UE20CS352: OOAD Mini Project

Project Title: REDDIT CLONE USING JAVA

Team Details:

SRN: **PES1UG20CS370** Name: **SAMARTH RAJENDRA**
SRN: **PES1UG20CS372** Name: **SANATH N BHARGAV**
SRN: **PES1UG20CS388** Name: **SATHWICK P**

This document contains the following -

Project Synopsis (Problem statement definition)

Use case diagrams

Use case specifications of four important

use cases

Class models

Architecture Patterns

Design Principles and Design patterns used (use case + desc)

Github link to code base

Screenshots of input and output

Individual contributions of team members

Project Synopsis (Problem Statement Definition)

->A Reddit clone in Java is a web application that allows users to ask and answer posts related to programming and technology. The application is built using the Java programming language and utilizes various technologies such as MySQL database.

->The application includes features such as user registration and authentication, posting and answering posts, commenting on posts, upvoting and downvoting comments and posts, searching for posts and receiving notifications.

->Each post has a particular status like open,closed etc which are the indicators of whether that particular post is answered orwhether it is still open to write an answer to.

->There is a post closing remark which says whether thepost was duplicate off topic not a real post etc.

->There are comments which we can add to the posts while writing an answer to it also there are tags to the post.

->There is an account class where there is information regarding a user where we can view all posts he posted. There is an option to change password email and phone number too.

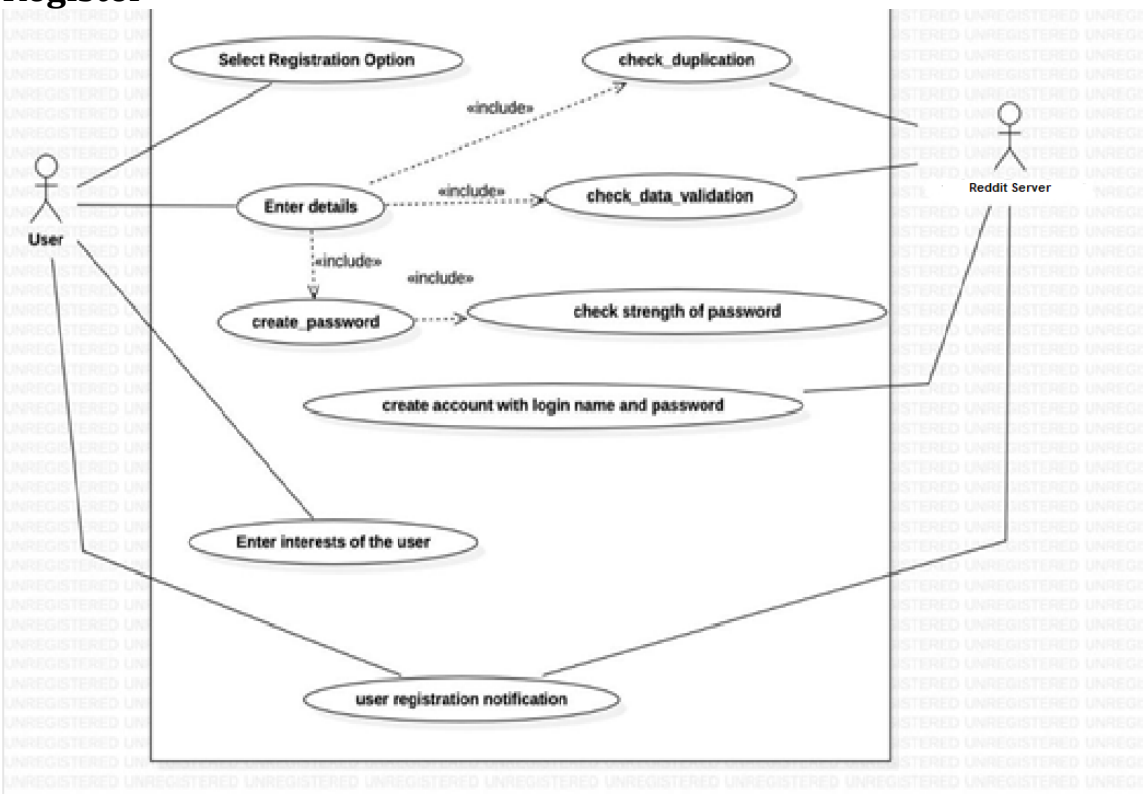
->Each member has an account. There are admins who can block members and unblock members, delete posts.

->Java Swing tutorial is a part of Java Foundation Classes (JFC) that will be used to create window-based applications. It will be built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

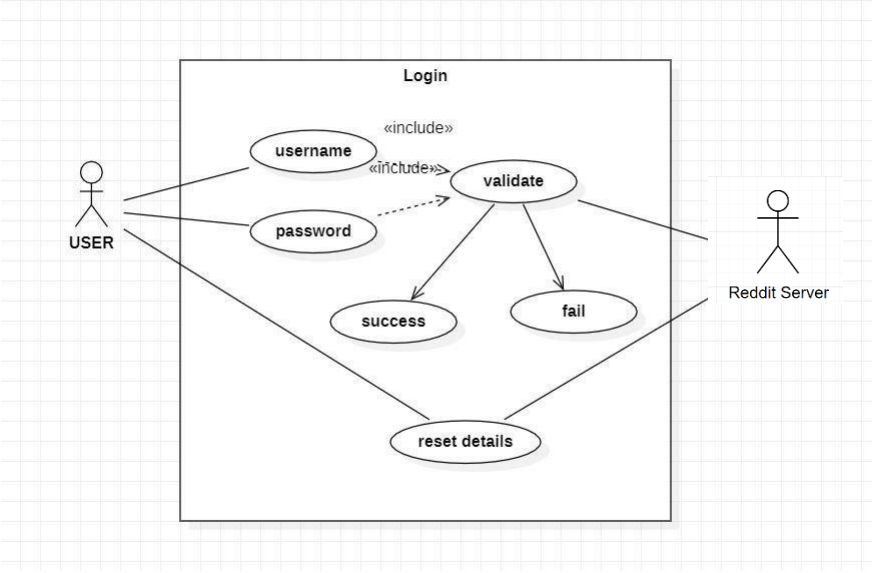
->The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc. Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Use Case Diagram with Description

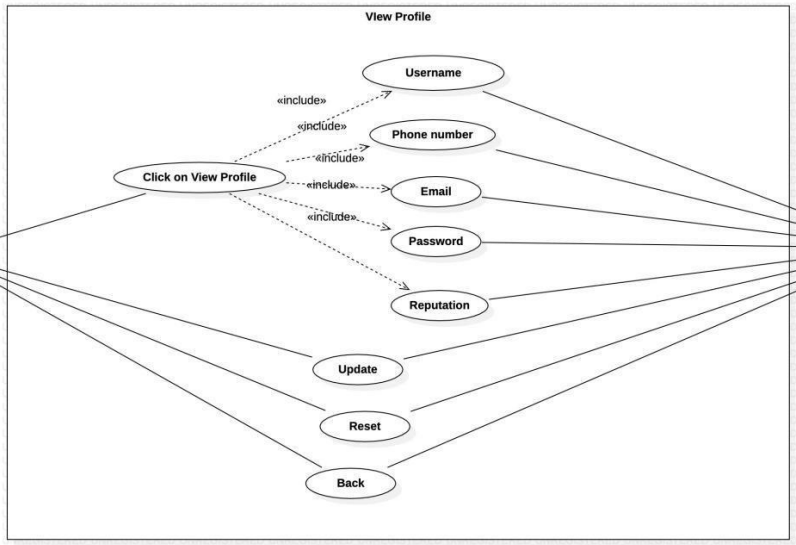
1. Register



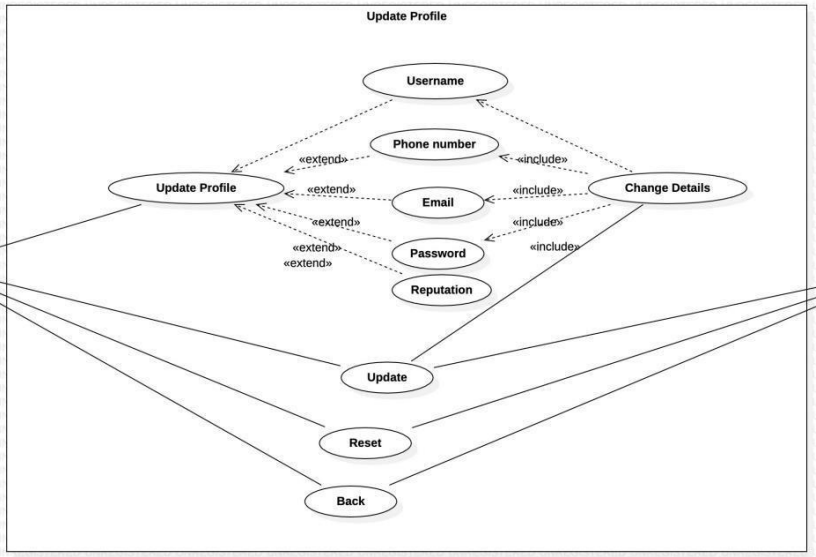
2. Login



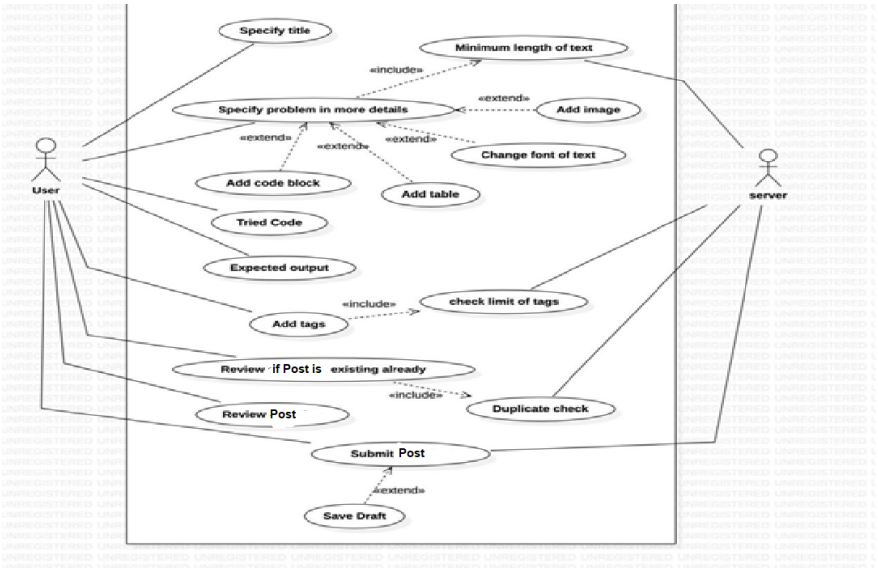
3. View Profile



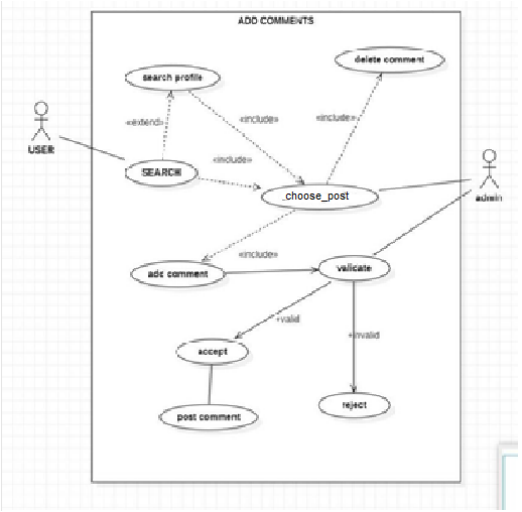
4. Update Credentials



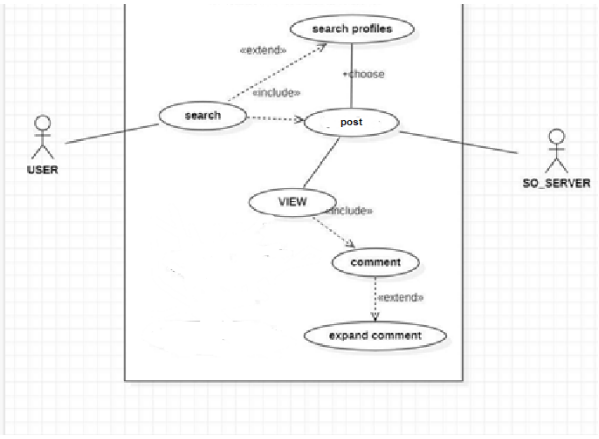
5. Create Posts



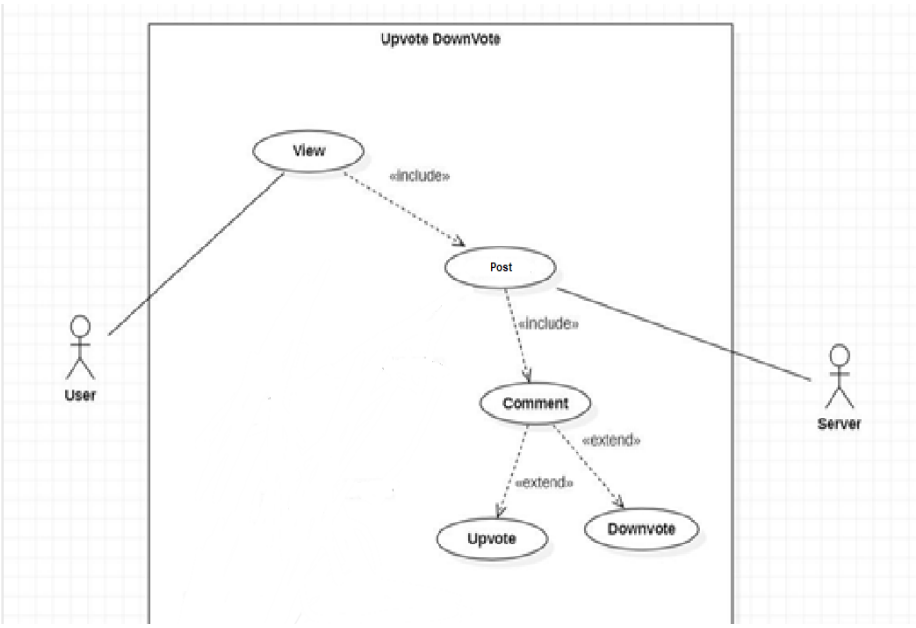
6. Add Comments



7. View Comments



8. Upvote and Downvote of Comments



Use Case Specifications for Important Use Cases:

1. **Use Case Name:** Register

Actors: User

Description: If the user is not registered onto the site then he/she can register onto the site by providing correct details and can create the password for 'logging in' in future. The user will recieve the notification of account creation and it will be registered into the database.

2. **Use Case Name:** Login

Actors: User

Description: User enters the username and password details ,the details are validated based on which it is accepted or rejected. User has option to reset the details and this is updated at the backend.

3. **Use Case Name**: View post

Actors: User

Description: User can view posts by searching for the user profile or searching for posts directly. He can choose newest, most frequent and active posts. User can direct back to homepage.

4. **Use Case Name**: View Profile

Actors: User

Description: User can view Profile by clicking on the profile or member button where you can see username , email, phone number , passwordand reputation score of the user , from this we can also go to update page and direct back to homepage .

5. **Use Case Name**: Update Credentials

Actors: User

Description: User can Update their credentials in the update page , wecan update name , phone number , email, password , and click on updatefor updating the credentials of the details that have been changed , alsowe can reset or direct back to homepage.

6. **Use Case Name**: Ask post

Actors: User

Description: The user can ask the post by specifying the title and detailed summary of post. You can also add tried and expected output code to the box. Make the post more visible by specifying thetags. You can review post or draft it.

7. **Use Case Name**: Add Comments

Actors: user

Description: user is allowed to search for a post and add a comment which is validated by server .if it is invalid it is rejected and if it is valid it is posted under the post and updated in the backend.

8. **Use Case Name:** View Comments

Actors: User

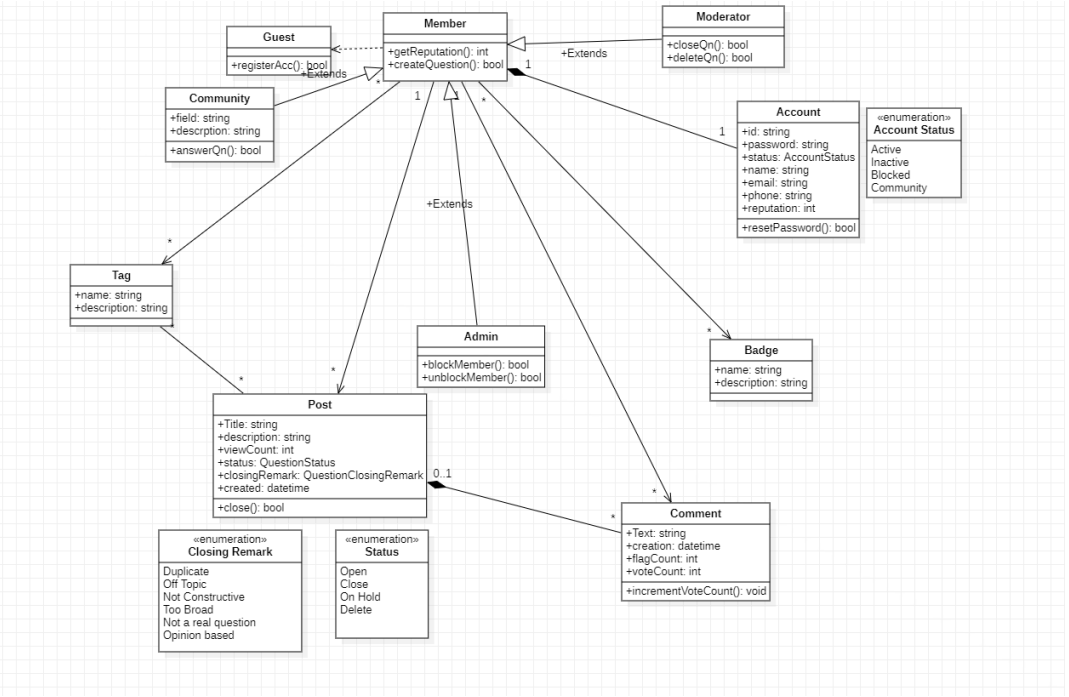
Description: User is allowed to search for a post and view the comment section and can also expand it.

9. **Use Case Name**: Upvote downvote

Actors: User

Description: He can view the posts. User can upvote ordownvote posts and comments made on the post.

- **Class model**


Guest
+registerAcc(): bool
Member
+getReputation(): int
+createQuestion(): bool
Moderator
+closeQn(): bool
+deleteQn(): bool
+Extends
+Extends
Community
+field: string
+descrption: string
+answerQn(): bool
Account
+id: string
+password: string
+status: AccountStatus
+name: string
+email: string
+phone: string
+reputation: int
+resetPassword(): bool
«enumeration»
Account Status
Active
Inactive
Blocked
Community
+Extends
Tag
+name: string
+description: string
Admin
+blockMember(): bool
+unblockMember(): bool
Badge
+name: string
+description: string
Post
+Title: string
+description: string
+viewCount: int
+status: QuestionStatus
+closingRemark: QuestionClosingRemark
+created: datetime
+close(): bool
0..1
Comment
+Text: string
+creation: datetime
+flagCount: int
+voteCount: int
+incrementVoteCount(): void
«enumeration»
Closing Remark
Duplicate
Off Topic
Not Constructive
Too Broad
Not a real question
Opinion based
«enumeration»
Status
Open
Close
On Hold
Delete

• **Architecture Patterns**

1.Model-View-Controller (MVC): This pattern is used to separate the application into three interconnected components, namely the Model, View, and Controller. The Model represents the data and the business logic, the View represents the user interface, and the Controller acts as an intermediary between the Model and the View.
CONTROLLER- conn.java,
MODEL: addcomments.java and etc.
VIEW: individual to every use case.

2.Event-Driven Pattern: Event-Driven Architecture is an agile approach in which services (operations) of the software are triggered byevents.

3.**Microservices Architecture Pattern:** This is a pattern that divides a large application into small, independent services that can be developed,deployed, and scaled independently. Each service is responsible for a specific functionality and communicates with other services over a network.

The design principles used in this project include:

1.Single responsibility Principle:
In the file post.java there are separate classes to instantiate the objectto instantiate comments and separate for answer.

2. Separation of Concerns (SoC): The project separates differentconcerns and functionalities into different classes, allowing for better maintainability and scalability. For example, post, comments and registration classes are different.

3. Dependency Inversion Principle (DIP): The high-level modules/classes in the project depend on abstractions rather thanconcrete implementations.

4. Keep It Simple, Stupid (KISS): The project is designed to besimple and easy to understand, with straightforward code and minimal complexity.

For example, for post view, we created the separate abstraction post.java and for registration, member.java is defined.

The design patterns used are:

• **Singleton pattern:** The Database Handler class implements the Singleton pattern, which restricts the instantiation of a class to a single object.

```
public class conn {
    public Connection c = null;
    public Statement stmt = null;
    private static conn instance;

    private conn(String connectionLink, String user, String pass)
    {
        try
        {
            Class.forName(className:"org.postgresql.Driver");
            c = DriverManager.getConnection(connectionLink, user, pass);
            System.out.println(x:"Connected");
        }
        catch (Exception e)
        {
            e.printStackTrace();
            System.err.println(e.getClass().getName()+": "+e.getMessage());
            System.exit(status:0);
        }
    }

    public static conn getInstance(String connectionLink, String user, String pass){
        if(instance == null){
            instance = new conn(connectionLink, user, pass);
        }
        return instance;
    }
```

• **Facade pattern:** Facade pattern hides the complexities of thesystem and provides an interface to the client using which the client can access the system. This type of design pattern comesunder structural pattern as this pattern adds an interface to existing system to hide its complexities.

```
public class view {
    Run | Debug
    public static void main(String[] args) {
        // showAccountView theView = new showAccountView(member);
        // post post=new post("","");
        String connectionLink = "jdbc:postgresql://127.0.0.1:5432/reddit";
        String user = "postgres";
        String pass = "1234";
        conn c1 = conn.getInstance(connectionLink, user, pass);
        new registrationView(c1);
        // theView.setVisible(true);
    }
}
```

• **Method overloading:** In post.java, it is used.

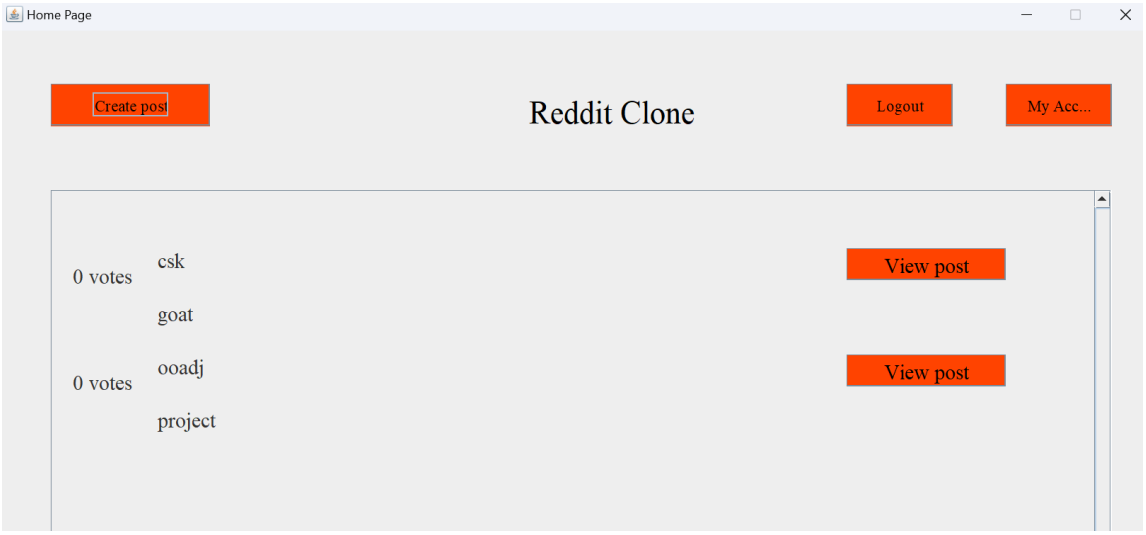
Individual contributions of team members:

SAMARTH RAJENDRA - POSTS FRONT AND BACKIMPLEMENTATION.
SANATH N BHARGAV - COMMENT SECTION FRONT AND BACKIMPLEMENTATION.
SATHWICK P - REGISTRATION FRONT AND BACK IMPLEMENTATION.

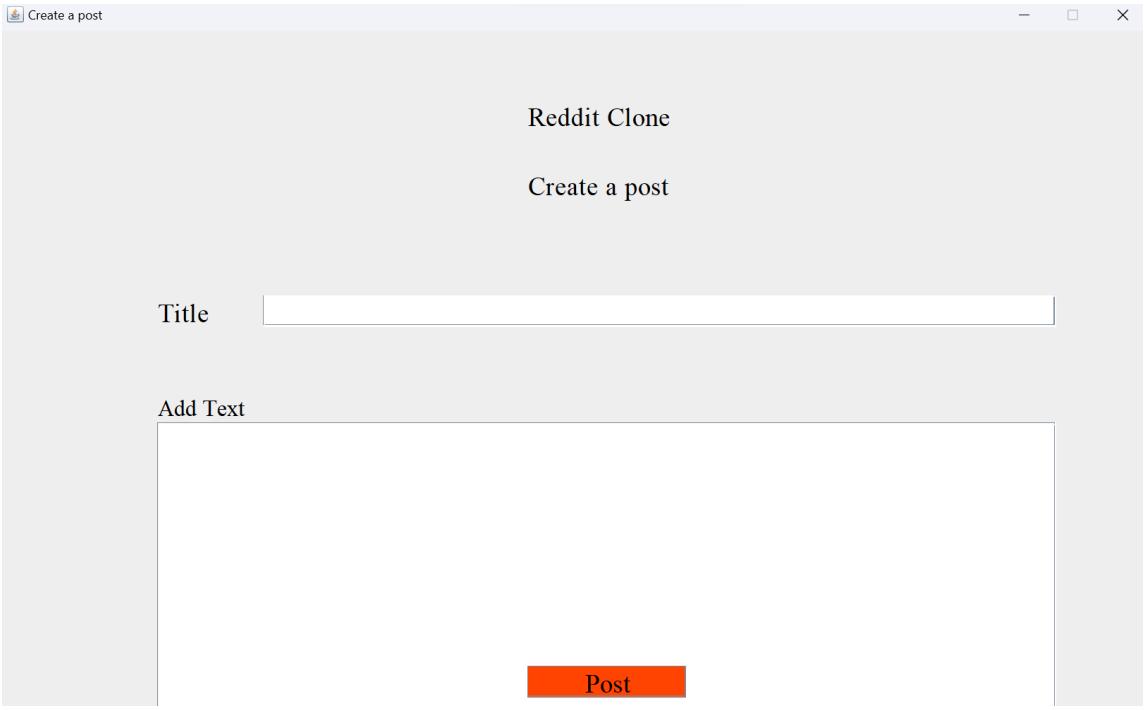
Screenshots of the GUI:

MEMBER INFORMATION:

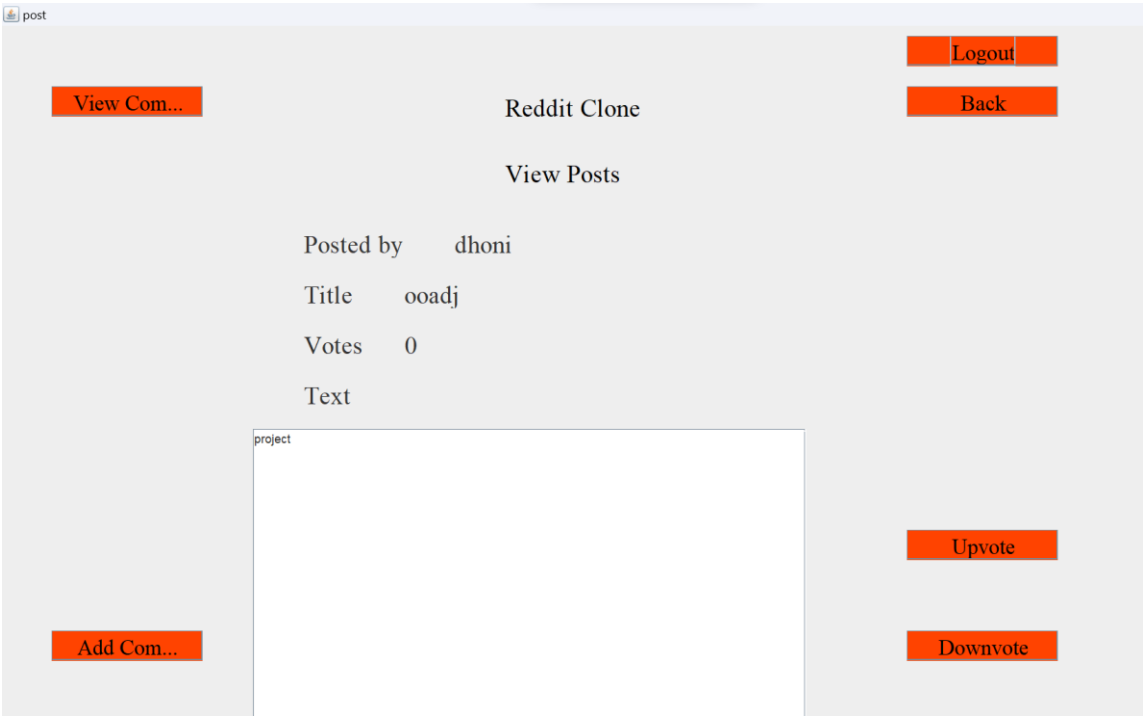
HOME PAGE:



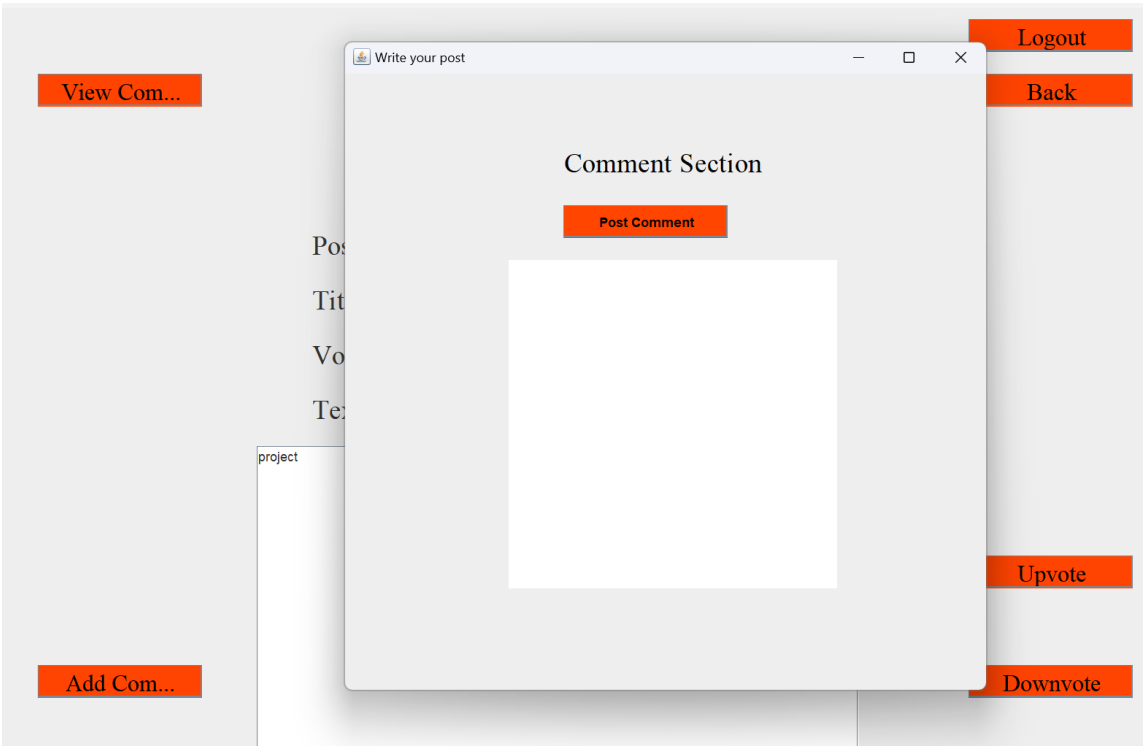
CREATE POST:



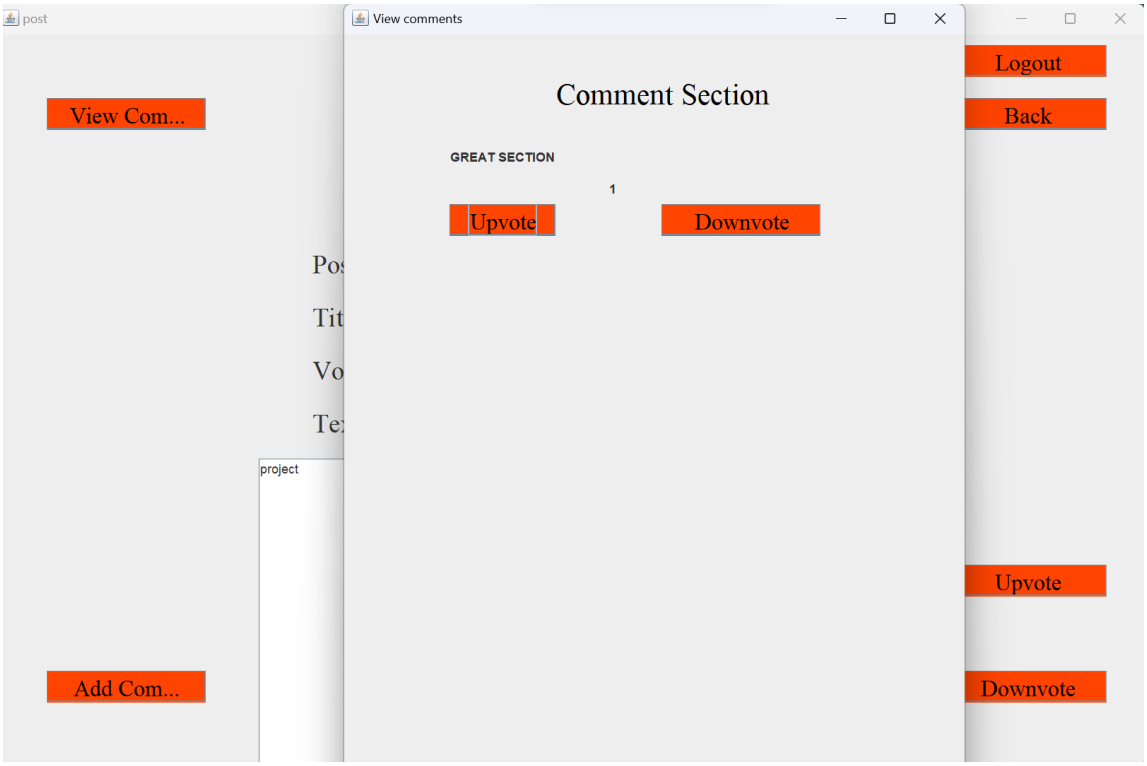
VIEW POST:



WRITE COMMENT:



VIEW COMMENTS:



ACCOUNT INFORMATION: