**ProfileBook - Project Documentation**

**📌 Introduction**

ProfileBook is an online social media platform that connects people virtually. Users can post content, message other users, and report users. Admins manage users, approve posts, and handle reported users. The project follows a full-stack architecture with Angular as the frontend and ASP.NET Core MVC with Web API as the backend.

**🛠️ Tech Stack**

• Frontend: Angular  
• Backend: ASP.NET Core MVC with Web API  
• Database: SQL Server (Entity Framework Core ORM)  
• File Storage: Server-side storage for profile pictures and posts  
• API Documentation: Swagger

**👤 User Stories**

1. User Authentication: Login, logout, and register.  
2. View Posts: See posts shared by other users.  
3. Create Post: Submit posts for admin approval.  
4. Receive Post Approval Notification.  
5. Like/Comment on posts.  
6. Message other users.  
7. Report inappropriate users.  
8. Search for users.

**🛡️ Admin Stories**

1. Admin Authentication: Login, logout, and register.  
2. CRUD on Users: Create, read, update, delete users.  
3. Approve Posts.  
4. Create User Groups.  
5. View reported users.

**⚙️ System Requirements**

• Frontend: Angular UI consuming Web API  
• Backend: ASP.NET Core MVC + Web API  
• ORM: Entity Framework Core  
• Database: SQL Server  
• Image Uploads: Server directories

**🏗️ Architecture Overview**

1. Frontend (Angular):  
 - User/Admin UI, routing, state management, API consumption.  
2. Backend (ASP.NET Core MVC):  
 - Web API, JWT authentication, EF Core.  
3. Database (SQL Server):  
 - Tables for Users, Posts, Messages, Reports, Groups.  
4. Image Uploads: Stored on server.  
5. API Documentation: Swagger.

**📅 Sprint Plan**

**Sprint I**

• Use Case Document  
• Database Schema Design  
• ORM Selection (EF Core)  
• Backend Setup (Controllers, Views)  
• Static Angular Frontend

**Sprint II**

• Angular Frontend Setup  
• Backend Implementation with JWT Authentication  
• CRUD Operations (Users, Posts, Messages)  
• Admin/User UI Layouts  
• Messaging API  
• Post Management

**Sprint III**

• Search & Filter  
• Post Approval (Admin)  
• Reported Users Management  
• Groups Management  
• Swagger API Testing  
• Final Integration

**🔑 Key Points for Implementation**

• Angular Router for navigation  
• State Management with Redux/Context API  
• API Handling   
• JWT Authentication  
• EF Core for Database  
• Separate Messaging API  
• Image Upload with IFormFile