Collaborative Image Editor

UE20CS352 OOADJ - Mini Project Report

Name: Abishek Deivam	SRN: PES1UG20CS012
Name: Allamaprabhu	SRN: PES1UG20CS031
Name: Amogh G S	SRN: PES1UG20CS034
Name: Aniruddha Kulkarni	SRN: PES1UG20CS055

SYNOPSIS

Overview:
The Image Editor project aims to develop a web application that enables users to edit images in various ways.
The features offered to the users will include manipulation, enhancement, and transformation of images.

Features:
The Image Editor application will have the following features:
Image manipulation:

- The user can crop, resize, rotate, and flip the image.
- They can also adjust brightness, contrast, and saturation, as well as apply filters like blur, sharpen, and grayscale.
- The user can draw on the image with different brush sizes, shapes, and colors. They can also add text to the image.
- The user can undo or redo any action to easily correct mistakes or revert changes.

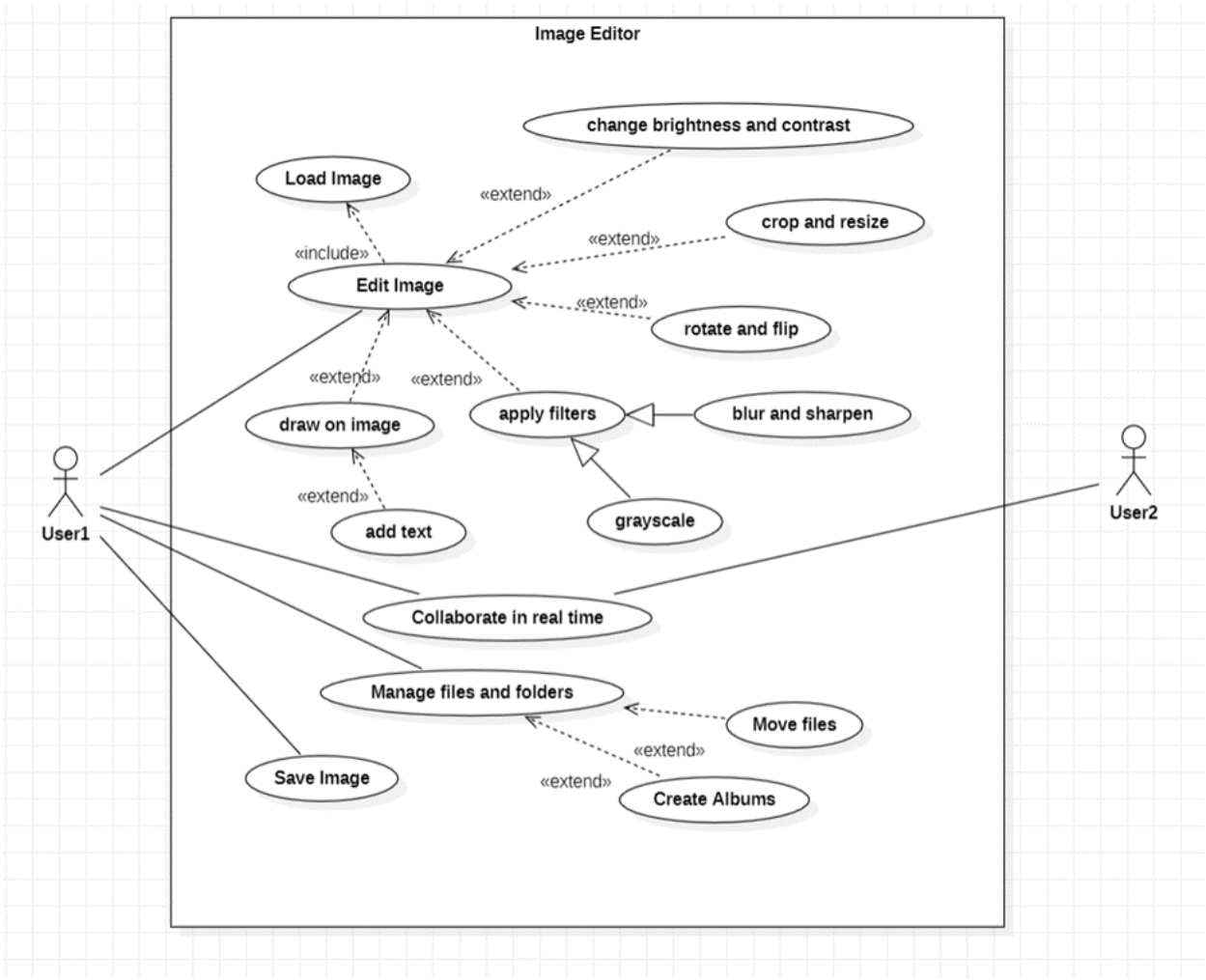
Save and export, folder management:

- The user can save the edited image in various file formats such as JPEG, PNG, and BMP.
- They can also export the image in different resolutions and sizes.
- Users can also create albums to store similar images.

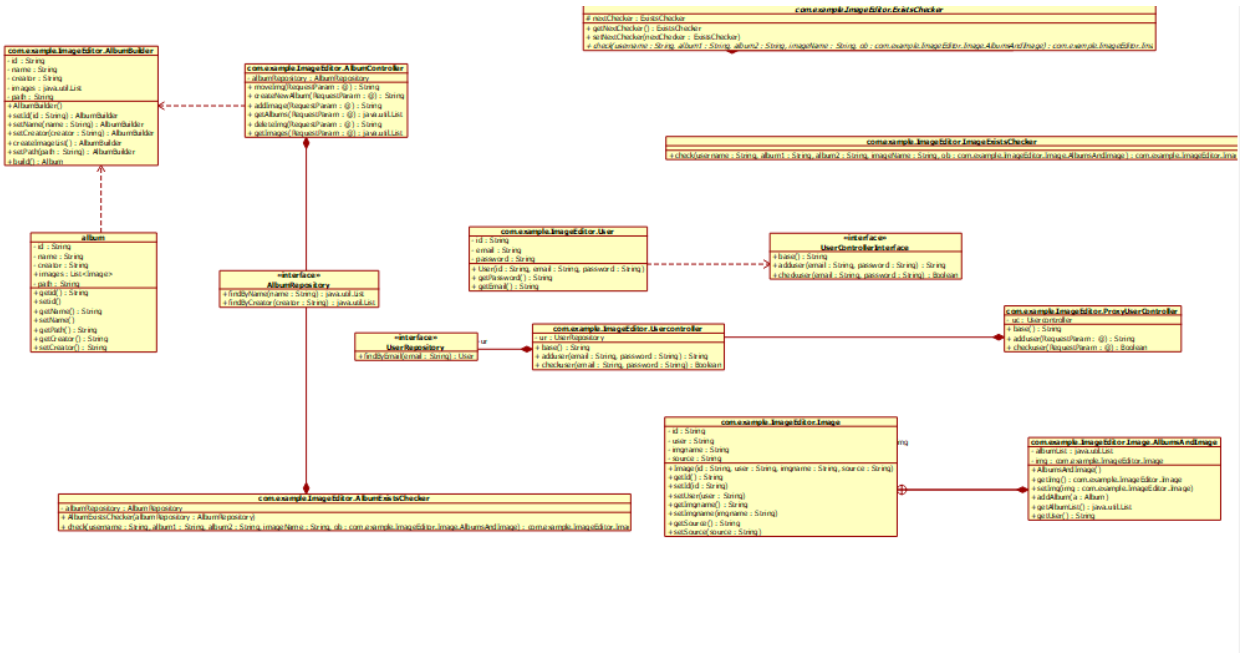
Collaborative image editing:

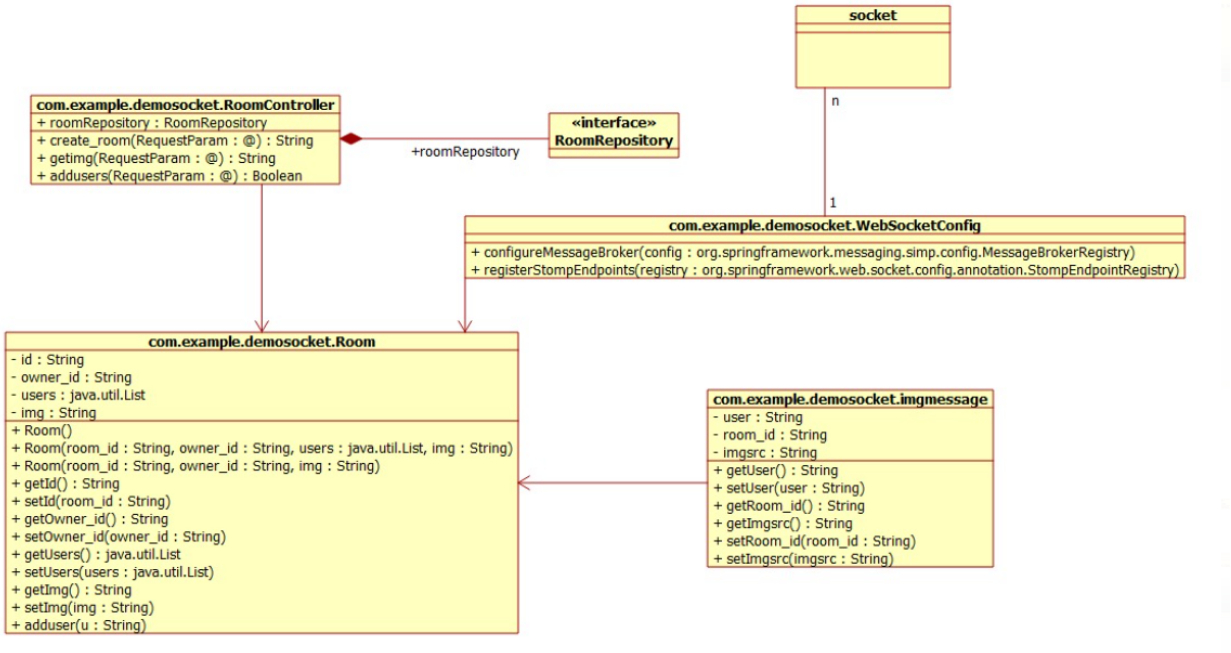
- Multiple users can simultaneously edit the same image at the same time.

Use Case Diagram

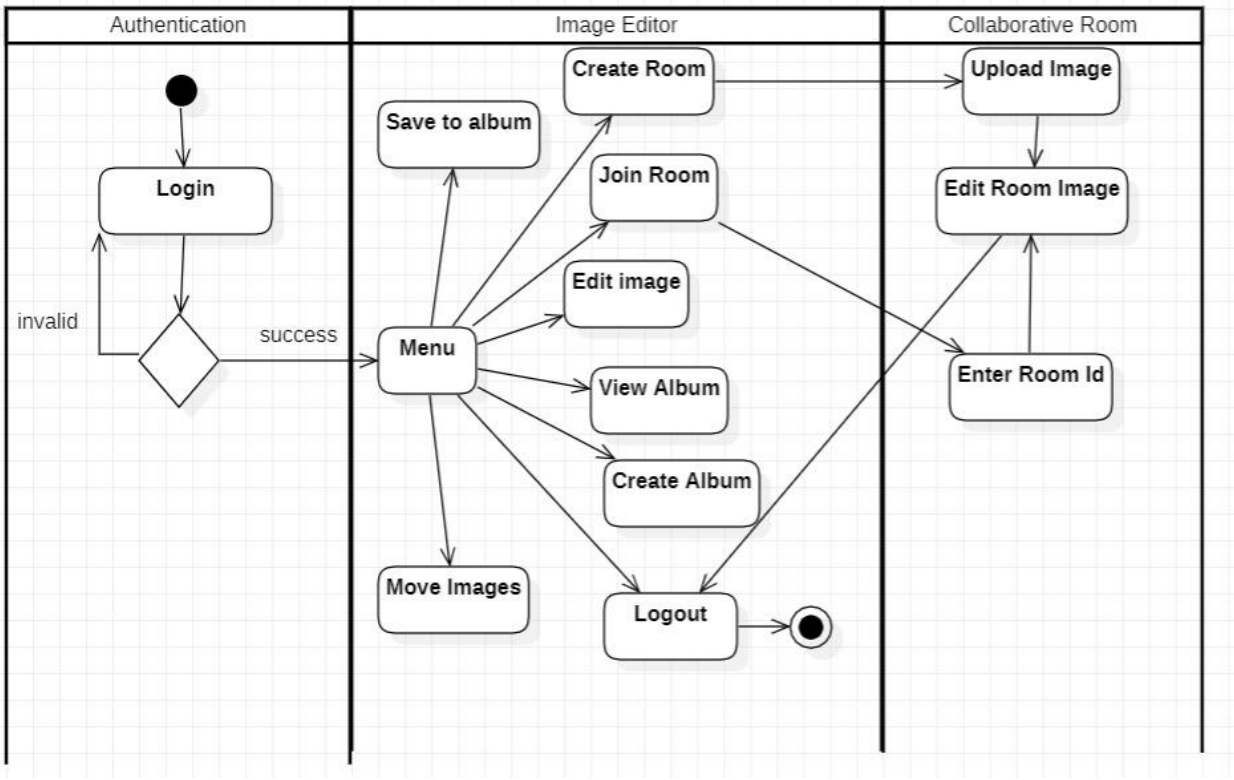


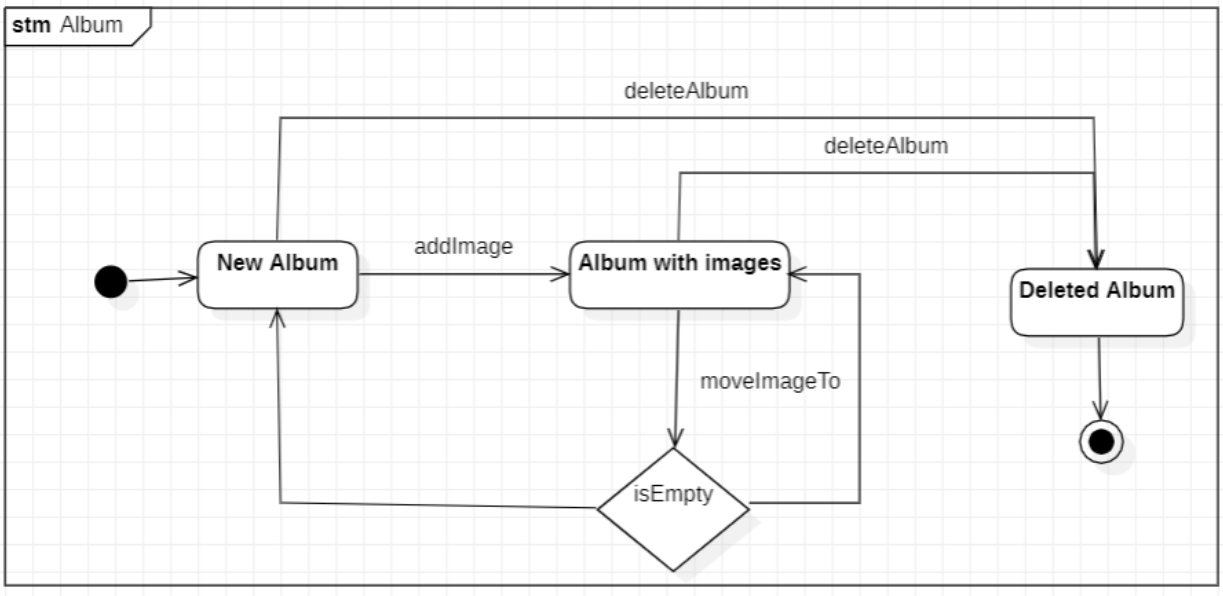
Class Diagram





Activity and State Diagram





Design Principles and Patterns Used

Design Principles Used:

- Single Responsibility Principle - Throughout this project, assigning the classes a single responsibility has been a priority. Every class handles only a certain specific kind of functionality, therefore, there is always only one reason to modify the class.

- Open-Closed Principle - The use of abstract classes ensures that we can always extend the functionality of it by subclassing it, and many such subclasses can be made to ensure high modularity and reusability.

Design Patterns Used:

- Observer pattern - The observer pattern was used to communicate changes in the room object to all the users connected to that room. This was implemented using sockets in a publish-subscribe model.

- Proxy Pattern - The proxy pattern was used for the authentication of the users during login. Any login request is received by the proxy controller and later forwarded to the main controller for authentication. This was implemented to ensure that the user does not flood the controller with multiple authentication requests.

- Builder pattern - The builder pattern was used to build an object of the class Album. Due to the complex nature of the Album class in the program, a step by step approach to create Album objects was necessary, thus making way for the builder pattern.