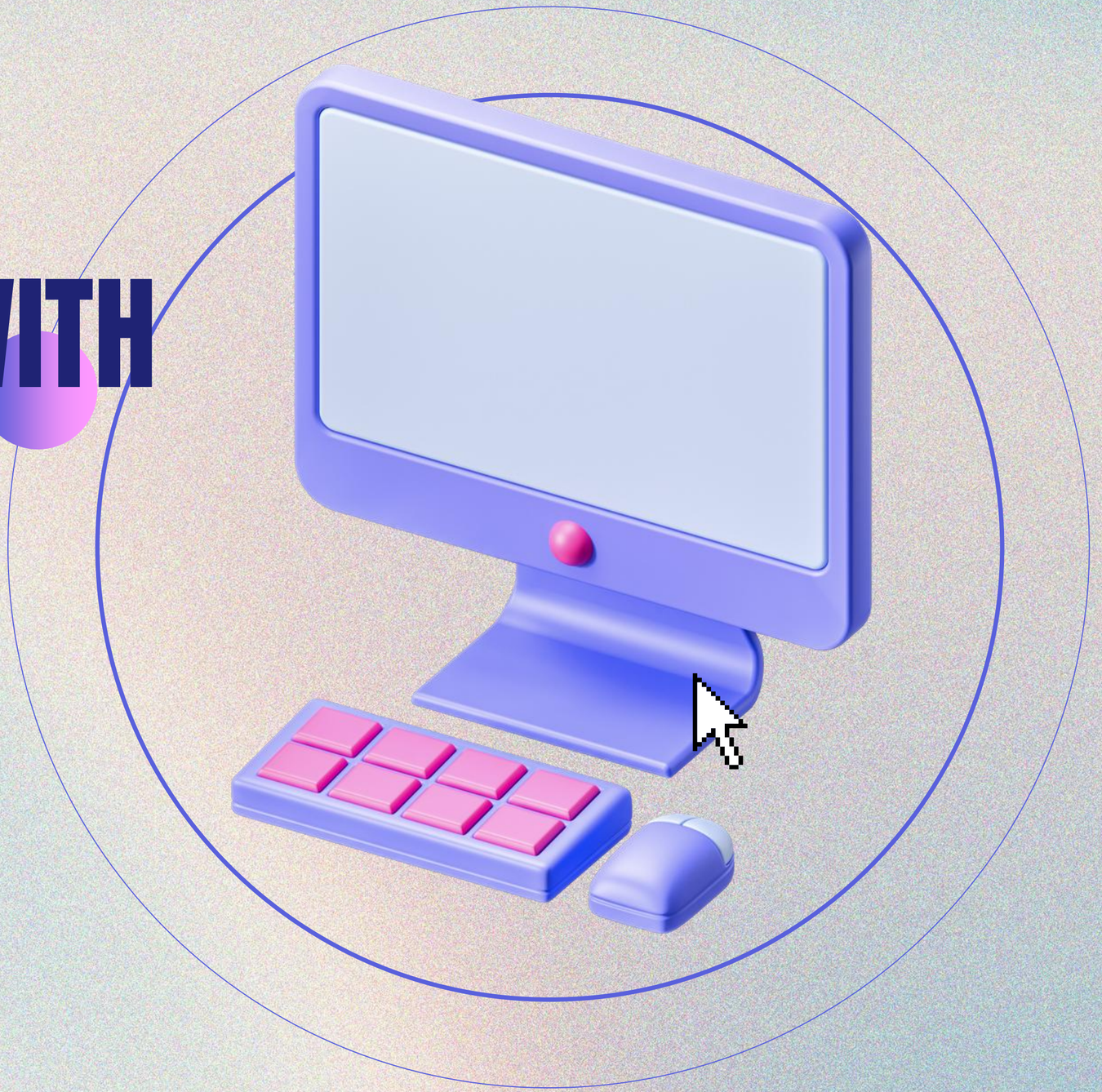
DOCKER COMPOSE WITH MULTI-CONTAINER APPLICATION

GROUP 5



INTRODUCTION

In this project, we learned how to use Docker to run a simple website using Flask and store data in Redis. We created containers for both Flask and Redis so they can work together easily. The project helped us understand how to run multiple services, link them, and test that they work using Docker Compose in a clear and practical way.

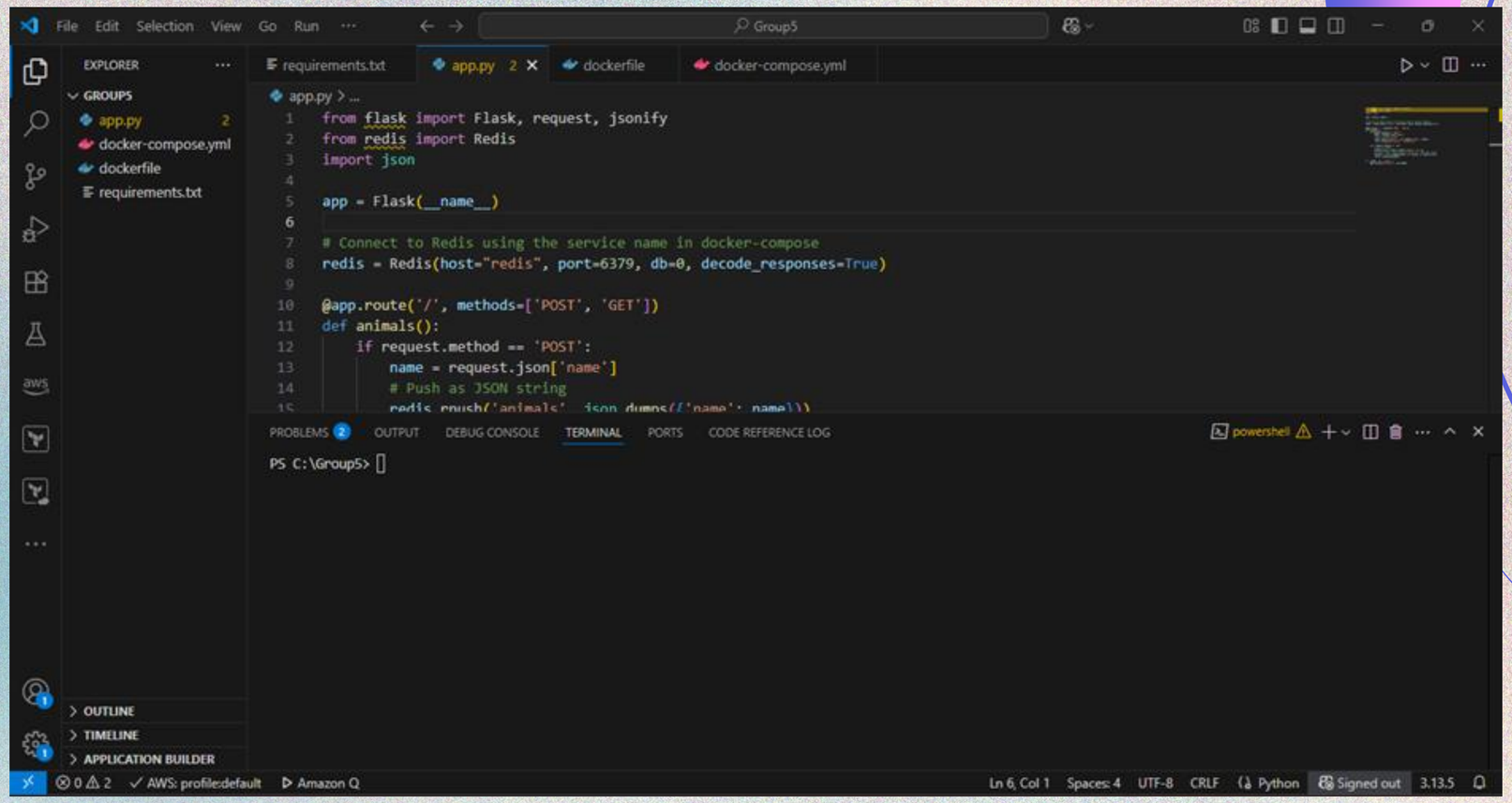
PROJECT OBJECTIVES

The objectives of the project was to;

- Containerize a Python Flask app using Docker.
- Use Redis as a database for data storage and retrieval.
- Link multiple services using Docker Compose.
- Test connectivity between services.

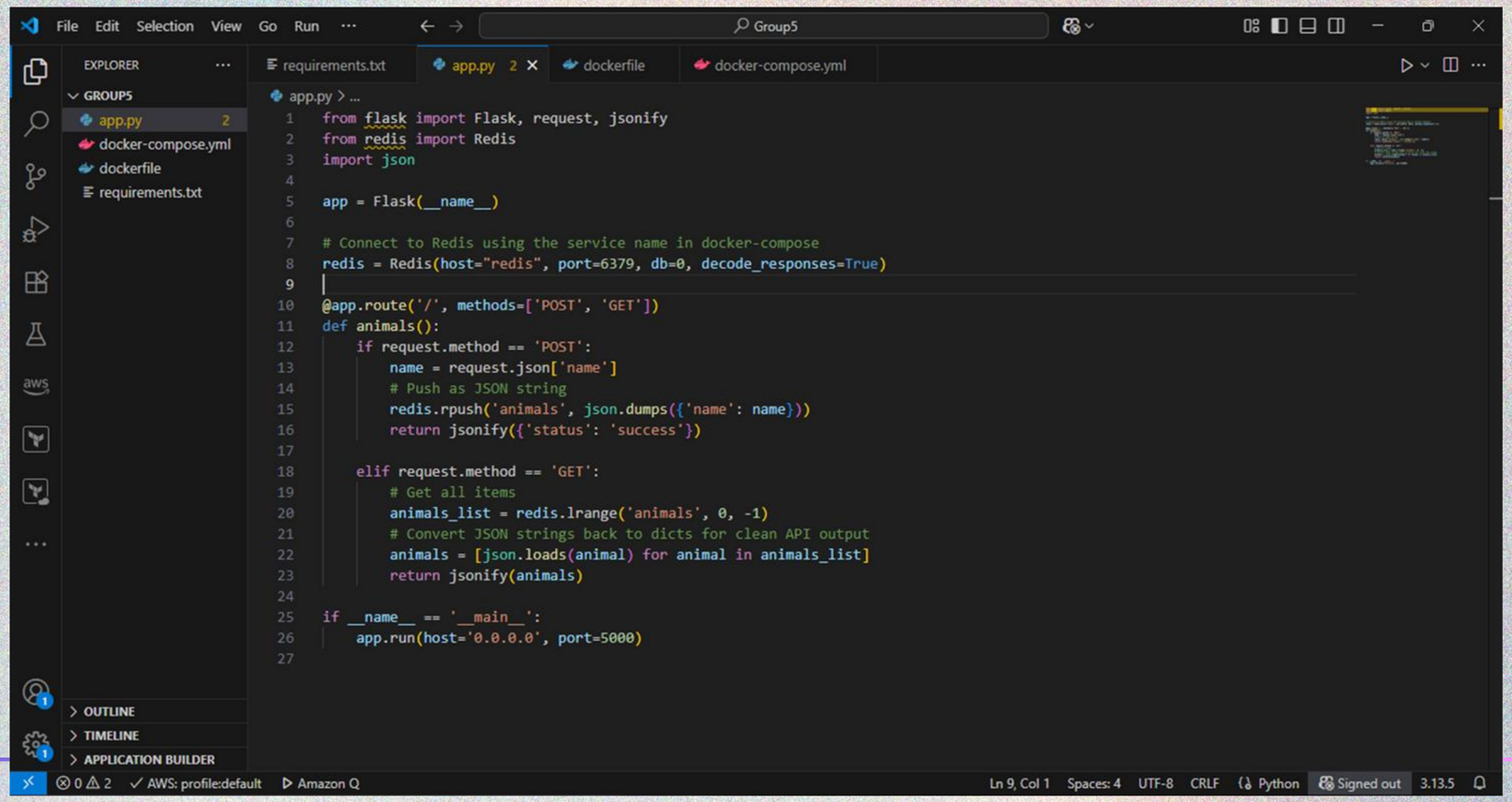
TASK 1: CREATE A PROJECT DIRECTORY

- Created a directory called Group5 to hold our files namely: app.py, requirements.txt, dockerfile and docker-compose.yml.



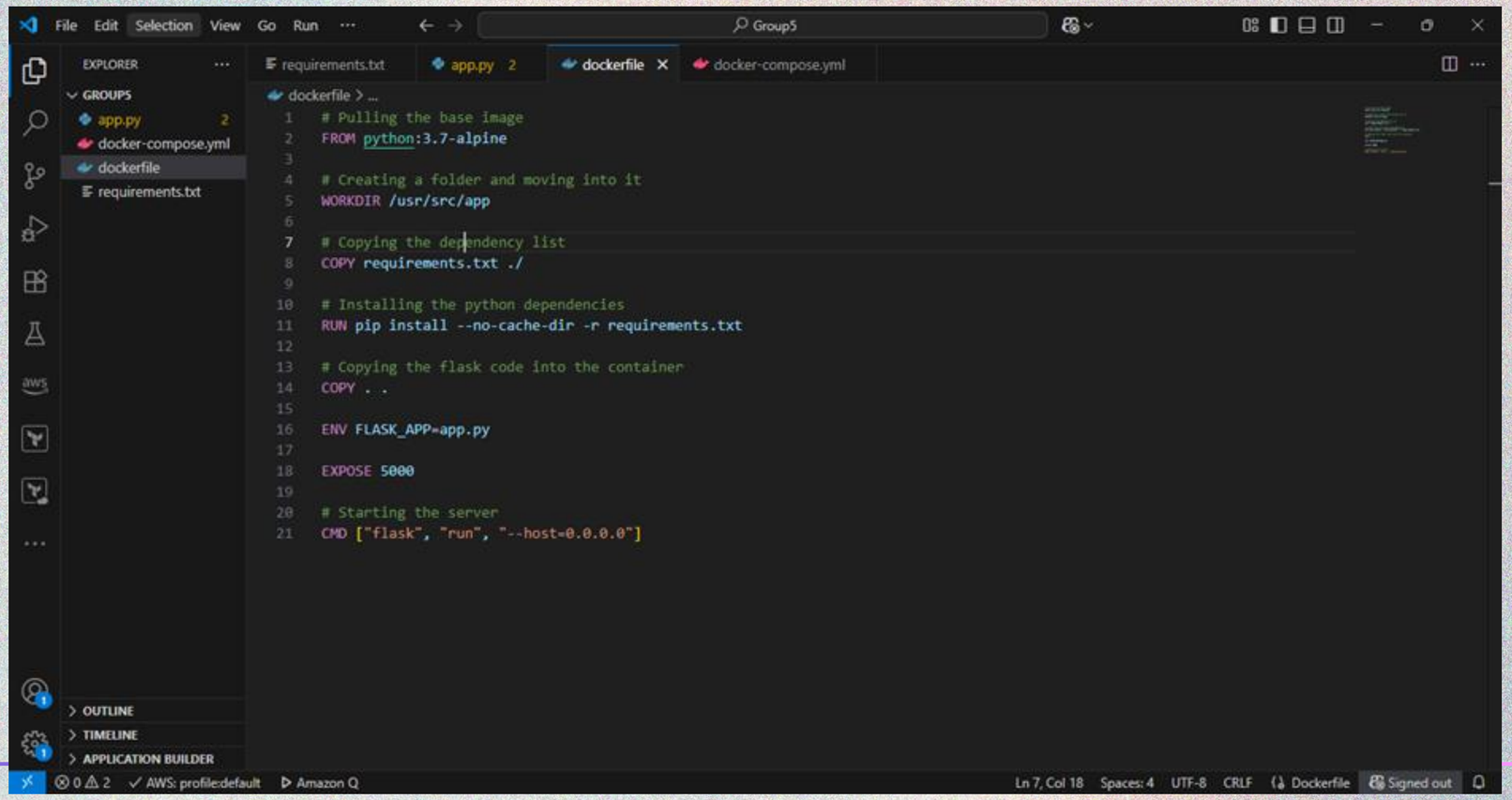
TASK 2: CREATE A WEB SERVICE

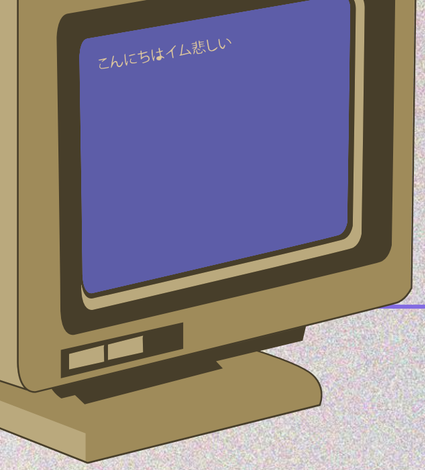
- We developed a basic Python Flask web application (app.py) and wrote a Dockerfile (dockerfile) to containerize it.



TASK 2: CREATE A WEB SERVICE

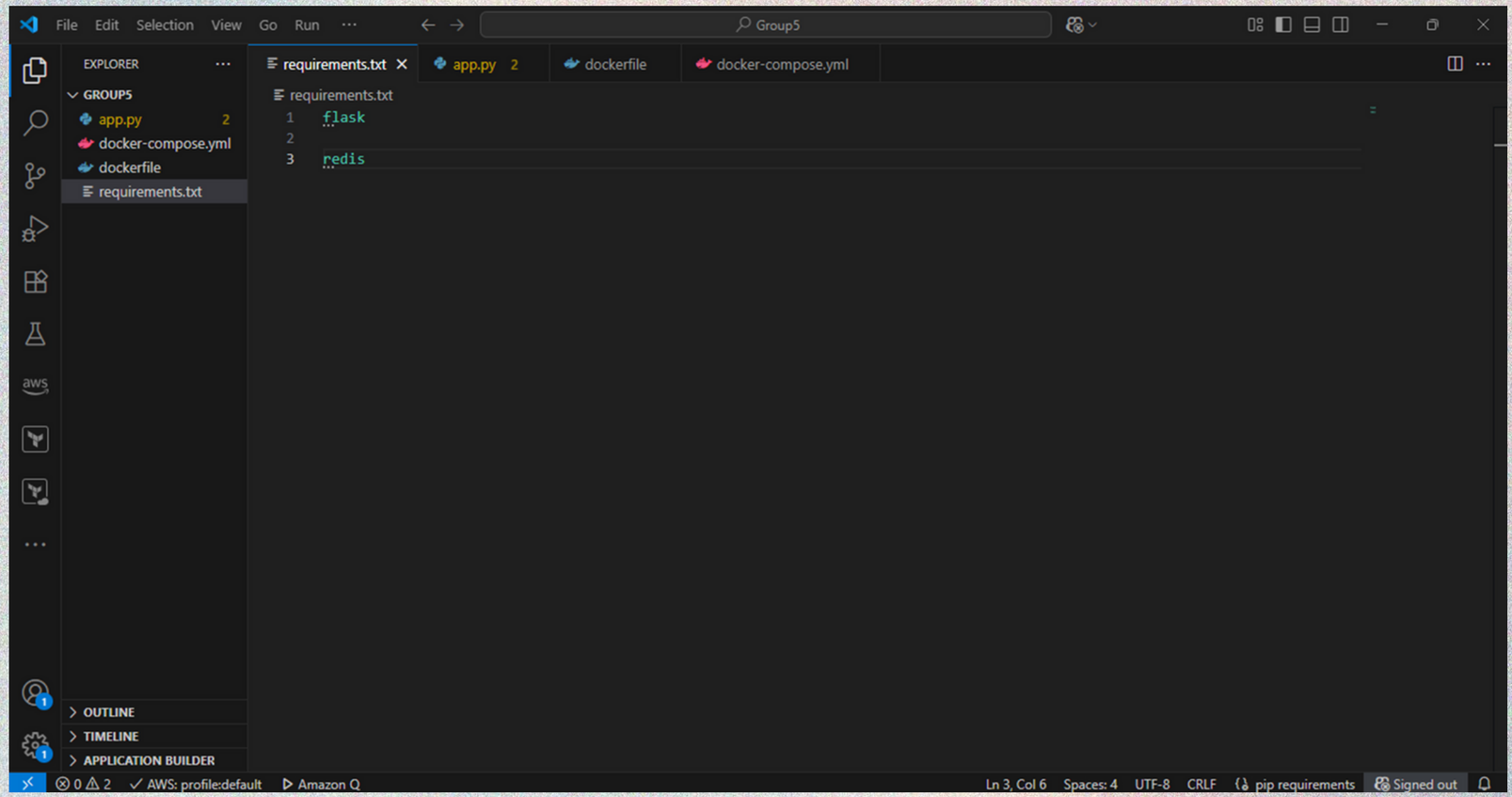
- We developed a basic Python Flask web application (app.py) and wrote a Dockerfile (dockerfile) to containerize it.

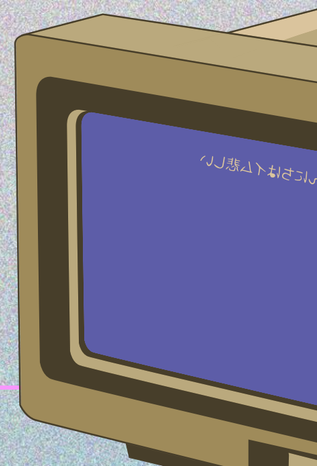




TASK 2: CREATE A WEB SERVICE

- We created requirements.txt to list needed packages.





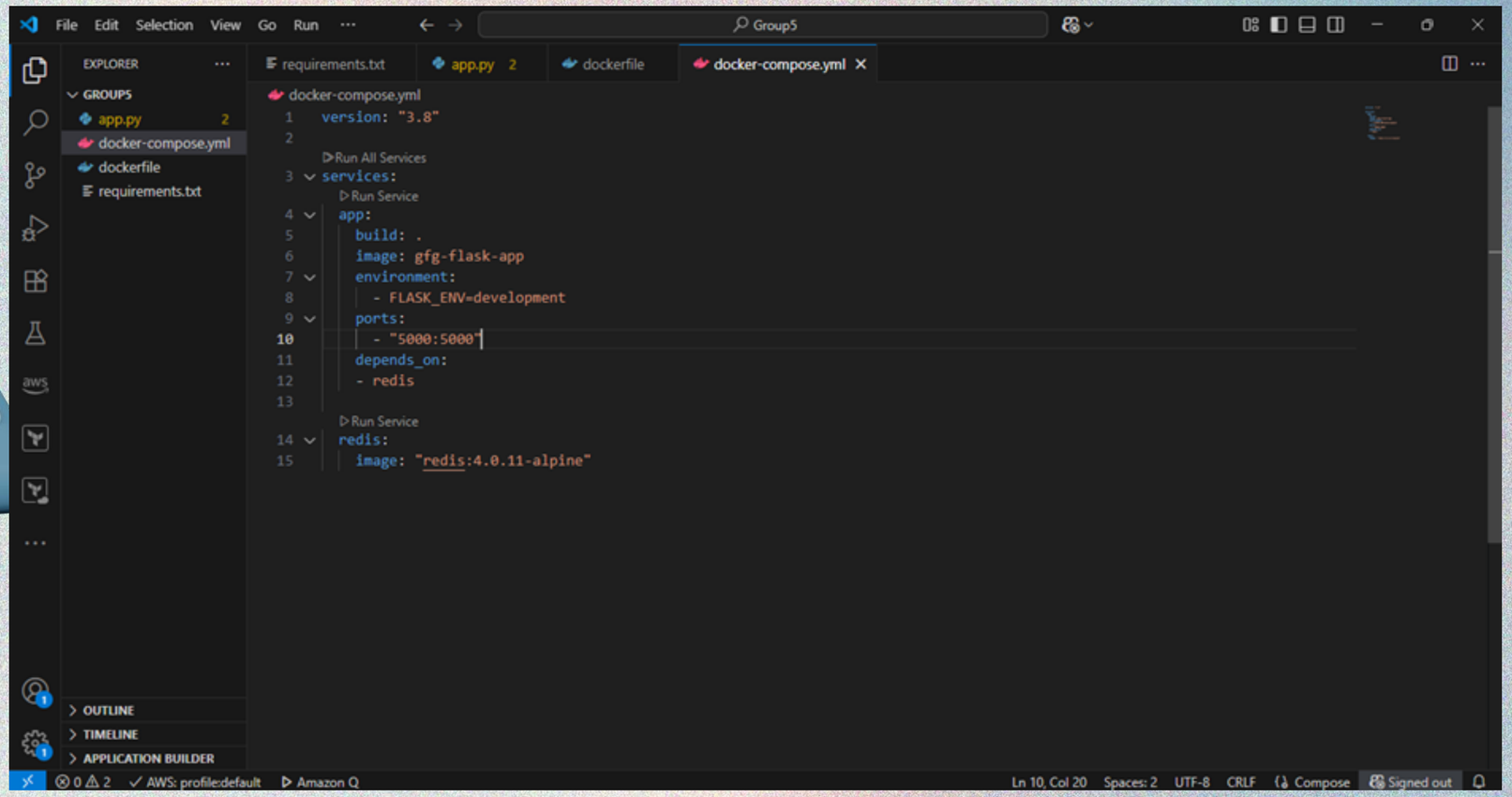
TASK 3: SET UP REDIS AS A DATABASE

- We used the official Redis image with no additional configuration.

```
    ▷ Run Service
14  redis:
15    image: "redis:4.0.11-alpine"
```

TASK 4: LINK SERVICES IN DOCKER COMPOSE

- We created docker-compose.yml to connect Flask and Redis.



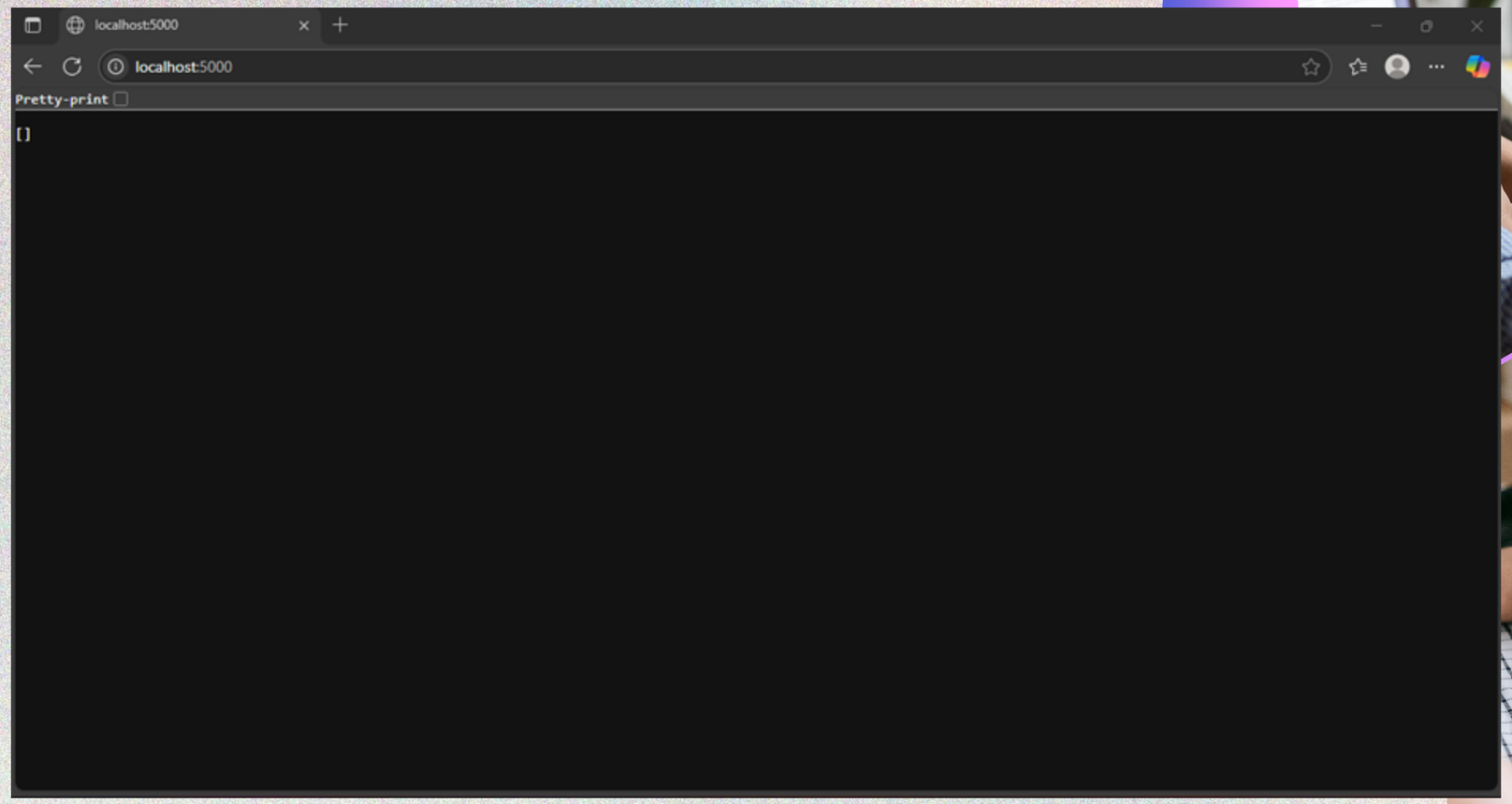
TASK 5: RUN AND TEST

- We used **docker-compose up** to start the app and Redis.

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CODE REFERENCE LOG

PS C:\Group5> docker-compose up
time="2025-07-11T10:56:09Z" level=warning msg="C:\\Group5\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please
 remove it to avoid potential confusion"
Attaching to app-1, redis-1
redis-1  | 1:C 11 Jul 10:56:11.052 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis-1  | 1:C 11 Jul 10:56:11.053 # Redis version=4.0.11, bits=64, commit=00000000, modified=0, pid=1, just started
redis-1  | 1:C 11 Jul 10:56:11.053 # Warning: no config file specified, using the default config. In order to specify a config file use redis-
server /path/to/redis.conf
redis-1  | 1:M 11 Jul 10:56:11.058 * Running mode=standalone, port=6379.
redis-1  | 1:M 11 Jul 10:56:11.059 # Server initialized
redis-1  | 1:M 11 Jul 10:56:11.060 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency an
d memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and ad
d it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
redis-1  | 1:M 11 Jul 10:56:11.060 * DB loaded from disk: 0.001 seconds
redis-1  | 1:M 11 Jul 10:56:11.060 * Ready to accept connections
app-1    |  * Serving Flask app 'app.py'
app-1    |  * Debug mode: on
app-1    | 'FLASK_ENV' is deprecated and will not be used in Flask 2.3. Use 'FLASK_DEBUG' instead.
```

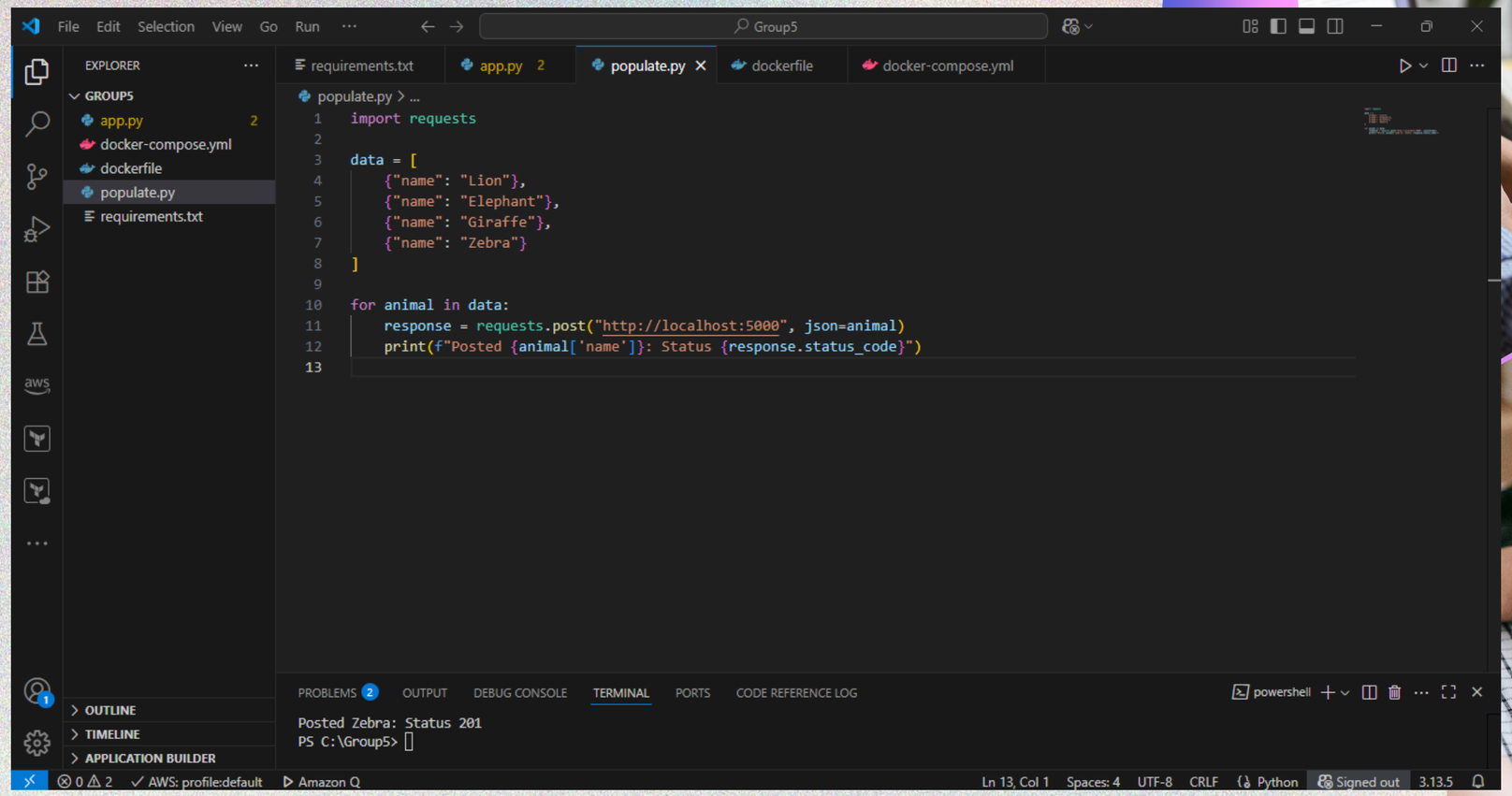
TASK 5: RUN AND TEST

- We visited **http://localhost:5000** in a browser for verification.



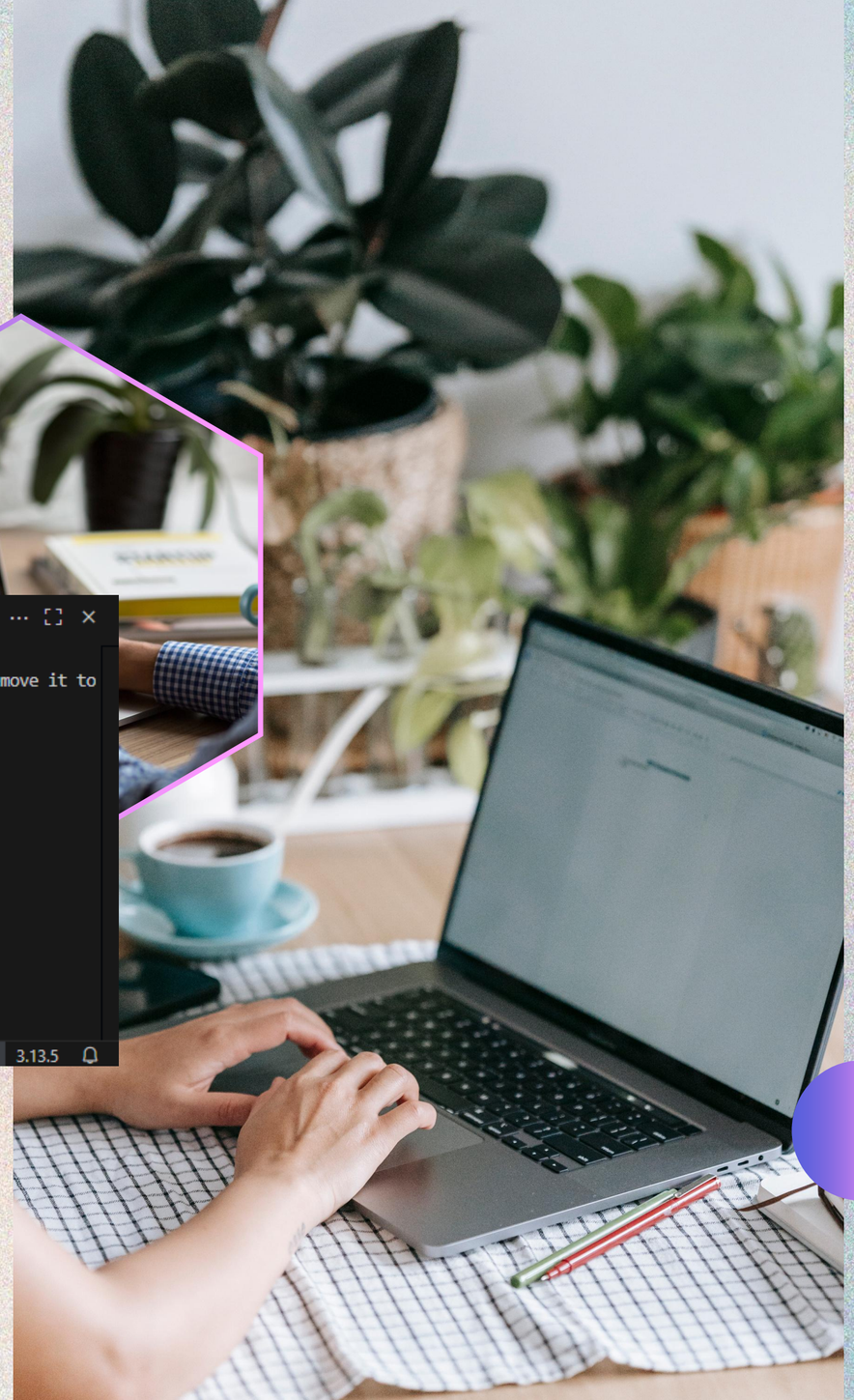
TASK 5: RUN AND TEST

- We tested adding data using a **populate.py** script.



TASK 5: RUN AND TEST

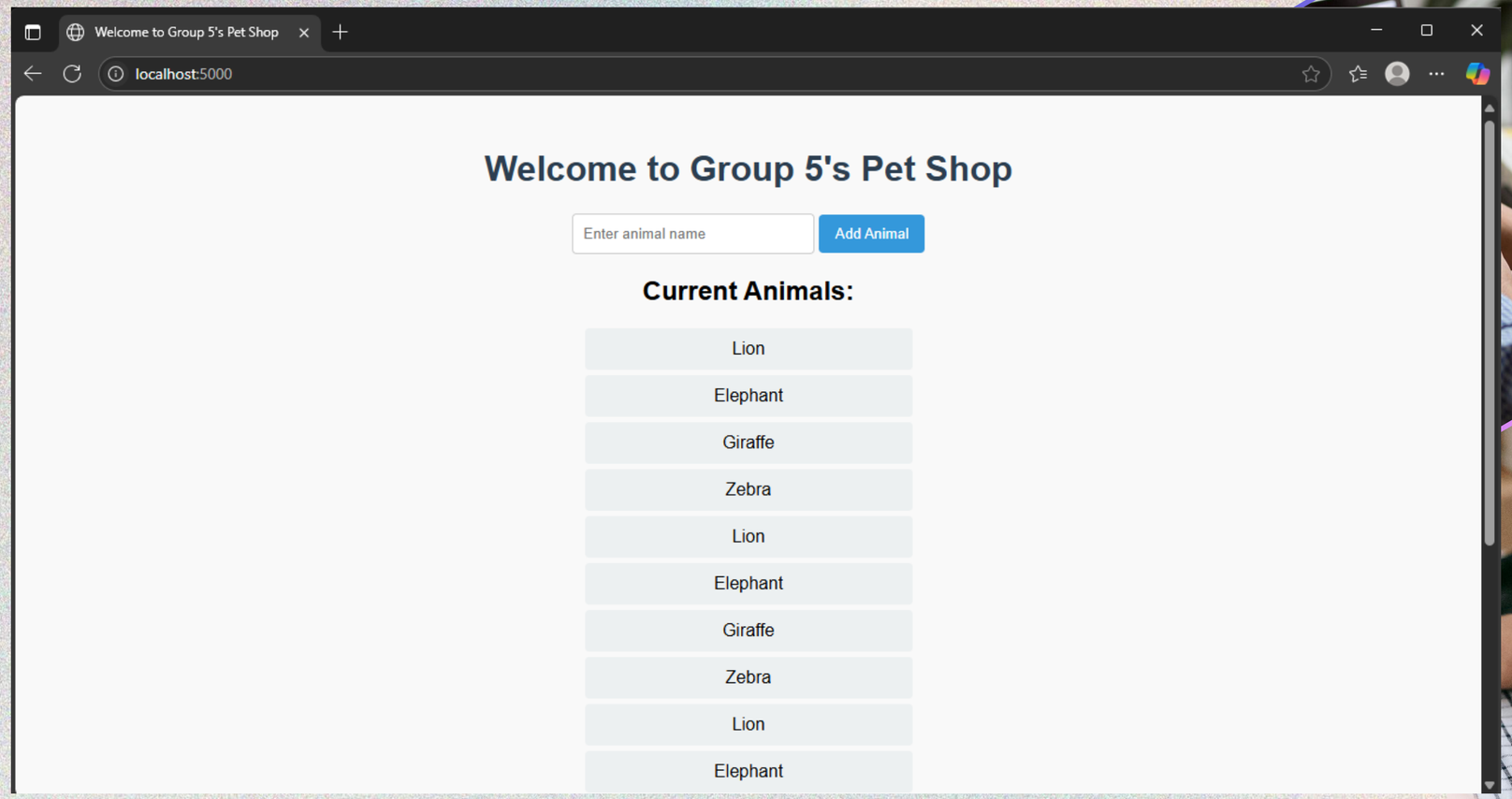
- We run our multi-container app with the command docker-compose up --build.

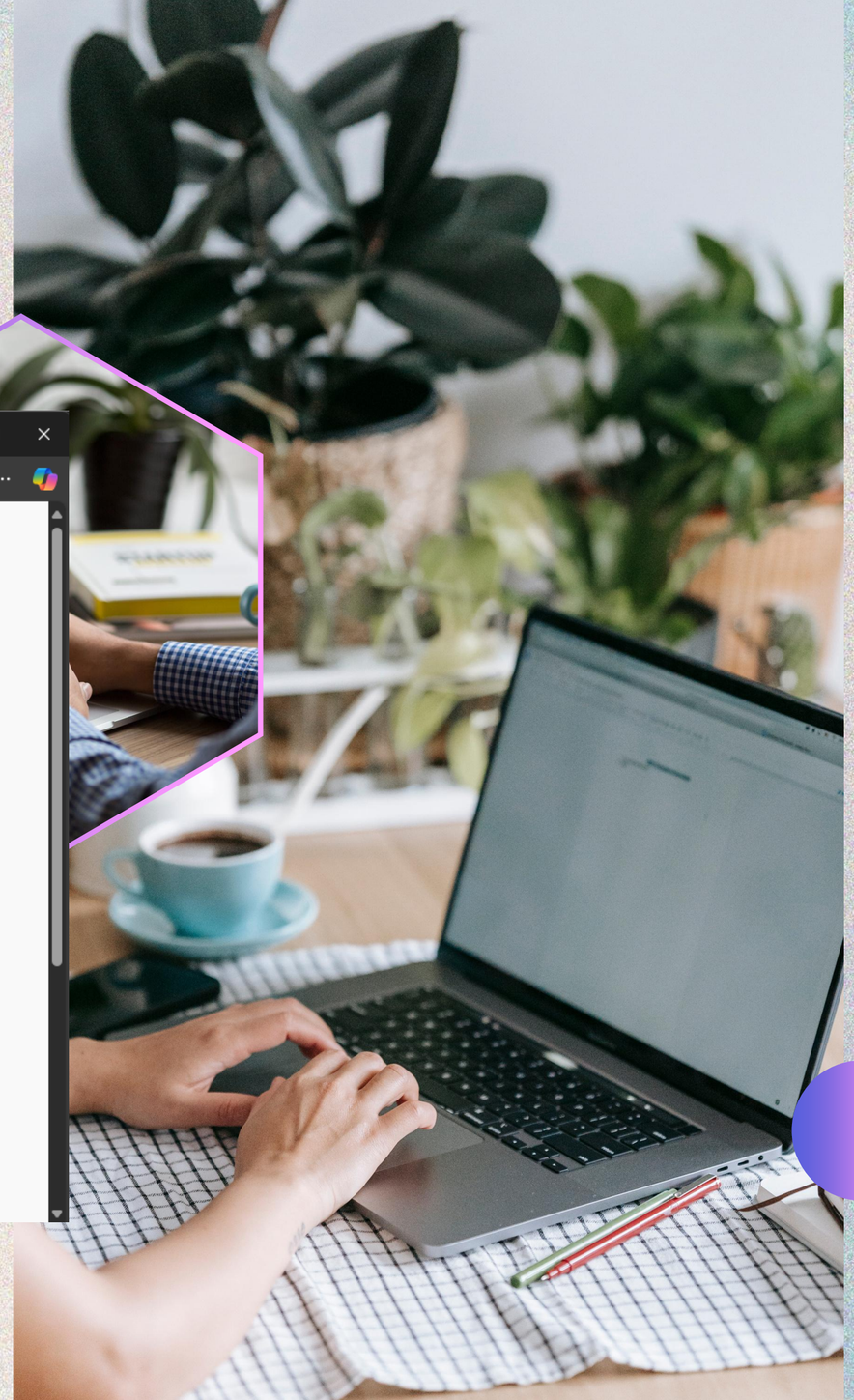


```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  CODE REFERENCE LOG        docker-compose

PS C:\Group5> docker-compose up --build
time="2025-07-11T16:00:52Z" level=warning msg="C:\\Group5\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to
 avoid potential confusion"
[+] Building 10.2s (13/13) FINISHED
 => [internal] load local bake definitions          0.0s
 => => reading from stdin 317B                      0.0s
 => [internal] load build definition from dockerfil 0.2s
 => => transferring dockerfile: 464B                0.1s
 => [internal] load metadata for docker.io/library/ 4.1s
 => [auth] library/python:pull token for registry-1 0.0s
 => [internal] load .dockerignore                   0.0s
 => => transferring context: 2B                     0.0s
 => [1/5] FROM docker.io/library/python:3.7-alpine@ 0.1s
 => => resolve docker.io/library/python:3.7-alpine@ 0.1s
 => [internal] load build context                   0.1s
 => => transferring context: 2.17kB                 0.1s
 => CACHED [2/5] WORKDIR /usr/src/app               0.0s

Amazon Q   Ln 52, Col 1  Spaces: 4  UTF-8  CRLF  Python  Signed out  3.13.5
```

TASK 5: RUN AND TEST

- The output after refresing our browser.





TASK 5: RUN AND TEST

- The output after running the command, python populate.py.

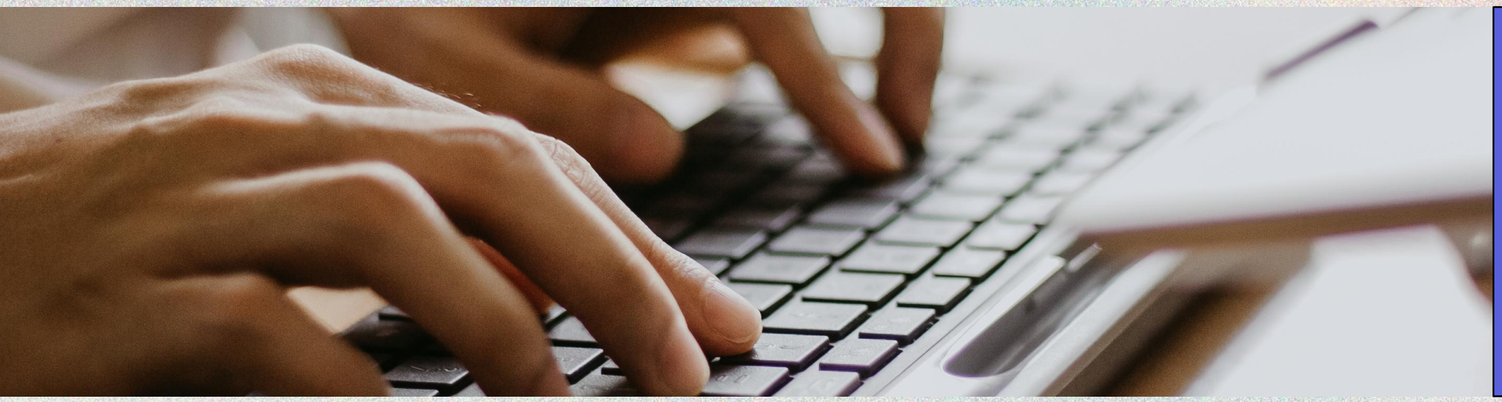
```
PROBLEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  CODE REFERENCE LOG        powershell

[+] Running 2/2
 ✓ Container group5-redis-1  Started                                    0.9s
 ✓ Container group5-app-1    Started                                    0.5s
PS C:\Group5> python populate.py
Posted Lion: Status 201
Posted Elephant: Status 201
Posted Giraffe: Status 201
Posted Zebra: Status 201
PS C:\Group5> python populate.py
Posted Lion: Status 201
Posted Elephant: Status 201
Posted Giraffe: Status 201
Posted Elephant: Status 201
Posted Giraffe: Status 201
Posted Giraffe: Status 201
Posted Zebra: Status 201
PS C:\Group5>

Amazon Q                Ln 15, Col 33  Spaces: 2  UTF-8  CRLF  {} Compose  Signed out
```

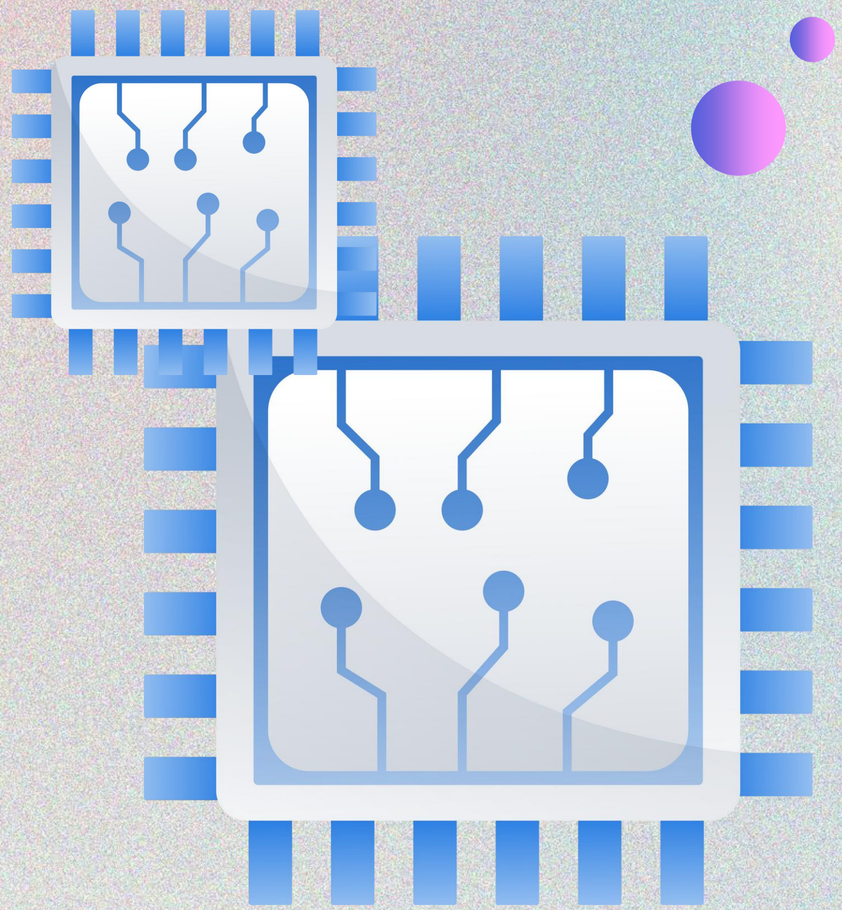
CHALLENGES

- We created another subfolder and placed the Dockerfile inside, causing build path issues.
- Python was not installed initially, requiring us to install Python before proceeding.
- We had to install the requests library to run populate.py successfully.
- Connection issues occurred when Redis was not recognized until the correct host and environment setup were used.

CONCLUSION

This project helped us understand how to build a simple web application and connect it to a database using containers. We practiced using Docker, Docker Compose, and basic Flask app development, preparing us for larger projects in cloud and container-based deployments.





MEET THE TEAM

- Macleana Mensah Oteng
- Esther Acheampong
- Selinam Fudzi-Amesu
- Clement Owusu Bempah
- Abigail Safoa
- Fahad Mohammed Gibrine

THANK YOU!