DOCUMENTATION FOR BACKEND

A. Title:

Assignment Title: Better Associate Software Engineer (Python React)
Assessment – Backend
Name: Khan Mohammed Ammaar
Email: khanmohammedammr@gmail.com
Date: August 7, 2025

B. Introduction:

This document describes the backend portion of the assessment where I developed RESTful APIs using Flask for managing tasks and comments. The backend serves as the API provider for task CRUD operations and supports automated testing to ensure reliability.

C. Environment Setup:

- **Operating System:** Windows 10
- **Python Version:** 3.12.1
- **Virtual Environment:** Created using Python's venv module
- **Installed Packages:** Flask, flask-cors, flask-sqlalchemy, flask-restful, pytest (installed via requirements.txt)
- **Git:** Used for version control, connected to GitHub fork

Setup Steps:

1. Created a project folder and navigated into it.
2. **Created and activated the virtual environment:**

```
python -m venv venv

venv\Scripts\activate
```

3. **Installed dependencies using:**

```
pip install -r requirements.txt
```

4. Forked the template repo on GitHub and cloned to local machine.

D. Project Structure:

/project-root

|--main.py

|-- models.py

|-- requirements.txt

|-- tests/

|-- venv/

|-- client/ (React frontend)

- **main.py:** Main Flask application file with API routes
- **models.py:** Contains SQLAlchemy models for Task and Comment
- **requirements.txt:** Python packages list
- **tests/:** Contains pytest tests for backend API endpoints
- **venv/:** Virtual environment folder
- **client/:** React frontend code

E. Solution Approach:

Backend (Flask):

- Created virtual environment and installed dependencies.
- Defined SQLAlchemy models for Task (with fields: id, title, description, status, etc.) and Comment (linked via foreign key).
- Implemented RESTful endpoints for tasks:
    - POST /tasks to add a task
    - GET /tasks to retrieve all tasks
    - GET /tasks/<id> for a specific task
    - PUT /tasks/<id> to update a task
    - DELETE /tasks/<id> to delete a task

    - POST /tasks/<id>/comments to add comments
- Wrote automated tests using pytest covering success and error scenarios for all endpoints.
- Used Flask Blueprints and Flask-RESTful for modular route management.

Assumptions:

- Assumed status field values as “pending” and “completed”.
- Focused on basic CRUD functionalities; did not include authentication or advanced filtering.

F. How to Run the Backend:

1. Navigate to the project root folder.
2. **Activate virtual environment:**

python -m venv venv

venv\Scripts\activate

3. **Install dependencies (only if not done):**

pip install -r requirements.txt

4. **Run the Flask app:**

python main.py

5. Backend API will be accessible at http://127.0.0.1:5000/

G. Screenshots:C:\Users\Khan Mohammed Ammaar\OneDrive\Pictures\Screenshots 1

J. Summary:

This backend implementation provides clean and tested REST APIs for managing tasks and comments. I learned the importance of structuring Flask applications and writing automated tests. In future iterations, I would add user authentication, input validation, and deploy the app to the cloud.