TAKE HOME ASSIGMENT (GOLANG)

OBJECTIVE:

Develop a simple event-driven task management service where users can create tasks, assign them to others, mark them as complete, and track progress. The service should be built using Go and incorporate various components to assess the developer's overall development expertise

INSTRUCTION:

You are tasked with building a RESTful API for a task management system. The system should have basic functionality such as creating tasks, updating their status, and fetching task information. The tasks can be assigned to different users, and users should be able to fetch their own tasks.

SYSTEM SHOULD HAVE THE FOLLOWING TECHNICAL REQUIREMENTS:

1. **Task Management API**:
   - Create Task: Endpoint to create a new task. A task must have a title, description, assigned user, status (Pending, In Progress, Completed), and due date.
   - Get Tasks: Endpoint to retrieve all tasks or filter tasks by status, user, or due date.
   - Update Task: Endpoint to update task details (e.g., change status, description, etc.).
   - Delete Task: Endpoint to delete a task.

2. **User Management API** (Simple User Authentication/Identification):
   - Create User: Endpoint to create a new user. A user has a name and email.
   - Assign Task: Assign a task to a user.

3. **Database**:
   - Use any relational database (e.g., PostgreSQL or MySQL) to persist the tasks and users.
   - Define appropriate schema for tasks and users.
4. **Docker Setup**:
   - The application should be containerized using Docker.
   - Use Docker Compose to spin up the application and database in separate containers.

5. **Testing**:
   - Implement unit tests for the business logic (e.g., task creation, assignment, and status updates).
   - Test coverage should focus on edge cases and error handling.

6. **API Documentation**:
   - Provide API documentation (e.g., using Swagger or a simple README) that describes how to interact with the API, including request/response formats.

OPTIONAL (BONUS):

1. **Task Notification System**: Implement a simple notification system that sends an email (or logs a notification) to the assigned user when their task's status changes to "Completed."
2. **Pagination and Sorting**: Implement pagination and sorting for the task list API.
3. **Security**: Add token-based authentication (e.g., JWT) for protecting certain endpoints (like creating or deleting tasks).

DELIVERABLES:

1. A GitHub repository containing:
2. Well-documented code.
3. Docker and Docker Compose configuration files.
4. SQL migration scripts (if any).
5. Unit tests.
6. README with instructions on how to set up and run the application, as well as how to run the tests.

EVALUATION CRITERIA:

1. **Code Quality**: Readability, maintainability, and use of idiomatic Go.
2. **Functionality**: The API works as expected and meets the requirements.
3. **Dockerization**: Proper use of Docker and Docker Compose for local development.
4. **Database Design**: Thoughtful and scalable schema design.
5. **Testing**: Adequate test coverage and meaningful test cases.
6. **Error Handling**: Graceful handling of errors and edge cases.
7. **Documentation**: Clear instructions and API documentation.

TIME: **24 hours**