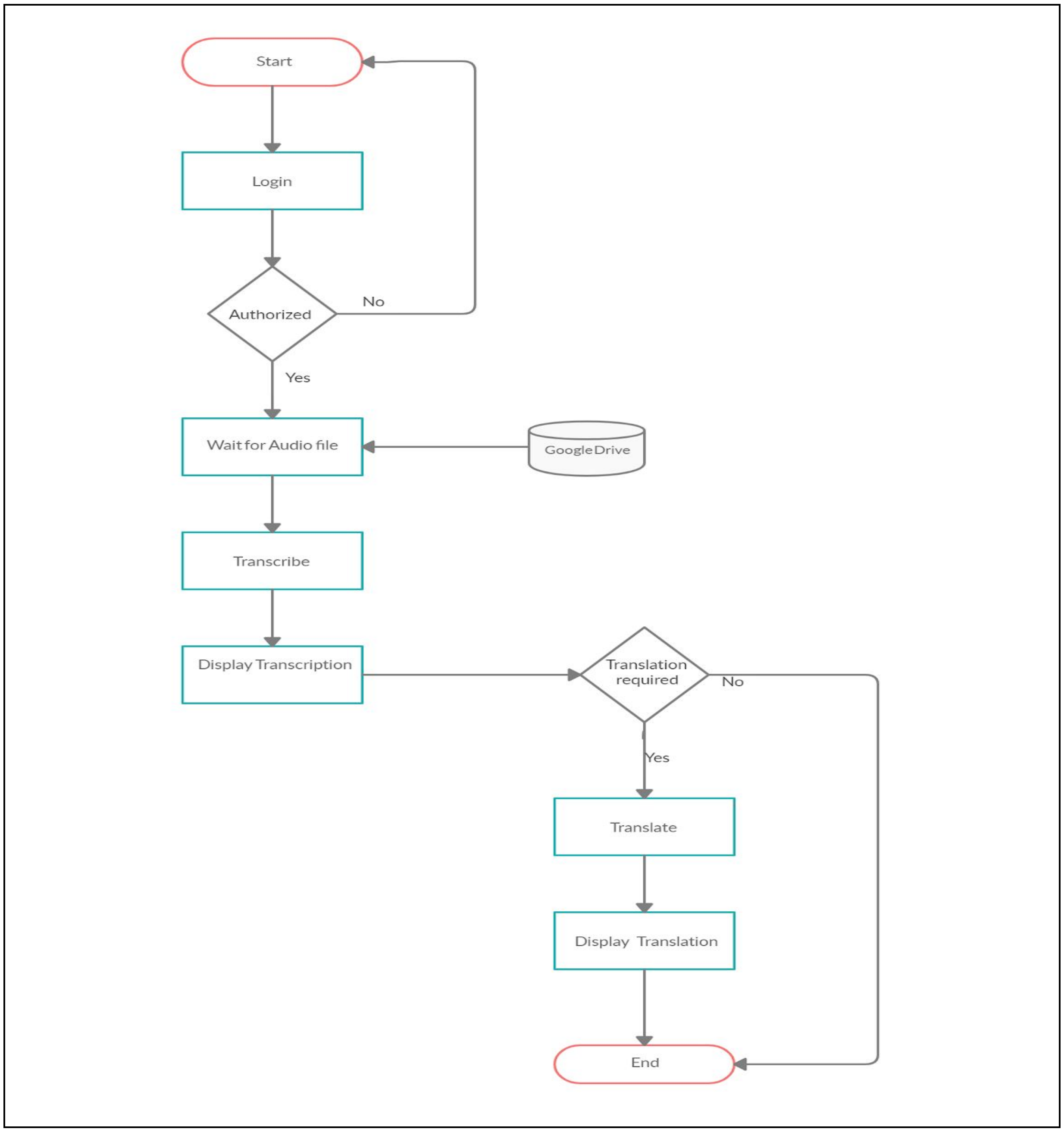
Designed and developed by:

Amna Khan (eq9642)

Project Description

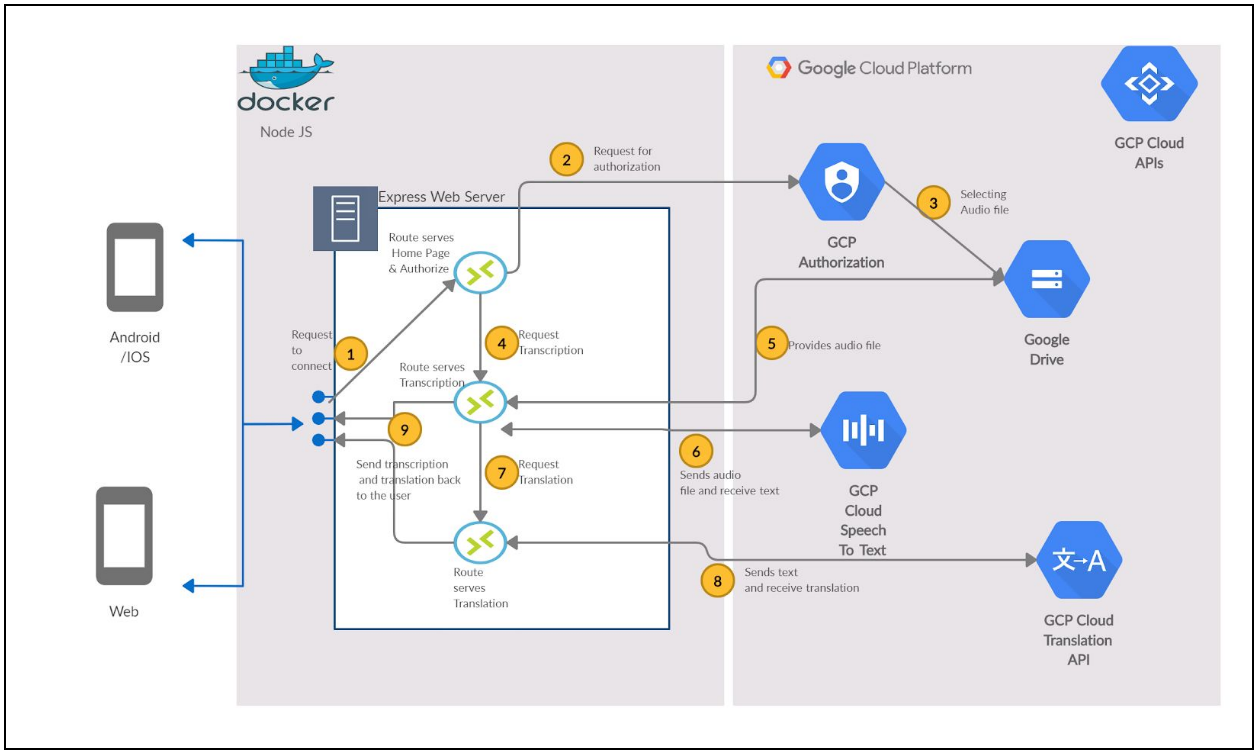
This cloud project provides an application that will transcribe the recording of meetings and provide the transcription in the form of text for note taking and reviewing the content of the meetings later. It will also provide the option to view the transcribed text in other supported languages to the team members working on the same project remotely in other countries

Flow Chart



Orchestration Model

This is a web-based project and is available via thin clients to anyone through a URL. So, To make these services work, I created a project on the Google Cloud platform, created API credentials, installed Google Cloud SDK which provides tools and libraries for interacting with Google Cloud products and services. The Cloud SDK comes with installable language-specific Google API Client Libraries.



The above model can be explained as follows. When a user from a thin client lands on the home page, before they can choose an audio file for further processing, they have to authenticate themselves using Google sign in. After that, they are able to select the audio file that needs to be transcribed from Google Drive. On pressing the Transcribe button, the request along with the file id of the audio is sent to the webserver which then grabs the audio file from Google Drive and provides it to the Speech to text API for transcribing. The API outputs the text which is then sent by the express server to the browser.

If the translation is needed, the browser sends this transcription (in English) again to the express web server which sends this text to the Google Cloud Translate API and the Translate API returns the translated text.

All these requests are sent and received via FETCH API which is a native JS library

Source code or Git access.

The complete source code of the project can be found here:
https://github.com/eq9642-AmnaKhan/GCP_Transcriber_Translator

(It is a private repo, and Professor's ID has been added as contributors)

Programming Languages Used

Frontend

The frontend is web-based and is implemented in HTML/CSS and JavaScript. Some of the design is built using Bootstrap

Backend

Backend Logic is implemented in JavaScript using Node JS.

Instruction of the Code and Technology

This application is running in a docker which is a container and it can be installed on any Linux machine. In this project, a Node JS Docker image is being used which is a JavaScript-based platform for server-side and networking applications. Any docker image can be downloaded from https://hub.docker.com/

Cloud services used in this project are Google Cloud Speech to Text API which is used for transcribing the audio files and Google cloud Translate API for translating the transcription into supported languages.
Another cloud service used is Google Cloud Storage. For storage, Google has several options. I used Google drive API, drive is commonly used for work and home that allows you to store and share your photos, videos, files, and more in the cloud.

Development method (Frontend Backend, DevOps, etc).
As I mentioned before, This project is web-based, it has a front end and a backend. The requests from the front end (browser) web page located at URL: nixbox.hopto.org is sent to the backend. The backend comprises Express JS as a web server that makes use of the REST APIs. The FETCH API which is native to JavaScript is used to send and receive thin client requests to the Express web server. These queries from the web server are then sent to Google Cloud Client Libraries for processing using Fetch API again. The processed data from the cloud libraries is then sent back to the webserver which then returns the data to the browser.

Code Explanation

Client-Side

It's a single page web app, the main page *filePicker.html* provides an interface to the application. All frontend JavaScript code is in *drivePicker.js* in which Google Cloud credentials have been specified including API key, Client ID, Project ID, and Scopes, obtained from Google API Console.

On the page load event, calls are made to load the API client library, the oauth2 library, and the picker library. After the successful authorization of the user the "Pick Audio'' button will show up. The implementation of the button click event will create and render a Picker for selecting any file on My Drive. When a file is picked a function *pickerCallback* gathers metadata from the picked documents and from calling get on the fileID and also displays the "Transcribe" button. The function that handles the Transcribe button event uses Fetch API to send fileID to the webserver and catches the response back in the form of text and displays in the text section on the page.

The Translate button event has a function that works pretty much the same way as Transcribe by making use of FETCH API but this time transcribed text is sent via the REST API Post method and in response, the translated text is received and displayed in the translate text area.

Server Side (webserver)

It is a Node JS application that includes several NPM packages:

- Express JS to handle requests from the client
- @google-cloud/speech package to connect to Speech-to-Text API, using Project ID and API key
- @google-cloud/translate package to connect to Translate API using an API key

Deployment Instructions

Docker setup

1. Create a directory with your server-side code (for this project it was Express JS webserver)
2. Create a Dockerfile by specifying these attributes
    a. Specify an image to be used

   b. Working directory
   c. Copy command to get package.json
   d. Run the‘ npm install’ command to install dependencies of the project
   e. Expose port# and run the node js file.
3. Build the docker with the following command.
   ```
   docker build -t folder/appname
   ```
4. Run the docker with this command
   ```
   docker run -p 3400:3400 -d foldername/appname
   ```

After this setup, the Express server is up inside the docker and running to serve the requests.

Setting Up Google Cloud Platform Client Libraries.

1. In the Google Cloud Console, on the project selector page, select or create a Google Cloud project.

2. Make sure that billing is enabled for your Cloud project.

3. Set up authentication: From API & Services -> Credentials
   a. In the Cloud Console, go to the Create service account key page.
   b. Create a new service account and a Key (private) which is a JSON file that contains your key and it downloads to your computer. assign it the role of owner.
   c. Create a new Client ID. The client id needs to have a whitelisted domain.
   d. Create an API key in the same project

4. Google APIs to enable in the Google Cloud Console. From the API & Services -> Library and search and enable all these libraries for this particular project.
   - Speech-to-Text API
   - Drive API
   - Picker API
   - Translate API

All of the client libraries or services that I mentioned use the same API key which was made at the start of the project. The service account key is a JSON file that contains project identifier, private key, and other project identification parameters associated with the service account which is also required by the project. Once the credentials have been set, the cloud services can be used for that particular project.

Development Environment

For development, install google cloud SDK
https://cloud.google.com/sdk/docs/install

After the SDK is installed, in the cmd prompt initialize by

```
gcloud init
```

This will take you through the process, by login and then select the Google Cloud Project, the project can be set up in a few minutes.

Now we can install any client library, for instance

```
npm install --save @google-cloud/speech
```

and start coding

Database model and diagram.

Currently, there is no database architecture implemented for this project.

Limitations:

1. By default, all files saved on Google drive are restricted to that Google account, so the Google APIs cannot access and process these files. The speech files need to be saved in the shared folder on Google drive. It is assumed that the team has a shareable folder on the Drive where they all save their recordings from online and in-person meetings.
2. The current implementation supports only audio files at this time.
3. Translation text is supported in these languages: