1. **Introduction**

The Image Gallery App allows users to upload, manage, and interact with images. This documentation outlines the technical design, functionalities, and usage instructions for the application.

1. **Project Architecture**
   1. Architecture Diagrams

* Database Diagram:

|  |
| --- |
|  |

* 1. Overview

This application is built using a React frontend, an ASP.NET Core backend, and an SQL Server database. The architecture is designed for scalability and maintainability.

1. **Technologies Used**

* **Frontend:** React, Typescript, Axios
* **Backend:** ASP.NET Core, Entity Framework Core
* **Database:** SQL Server
* **Authentication:** JWT (JSON Web Tokens)
* **State Management:** MobX
* **API Testing:** Postman

1. **Application Features**
   1. User Management

* **Registration:** Users can register with their email and password.
* **Login:** Users can use the email and password they registered with to log in.
* **Change Password:** Users can reset their passwords when logged in or out.
* **Email Verification:** A verification email is sent upon registration and when resetting passwords.
  1. Image Management
* **Upload:** Users can upload images with titles and a description and also select an image category.
* **Gallery View:** Images are displayed in a grid format.
* **Image Details:** Users can view the details of each image on click.
* **Delete:** Users can delete images.
* **Comments:** Users can leave comments on images and delete their comments.
* **Profiles:** Users can set images in their library as profile pictures.
  1. Frontend Validations
* Validations are applied for image title, description, and category.
* Validations are applied login credentials and registration credentials.
* Client-side validation messages are displayed for user feedback.
  1. Pagination
* The application implements pagination for loading images in chunks.

1. **User Roles and Permissions**

* User: Can upload and manage their own images, leave comments, and view galleries.

1. **API Endpoints**

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /api/account/register | Post | Register a new user |
| /api/account/login | Post | Log in an existing user |
| /api/account/verifyEmail | Post | Send verification email |
| /api/account/resendEmailConfirmLink | Post | Resend verification email |
| /api/account/ | Get | Get currently logged in user |
| /api/account/refreshToken | Post | Send refresh token |
| /api/account/forgotPassword | Post | Send password reset link |
| /api/account/resetPassword | Post | Send up new password |
| /api/categories/ | Get | Get all categories |
| /api/categories/(id) | Get | Get individual categories by Id |
| /api/categories/ | Post | Create a new category |
| /api/categories/(id) | Put | Edit a category name |
| /api/categories/(id) | Delete | Delete a category |
| /api/photos/ | Post | Upload a photo |
| /api/photos/(id) | Delete | Delete a photo |
| /api/photos/(id)/setMain | Post | Set a photo as a profile picture |
| /api/photos/ | Get | Get all the photos in the database |
| /api/photos/(username) | Get | Get photos uploaded by a user |

Example API Request

* Register User:

|  |
| --- |
| Post [https://localhost:5000/api/account/register](https://localhost5000/api/account/register)  {  “fullname”**:** “Charmaine Mogotlane”,  “username”**:** “Maine”,  “password”**:** “Pa$$w0rd”  } |

1. **Setup Instructions** 
   1. Prerequisites

* .Net SDK
* Node.js
* SQL Server
  1. Configuration Steps

1. Clone the repository <https://github.com/charmaine-nonhlanhla/ImageGallery.git>.
2. Open SQL Server Management Studio (SSMS) and create a new database named ImageGallery.

|  |
| --- |
| CREATE DATABASE ImageGallery; |

1. Configure connection strings in appsettings.json (API/ Backend).

|  |
| --- |
| {  “ConnectionStrings”: {  “DefaultConnection”: “Server=YOUR\_SERVER\_NAME;Database=ImageGallery;Trusted\_Connection=True;”  }  } |

Replace YOUR\_SERVER\_NAME with the name of your SQL Server instance.

1. Install frontend dependencies:

|  |
| --- |
| cd client-app  npm install |

1. Migrate the database:

|  |
| --- |
| cd api  dotnet ef database update |

1. **Running the Application**
   1. Backend
2. Open the terminal in Visual Studio Code (vscode) and navigate to the root folder of the project

|  |
| --- |
| PS C:\Users\CMogotlane.SINGULAR\IMGALLERY\ImageGallery> |

1. **Add a Migration:** Run the following command to create a migration:

|  |
| --- |
| dotnet ef migrations add IMGAPP -p Persistence -s API |

1. **Update the database:** After a successful migration, navigate into the API project directory.

|  |
| --- |
| cd API |

1. Run the database update:

|  |
| --- |
| dotnet ef database update |

1. Run the Application: To ensure there are no build errors, run:

|  |
| --- |
| dotnet watch |

* After confirming that there are no build errors, stop the process using Ctrl + C.

1. Finally, run the application:

|  |
| --- |
| dotnet run |

* The API will be hosted on <https://localhost:5000.>
  1. Frontend

Start the frontend development server:

|  |
| --- |
| cd client-app  npm run de v || npm start |

* The API will be hosted on [https://localhost:3000.](https://localhost:5000.)

1. **Bonus Features**

* Verification Email: Sent upon user registration for confirmation.
* Comments on Images: Users can leave comments on images.
* User profile pictures: Users are able to set profile pictures.
* User followings: Users are able to follow each other.
* Pagination: User can view images in chunks using paged lists.