UE20CS352-Object Oriented Analysis and Design using Java

Project Title:  **A PHOTO GALLERY APPLICATION USING JAVA SPRINGBOOT MVC , HIBERNATE , JPA AND THYMELEAF WITH MYSQL DATABASE**

Team Member details:

DIVIJA L PES1UG20CS134

DHATHRI G PES1UG20CS141

GAURAV DNYANESH MAHAJAN PES1UG20CS150

JEEVAN R PES1UG20CS179

Problem Statement:

The idea is to create a Photo Gallery application that provides the ability to organize a digital photo collection in a Gallery view. People can upload images, view the gallery, update the image, add images to it, search for desired pictures etc. The admin manages the view requests made by the users.

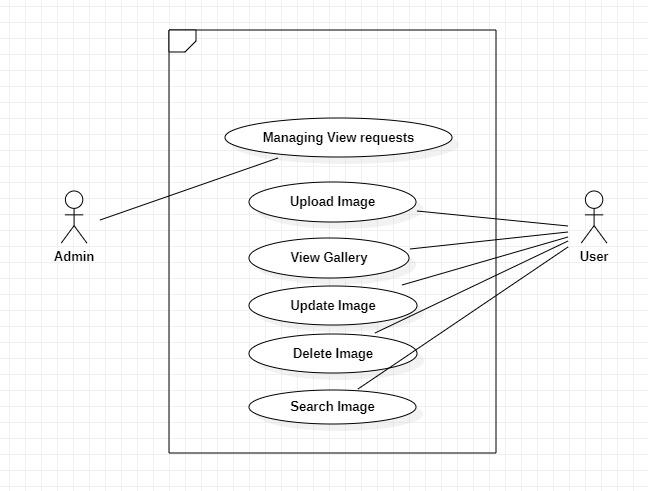
Features:

* ADMIN SUBSYSTEM :

1. Manages view requests made by the user

* **USER SUBSYSTEM :**
* **Display photos inside album with thumbnail**
* **Add photo**
* **Remove photo**
* **Update the photo**
* **Search photo**

Use Case Diagram:



USE CASE DESCRIPTIONS:

1. Searching for photos

| Name | Search for photos |
| --- | --- |
| Summary/Overview: | User can search their desired photo |
| Actor: | User |
| Pre-condition: | The actors must be registered and logged -in. The user must search for a valid owner  name . |
| Description: | 1. The users can search for their photos via owner name or album name 2. If valid details are entered, the photos will be displayed |
| Exceptions: | If invalid owner name is entered ,it should be displayed as not found. |
| Alternate Flows: | None |
| Post-conditions: | The photos searched by users are displayed. |

1. Uploading Images

| Name | Upload Image |
| --- | --- |
| Summary/Overview: | Users can upload images in the application. |
| Actor: | User |
| Pre-condition: | None. |
| Description: | Users should be able to upload desired images in the photo gallery application. |
| Exceptions: | None |
| Alternate Flows: | None |
| Post-conditions: | The database must be updated. |

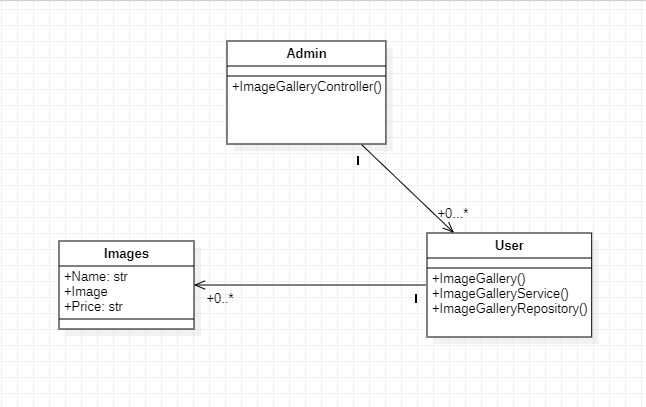
3. Deleting the photos

| Name | Delete Images |
| --- | --- |
| Summary/Overview: | Users should be able to delete images. |
| Actor: | User |
| Pre-condition: | That particular user should have uploaded that image in order to delete it. |
| Description: | Deleted images should be removed from the database. |
| Exceptions: | None. |
| Alternate Flows: | None. |
| Post-conditions: | The database needs to be updated. |

4. Updating the Images

| Name | Update Image |
| --- | --- |
| Summary/Overview: | Updating the images uploaded by the users. |
| Actor: | User |
| Pre-condition: | The user should have already uploaded the image. |
| Description: | The user is allowed to update the image that has already been uploaded. |
| Exceptions: | None. |
| Post-conditions: | The database should be updated. |

Class Model:



Architecture Patterns

When developing a photo gallery app, it is important to consider the architectural patterns that will be used to ensure the app is both functional and maintainable. So we have used the Model-View-Controller (MVC) Pattern.

1. MODEL:

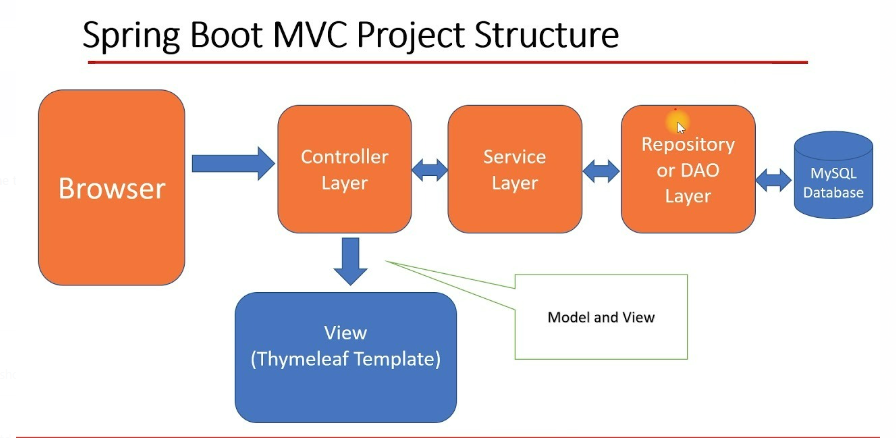
* The model component represents the application data and business logic. It is in charge of storing and managing data, as well as offering mechanisms for accessing and manipulating that data.
* The model in a photo gallery application could comprise classes that represent photos, as well as methods for adding, removing, and updating them.

1. **VIEW:**

* The view component represents the app's user interface. It is in charge of delivering the model's data in an understandable manner to the user.
* The view of a photo gallery application could comprise screens for displaying photos as well as tools for interacting with them, such as buttons for adding, deleting, or editing photos

1. **CONTROLLER:**

* The controller component serves as a link between the model and the view. It is in charge of processing user input, updating the model as needed, and updating the view to reflect model changes.
* The controller in a photo gallery application includes classes to handle user input from the view, such as adding a new photo or modifying an old one, and updating the model and view accordingly.



Design Principles

* Single Responsibility Principle (SRP)

Each class should have a single responsibility and should focus on doing that one thing well. In this application, different classes should be responsible for handling different functionalities such as image upload, image retrieval, and search.

* Liskov Substitution Principle (LSP)

Derived classes should be substitutable for their base classes. This means that any class that inherits from another class should be able to replace that class without causing any problems. For example, any new class that is added to handle image processing should be able to replace the existing image processing class without causing any issues.

* Open/Closed Principle (OCP)

The application should be open for extension but closed for modification. This means that it should be easy to add new features or functionalities without changing the existing code. For example, if a new type of image file is supported, the code should be written in such a way that it can easily handle it without needing to modify the existing code.

* Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules. Instead, both should depend on abstractions. In this application, the business logic should be separated from the database logic by using interfaces. This makes it easier to replace the database with a different one in the future without affecting the business logic.

* Interface Segregation Principle (ISP)

A class should not be forced to implement an interface that it does not use. Instead, it should only implement the methods that it needs. This helps to keep the code clean and easy to maintain.

Design Patterns

1. Model-View-Controller (MVC) pattern:

The code implements the MVC pattern by separating the application logic into three interconnected components:

* Model (Image class and ImageService interface)
* View (HTML templates)
* Controller (ImageController class).

1. RedirectView pattern

The RedirectView class is used to redirect the user to the homepage after saving the image.

1. Singleton pattern

This design pattern ensures that a class has only one instance and provides a global point of access to that instance. It's implemented using a bean with the @Component annotation and @Autowired to inject the instance of the Singleton class.

1. Template method design pattern

* is a behavioral design pattern that defines a skeleton of an algorithm in a base class and lets subclasses override specific steps of the algorithm without changing its structure.
* can be used to create a base class that defines the common algorithm for displaying photos, while allowing subclasses to implement specific steps for doing update/delete/view operations.

1. Front Controller design pattern

It is a widely used pattern in web application development that provides a centralized point of control for handling requests.It provides a unified entry point for all incoming requests.

1. Dependency injection design pattern:

* It is implemented using Spring’s inversion of control (IoC)container.The IoC container is responsible for creating and managing the lifecycle of objects in the application
* Inject the dependencies into the components that require them using the @Autowired annotation. This annotation tells Spring to automatically wire the required beans into the component at runtime

1. Proxy design pattern:

* Proxy Design Pattern can be used to control access to the images.
* The images are stored in the MySQL database, and since we are using JPA as an interface to MySQL , inherently the proxy object is used to manage access to the database content.

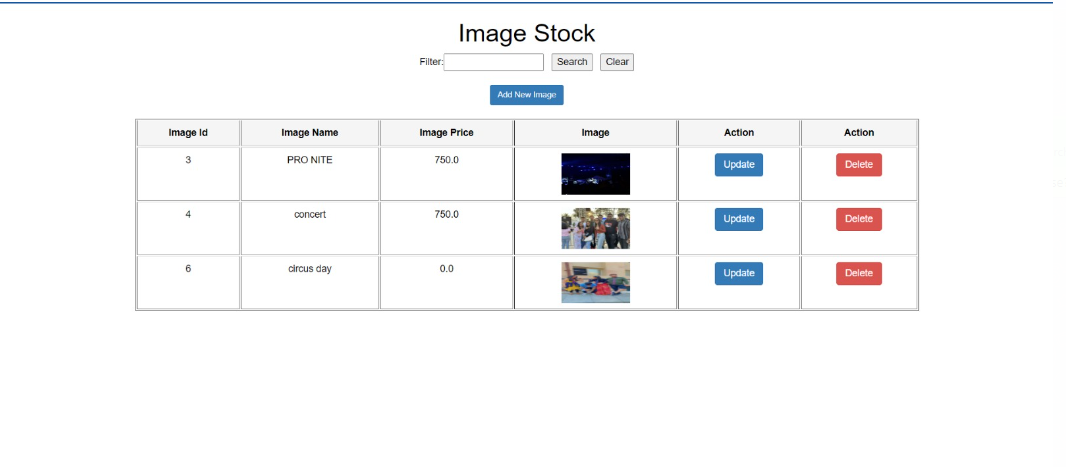
1. Decorator design pattern:

It is a structural design pattern that allows behavior to be added to an individual object, either statically or dynamically, without affecting the behavior of other objects from the same class. Here we used it as an InputStream is read in order to accept an image as an input.

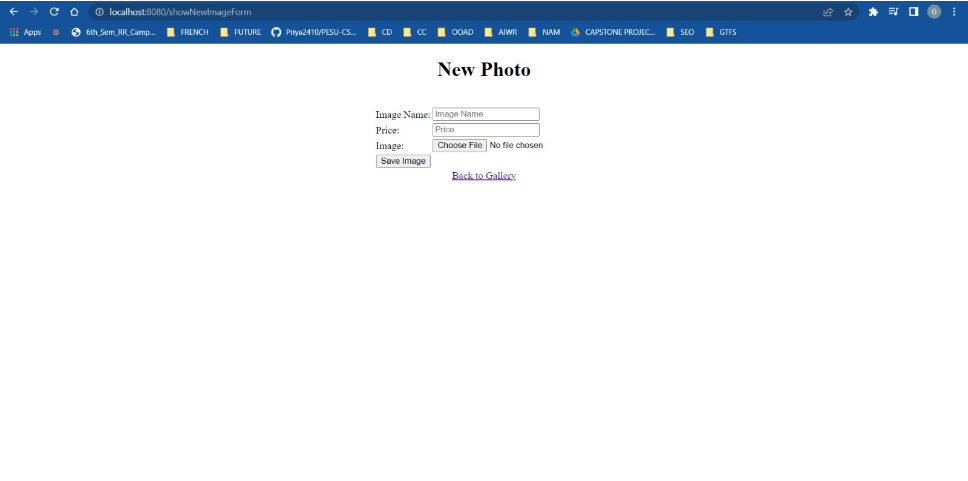
**Screenshots with input values populated and output shown (Use white**

**background screens)**

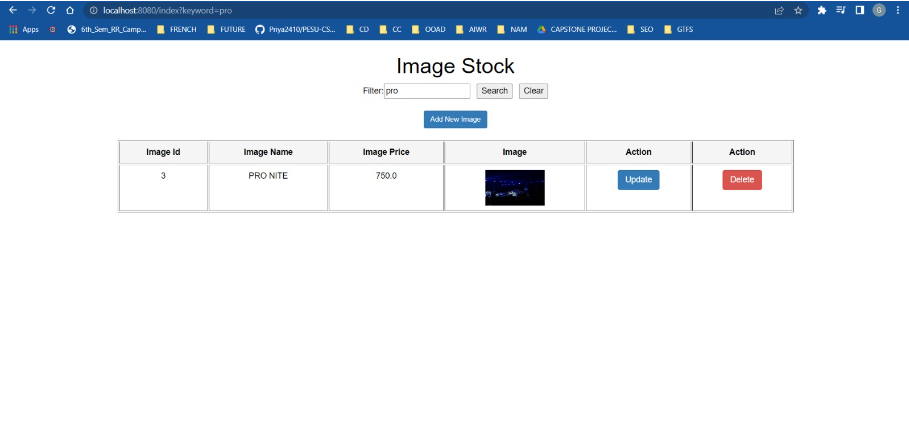
**View**

****

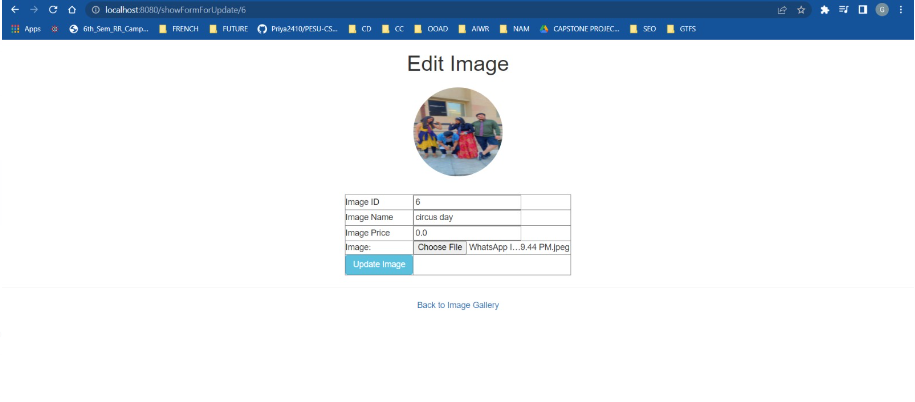
**Uploading an Image**

****

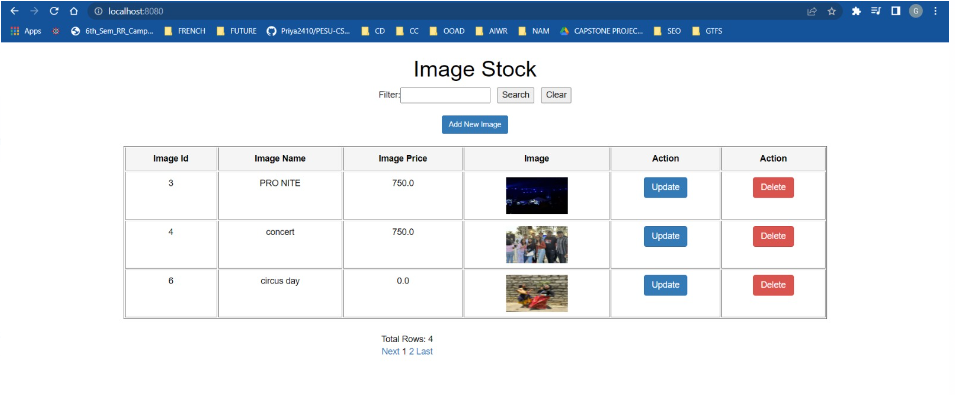
**Search for an Image**

****

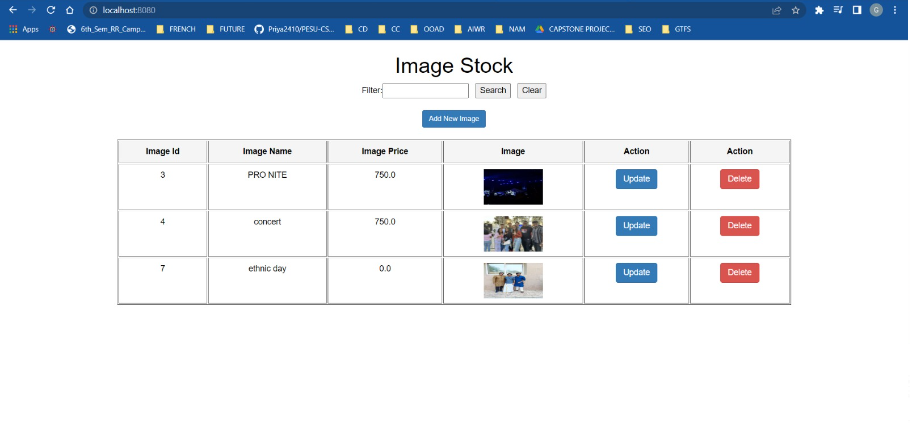
**Editing/Updating an Image**

****

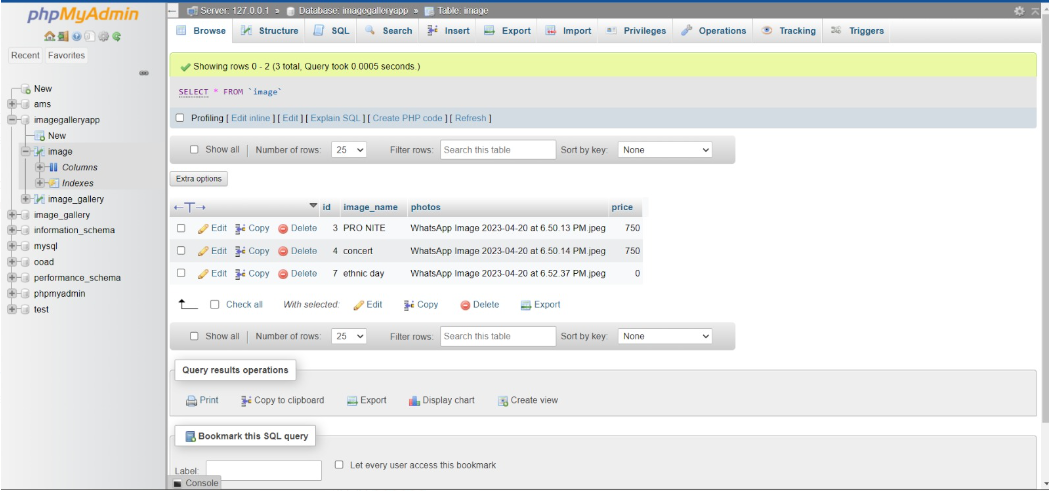
**Post Updation**

****

**Post Deletion**



**Database**



Github link:

<https://github.com/Gaurav-22-11/IMAGE-GALLERY-WEB-APP-USING-SPRINGBOOT-MVC>

Individual contributions of the team members:

Divija:

Deleting the image

Dhathri:

Searching the image

Gaurav:

Uploading the image

Jeevan:

Updating the image