



## **CoDeKu Paid Mini Tasks**

<b>Project Code</b>	<b>CPT-DC-006</b>
<b>Project Name</b>	<b>Simple web app with flask</b>
<b>Your Name</b>	<b>Amalsha Fernando</b>
<b>Your Email</b>	<b>amalshaf6@gmail.com</b>
<b>Your WhatsApp Number</b>	<b>0712586284</b>
<b>Submission Date</b>	<b>29-07-2024</b>
<b>Task</b>	<b>Creating a simple Flask web application that displays "Hello, CoDeKu!", containerize it with Docker, and test it by running the container.</b>

## Contents

<b>Summary .....</b>	<b>3</b>
<b>Project Objectives.....</b>	<b>3</b>
<b>Project Timeline.....</b>	<b>3</b>
<b>Resources.....</b>	<b>3</b>
<b>Expected Outcomes .....</b>	<b>3</b>
<b>Methodology.....</b>	<b>4</b>
<b>1. Setting Up the Project Environment: .....</b>	<b>4</b>
<b>2. Developing the Flask Application: .....</b>	<b>4</b>
<b>3. Containerizing the Application: .....</b>	<b>5</b>
<b>4. Building and Running the Docker Container:.....</b>	<b>6</b>
<b>5. Testing the Application: .....</b>	<b>8</b>
<b>Conclusion .....</b>	<b>8</b>

## Summary

This project aims to develop a simple web application using Flask, a lightweight Python web framework, and containerize it using Docker. The application will display a "Hello,CoDeKu!" message on a web page. The primary goal is to demonstrate the process of building, containerizing, and deploying a web application in a consistent and reproducible environment.

In modern software development, containerization has become a standard practice for ensuring applications run consistently across different environments. Docker is a popular containerization platform that allows developers to package applications and their dependencies into a portable container. This project addresses the need for understanding and implementing a basic web application using Flask and Docker, which are widely used technologies in the industry.

## Project Objectives

### Project Timeline

Task	Spent Time Duration
Set up project environment	2 min
Develop Flask application	10 min
Create Dockerfile	2 min
Build and test Docker image	1 hour

### Resources

- Personnel: 1 developer
- Software: Python, Flask, Docker, vscode
- Hardware: Development machine with Docker installed

### Expected Outcomes

- A working Flask web application that displays "Hello, World!".
- A Docker container image for the Flask application.
- Documentation of the development and containerization process

# Methodology

## 1. Setting Up the Project Environment:

- Create a project directory named flask-docker-app.
- Set up a virtual environment and activate it:

```
python -m venv venv  
source venv/bin/activate
```

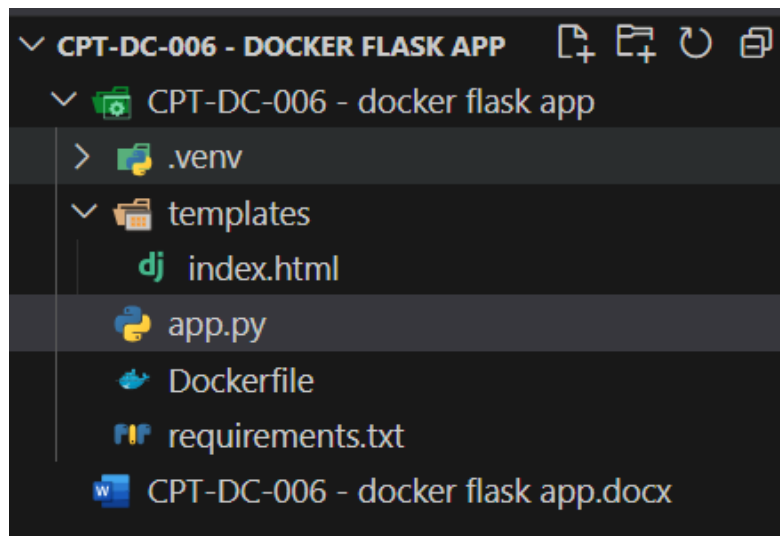


Figure 1- project structure

## 2. Developing the Flask Application:

- Create app.py with the following content:

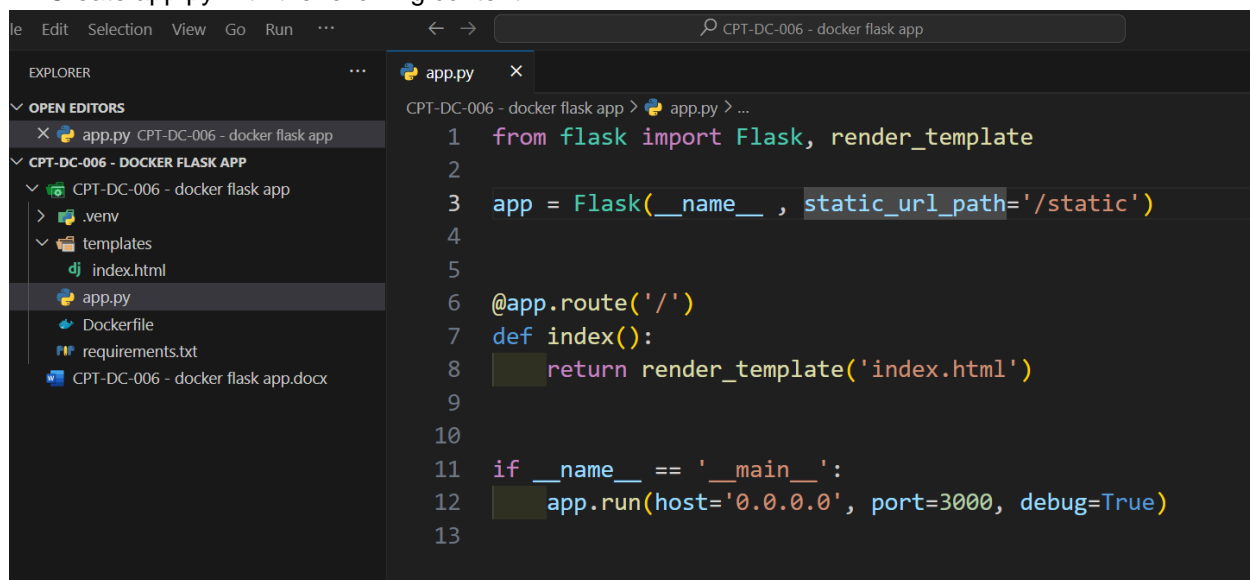
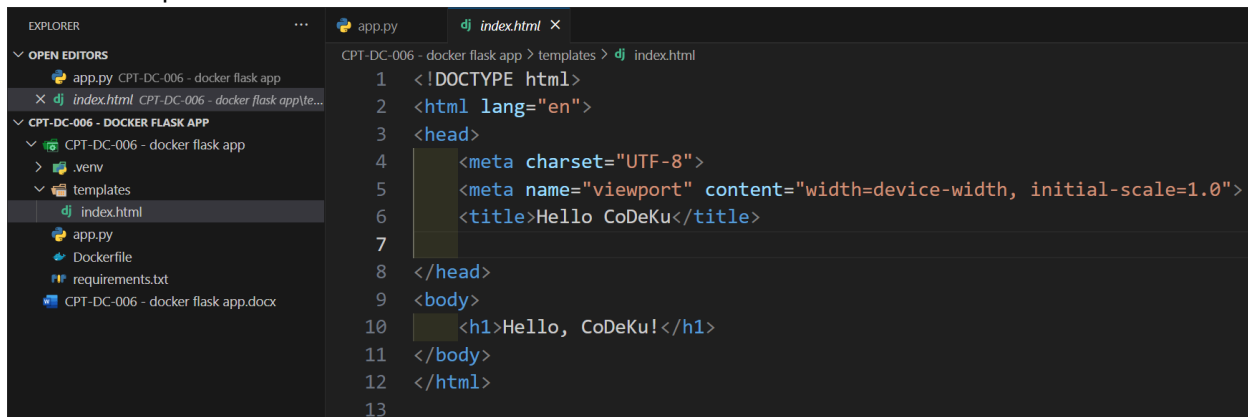


Figure 2

- Create templates/index.html:



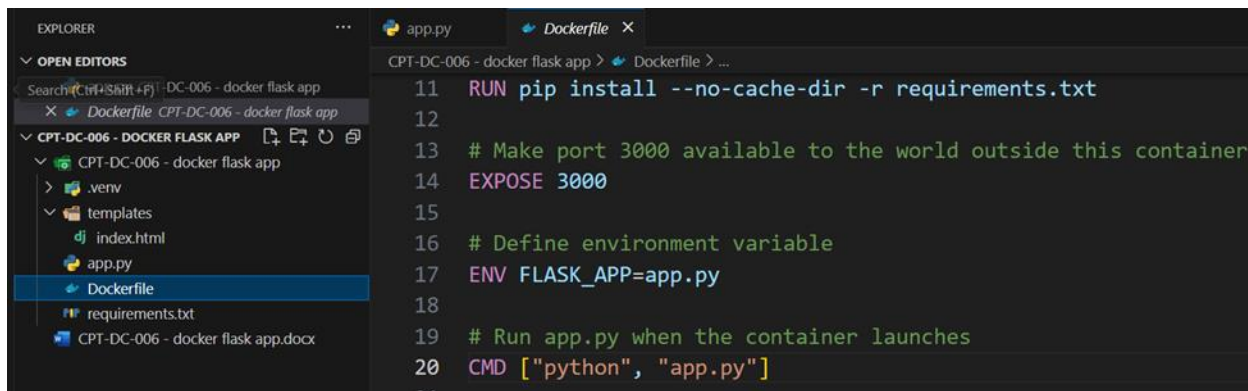
The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure for 'CPT-DC-006 - DOCKER FLASK APP', including files like app.py, Dockerfile, requirements.txt, and a templates directory. The templates directory is expanded, showing index.html. The main editor window has 'index.html' open, displaying the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Hello CoDeKu</title>
7
8 </head>
9 <body>
10    <h1>Hello, CoDeKu!</h1>
11 </body>
12 </html>
13
```

Figure 3

### 3. Containerizing the Application:

- Create a Dockerfile:



The screenshot shows the Visual Studio Code interface with the 'Dockerfile' open in the editor. The Explorer sidebar on the left shows the project structure, with 'Dockerfile' highlighted. The Dockerfile contains the following instructions:

```
11 RUN pip install --no-cache-dir -r requirements.txt
12
13 # Make port 3000 available to the world outside this container
14 EXPOSE 3000
15
16 # Define environment variable
17 ENV FLASK_APP=app.py
18
19 # Run app.py when the container launches
20 CMD ["python", "app.py"]
21
```

Figure 4

## 4. Building and Running the Docker Container:

### - Build the Docker image:

`docker build -t amalsha/hello_codeku:latest .`

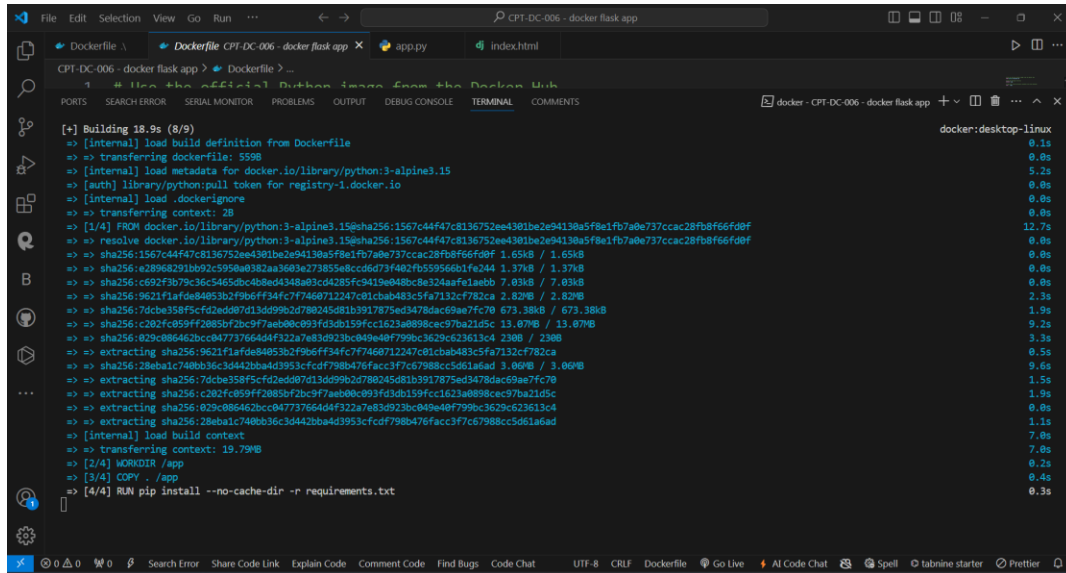
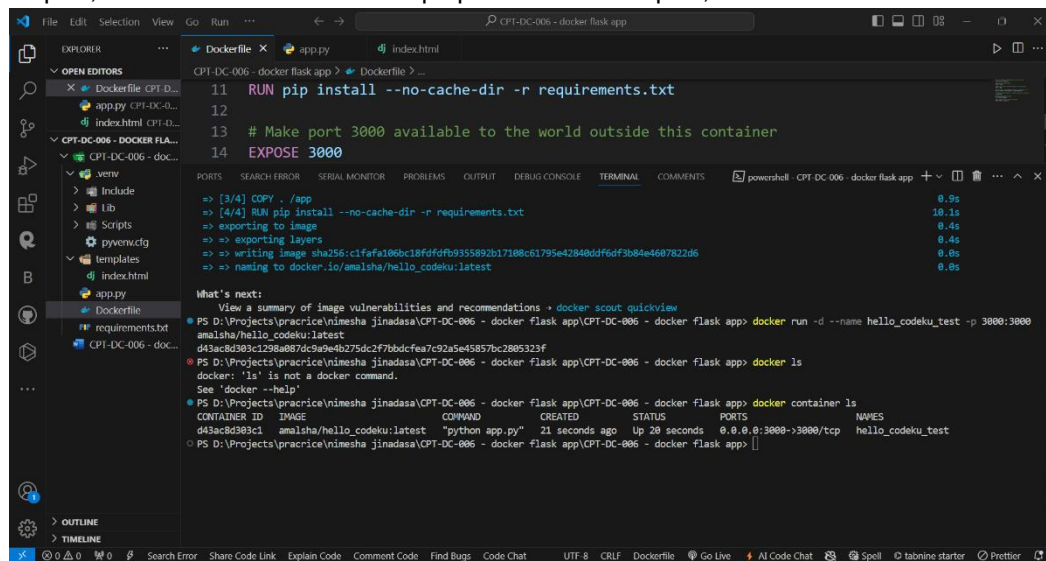


Figure 5

### - Run the Docker container:

- `docker run -p 3000:3000 flask-docker-app`
- here first port, 3000 is dedicated for local pc port and second port, 3000 is dedicated for second port



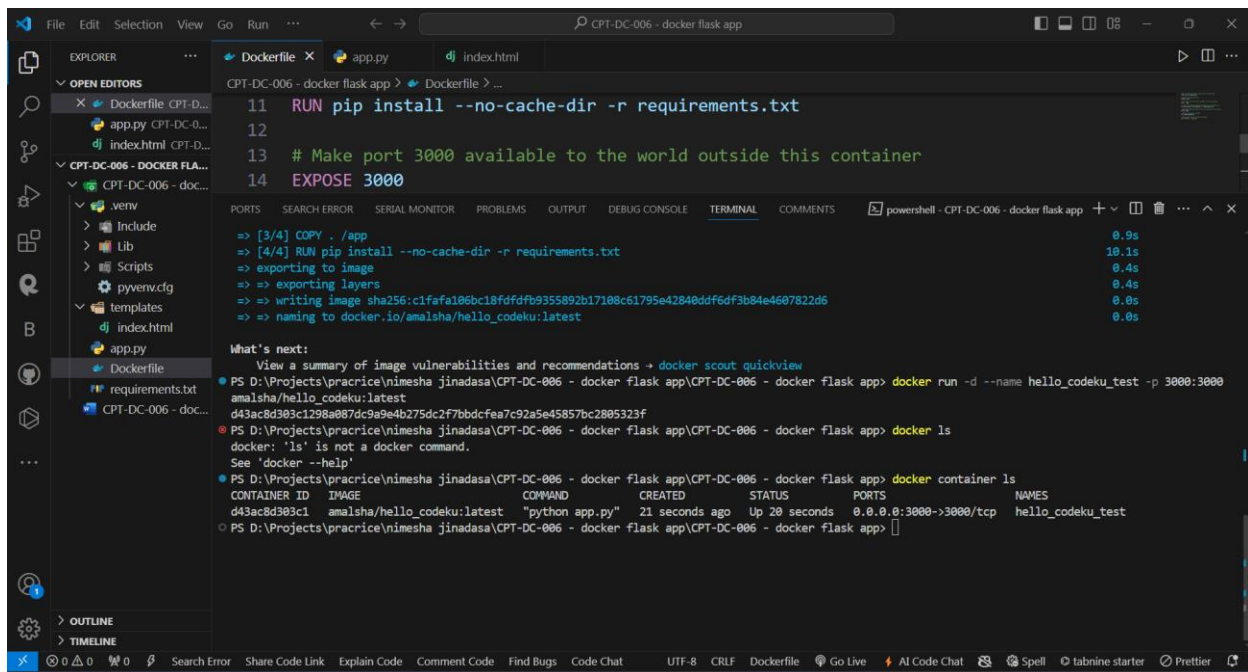


Figure 6

The docker ps command can show currently running containers. At first, it displays running containers, and after stopping containers, it shows nothing

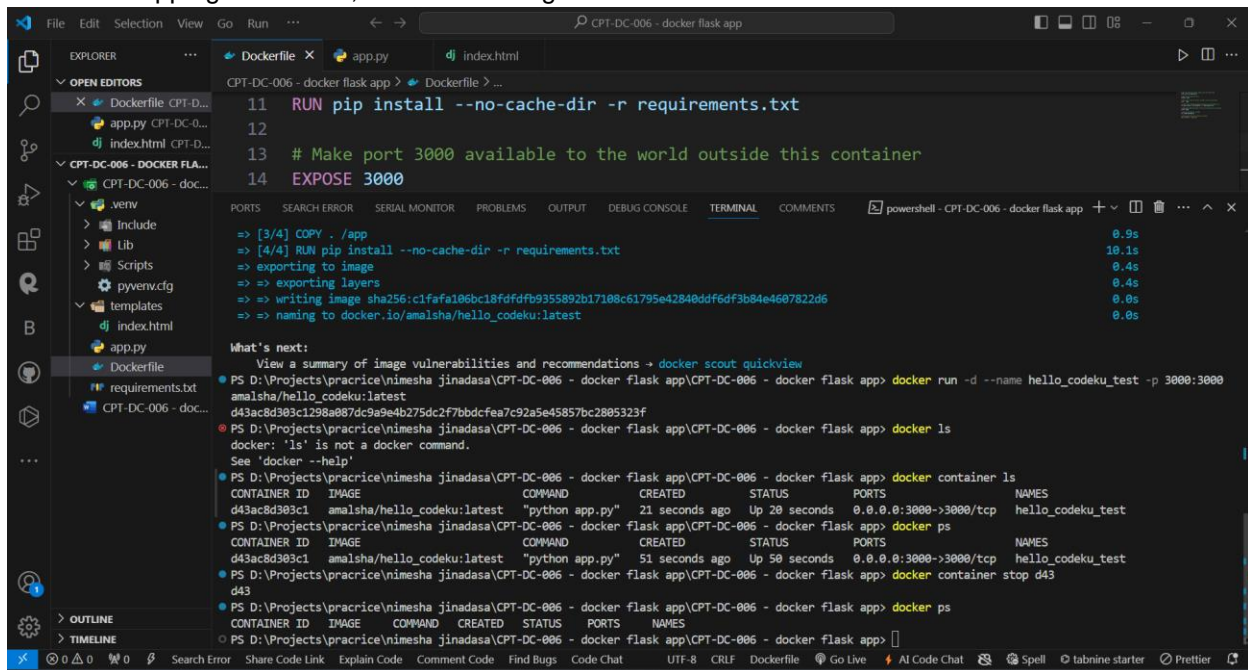
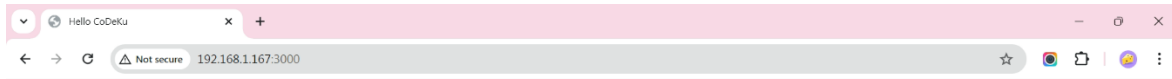


Figure 7

## 5. Testing the Application:

- When open a web browser and visit `http://localhost:5000` to see the "Hello, CoDeKu!" message rendered from the HTML file.



**Hello, CoDeKu!**

## Conclusion

This project will demonstrate the fundamental steps of developing and containerizing a simple web application using Flask and Docker. It serves as a foundational example for more complex applications and containerization processes.