25 Scott Street, Waverley, JHB 2090 / PO Box 785261, Sandton 2146 / Tel: +27 (0) 10 003 0700 / Email: hello@singular.co.za / www.singular.co.za

Image Gallery Application

1. Introduction

1.1. Project overview

Image Gallery App is a web-based image-sharing platform where users can upload, share, and interact with images. The application allows users to create an account, upload images, and engage with other users through likes and comments. It serves as a community-driven space for sharing creativity and visual content.

2. Project Architecture

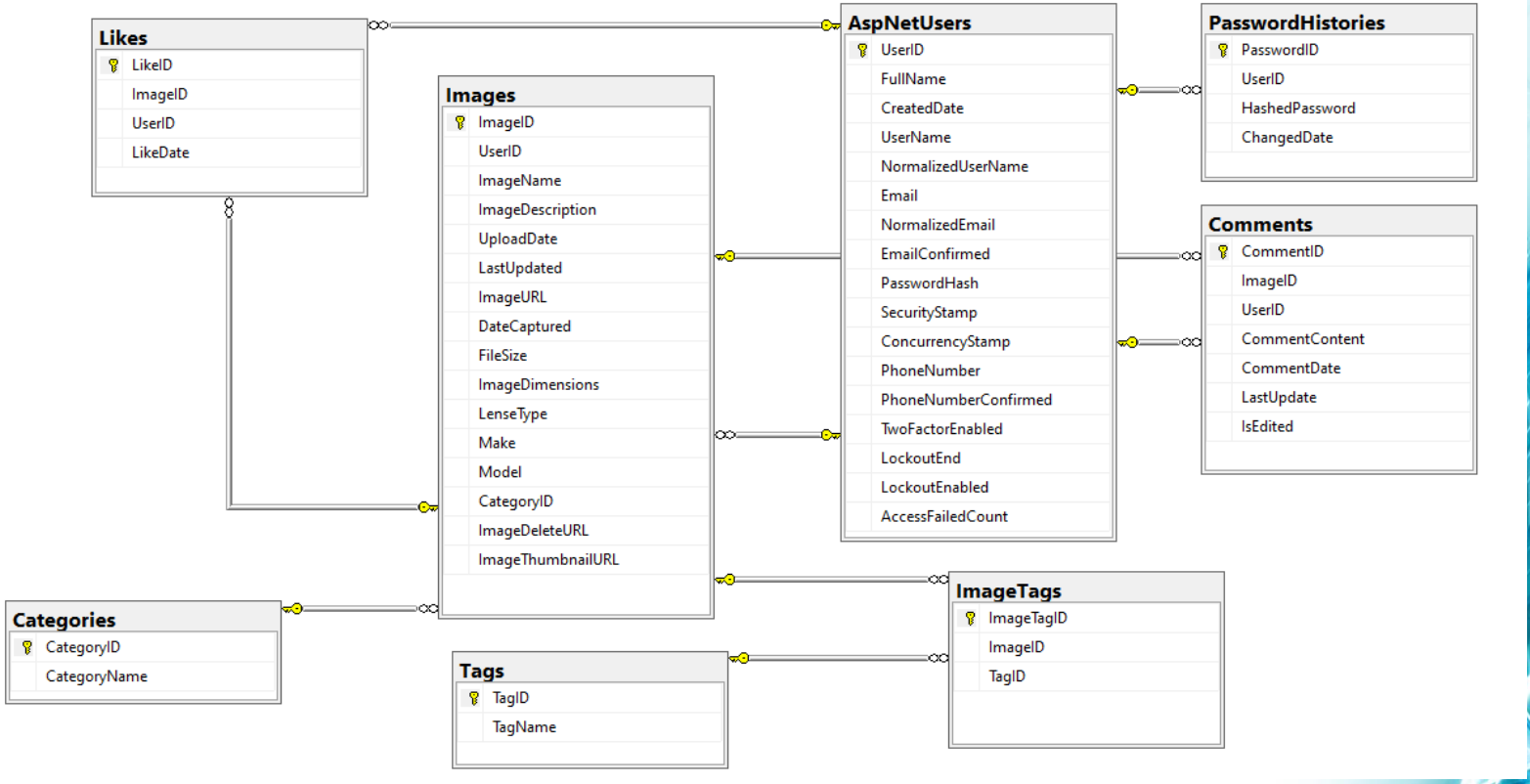
2.1. Project Technologies

- Backend Technologies: ASP.Net Core
- Frontend Technologies: Vite + React Js
- Database Technology: Microsoft SQL Server
- Cloud Storage: Imgbb.com

2.2. Database Diagram

Given the project's complexity and its rich set of features, a comprehensive set of tables was required to support the application's diverse functionality. Below is an example of the Entity-Relationship Diagram (ERD). ***In the next page***

Diagram 1


Likes
LikeID
ImageID
UserID
LikeDate
Images
ImageID
UserID
ImageName
ImageDescription
UploadDate
LastUpdated
ImageURL
DateCaptured
FileSize
ImageDimensions
LenseType
Make
Model
CategoryID
ImageDeleteURL
ImageThumbnailURL
AspNetUsers
UserID
FullName
CreatedDate
UserName
NormalizedUserName
Email
NormalizedEmail
EmailConfirmed
PasswordHash
SecurityStamp
ConcurrencyStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEnd
LockoutEnabled
AccessFailedCount
PasswordHistories
PasswordID
UserID
HashedPassword
ChangedDate
Comments
CommentID
ImageID
UserID
CommentContent
CommentDate
LastUpdate
IsEdited
Categories
CategoryID
CategoryName
Tags
TagID
TagName
ImageTags
ImageTagID
ImageID
TagID

3. Technologies

3.1. Front-end Technologies:

React was utilized for the front-end development, incorporating the following React packages to fulfill the project requirements.

- "axios": "^1.7.2",
- "exifreader": "^4.23.3",
- "react-dropzone": "^14.2.3",
- "react-infinite-scroll-component": "^6.1.0",
- "react-router-dom": "^6.24.0",
- "react-token-auth": "^2.3.8"

3.2. Back-end Technologies:

For the back-end, ASP.NET Core was used along with Entity Framework Core and ASP.NET Core Identity.

3.3. Database:

Microsoft SQL Server 2019 was selected as the database of choice.

3.4. Cloud Storage:

IMGBB.com was selected for cloud storage of images due to its high availability, fast performance, and free pricing model. Its API is also user-friendly, offering easy control and management of images.

3.5. Authentication:

JWT (JSON Web Token) was used for authentication and authorization, offering several benefits, including secure and stateless communication between the client and server. JWT tokens are compact, making them ideal for use in HTTP headers, and are easy to transmit across various environments. They support claims-based authentication, allowing for secure data exchange, and are digitally signed, ensuring data integrity and preventing tampering.

3.6. API Testing

For API testing, I primarily used Swagger, occasionally complemented by Postman. Swagger was preferred because it provides a comprehensive view of all available routes and displays the DTOs (Data Transfer Objects) used within the application. It also offers examples of request payloads and expected responses, making it easier to understand the data flow. This streamlined approach made it quicker and more efficient than Postman for accessing and testing endpoints.

4. Development Environment

4.1. Prerequisites:

4.1.1. Prerequisites For Back-end:

- .NET SDK (version 8.0+)
- Visual Studio 2019+ (or Visual Studio Code with C# extensions)
- Microsoft SQL Server
- Mailhog Email Server

4.1.2. Prerequisites For Front-end:

- Node.JS
- NPM

4.2. Running the Local Development Environment:

4.2.1. Setup For Back-end:

To set up the local test environment for ASP.Net Core assuming you have installed the prerequisites.

Step 1: Clone the git repository

Step 2: Navigate to the directory named "`api`" and open it using Visual studio, Vs Code or your terminal of choice

Step 3: In the terminal, or your IDE's terminal run the command below. This will download and install all the NuGet packages listed in the `.csproj` file.

```
$ dotnet restore
```

Step 4: Configure the connection string in the `appsettings.json` file with your chosen Database. Each database uses a different connection string so you can find this by doing a simple google search.

Step 5: Run the database migration by using command

```
$ dotnet ef migrations add <GiveYourMigrationAName>
```

Step 6: To create or update your database you need to run the command

```
$ dotnet ef database update
```

Step 7: To run the application, you can use the command

```
$ dotnet watch run
```

Step 8: Run the Mailhog server, this will allow you to send and receive test emails. Without this server the application will throw an error on tasks that require sending emails.

4.2.2. Setup For Front-end:

To set up the local test environment for (Vite + React) to run assuming you have installed the prerequisites.

Step 1: Clone the git repository

Step 2: Navigate to the "`image_gallery_app_frontend`" directory.

Step 3: Open the terminal and run the command below and wait for installation of node packages to finish

```
$ npm install
```

Step 4: Run the application using the command below

```
$ npm run dev
```

5. API Endpoints

5.1. Account Endpoints

5.1.1. Register [POST]: (api/account/register)

This endpoint allows new users to create an account by submitting their registration details, such as username, email, and password.

5.1.2. Login [POST]: (api/account/login)

The login endpoint authenticates users by verifying their credentials (username and password), providing a JWT token upon successful login for secure access to protected resources.

5.1.3. Logout [GET]: (api/account/logout)

This endpoint logs the user out of the application, invalidating their current session and ensuring that their JWT token is no longer active.

5.1.4. Forgot Password [POST]: (api/account/forgotpassword)

If a user forgets their password, this endpoint allows them to request a password reset link, which is sent to their registered email address.

5.1.5. Change Password [POST]: (api/account/changepassword)

Authenticated users can change their existing password using this endpoint by providing their current password along with the new one.

5.1.6. Reset Password [POST]: (api/account/resetpassword)

This endpoint is used when a user wants to reset their password via a reset link sent to their email, allowing them to set a new password without needing to log in.

5.1.7. Confirm Email [GET]: (api/account/confirmemail)

This endpoint is used to verify a user's email address by confirming the token sent in the account activation email, ensuring the email provided is valid and owned by the user.

5.2. Category Endpoints

5.2.1. Get Category [GET]: (api/category)

This endpoint retrieves a list of categories from the database. It is typically used to display available categories within the application, providing users with options for filtering, navigation, or organizing content based on specific categories.

5.3. Comment Endpoints

5.3.1. Create Comment [POST]: (api/comment/{ImageId})

This endpoint allows users to add a new comment to a specific image identified by the `ImageId`. Users can submit their comment content, which is then linked to the specified image.

5.3.2. Get Comment [GET]: (api/comment/{commentId})

This endpoint retrieves a specific comment using its `commentId`. It is used to view the details of a comment, including the content and metadata such as the author and timestamp.

5.3.3. Update Comment [PATCH]: (api/comment/{commentId})

This endpoint allows users to update the content of an existing comment identified by `commentId`. It is typically used to edit or correct a comment after it has been posted.

5.3.4. Delete Comment [DELETE]: (api/comment/{commentId})

This endpoint enables users to delete a specific comment by its `commentId`. Once deleted, the comment is removed from the system and no longer visible on the associated image.

5.4. Image Endpoints

5.4.1. Get Images [GET]: (api/image)

This endpoint retrieves a list of images, and includes these options, filtering, sorting, or pagination. It is used to display available images to users.

5.4.2. Post Image [POST]: (api/image)

This endpoint allows users to upload a new image to the platform. Users can provide image files along with metadata like title, description, or category.

5.4.3. Get My Images [GET]: (api/image/mylibrary)

This endpoint retrieves images uploaded by the currently authenticated user, allowing them to view and manage their personal image library.

5.4.4. Get Image by ID [GET]: (api/image/{imageId})

This endpoint fetches the details of a specific image using its `imageId`. It provides information such as the image file, metadata, and any associated comments or tags.

5.4.5. Update Image Details [PATCH]: (api/image/{imageId})

This endpoint allows users to update the details of an existing image identified by `imageId`. Updates can include changes to the image's title and description.

5.4.6. Delete Image [DELETE]: (api/image/{imageId})

This endpoint enables users to delete an image by its imageId. Once deleted, the image is removed from the platform along with its associated data.

5.5. Like Endpoint

5.5.1. Toggle Like [POST]: (api/likes/{imageId})

This endpoint allows users to like or unlike an image identified by `imageId`. When a user sends a request to this endpoint, it toggles the like status: adding a like if the image is currently unliked, or removing the like if it is already liked. This feature helps users express their preferences and engage with content on the platform.

6. App Use

6.1. User Authentication and Account Management

6.1.1. Registration Page:

Users can register by filling out a form with their username, email, and password. Upon successful registration, they are prompted to verify their email via a confirmation link sent to their inbox.

6.1.2. Login Page:

Users can log in using their credentials. Successful authentication grants access to the platform, and the user is issued a JWT token for secure browsing of protected areas.

Users who have not yet confirmed their email will be denied access to the platform until they confirm this is a security feature to prevent fake bot accounts. Users can also be locked out of their account for a period if they submit too many incorrect password attempts.

6.1.3. Forgot Password:

If users forget their password, they can request a password reset link by providing their registered email. The app will redirect them through a secure link so they can reset their password securely.

6.1.4. Reset and Change Password:

Users can reset their password through a secure link provided via email if requested, or update their password within their account if already logged in. For security:

- The app prevents usage of any previous passwords that have been used within the last 6 months.
- Needs a password complexity of at least 8 characters including capital, lowercase, and special characters

6.1.5. Logout:

A logout button allows users to terminate their session instantly by blacklisting their current JWT token and removing it from their browser, ensuring account security.

6.2. Image Browsing and Interaction

6.2.1. Homepage / Gallery:

The homepage displays a gallery of images fetched from the `/api/image` endpoint. Users can scroll through images, filter them by category, and scroll through the page seamlessly using the infinite scroll feature.

6.2.2. Image Upload:

A dedicated upload section allows users to post images directly from their device. The upload form includes fields for the image file, title, description, and category selection. After submission, images are displayed in the user's library and the public gallery.

6.2.3. Personal Image Library:

Users have access to a "My Library" section, displaying all images they have uploaded. They can manage their images, including viewing, updating details, or deleting them directly from this interface.

6.2.4. Detailed Image View:

By clicking on an image, users are taken to a detailed view that shows the image, title, description, category, and comments. This page also allows users to interact by liking the image or adding comments.

6.3. Comments and Feedback

6.3.1. Add Comment:

Users can leave comments on images, providing feedback or engaging in discussions. A simple form below each image allows for quick comment submission.

6.3.2. Edit and Delete Comments:

Users can edit their comments if corrections are needed or delete them entirely, giving them control over their posted feedback.

6.4. Liking System

6.4.1. Like/Unlike Images:

Users can express their appreciation by liking images. The like button toggles based on the user's interaction, showing real-time feedback on the number of like engagements the image has accumulated since being posted.