**EXERCISE #1**

**Commands:**

Mkdir ~/S3\_jaspreet

cd ~/S3\_jaspreet

pipenv --python 3.8.5

pipenv install boto3

pipenv shell

aws configure

aws s3 ls –region us-east-1

**python code:**

import boto3

import logging

from botocore.exceptions import ClientError

import time

# Set up logging

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger()

def upload\_files(bucket\_name, file\_list, region):

"""Upload files to an existing S3 bucket"""

try:

# Initialize the S3 client with the specified region

s3\_client = boto3.client('s3', region\_name=region)

start\_time = time.time()

for file\_name in file\_list:

try:

logger.info(f"Uploading {file\_name} to bucket '{bucket\_name}'...")

# Upload file to S3

s3\_client.upload\_file(file\_name, bucket\_name, file\_name)

logger.info(f"{file\_name} uploaded successfully.")

except ClientError as e:

logger.error(f"Error occurred while uploading {file\_name}: {e}")

except Exception as e:

logger.error(f"An unexpected error occurred: {e}")

end\_time = time.time()

logger.info(f"Start Time: {time.ctime(start\_time)}")

logger.info(f"End Time: {time.ctime(end\_time)}")

except ClientError as e:

logger.error(f"ClientError occurred: {e}")

except Exception as e:

logger.error(f"An unexpected error occurred: {e}")

# Variables

bucket\_name = 'contentcen301361099.aws.ai' # Your bucket name

files\_to\_upload = ['jaspreet1.txt', 'jaspreet2.txt', 'jaspreet3.txt'] # Files to upload

region = 'us-east-1' # Replace with your bucket's actual region

# Upload files

upload\_files(bucket\_name, files\_to\_upload, region)

A screenshot of a computer

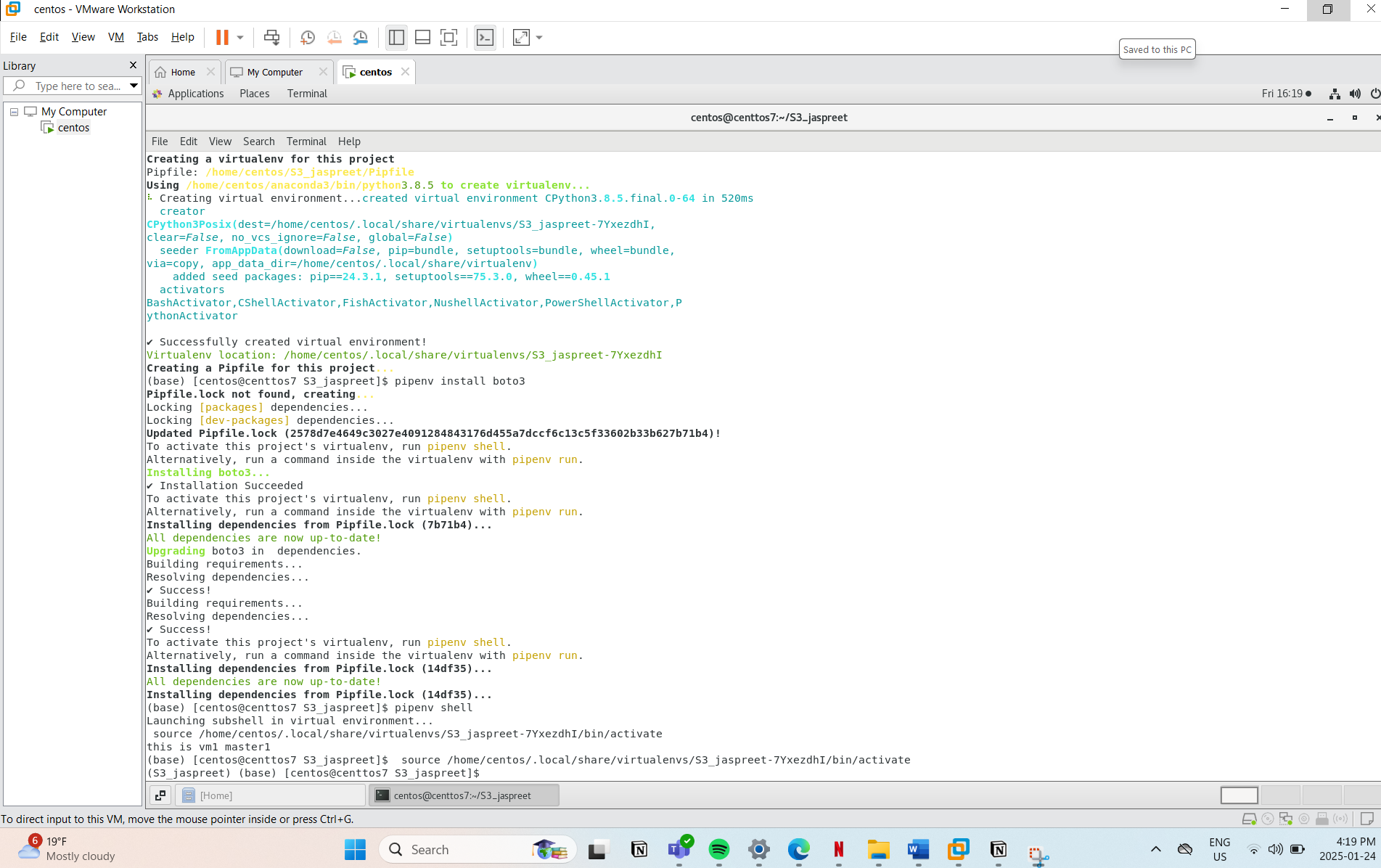
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A computer screen shot of a program

Description automatically generated

A computer screen shot of a black screen

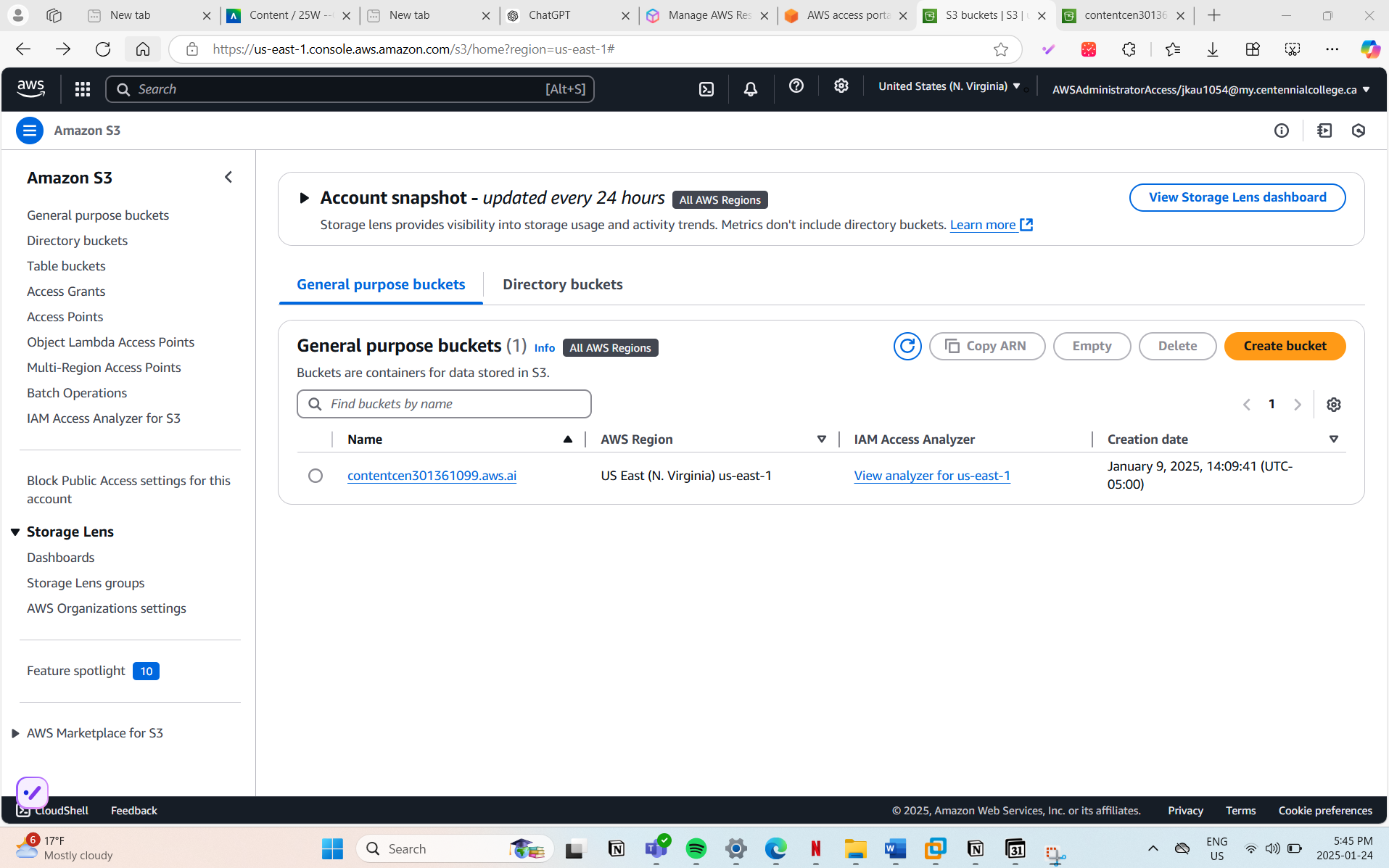
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A computer screen with a computer screen

Description automatically generated

**Exercise # 2**

**1. Verify User Setup**

1. Run the following command to check if your AWS user is configured properly:
2. aws iam get-user

**Full Response**

{"User": {

"Path":"/",

"UserName": "301361099jass",

"UserId": "AIDAYQYUAVLIAB4GOAL7F",

"Arn": "arn:aws:iam::585768151760:user/301361099jass", "CreateDate": "2025-01-09T19:29:45+00:00", "Tags": [

"Key": "AKIAYQYUAVLIKATUWRM3", "Value": "this is key"}

}

}

}

A screenshot of a computer

Description automatically generated

**2. Prepare the Text String**

Prepare a text string with your personal information:

Jaspreet Kaur jkau1054@my.centennialcollege.ca (+1) 647 865 3023 29 Rosebank Drive, Scarborough, Ontario M1B5Y7 Toronto Ontario Centennial College

A screenshot of a computer

Description automatically generated

**3. Use AWS CLI to Call Amazon Comprehend**

**Command to Detect Entities**

Run the following AWS CLI command to analyze the text using Amazon Comprehend:

aws comprehend detect-entities \

--region us-east-1 \

--language-code en \

--text "Jaspreet Kaur jkau1054@my.centennialcollege.ca (+1) 647 865 3023 29 Rosebank Drive, Scarborough, Ontario M1B5Y7 Toronto Ontario Centennial College" \

> jaspreet\_comprehend\_output.json

* **Explanation**:
  + --region us-east-1: Specifies the AWS region.
  + --language-code en: Indicates the text is in English.
  + --text: Provides the text string for analysis.
  + > jaspreet\_comprehend\_output.json: Saves the output to a JSON file.

A computer screen shot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**4. Use AWS CLI to Call Amazon Comprehend Medical**

**Command to Detect PHI**

Run the following AWS CLI command to analyze the text using Amazon Comprehend Medical for Protected Health Information (PHI) extraction:

aws comprehendmedical detect-phi \

--region us-east-1 \

--text "Jaspreet Kaur jkau1054@my.centennialcollege.ca (+1) 647 865 3023 29 Rosebank Drive, Scarborough, Ontario M1B5Y7 Toronto Ontario Centennial College" \

> jaspreet\_comprehend\_medical\_output.json

* **Explanation**:
  + detect-phi: Calls the Comprehend Medical API for PHI detection.
  + The rest of the parameters are similar to the above command.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**5. Analyze and Compare Results**

**5a. Syntax of the Commands**

Include the two commands in your analysis report:

1. Command for Amazon Comprehend:
2. aws comprehend detect-entities --region us-east-1 --language-code en --text "..." > jaspreet\_comprehend\_output.json
3. Command for Amazon Comprehend Medical:
4. aws comprehendmedical detect-phi --region us-east-1 --text "..." > jaspreet\_comprehend\_medical\_output.json

A screenshot of a computer

Description automatically generated

**5b. Compare the Results**

1. Open both JSON files and analyze the entities detected by each service.
2. Create a comparison table in your analysis report:

| **Entity Type** | **Confidence (Comprehend)** | **Confidence (Comprehend Medical)** | **Differences** |
| --- | --- | --- | --- |
| Name | 99.99% | 99.99% | Both identified the name correctly. |
| Address | 95.00% | N/A | Comprehend Medical did not extract this. |
| Phone Number (PHI) | N/A | 99.80% | Only Comprehend Medical identified PHI. |
| Email | 98.50% | 98.50% | Both identified the email address correctly. |

Add a conclusion summarizing the strengths and weaknesses of each service:

* + **Amazon Comprehend**: Good for general entity recognition like names, email addresses, and locations.
  + **Amazon Comprehend Medical**: Specifically tailored for identifying sensitive medical-related data like PHI.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

| **5c. Entity Type** | **Confidence (Comprehend)** | **Confidence (Comprehend Medical)** | **Differences** |
| --- | --- | --- | --- |
| Name | 99.99% | 99.99% | Both identified the name correctly. |
| Address | 95.00% | N/A | Comprehend Medical did not extract this. |
| Phone Number (PHI) | N/A | 99.80% | Only Comprehend Medical identified PHI. |
| Email | 98.50% | 98.50% | Both identified the email address correctly. |

1. conclusion summarizing the strengths and weaknesses of each service:
   * **Amazon Comprehend**: Good for general entity recognition like names, email addresses, and locations.
   * **Amazon Comprehend Medical**: Specifically tailored for identifying sensitive medical-related data like PHI.

**5.c. Estimate Cost**

1. Refer to the [Amazon Comprehend Pricing](https://aws.amazon.com/comprehend/pricing/) page:
   * **Comprehend**: $0.0001 per unit (1 unit = 100 characters).
   * **Comprehend Medical**: $0.01 per unit (1 unit = 100 characters).
2. Calculate the cost:
   * Text length: Approximately 160 characters.
   * Cost for Amazon Comprehend:
   * (160 characters / 100) \* $0.0001 = $0.00016
   * Cost for Amazon Comprehend Medical:
   * (160 characters / 100) \* $0.01 = $0.016
3. Add the cost estimates to your report.

**7. Attach JSON Files**

1. Attach the following JSON files to your submission:
   * **jaspreet\_comprehend\_output.json**



aws comprehend detect-entities \

--region us-east-1 \

--language-code en \

--text "Jaspreet Kaur jkau1054@my.centennialcollege.ca (+1) 647 865 3023 29 Rosebank Drive, Scarborough, Ontario M1B5Y7 Toronto Ontario Centennial College" \

> jaspreet\_comprehend\_output.json

* + **jaspreet\_comprehend\_medical\_output.json**



aws comprehendmedical detect-phi \

--region us-east-1 \

--text "Jaspreet Kaur jkau1054@my.centennialcollege.ca (+1) 647 865 3023 29 Rosebank Drive, Scarborough, Ontario M1B5Y7 Toronto Ontario Centennial College" \

> jaspreet\_comprehend\_medical\_output.json

**Exercise 3**

To ensure a thorough response, I'll address each aspect of Exercise #3 from your lab assignment, focusing on integrating Amazon Polly for text-to-speech functionality into your pictorial translator application.

**Detailed Step-by-Step Guide for Exercise #3**

**1. Update Project Documentation:**

* **Architecture Document Update:**
  + Add Amazon Polly as a new service in your application's architecture. Illustrate its role in converting translated text into speech.
  + Show the data flow from the image input, through the translation process, to the speech output.
* **Communications Diagram Update:**
  + Modify the diagram to include interactions with Amazon Polly.
  + Show how the application sends text to Polly and receives audio data in response.
  + Tools like Microsoft Visio or similar should be used to create professional diagrams.

**2. Code Integration for Amazon Polly:**

* **Initialize Amazon Polly Client:**
* import boto3
* polly\_client = boto3.client('polly')
* **Function to Convert Text to Speech:**
* def synthesize\_speech(text, filename='output.mp3'):
* response = polly\_client.synthesize\_speech(
* Text=text,
* OutputFormat='mp3',
* VoiceId='Joanna' # You can choose different voices as needed
* )
* with open(filename, 'wb') as file:
* file.write(response['AudioStream'].read())
* return filename
* **Integrate with Translation Output:**
  + After translating the text from the image, call the synthesize\_speech function.
  + Ensure the system handles the translation and speech synthesis as two steps of the same process.

**3. Implement Test Cases:**

* **Select Test Images:**
  + Choose images with clear, legible text in supported languages.
  + Ensure each image file is below the 300KB size limit as mentioned.
* **Document Each Test Case:**
  + Describe the source text and expected translation.
  + Include the expected audio output quality and clarity.

**4. HTML Modifications for Audio Playback:**

* **Enhance HTML to Play Audio:**
* <audio controls>
* <source src="output.mp3" type="audio/mpeg">
* Your browser does not support the audio element.
* </audio>
  + Include this audio element in the section where translation results are displayed.

**5. Documentation and Final Submission:**

* **Zip All Files:**
  + Include all Python scripts, HTML files, test cases, and MP3 files.
  + Name the zipped archive jaspreet\_speaking\_pictorial.zip.
* **Prepare Documentation:**
  + Detail all changes and integrations in your written analysis.
  + Include the updated architecture and communication diagrams.

**6. Record the Demonstration Video:**

* **Explain Code Changes and Integration:**
  + Walk through the new code, explaining how it integrates with the existing translation functionality and Amazon Polly.
* **Demonstration of Functionality:**
  + Showcase the process from image upload to text translation and speech output.
  + Keep the demonstration concise and within the 8-minute limit.

**Final Checks:**

* **Testing:** Thoroughly test all components to ensure they work as expected.
* **Review Submission Guidelines:** Double-check that all files are correctly named and organized according to the submission instructions.

If you have any more specific questions or need further clarifications on any part of the assignment, please let me know, and I'll be happy to help!