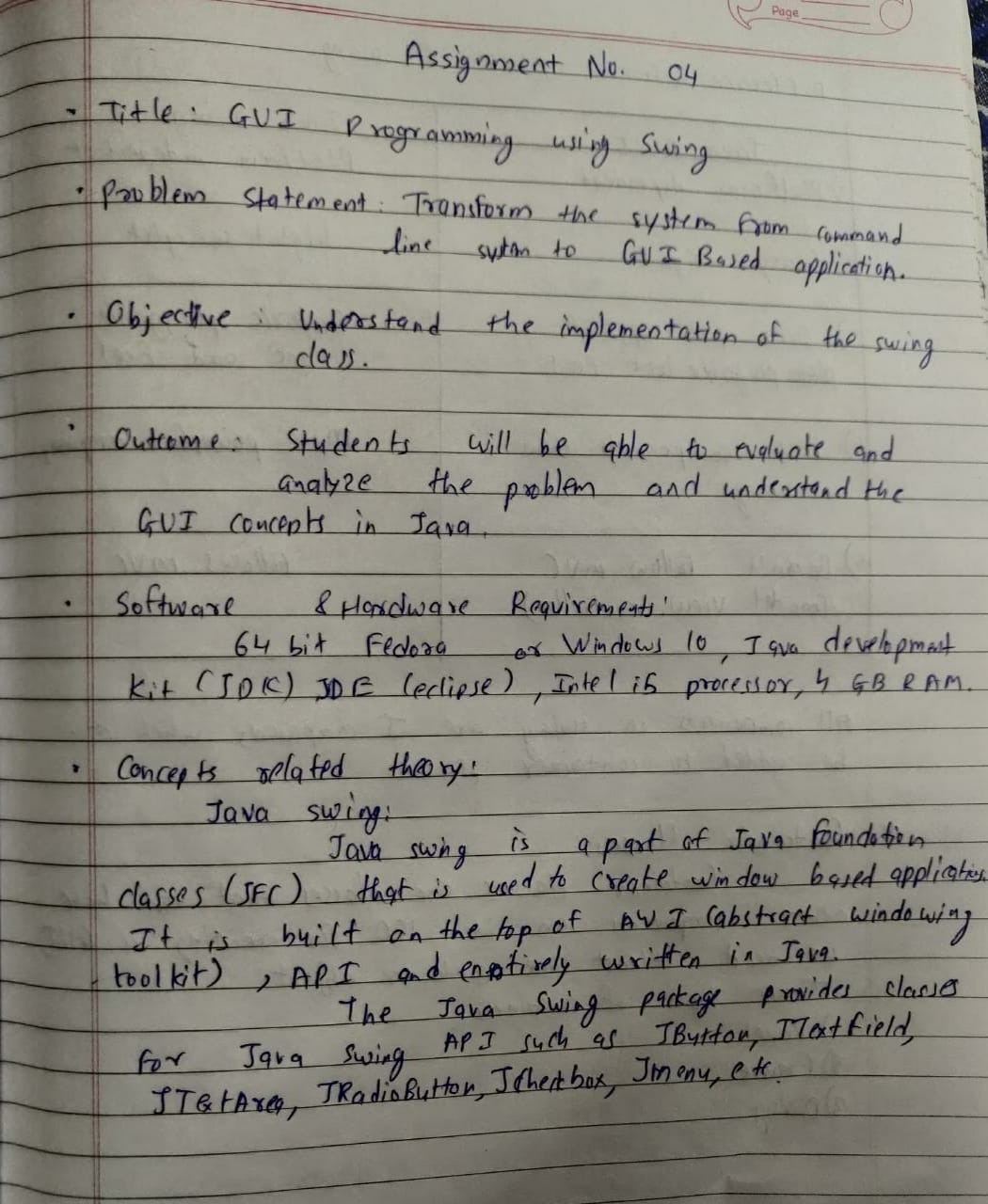
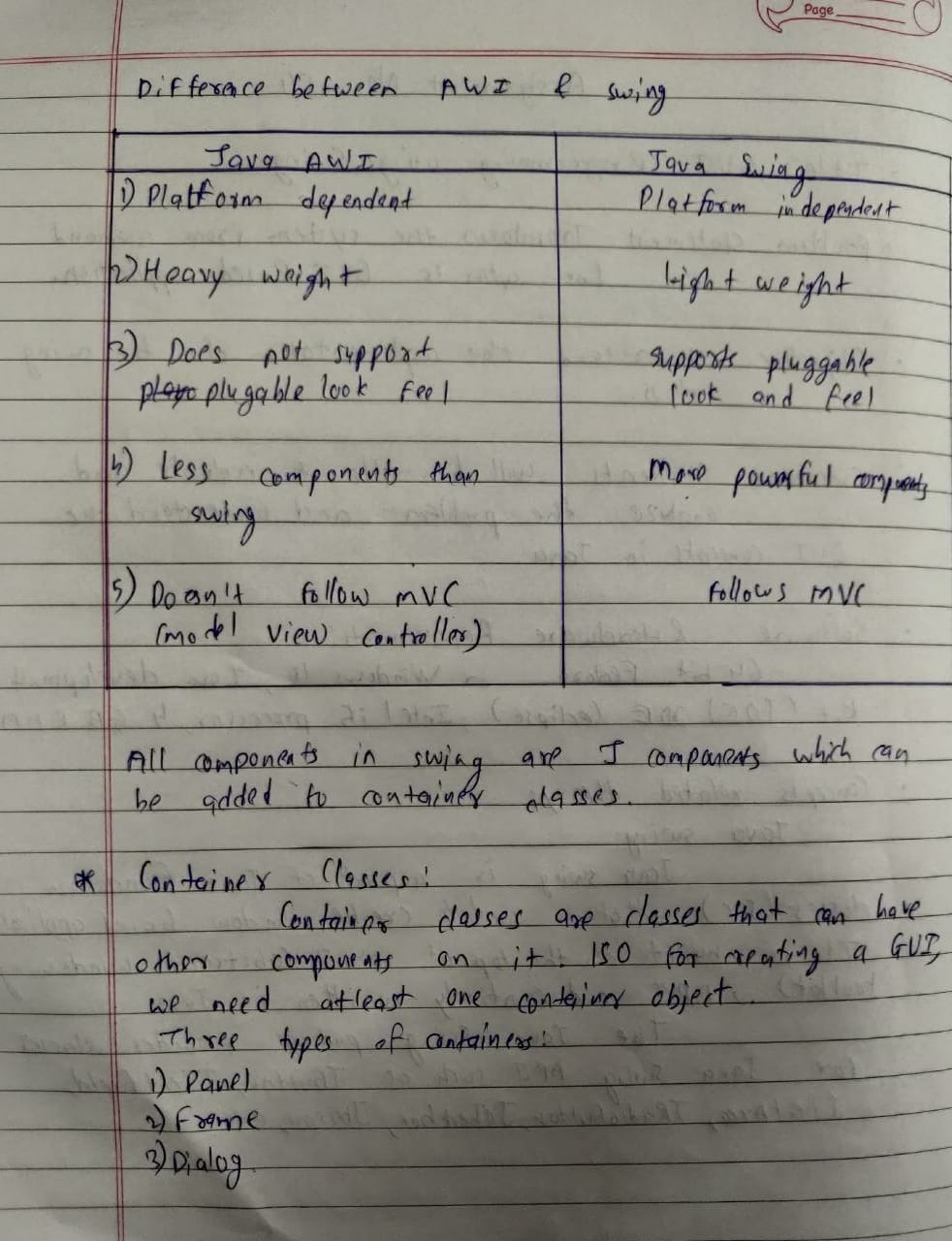
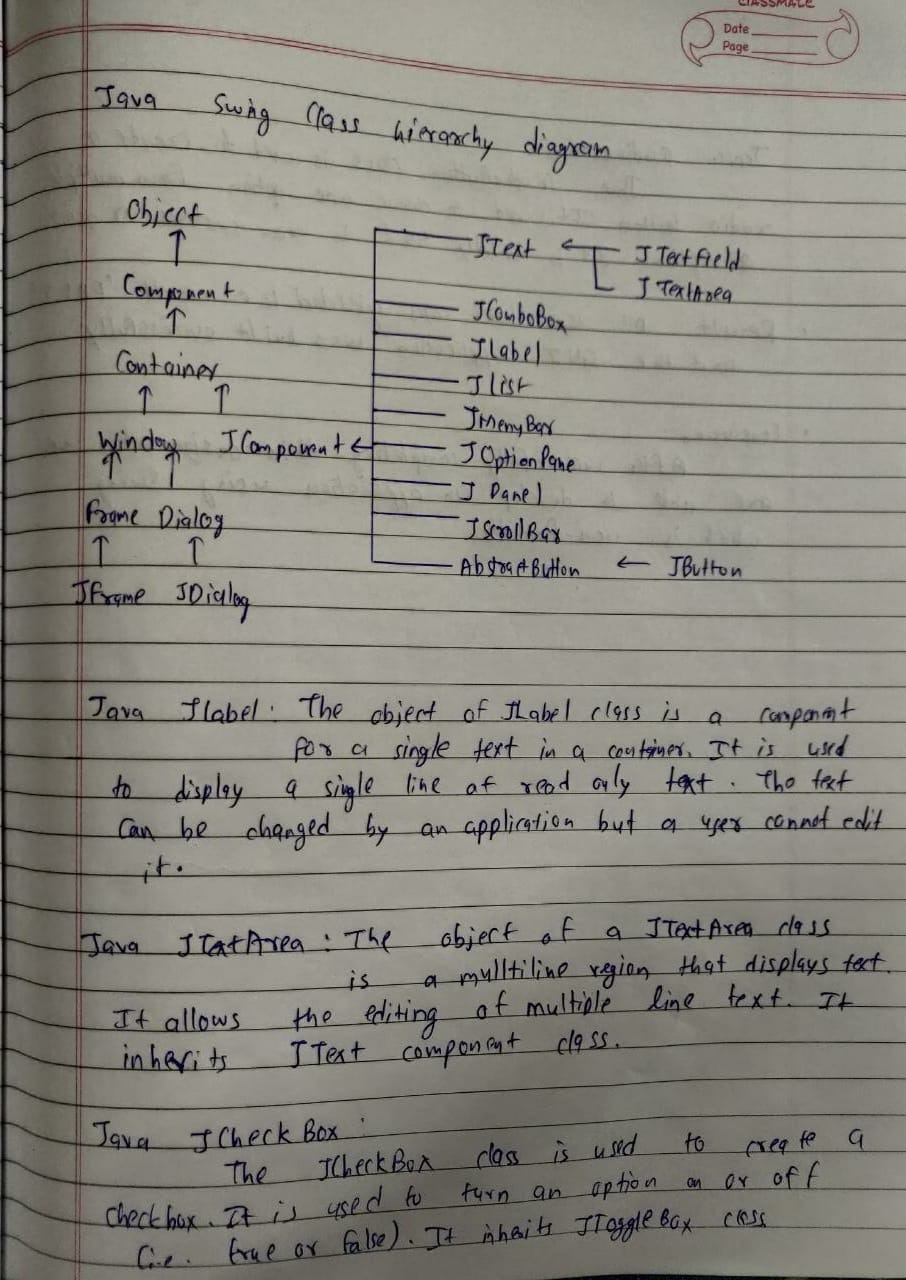
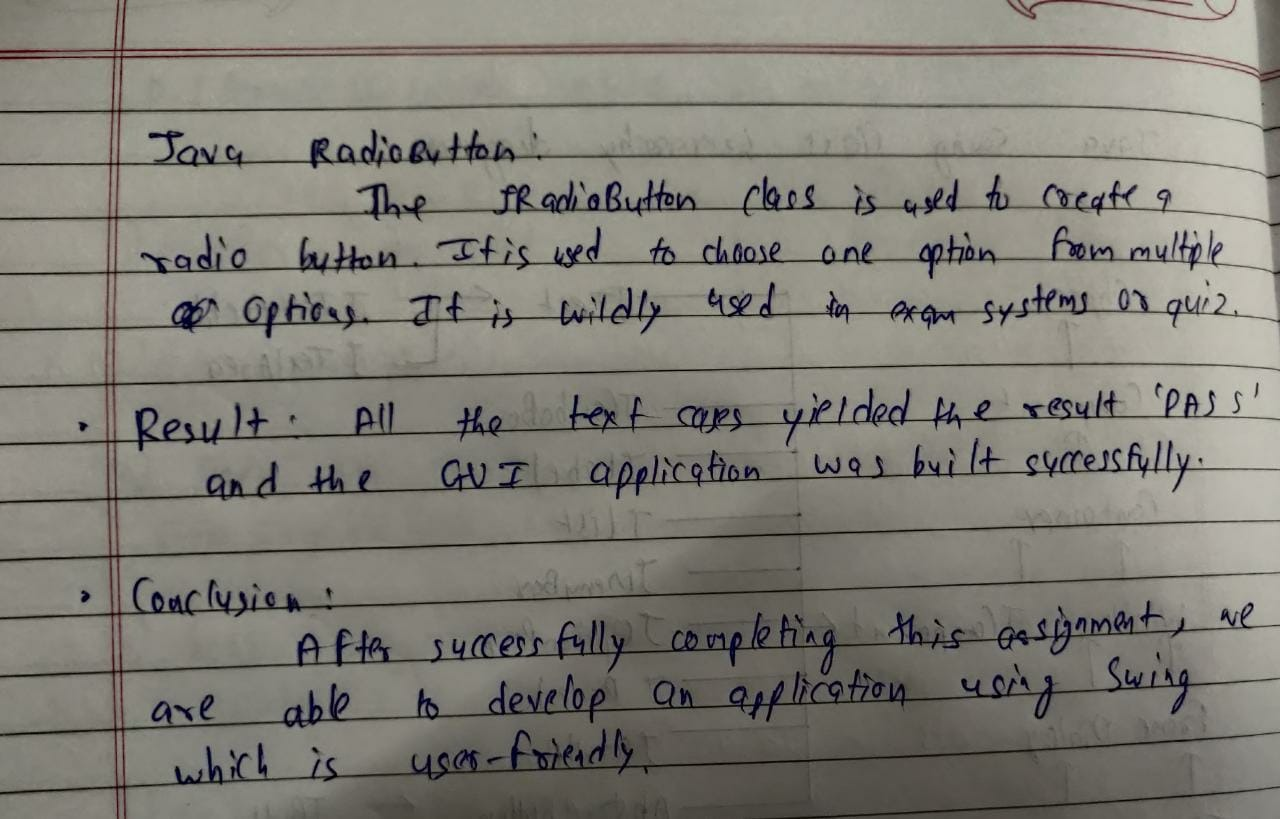
*Write-up*









*Code*

Server.java

package movierec;  
  
import java.io.IOException;  
import java.net.ServerSocket;  
import java.net.Socket;  
import java.sql.\*;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;  
  
public class Server {  
 private static final int PORT = 9169;  
 private static final ExecutorService pool = Executors.newFixedThreadPool(5);  
  
 public static void main(String[] args) throws IOException, SQLException, ClassNotFoundException {  
 String name, pass, url;  
 Connection con;  
 Class.forName("com.mysql.cj.jdbc.Driver");  
 url = "jdbc:mysql://localhost:3306/moviedb";  
 name = "root";  
 pass = "mysql";  
 con = DriverManager.getConnection(url, name, pass);  
 ServerSocket listener = new ServerSocket(PORT);  
 while(true){  
  
 System.out.println("Waiting For Client");  
 Socket client = listener.accept();  
 System.out.println("Client Accepted");  
 ClientHandler clientThread = new ClientHandler(client,con);  
  
 pool.execute(clientThread);  
  
 }  
 }  
}

ClientHandler.java

package movierec;  
  
import java.sql.\*;  
import java.io.\*;  
import java.net.Socket;  
import java.util.ArrayList;  
import java.util.Random;  
  
public class ClientHandler implements Runnable {  
 private Socket client;  
 private BufferedReader in;  
 private PrintWriter out;  
 Connection con;  
 ArrayList<String> usernames = new ArrayList<>();  
 Statement stmt;  
 ArrayList<String> movieswatched = new ArrayList<>();  
 ArrayList<String> moviesnotwatched = new ArrayList<>();  
 boolean contain, passeq;  
  
 public ClientHandler(Socket clientSocket, Connection con) throws IOException, SQLException {  
 this.client = clientSocket;  
 this.con = con;  
 stmt = con.createStatement();  
 in = new BufferedReader(new InputStreamReader(client.getInputStream()));  
 out = new PrintWriter(client.getOutputStream(), true);  
 }  
  
 @Override  
 public void run() {  
 try {  
 // Getting the usernames  
 ResultSet rs = stmt.executeQuery("SELECT username FROM users");  
 while (rs.next()) {  
 usernames.add(rs.getString(1));  
 }  
 while (true) {  
 Statement stmt = con.createStatement();  
 String request = in.readLine();  
 // Checking for socket close case  
 if (request.equals("911")) {  
 try {  
 in.close();  
 out.close();  
 client.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 break;  
 }  
 // Log In request  
 if (request.equals("1")) {  
 while (true) {  
 String username = in.readLine();  
 String password = in.readLine();  
 // Check if username exists  
 contain = usernames.contains(username);  
 if (contain) {  
 // Username existed. check for password  
 out.println("1");  
 rs = stmt.executeQuery("SELECT password FROM users WHERE username=\"" + username + "\";");  
 rs.next();  
 passeq = password.equals(rs.getString(1));  
 if (!passeq) {  
 // Password didn't match  
 out.println("0");  
 } else {  
 // Password matched  
 out.println("1");  
  
 while (true) {  
 String clientresp;  
 clientresp = in.readLine();  
 // Sign out case  
 if (clientresp.equals("3")) {  
 break;  
 } else if (clientresp.equals("x")) { // Delete account case  
 stmt.executeUpdate("ALTER TABLE movietable drop column " + username + ";");  
 stmt.executeUpdate("delete from users where username=\""+username+"\";");  
 break;  
 }  
 // Rewatch Case  
 else if (clientresp.equals("2")) {  
 rs = stmt.executeQuery("SELECT MovieName FROM movietable WHERE " + username + "=\"Y\";");  
 while (rs.next()) {  
 movieswatched.add(rs.getString(1));  
 }  
 String randomMovie;  
  
  
 if (movieswatched.isEmpty()) out.println("404");  
 else {  
 out.println("1");  
 int index = new Random().nextInt(movieswatched.size());  
 randomMovie = movieswatched.get(index);  
 out.println(randomMovie);  
 movieswatched.remove(index);  
 }  
 }  
 // New Recommendation Case  
 else {  
 rs = stmt.executeQuery("SELECT MovieName FROM movietable WHERE " + username + "=\"N\";");  
 while (rs.next()) {  
 moviesnotwatched.add(rs.getString(1));  
 }  
 String randomMovie = "";  
  
 if (moviesnotwatched.isEmpty()) out.println("404");  
 else {  
 out.println("1");  
 int index = new Random().nextInt(moviesnotwatched.size());  
 randomMovie = moviesnotwatched.get(index);  
 out.println(randomMovie);  
  
 moviesnotwatched.remove(index);  
  
  
 stmt.executeUpdate("UPDATE movietable set " + username + "=\"Y\" where MovieName=\"" + randomMovie + "\"");  
 }  
 }  
 }  
 break;  
 }  
 }  
 // Username didn't match  
 else {  
 out.println("0");  
 }  
 }  
 }  
 // Sign Up Case  
 if (request.equals("2")) {  
 while (true) {  
 String usernamesignup = in.readLine();  
 String passwordsignup = in.readLine();  
 if (usernames.contains(usernamesignup)) {  
 // Username Already Exists  
 out.println("0");  
 } else {  
 usernames.add(usernamesignup);  
 stmt.executeUpdate("ALTER TABLE movietable ADD " + usernamesignup + " varchar(1);");  
 stmt.executeUpdate("UPDATE movietable SET " + usernamesignup + "=\"N\"");  
 stmt.executeUpdate("INSERT INTO users values(\"" + usernamesignup + "\",\"" + passwordsignup + "\");");  
 out.println("1");  
 break;  
 }  
 }  
 }  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 in.close();  
 out.close();  
 client.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
 }  
}

Client.java

package movierec;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.\*;  
import java.net.Socket;  
  
import movierec.frames.\*;  
  
public class Client {  
 JFrame homescreen, signuppage, loginpage, recommendationpage;  
 JButton signUp, login, exit, rewatchbutton, newrecbutton, signout, deleteacc, loginbutton, signupbutton;  
 JLabel suerrormessage, lierrormessage, newrecommendation, rewatchrecommendation;  
 JTextArea suusernamefield, liusernamefield;  
 JPasswordField supasswordfield, lipasswordfield;  
 public ActionListener homescreensu, homescreenli, actionListenersignup, actionListenerlogin, rewatchal, newrecal, signoutal, deleteaccal;  
 BufferedReader keyboard, reader;  
 PrintWriter out;  
  
 public Client() {  
 try {  
 Socket socket = new Socket("127.0.0.1", 9169);  
 new frames();  
 homescreen = frames.*getHomescreen*();  
 signUp = frames.*getSignUp*();  
 login = frames.*getLogin*();  
 exit = frames.*getExit*();  
 homescreen.setVisible(true);  
 keyboard = new BufferedReader(new InputStreamReader(System.*in*));  
 reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 out = new PrintWriter(socket.getOutputStream(), true);  
 homescreensu = getHomescreensu();  
 signUp.addActionListener(homescreensu);  
 homescreenli = getHomescreenli();  
 login.addActionListener(homescreenli);  
 exit.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 homescreen.dispose();  
 System.*exit*(0);  
 }  
 });  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) throws IOException {  
 new Client();  
  
 }  
  
 public ActionListener signup() {  
 actionListenersignup = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String username = suusernamefield.getText();  
 String password = supasswordfield.getText();  
 out.println(username);  
 out.println(password);  
  
 try {  
 String serverresp = reader.readLine();  
 if (serverresp.equals("0")) {  
 // Existing user condition  
 suerrormessage.setBounds(700, 200, 300, 30);  
 suerrormessage.setText("Username Already Exists!");  
 } else {  
 // Account will be created successfully here  
 signuppage.dispose();  
 homescreen.setVisible(true);  
 }  
 } catch (Exception exception) {  
 exception.printStackTrace();  
 }  
 }  
 };  
 return actionListenersignup;  
 }  
  
  
 public ActionListener login() {  
  
 actionListenerlogin = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String usernamelogin = liusernamefield.getText();  
 String passwordlogin = lipasswordfield.getText();  
 out.println(usernamelogin);  
 out.println(passwordlogin);  
  
 try {  
 String serverresplogin = reader.readLine();  
 if (serverresplogin.equals("0")) {  
 lierrormessage.setText("Username Doesn't Exist!");  
 } else {  
 String serverresplog = reader.readLine();  
 if (serverresplog.equals("0")) {  
 lierrormessage.setText("Incorrect Password");  
 } else {  
 // Successful login  
 loginpage.setVisible(false);  
 recommendationpage = getrecommendationframe();  
 recommendationpage.setVisible(true);  
 }  
  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
 };  
 return actionListenerlogin;  
 }  
  
 public JFrame getrecommendationframe() {  
 recommendationpage = frames.*getrecommendationframe*();  
 rewatchbutton = frames.*getRewatchbutton*();  
 newrecbutton = frames.*getNewrecbutton*();  
 signout = frames.*getSignoutbutton*();  
 deleteacc = frames.*getDeleteaccbutton*();  
 newrecommendation = frames.*getNewrecommendationlabel*();  
 rewatchrecommendation = frames.*getRewatchrecommendationlabel*();  
 rewatchal = rewatch();  
 rewatchbutton.addActionListener(rewatchal);  
 newrecal = newrecommend();  
 newrecbutton.addActionListener(newrecal);  
 signoutal = signout();  
 signout.addActionListener(signoutal);  
 deleteaccal = deleteacc();  
 deleteacc.addActionListener(deleteaccal);  
 return recommendationpage;  
 }  
  
 public ActionListener newrecommend() {  
 return new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 out.println("1");  
 try {  
 String statusrec = reader.readLine();  
 if (statusrec.equals("404")) {  
 newrecommendation.setText("We're out of movies :(");  
 } else {  
 String movierec = reader.readLine();  
 newrecommendation.setText(movierec);  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
 };  
 }  
  
 public ActionListener rewatch() {  
 return new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 out.println("2");  
 try {  
 String statusrew = reader.readLine();  
 if (statusrew.equals("404")) {  
 rewatchrecommendation.setText("You haven't watched anything.");  
 } else {  
 String rewrec = reader.readLine();  
 rewatchrecommendation.setText(rewrec);  
 }  
 } catch (IOException ioException) {  
 ioException.printStackTrace();  
 }  
 }  
 };  
 }  
  
 public ActionListener signout() {  
 return new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 out.println("3");  
 recommendationpage.dispose();  
 System.*exit*(0);  
 }  
 };  
 }  
  
 public ActionListener deleteacc() {  
 return new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 out.println("x");  
 recommendationpage.dispose();  
 System.*exit*(0);  
 }  
 };  
 }  
  
 public ActionListener getHomescreensu() {  
 return new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 out.println(2);  
 homescreen.dispose();  
 signuppage = frames.*getSignupframe*();  
 suusernamefield = frames.*getSuusernamefield*();  
 supasswordfield = frames.*getSupasswordfield*();  
 suerrormessage = frames.*getSuerrormessage*();  
 signupbutton = frames.*getSignupbutton*();  
 ActionListener signupbuttonactionlistener = signup();  
 signupbutton.addActionListener(signupbuttonactionlistener);  
 signuppage.setVisible(true);  
 }  
 };  
 }  
  
 public ActionListener getHomescreenli() {  
 return new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 out.println(1);  
 loginpage = frames.*getLoginframe*();  
 homescreen.dispose();  
 liusernamefield = frames.*getLiusernamefield*();  
 lipasswordfield = frames.*getLipasswordfield*();  
 lierrormessage = frames.*getLierrormessage*();  
 ActionListener loginbuttonactionlistener = login();  
 loginbutton = frames.*getLoginbutton*();  
 loginbutton.addActionListener(loginbuttonactionlistener);  
 loginpage.setVisible(true);  
 }  
 };  
 }  
}

frames.java

package movierec;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionListener;  
  
public class frames {  
 static JFrame *homescreen*, *recframe*, *loginframe*, *signuppage*;  
 static JButton *login*, *signUp*, *exit*, *newrecbutton*, *rewatchbutton*, *signout*, *deleteacc*, *loginbutton*, *signupbutton*;  
 static JLabel *newrecommendation*, *rewatchrecommendation*, *lierrormessage*, *suerrormessage*;  
 static JTextArea *suusernamefield*, *liusernamefield*;  
 static JPasswordField *supasswordfield*, *lipasswordfield*;  
  
 public frames() {  
 *setuphomescreenframe*();  
 *setupsignupframe*();  
 *setuploginframe*();  
 *setuprecommendationframe*();  
 }  
  
 public static JButton getLogin() {  
 return *login*;  
 }  
  
 public static JButton getExit() {  
 return *exit*;  
 }  
  
 public static JButton getSignUp() {  
 return *signUp*;  
 }  
  
 public static void setuphomescreenframe() {  
 *homescreen* = new JFrame();  
 *homescreen*.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 *homescreen*.setResizable(true);  
 *homescreen*.setPreferredSize(new Dimension(1540, 825));  
 *homescreen*.setTitle("Movie recommender");  
// backg = new JLabel(new ImageIcon("D:\\SEMV\\SDL\\Assignment4\\assets\\moviegrid.jpg"));  
// backg.setBounds(0,0,1920,1080);  
 JPanel panel = new JPanel();  
 panel.setLayout(null);  
 panel.setBackground(Color.*DARK\_GRAY*);  
 *signUp* = new JButton("Sign Up");  
 *signUp*.setBounds(865, 540, 100, 100);  
 *signUp*.setBackground(Color.*orange*);  
 *login* = new JButton("Login");  
 *login*.setBounds(565, 540, 100, 100);  
 *login*.setBackground(Color.*orange*);  
 JLabel label1 = new JLabel("Movie Recommendations");  
 label1.setFont(new Font(Font.*SANS\_SERIF*, Font.*PLAIN*, 50));  
 label1.setBounds(490, 150, 650, 50);  
 label1.setForeground(Color.*lightGray*);  
 *exit* = new JButton("Exit");  
 *exit*.setBounds(1400, 700, 60, 60);  
 *exit*.setBackground(Color.*pink*);  
// panel.add(backg);  
 panel.add(*login*);  
 panel.add(*signUp*);  
 panel.add(*exit*);  
 panel.add(label1);  
 *homescreen*.add(panel);  
 *homescreen*.pack();  
 }  
  
 public static void setuprecommendationframe() {  
 *recframe* = new JFrame();  
 *recframe*.setPreferredSize(new Dimension(1540, 825));  
 *recframe*.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 *recframe*.setResizable(true);  
 *recframe*.setTitle("LoginPage");  
 JPanel recpanel = new JPanel();  
 recpanel.setLayout(null);  
 recpanel.setBackground(Color.*DARK\_GRAY*);  
 JLabel title = new JLabel("Click on the desired button");  
 title.setBounds(600, 50, 900, 50);  
 title.setFont(new Font("title", Font.*ITALIC*, 30));  
 title.setForeground(Color.*lightGray*);  
 *newrecbutton* = new JButton("New Recommendation");  
 *newrecbutton*.setBounds(300, 200, 200, 50);  
 *newrecbutton*.setBackground(Color.*orange*);  
 *newrecommendation* = new JLabel("");  
 *newrecommendation*.setFont(new Font("new reco.", Font.*BOLD*, 15));  
 *newrecommendation*.setBounds(600, 200, 700, 20);  
 *newrecommendation*.setForeground(Color.*lightGray*);  
 *rewatchbutton* = new JButton("Rewatch recommendation");  
 *rewatchbutton*.setBounds(300, 300, 200, 50);  
 *rewatchbutton*.setBackground(Color.*orange*);  
 *rewatchrecommendation* = new JLabel("");  
 *rewatchrecommendation*.setFont(new Font("rewatch reco.", Font.*BOLD*, 15));  
 *rewatchrecommendation*.setBounds(600, 300, 700, 15);  
 *rewatchrecommendation*.setForeground(Color.*lightGray*);  
 *signout* = new JButton("Sign Out");  
 *signout*.setBounds(1300, 10, 200, 50);  
 *signout*.setBackground(Color.*WHITE*);  
 *deleteacc* = new JButton("Delete Account");  
 *deleteacc*.setBounds(1300, 600, 200, 50);  
 *deleteacc*.setBackground(Color.*RED*);  
 recpanel.add(title);  
 recpanel.add(*newrecbutton*);  
 recpanel.add(*signout*);  
 recpanel.add(*rewatchrecommendation*);  
 recpanel.add(*rewatchbutton*);  
 recpanel.add(*newrecommendation*);  
 recpanel.add(*deleteacc*);  
 *recframe*.add(recpanel);  
 *recframe*.pack();  
 }  
  
 public static void setuploginframe() {  
 *loginframe* = new JFrame();  
 *loginframe*.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 *loginframe*.setResizable(true);  
 *loginframe*.setPreferredSize(new Dimension(1540, 825));  
 *loginframe*.setTitle("LoginPage");  
 JPanel loginpanel = new JPanel();  
 loginpanel.setLayout(null);  
 loginpanel.setBackground(Color.*DARK\_GRAY*);  
 *loginbutton* = new JButton("Login");  
 *loginbutton*.setBounds(720, 600, 100, 50);  
 *loginbutton*.setBackground(Color.*orange*);  
 JLabel liusername = new JLabel("Username:");  
 liusername.setFont(new Font("username", Font.*BOLD*, 15));  
 liusername.setBounds(600, 300, 100, 15);  
 liusername.setForeground(Color.*lightGray*);  
 JLabel lipassword = new JLabel("Password:");  
 lipassword.setFont(new Font("password", Font.*BOLD*, 15));  
 lipassword.setBounds(600, 400, 100, 15);  
 lipassword.setForeground(Color.*lightGray*);  
 JLabel log\_in = new JLabel("Enter your credentials");  
 log\_in.setFont(new Font("login", Font.*PLAIN*, 50));  
 log\_in.setBounds(515, 150, 650, 60);  
 log\_in.setForeground(Color.*lightGray*);  
 *lierrormessage* = new JLabel("");  
 *lierrormessage*.setForeground(Color.*red*);  
 *lierrormessage*.setFont(new Font(Font.*SANS\_SERIF*, Font.*PLAIN*, 15));  
 *lierrormessage*.setBounds(700, 200, 300, 30);  
 *liusernamefield* = new JTextArea();  
 *lipasswordfield* = new JPasswordField();  
 *liusernamefield*.setBounds(700, 300, 200, 20);  
 *lipasswordfield*.setBounds(700, 400, 200, 20);  
  
  
 loginpanel.add(log\_in);  
 loginpanel.add(*loginbutton*);  
 loginpanel.add(*lierrormessage*);  
 loginpanel.add(liusername);  
 loginpanel.add(lipassword);  
 loginpanel.add(*lipasswordfield*);  
 loginpanel.add(*liusernamefield*);  
 *loginframe*.add(loginpanel);  
 *loginframe*.pack();  
  
  
 }  
  
 public static void setupsignupframe() {  
 *signuppage* = new JFrame();  
 *signuppage*.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 *signuppage*.setResizable(true);  
 *signuppage*.setPreferredSize(new Dimension(1540, 825));  
 *signuppage*.setTitle("Sign Up");  
 JPanel signuppanel = new JPanel();  
 signuppanel.setLayout(null);  
 signuppanel.setBackground(Color.*DARK\_GRAY*);  
 *signupbutton* = new JButton("Sign Up");  
 *signupbutton*.setBounds(720, 600, 100, 50);  
 *signupbutton*.setBackground(Color.*orange*);  
 JLabel suusername = new JLabel("Username:");  
 suusername.setFont(new Font("username", Font.*BOLD*, 15));  
 suusername.setBounds(600, 300, 100, 15);  
 suusername.setForeground(Color.*lightGray*);  
 JLabel supassword = new JLabel("Password:");  
 supassword.setFont(new Font("password", Font.*BOLD*, 15));  
 supassword.setBounds(600, 400, 100, 15);  
 supassword.setForeground(Color.*lightGray*);  
 JLabel sign\_up = new JLabel("Enter Details to register");  
 sign\_up.setFont(new Font("signup", Font.*PLAIN*, 50));  
 sign\_up.setBounds(515, 150, 650, 60);  
 sign\_up.setForeground(Color.*lightGray*);  
 *suerrormessage* = new JLabel("");  
 *suerrormessage*.setForeground(Color.*red*);  
 *suerrormessage*.setFont(new Font("Error Message", Font.*PLAIN*, 15));  
 *suerrormessage*.setBounds(700, 200, 300, 30);  
 *suusernamefield* = new JTextArea();  
 *supasswordfield* = new JPasswordField();  
 *suusernamefield*.setBounds(700, 300, 200, 20);  
 *supasswordfield*.setBounds(700, 400, 200, 20);  
  
  
 signuppanel.add(sign\_up);  
 signuppanel.add(*signupbutton*);  
 signuppanel.add(*suerrormessage*);  
 signuppanel.add(suusername);  
 signuppanel.add(supassword);  
 signuppanel.add(*supasswordfield*);  
 signuppanel.add(*suusernamefield*);  
 *signuppage*.add(signuppanel);  
 *signuppage*.pack();  
 }  
  
 public static JFrame getHomescreen() {  
 return *homescreen*;  
 }  
  
 public static JFrame getLoginframe() {  
 return *loginframe*;  
 }  
  
 public static JFrame getSignupframe() {  
 return *signuppage*;  
 }  
  
 public static JFrame getrecommendationframe() {  
 return *recframe*;  
 }  
  
 public static JLabel getNewrecommendationlabel() {  
 return *newrecommendation*;  
 }  
  
 public static JLabel getRewatchrecommendationlabel() {  
 return *rewatchrecommendation*;  
 }  
  
 public static JButton getRewatchbutton() {  
 return *rewatchbutton*;  
 }  
  
 public static JButton getDeleteaccbutton() {  
 return *deleteacc*;  
 }  
  
 public static JButton getSignoutbutton() {  
 return *signout*;  
 }  
  
 public static JButton getNewrecbutton() {  
 return *newrecbutton*;  
 }  
  
 public static JButton getLoginbutton() {  
 return *loginbutton*;  
 }  
  
 public static JButton getSignupbutton() {  
 return *signupbutton*;  
 }  
  
 public static JPasswordField getLipasswordfield() {  
 return *lipasswordfield*;  
 }  
  
 public static JPasswordField getSupasswordfield() {  
 return *supasswordfield*;  
 }  
  
 public static JTextArea getLiusernamefield() {  
 return *liusernamefield*;  
 }  
  
 public static JTextArea getSuusernamefield() {  
 return *suusernamefield*;  
 }  
  
 public static JLabel getLierrormessage() {  
 return *lierrormessage*;  
 }  
  
 public static JLabel getSuerrormessage() {  
 return *suerrormessage*;  
 }  
}

Output

