جامعة الأميرة
نورة بنت عبد الرحمن
كلية علوم الحاسب والمعلومات

Princess Nourah bint
Abdulrahman University
College of Computer and
Information Sciences

CS 313: Advanced Programming Language
Term Project

Title:Gym management System

Section: 5C4

Group no. 2

No.	Student Name	Student Number
1	Reemah Alruwaitea	444008771
2	jood Aldawsri	444008904
3	Lamyaa Fahad Altaweel	444008854
4	Tebyan Alrashoud	444009455
5	Layan Abdullah Alanazi	444008777
6	Mayasim alotaibi	444008897

First Semester 2024

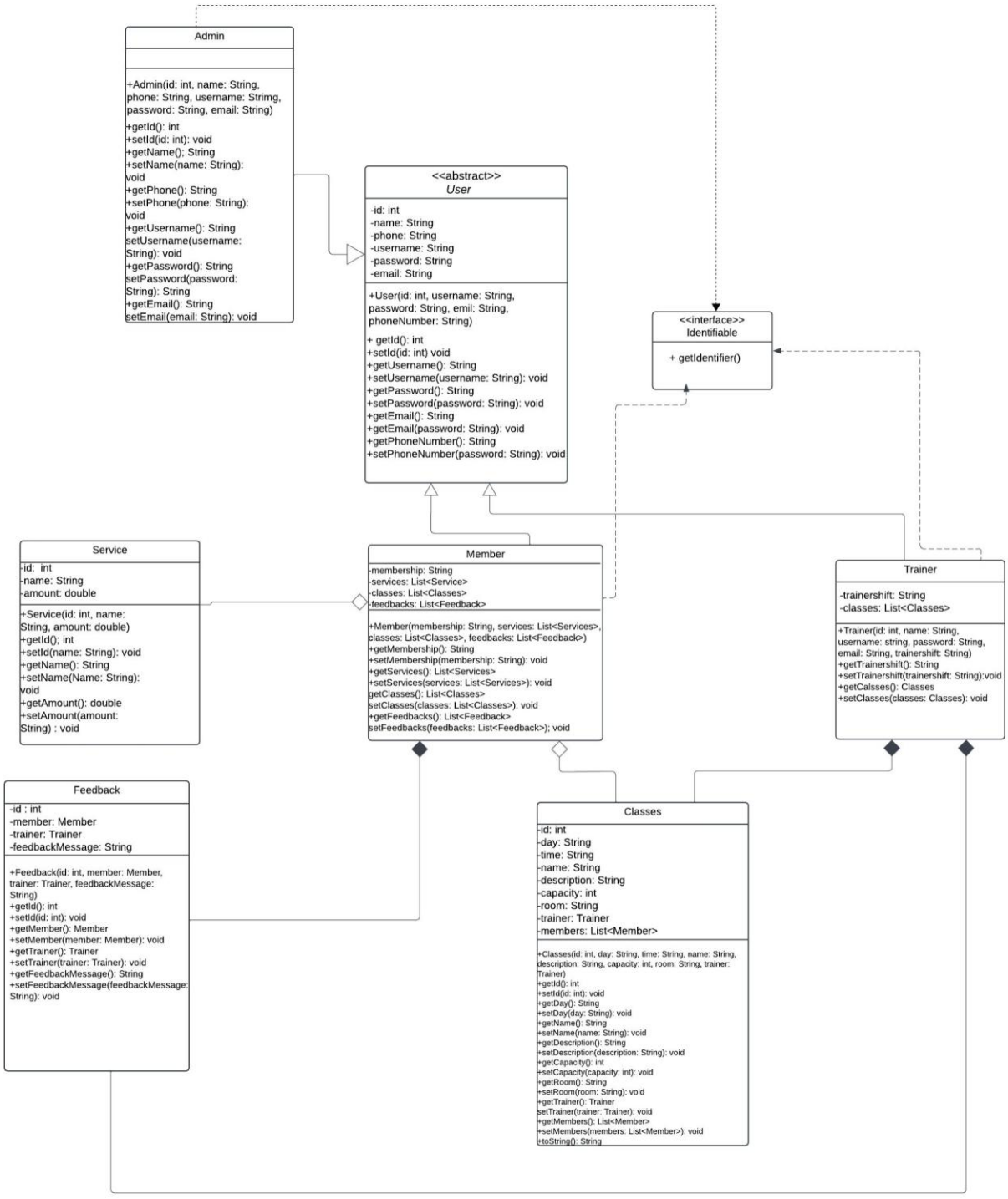
Content	Mark Obtain
Project Description	
Class Diagram	
GUI Design	
Database Tables	
Connection and Queries on Database	
Appendix (code)	
TOTAL SCORE	/10

Contents

1. Project Description

This project aims to enhance gym operations and improve user experience by developing a fully functional **Gym Membership Management System** using the JavaFX API and a Oracle database. The system offers a user-friendly platform for gym owners, staff members, streamlining the management of memberships, services, and daily records. Key features include the ability for members to book training sessions, view schedules, and manage their profiles; trainers can efficiently manage sessions and provide feedback; and administrators have user management capabilities. Additionally, the system keeps track of customer activities related to their chosen services and displays all available service packages, ensuring a comprehensive solution for managing various aspects of a fitness center. The intuitive JavaFX-based interface enhances usability, making it easier for all users to navigate their fitness journey.

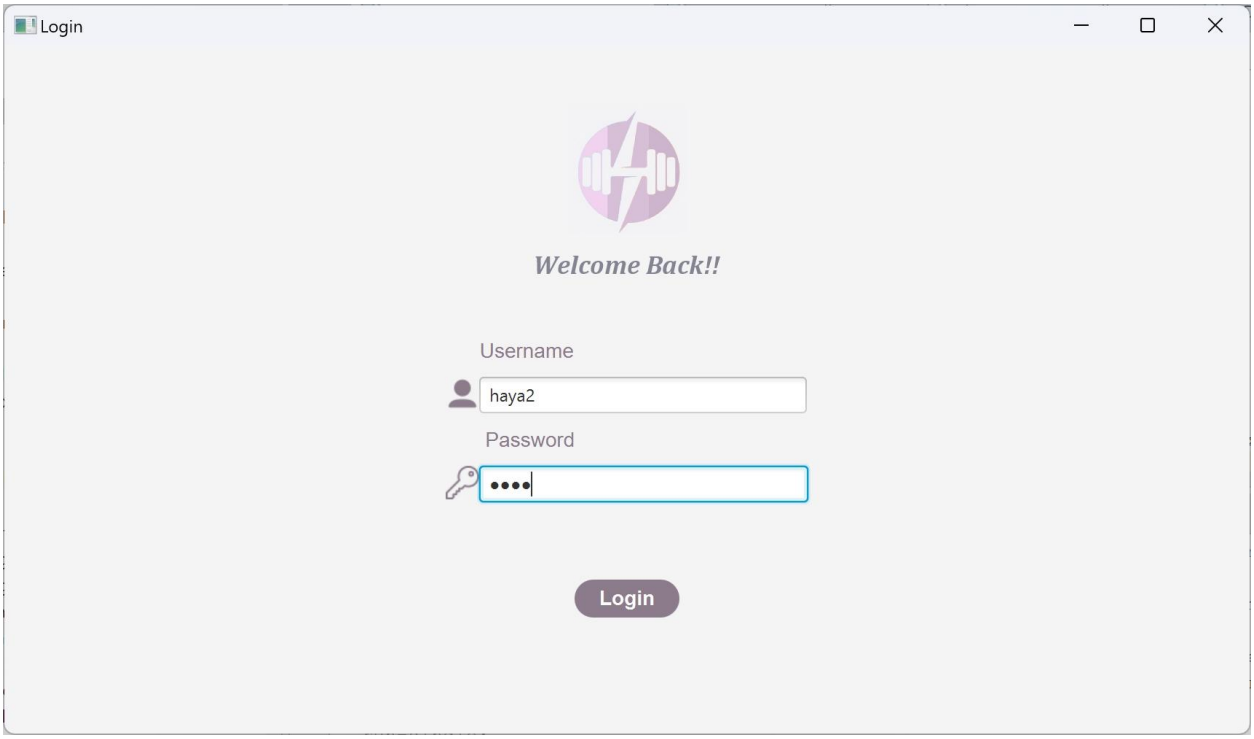
2. Class Diagram

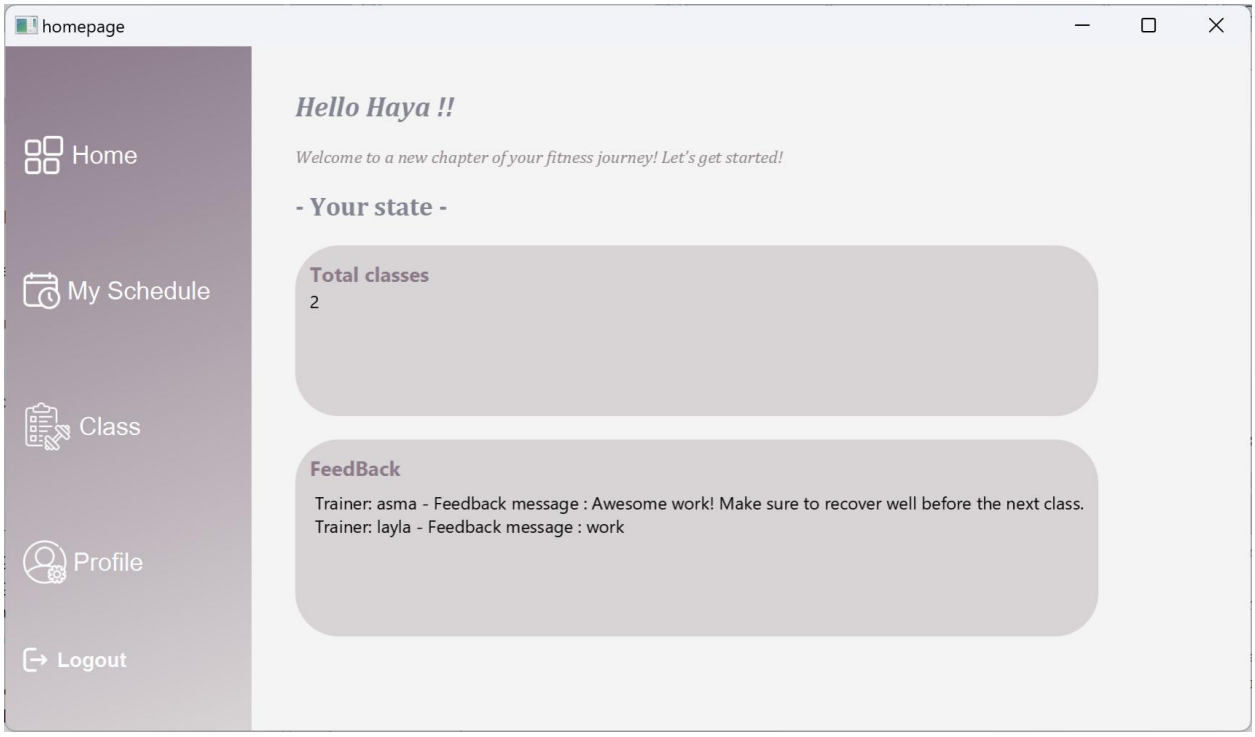


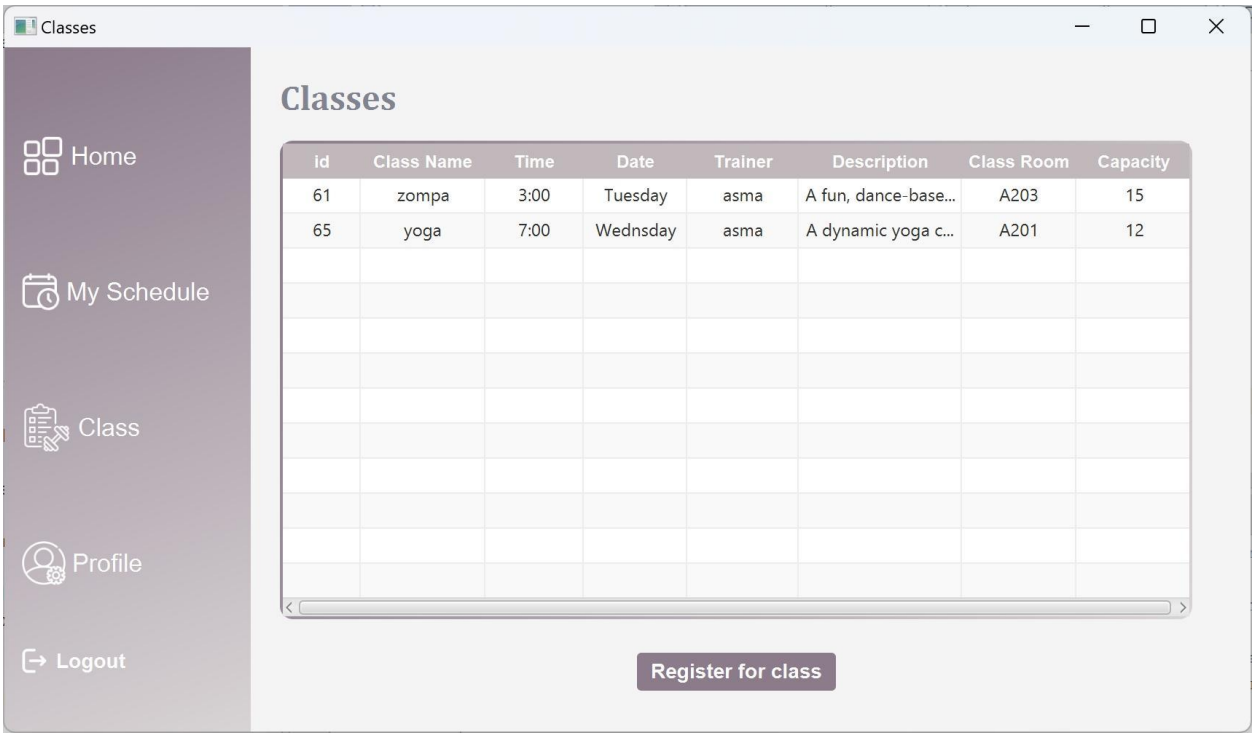
1. User
Attributes: id, name, phone, username, password, email
Description: Base class for all users, with standard getters and setters.
2. Admin (inherits from User)
• Description: Manages system settings, user accounts, and services.
3. Trainer (inherits from User)
Attributes: trainershift, classes (list)
Description: Conducts classes and manages schedules.
4. Member (inherits from User)
Attributes: membership, services, classes, feedbacks
Description: Participates in services and classes, provides feedback.
5. Classes
Attributes: id, day, time, name, description, capacity, room, trainer, members
Description: Represents scheduled classes with details on trainers and members.
6. Service
Attributes: id, name, amount
Description: Additional services like personal training.
7. Feedback
Attributes: id, member, trainer, feedbackMessage
Description: Captures member feedback on trainers and services.
8. Identifiable (Interface)
Methods: getIdentifier ()
Description: Provides unique identification for objects.

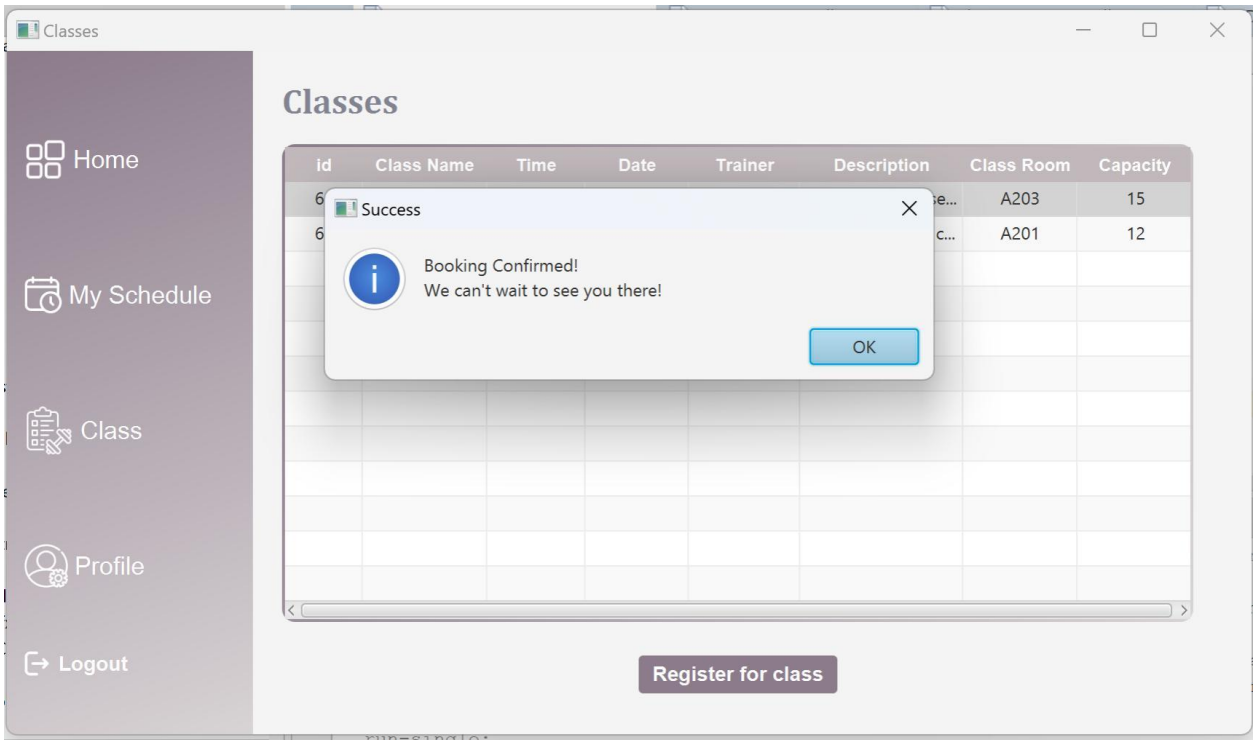
About the Relationship:-
Inheritance: Admin, Trainer, and Member derive from User.
Associations:
Trainers manage multiple Classes. o Classes link to multiple Members and
one Trainer.
Members can access multiple
Services and Classes.
o Feedback connects Members
and Trainers.
Composition: Classes require Trainers and Members, while Members maintain lists of Classes, Services, and Feedback.
Dependencies: Feedback relies on
Members and Trainers; Services are linked to Members.

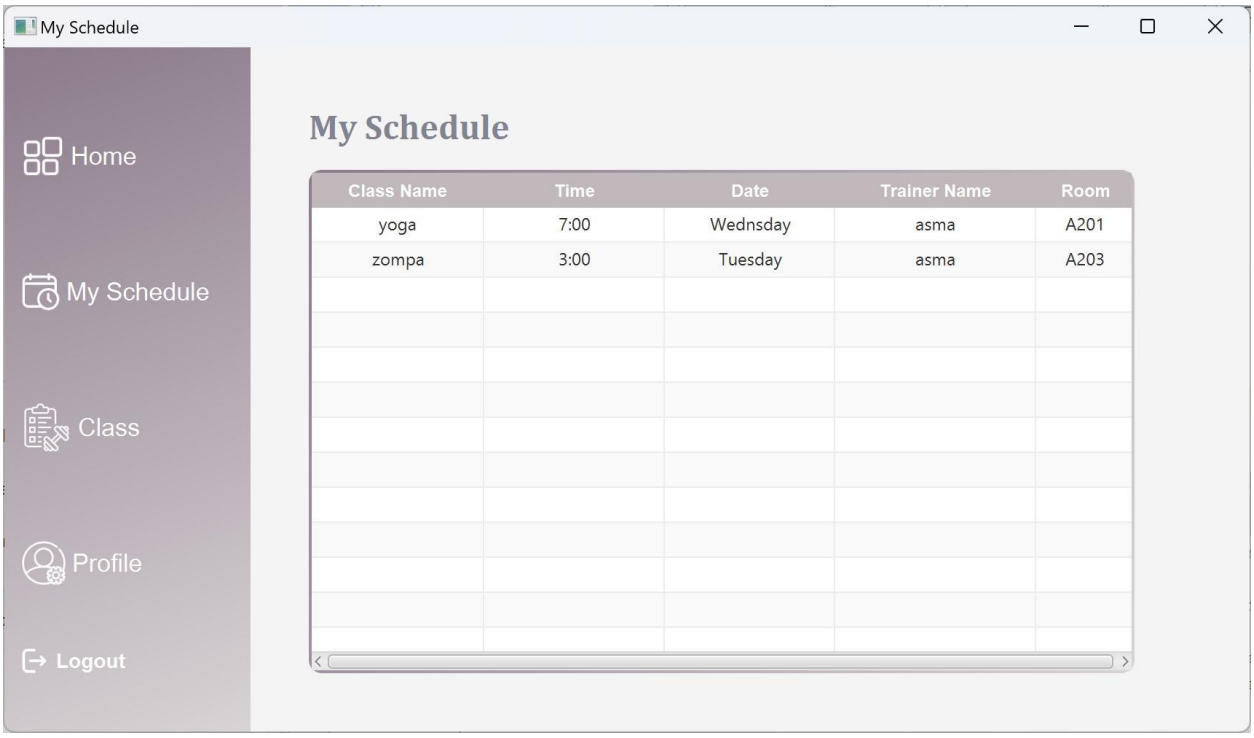
3. GUI Design

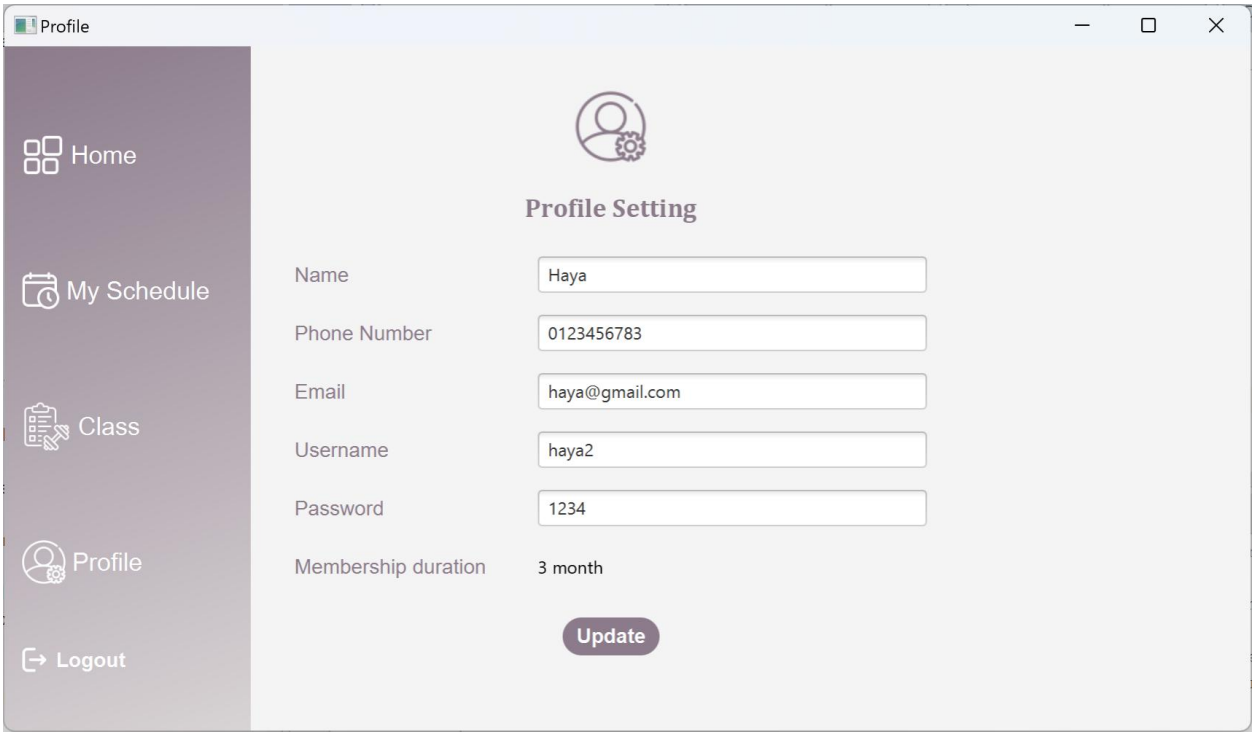

Login
Welcome Back!!
Username
haya2
Password
Login

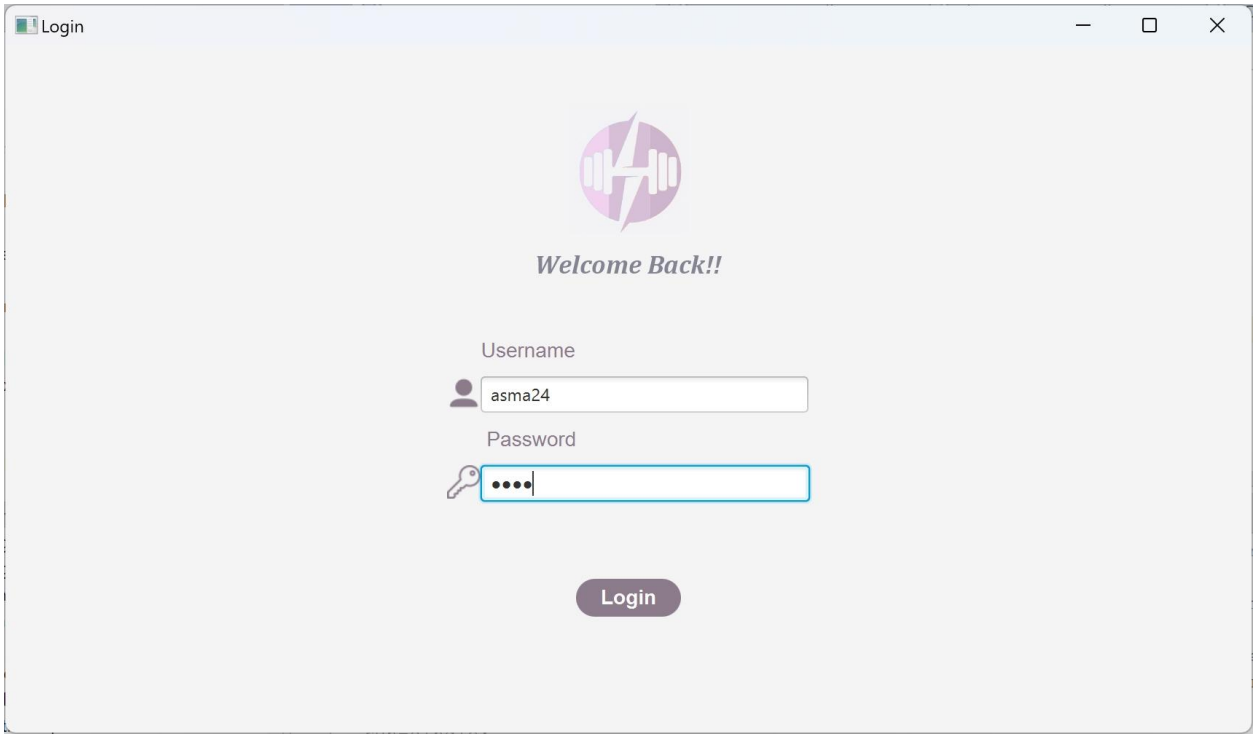

homepage
Home
My Schedule
Class
Profile
Logout
Hello Haya !!
Welcome to a new chapter of your fitness journey! Let's get started!
- Your state -
Total classes
2
FeedBack
Trainer: asma - Feedback message : Awesome work! Make sure to recover well before the next class.
Trainer: layla - Feedback message : work

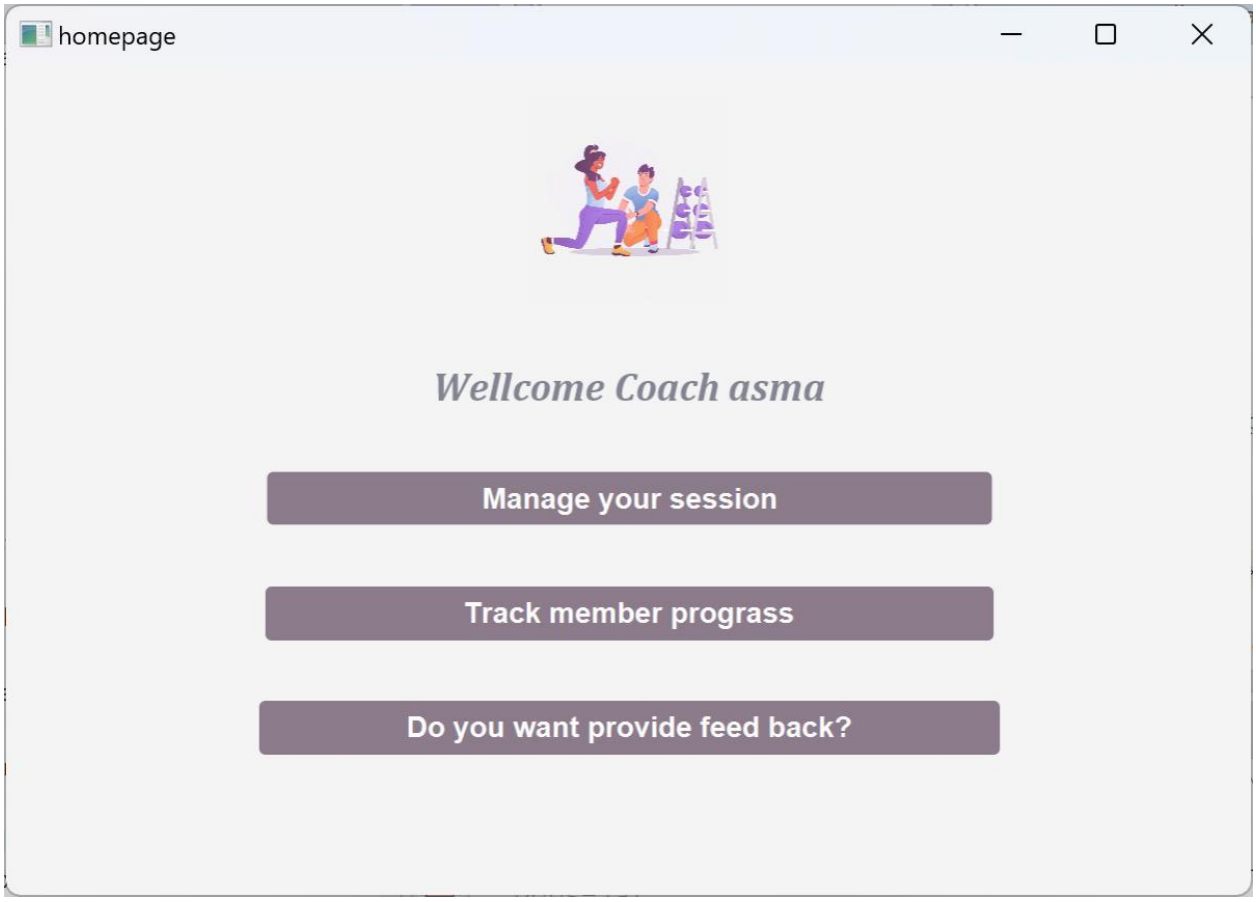

Classes
Home
My Schedule
Class
Profile
Logout
Classes
id
Class Name
Time
Date
Trainer
Description
Class Room
Capacity
61
zompa
3:00
Tuesday
asma
A fun, dance-base...
A203
15
65
yoga
7:00
Wednsday
asma
A dynamic yoga c...
A201
12
Register for class

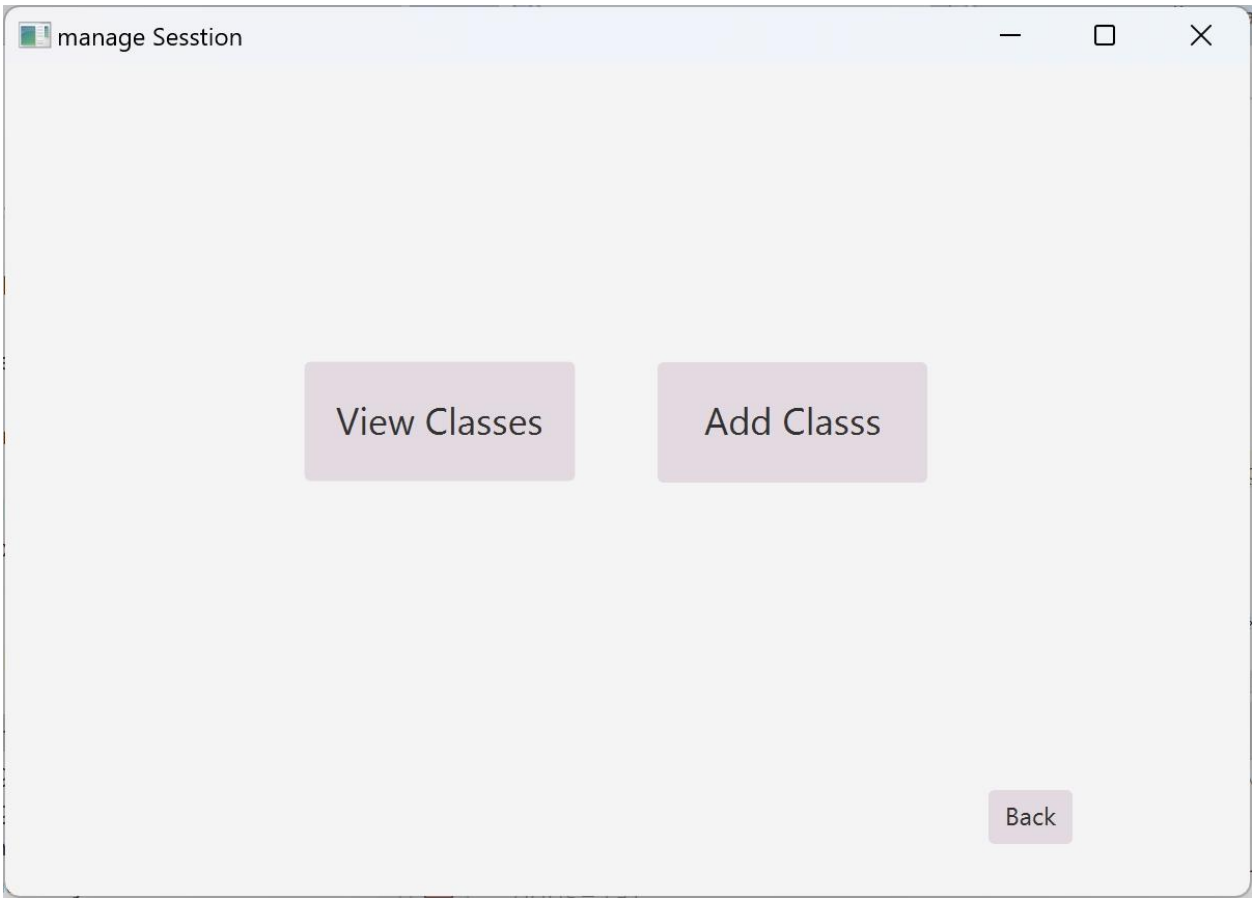

Classes
Home
My Schedule
Class
Profile
Logout
Classes
id
Class Name
Time
Date
Trainer
Description
Class Room
Capacity
A203
15
A201
12
Success
Booking Confirmed!
We can't wait to see you there!
OK
Register for class

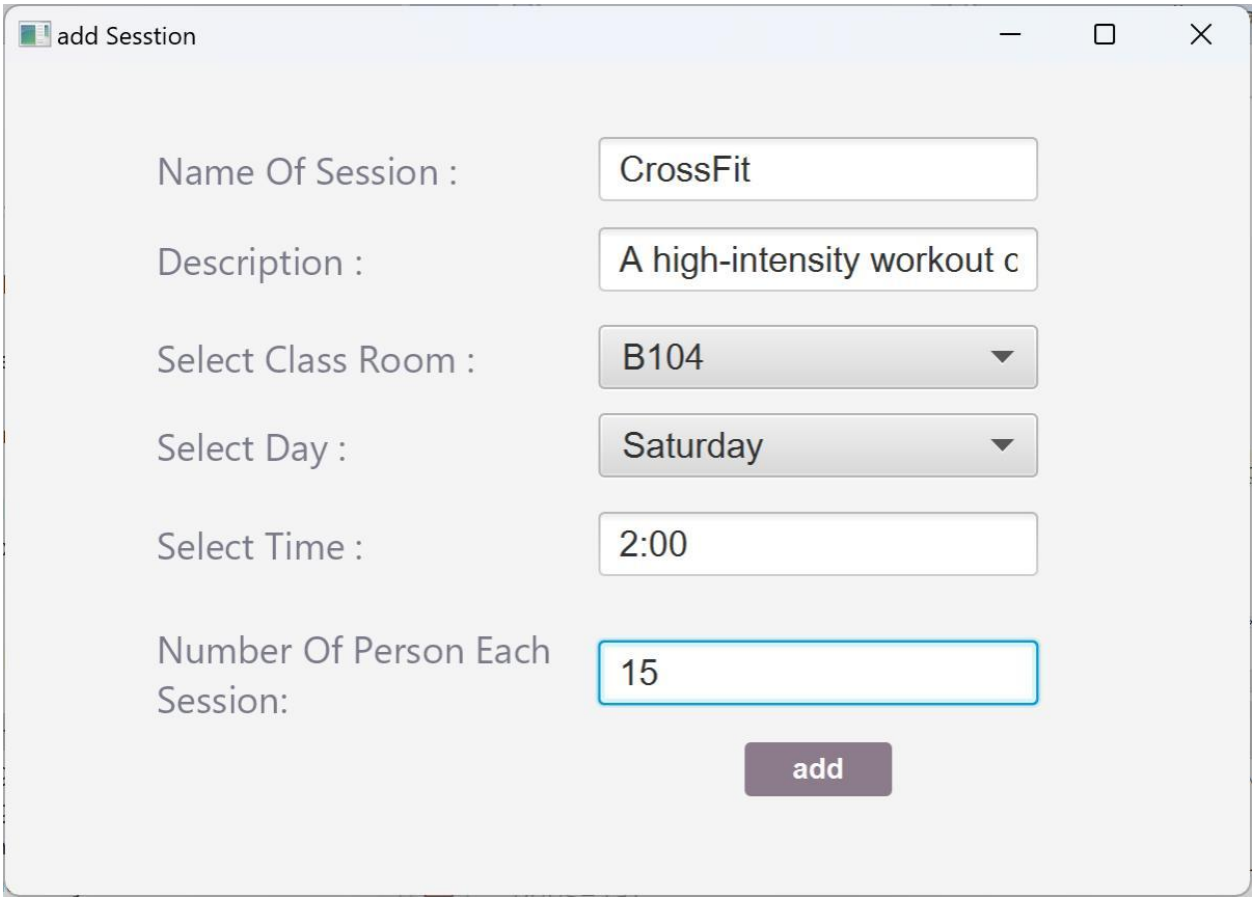

My Schedule
Home
My Schedule
Class
Profile
Logout
My Schedule
Class Name
Time
Date
Trainer Name
Room
yoga
7:00
Wednsday
asma
A201
zompa
3:00
Tuesday
asma
A203

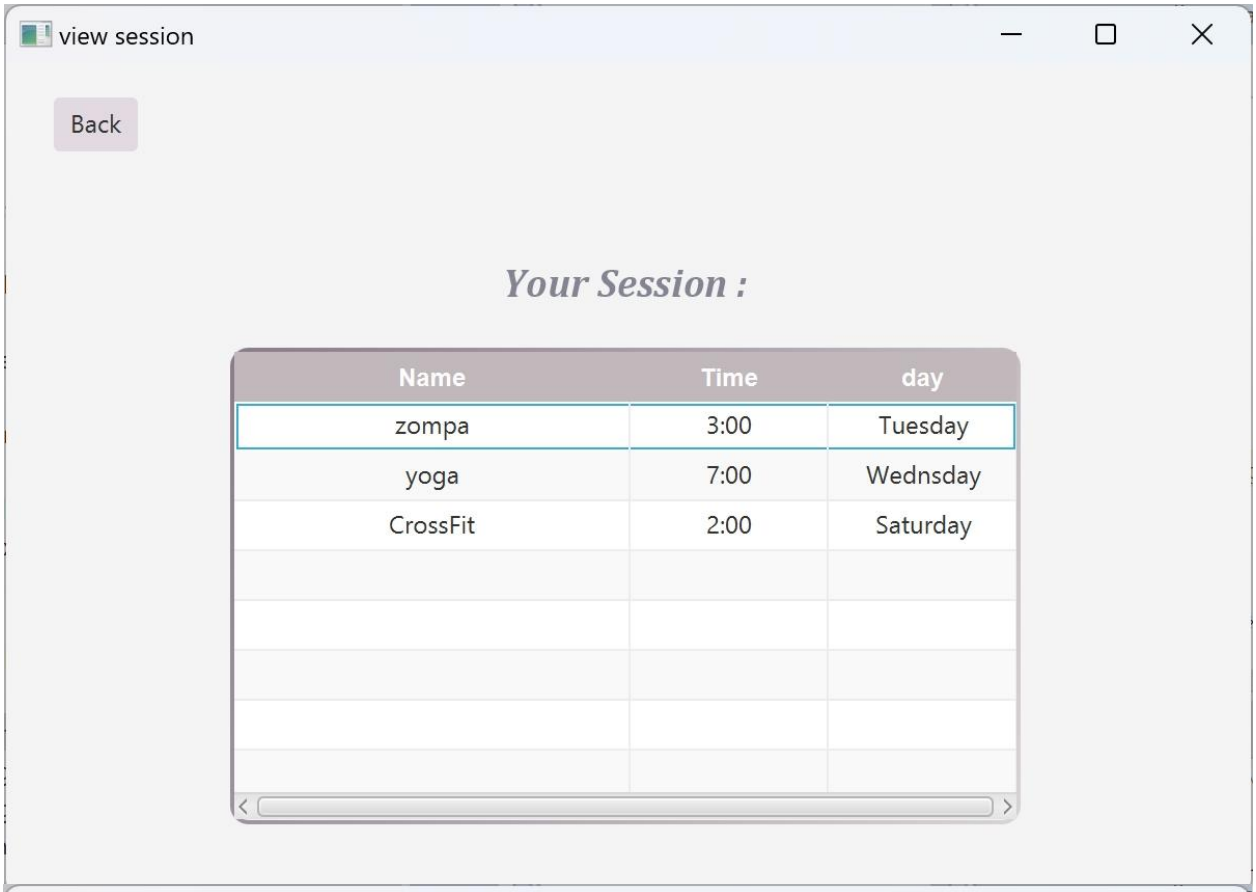

Profile
Home
My Schedule
Class
Profile
Logout
Profile Setting
Name
Haya
Phone Number
0123456783
Email
haya@gmail.com
Username
haya2
Password
1234
Membership duration
3 month
Update

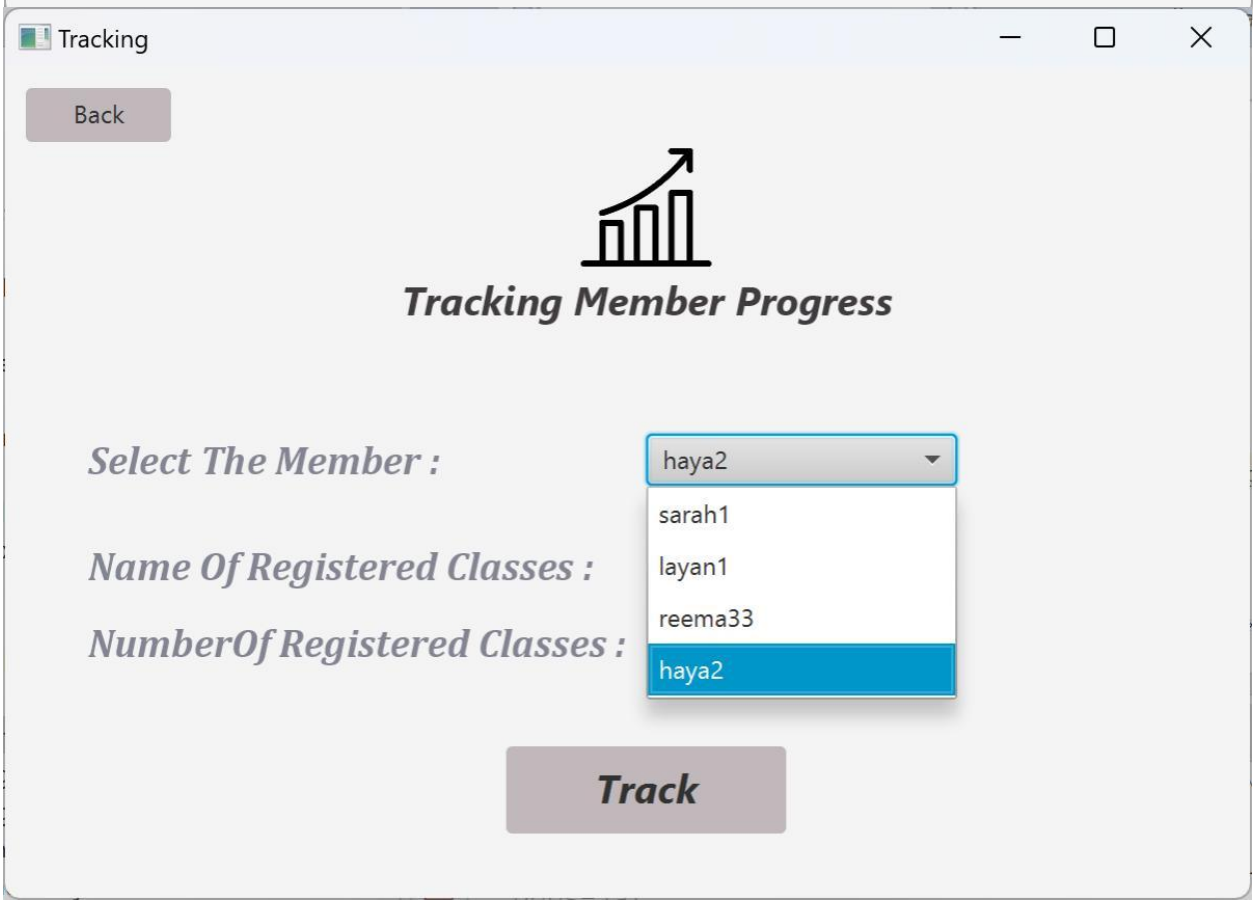

Login
Welcome Back!!
Username
asma24
Password
Login

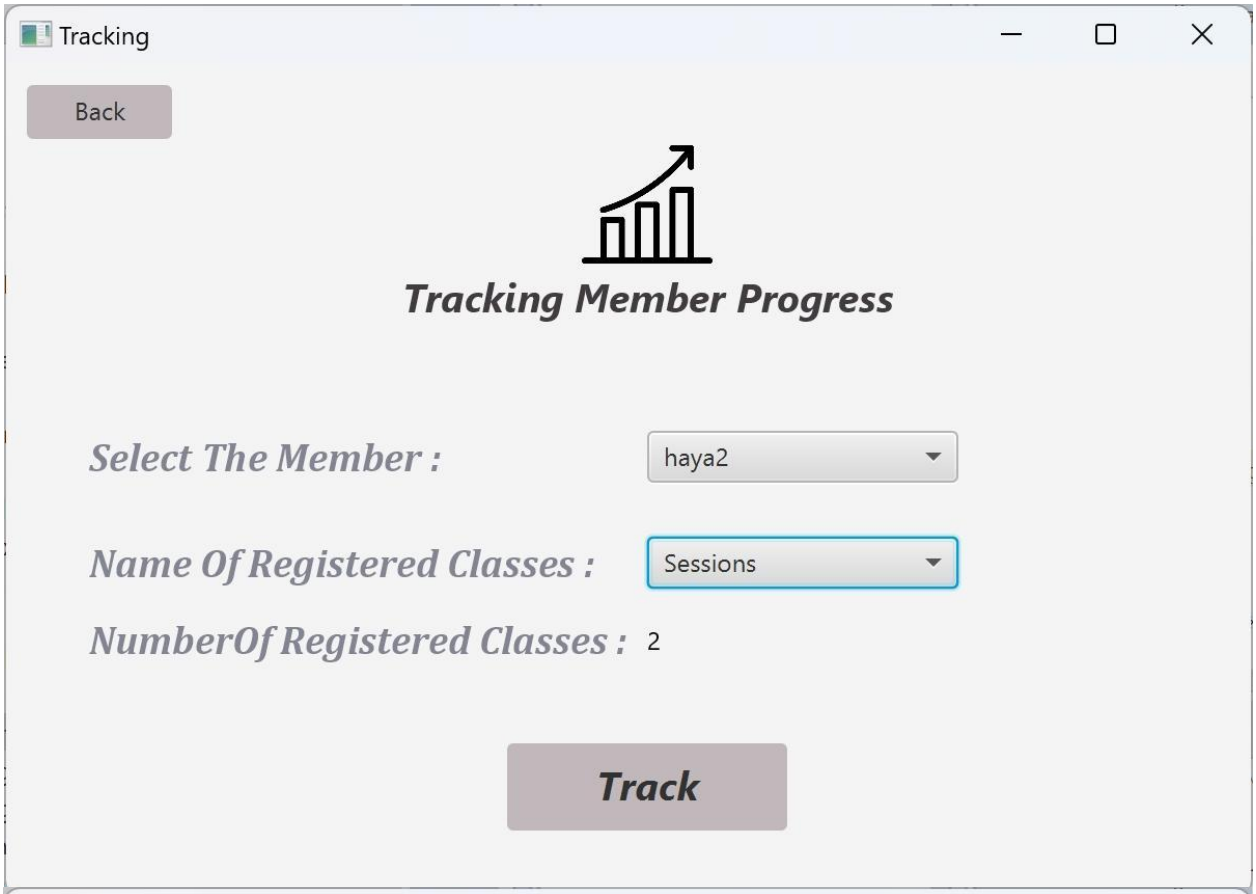

homepage
Wellcome Coach asma
Manage your session
Track member prograss
Do you want provide feed back?

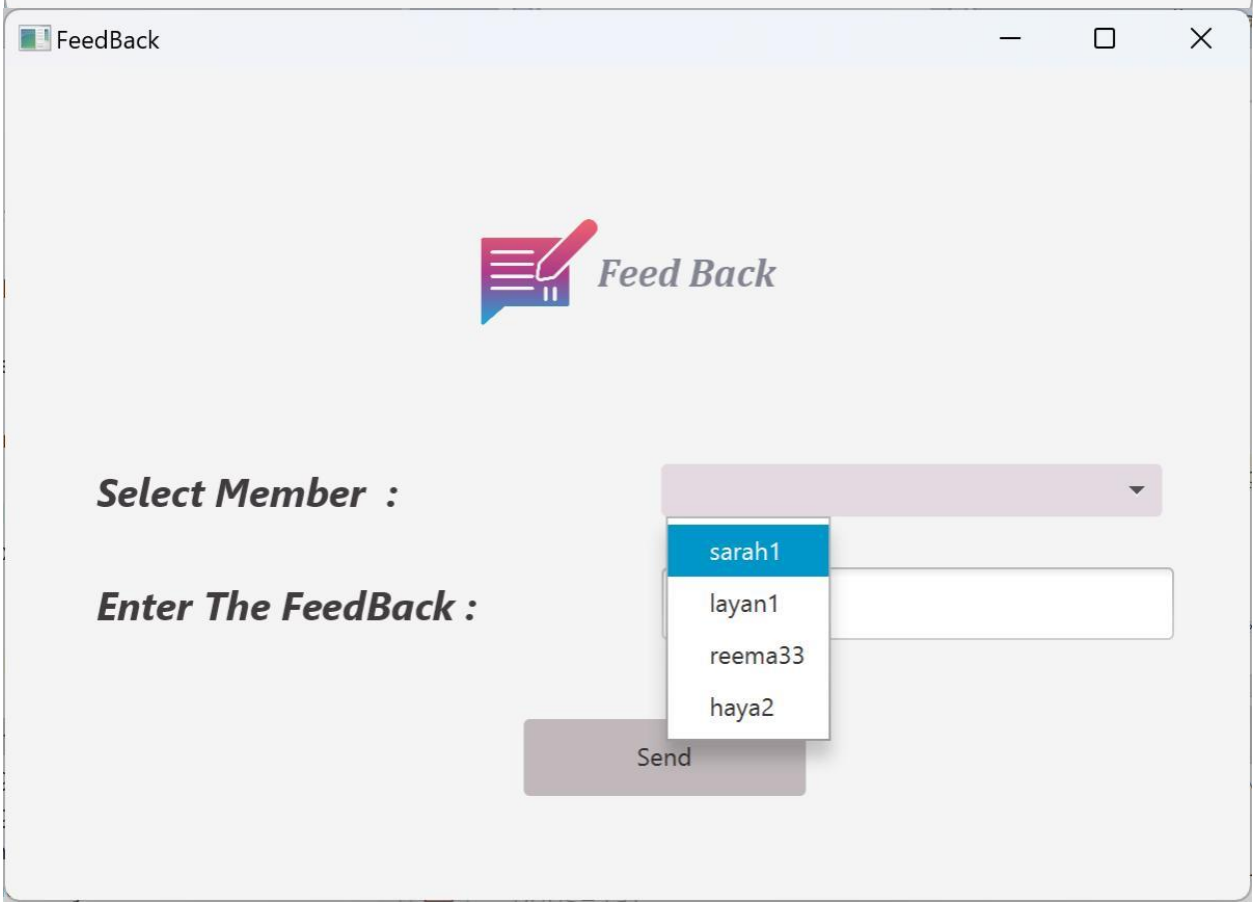

manage Sesstion
View Classes
Add Classs
Back

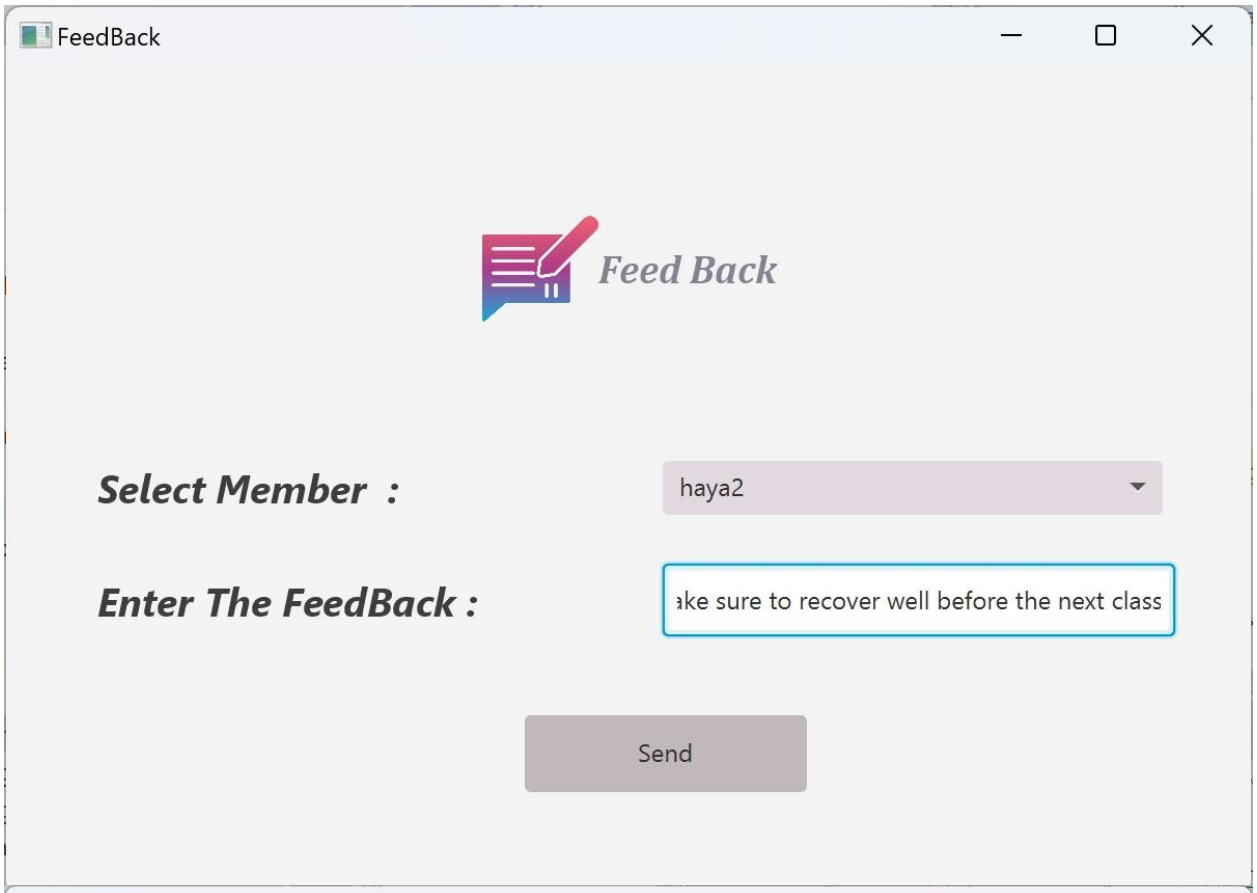

add Sesstion
Name Of Session :
CrossFit
Description :
A high-intensity workout c
Select Class Room :
B104
Select Day :
Saturday
Select Time :
2:00
Number Of Person Each Session:
15
add

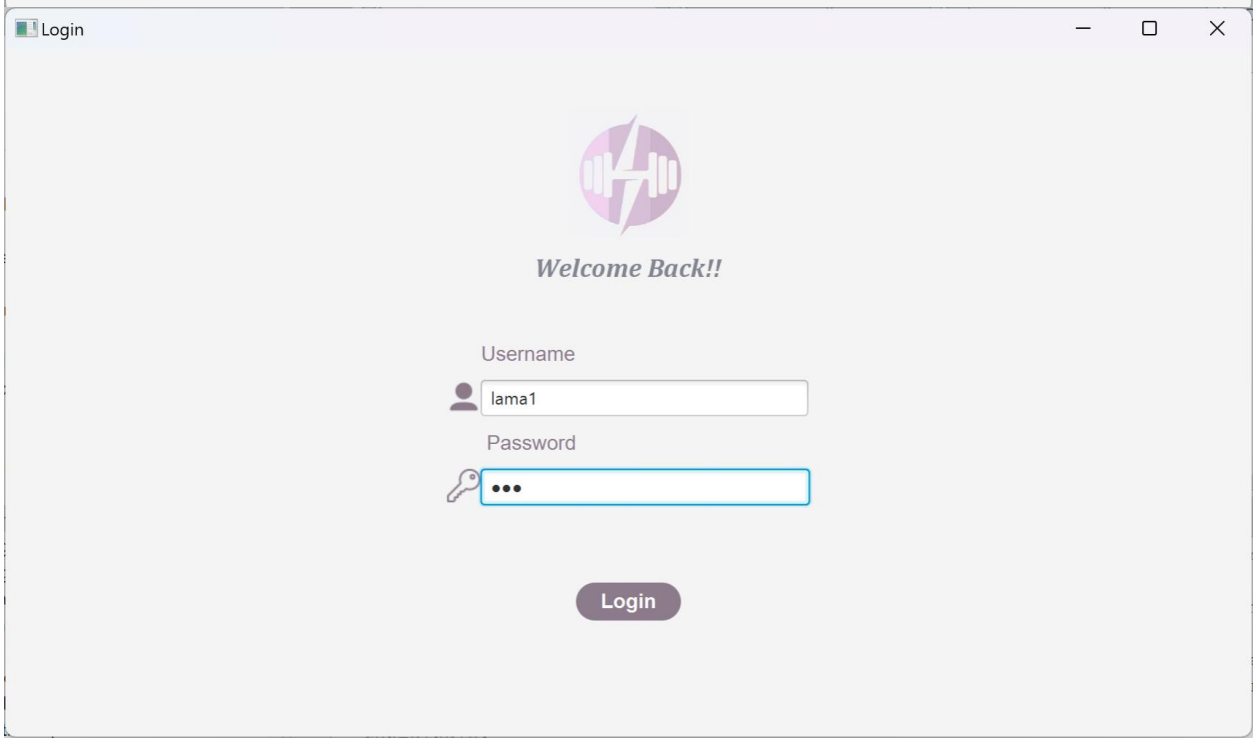

view session
Back
Your Session :
Name
Time
day
zompa
3:00
Tuesday
yoga
7:00
Wednsday
CrossFit
2:00
Saturday

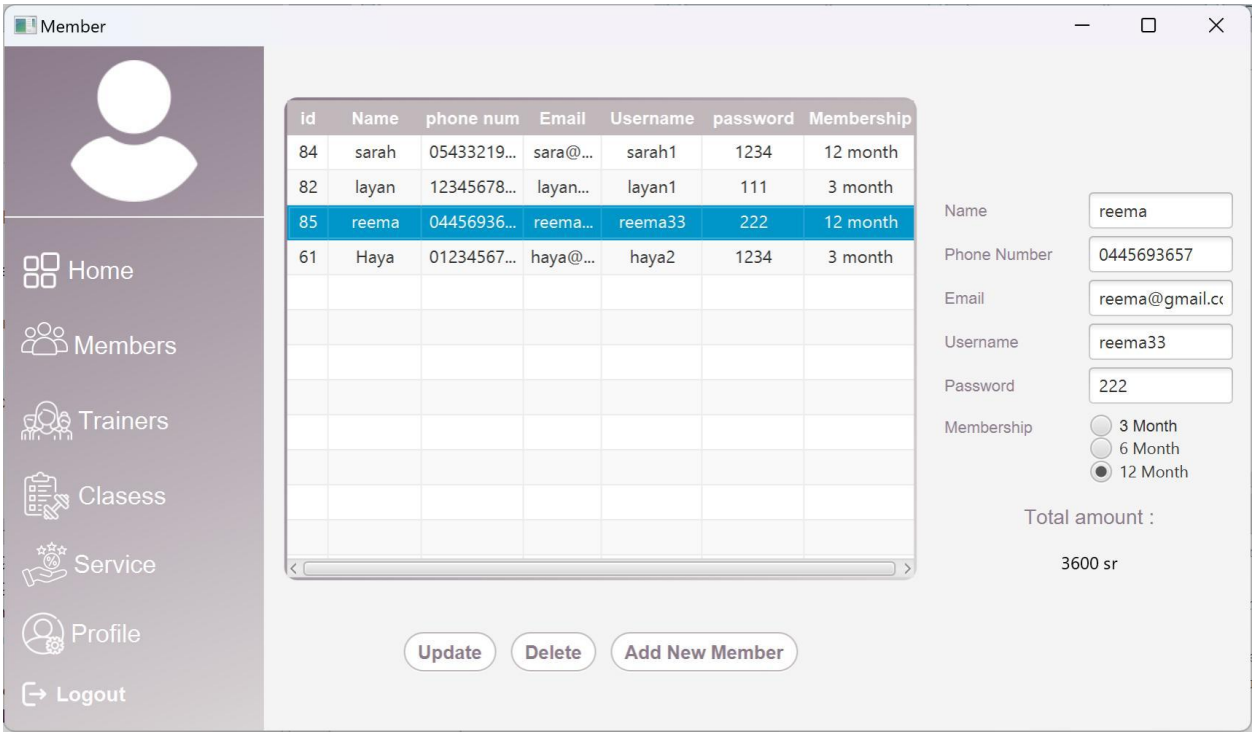

Tracking
Back
Tracking Member Progress
Select The Member :
haya2
sarah1
layan1
reema33
haya2
Name Of Registered Classes :
NumberOf Registered Classes :
Track

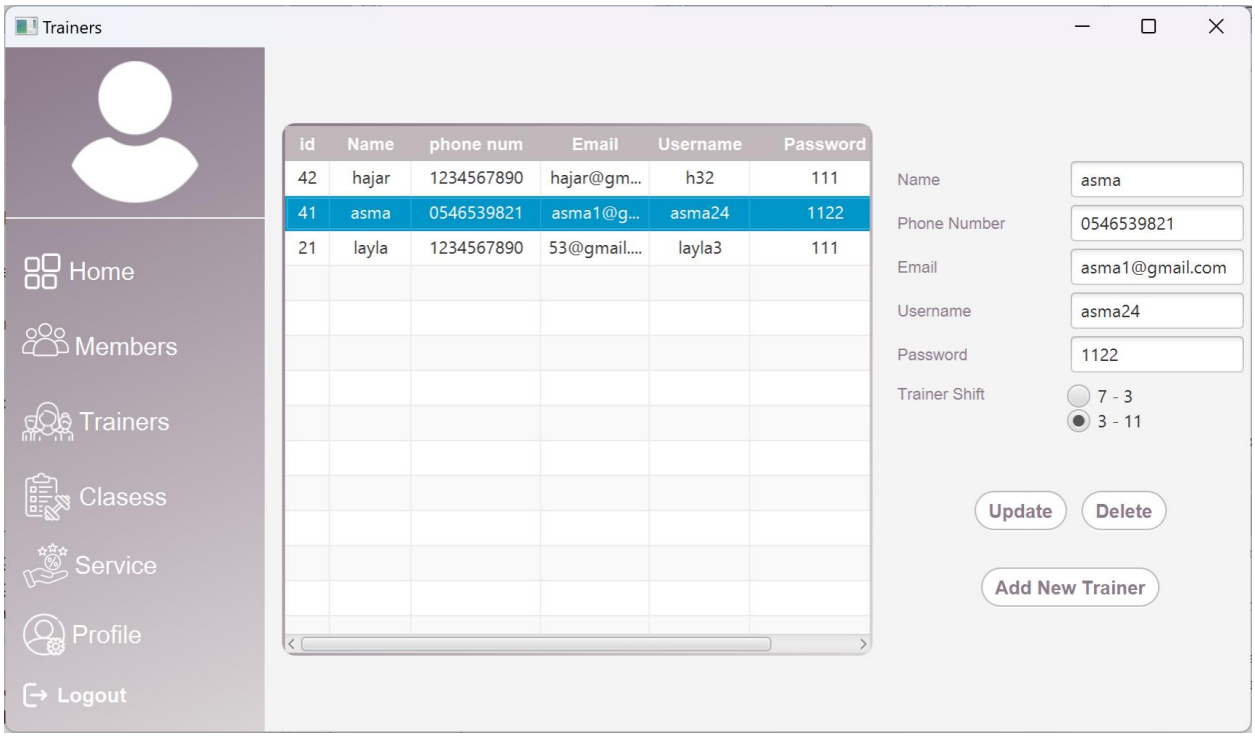

Tracking
Back
Tracking Member Progress
Select The Member :
haya2
Name Of Registered Classes :
Sessions
NumberOf Registered Classes :
2
Track

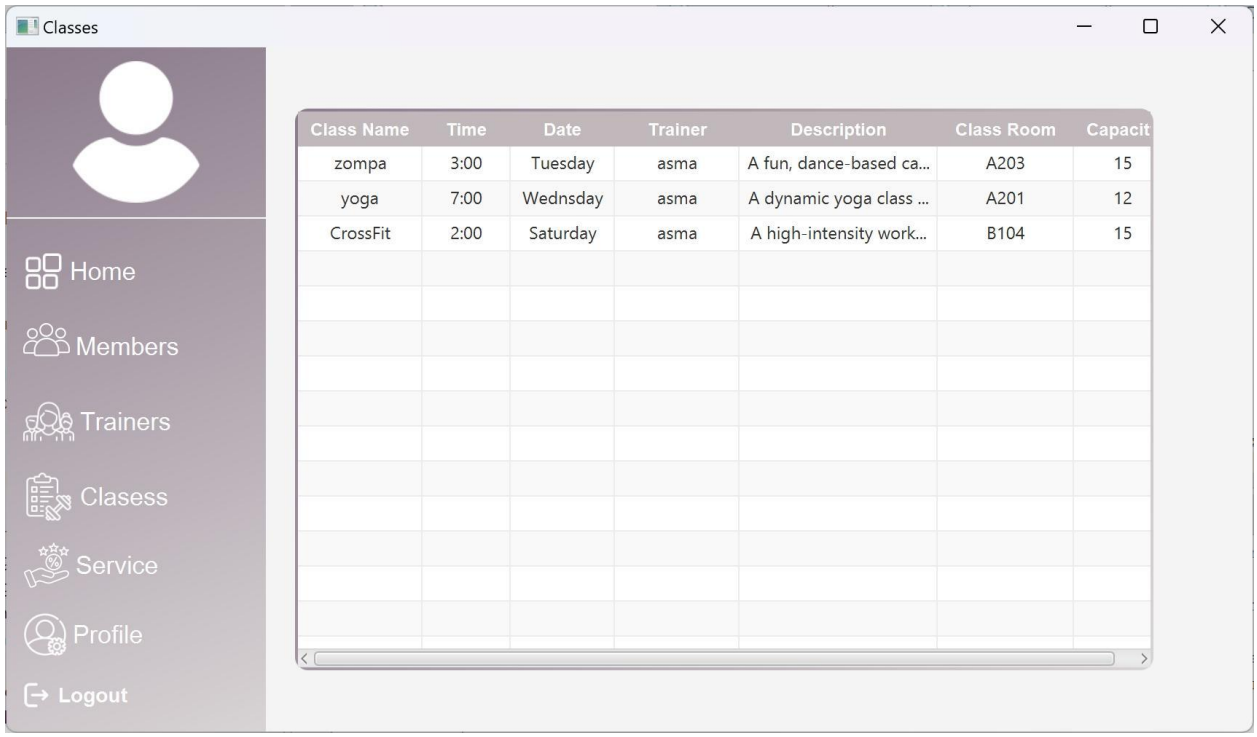

FeedBack
Feed Back
Select Member :
Enter The FeedBack :
sarah1
layan1
reema33
haya2
Send

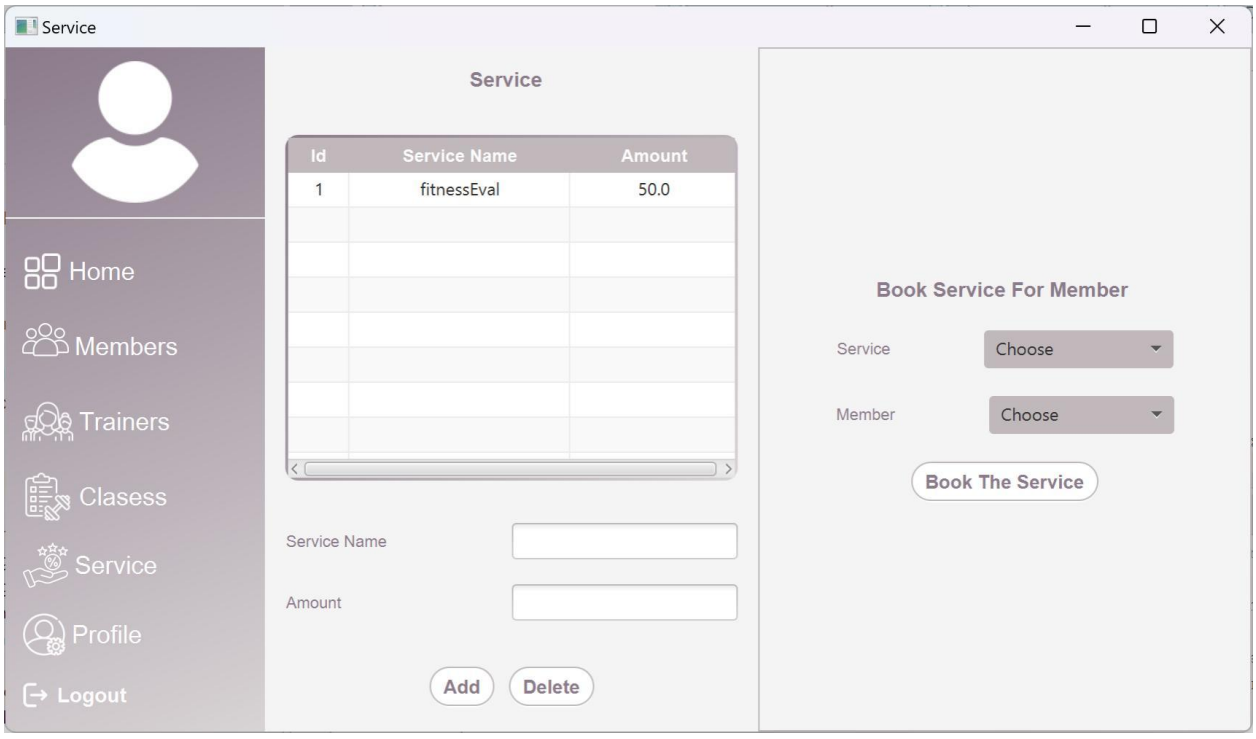

FeedBack
Feed Back
Select Member :
haya2
Enter The FeedBack :
ake sure to recover well before the next class
Send

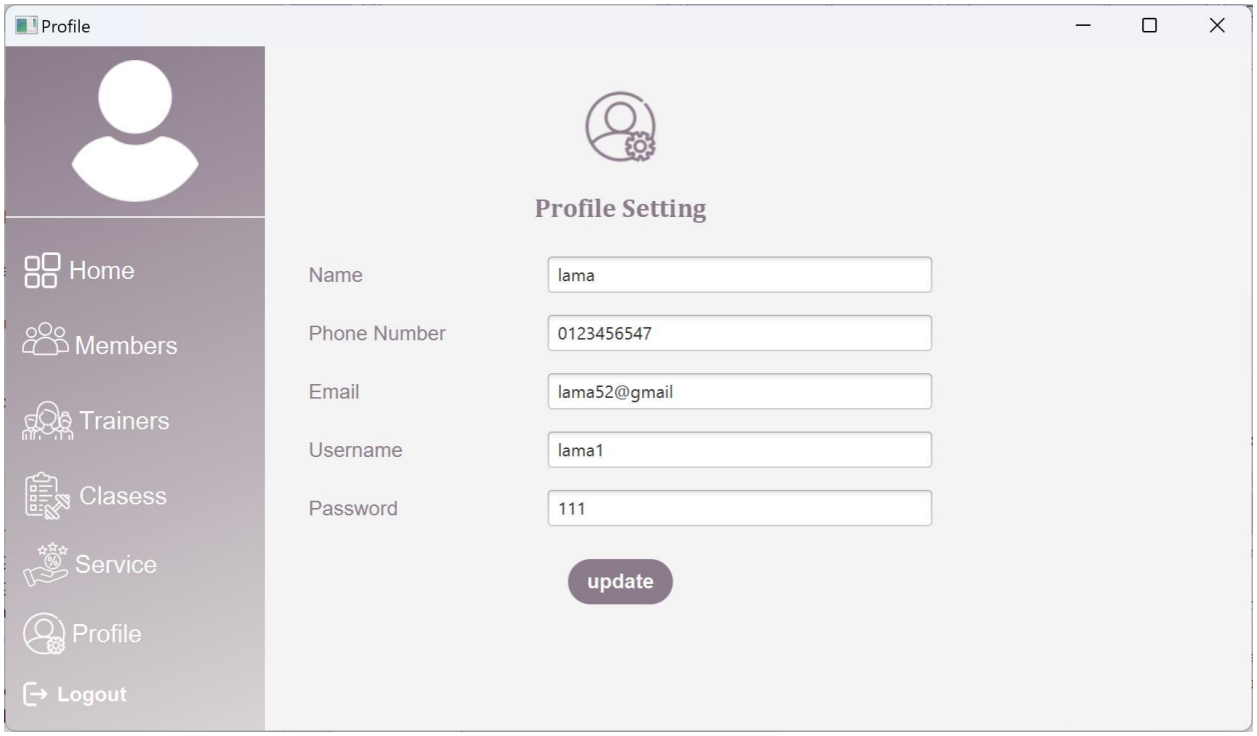

Login
Welcome Back!!
Username
lama1
Password
Login


Member
Home
Members
Trainers
Clasess
Service
Profile
Logout
id Name phone num Email Username password Membership
84 sarah 05433219... sara@... sarah1 1234 12 month
82 layan 12345678... layan... layan1 111 3 month
85 reema 04456936... reema... reema33 222 12 month
61 Haya 01234567... haya@... haya2 1234 3 month
Name reema
Phone Number 0445693657
Email reema@gmail.cc
Username reema33
Password 222
Membership
3 Month
6 Month
12 Month
Total amount :
3600 sr
Update
Delete
Add New Member


Trainers
Home
Members
Trainers
Clasess
Service
Profile
Logout
id Name phone num Email Username Password
42 hajar 1234567890 hajar@gm... h32 111
41 asma 0546539821 asma1@g... asma24 1122
21 layla 1234567890 53@gmail.... layla3 111
Name asma
Phone Number 0546539821
Email asma1@gmail.com
Username asma24
Password 1122
Trainer Shift
7 - 3
3 - 11
Update
Delete
Add New Trainer


Classes
Home
Members
Trainers
Clasess
Service
Profile
Logout
Class Name
Time
Date
Trainer
Description
Class Room
Capacit
zompa
3:00
Tuesday
asma
A fun, dance-based ca...
A203
15
yoga
7:00
Wednsday
asma
A dynamic yoga class ...
A201
12
CrossFit
2:00
Saturday
asma
A high-intensity work...
B104
15


Service
Home
Members
Trainers
Clasess
Service
Profile
Logout
Service
Id
Service Name
Amount
1
fitnessEval
50.0
Service Name
Amount
Add
Delete
Book Service For Member
Service
Choose
Member
Choose
Book The Service

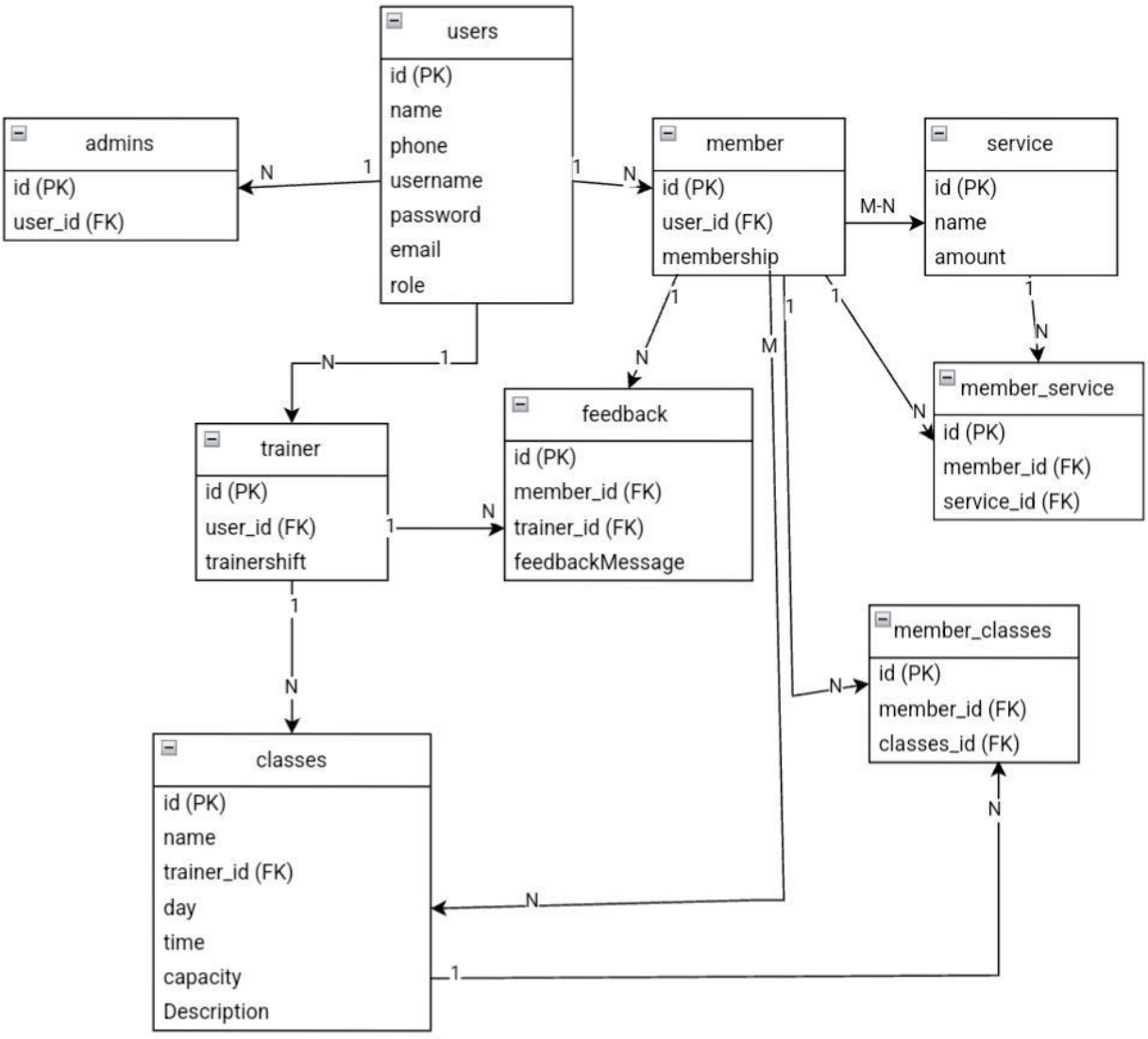

Profile
Home
Members
Trainers
Clasess
Service
Profile
Logout
Profile Setting
Name
lama
Phone Number
0123456547
Email
lama52@gmail
Username
lama1
Password
111
update

4. Database Tables

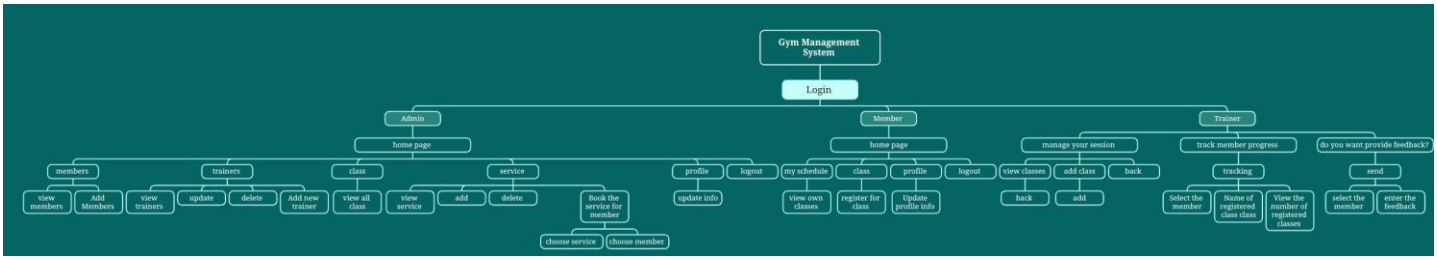
```
CREATE TABLE users (
   id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   phone CHAR(10),
   username VARCHAR(50) NOT NULL UNIQUE,
   password VARCHAR(100) NOT NULL,
   email VARCHAR(100),
   role VARCHAR(20)
);

CREATE TABLE trainer(
   id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   user_id NUMBER NOT NULL,
   trainershift VARCHAR(50),
   CONSTRAINT fk_traineruser FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE admins (
   id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   user_id NUMBER NOT NULL,
   CONSTRAINT fk_adminuser FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE member (
   id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   user_id NUMBER NOT NULL,
   membership VARCHAR(50),
   CONSTRAINT fk_memberuser FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE service (
   id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   amount NUMBER NOT NULL
);

CREATE TABLE classes (
   id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   trainer_id NUMBER NOT NULL,
```

```
    day VARCHAR(10),
    time VARCHAR(10),
    capacity VARCHAR(10),
    Description VARCHAR(100),
    CONSTRAINT fktrainer FOREIGN KEY (trainer_id) REFERENCES trainer(id)
);

CREATE TABLE feedback (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
    member_id NUMBER NOT NULL,
    trainer_id NUMBER NOT NULL,
    feedbackMessage VARCHAR(100) NOT NULL,
    CONSTRAINT fkmemberid FOREIGN KEY (member_id) REFERENCES member(id),
    CONSTRAINT fktraineiId FOREIGN KEY (trainer_id) REFERENCES trainer(id)
);

CREATE TABLE member_service (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
    member_id NUMBER NOT NULL,
    service_id NUMBER NOT NULL,
    CONSTRAINT fkmember FOREIGN KEY (member_id) REFERENCES member(id),
    CONSTRAINT fkservice FOREIGN KEY (service_id) REFERENCES service(id)
);
CREATE TABLE member_classes(
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
    member_id NUMBER NOT NULL,
    classes_id NUMBER NOT NULL,
    CONSTRAINT usermember_id FOREIGN KEY (member_id) REFERENCES member(id) ON DELETE
CASCADE,
    CONSTRAINT fk_classes FOREIGN KEY (classes_id) REFERENCES classes(id) ON DELETE CASCADE
)
```

DataBase schema:



System hierarchy:



5. Connection and Queries on Database

```java
package gymmanagementsystem.DB;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class connect {

    private static final String URL = "jdbc:oracle:thin:@localhost:1521:xe";
    private static final String USER = "system";
    private static final String PASSWORD = "1234";

    private static connect instance;
    private Connection connection;

    private connect() {
        try {
            connection = DriverManager.getConnection( url: URL,  user: USER,  password: PASSWORD);
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException( message: "Failed to connect to the database.");
        }
    }

    // Singleton pattern: Get the single instance of the connect class
    public static connect getInstance() {
        if (instance == null) {
            instance = new connect();
        }
        return instance;
    }
```

```java

    // Get the connection (check if it's closed, and reconnect if needed)
    public Connection getConnection() {
        try {
            // If the connection is null or closed, create a new one
            if (connection == null || connection.isClosed()) {
                connection = DriverManager.getConnection( url: URL,  user: USER,  password: PASSWORD);
            }
        } catch (SQLException e) {
            e.printStackTrace();
            throw new RuntimeException( message: "Failed to reconnect to the database.");
        }
        return connection;
    }


}
```

```java
// Method to add a new book to the database
public static boolean insertMember(Member member) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    PreparedStatement memberStmt = null;

    String sql = "INSERT INTO users(name, phone, username, password, email,role) VALUES (?, ?, ?, ?, ?,?)";
    String getIdSql = "SELECT id FROM users WHERE username = ?";

    try {
        conn = connect.getInstance().getConnection();

        // Insert into the users table
        pstmt = conn.prepareStatement( string:sql);
        pstmt.setString( i:1,  string:member.getName());
        pstmt.setString( i:2,  string:member.getPhone());
        pstmt.setString( i:3,  string:member.getUsername());
        pstmt.setString( i:4,  string:member.getPassword());
        pstmt.setString( i:5,  string:member.getEmail());
        pstmt.setString( i:6,  string:"member");

        int affectedRows = pstmt.executeUpdate();
        if (affectedRows == 0) {
            throw new SQLException( reason:"Creating user failed, no rows affected.");
        }

        // Retrieve the generated user ID by querying using the unique username
        int userId;
        try (PreparedStatement getIdStmt = conn.prepareStatement( string:getIdSql)) {
            getIdStmt.setString( i:1,  string:member.getUsername());
            try ( var rs = getIdStmt.executeQuery()) {
                if (rs.next()) {
```

```java
                // Insert into the member table with the generated user_id
                String memberSql = "INSERT INTO member (user_id, membership) VALUES (?, ?)";
                memberStmt = conn.prepareStatement( string:memberSql);
                memberStmt.setInt( i:1,  i:userId);
                memberStmt.setString( i:2,  string:member.getMembership());

                memberStmt.executeUpdate();
                return true;
            } catch (SQLException e) {
                e.printStackTrace();
                return false;
            } finally {
                // Close resources
                try {
                    if (pstmt != null) {
                        pstmt.close();
                    }
                    if (memberStmt != null) {
                        memberStmt.close();
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
```

```java
public static boolean insertTrainer(Trainer trainer) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    PreparedStatement trainerStmt = null;

    String sql = "INSERT INTO users(name, phone, username, password, email,role) VALUES (?, ?, ?, ?, ?,?)";
    String getIdSql = "SELECT id FROM users WHERE username = ?";

    try {
        conn = connect.getInstance().getConnection();

        // Insert into the "users" table
        pstmt = conn.prepareStatement( string:sql);
        pstmt.setString( i:1,  string:trainer.getName());
        pstmt.setString( i:2,  string:trainer.getPhone());
        pstmt.setString( i:3,  string:trainer.getUsername());
        pstmt.setString( i:4,  string:trainer.getPassword());
        pstmt.setString( i:5,  string:trainer.getEmail());
        pstmt.setString( i:6,  string:"trainer");

        int affectedRows = pstmt.executeUpdate();
        if (affectedRows == 0) {
            throw new SQLException( reason:"Creating user failed, no rows affected.");
        }

        // Retrieve the generated user ID by querying using the unique username
        int userId;
        try (PreparedStatement getIdStmt = conn.prepareStatement( string:getIdSql)) {
            getIdStmt.setString( i:1,  string:trainer.getUsername());
            try ( var rs = getIdStmt.executeQuery()) {
                if (rs.next()) {
                    userId = rs.getInt( string:"id");
                } else {
```

```
            // Insert into the trainer table with the generated user_id
            String trainerSql = "INSERT INTO trainer (user_id, trainershift) VALUES (?, ?)";
            trainerStmt = conn.prepareStatement( string:trainerSql);
            trainerStmt.setInt( i:1, i1:userId);
            trainerStmt.setString( i:2, string:trainer.getTrainershift());

            trainerStmt.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally {
            // Close resources
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
                if (trainerStmt != null) {
                    trainerStmt.close();
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    public static boolean insertAdmin(Admin admin) {
        Connection conn = null;
        PreparedStatement pstmt = null;
        PreparedStatement adminStmt = null;

        String sql = "INSERT INTO users(name, phone, username, password, email,role) VALUES (?,?, ?, ?, ?, ?)";
        String getIdSql = "SELECT id FROM users WHERE username = ?";

        try {
            conn = connect.getInstance().getConnection();

            // Insert into the users table
            pstmt = conn.prepareStatement( string:sql);
            pstmt.setString( i:1, string:admin.getName());
            pstmt.setString( i:2, string:admin.getPhone());
            pstmt.setString( i:3, string:admin.getUsername());
            pstmt.setString( i:4, string:admin.getPassword());
            pstmt.setString( i:5, string:admin.getEmail());
            pstmt.setString( i:6, string:"admin");

            int affectedRows = pstmt.executeUpdate();
            if (affectedRows == 0) {
                throw new SQLException( reason:"Creating user failed, no rows affected.");
            }

            // Retrieve the generated user ID by querying using the unique username
            int userId;
            try (PreparedStatement getIdStmt = conn.prepareStatement( string:getIdSql)) {
                getIdStmt.setString( i:1, string:admin.getUsername());
                try ( var rs = getIdStmt.executeQuery()) {
                    if (rs.next()) {
                        userId = rs.getInt( string:"id");
                    } else {
                        userId = rs.getInt( string:"id");
                    } else {
                        throw new SQLException( reason:"Creating user failed, no ID obtained.");
                    }
                }
            }

            // Insert into the "admin" table with the generated user_id
            String adminSql = "INSERT INTO admins (user_id) VALUES (?)";
            adminStmt = conn.prepareStatement( string:adminSql);
            adminStmt.setInt( i:1, i1:userId);

            adminStmt.executeUpdate();
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally {
            // Close resources
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
                if (adminStmt != null) {
                    adminStmt.close();
                }

            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

```
    public static boolean insertClass(Classes classes) {
        Connection conn = null;
        PreparedStatement pstmt = null;

        String sql = "INSERT INTO classes( name,trainer_id,day, time, description,room,capacity) VALUES (?, ?, ?, ?, ?,?

        try {
            conn = connect.getInstance().getConnection();

            pstmt = conn.prepareStatement( string:sql);

            pstmt.setString( i:1, string:classes.getName());
            pstmt.setInt( i:2, i1:classes.getTrainer().getId());
            pstmt.setString( i:3, string:classes.getDay());
            pstmt.setString( i:4, string:classes.getTime());
            pstmt.setString( i:5, string:classes.getDescription());
            pstmt.setString( i:6, string:classes.getRoom());
            pstmt.setInt( i:7, i1:classes.getCapacity());

            pstmt.executeUpdate();
            return true;

        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally {
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
public static boolean insertService(Service service) {
    Connection conn = null;
    PreparedStatement pstmt = null;

    String sql = "INSERT INTO service( name,amount) VALUES (?, ?)";

    try {
        conn = connect.getInstance().getConnection();

        pstmt = conn.prepareStatement( string:sql);

        pstmt.setString( i:1, string:service.getName());
        pstmt.setDouble( i:2, d:service.getAmount());

        pstmt.executeUpdate();
        return true;

    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    } finally {
        try {
            if (pstmt != null) {
                pstmt.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```java
public static boolean insertMemberClasses(int memberId, int classesId) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    String sql = """
            INSERT INTO member_classes (member_id, classes_id)
            VALUES (?, ?)
            """;

    try {
        // Establish connection
        conn = connect.getInstance().getConnection();

        // Prepare the SQL statement
        pstmt = conn.prepareStatement( string:sql);

        // Set the parameters for the prepared statement
        pstmt.setInt( i:1,  i1:memberId);
        pstmt.setInt( i:2,  i1:classesId);

        // Execute the update
        pstmt.executeUpdate();
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    } finally {
        // Close resources
        try {
            if (pstmt != null) {
                pstmt.close();
            }
            if (conn != null) {
```

```java
public static <T> List<T> loadAll(String type) {
    List<T> records = new ArrayList<>();
    if (type.equals( anObject:"Member") || type.equals( anObject:"Trainer")) {
        String sql = getSqlQuery(type);

        if (sql != null) {
            try (
                    Connection conn = connect.getInstance().getConnection();
                    PreparedStatement pstmt = conn.prepareStatement( string:sql);
                    ResultSet rs = pstmt.executeQuery()) {

                while (rs.next()) {
                    // Get the actual `id` of the member or trainer from their respective tables
                    int id = rs.getInt(type.equals( anObject:"Member") ? "member_id" : "trainer_id");  // Use correct ID
                    String name = rs.getString( string:"name");
                    String phone = rs.getString( string:"phone");
                    String username = rs.getString( string:"username");
                    String pass = rs.getString( string:"password");
                    String email = rs.getString( string:"email");

                    if (type.equals( anObject:"Member")) {
                        String membership = rs.getString( string:"membership");
                        Member m = new Member(id, name, phone, username,  password:pass, email, membership);
                        records.add((T) m);
                    } else if (type.equals( anObject:"Trainer")) {
                        String trainershift = rs.getString( string:"trainershift");
                        Trainer t = new Trainer(id, name, phone, username,  password:pass, email, trainershift);
                        records.add((T) t);
                    }
                }
            } catch (SQLException e) {
                e.printStackTrace();
```

```java
// Helper method to get the SQL query based on type
private static String getSqlQuery(String type) {
    if (type.equals( anObject:"Member")) {
        return """
            SELECT u.id AS user_id, m.id AS member_id, u.name, u.phone, u.username, u.password, u.email,
                    m.membership
            FROM users u
            JOIN member m ON u.id = m.user_id
            """;
    } else if (type.equals( anObject:"Trainer")) {
        return """
            SELECT u.id AS user_id, t.id AS trainer_id, u.name, u.phone, u.username, u.password, u.email,
                    t.trainershift
            FROM users u
            JOIN trainer t ON u.id = t.user_id
            """;
    }
    // Add more conditions for other types as needed
    return null; // Unsupported type
}
```

```java
public static List<Classes> loadClassesByTrainer(int trainerId) {
    List<Classes> classesList = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to load all classes for a specific trainer
    String sql = """
            SELECT c.id AS class_id, c.name, c.day, c.time, c.capacity, c.Description, c.room ,c.trainer_id
            FROM classes c
            WHERE c.trainer_id = ?
            """;

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement(string:sql);
        pstmt.setInt(i:1, i1:trainerId);
        rs = pstmt.executeQuery();
        while (rs.next()) {
            int classId = rs.getInt(string:"class_id");
            String className = rs.getString(string:"name");
            String day = rs.getString(string:"day");
            String time = rs.getString(string:"time");
            int capacity = rs.getInt(string:"capacity");
            String description = rs.getString(string:"Description");
            String room = rs.getString(string:"room");

            // Get the Trainer object using the trainerId
            Trainer trainer = getTrainerById(trainerId);

            Classes c = new Classes(id:classId, day, time, name:className, description, capacity, trainer, room);

            classesList.add(e:c);
// Method to delete a service by its ID
public static boolean deleteServiceById(int serviceId) {
    Connection conn = null;
    PreparedStatement pstmt = null;

    String sql = "DELETE FROM service WHERE id = ?";

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement(string:sql);
        pstmt.setInt(i:1, i1:serviceId);
         pstmt.executeUpdate();
        return true;

    } catch (SQLException e) {
        e.printStackTrace();
        return false;  // Return false if there was an error
    } finally {
        // Close resources to prevent memory leaks
        try {
            if (pstmt != null) {
                pstmt.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }}
```

```java
public static List<Service> loadAllServices() {
    List<Service> services = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    String sql = "SELECT id, name, amount FROM service";

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        rs = pstmt.executeQuery();
        while (rs.next()) {
            int id = rs.getInt( string:"id");
            String name = rs.getString( string:"name");
            double amount = rs.getDouble( string:"amount");

            Service service = new Service(id, name, amount);
            services.add( e:service);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (pstmt != null) {
                pstmt.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
```

```java
// Method to book a service for a member by their username
public static boolean bookServiceByUsername(String username, int serviceId) {
    Connection conn = null;
    PreparedStatement pstmtMemberId = null;
    PreparedStatement pstmtBookService = null;

    String getMemberIdSql = """
            SELECT m.id AS member_id
            FROM users u
            JOIN member m ON u.id = m.user_id
            WHERE u.username = ?
            """;

    String bookServiceSql = """
            INSERT INTO member_service (member_id, service_id)
            VALUES (?, ?)
            """;

    try {
        conn = connect.getInstance().getConnection();
        //Get the member_id by username
        pstmtMemberId = conn.prepareStatement( string:getMemberIdSql);
        pstmtMemberId.setString( i:1,  string:username);
        ResultSet rs = pstmtMemberId.executeQuery();

        if (rs.next()) {
            int memberId = rs.getInt( string:"member_id");
            // Book the service by inserting into the member_service table
            pstmtBookService = conn.prepareStatement( string:bookServiceSql);
            pstmtBookService.setInt( i:1,  i1:memberId);
            pstmtBookService.setInt( i:2,  i1:serviceId);
            pstmtBookService.executeUpdate();
```

```java
// Method to get all service IDs
public static List<Integer> getAllServiceIds() {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    List<Integer> serviceIds = new ArrayList<>();

    String sql = "SELECT id FROM service";  // SQL query to get all service IDs

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        rs = pstmt.executeQuery();

        while (rs.next()) {
            int serviceId = rs.getInt( string:"id");
            serviceIds.add( e:serviceId);  // Add service ID to the list
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (pstmt != null) {
                pstmt.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```java
public static List<String> loadMemberUsernamesByRole() {
    List<String> usernames = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to load all usernames of users who have the role 'member'
    String sql = """
            SELECT u.username
            FROM users u
            WHERE u.role = 'member'
            """;

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        rs = pstmt.executeQuery();

        while (rs.next()) {
            String username = rs.getString( string:"username");
            usernames.add( e:username);  // Add each username to the list
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (pstmt != null) {
                pstmt.close();
            }
```

```java
public static int countMemberClassesByUsername(String username) {
    int classCount = 0;
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to count the number of member_classes for a specific member by username
    String sql = """
            SELECT COUNT(*) AS class_count
            FROM member_classes mc
            JOIN member m ON mc.member_id = m.id
            JOIN users u ON m.user_id = u.id
            WHERE u.username = ?
            """;

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        pstmt.setString( i:1,  string:username);
        rs = pstmt.executeQuery();

        if (rs.next()) {
            classCount = rs.getInt( string:"class_count");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (pstmt != null) {
```

```java
public static List<String> getClassNamesByUsername(String username) {
    List<String> classNames = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to get the class names for a specific member by username
    String sql = """
            SELECT c.name AS class_name
            FROM member_classes mc
            JOIN member m ON mc.member_id = m.id
            JOIN users u ON m.user_id = u.id
            JOIN classes c ON mc.classes_id = c.id
            WHERE u.username = ?
            """;

    try {
        conn = connect.getInstance().getConnection();

        pstmt = conn.prepareStatement( string:sql);
        pstmt.setString( i:1,  string:username);
        rs = pstmt.executeQuery();

        while (rs.next()) {
            String className = rs.getString( string:"class_name");
            classNames.add( e:className);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
```

```java
public static boolean insertFeedback(String username, int trainerId, String feedbackMessage) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to get the member_id based on the username
    String getMemberIdSql = "SELECT m.id FROM member m JOIN users u ON m.user_id = u.id WHERE u.username = ?";
    // SQL query to insert feedback into the feedback table
    String insertFeedbackSql = "INSERT INTO feedback (member_id, trainer_id, feedbackMessage) VALUES (?, ?, ?)";

    try {
        conn = connect.getInstance().getConnection();

        pstmt = conn.prepareStatement( string:getMemberIdSql);
        pstmt.setString( i:1,  string:username);
        rs = pstmt.executeQuery();

        if (rs.next()) {
            int memberId = rs.getInt( string:"id");

            pstmt.close();

            pstmt = conn.prepareStatement( string:insertFeedbackSql);
            pstmt.setInt( i:1,  i1:memberId);
            pstmt.setInt( i:2,  i1:trainerId);
            pstmt.setString( i:3,  string:feedbackMessage);
            pstmt.executeUpdate();

        }
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        // Error during database operation
```

```java
public static List<Classes> loadClassesByMemberId(int memberId) {
    List<Classes> classesList = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to get all the classes register by the member
    String sql = """
            SELECT c.id AS class_id, c.name AS class_name, c.day, c.time, c.capacity, c.Description, c.room,
                   t.id AS trainer_id, u.name AS trainer_name, u.phone AS trainer_phone, u.username AS trainer_usernam
                   u.password AS trainer_password, u.email AS trainer_email, t.trainershift
            FROM classes c
            JOIN member_classes mc ON c.id = mc.classes_id
            JOIN trainer t ON c.trainer_id = t.id
            JOIN users u ON t.user_id = u.id
            WHERE mc.member_id = ?
            """;

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        pstmt.setInt( i:1,  i1:memberId);
        rs = pstmt.executeQuery();

        while (rs.next()) {
            int classId = rs.getInt( string:"class_id");
            String className = rs.getString( string:"class_name");
            String day = rs.getString( string:"day");
            String time = rs.getString( string:"time");
            int capacity = rs.getInt( string:"capacity");
            String description = rs.getString( string:"Description");
            String room = rs.getString( string:"room");
```

```java
public static int getTrainerIdByUserId(int userId) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to get the trainer_id based on the user_id
    String sql = "SELECT id FROM trainer WHERE user_id = ?";

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        pstmt.setInt( i:1,  i1:userId);
        rs = pstmt.executeQuery();

        if (rs.next()) {
            return rs.getInt( string:"id");
        } else {
            System.out.println("No trainer found for user_id: " + userId);
            return -1;  // Return -1 if no trainer is found for the given user_id
        }

    } catch (SQLException e) {
        e.printStackTrace();
        return -1;  // Return -1 if an error occurs during the query
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (pstmt != null) {
                pstmt.close();
            }
            if (conn != null) {
```

```java
public static int countClassesForMember(int memberId) {
    int classCount = 0;
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    // SQL query to count the number of classes for the given member
    String sql = """
            SELECT COUNT(*) AS class_count
            FROM member_classes mc
            JOIN classes c ON mc.classes_id = c.id
            WHERE mc.member_id = ?
            """;

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        pstmt.setInt( i:1,  i1:memberId);
        rs = pstmt.executeQuery();

        if (rs.next()) {
            classCount = rs.getInt( string:"class_count");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (pstmt != null) {
                pstmt.close();
```

```
public static User authenticateUser(String username, String password) throws SQLException {
    Connection conn = null;
    PreparedStatement stmt = null;
    User user = null;
    String sql = """
            SELECT u.id AS user_id, u.name, u.phone, u.username, u.password, u.email, u.role,
                d.id AS admin_id  , m.id AS member_id, m.membership,t.id AS trainer_id, t.trainershift
            FROM users u
            LEFT JOIN member m ON u.id = m.user_id
            LEFT JOIN trainer t ON u.id = t.user_id
                LEFT JOIN admins d ON u.id = d.user_id
            WHERE u.username = ? AND u.password = ?
            """;

    try {
        conn = connect.getInstance().getConnection();
        stmt = conn.prepareStatement( string:sql);
        stmt.setString( i:1,  string:username);
        stmt.setString( i:2,  string:password);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            String name = rs.getString( string:"name");
            String phone = rs.getString( string:"phone");
            String usernameDb = rs.getString( string:"username");
            String pass = rs.getString( string:"password");
            String email = rs.getString( string:"email");
            String role = rs.getString( string:"role");

            if ("member".equals( anObject:role)) {
                String membership = rs.getString( string:"membership");
```

```
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {

            String name = rs.getString( string:"name");
            String phone = rs.getString( string:"phone");
            String usernameDb = rs.getString( string:"username");
            String pass = rs.getString( string:"password");
            String email = rs.getString( string:"email");
            String role = rs.getString( string:"role");

            if ("member".equals( anObject:role)) {
                String membership = rs.getString( string:"membership");
                int id = rs.getInt( string:"member_id");
                user = new Member(id, name, phone,  username:usernameDb,  password:pass, email, membership);
            } else if ("trainer".equals( anObject:role)) {
                String trainershift = rs.getString( string:"trainershift");
                int id = rs.getInt( string:"trainer_id");
                user = new Trainer(id, name, phone,  username:usernameDb,  password:pass, email, trainershift);
            } else if ("admin".equals( anObject:role)) {
                int id = rs.getInt( string:"admin_id");
                user = new Admin(id, name, phone,  username:usernameDb,  password:pass, email);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if (stmt != null) {
            stmt.close();
        }
    }
    return user;
```

```
public static boolean deleteMember(int memberId) {
    Connection conn = null;
    PreparedStatement memberStmt = null;
    PreparedStatement userStmt = null;

    String memberSql = "DELETE FROM member WHERE user_id = ?";
    String userSql = "DELETE FROM users WHERE id = ?";

    try {
        conn = connect.getInstance().getConnection();

        // Delete from members table
        memberStmt = conn.prepareStatement( string:memberSql);
        memberStmt.setInt( i:1,  i1:memberId);
        memberStmt.executeUpdate();

        // Delete from users table
        userStmt = conn.prepareStatement( string:userSql);
        userStmt.setInt( i:1,  i1:memberId);
        userStmt.executeUpdate();

        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    } finally {
        try {
            if (memberStmt != null) {
                memberStmt.close();
            }
            if (userStmt != null) {
                userStmt.close();
```

```
public static boolean deleteTrainer(int trainerId) {
    Connection conn = null;
    PreparedStatement trainerStmt = null;
    PreparedStatement userStmt = null;

    // SQL queries for deleting from the trainer and users tables
    String trainerSql = "DELETE FROM trainer WHERE id = ?";
    String userSql = "DELETE FROM users WHERE id = (SELECT user_id FROM trainer WHERE id = ?)";

    try {
        conn = connect.getInstance().getConnection();
        trainerStmt = conn.prepareStatement( string:trainerSql);
        trainerStmt.setInt( i:1,  i1:trainerId);
        int trainerRowsAffected = trainerStmt.executeUpdate();
        if (trainerRowsAffected == 0) {
            throw new SQLException( reason:"Deleting from trainer failed, no rows affected.");
        }

        //Delete from the users table
        userStmt = conn.prepareStatement( string:userSql);
        userStmt.setInt( i:1,  i1:trainerId);
        int userRowsAffected = userStmt.executeUpdate();
        if (userRowsAffected == 0) {
            throw new SQLException( reason:"Deleting user failed, no rows affected.");
        }

        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    } finally {
        // Close resources
```

```
public static List<Classes> loadClasses() {
    List<Classes> classes = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;

    String sql = """
            SELECT c.id AS class_id, c.name AS class_name, c.day, c.time, c.capacity, c.Description ,c.room ,
                c.trainer_id
            FROM classes c
            """;

    try {
        conn = connect.getInstance().getConnection();
        pstmt = conn.prepareStatement( string:sql);
        ResultSet rs = pstmt.executeQuery();

        while (rs.next()) {
            int classId = rs.getInt( string:"class_id");
            String className = rs.getString( string:"class_name");
            String day = rs.getString( string:"day");
            String time = rs.getString( string:"time");
            int capacity = rs.getInt( string:"capacity");
            String description = rs.getString( string:"Description");
            String room = rs.getString( string:"room");

            int trainerId = rs.getInt( string:"trainer_id");

            // Get the Trainer object by trainer_id using the getTrainerById method
            Trainer trainer = getTrainerById(trainerId);
            if (trainer != null) {
                Classes c = new Classes( id:classId, day, time,  name:className, description, capacity, trainer, room);
                classes.add( e:c);
```

```
    public static boolean updateAdmin(Admin admin) {
        Connection conn = null;
        PreparedStatement stmt = null;

        // SQL Queries
        String getUserIdSql = "SELECT user_id FROM admins WHERE id = ?";
        String updateUserSql = "UPDATE users SET name = ?, phone = ?, email = ?, username = ?, password = ? WHERE id = ?"

        try {
            conn = connect.getInstance().getConnection();

            // Get the user_id from the admins table using the admin_id
            PreparedStatement getUserIdStmt = conn.prepareStatement(string:getUserIdSql);
            getUserIdStmt.setInt(i:1, i1:admin.getId());
            ResultSet rs = getUserIdStmt.executeQuery();

            if (rs.next()) {
                int userId = rs.getInt(string:"user_id");

                // Update the users table with the new details
                PreparedStatement updateUserStmt = conn.prepareStatement(string:updateUserSql);
                updateUserStmt.setString(i:1, string:admin.getName());
                updateUserStmt.setString(i:2, string:admin.getPhone());
                updateUserStmt.setString(i:3, string:admin.getEmail());
                updateUserStmt.setString(i:4, string:admin.getUsername());
                updateUserStmt.setString(i:5, string:admin.getPassword());
                updateUserStmt.setInt(i:6, i1:userId);
                updateUserStmt.executeUpdate();

                return true;
            } else {
                return false;
```

```
    public static boolean updateMember(Member member) {
        Connection conn = null;
        PreparedStatement memberStmt = null;

        String getUserIdSql = "SELECT user_id FROM member WHERE id = ?";
        String updateUserSql = "UPDATE users SET name = ?, phone = ?, email = ?, username = ? , password = ? WHERE id = ?";
        String updateMemberSql = "UPDATE member SET membership = ? WHERE id = ?";

        try {
            conn = connect.getInstance().getConnection();
            // Get the user_id from the member table using the member_id
            PreparedStatement getUserIdStmt = conn.prepareStatement(string:getUserIdSql);
            getUserIdStmt.setInt(i:1, i1:member.getId());
            ResultSet rs = getUserIdStmt.executeQuery();

            if (rs.next()) {
                int userId = rs.getInt(string:"user_id");

                // Update the users table with the new details
                PreparedStatement updateUserStmt = conn.prepareStatement(string:updateUserSql);
                updateUserStmt.setString(i:1, string:member.getName());
                updateUserStmt.setString(i:2, string:member.getPhone());
                updateUserStmt.setString(i:3, string:member.getEmail());
                updateUserStmt.setString(i:4, string:member.getUsername());
                updateUserStmt.setString(i:5, string:member.getPassword());
                updateUserStmt.setInt(i:6, i1:userId);
                updateUserStmt.executeUpdate();

                //Update the membership in the member table
                PreparedStatement updateMemberStmt = conn.prepareStatement(string:updateMemberSql);
                updateMemberStmt.setString(i:1, string:member.getMembership());
                updateMemberStmt.setInt(i:2, i1:member.getId());
```

```
    public static boolean updateTrainer(Trainer trainer) {
        String getUserIdSql = "SELECT user_id FROM trainer WHERE id = ?";
        String updateUserSql = "UPDATE users SET name = ?, phone = ?, email = ?, username = ?, password = ? WHERE id = ?";
        String updateTrainerSql = "UPDATE trainer SET trainershift = ? WHERE id = ?";


        try (Connection conn = connect.getInstance().getConnection();
                PreparedStatement getUserIdStmt = conn.prepareStatement(string:getUserIdSql);
                PreparedStatement updateUserStmt = conn.prepareStatement(string:updateUserSql);
                PreparedStatement updateTrainerStmt = conn.prepareStatement(string:updateTrainerSql)
                ) {
            System.out.println("Updating trainer with ID: " + trainer.getId());

            // Get the user_id from the trainer table using the trainer_id
            getUserIdStmt.setInt(i:1, i1:trainer.getId());
            try (ResultSet rs = getUserIdStmt.executeQuery()) {
                if (rs.next()) {
                    int userId = rs.getInt(string:"user_id");
                    System.out.println("Found user_id: " + userId);

                    // Update the users table with the new details
                    updateUserStmt.setString(i:1, string:trainer.getName());
                    updateUserStmt.setString(i:2, string:trainer.getPhone());
                    updateUserStmt.setString(i:3, string:trainer.getEmail());
                    updateUserStmt.setString(i:4, string:trainer.getUsername());
                    updateUserStmt.setString(i:5, string:trainer.getPassword());
                    updateUserStmt.setInt(i:6, i1:userId);
                    updateUserStmt.executeUpdate();

                    // Update the trainer shift in the trainer table
                    updateTrainerStmt.setString(i:1, string:trainer.getTrainershift());
                    updateTrainerStmt.setInt(i:2, i1:trainer.getId());
```

```java
public static List<Feedback> loadFeedbackByMemberId(int memberId) {
    List<Feedback> feedbackList = new ArrayList<>();

    String sql = "SELECT id, member_id, trainer_id, feedbackMessage FROM feedback WHERE member_id = ?";

    try (Connection conn = connect.getInstance().getConnection();
         PreparedStatement pstmt = conn.prepareStatement(string:sql)) {
        pstmt.setInt(i:1, i1:memberId);
        ResultSet rs = pstmt.executeQuery();

        Member member = getMemberById(memberId);
        if (member == null) {
            System.out.println("No member found with ID: " + memberId);
            return feedbackList; // Return empty list if member is not found
        }
        while (rs.next()) {
            int feedbackId = rs.getInt(string:"id");
            int trainerId = rs.getInt(string:"trainer_id");
            String feedbackMessage = rs.getString(string:"feedbackMessage");

            Trainer trainer = getTrainerById(trainerId);

            Feedback feedback = new Feedback(id:feedbackId, member, trainer, feedbackMessage);
            feedbackList.add(e:feedback);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

```java
public static List<String> getFeedbackForMemberById(int memberId) {
    List<String> feedbackMessages = new ArrayList<>();
    String query = "SELECT f.feedbackMessage, u.name AS trainer_name "
            + "FROM feedback f "
            + "JOIN member m ON f.member_id = m.id "
            + "JOIN trainer t ON f.trainer_id = t.id "
            + "JOIN users u ON t.user_id = u.id "
            + "WHERE m.id = ?";

    try {
        Connection conn = connect.getInstance().getConnection();

        PreparedStatement preparedStatement = conn.prepareStatement(string:query);

        preparedStatement.setInt(i:1, i1:memberId);
        ResultSet resultSet = preparedStatement.executeQuery();

        while (resultSet.next()) {
            String feedbackMessage = resultSet.getString(string:"feedbackMessage");
            String trainerName = resultSet.getString(string:"trainer_name");
            feedbackMessages.add(" Trainer: " + trainerName + " - Feedback message : '" + feedbackMessage +"'");
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
    return feedbackMessages;
}
```

```java
public static Member getMemberById(int memberId) {
    String sql = """
                 SELECT m.id AS member_id, u.id AS user_id, u.name, u.phone,
                        u.username, u.password, u.email, m.membership
                 FROM member m
                 JOIN users u ON m.user_id = u.id
                 WHERE m.id = ?
                 """;

    try {
        Connection conn = connect.getInstance().getConnection();

        PreparedStatement pstmt = conn.prepareStatement(sql);

        pstmt.setInt(1, memberId);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            int userId = rs.getInt("user_id");
            String name = rs.getString("name");
            String phone = rs.getString("phone");
            String username = rs.getString("username");
            String password = rs.getString("password");
            String email = rs.getString("email");
            String membership = rs.getString("membership");
            return new Member(userId, name, phone, username, password, email, membership);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
```

```java
public static Trainer getTrainerById(int trainerId) {
    String sql = """
                 SELECT u.id AS user_id, u.name, u.phone,
                        u.username, u.password, u.email, t.trainershift
                 FROM trainer t
                 JOIN users u ON u.id = t.user_id
                 WHERE t.id = ?
                 """;

    try {
        Connection conn = connect.getInstance().getConnection();

        PreparedStatement pstmt = conn.prepareStatement(sql);

        pstmt.setInt(1, trainerId);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            int userId = rs.getInt("user_id");
            String name = rs.getString("name");
            String phone = rs.getString("phone");
            String username = rs.getString("username");
            String password = rs.getString("password");

            String email = rs.getString("email");
            String trainerShift = rs.getString("trainershift");
            return new Trainer(userId, name, phone, username, password, email, trainershift:trainerShift);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (Exception e) {
```

```java
public static int getMemberCount() {
    int count = 0;

    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    String sql = "SELECT COUNT(*) FROM member";

    try {
        conn = connect.getInstance().getConnection();
        stmt = conn.prepareStatement(sql);
        rs = stmt.executeQuery();

        if (rs.next()) {
            count = rs.getInt(1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    return count;
```

```java
public static int getTrainerCount() {
    int count = 0;

    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    String sql = "SELECT COUNT(*) FROM trainer";

    try {
        conn = connect.getInstance().getConnection();

        stmt = conn.prepareStatement( string:sql);
        rs = stmt.executeQuery();

        if (rs.next()) {
            count = rs.getInt( i:1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (stmt != null) {
                stmt.close();
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```java
public static int getClassCount() {
    int count = 0;

    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet rs = null;
    String sql = "SELECT COUNT(*) FROM classes";  // SQL query to count classes

    try {
        conn = connect.getInstance().getConnection();

        stmt = conn.prepareStatement( string:sql);
        rs = stmt.executeQuery();
        if (rs.next()) {
            count = rs.getInt( i:1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
            if (stmt != null) {
                stmt.close();
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```java
public static Admin getAdminById(int adminId) {
String sql = """
             SELECT a.id AS admin_id, u.id AS user_id, u.name, u.phone,
                    u.username, u.password, u.email, u.role
             FROM admins a
             JOIN users u ON a.user_id = u.id
             WHERE a.id = ?
             """;

try {
    Connection conn = connect.getInstance().getConnection();

    PreparedStatement pstmt = conn.prepareStatement( string:sql);

    pstmt.setInt( i:1,  i1:adminId);
    ResultSet rs = pstmt.executeQuery();

    if (rs.next()) {
        int userId = rs.getInt( string:"user_id");
        String name = rs.getString( string:"name");
        String phone = rs.getString( string:"phone");
        String username = rs.getString( string:"username");
        String password = rs.getString( string:"password");
        String email = rs.getString( string:"email");
        return new Admin( id:userId, name, phone, username, password, email);
    } else {
        System.out.println("No admin found with ID: " + adminId);
    }
} catch (SQLException e) {
    e.printStackTrace();
} catch (Exception e) {
```