Lab 1: Calling, Building, and Securing APIs

Created an Account and Connected to the Azure Vision API

1. Signed up for a Microsoft Azure student account at [Azure Free for Students](#).
2. Created an instance of the Computer Vision service from the Azure portal.
3. Generated an API endpoint and subscription key for the Computer Vision service.
4. Updated the provided analyze.py script with the generated endpoint and subscription key.

Deliverables:

- Successfully connected to the Azure Vision API and retrieved text from a test image using the OCR feature.

Secured the Credentials

1. Removed hard coded credentials from the analyze.py script.
2. Created a .env file with the following structure:
   ```
   AZURE_VISION_API_KEY="<your-subscription-key>"
   AZURE_VISION_ENDPOINT="<your-api-endpoint>”
   ```
3. Used the python-dotenv library to load credentials securely from the .env file.
4. Added .env to .gitignore to prevent committing credentials to the Git repository.

Best Practices I Followed:

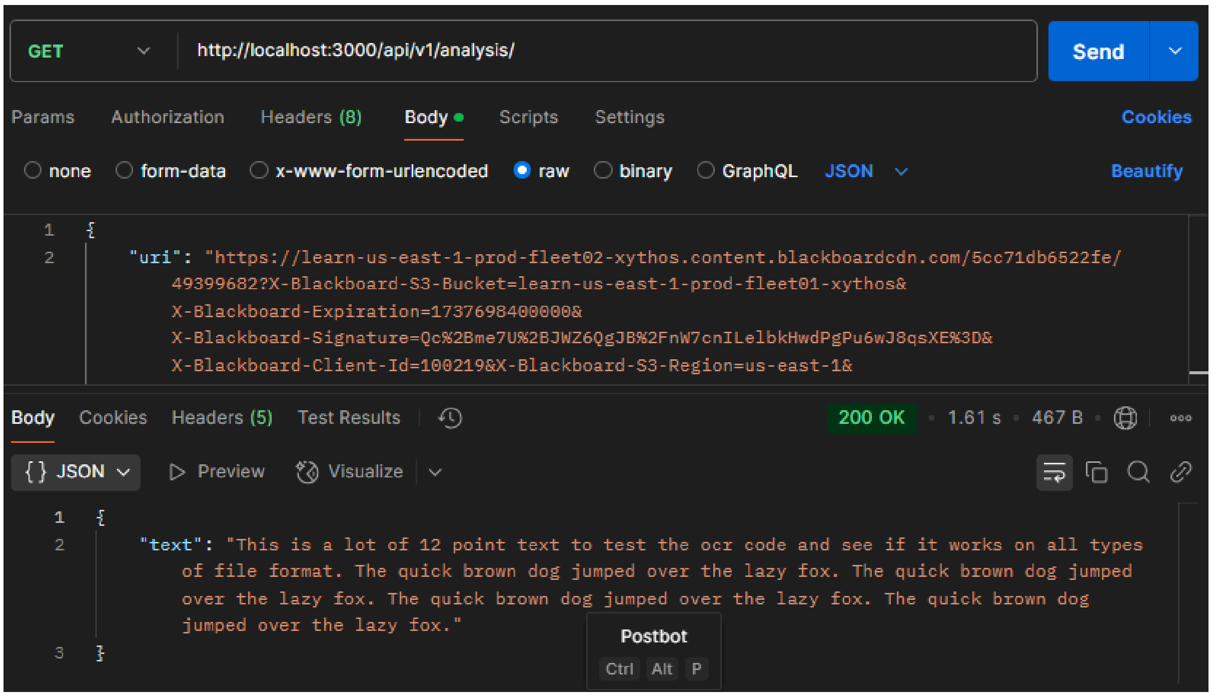
- Instead of hard-coding credentials in the code, added the .env file to .gitignore to ensure sensitive information is not exposed.
- Used environment variables to securely access credentials during runtime.
- Committed the updated code to GitHub without sensitive credentials.

Run the API Endpoint

1. Installed dependencies from requirements.txt using:
   ```
   pip install -r requirements.txt
   ```
2. Started the Flask server by running:
   ```
   python app.py
   ```
3. Tested the API by visiting http://localhost:3000/ to ensure the Flask server was running and functional.

Invoke the API with Inputs

1. Used the /api/v1/analysis/ endpoint to analyze a test image.
2. Made a request using Postman:
   Set the method to GET and URL to "http://localhost:3000/api/v1/analysis/"
   Add a JSON body:
   "{
   "uri": "https://example.com/image.jpg"
   }"
3. Verified the response contained the text extracted from the test image.



http://localhost:3000/api/v1/analysis/

```
{
    "uri": "https://learn-us-east-1-prod-fleet02-xythos.content.blackboardcdn.com/5cc71db6522fe/
        49399682?X-Blackboard-S3-Bucket=learn-us-east-1-prod-fleet01-xythos&
        X-Blackboard-Expiration=1737698400000&
        X-Blackboard-Signature=Qc%2Bme7U%2BJWZ6QgJB%2FnW7cnILelbkHwdPgPu6wJ8qsXE%3D&
        X-Blackboard-Client-Id=100219&X-Blackboard-S3-Region=us-east-1&
```

200 OK • 1.61 s • 467 B

```
{
    "text": "This is a lot of 12 point text to test the ocr code and see if it works on all types
        of file format. The quick brown dog jumped over the lazy fox. The quick brown dog jumped
        over the lazy fox. The quick brown dog jumped over the lazy fox. The quick brown dog
        jumped over the lazy fox."
}
```

Deliverables:

- Demonstrated successful invocation of the API with example inputs.
- Verified output using both Postman and CURL.

Hard-Coding Credentials is a Bad Idea Because,

- Hard-coded credentials can lead to:
  - **Security vulnerabilities** if the code is exposed.
  - **Accidental leaks** if pushed to version control systems like Git.
- Best practices include:
  - Using environment variables or secret management tools.
  - Rotating credentials regularly.
  - Applying least-privilege access to reduce risk.

Final Deliverables

1. Successfully connected to the Azure Vision API.
2. Demonstrated the working Flask API endpoint using Postman and CURL.
3. Committed the code to GitHub without exposing sensitive credentials.
4. Secured credentials using a .env file.