Online Picture Management

Objective:

Online Picture Management is an online application to be built as a A social-networking website that is to share Photos.

Users of the System:

1. Admin
2. User

Functional Requirements:

- Should provide a common platform where people can Share photos.
- Should provide for automatic tagging of pictures and categorise them.
- Should have the ability to tag social discipline violators using their UID
- Authenticity for adding users is utmost important for such a website.
- Definitely one should not be allowed to have more than one profile, validation of user should be done using email id.
- **The image should be in an encrypted format.**

While the above ones are the basic functional features expected, the below ones can be nice to have add-on features:

- ➢ There are lot of freeware & open source applications available for many social functions. The team is expected to search & leverage these to the maximum.
- ➢ Multi-factor authentication for the sign-in process

Output/ Post Condition:

- ➢ Reports for users
- ➢ Reports for Site admin..the operational reports to make the site better in future

Non-Functional Requirements:

Security	• App Platform –UserName/Password-Based Credentials • Sensitive data has to be categorized and stored in a secure manner • Secure connection for transmission of any data
Performance	• Peak Load Performance • Online Picture Management-< 3 Sec • Admin application < 2 Sec • Non Peak Load Performance
Availability	• 99.99 % Availability
Standard Features	• Scalability • Maintainability • Usability • Availability • Failover
Logging &	• The system should support logging(app/web/DB) & auditing at

Auditing	all levels
Monitoring	• Should be able to monitor via as-is enterprise monitoring tools
Cloud	• The Solution should be made Cloud-ready and should have a minimum impact when moving away to Cloud infrastructure
Browser Compatible	• IE 7+ • Mozilla Firefox Latest – 15 • Google Chrome Latest – 20 • Mobile Ready

Technology Stack

Front End	React Google Material Design Bootstrap / Bulma
Server Side	Spring Boot Spring Web (Rest Controller) Spring Security Spring AOP Spring Hibernate
Core Platform	OpenJDK 11
Database	MySQL or H2

Platform Pre-requisites (Do's and Don'ts):

1. The React app should run in port 8081. Do not run the React app in the port: 3000.
2. Spring boot app should run in port 8080.

Key points to remember:

1. The id (for frontend) and attributes(backend) mentioned in the SRS should not be modified at any cost. Failing to do may fail test cases.
2. Remember to check the screenshots provided with the SRS. Strictly adhere to id mapping and attribute mapping. Failing to do may fail test cases.
3. Strictly adhere to the proper project scaffolding (Folder structure), coding conventions, method definitions and return types.
4. Adhere strictly to the endpoints given below.

Application assumptions:

1. The login page should be the first page rendered when the application loads.

2. Manual routing should be restricted by using AuthGaurd by implementing the canActivate interface. For example, if the user enters as http://localhost:3000/signup or http://localhost:3000/home the page should not navigate to the corresponding page instead it should redirect to the login page.
3. Unless logged into the system, the user cannot navigate to any other pages.
4. Logging out must again redirect to the login page.
5. To navigate to the admin side, you can store a user type as admin in the database with a username and password as admin.
6. Use admin/admin as the username and password to navigate to the admin dashboard.

Validations:

1. Basic email validation should be performed.
2. Basic mobile validation should be performed.

Project Tasks:

API Endpoints:

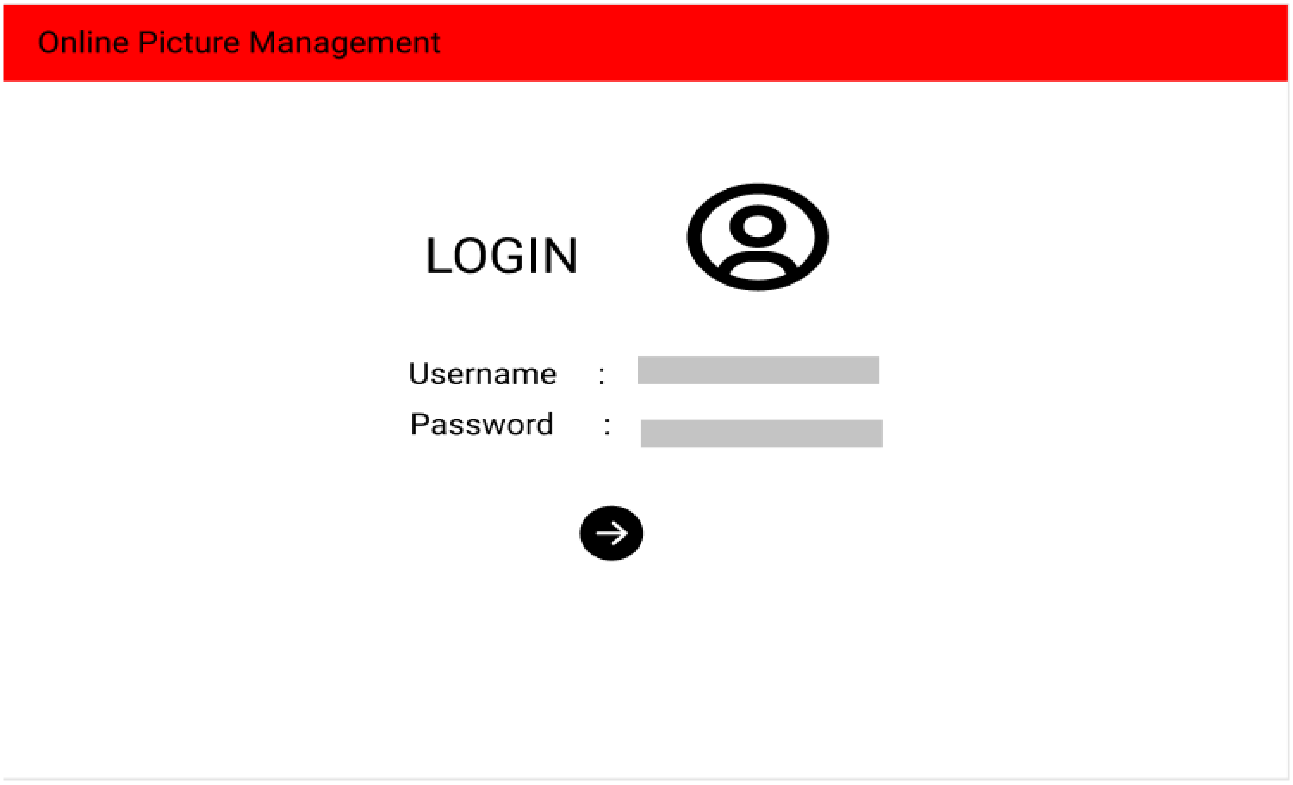
USER			
Action	URL	Method	Response
Login	/login	POST	true/false
Signup	/signup	POST	true/false
Get All image – Home	/image	GET	Array of image
Add Comment	/image/{id}	POST	Comment added to Image
Get Image By Id	/image/{id}	GET	Image Details
Delete Comment	/comment/delete	DELETE	Comment Deleted
Update Comment	/comment/update/{id}	PUT	Comment Updated
ADMIN			
Action	URL	Method	Response
Get All User	/admin	GET	Array of Users
Add User	/admin/add User	POST	User added
Delete User	/admin/delete/{id}	DELETE	User deleted
User Edit	/admin/userEdit/{id}	PUT	Save the Changes
Get All Image	/admin/image	GET	Array of Images
Delete Image	/admin/image/{id}	DELETE	Image deleted
Image Edit	/admin/imageEdit/{id}	PUT	Save the Changes
Get All Comment	/admin/comment	GET	Array of Comments

Frontend:

User:

Login:

Output Screenshot:


Online Picture Management

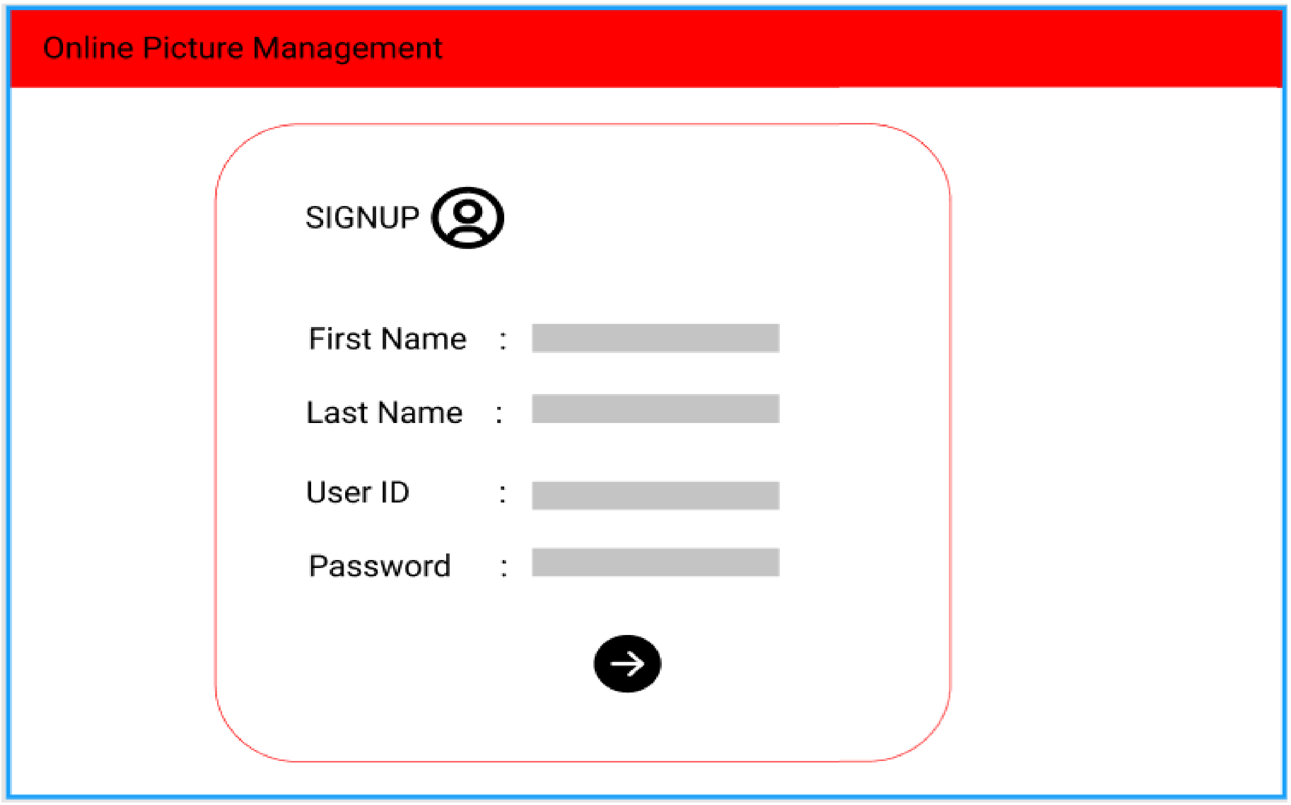
LOGIN

Username :

Password :

Signup:

Output Screenshot:


Online Picture Management

SIGNUP

First Name :

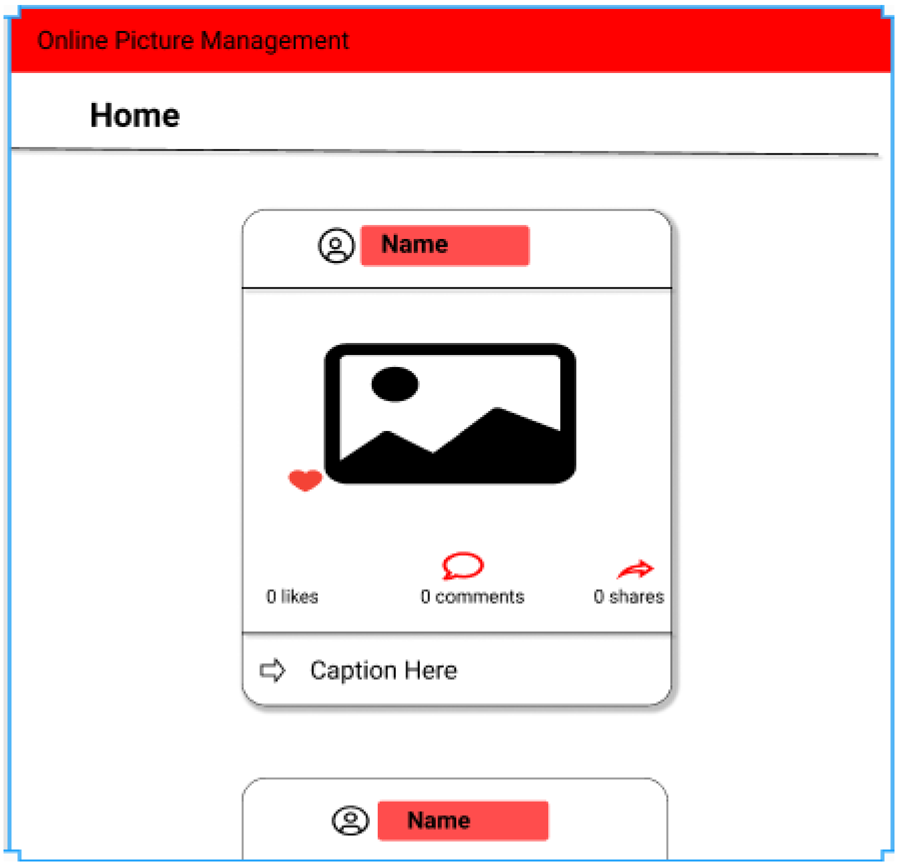
Last Name :

User ID :

Password :

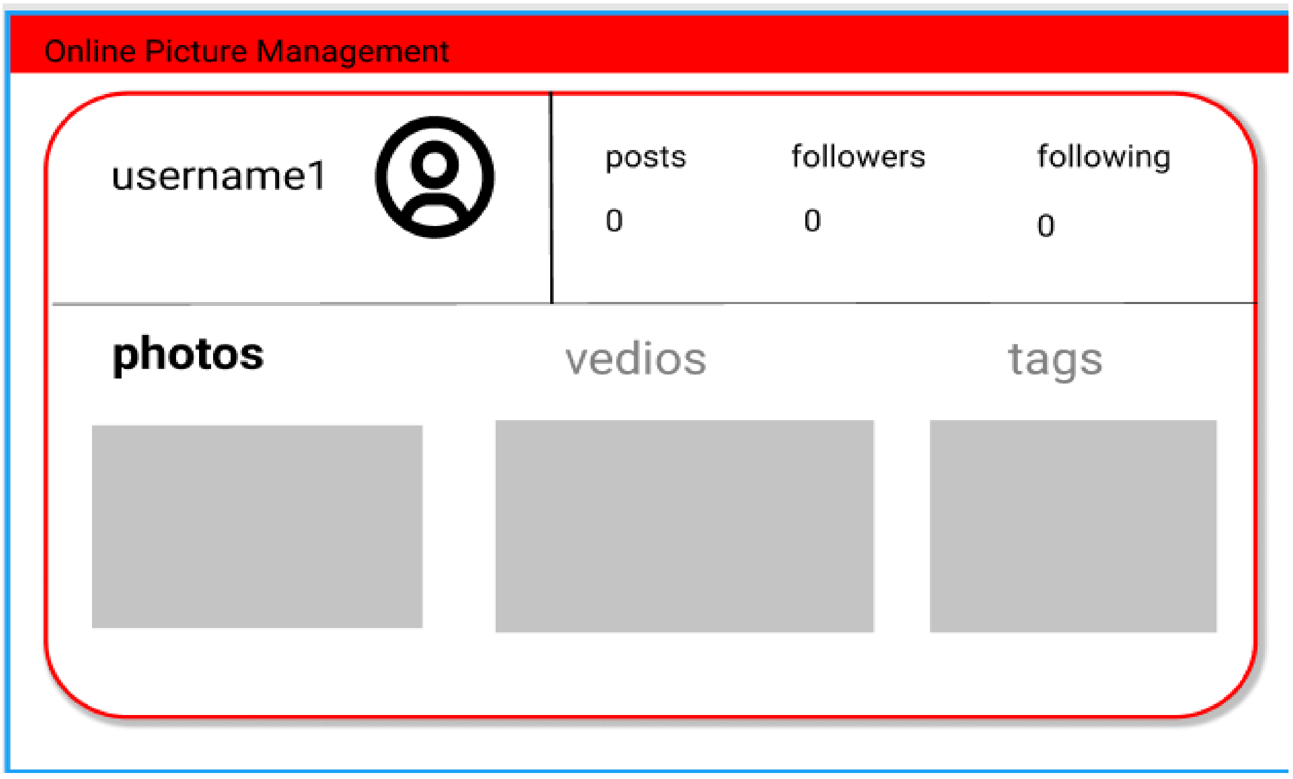
Home:

Output Screenshot:



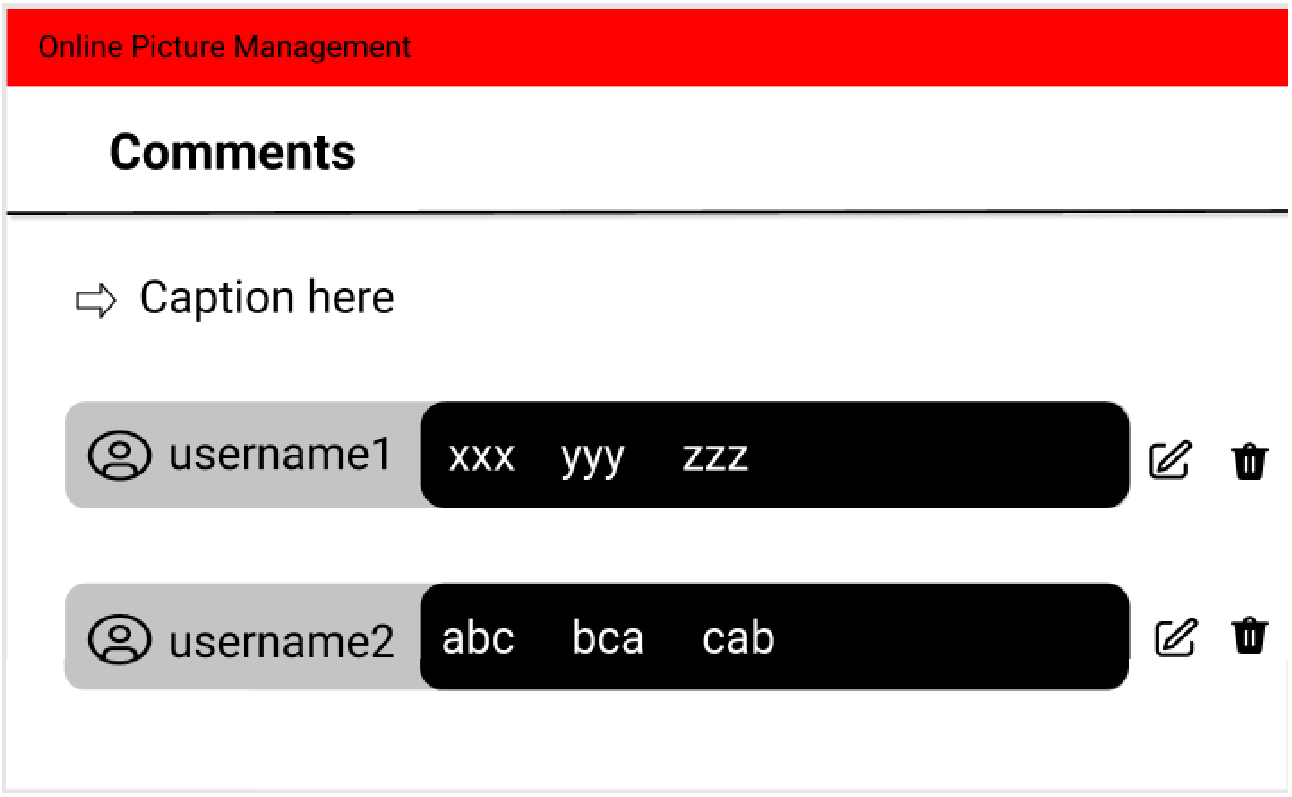
Profile:

Output Screenshot:



Comments:

Output Screenshot:



Online Picture Management

Comments

Caption here

username1	xxx yyy zzz
username2	abc bca cab

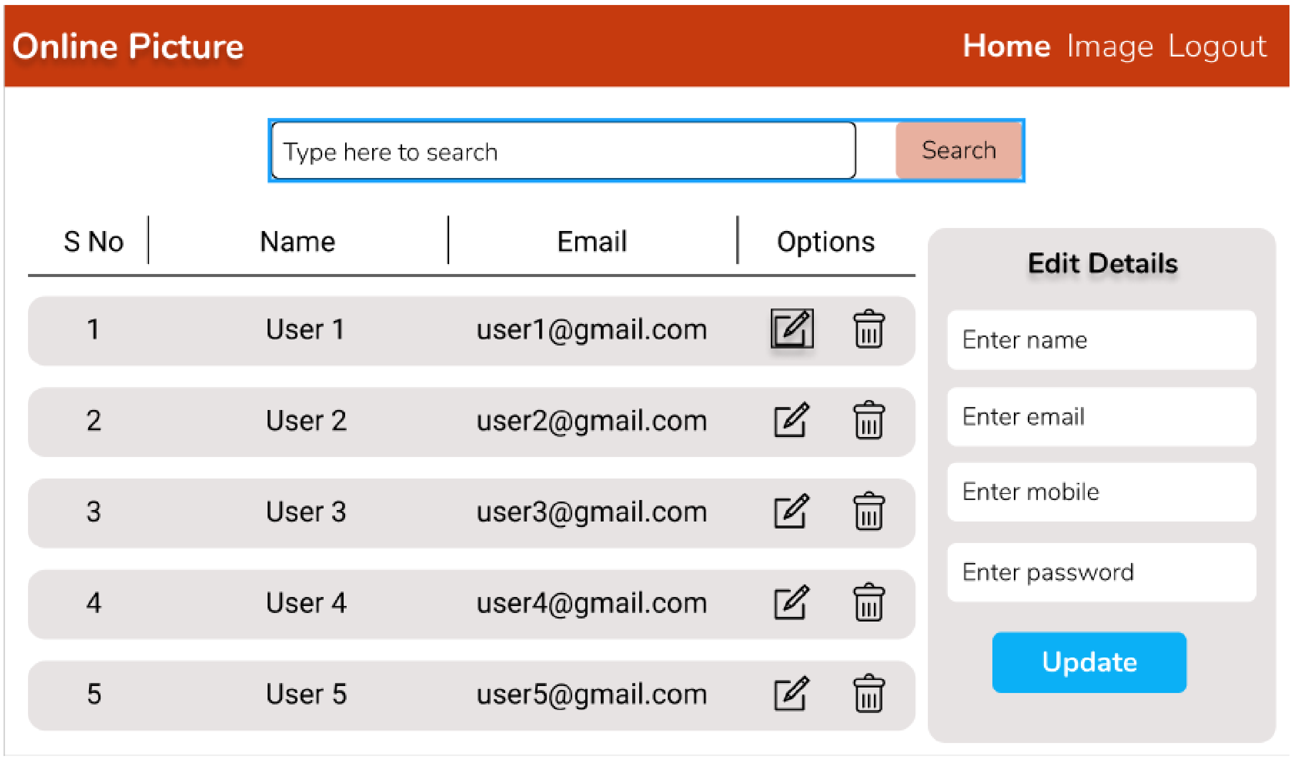
Admin:

Home:

Output Screenshot:

<u>Backend:</u>

<u>Class and Method description:</u>



Online Picture

Home Image Logout

Type here to search

Search

S No	Name	Email	Options
1	User 1	user1@gmail.com	
2	User 2	user2@gmail.com	
3	User 3	user3@gmail.com	
4	User 4	user4@gmail.com	
5	User 5	user5@gmail.com	

Edit Details

Enter name

Enter email

Enter mobile

Enter password

Update

<u>Model Layer:</u>

1. UserModel: This class stores the user type (admin or user) and all user information.
    a. Attributes:
        i. email: String
        ii. password: String
        iii. username: String
        iv. mobileNumber: String
        v. active: Boolean
        vi. role: String
    b. Methods: -

2. LoginModel: This class contains the email and password of the user.
    a. Attributes:
        i. email: String
        ii. password: String
    b. Methods: -
3. ImageModel: This class stores the details of the product.
    a. Attributes:
        i. imageId: String
        ii. imageName: String
        iii. image: Blob
        iv. imageTag: String
        v. comments: CommentModel
    b. Methods: -
4. CommentModel: This class stores the cart items.
    a. Attributes:
        i. commentId: String
        ii. Comment: String
        iii. userId: UserModel

   b. Methods: -

Controller Layer:

5. SignupController: This class control the user signup
   a. Attributes: -
   b. Methods:
      i. saveUser(UserModel user): This method helps to store users in the database and return true or false based on the database transaction.
6. LoginController: This class controls the user login.
   a. Attributes: -
   b. Methods:
      i. checkUser(LoginModel data): This method helps the user to sign up for the application and must return true or false
7. UserController: This class controls the add/edit/update/view User.
   a. Attributes: -
   b. Methods:
      i. List<UserModel> getUser(): This method helps the admin to fetch all User from the database.
      ii. List< UserModel > getOnlineUser(): This method helps to retrieve all the online user from the database.
      iii. UserModel userEditData(String id): This method helps to retrieve a user from the database based on the user id.
      iv. userEditSave(UserModel data): This method helps to edit a user and save it to the database.
      v. userDelete (String id): This method helps to delete a user from the database.
8. ImageController: This class helps in adding product to the cart, deleting the Image from the cart, updating items in the cart.
   a. Attributes: -
   b. Methods:
      i. addImage(Blob image): This method helps the user to add the image to the database.
      ii. ImageModel showImage(String id): This method helps to view the image.
      iii. deleteImage(String id): This method helps to delete a image from the database.
      iv. updateImage(String id): This method helps to update a image from the database.

9. CommentController: This class helps in adding, deleting, updating the comment.
    a. Attributes: -
    b. Methods:
        i. addComment(String Id): This method helps the user to add the Comment to the database.
        ii. deleteComment(String id): This method helps to delete a Comment from the database.
        iii. updateComment(String id): This method helps to update a Comment from the database.