ASP.Net Core WebAPI Project

📌 Project Title:

[Each student selects a unique business domain]

📍 Objective:

Develop a full-stack web application with a backend API built in **ASP.NET Web API** and a frontend developed using **Angular**. The application should serve a business use case chosen by the student.

🔧 Functional Requirements

1. Authentication & Authorization

- Users should be able to register and log in.
- Role-based access (e.g., Admin, User, Manager, etc.)
- Use JWT for secure communication.

2. Entities and Modules

Each student must:

- Design at least **3 main entities** (e.g., Product, Order, Customer).
- Implement full **CRUD operations** for these entities.
- Example (to be adapted per student's idea):
    - A "Book" in a Library System
    - A "Service" in a Booking App
    - A "Task" in a Project Management App

3. API Structure

- RESTful APIs
- Routes must be logically grouped and follow standard HTTP verbs.
- Proper status codes (200, 400, 401, 404, 500)

4. Validation & Error Handling

- Server-side validation for all inputs.
- Meaningful error messages returned from the API.

5. Database

- Use SQL Server.
- Create relational tables with appropriate foreign keys.
- Use Entity Framework OR Dapper.

💻 Frontend (Angular) Requirements

1. Authentication Integration

- Token storage (e.g., localStorage).
- Login form and protected routes.

2. Dashboard View

- Overview of the main entities (e.g., list recent items, analytics count).

3. Forms for CRUD

- Forms for adding/editing/deleting items.
- Use Angular reactive forms with validation.

4. Routing

- Implement routing for the main modules (e.g., /products, /users).
- Use route guards for restricted views.

5. API Communication

- Use Angular services to consume Web API.
- Use HttpClient and observables properly.

📶 Extra Features (choose at least one)

- Pagination and Search.
- File upload (e.g., image for product/user).
- Charts (e.g., ng2-charts) for displaying analytics.

□ Non-Functional Requirements

- Responsive UI (Bootstrap or Angular Material).
- Code should be clean and follow standard naming conventions.
- Use GitHub for version control.
- Each commit should have a meaningful message.

📌 Deliverables

- Source code for frontend and backend.
- Database .
- Postman collection for API testing.
- Short documentation (PDF or README).

📍 Note for Students

You are free to choose your **business context** (e.g., Clinic system, E-Commerce, Booking system, School Management...) , but your implementation must match the **technical specs** above.