PES University, Bengaluru

Department of Computer Science & Engineering

Session: Jan - May 2023

Subject: Object Oriented Analysis and Design with Java

Title: SOCIAL MEDIA CLONE

Team members:

Aaditya Vikram Saravana Bhavan	***PES1UG20CS528***
Dhruv Garodia	***PES1UG20CS527***
Vishnushree Menon	***PES1UG20CS510***
Nikhil Ravallu	***PES1UG21CS830***

Project Description

The goal of this project is to develop a social networking web application with various features such as login, logout, registration, profile viewing, messaging, friend management, and search functionality. The application is to be developed using the Spring Boot framework and will support internationalization in English and Russian languages, as well as localization for different regions.

The login and logout functionality will allow users to securely access their account, while the registration feature will enable new users to create a profile. Once logged in, users will be able to view their own profile, edit their information, and upload a profile image via drag and drop. They will also

be able to view other users' profiles and send friend requests.

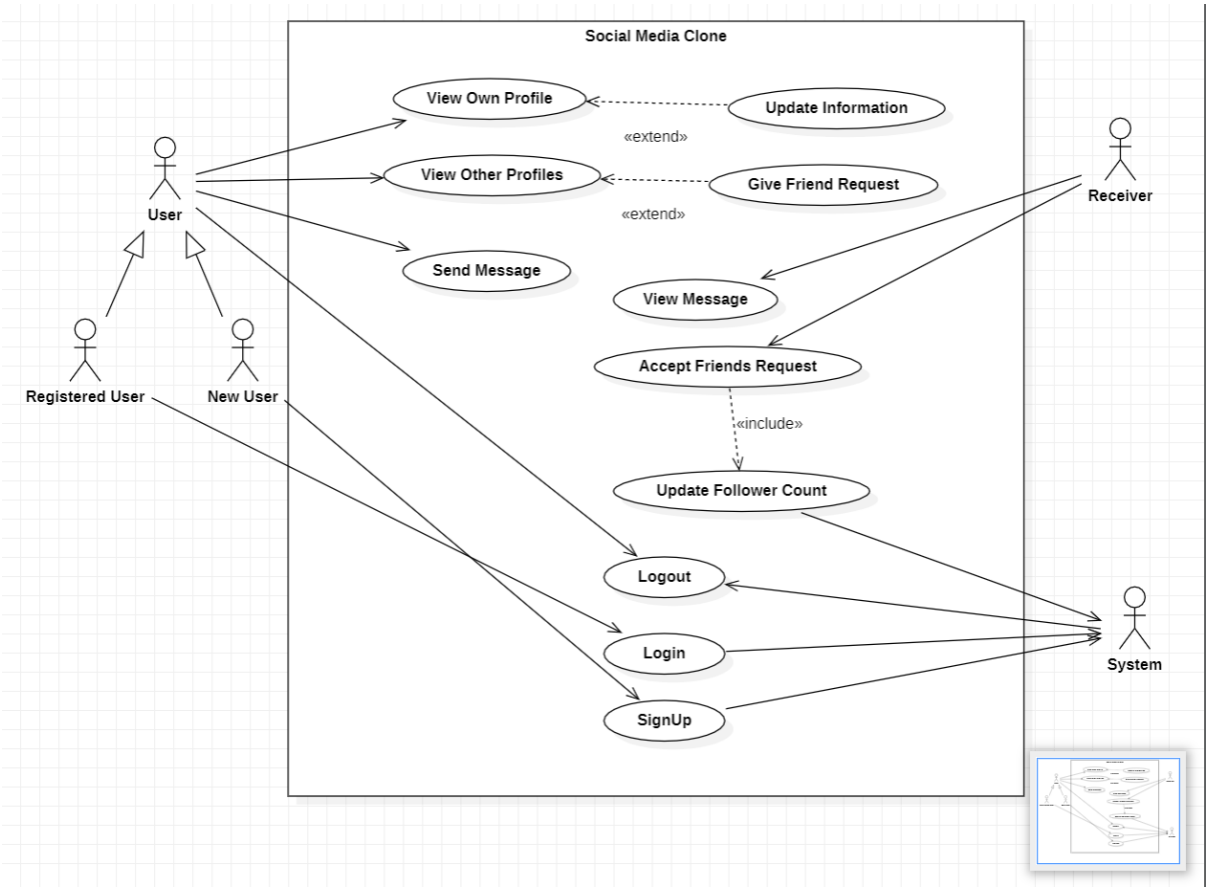
Friend management will be a key feature of the application, allowing users to add and remove friends, as well as accept or decline friend requests. Users will be able to view their friends' profiles and messaging history. The messaging feature will allow users to send messages to their friends and view their message history with individual friends.

The application will also include a search function that will enable users to find other users based on specific criteria, such as name, location, or interests. Users will also be able to view the friends lists of other users.

To ensure security and privacy, users will be able to change their password and update their profile information at any time. The application will also support internationalization in English and Russian languages, allowing users to select their preferred language for the interface.

Overall, this social networking web application will provide a comprehensive and user-friendly platform for users to connect with others, manage their friendships, and stay in touch with their messaging history. Using Spring Boot, this application will be robust, scalable, and easily maintainable, making it an ideal choice for large-scale social networking platforms.

I. Use Case Diagram:



Use Case: Login

Description: This use case allows registered users to access their accounts by providing their login credentials.

Alternate Flow:

If the user enters incorrect login credentials, the system will display an error message and prompt the user to enter correct login credentials.

If the user has forgotten their password, they can click on the "forgot password" link to reset their password. The system will prompt the user to provide their registered email address or username, and send a password reset link to their email.

If the user's account has been locked due to multiple unsuccessful login attempts, the system will display a message informing the user that their account has been locked and to contact the administrator for assistance.

Exception Flow:

If the user does not have a registered account, the system will display an error message informing them that they need to register first before they can access their account.

If there is an issue with the system or server, the system will display an error message informing the user to try again later.

If the user's account has been deactivated, the system will display an error message informing the user that their account has been deactivated and to contact the administrator for assistance.

Use Case: Logout

Description: This use case allows the user to securely log out of their account, ensuring the privacy and security of their account.

Alternate Flow:

If the user is on a public or shared device, they will be prompted to confirm that they want to log out to prevent unauthorized access to their account.

If the user has not saved their work or completed a task, they will be prompted to save their work or complete the task before logging out.

Exception Flow:

If the user is experiencing a technical issue or network interruption, the system may not be able to complete the logout process, and the user may be required to log out manually or try again later.

If the user has already logged out or their session has expired, the system will display an error message and prompt the user to log in again to access their account.

Use Case: Register

Description: This use case allows new users to create a new account on the system.

Alternate Flow:

If the user enters invalid or incomplete information, the system will display an error message and prompt the user to correct the information.

If the user's chosen username or email address is already taken, the system will display an error message and prompt the user to choose a different one.

If the user has entered all the required information correctly, the system will create their account and redirect them to the login page.

Exception Flow:

If there is an issue with the system or server, the system will display an error message informing the user to try again later.

If the user is under the age of 13, the system may not allow them to register due to legal restrictions.

Use Case: View Own Profile

Description: This use case allows the user to view their own profile information, such as their username, profile picture, and other personal information.

Alternate Flow:

If the user wants to edit their profile information, they can click on the "edit profile" button to make changes.

If the user has not completed their profile, the system will prompt them to complete it before allowing them to view their profile.

Exception Flow:

If the user is not logged in, the system will redirect them to the login page to log in first before they can view their own profile.

Use Case: View Other Users Profiles

Description: This use case allows the user to view other users' profile information, such as their username, profile picture, and other personal information.

Alternate Flow:

If the user wants to add the user as a friend, they can click on the "add friend" button to send a friend request.

If the user is not friends with the user they are trying to view, they may not be able to see certain information, such as their friends list.

Exception Flow:

If the user does not have permission to view the other user's profile, the system will display an error message informing the user that they are not authorized to view the profile.

Use Case: Send Messages

Description: This use case allows the user to send messages to other users.

Alternate Flow:

If the user wants to send a message to a user they are not friends with, they will need to send a friend request first before they can start messaging.

If the user wants to send a message to a group, they can create a group chat and add members to the chat.

Exception Flow:

If there is an issue with the system or server, the message may not be delivered, and the user may need to try sending the message again later.

Use Case: Add and Remove Friends

Description: This use case allows the user to add and remove friends from their friends list.

Alternate Flow:

If the user wants to add a friend, they can search for the user and send a friend request. The other user will then need to accept the request before they become friends.

If the user wants to remove a friend, they can go to their friends list and select the friend they want to remove, then click on the "remove friend" button.

Exception Flow:

If the user is not friends with the user they are trying to remove, the system will display an error message informing the user that they are not authorized to remove the friend.

If the user has sent a friend request, but the other user has declined the request, the system will display an error message informing the user that the request was declined and they cannot add the user as a friend.

Use Case: Accept and Decline Friend Requests

Description: This use case allows the user to accept or decline friend requests from other users.

Alternate Flow:

If the user wants to accept a friend request, they can go to their friend requests and click on the "accept" button. The user who sent the request will then become a friend.

If the user wants to decline a friend request, they can go to their friend requests and click on the "decline" button. The user who sent the request will not become a friend.

Exception Flow:

If the user has already accepted or declined a friend request, the system will display an error message informing the user that the request has already been processed.

Use Case: Update Profile Information

Description: This use case allows the user to update their profile information, such as their name, profile picture, and other personal information.

Alternate Flow:

If the user wants to change their profile picture, they can upload a new image by dragging and dropping it onto the profile picture.

If the user wants to change other information, such as their name or bio, they can edit the information and save the changes.

Exception Flow:

If the user enters invalid or incomplete information, the system will display an error message and prompt the user to correct the information.

If the user tries to change their username, the system will check if the new username is available and display an error message if it is already taken.

Use Case: Change Password

Description: This use case allows the user to change their account password.

Alternate Flow:

If the user has forgotten their password, they can click on the "forgot password" link and follow the steps to reset their password.

If the user remembers their current password, they can enter it along with their new password and save the changes.

Exception Flow:

If the user enters an incorrect current password, the system will display an error message and prompt the user to enter the correct password before they can change their password.

Use Case: Search for Other Users

Description: This use case allows the user to search for other users by username or other personal information.

Alternate Flow:

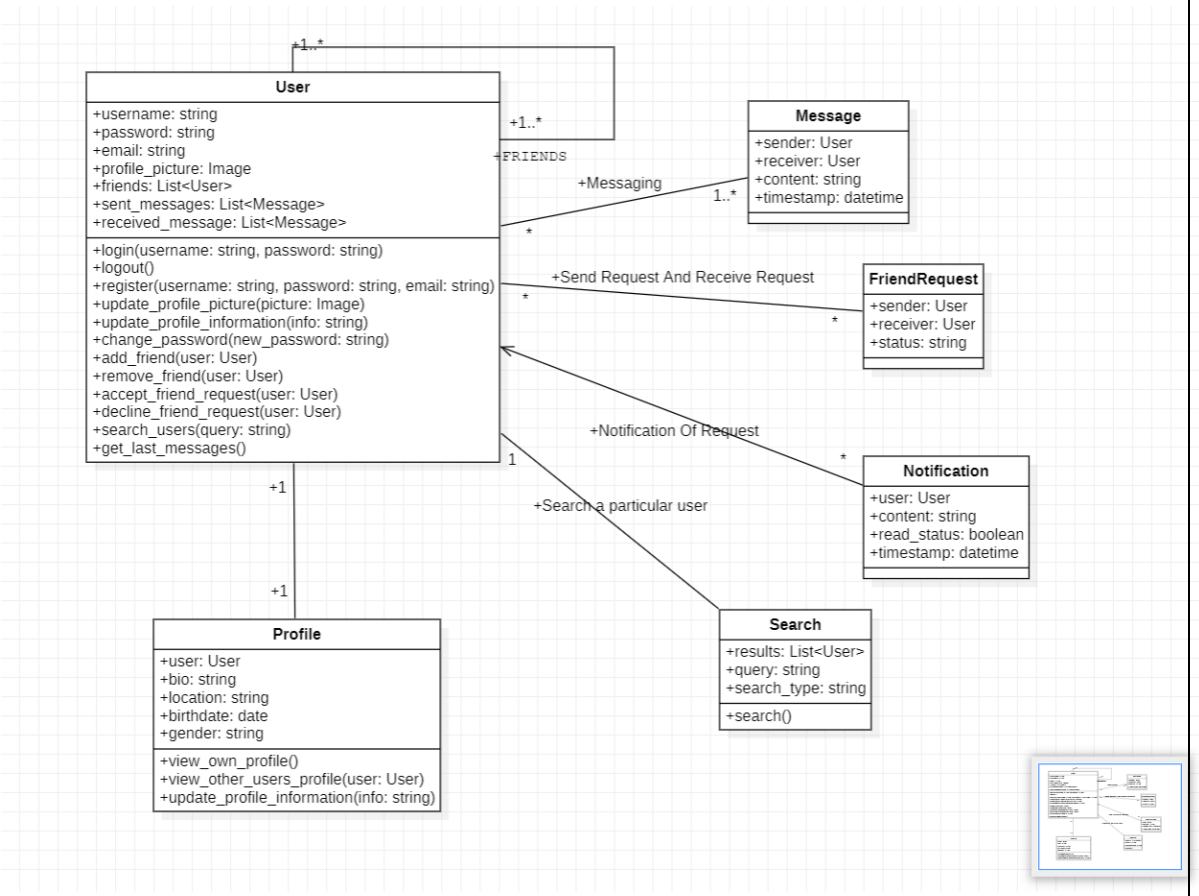
If the user wants to filter their search results, they can use filters such as location, age, or interests.

If the user finds a user they want to connect with, they can send a friend request or message them directly.

Exception Flow:

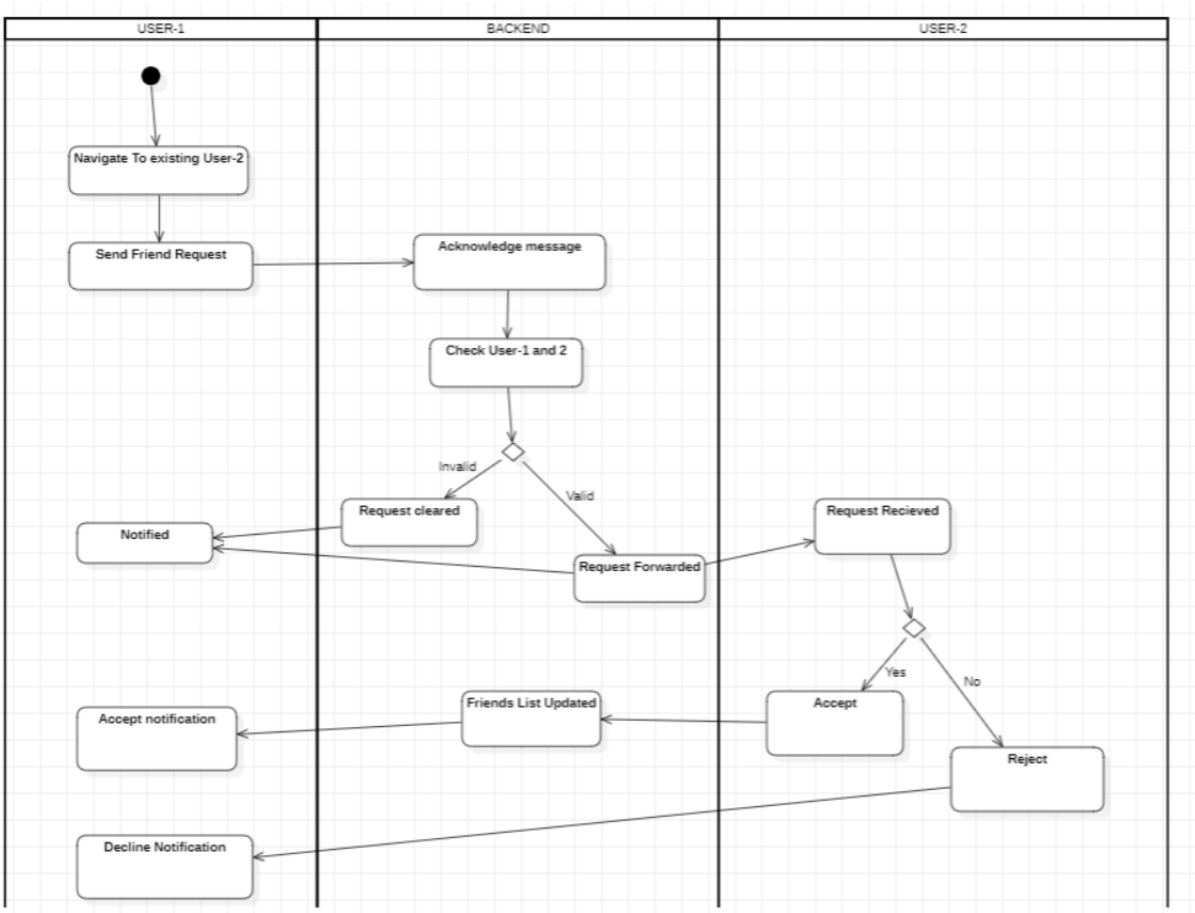
If there are no matching results for the user's search, the system will display a message informing the user that no results were found.

II. Class Diagram:

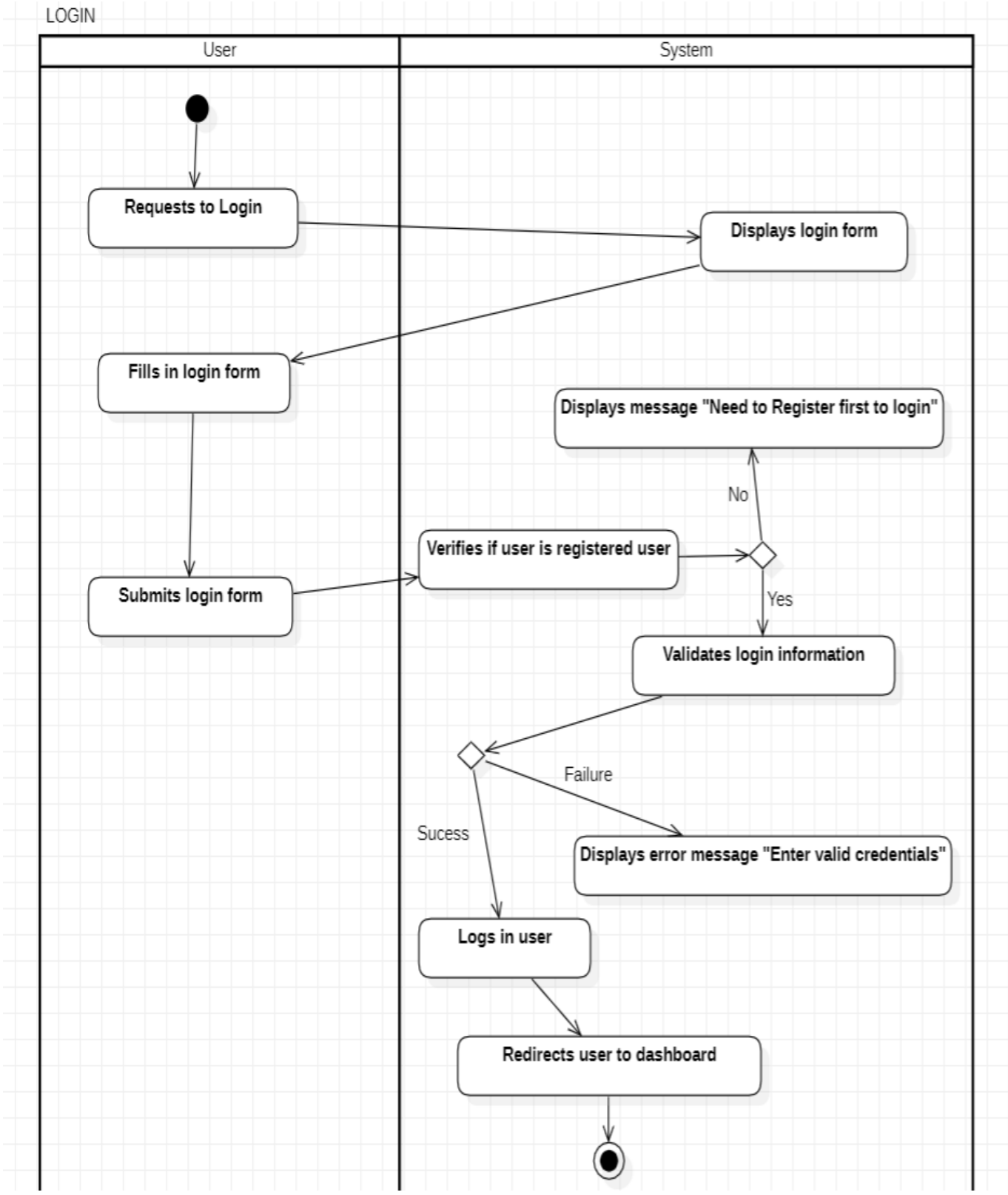


III. Activity Diagram:

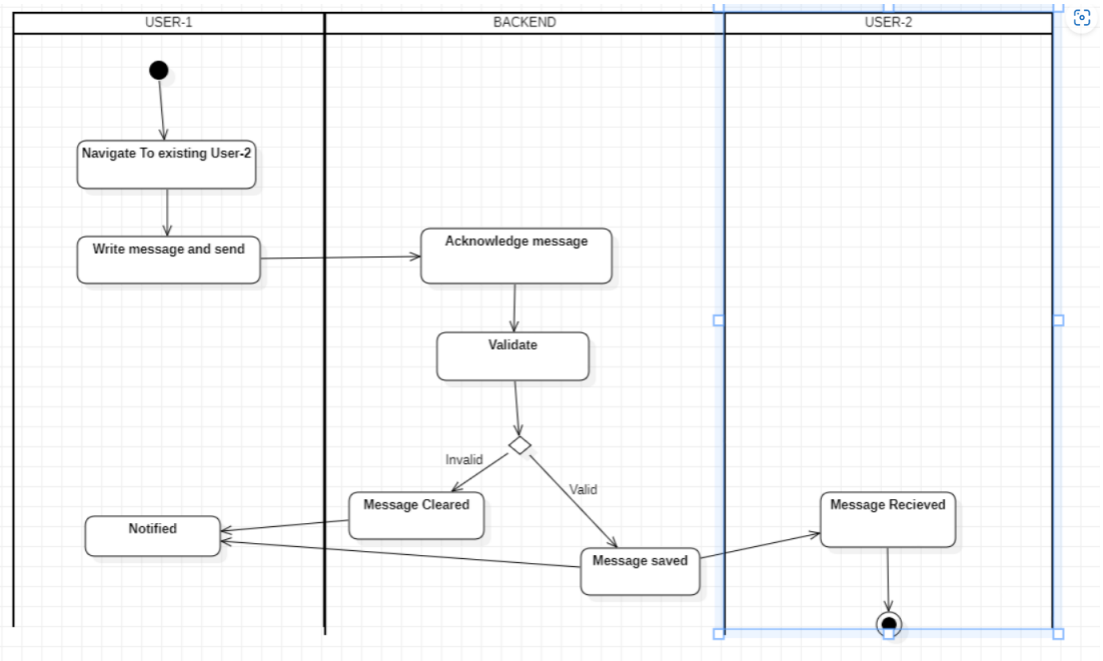
Name: Friend Request


USER-1
BACKEND
USER-2
Navigate To existing User-2
Send Friend Request
Acknowledge message
Check User-1 and 2
Invalid
Valid
Request cleared
Notified
Request Forwarded
Request Recieved
Yes
No
Accept
Friends List Updated
Accept notification
Reject
Decline Notification

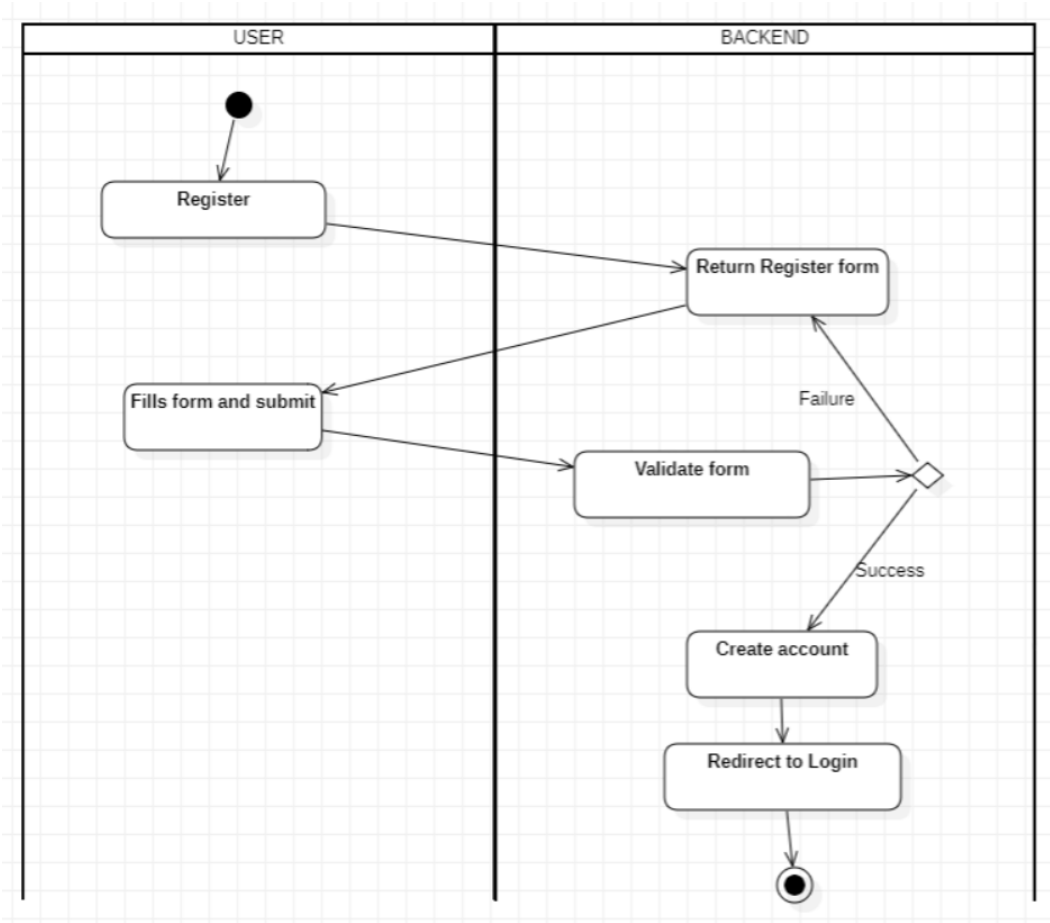
Name: Login



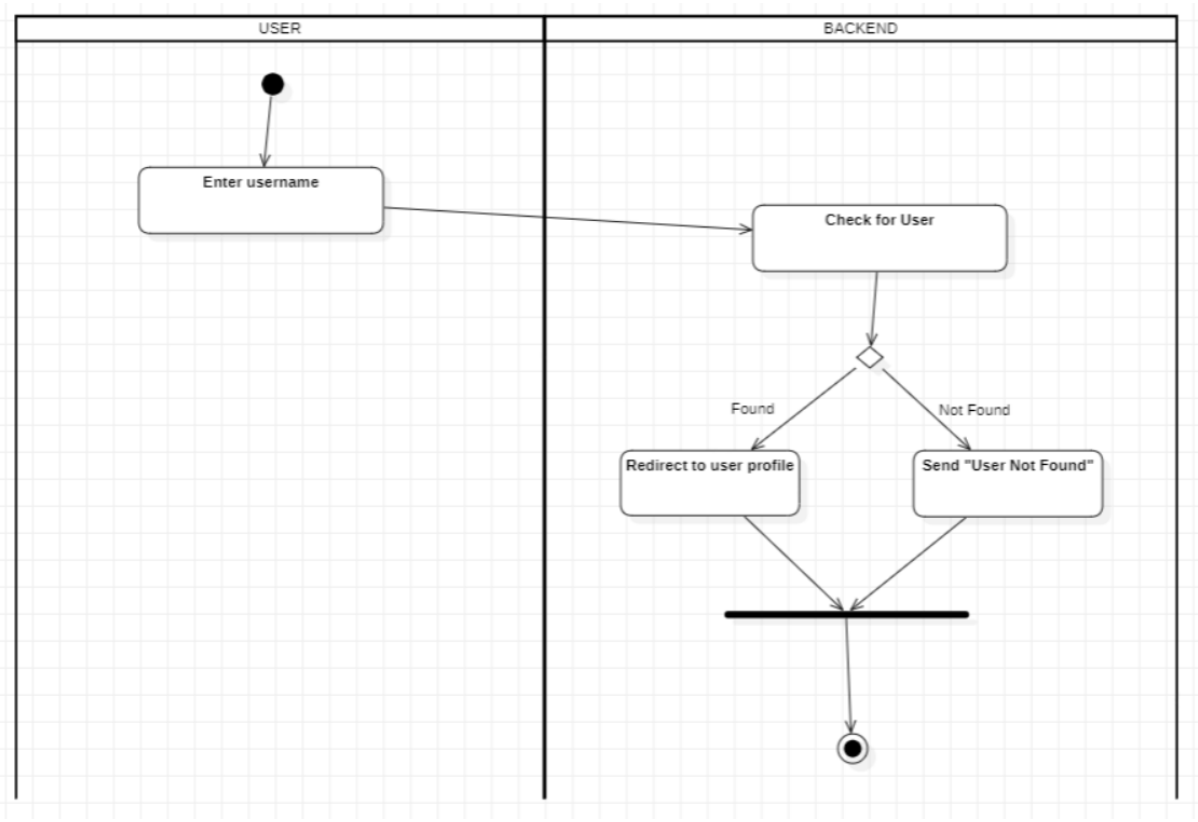
Name: Messaging



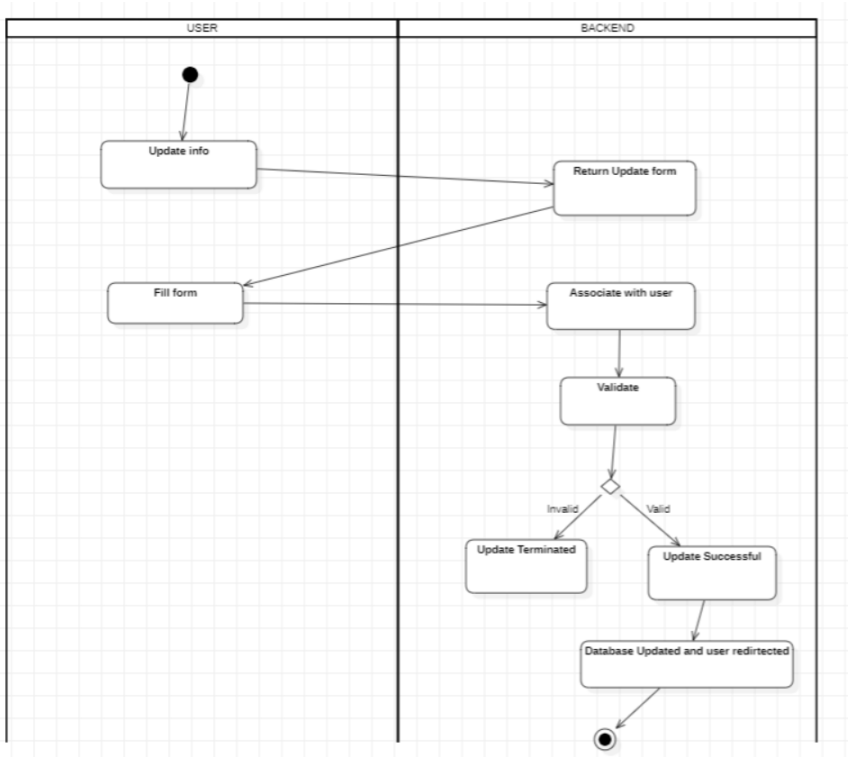
Name: Register



Search User



Update Profile



MVC Architecture Pattern: The MVC architecture pattern can be applied to separate the application into three interconnected parts: the Model, the View, and the Controller.

Model: This component is responsible for data storage and business logic. In this case, the Model would handle user account information, friend lists, messages, and profile images.

View: This component is responsible for displaying data to the user. In this case, the View would handle displaying user profile information, friend lists, messages, and profile images.

Controller: This component is responsible for handling user requests and updating the Model as needed. In this case, the Controller would handle login/logout requests, user registration requests, messaging requests, friend requests, profile information updates, and profile image uploads.

Design Principles: Several design principles can be applied to ensure that the application is well-structured and maintainable:

Separation of Concerns: Different components of the application should be responsible for different areas of functionality. In this case, the Model should handle data storage and business logic, the View should handle displaying data to the user, and the Controller should handle user requests and update the Model as needed.

Single Responsibility: Each component of the application should have a single responsibility. For example, the Controller should be responsible for handling requests from the View and updating the Model, but not for handling data storage.

Dependency Injection: Dependencies should be injected into a class rather than created within the class. This makes the code more modular and easier to test.

Design Patterns: Several design patterns can be applied to improve the application's maintainability and flexibility:

Singleton pattern: A centralized authentication system is created to handle user login and logout requests. This ensures that only one instance of the authentication system is created and used throughout the application.

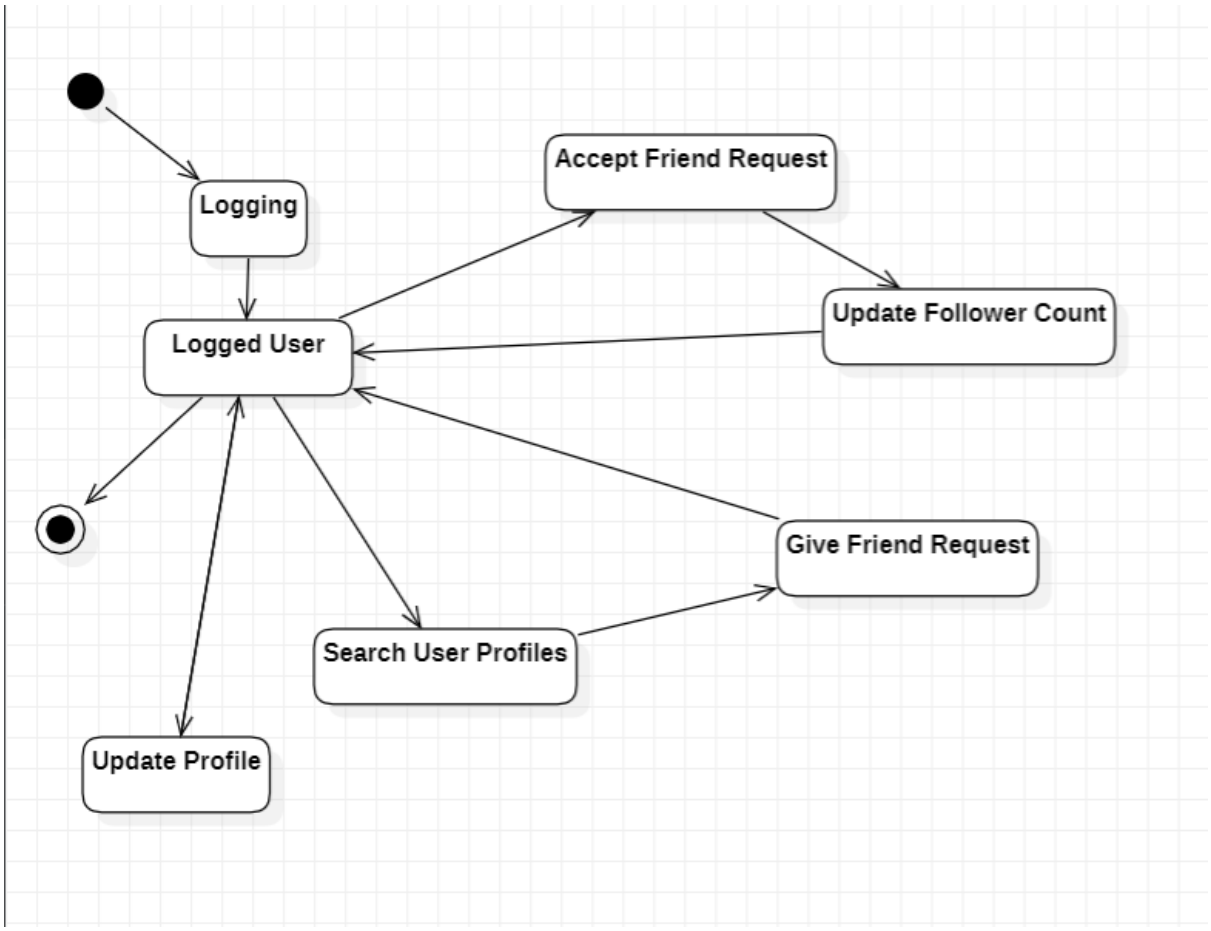
Factory pattern: A factory is used to create user profiles and handle user registration requests. This allows for flexible and configurable creation of user profiles based on different criteria.

Observer pattern: The Observer pattern is used to notify the View component of changes to the Model component, such as when a user updates their profile information or accepts a friend request.

Strategy pattern: The Strategy pattern is used to handle different authentication strategies, such as password-based authentication or social media-based authentication.

Template method pattern: The Template Method pattern is used to define the basic structure of the authentication and registration processes while allowing subclasses to provide specific implementations for different types of authentication or registration methods.

IV. State Diagram:



Team Contributions:

PES1UG20CS528	Worked On the integration of controller code with the front end. Worked primarily as a mediator to see if the contents were stored in the required format of the database.
PES1UG20CS527	Worked on the front end and ensured that all the design principles were being followed while building the layout of the website.
PES1UG20CS510	Worked on the front end and bootstrapping. Came up with ideas of what design principles will be suitable for our project and tested the feasibility for the same. Ensured that both the frontend and backend were following the principles.
PES1UG20CS830	Worked on the backend, helped in building the model, and code out the design principles. Also helped in coding some parts of the controller code.