1. Task: Build a Blogging Platform (ASP.NET Core + Angular)

Situation: Your company needs a simple blogging platform where users can create, read, update, and delete blog posts.

Task Requirements:

- **Backend (ASP.NET Core):**
  - Create a `BlogPost` model with fields like `PostId`, `Title`, `Content`, `AuthorId`, and `CreatedAt`.
  - Implement the following API endpoints:
    - `POST /posts`: Create a new blog post.
    - `GET /posts`: Get all blog posts.
    - `GET /posts/{id}`: Get a single post by its ID.
    - `PUT /posts/{id}`: Edit an existing blog post.
    - `DELETE /posts/{id}`: Delete a blog post.
  - Implement basic validation for blog post fields (e.g., `Title` and `Content` should not be empty).
- **Frontend (Angular):**
  - Create components to display a list of blog posts, a form to create new posts, and a detail page for viewing individual blog posts.
  - Use Angular Forms for creating and editing blog posts.
  - Implement navigation using Angular Router for different pages (home, post details, create post).
  - Use Angular services to communicate with the backend API.

---

2. Task: Implement a Real-Time Chat Application (SignalR + React)

Situation: You need to create a real-time chat application where users can send and receive messages instantly.

Task Requirements:

- **Backend (ASP.NET Core):**
  - Use **SignalR** in ASP.NET Core to handle real-time communication.
  - Create a `Message` model with fields like `SenderId`, `ReceiverId`, `Content`, and `Timestamp`.
  - Create a SignalR Hub to manage connections and broadcast messages to clients.
  - Implement API endpoints to retrieve message history for a user.
- **Frontend (React):**
  - Use React to create the chat interface.

- Establish a real-time connection with the SignalR Hub to send and receive messages.
- Display the list of messages as they arrive in real-time.
- Allow users to send messages to other users in real-time.
- Optionally, store message history locally or in a database for future reference.

---

3. Task: Build a Task Management Application (ASP.NET Core + Angular)

Situation: You need to build a simple task management application where users can create, assign, and track tasks.

Task Requirements:

- **Backend (ASP.NET Core):**
  - Create a `Task` model with fields like `Title`, `Description`, `AssignedTo`, `Status` (e.g., `Pending`, `In Progress`, `Completed`), and `DueDate`.
  - Implement the following API endpoints:
    - `POST /tasks`: Create a new task.
    - `GET /tasks`: Get a list of all tasks.
    - `GET /tasks/{id}`: Retrieve a task by ID.
    - `PUT /tasks/{id}`: Update a task's status or details.
    - `DELETE /tasks/{id}`: Delete a task.
  - Implement basic validation (e.g., required fields, proper date format).
- **Frontend (Angular):**
  - Create a list of tasks and display their status.
  - Implement filters to view tasks by status or assigned user.
  - Create a form to add or edit tasks.
  - Use Angular Forms to handle task creation and editing.
  - Implement routing to navigate between task lists and task details.