package TVShowLibrary;

import java.util.ArrayList;

import java.util.Scanner;

import java.io.\*;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import javax.swing.SwingUtilities;

class User {

    public String Username;

    private String Password;

    public User(String Username, String Password) {

        this.Username = Username;

        this.Password = Password;

    }

    String GetPassword() {

        return Password;

    }

}

 public class Show {

    String name;

    double rating;

    int ratingCount;

    public Show(String name, double rating, int ratingCount) {

        this.name = name;

        this.rating = rating;

        this.ratingCount = ratingCount;

    }

    public void addRating(int newRating) {

        rating = ((rating \* ratingCount) + newRating) / (ratingCount + 1);

        ratingCount++;

    }

    public String getStars() {

        int stars = (int)Math.round(rating);

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < stars; i++) sb.append("★");

        for (int i = stars; i < 5; i++) sb.append("☆");

        return sb.toString();

    }

    public String getCategory() {

        if (rating >= 4.5) return "Amazing";

        else if (rating >= 3.5) return "Good";

        else if (rating >= 2.5) return "Average";

        else if (rating >= 1.5) return "Poor";

        else return "Bad";

    }

    @Override

    public String toString() {

        return name + " | " + getStars() + " (" + String.format("%.2f", rating) + " from " + ratingCount + " ratings) | " + getCategory();

    }

}

public class LoginPage  {

    ArrayList<User> Profiles = new ArrayList<>();

    public int noofUsers = 0;

    Scanner sc = new Scanner(System.in);

    private final String FILE\_NAME = "users.txt";

    public LoginPage() {

        loadUsersFromFile();

    }

    public void CreateAccount() {

        String setUsername;

        String setPassword;

        System.out.println("Enter Username:");

        setUsername = sc.nextLine();

        System.out.println("Enter Password:");

        setPassword = sc.nextLine();

        Profiles.add(new User(setUsername, setPassword));

        System.out.println("Account Created Successfully!");

        noofUsers++;

        saveUsersToFile();

    }

    public User Login() { // Return User

        String setUsername;

        String setPassword;

        String TempUsercheck;

        String TempPassCheck;

        if (noofUsers < 1) { // Check case for no Profiles

            System.out.println("No Profiles Avalible...\nRedirecting to Account Creation...");

            System.out.println("Enter Username:");

            setUsername = sc.nextLine();

            System.out.println("Enter Password:");

            setPassword = sc.nextLine();

            User newUser = new User(setUsername, setPassword);

            Profiles.add(newUser);

            System.out.println("Account Created Successfully!");

            noofUsers++;

            saveUsersToFile();

            return newUser;

        }

        System.out.println("Enter Username:");

        setUsername = sc.nextLine();

        System.out.println("Enter Password:");

        setPassword = sc.nextLine();

        for (User user : Profiles) {

            TempUsercheck = user.Username;

            TempPassCheck = user.GetPassword(); // Private Password Access

            if (TempUsercheck.equals(setUsername) && TempPassCheck.equals(setPassword)) {

                System.out.println("Login Successful! Welcome " + setUsername);

                return user; // Return the logged-in user

            }

        }

        System.out.println("Login Failed! Invalid Username or Password.");

        return null;

    }

    // Save all users to file

   public void saveUsersToFile() {

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(FILE\_NAME))) {

            for (User user : Profiles) {

                bw.write(user.Username + "," + user.GetPassword());

                bw.newLine();

            }

        } catch (IOException e) {

            System.out.println("Error saving users: " + e.getMessage());

        }

    }

    // Load all users from file

    private void loadUsersFromFile() {

        File file = new File(FILE\_NAME);

        if (!file.exists()) return;

        try (BufferedReader br = new BufferedReader(new FileReader(FILE\_NAME))) {

            String line;

            while ((line = br.readLine()) != null) {

                String[] parts = line.split(",", 2);

                if (parts.length == 2) {

                    Profiles.add(new User(parts[0], parts[1]));

                    noofUsers++;

                }

            }

        } catch (IOException e) {

            System.out.println("Error loading users: " + e.getMessage());

        }

    }

}

public class LoginGUI extends JFrame {

    private JTextField usernameField;

    private JPasswordField passwordField;

    private LoginPage loginPage;

    public static Boolean signin;

    public LoginGUI() {

        loginPage = new LoginPage();

        setTitle("Login - TV Show Library");

        setSize(350, 200);

        setDefaultCloseOperation(EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        setLayout(new GridLayout(4, 2, 5, 5));

        JLabel userLabel = new JLabel("Username:");

        JLabel passLabel = new JLabel("Password:");

        usernameField = new JTextField();

        passwordField = new JPasswordField();

        JButton loginButton = new JButton("Login");

        JButton signupButton = new JButton("Create Account");

        add(userLabel);

        add(usernameField);

        add(passLabel);

        add(passwordField);

        add(loginButton);

        add(signupButton);

        // Login button action

        loginButton.addActionListener(e -> handleLogin());

        // Signup button action

        signupButton.addActionListener(e -> handleSignup());

        setVisible(true);

    }

    private void handleLogin() {

        String username = usernameField.getText().trim();

        String password = new String(passwordField.getPassword());

        for (User user : loginPage.Profiles) {

            if (user.Username.equals(username) && user.GetPassword().equals(password)) {

                JOptionPane.showMessageDialog(this, "Login successful! Welcome " + username);

                Runner.Signed\_in\_User = user;

                this.dispose();

                new TvShowGUI();  // Launch TV show GUI

               signin=true;

                return;

            }

        }

        JOptionPane.showMessageDialog(this, "Invalid username or password", "Login Failed", JOptionPane.ERROR\_MESSAGE);

    }

    private void handleSignup() {

        String username = usernameField.getText().trim();

        String password = new String(passwordField.getPassword());

        if (username.isEmpty() || password.isEmpty()) {

            JOptionPane.showMessageDialog(this, "Username and password cannot be empty.");

            return;

        }

        // Check if username exists

        for (User user : loginPage.Profiles) {

            if (user.Username.equalsIgnoreCase(username)) {

                JOptionPane.showMessageDialog(this, "Username already exists.", "Signup Failed", JOptionPane.ERROR\_MESSAGE);

                return;

            }

        }

        // Create new account

        User newUser = new User(username, password);

        loginPage.Profiles.add(newUser);

        loginPage.noofUsers++;

        JOptionPane.showMessageDialog(this, "Account created successfully! You can now log in.");

        loginPage.saveUsersToFile();

    }

    public static Boolean CheckSignin(){

    if(signin=true){

        return true;

    }

    return false;

    }

    public void closeGUI(){

      setDefaultCloseOperation(EXIT\_ON\_CLOSE);

    }

}

class TVShowLibrary {

    private static final String FILE\_NAME = "shows.txt";

    ArrayList<Show> shows = new ArrayList<>();

    public TVShowLibrary() {

        loadShowsFromFile();

    }

    private String getValidInput(Scanner sc, String prompt) {

        String input;

        do {

            System.out.println(prompt);

            input = sc.nextLine().trim();

            if (input.isEmpty()) {

                System.out.println("Input cannot be empty. Please try again.");

            }

        } while (input.isEmpty());

        return input;

    }

    public void AddShow() {

        Scanner sc = new Scanner(System.in);

        String showName = getValidInput(sc, "Enter the name of the show to add:");

        for (Show s : shows) {

            if (s.name.equalsIgnoreCase(showName)) {

                System.out.println("Show already exists.");

                return;

            }

        }

        shows.add(new Show(showName, 0, 0));

        saveShowsToFile();

        System.out.println("Show '" + showName + "' added to the library.");

    }

    public void RemoveShow() {

        Scanner sc = new Scanner(System.in);

        String showName = getValidInput(sc, "Enter the name of the show to remove:");

        Show found = null;

        for (Show s : shows) {

            if (s.name.equalsIgnoreCase(showName)) {

                found = s;

                break;

            }

        }

        if (found == null) {

            System.out.println("Show '" + showName + "' not found in the library.");

            return;

        }

        shows.remove(found);

        saveShowsToFile();

        System.out.println("Show '" + showName + "' removed from the library.");

    }

    public void DisplayShowsByCategory() {

        if (shows.isEmpty()) {

            System.out.println("No shows available in the library.");

            return;

        }

        Map<String, List<Show>> categories = new LinkedHashMap<>();

        categories.put("Amazing", new ArrayList<>());

        categories.put("Good", new ArrayList<>());

        categories.put("Average", new ArrayList<>());

        categories.put("Poor", new ArrayList<>());

        categories.put("Bad", new ArrayList<>());

        for (Show show : shows) {

            categories.get(show.getCategory()).add(show);

        }

        for (String category : categories.keySet()) {

            System.out.println("\n" + category + " Shows:");

            if (categories.get(category).isEmpty()) {

                System.out.println("  None");

            } else {

                for (Show show : categories.get(category)) {

                    System.out.println("  " + show.name + " | " + show.getStars() + " (" + String.format("%.2f", show.rating) + " from " + show.ratingCount + " ratings)");

                }

            }

        }

    }

    public void SearchShow() {

        Scanner sc = new Scanner(System.in);

        String showName = getValidInput(sc, "Enter the name of the show to search:");

        for (Show s : shows) {

            if (s.name.equalsIgnoreCase(showName)) {

                System.out.println("Show found: " + s);

                return;

            }

        }

        System.out.println("Show '" + showName + "' not found in the library.");

    }

    public void SortShows() {

        shows.sort(Comparator.comparingDouble((Show s) -> -s.rating));

        System.out.println("Shows sorted by rating (highest first):");

        for (Show show : shows) {

            System.out.println("  " + show);

        }

    }

    public void RateShow() {

        Scanner sc = new Scanner(System.in);

        String showName = getValidInput(sc, "Enter the name of the show to rate:");

        Show found = null;

        for (Show s : shows) {

            if (s.name.equalsIgnoreCase(showName)) {

                found = s;

                break;

            }

        }

        if (found == null) {

            System.out.println("Show not found.");

            return;

        }

        System.out.print("Enter rating (1-5): ");

        if (!sc.hasNextInt()) {

            System.out.println("Invalid input! Please enter a number between 1 and 5.");

            sc.next();

            return;

        }

        int rating = sc.nextInt();

        if (rating < 1 || rating > 5) {

            System.out.println("Invalid rating! Must be between 1 and 5.");

            return;

        }

        found.addRating(rating);

        saveShowsToFile();

        System.out.println("Show '" + showName + "' rated with " + rating + " stars.");

    }

    public void saveShowsToFile() {

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(FILE\_NAME))) {

            for (Show show : shows) {

                bw.write(show.name + "," + show.rating + "," + show.ratingCount);

                bw.newLine();

            }

        } catch (IOException e) {

            System.out.println("Error saving shows: " + e.getMessage());

        }

    }

    private void loadShowsFromFile() {

        File file = new File(FILE\_NAME);

        if (!file.exists()) return;

        try (BufferedReader br = new BufferedReader(new FileReader(FILE\_NAME))) {

            String line;

            while ((line = br.readLine()) != null) {

                String[] parts = line.split(",", 3);

                if (parts.length == 3) {

                    shows.add(new Show(parts[0], Double.parseDouble(parts[1]), Integer.parseInt(parts[2])));

                }

            }

        } catch (IOException e) {

            System.out.println("Error loading shows: " + e.getMessage());

        }

    }

}

public class TvShowGUI {

    private final TVShowLibrary library;

    public TvShowGUI() {

        library = new TVShowLibrary();

        createGUI();

    }

    private void createGUI() {

        JFrame frame = new JFrame("TV Show Library");

        frame.setSize(500, 400);

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setLayout(new GridLayout(6, 1, 10, 10));

        frame.setLocationRelativeTo(null);

        JButton addBtn = new JButton("Add Show");

        JButton removeBtn = new JButton("Remove Show");

        JButton displayBtn = new JButton("Display All Shows");

        JButton searchBtn = new JButton("Search Show");

        JButton rateBtn = new JButton("Rate Show");

        JButton exitBtn = new JButton("Exit");

        addBtn.addActionListener(this::handleAdd);

        removeBtn.addActionListener(this::handleRemove);

        displayBtn.addActionListener(this::handleDisplay);

        searchBtn.addActionListener(this::handleSearch);

        rateBtn.addActionListener(this::handleRate);

        exitBtn.addActionListener(e -> System.exit(0));

        frame.add(addBtn);

        frame.add(removeBtn);

        frame.add(displayBtn);

        frame.add(searchBtn);

        frame.add(rateBtn);

        frame.add(exitBtn);

        frame.setVisible(true);

    }

    private void handleAdd(ActionEvent e) {

        String name = JOptionPane.showInputDialog("Enter show name:");

        if (name == null || name.trim().isEmpty()) {

            showMessage("Invalid input.");

            return;

        }

        name = name.trim();

        for (Show s : library.shows) {

            if (s.name.equalsIgnoreCase(name)) {

                showMessage("Show already exists.");

                return;

            }

        }

        library.shows.add(new Show(name, 0, 0));

        showMessage("Show added.");

    }

    private void handleRemove(ActionEvent e) {

        String name = JOptionPane.showInputDialog("Enter show name to remove:");

        if (name == null || name.trim().isEmpty()) {

            showMessage("Invalid input.");

            return;

        }

        Show found = null;

        for (Show s : library.shows) {

            if (s.name.equalsIgnoreCase(name.trim())) {

                found = s;

                break;

            }

        }

        if (found != null) {

            library.shows.remove(found);

            library.saveShowsToFile();

            showMessage("Show removed.");

        } else {

            showMessage("Show not found.");

        }

    }

    private void handleDisplay(ActionEvent e) {

        if (library.shows.isEmpty()) {

            showMessage("No shows in library.");

            return;

        }

        StringBuilder sb = new StringBuilder();

        for (Show s : library.shows) {

            sb.append(s.toString()).append("\n");

        }

        showLargeText("All Shows", sb.toString());

    }

    private void handleSearch(ActionEvent e) {

        String name = JOptionPane.showInputDialog("Enter show name to search:");

        if (name == null || name.trim().isEmpty()) {

            showMessage("Invalid input.");

            return;

        }

        for (Show s : library.shows) {

            if (s.name.equalsIgnoreCase(name.trim())) {

                showMessage("Found:\n" + s);

                return;

            }

        }

        showMessage("Show not found.");

    }

    private void handleRate(ActionEvent e) {

        String name = JOptionPane.showInputDialog("Enter show name to rate:");

        if (name == null || name.trim().isEmpty()) {

            showMessage("Invalid input.");

            return;

        }

        Show found = null;

        for (Show s : library.shows) {

            if (s.name.equalsIgnoreCase(name.trim())) {

                found = s;

                break;

            }

        }

        if (found == null) {

            showMessage("Show not found.");

            return;

        }

        String ratingStr = JOptionPane.showInputDialog("Enter rating (1-5):");

        try {

            int rating = Integer.parseInt(ratingStr);

            if (rating < 1 || rating > 5) throw new NumberFormatException();

            found.addRating(rating);

            library.saveShowsToFile();

            showMessage("Rating added!");

        } catch (NumberFormatException ex) {

            showMessage("Invalid rating.");

        }

    }

    private void showMessage(String msg) {

        JOptionPane.showMessageDialog(null, msg);

    }

    private void showLargeText(String title, String msg) {

        JTextArea area = new JTextArea(15, 40);

        area.setText(msg);

        area.setEditable(false);

        JScrollPane scroll = new JScrollPane(area);

        JOptionPane.showMessageDialog(null, scroll, title, JOptionPane.INFORMATION\_MESSAGE);

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(TvShowGUI::new);

    }

}

public class Runner {

    public static User Signed\_in\_User = null;

    public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> {

            LoginGUI loginGUI = new LoginGUI();

            loginGUI.show(); // Display login window

        });

    }

}