1. Task: Build a CRUD Application for Managing Users (ASP.NET Core + React)

Situation: Your company needs an internal application to manage user information. The app should allow users to create, read, update, and delete user profiles.

Task Requirements:

- **Backend (ASP.NET Core):**
  - Create an API with ASP.NET Core to manage users.
  - Define a `User` model with fields such as `Id`, `Name`, `Email`, `DateOfBirth`, and `Role`.
  - Implement the following API endpoints:
    - `POST /users`: Create a new user.
    - `GET /users`: Retrieve a list of all users.
    - `GET /users/{id}`: Retrieve a user by ID.
    - `PUT /users/{id}`: Update user information.
    - `DELETE /users/{id}`: Delete a user.
  - Implement validation for input data (e.g., ensure email format is correct).
  - Use Entity Framework Core to interact with the database.
- **Frontend (React):**
  - Use React to create a user interface for viewing and managing users.
  - Create a list view for displaying users and buttons to create, edit, or delete users.
  - Create a form for adding new users and editing existing users.
  - Use Axios or Fetch API to communicate with the backend.
  - Implement form validation on the frontend (e.g., required fields, email validation).

---

2. Task: Build a Simple Authentication System (ASP.NET Core + Angular)

Situation: Your team needs a simple user authentication system where users can sign up, log in, and access protected routes.

Task Requirements:

- **Backend (ASP.NET Core):**
  - Create an authentication system using JWT (JSON Web Token).
  - Implement user registration (`POST /auth/register`) and login (`POST /auth/login`) endpoints.
  - Use ASP.NET Core Identity for managing user information (e.g., username, password).
  - Hash the password before saving it in the database.
  - Use JWT for authenticating users, and include the token in the response after login.

    - Create a protected route (e.g., `GET /user/profile`) that requires a valid JWT token to access.
- **Frontend (Angular):**
    - Build a login form and registration form using Angular.
    - Store the JWT token in local storage or session storage after successful login.
    - Create a user profile page that fetches data from the protected route, and displays user details (such as username, email).
    - Implement a logout functionality that removes the JWT token and redirects the user to the login page.

---

3. Task: Build a Product Catalog System (ASP.NET Core + React)

Situation: Your team needs an e-commerce product catalog system where users can browse products, filter them by category, and view product details.

Task Requirements:

- **Backend (ASP.NET Core):**
    - Create a `Product` model with fields like `ProductId`, `Name`, `Category`, `Price`, `Description`, and `ImageUrl`.
    - Implement API endpoints to retrieve all products (`GET /products`) and filter products by category (`GET /products?category=categoryName`).
    - Implement an endpoint to get detailed information about a single product (`GET /products/{id}`).
- **Frontend (React):**
    - Use React to display a list of products fetched from the API.
    - Create a filtering system that allows users to filter products by category.
    - Display product details when a user clicks on a product.
    - Implement pagination or lazy loading to handle large sets of products efficiently.
    - Use Axios to fetch data from the API.