Develop a Task Management REST API using Go and Gin Framework

Objective:

The objective of this task is to create a simple Task Management REST API using Go programming language and Gin Framework. This API will support basic CRUD operations for managing tasks.

Requirements

1. Implement a REST API with the following endpoints:
    - GET /tasks: Get a list of all tasks.
    - GET /tasks/:id: Get the details of a specific task.
    - PUT /tasks/:id: Update a specific task. This endpoint should accept a JSON body with the new details of the task.
    - DELETE /tasks/:id: Delete a specific task.
    - POST /tasks: Create a new task. This endpoint should accept a JSON body with the task's title, description, due date, and status.
2. Use an in-memory database to store tasks. Database integration with persistent storage will be covered in later lessons, so for this task, focus on implementing data storage in memory.
3. Ensure proper error handling and response codes for different scenarios such as successful operations, invalid requests, and resources not found.
4. Provide clear and concise documentation for each endpoint using postman, including expected request payloads and response formats.
5. Utilize Postman to test each endpoint of the Task Management API.

Instructions

1. Use Go programming language and Gin Framework to develop the API.
2. Implement the specified endpoints adhering to the defined requirements.
3. Utilize an in-memory database to store task data.
4. Test the API endpoints using appropriate tools (e.g., Postman, curl).
5. Write clean, well-structured, and maintainable code with proper comments.
6. Ensure the code is properly formatted and follows best practices for Go development.
7. Document the API endpoints with details on request and response formats.
8. Submit your code along with any necessary instructions for running and testing the API.

Folder Structure:

- Follow the following folder structure for this task

```
task_manager/
├── main.go
├── controllers/
│   └── task_controller.go
├── models/
│   └── task.go
├── data/
│   └── task_service.go
├── router/
│   └── router.go
├── docs/
│   └── api_documentation.md
└── go.mod
```

- `main.go:` Entry point of the application.
- `controllers/task_controller.go:` Handles incoming HTTP requests and invokes the appropriate service methods.
- `models/:` Defines the data structures used in the application.
- `data/task_service.go:` Contains business logic and data manipulation functions.
- `router/router.go:` Sets up the routes and initializes the Gin router and Defines the routing configuration for the API.
- `docs/api_documentation.md:` Contains API documentation and other related documentation.

Evaluation Criteria:

- Implementation of all required endpoints according to specifications.
- Correct handling of various HTTP methods and response codes.
- Proper error handling and validation of input data.
- Efficient and well-structured code following Go best practices.
- Clear and comprehensive documentation of API endpoints.
- Compliance with the provided instructions and requirements.

Note

- Remember that this task is focused on backend development skills using Go and Gin Framework. Avoid unnecessary complexity in the implementation.

- Database integration with persistent storage will be addressed in subsequent lessons; hence, focus on implementing data storage in memory for this task.