📘 TaskHub – C# REST API Assignment

🎯 Objective

Design and implement a **C# RESTful API** for a simple **Task Management System** using **ASP.NET Core**. This project evaluates your knowledge in:

- REST API design
- Object-Oriented Programming (OOP)
- Dependency Injection
- Async programming
- Code structure and best practices

📦 Project Overview

You will build a Task Management System where users can create, manage, and track tasks. Tasks should be assigned to users and categorized by status and priority.

📐 Entities

👤 User

- `Id` (int)
- `Name` (string)
- `Email` (string)

📋 TaskItem

- `Id` (int)
- `Title` (string)
- `Description` (string)
- `Status` (enum): `Pending`, `InProgress`, `Completed`
- `Priority` (enum): `Low`, `Medium`, `High`
- `DueDate` (DateTime)
- `UserId` (int)

🌐 API Endpoints

👥 User APIs

Method	Endpoint	Description
GET	`/api/users`	Get all users
GET	`/api/users/{id}`	Get user by ID
POST	`/api/users`	Create new user
PUT	`/api/users/{id}`	Update user details
DELETE	`/api/users/{id}`	Delete user

✅ Task APIs

Method	Endpoint	Description
GET	`/api/tasks`	Get all tasks
GET	`/api/tasks/{id}`	Get task by ID
GET	`/api/users/{id}/tasks`	Get all tasks for a user
POST	`/api/tasks`	Create new task
PUT	`/api/tasks/{id}`	Update task
PATCH	`/api/tasks/{id}/status`	Update task status only
DELETE	`/api/tasks/{id}`	Delete task

🧰 Features to Implement

🎯 Functional Requirements

- Assign tasks to users
- Filter tasks by:
  - Status
  - Priority
  - Due Date Range
- Sort tasks by:
  - Due Date
  - Priority
- Input validation (e.g., due date must be in the future)
- Enforce status transitions (`Pending → InProgress → Completed`)

💻 Technical Requirements

- Use ASP.NET Core Web API (.NET 6 or higher)
- Use DTOs and models with mapping logic
- Implement repository and service layers

- Register services using built-in DI container
- Use `async/await` for all methods
- Proper use of HTTP status codes

👩‍💻 Object-Oriented Design Expectations

- Use interfaces (e.g., `ITaskService`, `IUserService`)
- Abstract service logic from controller
- Follow the Repository Pattern
- Demonstrate encapsulation, inheritance, and polymorphism

✨ Bonus (Optional)

- Basic static user authentication via header
- Export tasks as CSV or JSON
- Swagger documentation for all endpoints
- Pagination on `GET /api/tasks`

📮 Submission Guidelines

1. Push code to GitHub or share as a zip file
2. Include a `README.md` with:
    - Project setup instructions
    - Sample API requests/responses
    - Explanation of OOP implementation and architecture

🏁 Good Luck!

Feel free to ask for clarifications if needed. Happy coding! 🚀