

## CS 313: Advanced Programming Language Term Project

Title: Gym membership management system.

Section: 5CA

Group no. 2

No.	Student Name	Student Number
1	Dana Alsubiheen	444008958
2	Duna Alomran	444008945
3	Areej Alyami	444008928
4	Joud Alrajeh	444008827
5	Layan Alenzi	444008882

First Semester 2024

<b>Content</b>	<b>Mark Obtain</b>
<b>Project Description</b>	
<b>Class Diagram</b>	
<b>GUI Design</b>	
<b>Database Tables</b>	
<b>Connection and Queries on Database</b>	
<b>Appendix (code)</b>	
<b>TOTAL SCORE</b>	<b>/10</b>

## Contents

1.	Project Description.....	4
2.	Class Diagram .....	6
3.	GUI Design.....	7
4.	Database Tables .....	15
5.	Connection and Queries on Database .....	16
6.	Appendix.....	31

# 1. Project Description

This project is a Gym Management System design to supply members, trainers and, admins with numerous tools and interface to oversee various gym activities and resources, here is a detail of the system's functions:

First of all the first view of gym application is the login.fxml interface that allows to the user to enter to the application by its ID and Password.

## Admin view:

### 1. User Management

- **Add User:** Administrators can register new users.
- **remove User:** allows administrators to remove users from the system by user id, and all information related to this user will be removed also, for example if trainer removed all classes for this trainer will be removed also, and for member all progress track information for this member will be removed.
- **Update User Information:** Administrators can modify existing user details.

### 2. Service Management

- displays all services provided by the gym.
- Administrators can add and remove service.

### 3. Offerings

Administrators can add offers.

### 4. Logout

## Trainer view:

### 1. Add new session

- Administrators can create new training sessions with specified all information related to this session.

### 2. Edit

- The trainer can edit the name and time of the class via class id.

### **3. Remove**

- Trainer can remove class by class id.

### **4. Track progress**

- Members can track their fitness achievements by entering their id.
- The overall feedback will appear after clicking submit button.

### **5. Logout**

## **Member view:**

1. This view shows the member id, member name, upcoming sessions.

### **2. Group classes**

- Member can view all available classes and select one of them to join it.
- After joining to the class, the class will appear at upcoming classes in member view.  
\*Note that, each member can join only to one class.

### **3. Edit profile**

- Each member can view and edit his/her profile at the same time.

### **4. Offers**

- The member can view all offers.

### **5. Logout**

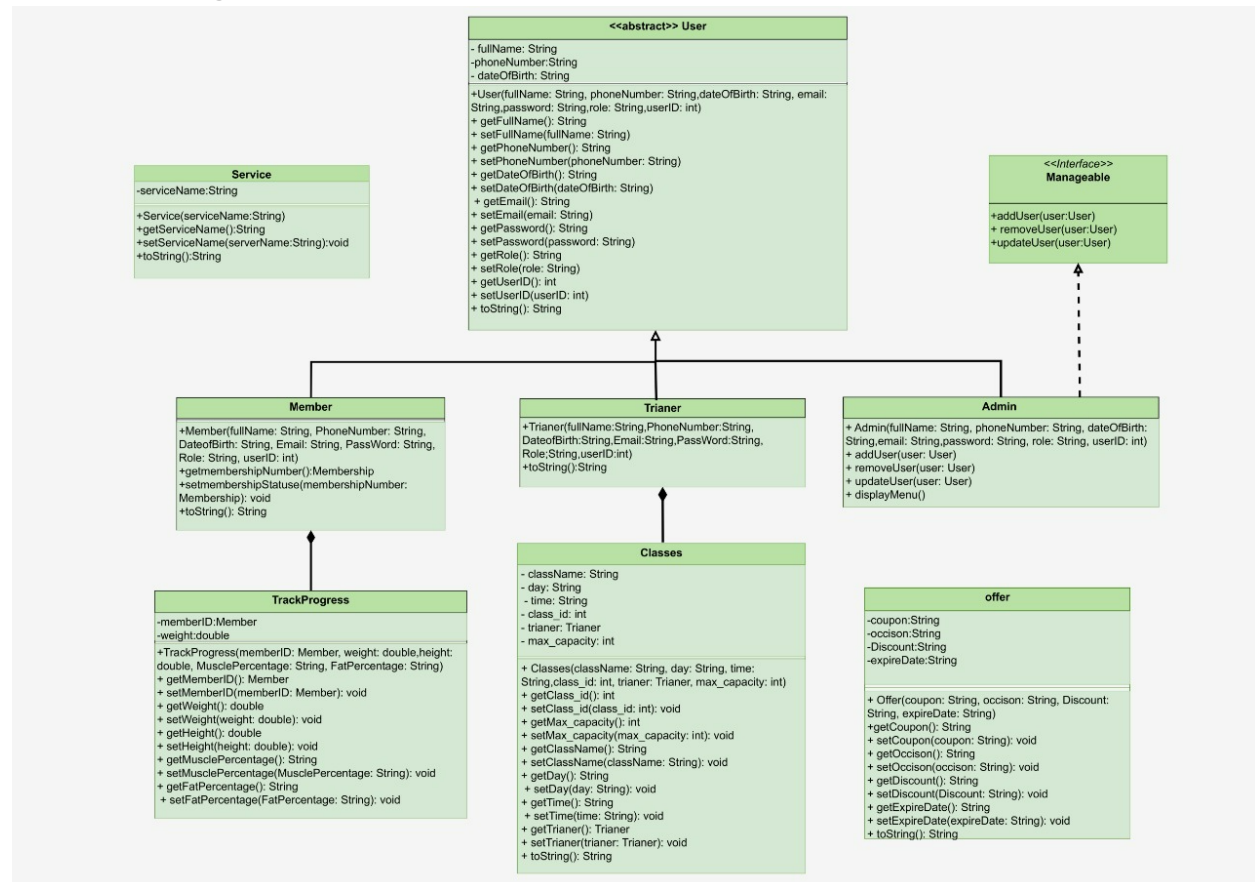
## **Database Management:**

- In this project we used Oracle database, the database schema for this project comprises several key tables that collectively store essential information related to users, classes, services, offers, and member progress tracking. Each table serves a specific purpose and contributes to the overall functionality of the system.

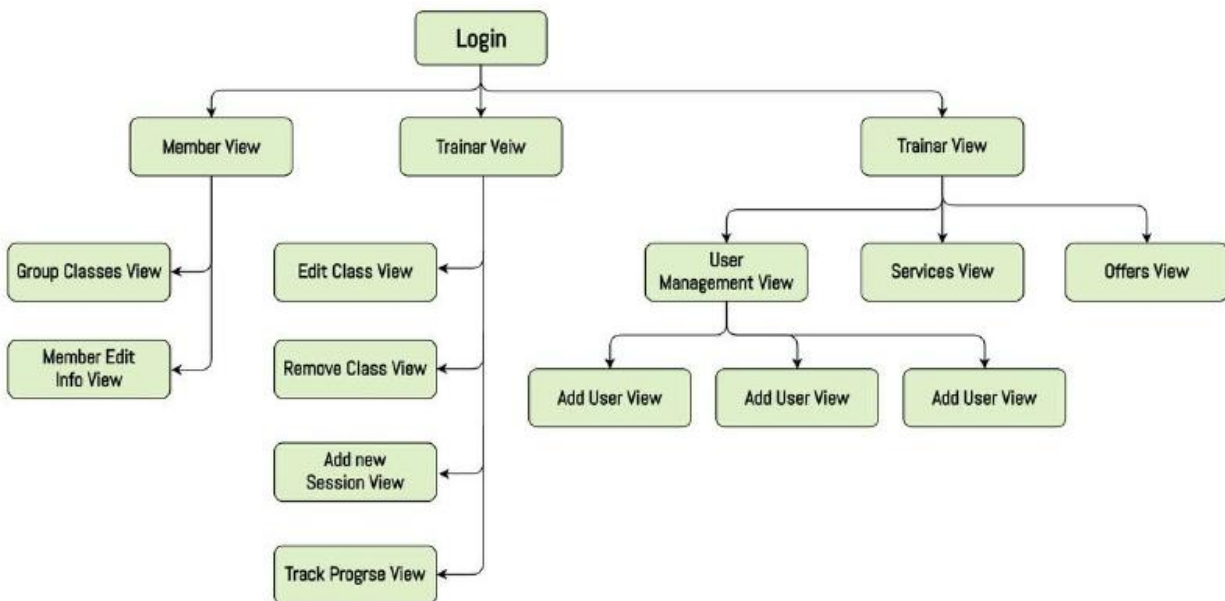
## **Interface Design and User Experience:**

- The system employs a CSS-based design, defined in Style.css, for consistent and user-friendly interfaces.

## 2. Class Diagram

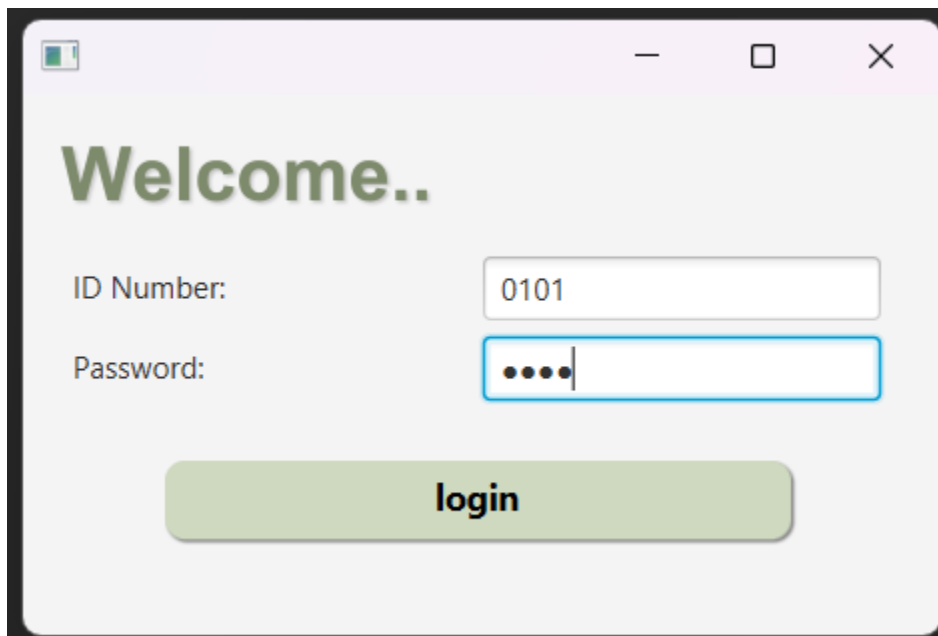


This is a hierarchy for GUI view:



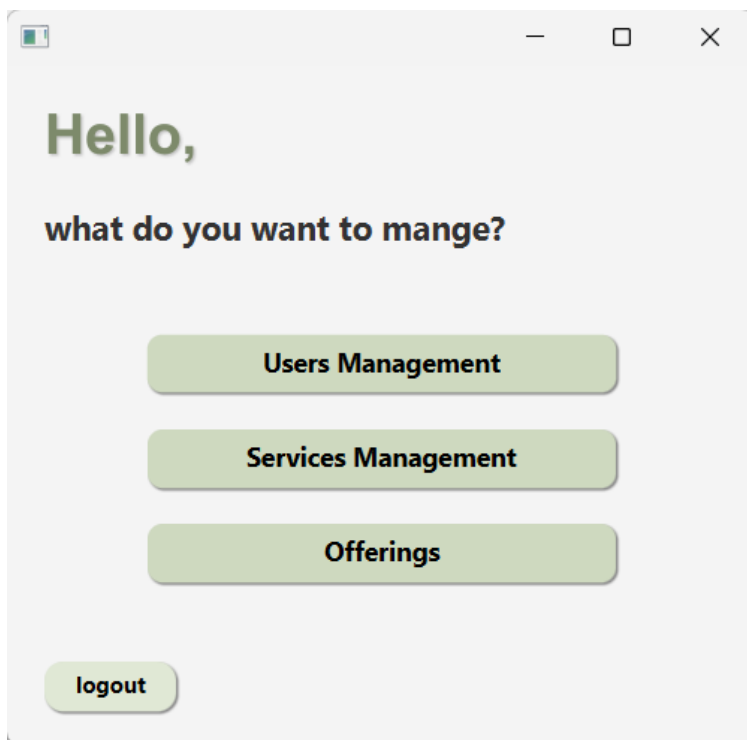
### 3. GUI Design

LoginControllre:



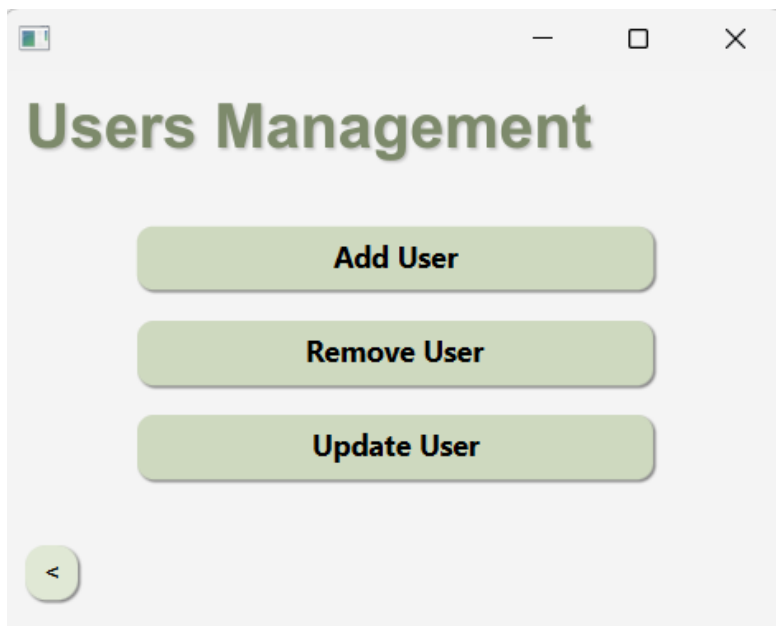
A mockup of a login window. The window has a title bar with a green icon, a minus sign, a maximize button, and a close button. The main content area has a light gray background. At the top, the text "Welcome.." is displayed in a large, bold, green font. Below this, there are two input fields. The first is labeled "ID Number:" and contains the text "0101". The second is labeled "Password:" and contains four black dots. Below the input fields is a large, rounded green button with the text "login" in black.

AdminViewControllre:



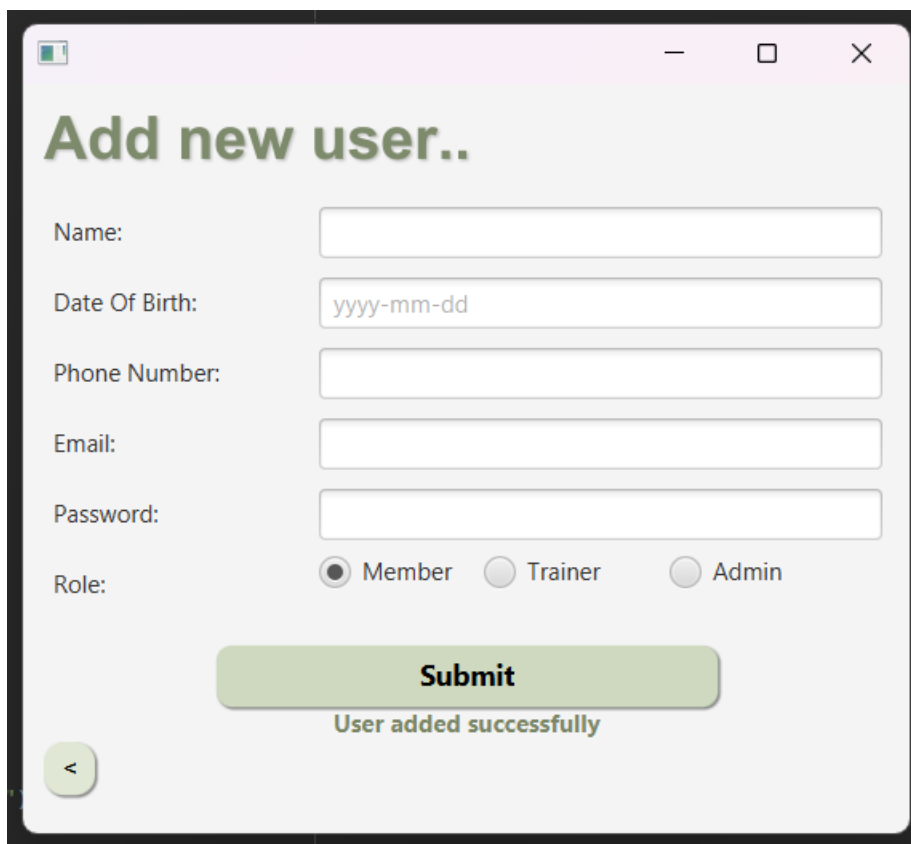
A mockup of an admin view window. The window has a title bar with a green icon, a minus sign, a maximize button, and a close button. The main content area has a light gray background. At the top, the text "Hello," is displayed in a large, bold, green font. Below this, the text "what do you want to mange?" is displayed in a bold, black font. Below the text are three rounded green buttons stacked vertically, each with black text: "Users Management", "Services Management", and "Offerings". At the bottom left, there is a small, rounded green button with the text "logout" in black.

UserManagmentCotroller:



A window titled "Users Management" with a light gray background. It features three large, rounded green buttons stacked vertically: "Add User", "Remove User", and "Update User". In the bottom-left corner, there is a small green circular button with a white left-pointing arrow. The window has standard OS window controls (minimize, maximize, close) in the top-right corner.

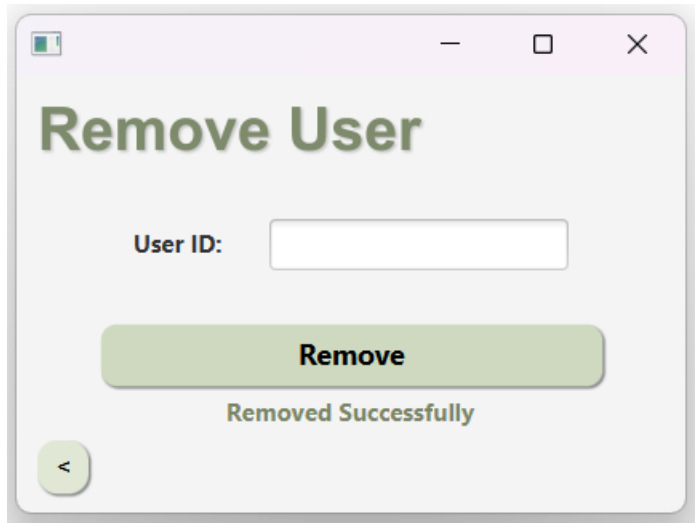
AddUserController:



A window titled "Add new user.." with a light gray background and a pink header bar. It contains several input fields: "Name:", "Date Of Birth:" (with a placeholder "yyyy-mm-dd"), "Phone Number:", "Email:", and "Password:". Below these is a "Role:" section with three radio buttons labeled "Member", "Trainer", and "Admin", where "Member" is selected. A large green "Submit" button is centered at the bottom. Below the button, the text "User added successfully" is displayed. In the bottom-left corner, there is a small green circular button with a white left-pointing arrow. The window has standard OS window controls in the top-right corner.

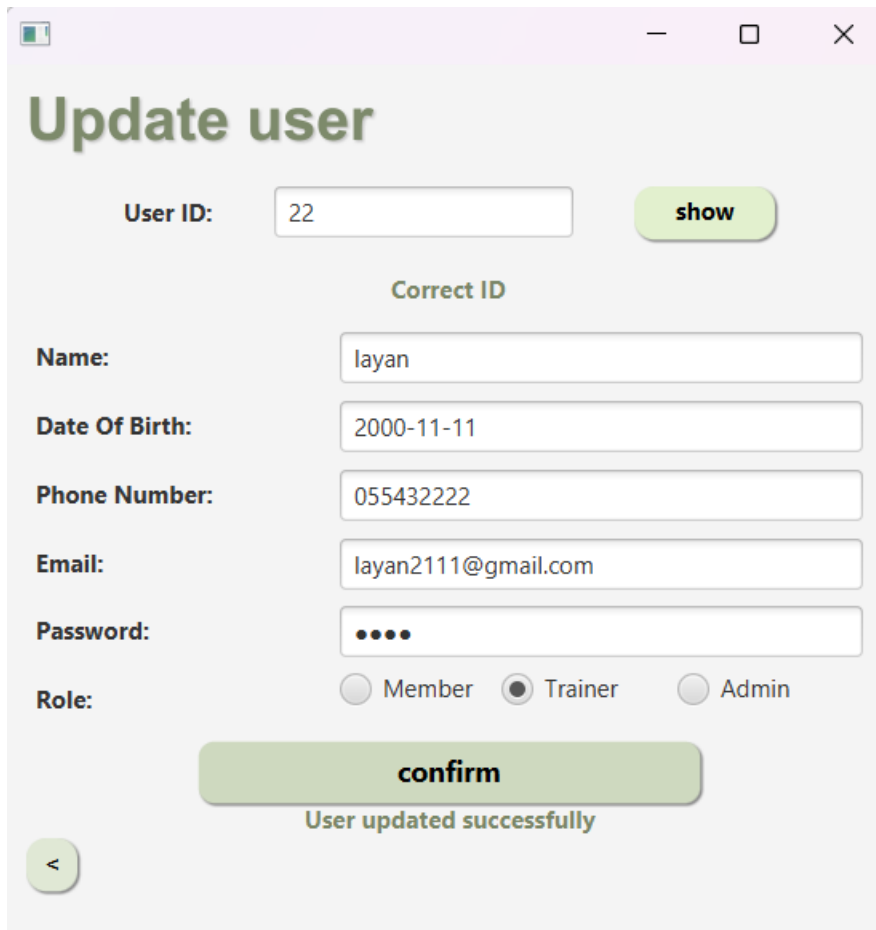


RemoveUserController:



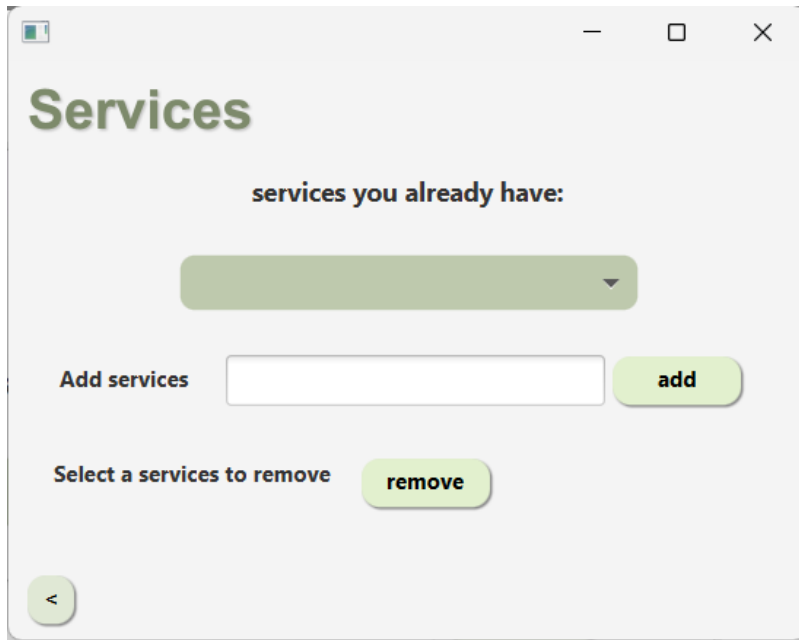
A dialog box titled "Remove User" with a light green header. It contains a "User ID:" label and a text input field. Below the input field is a green button labeled "Remove". Underneath the button, the text "Removed Successfully" is displayed. In the bottom-left corner, there is a green circular button with a white left-pointing arrow.

UpdateUserController:



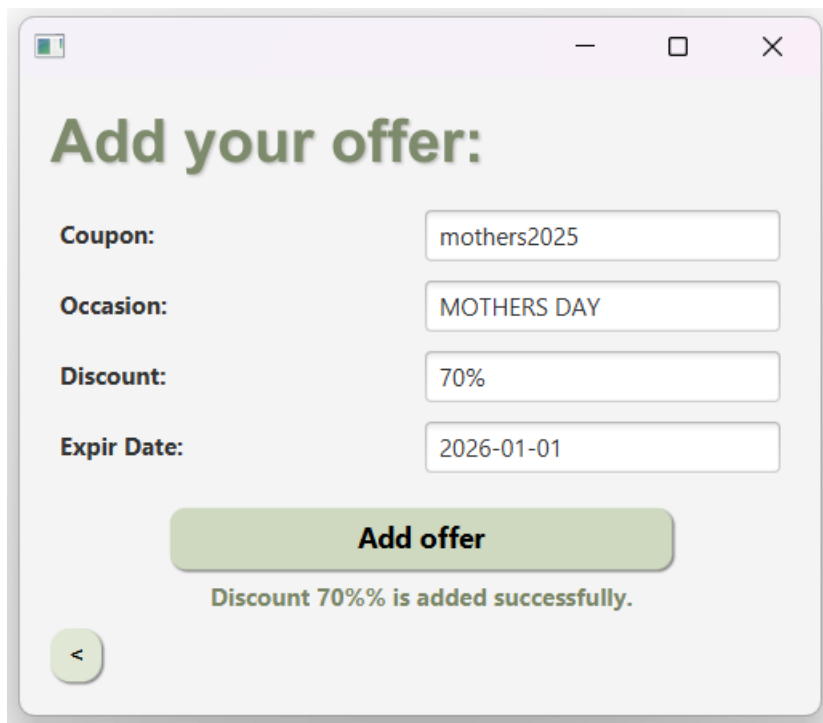
A form titled "Update user" with a light green header. It features a "User ID:" label and a text input field containing the value "22". To the right of the input field is a green button labeled "show". Below this, the text "Correct ID" is displayed. The form contains several labeled input fields: "Name:" with the value "layan", "Date Of Birth:" with the value "2000-11-11", "Phone Number:" with the value "055432222", "Email:" with the value "layan2111@gmail.com", and "Password:" with masked characters "••••". Below these fields is a "Role:" label followed by three radio buttons: "Member", "Trainer" (which is selected), and "Admin". At the bottom, there is a green button labeled "confirm". Below the button, the text "User updated successfully" is displayed. In the bottom-left corner, there is a green circular button with a white left-pointing arrow.

ServiceViewControllre:



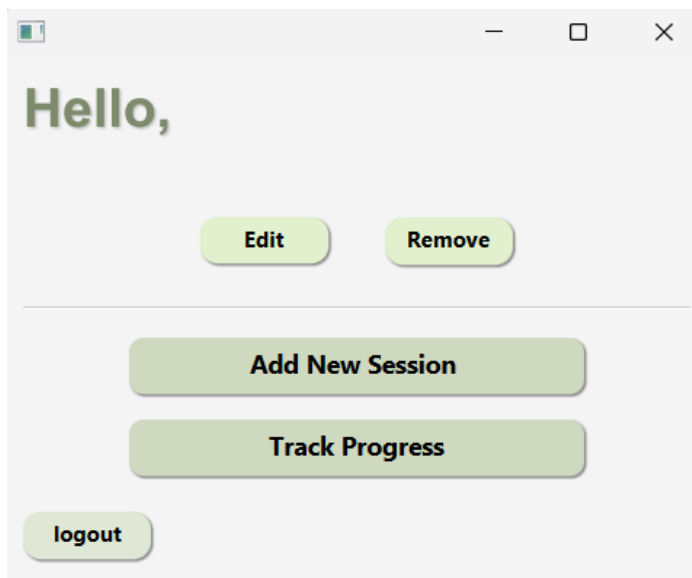
The screenshot shows a window titled "Services" with a light gray background. At the top, there's a header "Services" in a bold, dark green font. Below it, the text "services you already have:" is displayed. Underneath, there's a green rectangular button with a white downward arrow. Further down, there's a section labeled "Add services" with a white text input field and a green "add" button. Below that, there's a section labeled "Select a services to remove" with a green "remove" button. At the bottom left, there's a green circular button with a white left arrow.

OffersControllre:

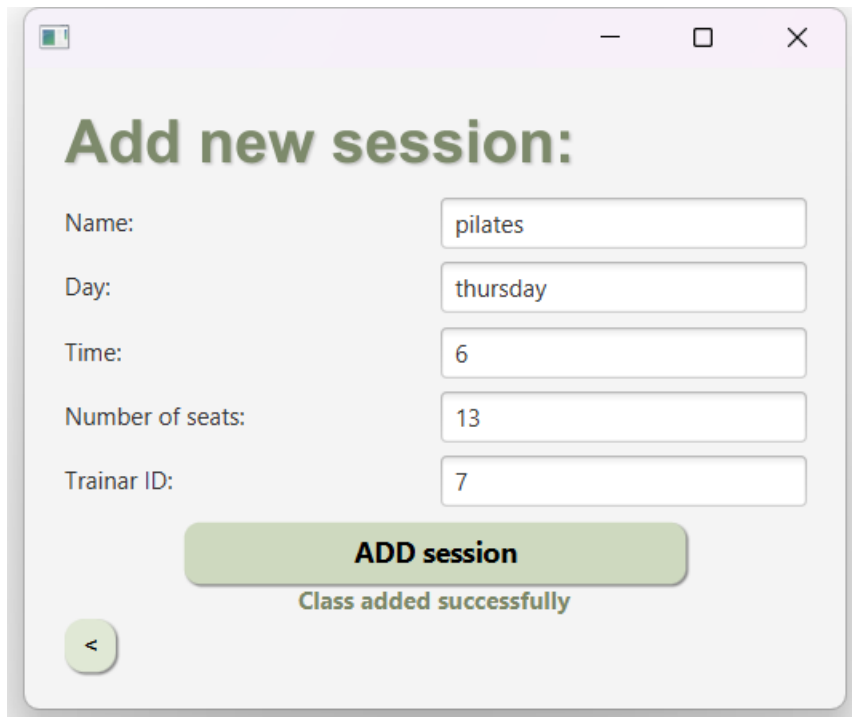


The screenshot shows a window titled "Add your offer:" with a light purple header. Below the header, there are four rows of form fields: "Coupon:" with a text input containing "mothers2025", "Occasion:" with a text input containing "MOTHERS DAY", "Discount:" with a text input containing "70%", and "Expir Date:" with a text input containing "2026-01-01". Below these fields is a large green "Add offer" button. Underneath the button, a message states "Discount 70%% is added successfully." (Note the double percent sign). At the bottom left, there's a green circular button with a white left arrow.

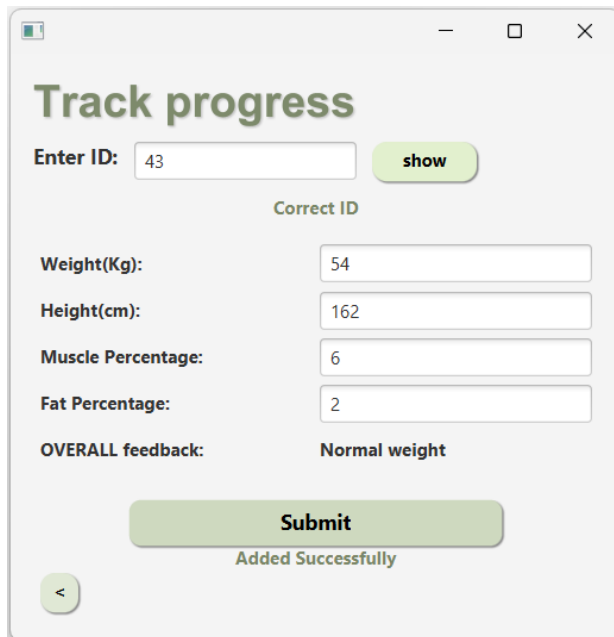
TrainerViewControllre:



AddSessionControllre:



TrackMemberPrograss:



**Track progress**

Enter ID:  show

Correct ID

Weight(Kg):

Height(cm):

Muscle Percentage:

Fat Percentage:

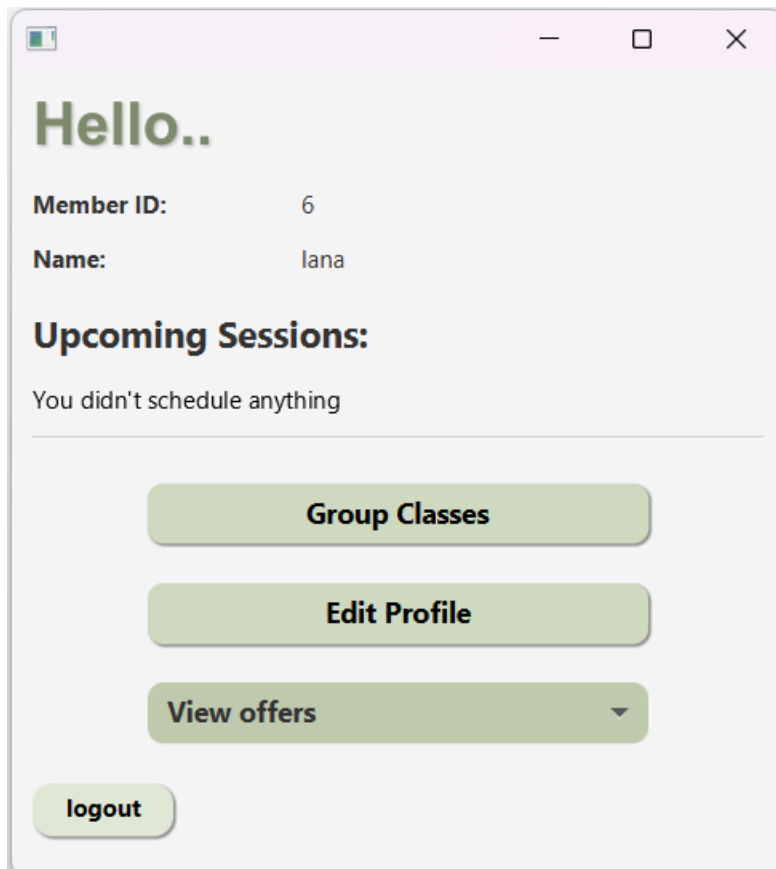
OVERALL feedback: Normal weight

Submit

Added Successfully

<

MemberViewController:



**Hello..**

Member ID: 6

Name: Iana

**Upcoming Sessions:**

You didn't schedule anything

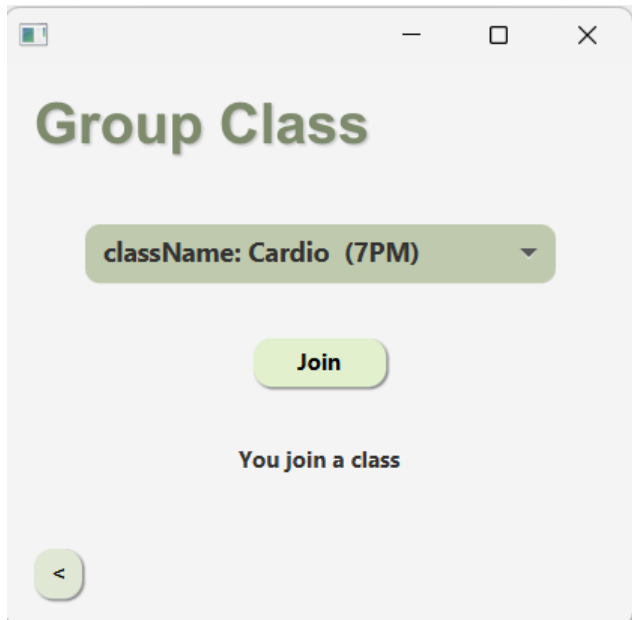
Group Classes

Edit Profile

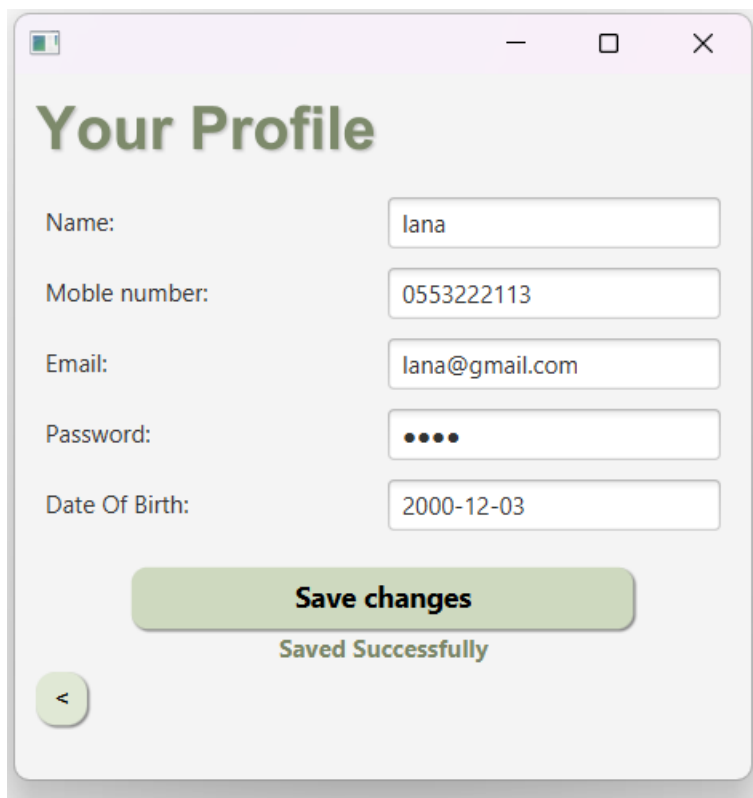
View offers

logout

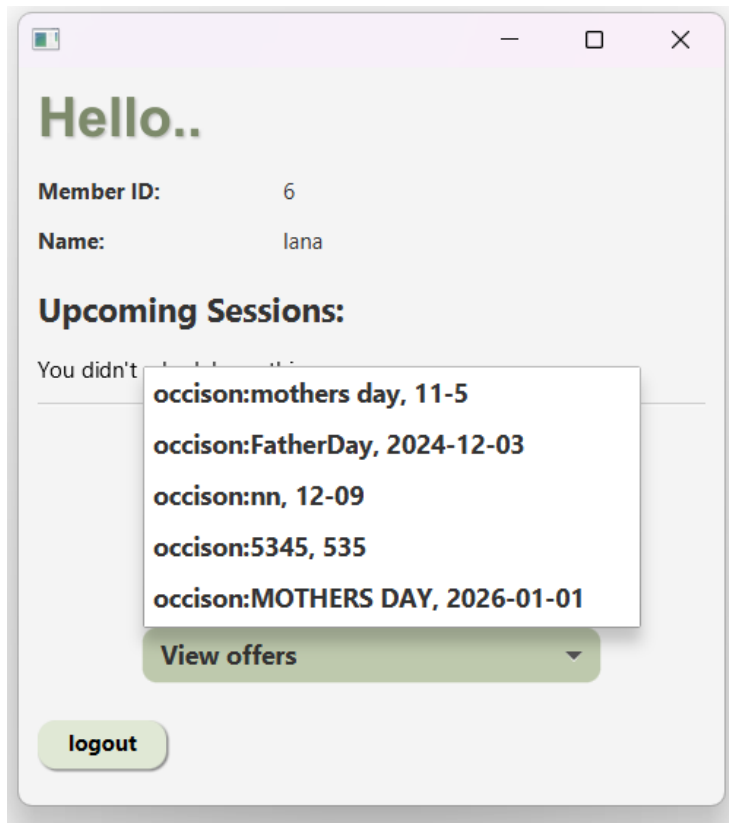
GroupClassesControllre:



EditeProfileControllre:



ViewOffers:



## 4. Database Tables

-- users Table --

```
CREATE TABLE useerss_ (  
    user_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  
    password VARCHAR(100),  
    email VARCHAR(100) UNIQUE,  
    full_name VARCHAR(100),  
    date_of_birth DATE,  
    phone_number VARCHAR(20),  
    role VARCHAR(20)  
);
```

```
CREATE TABLE Classes__ (  
    class_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  
    class_name VARCHAR(20) NOT NULL,  
    day VARCHAR(100),  
    time VARCHAR(100),  
    max_capacity NUMBER(2),  
    trainer_id NUMBER,  
    FOREIGN KEY (trainer_id) REFERENCES useerss_(user_id)  
);
```

-- Service Table

```
CREATE TABLE services (  
    service_name VARCHAR(20)  
);
```

```
CREATE TABLE Offer (  
    coupon VARCHAR2(50),  
    occasion VARCHAR2(100),  
    discount VARCHAR2(10),  
    expireDate VARCHAR2(10)  
);
```

```
CREATE TABLE traProgres__ (  
    member_id NUMBER NOT NULL,  
    weight NUMBER(5, 2),  
    height NUMBER(5, 2),  
    MusclePercentage VARCHAR2(10),  
    FatPercentage VARCHAR2(10),  
    FOREIGN KEY (member_id) REFERENCES useerss_(user_id)  
);
```

## 5. Connection and Queries on Database

```
import java.sql.Connection;

import java.sql.Statement;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DataBaseConnection {

    // data base connection details

    private static final String db_url = "jdbc:oracle:thin:@192.168.100.18:1521:xe";

    private static final String db_user = "system";

    private static final String db_pass = "1234";

    public Connection getCon() {

        return con;

    }

    // singleton instance and connection

    private static DataBaseConnection handler = null;

    public Connection con = null;

    private Statement stmt = null;

    private DataBaseConnection() {

        createConnection();

    }

    // method to get the singleton instance of databaseConnection

    public static DataBaseConnection getInstance() throws SQLException {

        if (handler == null || handler.con.isClosed()) { // create new instance

            handler = new DataBaseConnection();

        }

        System.out.println("");

        return handler;

    }

}
```



```

    }

    public void createConnection() {

        try {

            Class.forName("oracle.jdbc.driver.OracleDriver"); // load the oracle JDBC driver

            con = DriverManager.getConnection(db_url, db_user, db_pass); //

            System.out.println("Connection established!");

        } catch (ClassNotFoundException e) {

            System.err.println("JDBC Driver Class not found: " + e.getMessage());

            e.printStackTrace();

        } catch (SQLException e) {

            System.err.println("SQL Exception " + e.getMessage());

            e.printStackTrace();

        } catch (Exception e) {

            System.err.println("Exception " + e.getMessage());

            e.printStackTrace();

        }

    }

}

//USER

    public void insertUser(String password, String email, String fullName, Date dateOfBirth, String phoneNumber,
String role) throws SQLException {

        String sql = "INSERT INTO useerss_ ( password, email, full_name, date_of_birth, phone_number, role) VALUES
(?, ?, ?, ?, ?, ?)";

        try ( PreparedStatement stmt = DataBaseConnection.getInstance().getCon().prepareStatement(sql)) {

            //stmt.setInt(1, userId);

            stmt.setString(1, password);

            stmt.setString(2, email);

            stmt.setString(3, fullName);

            stmt.setDate(4, dateOfBirth);


```

```

        stmt.setString(5, phoneNumber);

        stmt.setString(6, role);

        stmt.executeUpdate();

    }

}

// Update User

public void updateUser(int userId, String password, String email, String fullName, Date dateOfBirth, String
phoneNumber, String role) throws SQLException {

    String sql = "UPDATE useerss_ SET password = ?, email = ?, full_name = ?, date_of_birth = ?, phone_number =
?, role =? WHERE user_id = ?";

    try ( PreparedStatement stmt = DataBaseConnection.getInstance().getCon().prepareStatement(sql)) {

        stmt.setString(1, password);

        stmt.setString(2, email);

        stmt.setString(3, fullName);

        stmt.setDate(4, dateOfBirth);

        stmt.setString(5, phoneNumber);

        stmt.setString(6, role);

        stmt.setInt(7, userId);

        stmt.executeUpdate();

    }

}

// Delete User

public void deleteUser(int userId) throws SQLException {

    String sql = "DELETE FROM useerss_ WHERE user_id = ?";

    try ( PreparedStatement stmt = DataBaseConnection.getInstance().getCon().prepareStatement(sql)) {

        stmt.setInt(1, userId);

        stmt.executeUpdate();

    }

}

```

```

    }

    //Get User By Id

    public User getUserById(int userId) throws SQLException {

        String sql = "SELECT * FROM useerss_ WHERE user_id = ?";

        PreparedStatement stmt = DataBaseConnection.getInstance().getCon().prepareStatement(sql);

        stmt.setInt(1, userId);

        ResultSet rs = stmt.executeQuery();

        User user = null;

        while (rs.next()) {

            String password = rs.getString("password");

            String email = rs.getString("email");

            String fullName = rs.getString("full_name");

            Date dateOfBirth = rs.getDate("date_of_birth");

            String phoneNumber = rs.getString("phone_number");

            String role = rs.getString("role");

            if (role.equalsIgnoreCase("ADMIN")) {

                user = new Admin(fullName, phoneNumber, dateOfBirth.toString(), email, password, role, userId);

            } else if (role.equalsIgnoreCase("TRAINER")) {

                user = new Trianer(fullName, phoneNumber, dateOfBirth.toString(), email, password, role, userId);

            } else if (role.equalsIgnoreCase("MEMBER")) {

                user = new Member(fullName, phoneNumber, dateOfBirth.toString(), email, password, role, userId);

            }

        }

        return user;

    }

    // Retrieve All Users

    public List<User> retrieveAllUsers() throws SQLException {

        List<User> users = new ArrayList<>();

```

```

String sql = "SELECT user_id, password, email, full_name, date_of_birth, phone_number FROM useerss_";

try ( PreparedStatement stmt = DataBaseConnection.getInstance().getCon().prepareStatement(sql); ResultSet
rs = stmt.executeQuery()) {

    while (rs.next()) {

        int userId = rs.getInt("user_id");

        String password = rs.getString("password");

        String email = rs.getString("email");

        String fullName = rs.getString("full_name");

        Date dateOfBirth = rs.getDate("date_of_birth");

        String phoneNumber = rs.getString("phone_number");

        String role = rs.getString("role");

        if (role.equalsIgnoreCase("ADMIN")) {

            users.add(new Admin(fullName, phoneNumber, dateOfBirth.toString(), email, password, role, userId));

        } else if (role.equalsIgnoreCase("TRAINER")) {

            users.add(new Trianer(fullName, phoneNumber, dateOfBirth.toString(), email, password, role, userId));

        } else if (role.equalsIgnoreCase("MEMBER")) {

            users.add(new Member(fullName, phoneNumber, dateOfBirth.toString(), email, password, role,
userId));

        }

    }

}

return users;

}

//Classes

public void insertClass(String className, String day, String time, int max_capacity, int trainerId) throws
SQLException {

    String sql = "INSERT INTO Classes__ (class_name, day, time, max_capacity, trainer_id) VALUES (?, ?, ?, ?, ?)";

```

```

try {

    conn = DataBaseConnection.getInstance().getCon();

    PreparedStatement stmt = conn.prepareStatement(sql);

    stmt.setString(1, className);

    stmt.setString(2, day);

    stmt.setString(3, time);

    stmt.setInt(4, max_capacity);

    stmt.setInt(5, trainerId);

    stmt.executeUpdate();

} catch (SQLException e) {

    e.printStackTrace();

}

}

// Update Class

public boolean updateClass(int classId, String className, String time) {

    String checkSql = "SELECT class_id FROM Classes__ WHERE class_id = ?";

    String updateSql = "UPDATE Classes__ SET class_name = ?, time = ? WHERE class_id = ?";

    try ( Connection conn = DataBaseConnection.getInstance().getCon()) {

        // Step 1: Check if the class_id exists in the database

        PreparedStatement checkStmt = conn.prepareStatement(checkSql);

        checkStmt.setInt(1, classId);

        ResultSet rs = checkStmt.executeQuery();

        // If no class_id is found, return false

        if (!rs.next()) {

            return false;

        }

        // Step 2: If class_id exists, perform the update

```

```

        PreparedStatement stmt = conn.prepareStatement(updateSql);

        stmt.setString(1, className);

        stmt.setString(2, time);

        stmt.setInt(3, classId);

        int rowsUpdated = stmt.executeUpdate();

        // If the update was successful, return true

        return rowsUpdated > 0;

    } catch (SQLException e) {

        e.printStackTrace();

        return false; // Return false in case of an exception

    }

}

public boolean deleteClass(int classId) {

    String checkSql = "SELECT class_id FROM Classes__ WHERE class_id = ?";

    String deleteSql = "DELETE FROM Classes__ WHERE class_id = ?";

    try ( Connection conn = DataBaseConnection.getInstance().getCon()) {

        // Step 1: Check if the class_id exists in the database

        PreparedStatement checkStmt = conn.prepareStatement(checkSql);

        checkStmt.setInt(1, classId);

        ResultSet rs = checkStmt.executeQuery();

        // If no class_id is found, return false

        if (!rs.next()) {

            return false; // Class ID does not exist

        }

        // Step 2: If class_id exists, proceed with the deletion

        PreparedStatement deleteStmt = conn.prepareStatement(deleteSql);

```

```

        deleteStmt.setInt(1, classId);

        int rowsDeleted = deleteStmt.executeUpdate();

        // If the class was deleted successfully, return true

        return rowsDeleted > 0;
    } catch (SQLException e) {

        e.printStackTrace();

        return false; // Return false in case of an error
    }
}

public boolean deleteClassByTrainerId(int trainerId) throws SQLException {

    String checkSql = "SELECT trainer_id FROM Classes__ WHERE trainer_id = ?"; // Check if trainer_id exists

    String deleteSql = "DELETE FROM Classes__ WHERE trainer_id = ?"; // Delete the class records for the given
trainer_id

    try ( Connection conn = DataBaseConnection.getInstance().getCon(); PreparedStatement checkStmt =
conn.prepareStatement(checkSql)) {

        // Step 1: Check if the trainer_id exists in the database

        checkStmt.setInt(1, trainerId);

        try ( ResultSet rs = checkStmt.executeQuery()) {

            if (!rs.next()) {

                // If trainer_id is not found, return false

                return false;

            }

        }

        // Step 2: If trainer_id exists, perform the deletion

        try ( PreparedStatement deleteStmt = conn.prepareStatement(deleteSql)) {

            deleteStmt.setInt(1, trainerId);

            int rowsDeleted = deleteStmt.executeUpdate();

            // If the deletion was successful (at least one row affected), return true

            return rowsDeleted > 0;
        }
    }
}

```

```

    }

    } catch (SQLException e) {

        e.printStackTrace();

        return false; // Return false if an error occurs

    }

}

public List<Classes> retrieveAllClasses() {

    List<Classes> classes = new ArrayList<>();

    String sql = "SELECT * FROM Classes__ ";

    try ( Connection conn = DataBaseConnection.getInstance().getCon(); PreparedStatement stmt =
conn.prepareStatement(sql); ResultSet rs = stmt.executeQuery()) {

        while (rs.next()) {

            int classId = rs.getInt("class_id");

            String className = rs.getString("class_name");

            String day = rs.getString("day");

            String time = rs.getString("time");

            int maxCapacity = rs.getInt("max_capacity");

            int trainerid = rs.getInt("trainer_id");

            Trianer t = new Trianer(" ", " ", " ", " ", " ", " ", " ", trainerid);

            // List<Member> m = retrieveAllMembers();

            classes.add(new Classes(className, day, time, classId, t, maxCapacity));

        }

    } catch (SQLException e) {

        e.printStackTrace();

    }

    return classes;

}

public void insertTrackProgress(int memberId, double weight, double height, String musclePercentage, String
fatPercentage) throws SQLException {

```



```
String sql = "INSERT INTO traProgres__ (member_id, weight, height, MusclePercentage, FatPercentage)
VALUES (?, ?, ?, ?, ?)";
```

```
try {

    conn = DataBaseConnection.getInstance().getCon();

    PreparedStatement stmt = conn.prepareStatement(sql);

    stmt.setInt(1, memberId);

    stmt.setDouble(2, weight);

    stmt.setDouble(3, height);

    stmt.setString(4, musclePercentage);

    stmt.setString(5, fatPercentage);

    stmt.executeUpdate();

} catch (SQLException e) {

    e.printStackTrace();

}

}
```

```
public boolean deleteTrackProgress(int id) throws SQLException {
```

```
String checkSql = "SELECT member_id FROM traProgres__ WHERE member_id = ?";
```

```
String deleteSql = "DELETE FROM traProgres__ WHERE member_id = ?";
```

```
try ( Connection conn = DataBaseConnection.getInstance().getCon(); PreparedStatement checkStmt =
conn.prepareStatement(checkSql)) {
```

```
    // Check if the member_id exists
```

```
    checkStmt.setInt(1, id);
```

```
    try ( ResultSet rs = checkStmt.executeQuery()) {
```

```
        if (!rs.next()) {
```

```
            // If member_id is not found, return false
```

```
            return false;
```

```

    }

}

//If member_id exists

try ( PreparedStatement deleteStmt = conn.prepareStatement(deleteSql)) {

    deleteStmt.setInt(1, id);

    int rowsDeleted = deleteStmt.executeUpdate();

    return rowsDeleted > 0;

}

} catch (SQLException e) {

    e.printStackTrace();

    return false; // Return false if an exception occurs

}

}

public void insertOffer(String coupon, String occasion, String discount, String expireDate) throws SQLException {

    String sql = "INSERT INTO Offer (coupon, occasion, discount, expireDate) VALUES (?, ?, ?, ?)";

    PreparedStatement stmt = null;

    conn = DataBaseConnection.getInstance().getCon();

    try {

        stmt = conn.prepareStatement(sql);

        stmt.setString(1, coupon);

        stmt.setString(2, occasion);

        stmt.setString(3, discount);

        stmt.setString(4, expireDate);

        stmt.executeUpdate();

    }

```

```

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        closeStatement(stmt);

    }

}

public List<offer> getAllOffers() throws SQLException {

    String sql = "SELECT * FROM Offer";

    List<offer> offers = new ArrayList<>();

    PreparedStatement stmt = null;

    ResultSet rs = null;

    try {

        conn = DataBaseConnection.getInstance().getCon();

        stmt = conn.prepareStatement(sql);

        rs = stmt.executeQuery();

        while (rs.next()) {

            offers.add(new offer(

                rs.getString("coupon"),

                rs.getString("occasion"),

                rs.getString("discount"),

                rs.getString("expireDate")

            ));

        }

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        if (rs != null) {

```

```

        rs.close();
    }

    closeStatement(stmt);
}

return offers;
}

//Service

public void insertService(String serviceName) {

    String sql = "INSERT INTO services (service_name) VALUES (?)";

    PreparedStatement stmt = null;

    try {

        conn = DataBaseConnection.getInstance().getCon();

        stmt = conn.prepareStatement(sql);

        stmt.setString(1, serviceName);

        stmt.executeUpdate();

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        closeStatement(stmt);

    }

}

public void deleteService(String serviceName) throws SQLException {

    String sql = "DELETE FROM services WHERE service_name = ?";

    try ( PreparedStatement stmt = DataBaseConnection.getInstance().getCon().prepareStatement(sql)) {

        stmt.setString(1, serviceName);

        stmt.executeUpdate();

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

```

```

    }
}

public List<Service> getAllServices() throws SQLException {

    String sql = "SELECT * FROM services";

    List<Service> services = new ArrayList<>();

    PreparedStatement stmt = null;

    ResultSet rs = null;

    try {

        conn = DataBaseConnection.getInstance().getCon();

        stmt = conn.prepareStatement(sql);

        rs = stmt.executeQuery();

        while (rs.next()) {

            services.add(new Service(rs.getString("service_name")));

        }

    } catch (SQLException e) {

        e.printStackTrace();

    } finally {

        rs.close();

        closeStatement(stmt);

    }

    return services;

}

private void closeResultSet(ResultSet rs) {

    if (rs != null) {

        try {

            rs.close();

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

}

```

```
        }  
    }  
}  
  
private void closeStatement(PreparedStatement stmt) {  
    if (stmt != null) {  
        try {  
            stmt.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}  
  
}
```

## 6. Appendix

```
package uml;
```

```
//=====
```

### UML Classes

```
public abstract class User {
```

```
    private String fullName, PhoneNumber, DateofBirth, Email, PassWord, Role;
```

```
    private int userID;
```

```
    public User(String fullName, String PhoneNumber, String DateofBirth, String Email, String PassWord, String Role, int userID) {
```

```
        this.fullName = fullName;
```

```
        this.PhoneNumber = PhoneNumber;
```

```
        this.DateofBirth = DateofBirth;
```

```
        this.Email = Email;
```

```
        this.PassWord = PassWord;
```

```
        this.Role = Role;
```

```
        this.userID = userID;
```

```
    }
```

```
    public String getFullName() {
```

```
        return fullName;
```

```
    }
```

```
    public void setFullName(String fullName) {
```

```
        this.fullName = fullName;
    }

    public String getPhoneNumber() {
        return PhoneNumber;
    }

    public void setPhoneNumber(String PhoneNumber) {
        this.PhoneNumber = PhoneNumber;
    }

    public String getDateofBirth() {
        return DateofBirth;
    }

    public void setDateofBirth(String DateofBirth) {
        this.DateofBirth = DateofBirth;
    }

    public String getEmail() {
        return Email;
    }

    public void setEmail(String Email) {
        this.Email = Email;
    }
}
```



```
public String getPassWord() {  
    return PassWord;  
}
```

```
public void setPassWord(String PassWord) {  
    this.PassWord = PassWord;  
}
```

```
public String getRole() {  
    return Role;  
}
```

```
public void setRole(String Role) {  
    this.Role = Role;  
}
```

```
public int getUserID() {  
    return userID;  
}
```

```
public void setUserID(int userID) {  
    this.userID = userID;  
}
```

```
@Override
```

```
public String toString() {
```

```

        return "User{" + "fullName=" + fullName + ", PhoneNumber=" + PhoneNumber + ", DateofBirth=" +
DateofBirth + ", Email=" + Email + ", PassWord=" + PassWord + ", Role=" + Role + ", userID=" + userID +
'}';

    }

}

//=====

import java.util.List;

public class Admin extends User implements Manageable{

    public Admin(String fullName, String PhoneNumber, String DateofBirth, String Email, String PassWord,
String Role, int userID) {

        super(fullName, PhoneNumber, DateofBirth, Email, PassWord, Role, userID);

    }

    @Override

    public void addUser(User user) {

        System.out.println("Adding user: " + user.getFullName());

    }

    @Override

    public void removeUser(User user) {

        System.out.println("Removing user: " + user.getFullName() );

    }

    @Override

    public void updateUser(User user) {

        System.out.println("Updating user: " + user.getFullName() );

    }

}

```

```

public void displayMenu() {

    System.out.println("1. Add User");

    System.out.println("2. Remove User");

    System.out.println("3. Update User");

}

}

//=====

public class Member extends User {

    public Member(String fullName, String PhoneNumber, String DateofBirth, String Email, String
    PassWord, String Role, int userID) {

        super(fullName, PhoneNumber, DateofBirth, Email, PassWord, Role, userID);

    }

    @Override

    public String toString() {

        return "Member{" + super.toString() + '}';

    }

}

//=====

public interface Manageable {

    void addUser(User user);

    void removeUser(User user);

    void updateUser(User user);

}

//=====

import java.util.List;

```

```

public class Classes {

    private String className, day, time;

    private int class_id;

    private Trianer trianer;

    private int max_capacity;

    public Classes() {

    }

    public Classes(String className, String day, String time, int class_id, Trianer trianer, int max_capacity)
    {

        this.className = className;

        this.day = day;

        this.time = time;

        this.class_id = class_id;

        this.trianer = trianer;


        this.max_capacity = max_capacity;

    }


    public int getClass_id() {

        return class_id;

    }


    public void setClass_id(int class_id) {

        this.class_id = class_id;

    }


    public int getMax_capacity() {

```

```
        return max_capacity;
    }

    public void setMax_capacity(int max_capacity) {
        this.max_capacity = max_capacity;
    }

    public String getClassName() {
        return className;
    }

    public void setClassName(String className) {
        this.className = className;
    }

    public String getDay() {
        return day;
    }

    public void setDay(String day) {
        this.day = day;
    }

    public String getTime() {
        return time;
    }
}
```

```

    public void setTime(String time) {
        this.time = time;
    }

    public Trianer getTrianer() {
        return trianer;
    }

    public void setTrianer(Trianer trianer) {
        this.trianer = trianer;
    }

    @Override
    public String toString() {
        return "className: " + className + " (" + time + "PM)";
    }

}

//=====

public class Service {

    private String serviceName;

    public Service(String serviceName) {
        this.serviceName = serviceName;
    }

```

```
}
```

```
public String getServiceName() {
```

```
    return serviceName;
```

```
}
```

```
public void setServiceName(String serviceName) {
```

```
    this.serviceName = serviceName;
```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    return serviceName ;
```

```
}
```

```
}
```

```
//=====
```

```
public class TrackProgress {
```

```
    private Member memberID;
```

```
    double weight, height;
```

```
    private String MusclePercentage , FatPercentage;
```

```
public TrackProgress(Member memberID, double weight, double height, String MusclePercentage,
String FatPercentage) {
```

```
    this.memberID = memberID;
```

```
    this.weight = weight;
```

```
    this.height = height;
```

```
    this.MusclePercentage = MusclePercentage;
```

```
    this.FatPercentage = FatPercentage;
```

```
}
```

```
public Member getMemberID() {
```

```
    return memberID;
```

```
}
```

```
public void setMemberID(Member memberID) {
```

```
    this.memberID = memberID;
```

```
}
```

```
public double getWeight() {
```

```
    return weight;
```

```
}
```

```
public void setWeight(double weight) {
```

```
    this.weight = weight;
```

```
}
```

```
public double getHeight() {
```



```

        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    public String getMusclePercentage() {
        return MusclePercentage;
    }

    public void setMusclePercentage(String MusclePercentage) {
        this.MusclePercentage = MusclePercentage;
    }

    public String getFatPercentage() {
        return FatPercentage;
    }

    public void setFatPercentage(String FatPercentage) {
        this.FatPercentage = FatPercentage;
    }

    @Override
    public String toString() {
        return "TrackProgress{" + "memberID=" + memberID + ", weight=" + weight + ", height=" + height +
            ", MusclePercentage=" + MusclePercentage + ", FatPercentage=" + FatPercentage + "}";
    }

```

```

    }

}

//=====

public class offer {

    private String coupon ,occison , Discount , expireDate;

    public offer(String coupon, String occison, String Discount, String expireDate) {

        this.coupon = coupon;

        this.occison = occison;

        this.Discount = Discount;

        this.expireDate = expireDate;

    }

    public String getCoupon() {

        return coupon;

    }

    public void setCoupon(String coupon) {

        this.coupon = coupon;

    }

    public String getOccison() {

        return occison;

```

```
}
```

```
public void setOccison(String occison) {
```

```
    this.occison = occison;
```

```
}
```

```
public String getDiscount() {
```

```
    return Discount;
```

```
}
```

```
public void setDiscount(String Discount) {
```

```
    this.Discount = Discount;
```

```
}
```

```
public String getExpireDate() {
```

```
    return expireDate;
```

```
}
```

```
public void setExpireDate(String expireDate) {
```

```
    this.expireDate = expireDate;
```

```
}
```

```
@Override
```

```
public String toString() {
```

```
    return "occison:" + occison + ", " + expireDate;
```

```
}
```

```
}
```

//=====

**Scema Tables:**

```
CREATE TABLE useerss_ (  
    user_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  
    password VARCHAR(100),  
    email VARCHAR(100) UNIQUE,  
    full_name VARCHAR(100),  
    date_of_birth DATE,  
    phone_number VARCHAR(20),  
    role VARCHAR(20)  
);
```

```
CREATE TABLE Classes__ (  
    class_id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  
    class_name VARCHAR(20) NOT NULL,  
    day VARCHAR(100),  
    time VARCHAR(100),  
    max_capacity NUMBER(2),  
    trainer_id NUMBER,  
    FOREIGN KEY (trainer_id) REFERENCES useerss_(user_id)  
);
```

```
CREATE TABLE services (  
    service_name VARCHAR(20)
```

```
);
```

```
CREATE TABLE Offer (
```

```
    coupon VARCHAR2(50),
```

```
    occasion VARCHAR2(100),
```

```
    discount VARCHAR2(10),
```

```
    expireDate VARCHAR2(10)
```

```
);
```

```
CREATE TABLE traProgres__ (
```

```
    member_id NUMBER NOT NULL,
```

```
    weight NUMBER(5, 2),
```

```
    height NUMBER(5, 2),
```

```
    MusclePercentage VARCHAR2(10),
```

```
    FatPercentage VARCHAR2(10),
```

```
    FOREIGN KEY (member_id) REFERENCES useerss_(user_id)
```

```
);
```

```
//===== gymproject package=====
```

```
import java.io.IOException;
```

```
import javafx.application.Application;
```

```
import javafx.fxml.FXMLLoader;
```

```
import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
```

```
import javafx.scene.control.Alert;
```

```
import javafx.scene.control.ButtonType;
```

```
import javafx.stage.Stage;
```

```
public class GymProject extends Application {
```

```
    private static Stage primeryStage;
```

```
    @Override
```

```
    public void start(Stage stage) throws Exception {
```

```

primaryStage = stage;

loginView();

stage.setOnCloseRequest(event -> {

    event.consume();

    handelClose("There is some unsaved info.. ");

});

}

// ----- LOGIN -----

public static void loginView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("login.fxml"));

        Scene scene = new Scene(root);

        primaryStage.setScene(scene);

        primaryStage.show();

        primaryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("If you exit, all info will be deleted.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

```

```
//----- MEMBER CONTROLLES -----

public static void memberView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("memberView.fxml"));

        System.out.println("22");

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("If you exit, all info will be deleted.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

public static void groupClassView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("groupClasses.fxml"));

        Scene scene = new Scene(root);
```



```

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("If you exit, all info will be deleted.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

public static void MemberEditInfoView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("MemberEditInfo.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    } catch (IOException ex) {

        System.out.println(ex);

```

```

        System.out.println("invaled");
    }
}

//----- TRAINER CONTROLLES -----

public static void trainerView() {
    Parent root;

    try {
        root = FXMLLoader.load(GymProject.class.getResource("TrainerView.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);
        primeryStage.show();
        primeryStage.setOnCloseRequest(event -> {
            event.consume();
            handelClose("There is some unsaved info.. ");

        });
    } catch (IOException ex) {
        System.out.println("invaled");
    }
}

public static void editClassView() {
    Parent root;

    try {

```

```

root = FXMLLoader.load(GymProject.class.getResource("EditClass.fxml"));

Scene scene = new Scene(root);

primeryStage.setScene(scene);
primeryStage.show();
primeryStage.setOnCloseRequest(event -> {
    event.consume();
    handelClose("There is some unsaved info.. ");

});
} catch (IOException ex) {
    System.out.println("invaled");
}
}

public static void RemoveClassView() {
    Parent root;

    try {
        root = FXMLLoader.load(GymProject.class.getResource("RemoveClass.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);
        primeryStage.show();
        primeryStage.setOnCloseRequest(event -> {
            event.consume();

```

```

        handelClose("There is some unsaved info.. ");

    });

} catch (IOException ex) {

    System.out.println("invaled");

}

}

public static void addSeccionView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("addSession.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

```

```

public static void trackProgressView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("trackProgress.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

//----- ADMINISTATOR CONTROLLES -----

public static void administratorView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("administratorView.fxml"));

        Scene scene = new Scene(root);

```

```

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

//1

public static void UsersManagementView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("UsersManagement.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    }

```

```

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

```

//2

```

public static void manageGymServiceView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("manageGymServices.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

        primeryStage.show();

        primeryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    } catch (IOException ex) {

        System.out.println("invaled");

    }

}

```

//3

```

public static void offersView() {

```

```

Parent root;

try {

    root = FXMLLoader.load(GymProject.class.getResource("offers.fxml"));

    Scene scene = new Scene(root);

    primeryStage.setScene(scene);

    primeryStage.show();

    primeryStage.setOnCloseRequest(event -> {

        event.consume();

        handelClose("There is some unsaved info.. ");

    });

} catch (IOException ex) {

    System.out.println("invaled");

}

}

//1.1

public static void AddUserView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("addUser.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);

```



```

primaryStage.show();

primaryStage.setOnCloseRequest(event -> {

    event.consume();

    handelClose("There is some unsaved info.. ");

});

} catch (IOException ex) {

    System.out.println("invaled");

}

}

//1.2

public static void RemoveUserView() {

    Parent root;

    try {

        root = FXMLLoader.load(GymProject.class.getResource("removeUser.fxml"));

        Scene scene = new Scene(root);

        primaryStage.setScene(scene);

        primaryStage.show();

        primaryStage.setOnCloseRequest(event -> {

            event.consume();

            handelClose("There is some unsaved info.. ");

        });

    } catch (IOException ex) {

```

```

        System.out.println("invaled");
    }
}

//1.3
public static void UpdateUserView() {
    Parent root;

    try {
        root = FXMLLoader.load(GymProject.class.getResource("updateUser.fxml"));

        Scene scene = new Scene(root);

        primeryStage.setScene(scene);
        primeryStage.show();
        primeryStage.setOnCloseRequest(event -> {
            event.consume();
            handelClose("There is some unsaved info.. ");

        });
    } catch (IOException ex) {
        System.out.println("invaled");
    }
}

//-----

private static void handelClose(String massage) {

    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

```

```

    alert.setTitle("Close App");

    alert.setHeaderText("Are you sure you want to close the App?");

    alert.setContentText(message);


    if (alert.showAndWait().get() == ButtonType.OK) {

        primaryStage.close();

    }

}

public static void main(String[] args) {

    launch(args);

}

}

//=====CSS style class=====

/* Style for all Text shapes */

.Root {

    -fx-fill: #7d8a6b;          /* Text color */

    -fx-font-size: 30px;        /* Font size */

    -fx-font-weight: bold;      /* Font weight */

    -fx-font-family: "Arial";    /* Font family */

    -fx-effect: dropshadow(one-pass-box, rgba(0, 0, 0, 0.2), 2, 0.0, 1, 1); /* Light shadow effect */

}

/* Style for all Button controls */

.button {

```

```

-fx-background-color: #ced9bf; /* Button background color */

-fx-text-fill: black; /* Text color */

-fx-font-weight: bold;

-fx-pref-width: 250px; /* Preferred width */

-fx-pref-height: 30px; /* Preferred height */

-fx-background-radius: 8px; /* Slightly curved edges */

-fx-font-size: 14px; /* Adjust font size as needed */

-fx-effect: dropshadow(one-pass-box, rgba(0, 0, 0, 0.4), 2, 0.5, 1, 1); /* shadow effect */

}

```

```

.button_Back{

-fx-background-color: #e0e8d5; /* Button background color */

-fx-text-fill: black; /* Text color */

-fx-font-weight: bold;

-fx-pref-width: 25px; /* Preferred width */

-fx-pref-height: 25px; /* Preferred height */

-fx-background-radius: 10px; /* Slightly curved edges */

-fx-font-size: 12px; /* Adjust font size as needed */

}

```

```

.button_event{

-fx-background-color: #e2f0ce; /* Button background color */

-fx-text-fill: black; /* Text color */

-fx-font-weight: bold;

-fx-pref-width: 70px; /* Preferred width */

-fx-pref-height: 25px; /* Preferred height */

}

```

```

    -fx-background-radius: 10px;    /* Slightly curved edges */

    -fx-font-size: 12px;          /* Adjust font size as needed */
}

/* Style for all Label controls */
.label_event {

    -fx-text-fill: #7a8768;    /* Text color */

    -fx-font-weight: bold;    /* Bold font */

    -fx-font-size: 12px;    /* Font size */
}

.button_logout{

    -fx-background-color: #e0e8d5;    /* Button background color */

    -fx-text-fill: black;    /* Text color */

    -fx-pref-width: 70px;    /* Preferred width */

    -fx-pref-height: 25px;    /* Preferred height */

    -fx-background-radius: 10px;    /* Slightly curved edges */

    -fx-font-size: 12px;    /* Adjust font size as needed */
}

.combo-box {

    -fx-background-color: #bec9ad; /* Light green background */

    -fx-text-fill: #000000; /* Black text color */

    -fx-font-weight: bold;

    -fx-pref-width: 250px;    /* Preferred width */

    -fx-pref-height: 30px;    /* Preferred height */

    -fx-background-radius: 8px;    /* Slightly curved edges */

```

```
-fx-font-size: 14px;  
}
```

```
.make_blod{  
    -fx-font-weight: bold;  
}
```

```
import database.handler;  
import java.sql.SQLException;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import javafx.scene.control.TextField;  
import uml.User;
```

```
public class AddSessionController {
```

```
    @FXML
```

```
    private TextField nametxt;
```

```
    @FXML
```

```
    private TextField daytxt;
```

```
    @FXML
```

```
    private TextField timetxt;
```

```
    @FXML
```

```
    private TextField seatstxt;
```

@FXML

private TextField idtxt;

@FXML

private Label lbl;

private handler handler = new handler();

@FXML

private void handelAdd(ActionEvent event) throws SQLException {

String name = nametxt.getText();

String day = daytxt.getText();

String time = timetxt.getText();

String seats = seatstxt.getText();

String id = idtxt.getText();

if (!name.isEmpty() && !day.isEmpty() && !time.isEmpty() && !seats.isEmpty()) {

try {

int seat = Integer.parseInt(seats);

int Tid = Integer.parseInt(id);

User user = handler.getUserById(Tid);

if (user == null) {

lbl.setText("Trainer with this ID does not exist.");

idtxt.clear();

} else if (!user.getRole().equals("TRAINER")) {

lbl.setText("This ID does not have trainer privileges!");

```

        idtxt.clear();

    } else {

        handler.insertClass(name, day, time, seat, Tid);

        lbl.setText("Class added successfully");

    }

} catch (NumberFormatException e) {

    lbl.setText("Invalid input for seats or ID. Please enter numbers.");

} catch (SQLException e) {

    lbl.setText("Error adding class: " + e.getMessage());

    e.printStackTrace();

}

} else {

    lbl.setText("Some fields are empty.");

}

}

```

@FXML

```

private void back(ActionEvent event) {

    GymProject.trainerView();

}

```

```

}

```

```

//=====

```

```

import database.handler;

```

```

import java.sql.Date;

```

```

import java.sql.SQLException;

```



```
import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.RadioButton;

import javafx.scene.control.TextField;
```

```
public class AddUserController {
```

```
    @FXML
```

```
    private TextField nametxt;
```

```
    @FXML
```

```
    private TextField DOBtxt;
```

```
    @FXML
```

```
    private TextField phoneNumtxt;
```

```
    @FXML
```

```
    private TextField emailtxt;
```

```
    @FXML
```

```
    private PasswordField passwordtxt;
```

```
    @FXML
```

```
    private RadioButton member;
```

```
    @FXML
```

```
    private RadioButton trainer;
```

```
    @FXML
```

```
    private RadioButton admin;
```

@FXML

private Label lbl;

private handler handler = new handler();

@FXML

private void handelConfirm(ActionEvent event) throws SQLException {

String nameVal = nametxt.getText();

String passwordVal = passwordtxt.getText();

String emailVal = emailtxt.getText();

String dateOfBirthVal = DOBtxt.getText();

String phoneNumberVal = phoneNumtxt.getText();

if (!nameVal.isEmpty()

&& !dateOfBirthVal.isEmpty() && !phoneNumberVal.isEmpty()

&& !emailVal.isEmpty() && !passwordVal.isEmpty()) {

String role = "";

if (member.isSelected()) {

role = "MEMBER";

} else if (trainer.isSelected()) {

role = "TRAINER";

} else if (admin.isSelected()) {

```

        role = "ADMIN";
    }

    try {
        handler.insertUser( passwordVal, emailVal, nameVal, Date.valueOf(dateOfBirthVal),
        phoneNumberVal, role);

        lbl.setText("User added successfully");

        nameTxt.clear();

        passwordTxt.clear();

        emailTxt.clear();

        DOBTxt.clear();

        phoneNumTxt.clear();
    } catch (SQLException ex) {

        lbl.setText("Error:" + ex);

    }

    } else {

        lbl.setText("Some field is empty");

    }

}

@FXML

private void back(ActionEvent event) {

    GymProject.UsersManagementView();

}

}

```

```
//=====
```

```
import javafx.event.ActionEvent;
```

```
import javafx.fxml.FXML;
```

```
public class AdministratorViewController {
```

```
    @FXML
```

```
    private void handelUsersManagement(ActionEvent event){
```

```
        GymProject.UsersManagementView();
```

```
    }
```

```
    @FXML
```

```
    private void handelManageGymServicess(ActionEvent event){
```

```
        GymProject.manageGymServicessView();
```

```
    }
```

```
    @FXML
```

```
    private void handelOffers(ActionEvent event){
```

```
        GymProject.offersView();
```

```
    }
```

```
    @FXML
```

```
    private void back(ActionEvent event){
```

```
        GymProject.loginView();
```

```
    }
```

```
}
```

```
//=====

import database.handler;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;


public class EditClassController {


    @FXML

    private Label confirmMessage;


    @FXML

    private TextField txtfld1;


    @FXML

    private TextField txtfld2;


    @FXML

    private TextField txtID;


    private handler handler = new handler();


    @FXML

    private void handelConfirm(ActionEvent event) {

        String id = txtID.getText();

        String name = txtfld1.getText();
```

```

String time = txtfld2.getText();

if (!name.isEmpty() && !time.isEmpty() && !id.isEmpty()) {

    int userId = Integer.parseInt(id);

    Boolean check = handler.updateClass(userId, name, time);

    if (check.equals(true)) {

        confirmMessage.setText("Class is updated.");

    } else {

        confirmMessage.setText("invalid ID");

        txtID.clear();

        txtfld1.clear();

        txtfld2.clear();

    }

} else {

    confirmMessage.setText("some felid empty");

}

}

@FXML

private void back(ActionEvent event) {

    GymProject.trainerView();

}

}

//=====

import database.handler;

```

```

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import uml.Classes;


public class GroupClassesController {


    @FXML

    private ComboBox<String> classComboBox;


    @FXML

    private Label lbl;


    private handler handler = new handler();


    public static List<String> memberClass;


    public void showClass() throws SQLException {

        List<Classes> list = handler.retrieveAllClasses();

        for (Classes ele : list) {

            classComboBox.getItems().add(ele.toString());

        }

    }
}

```

@FXML

```
private void handleJoin(ActionEvent event) {
```

```
    String selectedClass = classComboBox.getSelectionModel().getSelectedItem(); // Get the selected service
```

```
    memberClass = new ArrayList<>();
```

```
    if (selectedClass != null) {
```

```
        memberClass.add(selectedClass);
```

```
        lbl.setText("You join a class");
```

```
    } else {
```

```
        lbl.setText("Please select a class");
```

```
    }
```

```
}
```

@FXML

```
private void back(ActionEvent event) {
```

```
    GymProject.memberView();
```

```
}
```

@FXML

```
public void initialize() throws SQLException {
```

```
    showClass();
```

```
}
```

```
}
```

```
//=====
```



```

import database.handler;

import java.sql.SQLException;

import java.util.List;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import uml.Service;


public class ManageGymServicesController {


    @FXML

    private ComboBox<String> servicesComboBox;


    @FXML

    private Label addlbl;

    @FXML

    private Label removelbl;

    @FXML

    private Label lbl;


    @FXML

    private TextField addServicetxt;

```

```

private ObservableList<String> gymServices;

private handler handler = new handler();

@FXML

public void initialize() throws SQLException{

    List<Service> list = handler.getAllServices();

    for (Service ele : list) {

        servicesComboBox.getItems().add(ele.toString());

    }

}

```

```

@FXML

private void handleAddService(ActionEvent event) {

    lbl.setText("");

    String newService = addServiceTxt.getText();

    if (!newService.isEmpty()) {

        handler.insertService(newService);

        servicesComboBox.getItems().add(newService);

        addServiceTxt.clear();

        lbl.setText("Added ");

    } else {

        lbl.setText("Please enter a service to add.");

    }

}

```

@FXML

```
private void handleRemoveService(ActionEvent event) throws SQLException {
```

```
    lbl.setText("");
```

```
    String selectedService = servicesComboBox.getSelectionModel().getSelectedItem(); // Get the  
    selected service
```

```
    if (selectedService != null) {
```

```
        handler.deleteService(selectedService);
```

```
        int index = servicesComboBox.getSelectionModel().getSelectedIndex();
```

```
        servicesComboBox.getItems().remove(index);
```

```
        lbl.setText("Removed");
```

```
    } else {
```

```
        lbl.setText("Please select a service to remove.");
```

```
    }
```

```
}
```

@FXML

```
private void back(ActionEvent event){
```

```
    GymProject.administratorView();
```

```
}
```

```
}
```

```
//=====
```

```
import database.SessionInfo;
```

```
import database.handler;
```

```
import java.net.URL;
```

```

import java.sql.Date;

import java.sql.SQLException;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import uml.User;


public class MemberEditInfoController implements Initializable {


    @FXML

    private TextField nametxt;

    @FXML

    private TextField phoneNumtxt;

    @FXML

    private TextField emailtxt;

    @FXML

    private PasswordField passwordtxt;

    @FXML

    private TextField DOBtxt;


    @FXML

    private Label savelbl;

```

```

private handler handler = new handler();

@FXML

private void handelSaveChanges(ActionEvent event) throws SQLException {

    String nameVal = nametxt.getText();

    String passwordVal = passwordtxt.getText();

    String emailVal = emailtxt.getText();

    String dateOfBirthVal = DOBtxt.getText();

    String phoneNumberVal = phoneNumtxt.getText();

    User user = SessionInfo.getInstance().getCurrentUser();

    int id = user.getUserID();

    if (!nameVal.isEmpty() && !dateOfBirthVal.isEmpty() && !passwordVal.isEmpty()

        && !emailVal.isEmpty() && !phoneNumberVal.isEmpty()) {

        handler.updateUser( id, passwordVal, emailVal, nameVal, Date.valueOf(dateOfBirthVal),
phoneNumberVal, "MEMBER");

        savelbl.setText("Saved Successfully");

    } else {

        savelbl.setText("Some field is empty");

    }

}

```

@FXML

```
private void back(ActionEvent event) {
```

```
    GymProject.memberView();
```

```
}
```

@Override

```
public void initialize(URL location, ResourceBundle resources) {
```

```
    try {
```

```
        User user = SessionInfo.getInstance().getCurrentUser();
```

```
        int userId = user.getUserID();
```

```
        nametxt.setText(user.getFullName());
```

```
        DOBtxt.setText(user.getDateOfBirth());
```

```
        phoneNumtxt.setText(user.getPhoneNumber());
```

```
        emailtxt.setText(user.getEmail());
```

```
        passwordtxt.setText(user.getPassWord());
```

```
    } catch (Exception ex) {
```

```
        System.out.println("!!");
```

```
    }
```

```
}
```

```
}
```

```
//=====
```

```
import database.SessionInfo;
```

```
import database.handler;
```

```
import java.sql.SQLException;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.text.Text;

import uml.User;

import uml.offer;


public class MemberViewController {


    @FXML

    private Label memberNumlbl;

    @FXML

    private Label nameLbl;


    @FXML

    private ComboBox offerCompo;


    @FXML

    private Text classtxt;


    private handler handler = new handler();


    private void upcomingClass() {

        List<String> list = GroupClassesController.memberClass;

        if(list == null){

```

```

list = new ArrayList<>();

}

if(!list.isEmpty())
for (String ele : list) {

    classtxt.setText(ele + "\n");

}else{

    classtxt.setText("You didn't schedule anything");

}

}

public void showOffer() throws SQLException {

    List<offer> list = handler.getAllOffers();

    for (offer ele : list) {

        oferCompo.getItems().add(ele.toString());

    }

    System.out.println("end showOffer");

}

@FXML

private void handleGroupClass(ActionEvent event) {

    GymProject.groupClassView();

}

@FXML

```



```

private void handleEditProfile(ActionEvent event) {

    GymProject.MemberEditInfoView();

}

@FXML

public void initialize() {

    try {

        showOffer();

        upcomingClass();

        User user = SessionInfo.getInstance().getCurrentUser();

        int userId = user.getUserID();

        memberNumlbl.setText(String.valueOf(userId));

        namelbl.setText(user.getFullName());

    } catch (SQLException ex) {

        System.out.println("Exception");

    }

}

@FXML

private void back(ActionEvent event) {

    GymProject.loginView();

}

}

//=====

import database.handler;

```

```
import java.sql.SQLException;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;


public class OffersController {


    @FXML

    private TextField coupontxt;


    @FXML

    private TextField occasiontxt;


    @FXML

    private TextField discounttxt;


    @FXML

    private TextField ExpirDate;


    @FXML

    private Label resultlbl;


    private handler handler = new handler();


    @FXML
```

```

private void handleSubmit() throws SQLException {

    String couponval = coupontxt.getText();

    String occasion = occasiontxt.getText();

    String discount = discounttxt.getText();

    String ExpirDatee = ExpirDate.getText();


    if (!coupontxt.getText().isEmpty() && !occasiontxt.getText().isEmpty() &&
        !discounttxt.getText().isEmpty() && !ExpirDate.getText().isEmpty()) {

        handler.insertOffer(couponval, occasion, discount, ExpirDatee);

        resultlbl.setText("Discount " + discounttxt.getText() + "% is added successfully.");

    } else {

        resultlbl.setText("Some field is empty");

    }

}

```

@FXML

```

private void back(ActionEvent event){

    GymProject.administratorView();

}

}

//=====

import database.handler;

import java.sql.SQLException;

import javafx.event.ActionEvent;

```

```

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;


public class RemoveClassController {


    @FXML

    private Label removeLbl;


    @FXML

    private TextField IDtxt;


    private handler handler = new handler();


    @FXML

    private void handelRemove(ActionEvent event) throws SQLException {

        try {

            if (IDtxt.getText().isEmpty()) {

                removeLbl.setText("Enter the ID");

            } else {

                int userId = Integer.parseInt(IDtxt.getText());

                Boolean check = handler.deleteClass(userId);

                if(check.equals(true)){

                    IDtxt.clear();

                    removeLbl.setText("Removed Successfully");

                }else{

                    removeLbl.setText("invalid ID");

```

```

        IDtxt.clear();
    }

}

} catch (NumberFormatException ex) {
    removeLbl.setText("Invalid ID format");
    ex.printStackTrace();
} catch (Exception ex) {
    ex.printStackTrace();
}
}

@FXML

private void back(ActionEvent event) {
    GymProject.trainerView();
}

}

//=====

import database.handler;
import java.sql.SQLException;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;

```

```

import uml.User;

public class RemoveUserController {

    @FXML

    private Label removeLbl;

    @FXML

    private TextField IDtxt;

    private handler handler = new handler();

    @FXML

    private void handelRemove(ActionEvent event) throws SQLException {

        try {

            if (IDtxt.getText().isEmpty()) {

                removeLbl.setText("Enter the ID");

            } else {

                int userId = Integer.parseInt(IDtxt.getText());

                User user = handler.getUserById(userId);

                if (user != null) {

                    //=====Alert

                    Alert alert = new Alert(Alert.AlertType.WARNING);

                    alert.setTitle("Remove User");

                    alert.setHeaderText("ALL THE INFORMATION FOR THIS USER WILL BE REMOVED!");

```

```

        alert.setContentText("This User will be removed.");

        if (alert.showAndWait().get() == ButtonType.OK) {

            handler.deleteTrackProgress(userId);

            handler.deleteClassByTrainerId(userId);

            handler.deleteUser(userId);

            IDtxt.clear();

            removebl.setText("Removed Successfully");

            System.out.println("User has been removed");

        }

        //=====

    } else {

        System.out.println("No such user.");

        removebl.setText("Invalid ID");

    }

}

} catch (NumberFormatException ex) {

    removebl.setText("Invalid ID format");

    ex.printStackTrace();

} catch (SQLException ex) {

    ex.printStackTrace();

} catch (Exception ex) {

    ex.printStackTrace();

}

}

```

```

        private void back(ActionEvent event
    ) {
        GymProject.UsersManagementView();
    }

}

//=====

import database.handler;

import java.net.URL;

import java.sql.SQLException;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import uml.User;

public class TrackProgressController implements Initializable {

    @FXML

    private Label invalid;

    @FXML

    private Label result;

    @FXML

```



```
private Label submit;
```

```
@FXML
```

```
private TextField ID;
```

```
@FXML
```

```
private TextField weight;
```

```
@FXML
```

```
private TextField height;
```

```
@FXML
```

```
private TextField MusclePercentage;
```

```
@FXML
```

```
private TextField FatPercentage;
```

```
private handler handler = new handler();
```

```
@FXML
```

```
private void handelShow(ActionEvent event) {
```

```
    try {
```

```
        if (ID.getText().isEmpty()) {
```

```
            invalid.setText("Enter the ID");
```

```
        } else {
```

```
            int userId = Integer.parseInt(ID.getText());
```

```
            User user = handler.getUserById(userId);
```

```
            if (user != null) {
```

```
                weight.setEditable(true);
```

```
        height.setEditable(true);

        MusclePercentage.setEditable(true);

        FatPercentage.setEditable(true);


        invalid.setText("Correct ID");


    } else {

        weight.setEditable(false);

        height.setEditable(false);

        MusclePercentage.setEditable(false);

        FatPercentage.setEditable(false);

        System.out.println("No such user.");

        invalid.setText("Invalid ID");

    }

}

} catch (NumberFormatException ex) {

    invalid.setText("Invalid ID format");

    ex.printStackTrace();

} catch (SQLException ex) {

    ex.printStackTrace();

} catch (Exception ex) {

    ex.printStackTrace();

}

}
```

@FXML

```
private void handelSubmit(ActionEvent event) throws SQLException {

    String FatPercentageval = FatPercentage.getText();

    String MusclePercentageval = MusclePercentage.getText();

    String weightval = weight.getText();

    String heightval = height.getText();

    String idt = ID.getText();

    if (!FatPercentage.getText().isEmpty() && !MusclePercentage.getText().isEmpty()

        && !weight.getText().isEmpty() && !height.getText().isEmpty()) {

        int id = Integer.parseInt(idt);

        User user = handler.getUserById(id);

        if (user != null) {

            double w = Double.parseDouble(weightval);

            double h = Double.parseDouble(heightval);

            String mss = calc(w, h);

            handler.insertTrackProgress(id, w, h, MusclePercentageval, FatPercentageval);

            result.setText(mss);

            submit.setText("Added Successfully");

        } else {

            invalid.setText("invalid ID");

        }

    } else {

        submit.setText("Some field is empty");

    }

}
```

```
}
```

```
@FXML
```

```
private void back(ActionEvent event) {
```

```
    GymProject.trainerView();
```

```
}
```

```
public String calc(double weightVal, double heightValCm) {
```

```
    double heightValM = heightValCm / 100;
```

```
    if (heightValM == 0) {
```

```
        return "Invalid height";
```

```
    }
```

```
    double bmi = weightVal / (heightValM * heightValM);
```

```
    if (bmi < 18.5) {
```

```
        return "Underweight";
```

```
    } else if (bmi >= 18.5 && bmi < 24.9) {
```

```
        return "Normal weight";
```

```
    } else if (bmi >= 25 && bmi < 29.9) {
```

```
        return "Overweight";
```

```
    } else {
```

```
        return "Obesity";
```

```
    }
```

```
}
```

```

@Override

public void initialize(URL location, ResourceBundle resources) {

    weight.setEditable(false);

    height.setEditable(false);

    MusclePercentage.setEditable(false);

    FatPercentage.setEditable(false);

}

}

//=====================================================

import database.handler;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;

public class TrainerViewController {

    private handler handler = new handler();

    @FXML

```

```

private GridPane classGridPane;

@FXML

private Label nameLbl;


@FXML

private Button Edit;

@FXML

private Button Remove;


@FXML

private void handelEdit(ActionEvent event) {

    GymProject.editClassView();

}


@FXML

private void handelManageSession(ActionEvent event) {

    GymProject.addSeccionView();

}


@FXML

private void handelRemove(ActionEvent event) {

    GymProject.RemoveClassView();

}


@FXML

private void handelTrackProgressView(ActionEvent event) {

    GymProject.trackProgressView();

```

```

    }

    @FXML

    private void back(ActionEvent event) {

        GymProject.loginView();

    }

}

//=====================================================

import database.handler;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.layout.GridPane;


public class TrainerViewController {

    private handler handler = new handler();

    @FXML

```

```

private GridPane classGridPane;

@FXML

private Label nameLbl;


@FXML

private Button Edit;

@FXML

private Button Remove;


@FXML

private void handelEdit(ActionEvent event) {

    GymProject.editClassView();

}


@FXML

private void handelManageSession(ActionEvent event) {

    GymProject.addSeccionView();

}


@FXML

private void handelRemove(ActionEvent event) {

    GymProject.RemoveClassView();

}


@FXML

private void handelTrackProgressView(ActionEvent event) {

    GymProject.trackProgressView();

```



```

    }

    @FXML

    private void back(ActionEvent event) {

        GymProject.loginView();

    }

}

//=====================================================

import database.handler;

import java.net.URL;

import java.sql.Date;

import java.sql.SQLException;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.RadioButton;

import javafx.scene.control.TextField;

import javafx.scene.control.ToggleGroup;

import uml.User;

```

```
public class UpdateUserController implements Initializable {
```

```
    @FXML
```

```
    private TextField nametxt;
```

```
    @FXML
```

```
    private TextField IDtxt;
```

```
    @FXML
```

```
    private TextField DOBtxt;
```

```
    @FXML
```

```
    private TextField phoneNumtxt;
```

```
    @FXML
```

```
    private TextField emailtxt;
```

```
    @FXML
```

```
    private PasswordField passwordtxt;
```

```
    @FXML
```

```
    private RadioButton member;
```

```
    @FXML
```

```
    private RadioButton trainer;
```

```
    @FXML
```

```
    private RadioButton admin;
```

```
    @FXML
```

```
    private Label showlbl;
```

```
    @FXML
```

```
    private Label changelbl;
```

```

private handler handler = new handler();

private ToggleGroup tg = new ToggleGroup();

@FXML

private void handelShow(ActionEvent event) {
    try {
        if (IDtxt.getText().isEmpty()) {
            showlbl.setText("Enter the ID");

            nametxt.setEditable(false);
            DOBtxt.setEditable(false);
            phoneNumtxt.setEditable(false);
            emailtxt.setEditable(false);
            passwordtxt.setEditable(false);
        } else {
            int userId = Integer.parseInt(IDtxt.getText());
            User user = handler.getUserById(userId);

            if (user != null) {
                System.out.println(user);
                showlbl.setText("Correct ID");
                nametxt.setEditable(true);
                DOBtxt.setEditable(true);
                phoneNumtxt.setEditable(true);
                emailtxt.setEditable(true);
                passwordtxt.setEditable(true);
            }
        }
    }
}

```

```
// Fill the fields with user data

nametxt.setText(user.getFullName());

DOBtn.setText(user.getDateOfBirth());

phoneNumtxt.setText(user.getPhoneNumber());

emailtxt.setText(user.getEmail());

passwordtxt.setText(user.getPassword());


if (user.getRole().equals("MEMBER")) {

    tg.selectToggle(member);

} else if (user.getRole().equals("TRAINER")) {

    tg.selectToggle(trainer);

} else {

    tg.selectToggle(admin);

}

} else {

    System.out.println("No such user.");

    showlbl.setText("Invalid ID");


    nametxt.clear();

    DOBtn.clear();

    phoneNumtxt.clear();

    emailtxt.clear();

    passwordtxt.clear();


    nametxt.setEditable(false);

    DOBtn.setEditable(false);
```

```

        phoneNumtxt.setEditable(false);

        emailtxt.setEditable(false);

        passwordtxt.setEditable(false);
    }
}

} catch (NumberFormatException ex) {
    showlbl.setText("Invalid ID format");
    ex.printStackTrace();
} catch (SQLException ex) {
    ex.printStackTrace();
} catch (Exception ex) {
    ex.printStackTrace();
}

}

```

@FXML

```

private void handelConfirm(ActionEvent event) {

    String idVal = IDtxt.getText();

    String nameVal = nametxt.getText();

    String passwordVal = passwordtxt.getText();

    String emailVal = emailtxt.getText();

    String dateOfBirthVal = DOBtxt.getText();

    String phoneNumberVal = phoneNumtxt.getText();

    if (!nameVal.isEmpty() || !idVal.isEmpty()

```

```

        || !dateOfBirthVal.isEmpty() || !phoneNumberVal.isEmpty()

        || !emailVal.isEmpty() || !passwordVal.isEmpty()) {

String role = "";

if (member.isSelected()) {

    role = "MEMBER";

} else if (trainer.isSelected()) {

    role = "TRAINER";

} else if (admin.isSelected()) {

    role = "ADMIN";

}

try {

    int userId = Integer.parseInt(idVal);

    handler.updateUser(userId, passwordVal, emailVal, nameVal, Date.valueOf(dateOfBirthVal),
phoneNumberVal, role);

    changelbl.setText("User updated successfully");

} catch (SQLException ex) {

    changelbl.setText("Error:" + ex);

}

} else {

    changelbl.setText("Some field is empty");

}

}

```

```

@FXML

private void back(ActionEvent event) {

    GymProject.UsersManagementView();
}

@Override

public void initialize(URL location, ResourceBundle resources) {

    member.setToggleGroup(tg);

    trainer.setToggleGroup(tg);

    admin.setToggleGroup(tg);

    nametxt.setEditable(false);

    DOBtxt.setEditable(false);

    phoneNumtxt.setEditable(false);

    emailtxt.setEditable(false);

    passwordtxt.setEditable(false);

}

}

//=====

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

public class UsersManagementController {

@FXML

private void handelAddUser(ActionEvent event) {

```

```

        GymProject.AddUserView();
    }

@FXML
private void handelRemoveUser(ActionEvent event) {
    GymProject.RemoveUserView();
}

@FXML
private void handelUpdateUser(ActionEvent event) {
    GymProject.UpdateUserView();
}

@FXML
private void back(ActionEvent event) {
    GymProject.administratorView();
}
}

//=====

import database.DataBaseConnection;
import database.SessionInfo;
import database.handler;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

```



```

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import uml.User;


public class loginController {


    @FXML

    private TextField IDtxt;

    @FXML

    private PasswordField passwordtxt;

    @FXML

    private Label errorlbl;


    private Connection conn = null;

    private handler handler = new handler();


    public void authenticateUser(int userId, String password) throws SQLException {

        String sql = "SELECT role FROM userss_ WHERE user_id = ? AND password = ?";


        try {

            conn = DataBaseConnection.getInstance().getCon();

            PreparedStatement stmt = conn.prepareStatement(sql);

            stmt.setInt(1, userId);

            stmt.setString(2, password);

```

```

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

    String role = rs.getString("role");

    User u = handler.getUserById(userId);

    SessionInfo.getInstance().setCurrentUser(u);

    switch (role) {

        case "MEMBER":

            GymProject.memberView();

            errorlbl.setText("");

            break;

        case "TRAINER":

            GymProject.trainerView();

            errorlbl.setText("");

            break;

        case "ADMIN":

            GymProject.administratorView();

            errorlbl.setText("");

            break;

        default:

            errorlbl.setText("Unrecognized role in the database.");

            break;

    }

} else {

    errorlbl.setText("ID or password is incorrect.");

}

```

```

        rs.close();

        stmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
        errorlbl.setText("Database error occurred.");
    }
}

@FXML
private void handleButtonAction(ActionEvent event) throws SQLException {
    if (IDtxt.getText().isEmpty() || passwordtxt.getText().isEmpty()) {
        errorlbl.setText("Some field is empty");
    } else if (IDtxt.getText().equals("0101") && passwordtxt.getText().equals("1234")) {
        GymProject.administratorView();
    } else {
        try {
            int useID = Integer.parseInt(IDtxt.getText());
            String pw = passwordtxt.getText();
            authenticateUser(useID, pw);
        } catch (NumberFormatException e) {
            errorlbl.setText("Invalid ID format. Please enter a number.");
        }
    }
}

```

}

17