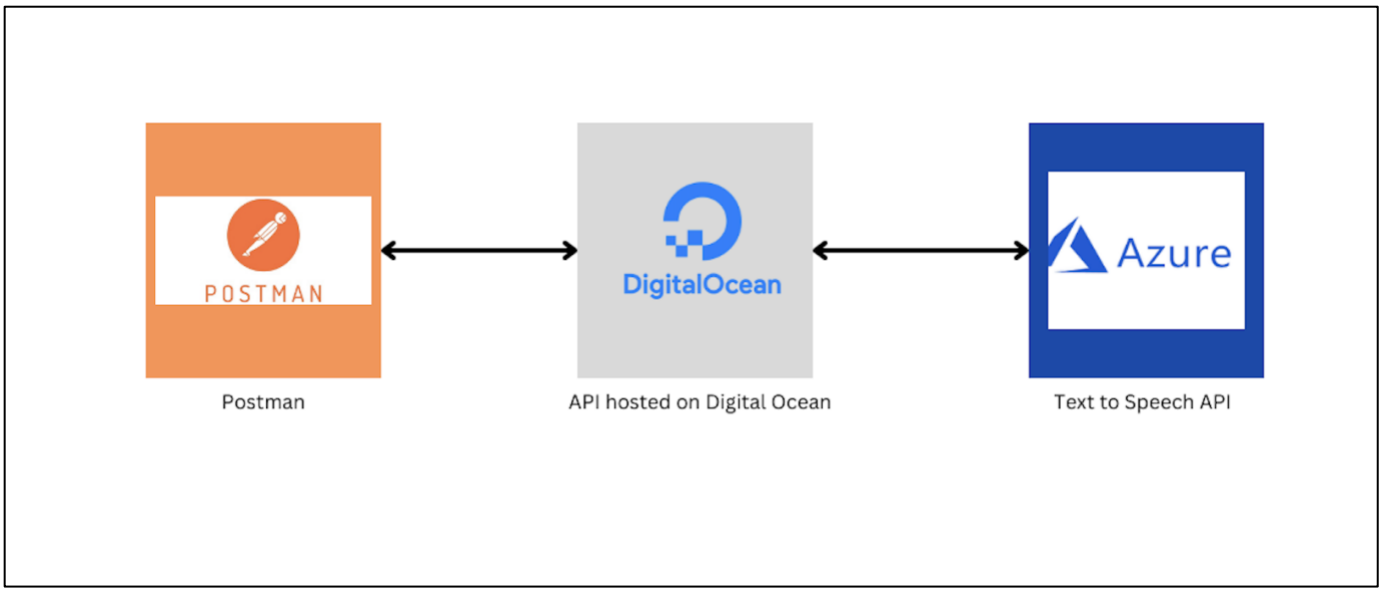
ITIS 6177 – System Integration Final Project Documentation

- Ajinkya Gadgil (801200445)

Introduction:

I have implemented Text to Speech conversion in this project using Microsoft Azure Cognitive Services Text-to-Speech API. Here when we pass the text to the API it returns the audio output which is the speech output of the input text.

Project Architecture and Explanation:



The above diagram explains the architecture of the project. There are three parts to it.

1. Postman/UI:
    a. This is used to call the digital ocean-hosted API to get the response. Here called maybe postman or a UI element to get the speech audio data.
2. API hosted on Digital Ocean:
    a. This is SI project API that in turn calls the Text to Speech API Azure APIs
3. Text to Speech API:
    a. These are cognitive service APIs for Text to Speech related operations provided by Azure.

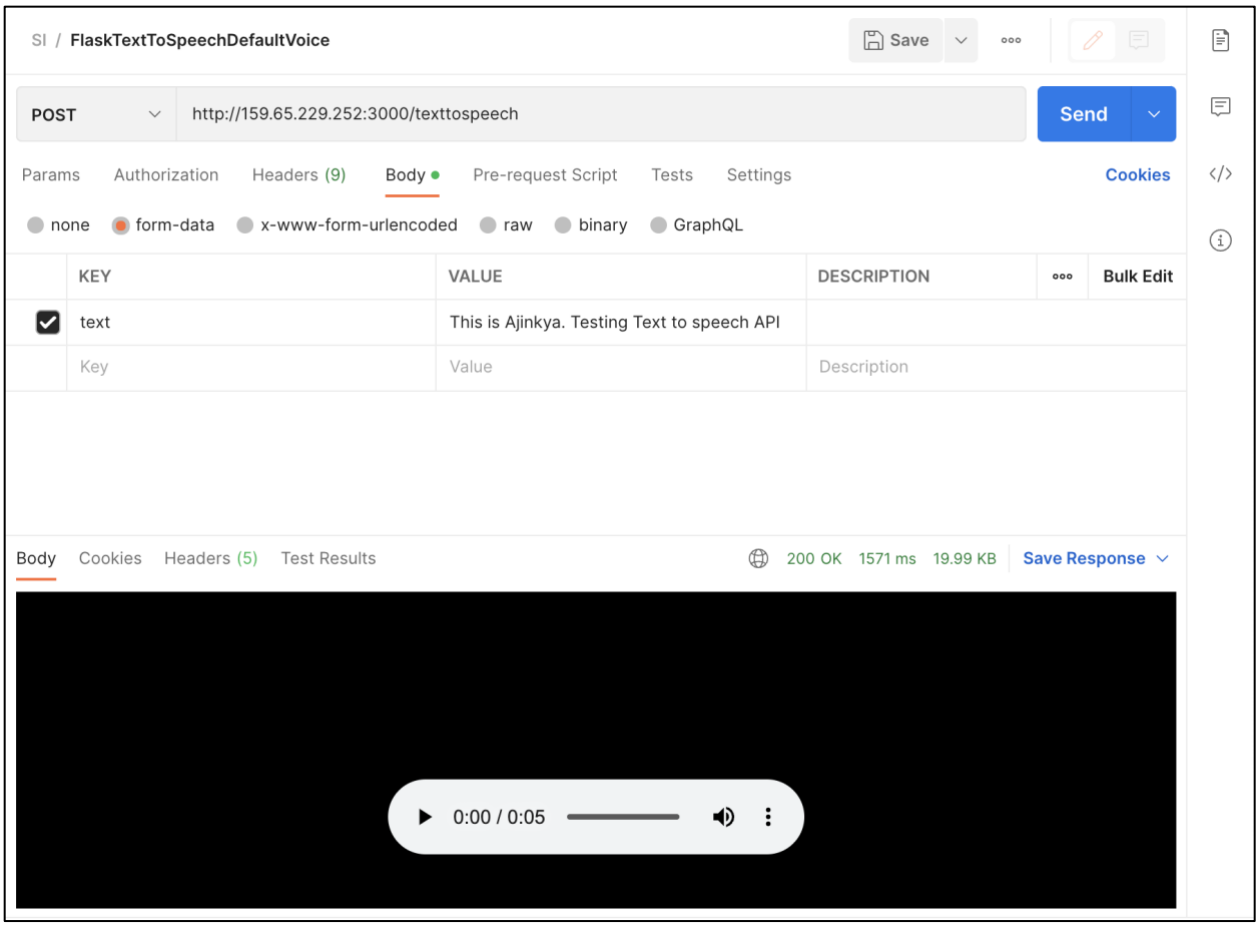
Here postman calls the API hosted on digital ocean with the required parameters and then using those parameters the digital ocean hosted API calls the Text to Speech APIs provided by Azure which returns the response to APIs hosted on the digital ocean which processes the response and returns it to the caller i.e., Postman or any UI which may choose how to use or modify the data.

Tech Stack and Scope:

1. I have used Python Flask for creating the APIs. Consumed Microsoft Cognitive Services Text to Speech APIs for converting text to speech.
2. Azure Text to Speech API
3. Digital Ocean for hosting the API
4. Git for version control
5. I have implemented majorly 3 APIs that complete text to speech process and have some flexibility as well.
    a. API that converts given text to speech and the speech audio is the default voice which is not configurable.
    b. The GET request returns the voice samples which can be used for output audio
    c. API that converts given text to speech audio using configurable voice samples that we get from the above API (b)

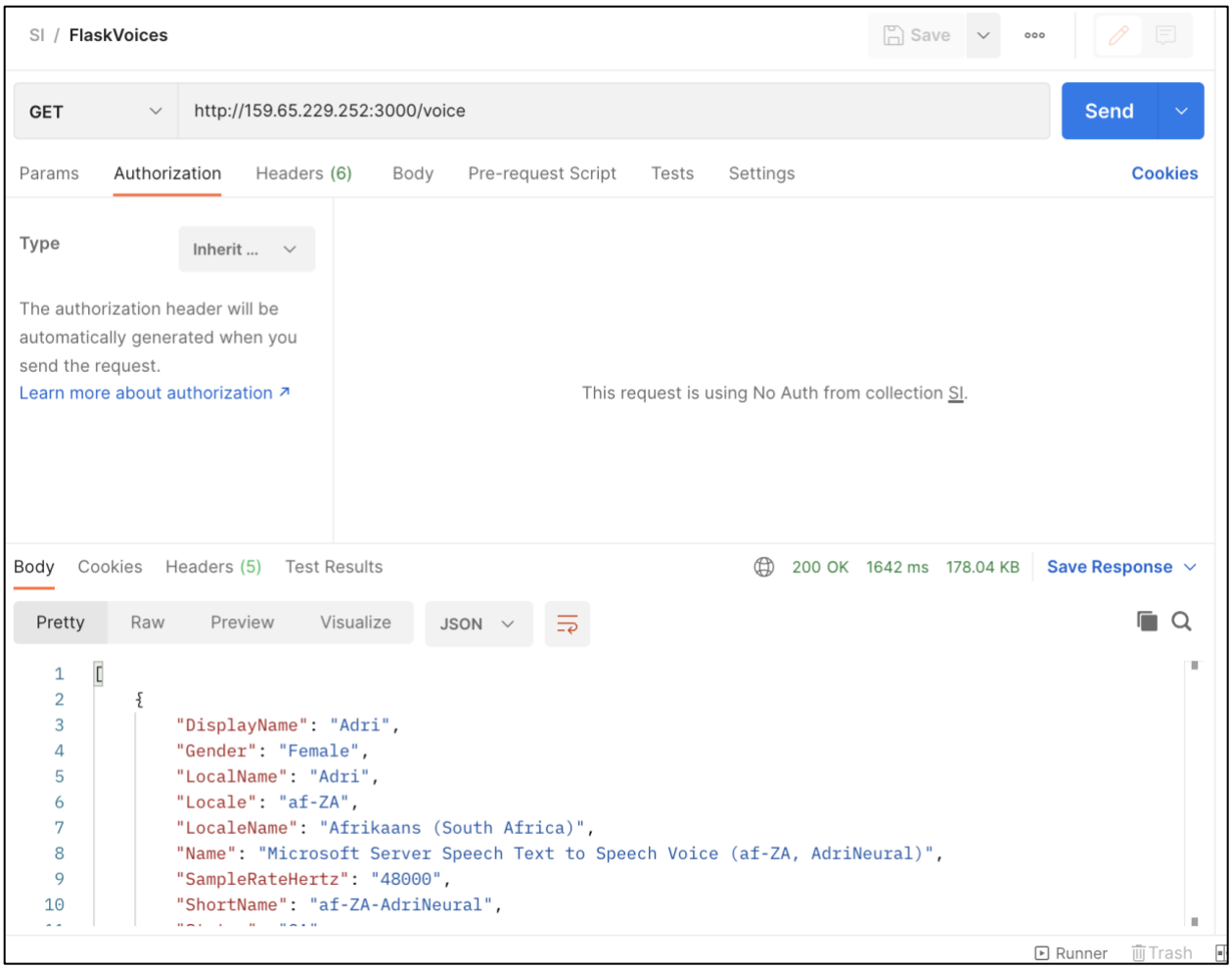
Instructions to run API from POSTMAN

1. **Text to Speech using default voice:**
   Endpoint URL: http://159.65.229.252:3000/texttospeech
   Request Type: POST
   Body: text to convert into speech as form-data



In the above screenshot, you can see that postman sends a post request to API hosted on digital ocean with text to be converted into the body(form-data) of the request. It returns status 200 i.e., OK with an audio file that can be played/saved which is the converted speech audio output of the text. Here we are not giving any particular voice sample and hence it uses default voice sample i.e. 'en-US-ChristopherNeural'.

2. **Voice API**
   Endpoint URL: http://159.65.229.252:3000/voice
   Request Type: GET

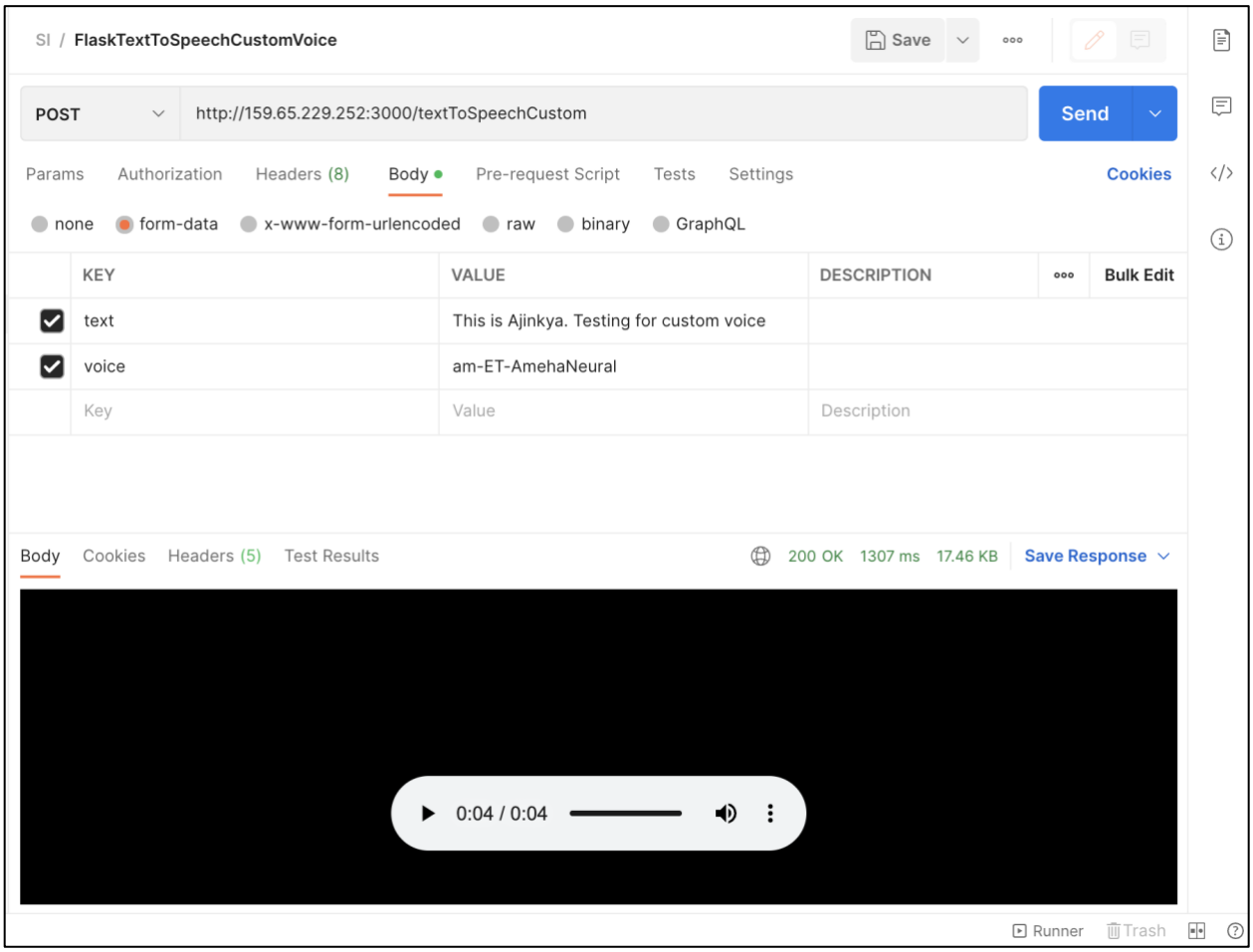


In the above screenshot, we call the voice API which returns an array of the object of all the voices that are supported by the Azure Text to Speech API and use one of these voice samples for our speech output. Example of one object is below


```
{
    "DisplayName": "Ameha",
    "Gender": "Male",
    "LocalName": "አምሀ",
    "Locale": "am-ET",
    "LocaleName": "Amharic (Ethiopia)",
    "Name": "Microsoft Server Speech Text to Speech Voice (am-ET, AmehaNeural)",
    "SampleRateHertz": "48000",
    "ShortName": "am-ET-AmehaNeural",
    "Status": "GA",
    "VoiceType": "Neural",
    "WordsPerMinute": "112"
}
```

In the object, we can use the short name to send it to azure API and it returns the speech output in that voice sample.

3. **Text to Speech using custom voice**
   Endpoint URL: http://159.65.229.252:3000/textToSpeechCustom
   Request Type: POST
   Body: Text to convert to Speech and voice sample to be used for speech output.



In the above screenshot, we send a post request with 2 parameters. First is a text which is to be converted into speech and second is a voice which is the short name of any object that is returned from voice API. This returns a status 200 with an audio file that can be played or saved. Here it converts the given text to speech, but the voice sample is different and would depend on the voice

Error Handling

Following are the status codes and its meaning which may be returned from the API

1. **200: Success** - Request is successful.

2. **400: Bad Request** - This is the most common error, and it occurs when some of the parameters are missing, empty or null.
3. **401: Unauthorized** – The request is not authorized to make sure that the token/auth key is correct.
4. **503: Service Unavailable** - When the server is down or unable to process a request.

Subscription Key Security:

As shown in the architecture we use the Azure Text to Speech APIs. To use Azure Text to Speech APIs we need the Subscription key or token which then verifies the identity of the caller. This subscription key or token cannot be made public or written in the code so I created a .env file that stores the key-value pair and we can get that securely using the os module in Python Flask.

Steps to Run the code

1. Clone the repository https://github.com/ajinkyagadgil/itis6177-project.git
2. Go to the project folder where there is requirements.txt file and run the command **'pip install -r requirements.txt'** which would install all the dependencies. (Make sure to have python and pip installed on your system)
3. Create a .env file and add key as **SUBSCRIPTION_KEY=<azure text to speech key>**
4. Run the code with command **python app.py** and use the endpoints to convert text to speech
5. Create a resource group and service of Azure Text to Speech API to get the endpoint and subscription key

Video Link of the demonstration:

I have prepared a short video of the demo hosted on the below google drive link

https://drive.google.com/file/d/1RQei84oCatTYb9ShgBmUrsCvvVJhhebK/view?usp=sharing