Project Report

‘MHZ Socio’

Object Oriented Programming Project

Submitted By:

Maqsood Ahmed [38186]

Hamza Hassan [38309]

M. Zeeshan Khan [38049]

Submitted To:

Prof. Syeda Amna Rizwan

Prof. Affefah Qureshi

MHZ Socio: Social Interaction Console-Based Application

Table of Contents:

Abstract

MHZ Socio is a console-based social interaction application designed to provide a basic yet comprehensive simulation of a social media platform. This project allows users to create profiles, manage friendships, send and receive messages, and create and view posts. It aims to deliver a fundamental understanding of how social media platforms operate, offering hands-on experience with user interaction, data management, and console-based application development.

Introduction

Social media platforms have become an integral part of modern communication, connecting people worldwide. MHZ Socio is developed to simulate the core functionalities of a social media platform in a console-based environment. This project aims to help users understand the underlying principles and mechanisms involved in such platforms. The application supports user registration and login, profile management, friend management, messaging, and post creation and viewing.

Methodology

The development of MHZ Socio follows a structured methodology, focusing on modular design and object-oriented programming principles. The application is implemented in Java, leveraging its robust features for handling user input, data management, and interactive functionalities. Key methodologies employed in the project include:

- **Requirement Analysis:** Identifying and defining the core features required for the social interaction application.
- **Design:** Creating a modular design using classes and methods to encapsulate functionalities like user management, friend management, messaging, and post management.
- **Implementation**: Coding the application in Java, ensuring each feature is implemented effectively and efficiently.
- **Testing:** Conducting thorough testing to identify and fix bugs, ensuring the application runs smoothly.
- **Documentation**: Documenting the code and creating a user guide for understanding and operating the application.

Implementation

The implementation of MHZ Socio involves several classes and methods, each responsible for different functionalities. Below is a detailed overview of the key components:

User Class

- Stores user information including username, password, personal details, friend list, posts, and messages.

Post Class

- Represents a post created by a user. Stores the content of the post and the timestamp of creation.

Message Class

- Represents a message sent between users. Stores the content, sender, recipient, and timestamp.

Menu Class

Handles the display and navigation of various menus such as the start menu, profile menu, friend menu, message menu, and home feed.

User Interaction Flow

1. Start Menu

- User registers or logs in.
- On successful login, the user is taken to their profile menu.

2. Profile Menu

- Options to view and edit profile, manage friends, send/view messages, create/view posts, and log out.

3. Friends Management

- Users can add friends, view friend requests, and manage their friend list.

4. Messaging

- Users can send and receive messages from friends.

5. Post Management

- Users can create posts and view posts from their friends in the home feed.

Conclusion

MHZ Socio provides a fundamental yet comprehensive simulation of a social media platform in a console-based environment. It serves as an excellent project for learning and understanding core concepts of user management, friend relationships, messaging systems, and content creation and viewing. The modular structure and object-oriented approach ensure ease of extension and enhancement with additional features, making it a valuable educational tool for aspiring developers.

Source Code:

```
import java.time.ZoneId;   // for taking input from user
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Scanner;

/*
    @author:                Maqsood Ahmed, Hamza Hassan and Zeeshan Khan
    @about Project:   The name of the project is 'Social Interaction System' where a user can register            his/herself and can
make friend, can post, search post, can comment, can like and interact the other user with messaging in private or can make the
group.
*/
interface utilityFunctions {
    // function for validating email
    public boolean validateEmail(String email);
    //validates password
    public boolean validatePassword(String password);
    // shows password requirements
    public void showPasswordRequirements();
    // confirms password
    public boolean confirmPassword(String confirmPassword, String password);
    // validates the phone number and it's length too
    public boolean validatePhoneNo(String phoneNo);
    // checks if DOB's format is correct or not (DDMMYYYY)
    public boolean validateDOBFormat(String dateOfBirth);
    // clears the screen
    public void clearScreen();
    // Project logo
    public void logo();
    // user will have to select one of the options below
    public void showStartMenu();
    // user will go to that selected option
    public void selectStartMenuOption();
    // shows user profile
    public void showProfile(User user);
    // selects option from showProfile function
    public void selectProfileOption();
    // message menu
    public void showMessageMenu();
    // friend handling menu
    public void manageFriendsMenu();
    // creates post of user
    public void createPost();
    // shows posts of his/her self
    public void showYourPosts();
    // shows profile information
    public void showProfileInformation();
     // Function to send a message to friends
    public void sendMessageToFriend();
    // Function to view received messages from friends
    public void viewReceivedMessages();
    // displays all friends
    public void viewFriends();
    // adds the friend
    public void addFriend();
    // home
    public void home();
};
// This is a class of main function where this application will be start
public class Application {
    public static final int MAX_USERS = 100;
    public static User[] users = new User[MAX_USERS];
    public static int userCount = 0;
    public static User[] getUsers() {
        return users;
    }
    public static int getUserCount() {
        return userCount;
    }

    public static void addUser(User user) {
        if (userCount < MAX_USERS) {
            users[userCount++] = user;
        } else {
            System.out.println("Maximum user limit reached.");
        }
    }

    public static User getUserById(String userId) {
        for (int i = 0; i < userCount; i++) {
            if (users[i].getUserId().equals(userId)) {
                return users[i];
            }
        }
        return null; // User not found
    }


    public static boolean checkUserById(User user) {
        for (int i = 0; i < userCount; i++) {
            if (users[i].getUserId().equals(user.getUserId())) {
                return true;
            }
        }
        return false; // User id not found
    }

    public static void signUp() {
        Scanner scanner = new Scanner(System.in); // for taking input from user
        Menu menu = new Menu();   // to use utility functions

        System.out.println("\n----------------------------------------------");
        System.out.println("-----------------| Sign Up |------------------");
        System.out.println("----------------------------------------------\n");

        // User newUser = new User("38186", "Maqsood Ahmed", "m@gmail.com", "Manjan421", "01012004", "Islamabad",
"03252770421", "BSCS"); // Create a new user object
        User newUser = new User(); // Create a new user object
        String confirmPassword;

        System.out.print("\tEnter user name:           ");
        newUser.setName(scanner.nextLine());

        System.out.print("\tEnter user id:             ");
        newUser.setUserId(scanner.next());

        System.out.print("\tEnter email id:            ");
        newUser.setEmail(scanner.next());
        // checks if email ends @gmail.com or not
        while (!menu.validateEmail(newUser.getEmail())) {
            System.out.print("\tInvalid email! Try again:  ");
            newUser.setEmail(scanner.next());
        }

        menu.showPasswordRequirements();
        System.out.print("\tEnter your password:       ");
        newUser.setPassword(scanner.next());
        // it takes input again and again until password satisfy the requirments
        while(!menu.validatePassword(newUser.getPassword())) {
            System.out.print("\tWeak password!:            ");
            newUser.setPassword(scanner.next());
        }

        System.out.print("\tConfirm your password:     ");
        confirmPassword = scanner.next();
        // it takes input again and again until confirmPassword matches the previous one
        while (!menu.confirmPassword(confirmPassword, newUser.getPassword())) {
            menu.showPasswordRequirements();
            System.out.print("\tNot matched, try again:    ");
            confirmPassword = scanner.next();
        }

        // it takes date of birth from user
        System.out.print("\tEnter your DOB:            ");
        newUser.setBirthDate(scanner.next());
        // it takes input again and again until DOB format is valid
        while (!menu.validateDOBFormat(newUser.getBirthDate())) {
            System.out.print("\tFormat is DDMMYYYY:        ");
            newUser.setBirthDate(scanner.next());
        }

        scanner.nextLine(); // Consume leftover newline

        System.out.print("\tEnter your location:       ");
        newUser.setLocation(scanner.nextLine());

        System.out.print("\tEnter your phone No:       ");
        String phoneNo = scanner.next();
        // it takes input again and again until phone number is valid
        while (!menu.validatePhoneNo(phoneNo)) {
            System.out.print("\tInvalid phone number! Try again: ");
            phoneNo = scanner.next();
        }
        newUser.setPhoneNo(phoneNo);

        scanner.nextLine(); // Consume leftover newline

        System.out.print("\tEnter your bio:            ");
        newUser.setBio(scanner.nextLine());

        addUser(newUser);
    }

    // user will login if credentials are already present in database
    public static User login(String userId, String password) {
        for (int i = 0; i < userCount; i++) {
            User userObj = users[i];
            if (userObj.getUserId().equals(userId) && userObj.getPassword().equals(password)) {
                return userObj; // credentials match
            }
        }
        return null; // user not found
    }


    public static void start() {
        Menu menu = new Menu();   // to use utility functions
        menu.showStartMenu();
    }

    // Entry point of this application
    public static void main(String args[]) {
        start();
    }
};

// Date class which will manages all the date and Time
class Date {

    public String getCurrentDateTime() {
        // Get the current date and time in Pakistan time zone
        ZonedDateTime now = ZonedDateTime.now(ZoneId.of("Asia/Karachi"));

        // Define the format for the date and time string
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

        // Format the date and time as a string
        String dateTimeString = now.format(formatter);

        // Return the formatted date and time string
        return dateTimeString;
    }
}


// class for mainting the comments
class Comment {
    private String commentId;
    private String comment;
    private String commentTimeStamp; // shows the time and date of a comment


    // default constructor
    public Comment() {
        this.commentId = "";
        this.comment = "";
        this.commentTimeStamp = "";
    }

    // parameterized constructor to set the comment
    public Comment(String comment, String commentTimeStamp, String commentId) {
        this.comment = comment;
        this.commentTimeStamp = commentTimeStamp;
        this.commentId = commentId;
    }

    // setter function for commenting
    public void setComment(String comment) {
        this.comment = comment;
    }

    //getter function for commenting
    public String getComment() {
        return comment;
    }

    // setter function for comment time stamp
    public void setCommentTimeStamp(String commentTimeStamp) {
        this.commentTimeStamp = commentTimeStamp;
    }

    // getter function to get the comment time stamp
    public String getCommentTimeStamp() {
        return commentTimeStamp;
    }

    public void setCommentId(String commentId) {
        this.commentId = commentId;
    }

    public String getCommentId() {
        return commentId;
    }
};

// class to manage the likes that friends do on posts
class Like {

    private int postLike;   // like
    private String likeTimeStamp; // like time stamp
    private String likeId;

    // default constructor;
    public Like() {
        this.postLike = 0;
    }

    public Like(int postLike, String likeTimeStamp, String likeId) {
        this.postLike = postLike;
        this.likeTimeStamp = likeTimeStamp;
        this.likeId = likeId;
    }

    // sets the post like of the user
    public void setPostLike(int postLike) {
        this.postLike = postLike;
    }

    // gets the post like of the user
    public int getPostLike() {
        return postLike;
    }

    public void setLikeId(String likeId) {
        this.likeId = likeId;
    }
};

// class for maintaining messages
class Message {
    private String messageId;
    private String messageContent;
    private String messageTimeStamp;
    private String senderId;

    public Message() {
        this.messageId = "";
        this.messageContent = "";
        this.messageTimeStamp = "";
        this.senderId = "";
    }

    public Message(String messageId, String messageContent, String messageTimeStamp, String senderId) {
        this.messageId = messageId;
        this.messageContent = messageContent;
        this.messageTimeStamp = messageTimeStamp;
        this.senderId = senderId;
    }

    public void setMessageId(String messageId) {
        this.messageId = messageId;
    }
    public String getMessageId() {
        return messageId;
    }

    public void setMessageContent(String messageContent) {
        this.messageContent = messageContent;
    }

    public String getMessageContent() {
        return messageContent;
    }

    public void setMessageTimeStamp(String messageTimeStamp) {
        this.messageTimeStamp = messageTimeStamp;
    }

    public String getMessageTimeStamp() {
        return messageTimeStamp;
    }

    public void setSenderId(String senderId) {
        this.senderId = senderId;
    }

    public String getSenderId() {
        return senderId;
    }
}

// class for managing the posts
class Post {
    private String postId;              // every post has a unique id
    private String postContent;      // content (the actual post)
    private String postTimeStamp;    // time and date, to show when the user posted

    public static final int MAX_COMMENTS = 100;
    private int commentCount = 0;
    private static Comment[] comments = new Comment[MAX_COMMENTS]; // objects of class Comment

    public static final int MAX_LIKES = 100;
    private int likeCount = 0;
    private static Like[] likes = new Like[MAX_LIKES];          // objects of class Like

    // setter function to set the post id
    public void setPostId(String postId) {
        this.postId = postId;
    }

    // getter function to get the post id
    public String getPostId() {
        return postId;
    }

    // setter function to set the content
    public void setPostContent(String postContent) {
        this.postContent = postContent;
    }

    // getter function to get the content
    public String getPostContent() {
        return postContent;
    }

    // setter function to set the time stamp of a post
    public void setPostTimeStamp(String postTimeStamp) {
        this.postTimeStamp = postTimeStamp;
    }

    // getter function to get the time stamp of a post
    public String getPostTimeStamp() {
        return postTimeStamp;
    }

    // Method to add a comment to the post
    public void addComment(Comment comment) {
        if (commentCount < MAX_COMMENTS) {
            comments[commentCount++] = comment;
        } else {
            System.out.println("Maximum number of comments reached.");
        }
    }

    // Method to add a like to the post
    public void addLike(Like like) {
        if (likeCount < MAX_LIKES) {
            likes[likeCount++] = like;
        } else {
            System.out.println("Maximum number of likes reached.");
        }
    }

    // Method to get the number of comments
    public int getCommentCount() {
        return commentCount;
    }

    // Method to get the number of likes
    public int getLikeCount() {
        return likeCount;
    }

    // Method to get all comments
    public Comment[] getComments() {
        return comments;
    }

    // Method to get all likes
    public Like[] getLikes() {
        return likes;
    }
}

// friend class for managing friendship in Social Application
class Friend {
    private String friendName;
    private String friendId;

    public Friend() {
        this.friendName = "";
        this.friendId = "";
    }

    public Friend(String friendName, String friendId) {
        this.friendName = friendName;
        this.friendId = friendId;
    }

    public void setFriendName(String friendName) {
        this.friendName = friendName;
    }
    public String getFriendName() {
        return friendName;
    }

    public void setFriendId(String friendId) {
        this.friendId = friendId;
    }

    public String getFriendId() {
        return friendId;
    }
}


// class for a user where everything related to user will be handled
class User {
    public String userId;    // a unique id for users
    private String name;     // name for a user
    private String email;    // a unique email for users
    private String password; // users password for login credentials
    private String birthDate;   // birthDate format should be DD/MM/YYYY
    private String location; // address of a user
    private String bio;       // bio for user
    private String phoneNo;    // phone number of a user

    public static final int MAX_POSTS = 100;           // max posts that user can post
    public int postCount = 0;                          // post counter
    public static Post posts[] = new Post[MAX_POSTS];  // max posts a user can do

    public static final int MAX_FRIENDS = 100;         // max 100 user can make friends
    public int friendCount = 0;                        // friends counter
    public Friend[] friends = new Friend[MAX_FRIENDS]; // 100 friend class objects

    public static final int MAX_MESSAGES = 100;
    public int messageCount = 0;                       // message counter
    public Message[] messages = new Message[MAX_MESSAGES];

    // default constructor which sets all the attributes to it's defualt values
    public User() {
        this.userId = "";
        this.name = "";
        this.email = "";
        this.password = "";
        this.birthDate = "";
        this.location = "";
        this.bio = "";
        this.phoneNo = "";
    }

    // parameterized constructor which sets the values
    public User(String userId, String name, String email, String password, String birthDate, String location, String phoneNo, String
bio) {
        this.userId = userId;
        this.name = name;
        this.email = email;
        this.password = password;
        this.birthDate = birthDate;
        this.location = location;
        this.bio = bio;
        this.phoneNo = phoneNo;
    }

    // sets the userId of the user
    public void setUserId(String userId) {
```

```
    this.userId = userId;
}

// gets the user id of the user
public String getUserId() {
    return userId;
}

// sets the name of the user
public void setName(String name) {
    this.name = name;
}

// gets the name of the user
public String getName() {
    return name;
}

// sets the email of the user
public void setEmail(String email) {
    this.email = email;
}

// gets the email of the user
public String getEmail() {
    return email;
}

// sets the password of the user
public void setPassword(String password) {
    this.password = password;
}

// gets the password of the user
public String getPassword() {
    return password;
}

// sets the birth date of the user
public void setBirthDate(String birthDate) {
    this.birthDate = birthDate;
}

// gets the birth date of the user
public String getBirthDate() {
    return birthDate;
}

// sets the location of the user
public void setLocation(String location) {
    this.location = location;
}

// gets the location of the user
public String getLocation() {
    return location;
}

public void setBio(String bio) {
    this.bio = bio;
}

public String getBio() {
    return bio;
}
```

```
    public void setPhoneNo(String phoneNo) {
        this.phoneNo = phoneNo;
    }

    public String getPhoneNo() {
        return phoneNo;
    }

    public void addFriend(Friend friend) {

        if(friendCount < MAX_FRIENDS) {
            friends[friendCount++] = friend;
            System.out.println("Friend has been Added in your friendlist!");
        } else {
            System.out.println("Maximum number of friends reached.");
        }
    }

    public boolean checkFriendById(String friendId) {
        for (int i = 0; i < friendCount; i++) {
            if (friends[i].getFriendId().equals(friendId)) {
                return true; // Friend found
            }
        }
        return false; // Friend not found
    }
};


// Menu class where all the utility functions will be implemented
class Menu implements utilityFunctions {

    Application application = new Application(); // to use signup and login functions
    Scanner scanner = new Scanner(System.in); // for taking input
    User loggedInUser = new User(); // user object for passing it in signup function
    Date date = new Date(); // it will be responsible for managing date and time

    private int choice;

    // to pass userId, password in login function
    private String userID;
    private String password;

    @Override // clears the screen
    public void clearScreen() {
        // Simplified clear screen logic
        System.out.print("\033[H\033[2J");
        System.out.flush();
    }

    @Override // shows password requirements
    public void showPasswordRequirements() {
        // pre-requirements for password
        System.out.println("\n---------------------------------------------");
        System.out.println("| Password must be 8 characters long         |");
        System.out.println("| Password must have one uppercase character |");
        System.out.println("| Password must have one lowercase character |");
        System.out.println("---------------------------------------------\n");
    }

    // Function to validate the password based on specific criteria
    public boolean validatePassword(String password) {

        boolean hasUpper = false, hasLower = false, hasDigit = false;
    if (password.length() < 8) {
        return false;
    }

    for (char c : password.toCharArray()) {
        if (Character.isUpperCase(c))
            hasUpper = true;

        if (Character.isLowerCase(c))
            hasLower = true;

        if (Character.isDigit(c))
            hasDigit = true;
    }

    // return true if all the conditions were true
    return hasUpper && hasLower && hasDigit;
}

@Override  // Project logo
public void logo() {
    System.out.println("\n----------------------------------------------");
    System.out.println("----------------------------------------------");
    System.out.println("\t   __  __  _  _  ____\n\t  |  \V  || || ||_  /\n\t  | |\V| || __ | / / \n\t  |_|  |_||_||_|/___|");
    System.out.println("\t   ___          _      \n\t  / __| ___  __ (_) ___ \n\t  \\__ \V _ \V _|| |/ _ \\\n\t  |___/\\___/\\__||_|\\___/");
}

// function for validating email
@Override
public boolean validateEmail(String email) {
    return email.endsWith("@gmail.com"); // checks if email ends '@gmail.com' or not
}

// confirms password
@Override
public boolean confirmPassword(String confirmPassword, String password) {
    return password.equals(confirmPassword); // checks if password or same or not
}

// checks if DOB's format is correct or not (DD/MM/YYYY)
@Override
public boolean validateDOBFormat(String dateOfBirth) {
    // Check if the dateOfBirth has exactly 8 digits
    if (dateOfBirth.length() != 8) {
        return false;
    }

    // Extract day, month, and year from dateOfBirth
    int day = Integer.parseInt(dateOfBirth.substring(0, 2));
    int month = Integer.parseInt(dateOfBirth.substring(2, 4));
    int year = Integer.parseInt(dateOfBirth.substring(4, 8));

    // Validate the year range
    if (year < 1924 || year > 2024) {
        return false;
    }

    // Validate the month range
    if (month < 1 || month > 12) {
        return false;
    }

    // Validate the day range
    if (day < 1 || day > 31) {
        return false;
    }

    return true;
}

// validates the phone number and it's length too
public boolean validatePhoneNo(String phoneNo) {
    if(phoneNo.length() < 11) {
        return false;
    }

    for(char c: phoneNo.toCharArray()) {
        if(!Character.isDigit(c)) {
            return false;
        }
    }

    return true;
}

@Override  // user will have to select one of the options below
public void showStartMenu() {
    logo(); // shows project logo
    System.out.println("\n----------------------------------------------");
    System.out.println("-----------------| START MENU |---------------");
    System.out.println("----------------------------------------------\n");
    System.out.println("\tPress 0 for exit!");
    System.out.println("\tPress 1 for Sign Up");
    System.out.println("\tPress 2 for Login");
    System.out.print("\n\tEnter your choice >> ");
    selectStartMenuOption();
}


@Override //
public void selectStartMenuOption() {
    choice = scanner.nextInt(); // taking input from user he/she must select valid option

    if(choice >= 0 && choice <= 2) {

        switch (choice) {

            case 0:
                System.out.println("\n\t[ Thank you and Have a Nice day <3 ]");
                return;

            case 1:
                application.signUp();
                // clearScreen(); // clears the screen
                logo(); // shows the logo
                showProfile(application.users[application.getUserCount() -1]);
                break;

            case 2:
                System.out.println("\n----------------------------------------------");
                System.out.println("-----------------| Login |--------------------");
                System.out.println("----------------------------------------------\n");
                System.out.print("\tEnter user id:   ");
                userID = scanner.next();
                System.out.print("\tEnter password:  ");
                password = scanner.next();

                loggedInUser = application.login(userID, password);
                if(loggedInUser != null) {
```

```
                logo(); // shows the logo
                showProfile(loggedInUser);
            }
            else {
                System.out.println("\n\t[ Credentials not matched! ]");
                showStartMenu(); // shows start menu
            }
            break;

        default:
            break;
    }
  } else {
      System.out.println("\t[ Invalid Option! Try again <3 ]");
      showStartMenu(); // shows start menu
  }
}


@Override // shows user profile
public void showProfile(User user) {
    this.loggedInUser = user; // Assign the logged-in user to the class variable
    String greet = "Welcome " + loggedInUser.getName().toUpperCase() + "!";
    System.out.println("\n----------------------------------------------");
    System.out.println("-----------------| PROFILE |------------------");
    System.out.println("----------------------------------------------\n");
    System.out.println("\tWelcome " + loggedInUser.getName().toUpperCase() + "!\n");
    System.out.println("\tPress 0 for Logout!");
    System.out.println("\tPress 1 for Home");
    System.out.println("\tPress 2 for create Post");
    System.out.println("\tPress 3 for show Message Menu");
    System.out.println("\tPress 4 for show Friends Menu");
    System.out.println("\tPress 5 for show Profile");
    System.out.print("\n\tEnter your choice >> ");
    selectProfileOption(); // user will select one from above options
}

@Override
public void selectProfileOption() {
    choice = scanner.nextInt(); // takes input from user

    switch (choice) {
        case 0:
            showStartMenu(); // redirect to start menu
            break;
        case 1:
            home();
            break;
        case 2:
            createPost();
            break;
        case 3:
            showMessageMenu();
            break;
        case 4:
            manageFriendsMenu();
            break;
        case 5:
            showProfileInformation();
            break;
        default:
            System.out.println("\t[ Invalid Option :( Try again <3 ]");
            showProfile(loggedInUser); // shows profile menu again
            break;
    }
}

@Override
public void home() {
    clearScreen();
    logo();
    System.out.println("\n---------------------------------------------");
    System.out.println("----------------| HOME PAGE |-----------------");
    System.out.println("---------------------------------------------\n");

    // Iterate through the friends of the logged-in user
    for (int i = 0; i < loggedInUser.friendCount; i++) {
        Friend friend = loggedInUser.friends[i];
        User friendUser = application.getUserById(friend.getFriendId());

        if (friendUser != null) {
            // Display friend's name
            System.out.println("\n----------------[ " + friendUser.getName() + "'s Posts ]----------------");

            // Iterate through the posts of the friend
            for (int j = friendUser.postCount - 1; j >= 0; j--) {
                Post post = friendUser.posts[j];
                System.out.println("\n---------------------------------------------");
                System.out.println(friendUser.getName() + " Time: [" + post.getPostTimeStamp() + "]");
                System.out.println("---------------------------------------------");
                System.out.println(post.getPostContent());
            }
        } else {
            System.out.println("\n---------------------------------------------");
            System.out.println("\tNothing to show :(");
            System.out.println("---------------------------------------------");
        }
    }

    System.out.println("\n---------------------------------------------");
    System.out.println("-------------| End of Home Page |-------------");
    System.out.println("---------------------------------------------\n");

    showProfile(loggedInUser);
}

@Override
public void createPost() {
    System.out.println("\n---------------------------------------------");
    System.out.println("-----------| Creating a post... |-------------");
    System.out.println("---------------------------------------------");
    System.out.println("\n------------[ Enter # for exit ]-------------");

    StringBuilder postContentBuilder = new StringBuilder();
    String postContent;

    do {
        System.out.print("\t>> ");
        postContent = scanner.nextLine();
        postContentBuilder.append(postContent).append("\n");
    } while (!postContent.equals("#"));

    postContent = postContentBuilder.toString().trim(); // Get the full post content

    // Ensure loggedInUser is not null
    if (loggedInUser != null) {
        // Ensure loggedInUser's posts array is not null
        if (loggedInUser.posts != null) {
            // Ensure there's space to add a new post
            if (loggedInUser.postCount < loggedInUser.posts.length) {
                // Create a new Post object
                Post newPost = new Post();
                newPost.setPostContent(postContent);
                newPost.setPostId(loggedInUser.getUserId());
                newPost.setPostTimeStamp(date.getCurrentDateTime());

                // Add the new post to the loggedInUser's posts array
                loggedInUser.posts[loggedInUser.postCount++] = newPost;

                System.out.println("\n----------------------------------------------");
                System.out.println("---------| Post created successfully! |-------");
                System.out.println("----------------------------------------------");
            } else {
                System.out.println("\n----------------------------------------------");
                System.out.println("------| Maximum number of posts reached! |----");
                System.out.println("----------------------------------------------");
            }
        } else {
            System.out.println("\n----------------------------------------------");
            System.out.println("----------| User's posts array is null! |-----");
            System.out.println("----------------------------------------------");
        }
    } else {
        System.out.println("\n----------------------------------------------");
        System.out.println("-----------| Logged in user is null! |--------");
        System.out.println("----------------------------------------------");
    }

    showProfile(loggedInUser); // back to profile menu
}

@Override // shows posts of his/her self
public void showYourPosts() {
    System.out.println("\n----------------------------------------------");
    System.out.println("-----------| Displaying Posts... |------------");
    System.out.println("----------------------------------------------\n");
    // displays in chronological order
    for(int postIdx = loggedInUser.postCount - 1; postIdx >= 0; --postIdx) {
        System.out.println("\n----------------------------------------------");
        System.out.println(loggedInUser.getName() + " Time: [" + loggedInUser.posts[postIdx].getPostTimeStamp() + "]");
        System.out.println("\n----------------------------------------------");
        System.out.println(loggedInUser.posts[postIdx].getPostContent());
        System.out.println("----------------------------------------------");
    }
}

@Override
public void showMessageMenu() {
    clearScreen();
    logo();
    System.out.println("\n----------------------------------------------");
    System.out.println("----------------| MESSAGE MENU |--------------");
    System.out.println("----------------------------------------------\n");
    System.out.println("\tPress 0 for Back to Profile");
    System.out.println("\tPress 1 for Send Message");
    System.out.println("\tPress 2 for View Messages");
    System.out.print("\tEnter your choice >> ");
    int choice = scanner.nextInt();

    switch (choice) {
        case 0:
            showProfile(loggedInUser);
            break;
        case 1:
            sendMessageToFriend();
            break;
        case 2:
            viewReceivedMessages();
            break;
        default:
            System.out.println("\t[ Invalid Option! Try again <3 ]");
            showMessageMenu();
            break;
    }
}

@Override // Function to send a message to friends
public void sendMessageToFriend() {
    clearScreen();
    logo();
    System.out.println("\n---------------------------------------------");
    System.out.println("----------| SEND MESSAGE TO FRIEND |---------");
    System.out.println("---------------------------------------------\n");

    // Display the list of friends
    System.out.println("\tYour Friends:");
    for (int i = 0; i < loggedInUser.friendCount; i++) {
        Friend friend = loggedInUser.friends[i];
        System.out.println("\t" + (i + 1) + ". " + friend.getFriendName() + " (" + friend.getFriendId() + ")");
    }

    // Ask user to select a friend
    System.out.print("\n\tEnter the number of the friend you want to message (0 to cancel): ");
    int friendChoice = scanner.nextInt();

    if (friendChoice == 0) {
        showMessageMenu(); // Go back to message menu
        return;
    }

    if (friendChoice < 1 || friendChoice > loggedInUser.friendCount) {
        System.out.println("\t[ Invalid choice! ]");
        sendMessageToFriend(); // Retry
        return;
    }

    // Get the selected friend
    Friend selectedFriend = loggedInUser.friends[friendChoice - 1];

    // Find the friend user object
    User friendUser = Application.getUserById(selectedFriend.getFriendId());
    if (friendUser != null) {
        scanner.nextLine(); // Consume newline
        System.out.print("\tEnter your message: ");
        String messageContent = scanner.nextLine();

        // Create a new message object
        Message message = new Message();
        message.setMessageId("MSG" + (friendUser.messageCount + 1)); // Generate a unique message ID
        message.setMessageContent(messageContent);
        message.setMessageTimeStamp(date.getCurrentDateTime());
        message.setSenderId(loggedInUser.getUserId()); // Set the sender ID

        // Add the message to the friend's message list
        friendUser.messages[friendUser.messageCount++] = message;

        System.out.println("\n\tMessage sent successfully!");
    } else {
        System.out.println("\n\t[ Friend not found.]");
    }

    // Wait for user input to continue
    System.out.print("\n\tPress Enter to continue...");
    scanner.nextLine();
    scanner.nextLine();
    showMessageMenu();
}


@Override // Function to view received messages from friends
public void viewReceivedMessages() {
    clearScreen();
    logo();
    System.out.println("\n----------------------------------------------");
    System.out.println("------| VIEW RECEIVED MESSAGES FROM FRIENDS |------");
    System.out.println("----------------------------------------------\n");

    // Check if there are any messages
    if (loggedInUser.messageCount == 0) {
        System.out.println("\tYou have no received messages.");
        System.out.println("\n\tPress Enter to go back...");
        scanner.nextLine();
        scanner.nextLine();
        showMessageMenu();
        return;
    }

    // Display received messages
    System.out.println("Received Messages:");
    for (int i = 0; i < loggedInUser.messageCount; i++) {
        Message message = loggedInUser.messages[i];
        System.out.println("\n\tMessage ID: " + message.getMessageId());
        System.out.println("\tFrom: " + message.getSenderId());
        System.out.println("\tContent: " + message.getMessageContent());
        System.out.println("\tTimestamp: " + message.getMessageTimeStamp());
    }

    // Wait for user input to continue
    System.out.println("\nPress Enter to go back...");
    scanner.nextLine();
    scanner.nextLine();
    showMessageMenu();
}

@Override
public void manageFriendsMenu() {
    clearScreen();
    logo();
    System.out.println("\n----------------------------------------------");
    System.out.println("---------------| FRIENDS MANAGEMENT |---------");
    System.out.println("----------------------------------------------\n");
    System.out.println("\tPress 0 for Back to Profile");
    System.out.println("\tPress 1 for Add Friend");
    System.out.println("\tPress 2 for View Friends");
    System.out.print("\n\tEnter your choice >> ");
    int choice = scanner.nextInt();

    switch (choice) {
        case 0:
            showProfile(loggedInUser);
            break;
        case 1:
            addFriend();
            break;
        case 2:
            viewFriends();
            break;
        default:
            System.out.println("\t[ Invalid Option! Try again <3 ]");
            manageFriendsMenu();
            break;
    }
}

@Override
public void showProfileInformation() {
    clearScreen();
    logo();
    System.out.println("\n----------------------------------------------");
    System.out.println("-----------| PROFILE INFORMATION |------------");
    System.out.println("----------------------------------------------\n");
    System.out.println("\tName: " + loggedInUser.getName());
    System.out.println("\tEmail: " + loggedInUser.getEmail());
    System.out.println("\tDate of Birth: " + loggedInUser.getBirthDate());
    System.out.println("\tLocation: " + loggedInUser.getLocation());
    System.out.println("\tPhone Number: " + loggedInUser.getPhoneNo());
    System.out.println("\tBio: " + loggedInUser.getBio());
    // displays all the posts
    showYourPosts();

    System.out.println("\n\tPress 0 to go back to Profile");
    System.out.print("\tEnter your choice >> ");
    int choice = scanner.nextInt();

    if (choice == 0) {
        showProfile(loggedInUser);
    } else {
        System.out.println("\t[ Invalid Option! Try again <3 ]");
        showProfileInformation();
    }
}


@Override
public void addFriend() {
    clearScreen();
    logo();
    System.out.println("\n----------------------------------------------");
    System.out.println("--------------| ADD FRIEND |------------------");
    System.out.println("----------------------------------------------\n");

    // Check if the user has reached the maximum number of friends
    if (loggedInUser.friendCount == User.MAX_FRIENDS) {
        System.out.println("\tYou have reached the maximum number of friends.");
        System.out.println("\n\tPress Enter to go back...");
        scanner.nextLine();
        scanner.nextLine();
        manageFriendsMenu();
        return;
    }

    // Prompt the user to enter the friend's ID
    System.out.print("\tEnter the ID of the friend you want to add: ");
    String friendId = scanner.next();

    // Check if the friend ID is valid
    User friendUser = application.getUserById(friendId);
    if (friendUser == null) {
        System.out.println("\tUser with ID " + friendId + " not found.");
```

```
            System.out.println("\n\tPress Enter to go back...");
            scanner.nextLine();
            scanner.nextLine();
            manageFriendsMenu();
            return;
        }

        // Check if the user is already friends with the specified friend
        if (loggedInUser.checkFriendById(friendUser.getUserId())) {
            System.out.println("\tYou are already friends with " + friendUser.getName() + ".");
            System.out.println("\n\tPress Enter to go back...");
            scanner.nextLine();
            scanner.nextLine();
            manageFriendsMenu();
            return;
        }

        // Add the friend
        Friend newFriend = new Friend(friendUser.getName(), friendUser.getUserId());
        loggedInUser.addFriend(newFriend);
        System.out.println("\tYou are now friends with " + friendUser.getName() + ".");

        // Wait for user input to continue
        System.out.println("\n\tPress Enter to go back...");
        scanner.nextLine();
        scanner.nextLine();
        manageFriendsMenu();
    }

    @Override
    public void viewFriends() {
        clearScreen();
        logo();
        System.out.println("\n----------------------------------------------");
        System.out.println("---------------| FRIENDS LIST |---------------");
        System.out.println("----------------------------------------------\n");

        // Check if the user has any friends
        if (loggedInUser.friendCount == 0) {
            System.out.println("\tYou have no friends :(");
            System.out.println("\n\tPress Enter to go back...");
            scanner.nextLine();
            scanner.nextLine();
            manageFriendsMenu();
            return;
        }

        // Display the list of friends
        System.out.println("\tYour Friends:");
        for (int i = 0; i < loggedInUser.friendCount; i++) {
            Friend friend = loggedInUser.friends[i];
            System.out.println("\t" + (i + 1) + ". " + friend.getFriendName() + " (ID: " + friend.getFriendId() + ")");
        }

        // Wait for user input to continue
        System.out.println("\n\tPress Enter to go back...");
        scanner.nextLine();
        scanner.nextLine();
        manageFriendsMenu();
    }
}
```

The End