Image Gallery App - README

**1. Project Overview**

The \*\*Image Gallery App\*\* is a web application that allows users to upload, view, like, and comment on images. Users can create accounts, verify their email addresses, and manage their own uploaded images and comments. The app incorporates security features such as secure login, password complexity enforcement, and account lockout after multiple failed login attempts.  
- \*\*Frontend:\*\* React (JSX)  
- \*\*Backend:\*\* ASP.NET Core (C#) with Entity Framework  
- \*\*Database:\*\* SQL Server

**2. Project Architecture**

**2.1 Component Overview**

- \*\*Frontend:\*\* React (JSX), handles user interactions, form validations, and data display from the backend API.  
- \*\*Backend:\*\* ASP.NET Core (C#) manages business logic, user authentication, image handling, and API requests.  
- \*\*Database:\*\* Entity Framework Code First for managing users, images, comments, and likes.  
- \*\*Email:\*\* SMTP email service for verification, OTPs, and password reset.

**2.2 Key Technologies**

- \*\*Frontend:\*\* React (JSX)  
- \*\*Backend:\*\* ASP.NET Core (C#)  
- \*\*Database:\*\* SQL Server with Entity Framework (Code First)  
- \*\*Authentication:\*\* Custom authentication with password complexity, email verification, and account lockout  
- \*\*Email Service:\*\* SMTP for email verification and password resets

**3. App Features and Functionalities**

**3.1 User Roles**

- \*\*Unregistered Users:\*\*  
 - Can register for an account.  
 - Can view publicly available images but can't interact with them.  
- \*\*Registered Users:\*\*  
 - Full access to upload, like, and comment on images.  
 - Manage their own images and comments (CRUD operations).  
 - Can reset their passwords via email.

**3.2 User Management**

- \*\*Registration:\*\* Users can register using mandatory fields (full name, email, password). After registration, a verification email is sent to activate the account.  
- \*\*Login:\*\* Users log in using their email and password. After 3 failed attempts, the account is locked for 5 minutes.  
- \*\*Password Reset:\*\* Users can reset passwords via a link sent to their email.

**3.3 Image Management**

- \*\*Upload:\*\* Users upload images with a title, description, and category. Supported file formats are validated.  
- \*\*Gallery View:\*\* Displays all uploaded images with options to like or comment.  
- \*\*Image Details:\*\* Each image has a dedicated page for viewing and interaction.

**3.4 Additional Features**  
- \*\*Comments and Likes:\*\* Users can like/unlike images and perform CRUD operations on their own comments.  
- \*\*Pagination:\*\* Efficient image browsing using pagination.

**4. API Endpoints**

| Method | Endpoint | Description |  
| ------ | -------- | ----------- |  
| POST | `/api/auth/register` | Registers a new user |  
| POST | `/api/auth/login` | Logs in a registered user |  
| POST | `/api/auth/forgotpassword` | Sends password reset email |  
| POST | `/api/auth/resetpassword` | Resets user password |  
| GET | `/api/images` | Fetches all images |  
| POST | `/api/images/upload` | Uploads a new image |  
| PUT | `/api/images/edit/{id}` | Edits an existing image |  
| DELETE | `/api/images/delete/{id}` | Deletes an image |  
| POST | `/api/images/{id}/like` | Likes or unlikes an image |  
| POST | `/api/images/{id}/comment` | Adds a comment |  
| PUT | `/api/comments/{id}` | Edits a comment |  
| DELETE | `/api/comments/{id}` | Deletes a comment |

**5. Running the Application**

**5.1 Backend (ASP.NET Core)**

Ensure the following prerequisites are installed:  
- .NET SDK  
- SQL Server  
  
\*\*Steps to run the backend:\*\*  
1. Clone the repository.  
2. Set up the database connection string in `appsettings.json`.  
3. Run migrations: `dotnet ef database update`  
4. Start the backend server: `dotnet run`

**5.2 Frontend (React)**

Ensure Node.js and npm are installed.  
  
\*\*Steps to run the frontend:\*\*  
1. Install dependencies: `npm install`  
2. Start the React development server: `npm start`

**6. Technologies Used**

- \*\*Frontend:\*\* React (JSX)  
- \*\*Backend:\*\* ASP.NET Core (C#)  
- \*\*Database:\*\* SQL Server (Entity Framework Code First)  
- \*\*Authentication:\*\* ASP.NET Core Identity with custom policies (e.g., password complexity)  
- \*\*State Management:\*\* React `useState` and Context API  
- \*\*Email Handling:\*\* SMTP for sending emails (verification and password resets)  
- \*\*API Testing:\*\* Postman  
- \*\*File Storage:\*\* Imgur (for image hosting)

**7. Security Features**

- \*\*Password Complexity:\*\* Passwords must include uppercase, lowercase, digit, and non-alphanumeric characters.  
- \*\*Account Lockout:\*\* After 3 failed login attempts, the account is locked for 5 minutes.  
- \*\*Email Confirmation:\*\* Users must confirm their email before logging in.

**8. Code Overview**

**8.1 Backend Controllers**

- \*\*ImageUploadController:\*\* Handles image uploads, editing, deletion, and fetching images from the database and Imgur.  
- \*\*CommentsController:\*\* Manages user comments on images with full CRUD operations.  
- \*\*LikesController:\*\* Manages image likes and user interactions.

**8.2 Database Models**

- \*\*ImageUpload:\*\* Stores information about uploaded images.  
- \*\*Comment:\*\* Stores user comments on images.  
- \*\*Like:\*\* Tracks likes and dislikes for images.  
- \*\*User:\*\* Custom user model extending IdentityUser for user management.

**8.3 Email Service**

- \*\*EmailSender:\*\* Uses SMTP to send account verification and password reset emails.