Full Stack Interview Takehome

Challenge Overview

You'll build a lightweight task management app with the following features:

- A **Flask-based backend** that supports task creation, completion, deletion, and statistics
- A **JavaScript frontend** (React or vanilla) that interacts with the backend API
- A **Dockerized environment** that allows everything to be run via `docker-compose`

These are Minimum Requirements. Try to impress us with the backend, frontend and additional features.

Requirements

1. Backend (Python + Flask)

Build a RESTful API with the following endpoints:

Method	Endpoint	Description
GET	`/tasks`	List all tasks
POST	`/tasks`	Create a task
PUT	`/tasks/<id>/complete`	Mark a task as completed
DELETE	`/tasks/<id>`	Delete a task
GET	`/tasks/stats`	Return stats (total, completed, pending)

Task fields:

```
json
CopyEdit
{
  "id": int or string,
  "title": string,
  "completed": boolean
}
```

Notes:

- Tasks can be stored in memory (e.g., a Python dict or list). No database is required.
- Validate incoming requests and return appropriate error messages (e.g., 404 on missing ID).
- Port: expose the Flask app on port **5000** in Docker.

---

2. Frontend (JavaScript — React or Vanilla)

Create a web UI with the following:

- A list of tasks displaying title and completion status
- A form/input to **add** a new task
- Buttons next to each task to **complete** or **delete** it
- A visible **statistics** section showing:
  - Total number of tasks
  - Number completed
  - Number pending

Notes:

- The frontend should make HTTP requests to the backend API.
- Keep styling minimal; focus on clarity and functionality.

- Expose the frontend on port **3000** in Docker.

---

3. Dockerization

Package both backend and frontend using Docker:

- Include a `Dockerfile` for the Flask backend
- Include a `Dockerfile` for the frontend (React preferred, but vanilla JS in `index.html` is fine)
- Include a working `docker-compose.yml` that:
  - Builds both services
  - Ensures the frontend can communicate with the backend
  - Allows the app to be run with:

```
docker-compose up --build
```

Sample API Usage

```
# Add a task
curl -X POST http://localhost:5000/tasks -H "Content-Type: application/json"
-d '{"title": "Write report"}'

# List tasks
curl http://localhost:5000/tasks

# Mark a task as complete
curl -X PUT http://localhost:5000/tasks/1/complete

# Delete a task
curl -X DELETE http://localhost:5000/tasks/1
```

```
# Get stats
curl http://localhost:5000/tasks/stats
```

Deliverables

Please submit either:

- A GitHub repo (public or private with access), **OR**
- A ZIP file containing:
  - `backend/` with Flask code and Dockerfile
  - `frontend/` with your frontend code and Dockerfile
  - A top-level `docker-compose.yml`
  - A `README.md` with:
    - How to build and run the app
    - Any assumptions or simplifications you made
    - Answers to these brief questions:
      - How did you handle API errors?
      - What tests would you write if given more time?
      - What would you improve with 1 extra hour?

Optional

Feel free to include one or two example tests (unit or integration) to show how you'd approach testing the API. Totally optional, but appreciated.

Time Expectation

This challenge is scoped for **~2 hours of work**. Don’t worry about polish or perfect structure — we’re more interested in how you approach problems and structure code.