General instructions

1. The approach of solving the Problem solely depends on the Candidate.
2. Make sure to have Draw.io diagrams for the workflows and application architecture.
3. Every configuration, code written should be pushed on git (Private Repo).
4. You are not permitted to share the doc with anyone, even with your colleagues.

Scenario

Task Management for a Software Development Team.

Imagine a software development team working on a complex project with multiple tasks and deadlines. They need a **Task Management API** to help them track and organize their tasks efficiently.

Your task is to understand the problem and create a **Task Management API** using **Go-gin Framework** as per the instructions.

Section A: Database Schema

The **SQLite database** schema for storing tasks will include the following fields:

- ID: The unique identifier for each task.
- Title: The title or name of the task.
- Description: A brief description of the task.
- Due Date: The deadline or due date for the task.
- Status: The status of the task (e.g., "pending," "completed," "in progress," etc.).

Section B: API Endpoints

Part 1: Create a new task

Endpoint: POST /tasks Accepts a JSON payload containing the task details (title, description, due date). Generates a unique ID for the task and stores it in the database. Returns the created task with the assigned ID.

Part 2: Retrieve a task

Endpoint: GET /tasks/{id} Accepts a task ID as a parameter. Retrieves the corresponding task from the database. Returns the task details if found, or an appropriate error message if not found.

Part 3: Update a task

Endpoint: PUT /tasks/{id} Accepts a task ID as a parameter. Accepts a JSON payload containing the updated task details (title, description, due date). Updates the corresponding task in the database. Returns the updated task if successful, or an appropriate error message if not found.

Part 4: Delete a task

Endpoint: DELETE /tasks/{id} Accepts a task ID as a parameter. Deletes the corresponding task from the database. Returns a success message if the deletion is successful, or an appropriate error message if not found.

Part 5: List all tasks

Endpoint: GET /tasks Retrieves all tasks from the database. Returns a list of tasks, including their details (title, description, due date).

Expected Output

For Part 1 (**Create a new task**), the expected output will be:

- If the request is valid and successful, the API will respond with the created task object including the assigned ID.
- If the request is invalid (e.g., missing required fields) or encounters a database error, the API will respond with an appropriate error message.

For Part 2 (**Retrieve a task**), the expected output will be:

- If the task with the given ID exists in the database, the API will respond with the task details.
- If the task with the given ID does not exist, the API will respond with an appropriate error message.

For Part 3 (**Update a task**), the expected output will be:

- If the task with the given ID exists in the database and the update is successful, the API will respond with the updated task details.
- If the task with the given ID does not exist or the update encounters a database error, the API will respond with an appropriate error message.

For Part 4 (**Delete a task**), the expected output will be:

- If the task with the given ID exists in the database and the deletion is successful, the API will respond with a success message.
- If the task with the given ID does not exist, the API will respond with an appropriate error message.

For Part 5 (**List all tasks**), the expected output will be:

- If there are tasks stored in the database, the API will respond with a list of tasks, including their details.
- If there are no tasks in the database, the API will respond with an empty list.