.NET Developers

TASK

TODO List:

Your task is to create a system for managing a simple To-do list.

The To-do list is a simple list of tasks. Each task has a title and could have a due date. When a task is completed, the user marks it as done.

When a user creates a new task, they can enter a title and a due date. The user should be able to see all their pending tasks and all their overdue tasks. Pending tasks are tasks that are not past or do not have a due date, overdue tasks are tasks that are past their due date. When a task is marked as done it should no longer appear on the to-do list.

Functional requirements:

1. Create a task - the user can create a new task and optionally set its due date
2. Edit task - the user can edit the title and the due date of the task as well as marking it as completed
3. List pending tasks – the user can see a list of tasks that are not past their due date and not marked as done.
4. List overdue tasks – the user can see a list of tasks that are past their due date and not marked as done.

Non-Functional requirements:

1. Backed: Create a minimal RESTful API using C# and Asp.Net. Use any LTS version of the .NET framework.
2. Database:
   a. **If using Docker** - Any database is accepted
   b. **If using non-Docker solution**- Entity Framework with in-memory database so the project could be started without a database server.
3. UI - one of the following is accepted:
   a. SPA using Angular or React – **Full Stack Developer**
   b. Documented Open API (Swagger UI) - **Backend Developer**
4. Cover 100% of the branches in one of Functional requirements 3 or 4 with Unit tests.

Code Quality and Good practices:

1. Demonstrate use of layered, clean or any other commonly used architecture.
2. Adhere to the .NET coding style - .NET Coding style and .NET Design guidelines
3. Code should be properly formatted
4. Even though not all code needs to be covered with unit test, all the code should be testable
5. The application should run on a system with the following software installed:
   a. Visual Studio 2022
   b. Visual Studio Code
   c. Node.js
   d. Docker Desktop
6. The application should compile out of the box from source.
7. The application should start without any extra setup.
8. The database model should be created according to the best practices. (e.g., choosing proper length for string data, proper primary keys, proper naming conventions etc.)
9. The source repository should have a good folder structure.
10. The source repository should include only files necessary to build and run the code, any build output should not be in version control.

Extra credit:

1. The entire application runs in a Docker environment with separate containers for frontend, backend and database.
2. The application could be debugged with Visual Studio with Docker compose.