MP8 Overview Session

CS 340 - Introduction to Computer Systems
Goals

In this MP:

- Build the **middleware** and **backend** for a stateful web server to explore the **Mandelbrot** set
- Use **Docker** to launch a S3 compatible object storage
- Use AWS boto3 library for accessing your **object** storage
Mandelbrot Microservice Overview

- **/mandelbrot**: If there is a **GET** request on the route
  
  `/mandelbrot/<colormap>/<real>:<imag>:<height>:<dim>:<iter>`, it will return the mandelbrot set generated off those parameters

```
/mandelbrot/cividis/-0.7
435:0.126129:0.00018972
901232843951:256:1024
```
Flask Overview

- **Provided Code:** The provided code can be found in `app.py` and will give you the two routes already defined that will render the frontend

```python
from flask import Flask, jsonify, send_file, render_template, request
import requests
import os
import io
import boto3
import base64

app = Flask(__name__)

@app.route('/
def index():
    return render_template('index.html')

@app.route('/all')
def all():
    return render_template('all.html')
```

Found the app using `python -m flask run`
- Set the FLASK_DEBUG environment variable to 1 to run in debug mode

Visit [https://127.0.0.1:5000/](https://127.0.0.1:5000/) to the application
**Helpful Functions & Modules**

**Programming in Python**

- **boto3.client**: Will initialize an s3 client
- **boto3.list_objects**: Will list all the objects stored in a given bucket
- **boto3.download_fileobj**: Can be used to retrieve an object out of the storage system
- **boto3.upload_fileobj**: Can be used to store an object in the s3 storage
- **send_file()**: Can be used to return the bytes of a file in a response
- **how to run MinIO**: `docker run -it --rm -p 9000:9000 -p 9090:9090 --name minio -e "MINIO_ROOT_USER=ROOTNAME" -e "MINIO_ROOT_PASSWORD=CHANGEME123" quay.io/minio/minio server /data --console-address :9090` (use this command to run a local instance of MinIO before running your stateful server)
MP8 Part 3

Creating a stateful web server
Maintain State in Server

- **Center real**: center point used in the real axis
- **Center Imaginary**: center point used in the imaginary axis
- **Height**: contains the unit height that will be used when generating the mandelbrot set
- **Dimensions**: render dimensions of the image
- **Iterations**: maximum iterations of the mandelbrot set
- **Colormap**: Matplotlib colormap used to generate the image
Modifying Server State

- **POST /moveup**: moves the center of the image up by 25% of the current height
- **POST /moveDown**: moves the center of the image down by 25% of the current height
- **POST /moveLeft**: moves the center of the image to the left by 25% of the current height
- **POST /moveRight**: moves the center of the image to the right by 25% of the current height
- **POST /zoomIn**: modifies the height by a factor of $1 / 1.4$
- **POST /zoomOut**: modifies the height by a factor of 1.4
Modifying Server State Cont.

- **POST /smallerImage**: modifies the dim of the image by a factor of $1 / 1.25$
- **POST /largerImage**: modifies the dim of the image by a factor of $1.25$
- **POST /moreIterations**: modifies the iter of the image by a factor of $2$
- **POST /lessIterations**: modifies the iter of the image by a factor of $1 / 2$
- **POST /changeColorMap**: changes the colormap to be equal to the colormap value in the JSON in the request's body
Generating the Mandelbrot Image

- **GET /mandelbrot:**
  - Check if a mandelbrot image with the same state values exists in the cache
    - Return the image if it exists in the cache
  - Make a request to the mandelbrot microservice if the image doesn’t exist
    - Store the returned image in the cache
    - Return the generated image
  - Respond with response code 200
State of the Cache

- **GET /storage:**
  - Return a JSON of every image stored as an array of entries
    - Each JSON object must contain a key which is the unique name for a given Mandelbrot image stored in the cache
    - Each JSON object must contain an image with base64-encoded PNG image binary data and 
      data:image/png;base64 prefix
MP8 Testing
MP8 Testing

- Run the tests by using `pytest`
  - You can specify a filter with `-k flag` after pytest to run a specific test
- To test locally, run the `docker` command to start the MinIO Instance
- Start the Mandelbrot Microservice
- Run your microservice last