import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class MovieRecommenderAWT extends Frame implements ActionListener {

// Panels for different views

private Panel loginPanel;

private Panel recommendationPanel;

// Login components

private Label usernameLabel;

private Label passwordLabel;

private TextField usernameField;

private TextField passwordField;

private Button loginButton;

private Label loginMessageLabel;

// Recommendation components

private Label userIdLabel;

private TextField userIdField;

private Label genreLabel;

private Choice genreChoice;

private Button recommendButton;

private Button addToWatchHistoryButton;

private Button viewWatchHistoryButton;

private TextArea resultArea;

private Button logoutButton;

// Define a simple Movie class

static class Movie {

String title;

String genre;

public Movie(String title, String genre) {

this.title = title;

this.genre = genre;

}

}

// Predefined user accounts (username -> password)

private Map<String, String> users = new HashMap<>();

private String loggedInUser = null; // To track the currently logged-in user

// Map to store watch history for each user

private Map<String, List<String>> watchHistory = new HashMap<>();

// List of movies

List<Movie> movies = List.of(

new Movie("Kathi", "Action"),

new Movie("Sura", "Action"),

new Movie("Amaran", "Drama"),

new Movie("Kanguva", "Sci-Fi"),

new Movie("Inception", "Sci-Fi"),

new Movie("Interstellar", "Sci-Fi"),

new Movie("Titanic", "Romance"),

new Movie("Parasite", "Drama"),

new Movie("The Shawshank Redemption", "Drama")

);

// Constructor for GUI setup

public MovieRecommenderAWT() {

setTitle("Movie Recommender System");

setSize(600, 500);

setLayout(new CardLayout());

// Initialize predefined users

users.put("user1", "password1");

users.put("user2", "password2");

// Initialize watch history map

users.keySet().forEach(user -> watchHistory.put(user, new ArrayList<>()));

// Initialize login panel

loginPanel = new Panel(new GridLayout(4, 2, 10, 10));

usernameLabel = new Label("Username:");

usernameField = new TextField(20);

passwordLabel = new Label("Password:");

passwordField = new TextField(20);

passwordField.setEchoChar('\*');

loginButton = new Button("Login");

loginMessageLabel = new Label("", Label.CENTER);

loginButton.addActionListener(this);

loginPanel.add(usernameLabel);

loginPanel.add(usernameField);

loginPanel.add(passwordLabel);

loginPanel.add(passwordField);

loginPanel.add(new Label()); // Spacer

loginPanel.add(loginButton);

loginPanel.add(new Label()); // Spacer

loginPanel.add(loginMessageLabel);

// Initialize recommendation panel

recommendationPanel = new Panel(new FlowLayout());

userIdLabel = new Label("Enter User ID:");

userIdField = new TextField(20);

genreLabel = new Label("Select Genre:");

genreChoice = new Choice();

genreChoice.add("Action");

genreChoice.add("Sci-Fi");

genreChoice.add("Drama");

genreChoice.add("Romance");

recommendButton = new Button("Recommend");

addToWatchHistoryButton = new Button("Add to Watch History");

viewWatchHistoryButton = new Button("View Watch History");

resultArea = new TextArea(15, 50);

resultArea.setEditable(false);

logoutButton = new Button("Logout");

recommendButton.addActionListener(this);

addToWatchHistoryButton.addActionListener(this);

viewWatchHistoryButton.addActionListener(this);

logoutButton.addActionListener(e -> logout());

recommendationPanel.add(userIdLabel);

recommendationPanel.add(userIdField);

recommendationPanel.add(genreLabel);

recommendationPanel.add(genreChoice);

recommendationPanel.add(recommendButton);

recommendationPanel.add(addToWatchHistoryButton);

recommendationPanel.add(viewWatchHistoryButton);

recommendationPanel.add(resultArea);

recommendationPanel.add(logoutButton);

// Add panels to the frame

add(loginPanel, "Login");

add(recommendationPanel, "Recommendation");

// Set window close operation

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

dispose();

}

});

// Show login panel by default

switchToPanel("Login");

}

// Handle button actions

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == loginButton) {

// Handle login

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

String password = passwordField.getText().trim();

if (users.containsKey(username) && users.get(username).equals(password)) {

loggedInUser = username;

loginMessageLabel.setText("");

switchToPanel("Recommendation");

} else {

loginMessageLabel.setText("Invalid username or password.");

}

} else if (e.getSource() == recommendButton) {

// Handle movie recommendation

String userId = userIdField.getText().trim();

// Check if User ID matches the logged-in user

if (userId.isEmpty() || !userId.equals(loggedInUser)) {

resultArea.setText("Invalid User ID. Please enter the correct User ID.");

return;

}

String selectedGenre = genreChoice.getSelectedItem();

List<String> recommendations = getRecommendations(selectedGenre);

// Display recommendations

resultArea.setText("User ID: " + userId + "\n");

resultArea.append("Recommended Movies (" + selectedGenre + "):\n");

for (String movie : recommendations) {

resultArea.append("- " + movie + "\n");

}

} else if (e.getSource() == addToWatchHistoryButton) {

// Add the first movie from recommendations to watch history

String selectedGenre = genreChoice.getSelectedItem();

List<String> recommendations = getRecommendations(selectedGenre);

if (!recommendations.isEmpty()) {

String movieToAdd = recommendations.get(0); // Add the first recommended movie

watchHistory.get(loggedInUser).add(movieToAdd);

resultArea.setText("Movie added to Watch History: " + movieToAdd);

} else {

resultArea.setText("No movies available to add.");

}

} else if (e.getSource() == viewWatchHistoryButton) {

// Display the user's watch history

List<String> history = watchHistory.get(loggedInUser);

resultArea.setText("Watch History for " + loggedInUser + ":\n");

for (String movie : history) {

resultArea.append("- " + movie + "\n");

}

}

}

// Method to get movie recommendations based on genre

private List<String> getRecommendations(String genre) {

List<String> recommendations = new ArrayList<>();

for (Movie movie : movies) {

if (movie.genre.equalsIgnoreCase(genre)) {

recommendations.add(movie.title);

}

}

return recommendations;

}

// Method to log out

private void logout() {

loggedInUser = null;

usernameField.setText("");

passwordField.setText("");

userIdField.setText("");

resultArea.setText("");

switchToPanel("Login");

}

// Helper method to switch panels

private void switchToPanel(String panelName) {

CardLayout layout = (CardLayout) getLayout();

layout.show(this, panelName);

}

// Main method to run the application

public static void main(String[] args) {

MovieRecommenderAWT app = new MovieRecommenderAWT();

app.setVisible(true);

}

}