MP6 Overview Session

CS 240 - The University of Illinois

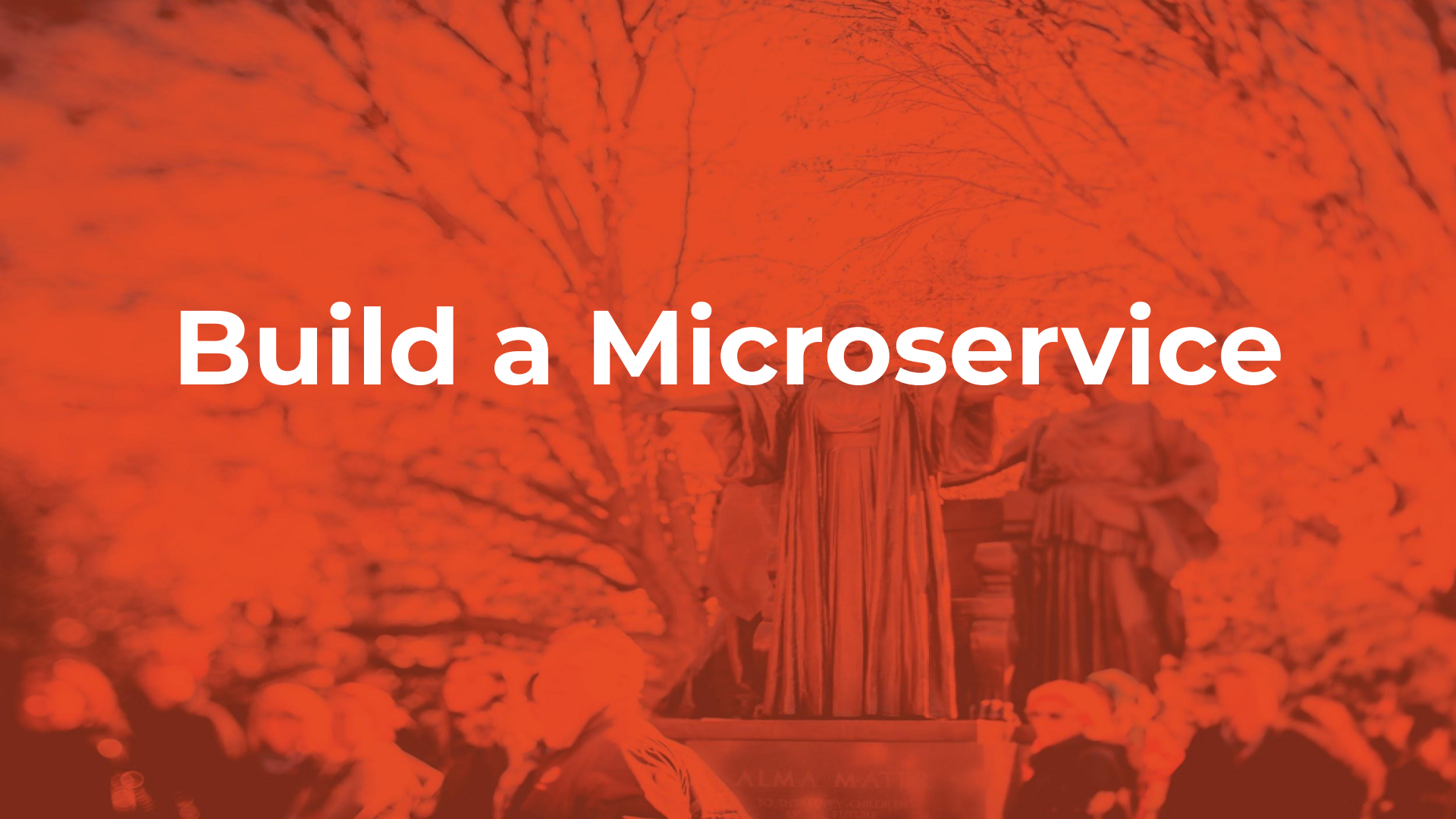
Eunice Zhou

March 7, 2022

Goals

In this MP, you will:

- create a web-based microservice that uses the program you have implemented in MP2
- build a flask application that will extract hidden GIFs from PNGs
- (extra credit) package your flask application in a docker container



Build a Microservice

Your Microservice

Extract GIF

PNG File

Choose File natalia.png

Submit

natalia.png

Your Microservice

Extract GIF

PNG File

Choose File no-uiuc-chunk.png

Submit

no-uiuc-chunk.png

Error 500
No uiuc chunk!

Flask

Your microservice will be a flask application. The template flask code is provided for you in *app.py*:

```
from flask import Flask
app = Flask(__name__)

# Route for "/" for a web-based interface to this microservice:
@app.route('/')
def index():
  return render_template("index.html")

# Extract a hidden "uiuc" GIF from a PNG image:
@app.route('/extract', methods=["POST"])
def extract_hidden_gif():
  # ...your code here...
```

Running Flask

- To run your flask program, run *python3 -m flask run*
  - You can also enable debug mode (auto reload on code change, better debugging info)
- Visit http://127.0.0.1:5000/ to see your application

Using Your MP2

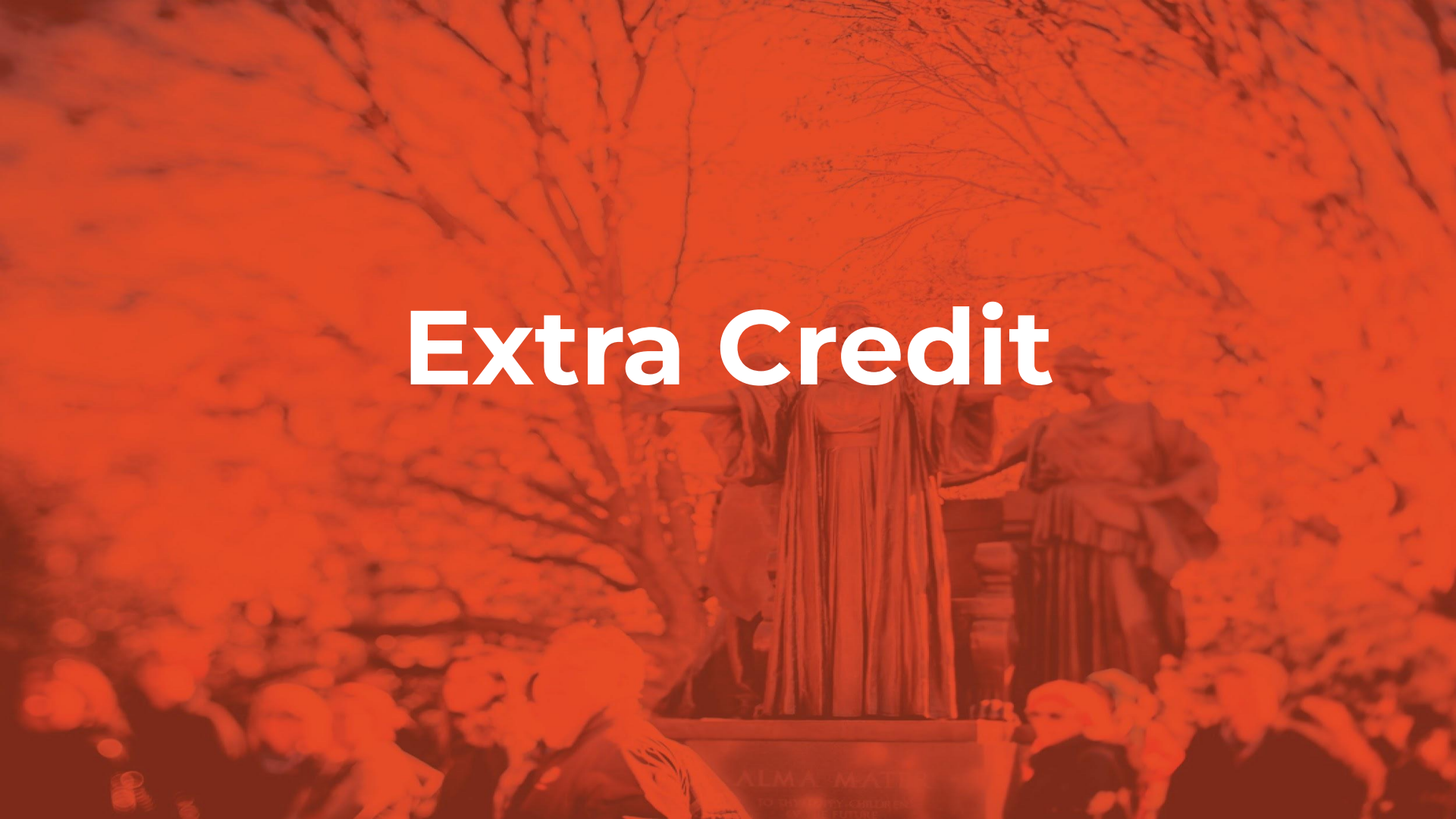
- Your flask application will use your *png-extractGIF* from MP2
- We provide a *Makefile* so you can run *make* to compile the *png-extractGIF* program using your MP2 code
- Test the program first to ensure it is complete and functional
  - Test using a PNG with a hidden GIF
  - Test using a PNG without a uiuc chunk (the program should run and not crash)
- The program should return 0 on success, non-zero on failure (including no uiuc chunk)

Your Tasks

- Create a *temp* directory if it doesn't exist
  - Useful function: *os.makedirs*
- Save the contents of the POST request (the png file) to the *temp* directory
  - You can access the png with *request.files['png']*
- Programmatically run the *./png-extractGIF* program
  - Useful function: *os.system*
  - The function returns the exit value of *./png-extractGIF*, which should be 0 on success and non-zero on failure

Your Tasks (cont.)

- If a GIF file is extracted, send the GIF in the response
  - Useful function: *send_file*
- If anything goes wrong, return an error as the response
  - *return “replace_this_with_a_good_error_message”, 500*



Extra Credit

Package as a Docker Container

For +5 extra credit points, you will wrap your flask application in a docker container.

- *Dockerfile* contains the instructions for building a docker image, which you will need to implement
- *.env* contains the commands to build and run the docker container
- *docker/entrypoint.sh* contains the instructions that will be run when the container is initiated

Useful Dockerfile Instructions

- *FROM* initializes a new build stage and sets the base image (e.g. python, ubuntu, gcc, etc.)
- *RUN* executes commands and commits the results to the container image
- *CMD* is the command the container executes by default when launching the built image
- *COPY* copies new files or directories from <src> to the container at the path <dest>

Useful Dockerfile Instructions

- *ENV* sets the environment variable <key> to the value <value>
- *ENTRYPOINT* is used to set executables that will always run when the container is initiated
- *WORKDIR* sets the working directory for the instructions that follow it in the Dockerfile

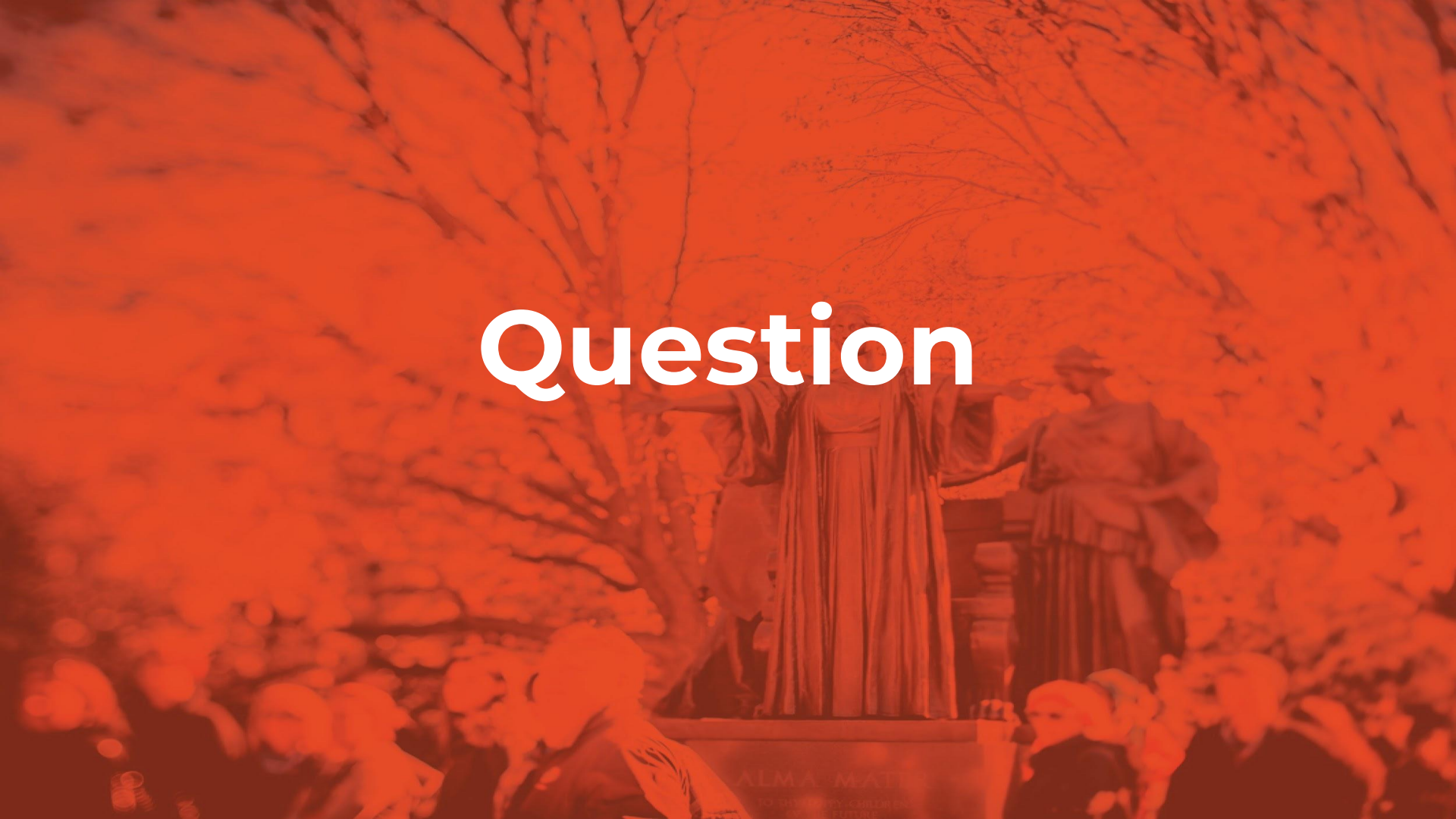
For more information, please refer to Dockerfile Reference and Best Practices for writing Dockerfiles in the official documentation

Testing

Run *pytest test_docker.py* to test your Dockerfile

Note that the test script doesn't automatically kill the docker container it launches. If you get an error saying the port is already occupied, you need to manually kill the container through

- either the Docker Desktop interface
- or the command line
  - *docker container ls*
  - *docker kill <container-name>*


Question