**Project Overview**

1. **Framework**: .NET Core (ASP.NET Core 7.0 or the latest available version).
2. **Database**: SQLite using Entity Framework Core.
3. **Real-Time Communication**: SignalR.
4. **Features**:
   * User-to-user chat.
   * File/media upload and storage.
   * CRUD operations for managing chat metadata.
5. **Tools**:
   * Swagger for API documentation.
   * Dependency injection for services.

**Steps to Create the Project**

1. **Set Up a New ASP.NET Core Project**:
   * Use dotnet new webapi to scaffold the project.
2. **Add Dependencies**:
   * Add SignalR: dotnet add package Microsoft.AspNetCore.SignalR
   * Add EF Core: dotnet add package Microsoft.EntityFrameworkCore.Sqlite
   * Add EF Core tools: dotnet add package Microsoft.EntityFrameworkCore.Design
   * Add Swagger: dotnet add package Swashbuckle.AspNetCore
3. **Create Models, Context, and Controllers**.
4. **Enable SignalR for Real-Time Communication**.
5. **Implement CRUD APIs** for chat and user management.
6. **Handle File Uploads** and Media Storage.

To extend the project with the requested features, we need to add:

1. **Default Admin User**:
   * Seed a default admin user into the database.
   * Allow admin users to manage other users, including creating new ones.
2. **User Roles**:
   * Include roles like Admin and User.
   * Allow role-based authorization using ASP.NET Core Identity.
3. **User Profiles**:
   * Add editable fields such as Avatar, Email, and other profile-related fields.
   * Implement endpoints for editing the user's profile.
4. **Active Directory Integration**:
   * Enable the project to read users from the Windows Active Directory.

### Features Overview

1. **JWT Authentication**:
   * Implement login and token generation with ASP.NET Core Identity.
   * Protect all controllers using [Authorize] attribute with JWT.
2. **Custom Token Expiry**:
   * Add functionality for the admin to define user-specific session duration (uptime or timeout).
   * Use this custom timeout when generating JWT tokens.

To implement these features, we need to:

1. **List All Users and Their Online Status**:
   * Use SignalR to track online users.
   * Exclude the currently logged-in user from the list.
2. **Display Chat History When Starting a Chat**:
   * Fetch chat messages from the database between the logged-in user and the selected user.
   * Provide an endpoint for retrieving this chat history.