**Dockerized Python To-Do application with docker compose**

**Objective**

You are given a Dockerized Python To-Do API application. Your goal is to:

* Build and run the app using Docker
* Validate its functionality by executing Python unit tests written in the Yaksha format

**Project Structure** you need to create all the below files in the same name

yaksha\_todo\_app/

app.py

requirements.txt

Dockerfile

docker-compose.yml

README.md

**Making the directory**

Mkdir yaksha\_todo\_app

cd yaksha\_todo\_app

Todo

In app.py, we will build the basic To-Do API using **Flask**.

Create app.py and implement the basic API with Flask. The app will have three endpoints:

* **GET /tasks**: To retrieve all tasks.
* **POST /tasks**: To create a new task.
* **PUT /tasks/{id}**: To mark a task as done.

**Dockerize the Flask Application**

1. **Create the Dockerfile**:
   * In the Dockerfile, define the steps for setting up the Flask app inside a Docker container. This will include:
     + Installing Python dependencies.
     + Exposing port 5000 for the app.
     + Running the Flask app.
2. **Create the docker-compose.yml**:
   * This file defines the Docker container configuration for the Flask app, ensuring that it runs in a network and is accessible on port 5000.
3. **Build and Run the Docker Container**:
   * In your terminal, build the Docker container using:

docker-compose up --build -d

* + This command will build the app’s Docker image and run it as a container in the background.

**Create docker-compose.yml**

version: '3.8'

services:

todo-app:

build: .

ports:

- "5000:5000"

**Inside requirements.txt**

Flask==2.1.1

requests==2.27.1

Important commands

Start the Docker Containers: docker-compose up --build -d

Open your browser or use curl: curl <http://localhost:5000/tasks>

pip install -r requirements.txt

**Task list**

* **Test 1: Verify Container is Running**  
  Ensure that the Docker container is running and accessible. You will check if the container with the name yaksha-todo is listed using the docker ps command.
* **Test 2: Verify GET Empty Tasks**  
  Ensure that if no tasks are present, a GET request to /tasks returns an empty list with HTTP status code 200.
* **Test 3: Verify POST New Task**  
  Post a new task using the POST method and ensure the response contains the task in the format {"task": "task\_name", "done": False} and returns status code 201.
* **Test 4: Verify GET Task After POST**  
  After posting a task, send a GET request to /tasks and verify that the task appears in the response.
* **Test 5: Mark Task as Done**  
  Send a PUT request to /tasks/{id} with a task ID to mark a task as done and verify the done field is set to True.

**Test the API**

Use tools like curl, Postman, or browser:

* GET /tasks → <http://localhost:5000/tasks>
* POST /tasks → Add a task using a JSON body

curl -X POST -H "Content-Type: application/json" -d '{"task":"Learn Docker"}' <http://localhost:5000/tasks>

**Sample screenshot**

A screen shot of a computer

AI-generated content may be incorrect.

**To Execute the test case**

* To Run the testcase In the project folder you will be provided with file to run the testcases
* Use the command to python test.py

**To run and push the code to git**

* To run the testcase
* Use the command pytest filename.py
* Screenshot to push the application to github

### -----x-----

#### You can run test cases as many numbers of times and at any stage of Development, to check how many test cases are passed/failed and accordingly refactor your code.

* 1. A computer screen with icons on it

     Description automatically generatedMake sure before final submission you commit all changes to git. For that open the project folder available on desktop
     1. Right click in folder and open Git Bash

A screenshot of a computer

Description automatically generated

* + 1. In Git bash terminal, run following commands
    2. git status

A screenshot of a computer screen

Description automatically generated

* + 1. git add .

A black screen with yellow and purple text

Description automatically generated

* + 1. git commit -m “First commit”

(You can provide any message every time you commit)

A screenshot of a computer

Description automatically generated

* + 1. git push

A computer screen with white text

Description automatically generated