COMP 3005

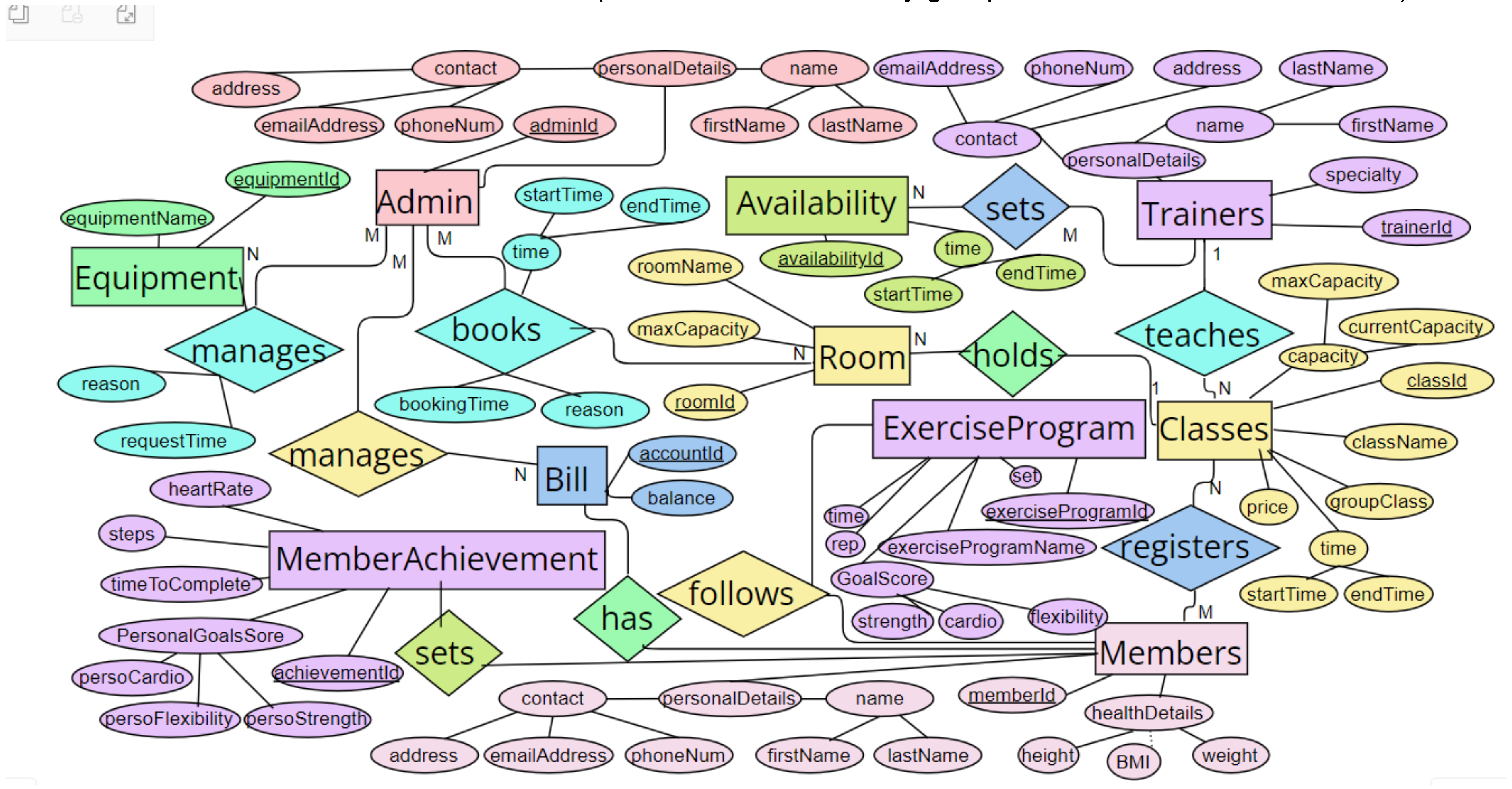
Final Project Report

Health and Fitness Club Management System

Group Number: 90

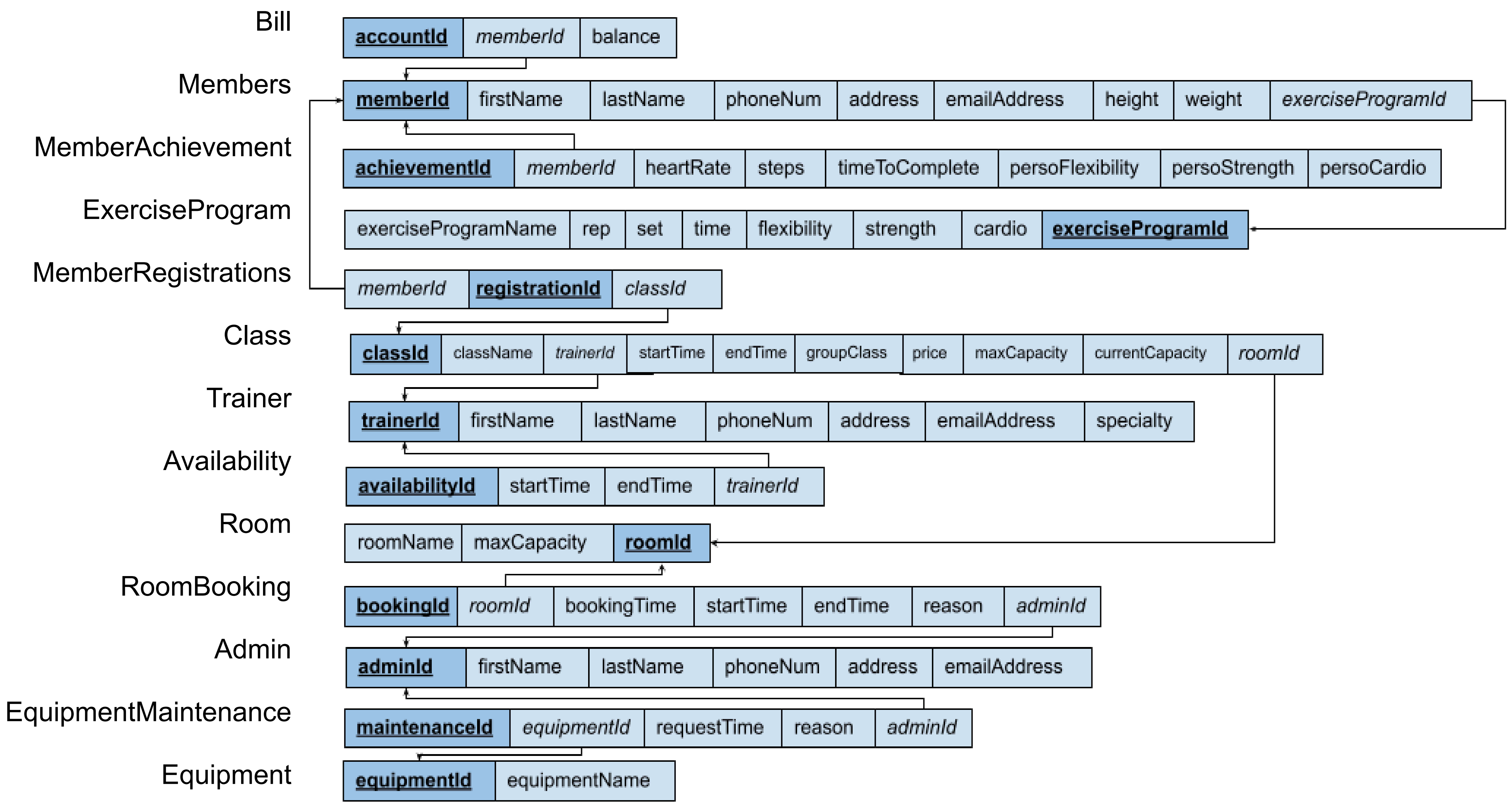
Name: Anya Gokulsing
Student ID: 101170595
Date : 10 April 2024
Course: COMP 3005 B

-- ER Model for the Health and Fitness Club (colour-coded to visually group the attributes of each relation)



Link :
https://online.visual-paradigm.com/app/diagrams/#diagram:workspace=hsuqeiwr&proj=0&id=1&type=ERDiagram&width=11&height=8.5&unit=inch

-- Relational Schema


Bill: accountId | memberId | balance
Members: memberId | firstName | lastName | phoneNum | address | emailAddress | height | weight | exerciseProgramId
MemberAchievement: achievementId | memberId | heartRate | steps | timeToComplete | persoFlexibility | persoStrength | persoCardio
ExerciseProgram: exerciseProgramName | rep | set | time | flexibility | strength | cardio | exerciseProgramId
MemberRegistrations: memberId | registrationId | classId
Class: classId | className | trainerId | startTime | endTime | groupClass | price | maxCapacity | currentCapacity | roomId
Trainer: trainerId | firstName | lastName | phoneNum | address | emailAddress | specialty
Availability: availabilityId | startTime | endTime | trainerId
Room: roomName | maxCapacity | roomId
RoomBooking: bookingId | roomId | bookingTime | startTime | endTime | reason | adminId
Admin: adminId | firstName | lastName | phoneNum | address | emailAddress
EquipmentMaintenance: maintenanceId | equipmentId | requestTime | reason | adminId
Equipment: equipmentId | equipmentName

Notes on the modelling:
Both accountId and memberId in the Bill table because the payment system might use different values for both even if in my example they happen to be the same
BMI is a calculated value (BMI = weight / (height * height))
Assume the exercise program is a print out and is hence considered single valued for the system
The personal goals and exercise program goals are strength, cardio and flexibility scores on a scale of 0 to 10 which is defined by the user and the program respectively
Dates are stored by using timestamps to store the startTime and endTime
Assuming that for personal classes no rooms need to be booked
Assuming only Admins can book rooms and create equipment maintenance holds
Assuming equipment id can be a serial number but for the UI I chose actual ints
Since start and end times are timestamps they preclude dates (demo purposes)

```
--DDL to create database

CREATE TABLE Admin (
    adminId SERIAL PRIMARY KEY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    phoneNum VARCHAR(20),
    address VARCHAR(255),
    emailAddress VARCHAR(255)
);
CREATE TABLE ExerciseProgram (
    exerciseProgramId SERIAL PRIMARY KEY,
    exerciseProgramName TEXT,
    rep INT,
    set INT,
    time INT,
    flexibility DOUBLE PRECISION,
    strength DOUBLE PRECISION,
    cardio DOUBLE PRECISION
```

```
);
CREATE TABLE Members(
    memberId SERIAL PRIMARY KEY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    phoneNum VARCHAR(20),
    address VARCHAR(255),
    emailAddress VARCHAR(255),
    height DOUBLE PRECISION,
    weight DOUBLE PRECISION,
    exerciseProgramId INT REFERENCES ExerciseProgram(exerciseProgramId )
);
CREATE TABLE MemberAchievement(
    achievementid SERIAL PRIMARY KEY,
    memberId INT REFERENCES Members(memberId),
    steps INT,
    heartRate DOUBLE PRECISION,
    timeToComplete INT,
    persoflexibility DOUBLE PRECISION,
    persostrength DOUBLE PRECISION,
    persocardio DOUBLE PRECISION
);
CREATE TABLE Trainer (
    trainerId SERIAL PRIMARY KEY,
    firstName VARCHAR(255),
    lastName VARCHAR(255),
    phoneNum VARCHAR(20),
    address VARCHAR(255),
    emailAddress VARCHAR(255),
    specialty VARCHAR(255)
);
CREATE TABLE Availability(
     availabilityId SERIAL PRIMARY KEY,
     startTime TIMESTAMP,
     endTime TIMESTAMP,
     trainerId INT REFERENCES Trainer(trainerId)
);
CREATE TABLE Room (
    roomId SERIAL PRIMARY KEY,
    roomName VARCHAR(255),
    maxcapacity INT
);
CREATE TABLE Equipment (
    equipmentId SERIAL PRIMARY KEY,
    equipmentName VARCHAR(255)
);
CREATE TABLE RoomBooking (
    bookingId SERIAL PRIMARY KEY,
    roomId INT REFERENCES Room(roomId),
    startTime TIMESTAMP,
    endTime TIMESTAMP,
    bookingTime TIMESTAMP,
    reason VARCHAR(255),
    adminId INT REFERENCES Admin(adminId)
);
CREATE TABLE EquipmentMaintenance (
    maintenance SERIAL PRIMARY KEY,
    equipmentId INT REFERENCES Equipment(equipmentId),
    requestTime TIMESTAMP,
    reason VARCHAR(255),
    adminId INT REFERENCES Admin(adminId)
 );
CREATE TABLE Classes(
    classId SERIAL PRIMARY KEY,
    className VARCHAR(255),
    trainerId INT,
    roomId INT,
    startTime TIMESTAMP,
    endTime TIMESTAMP,
    price DOUBLE PRECISION,
    maxcapacity INT,
    currentcapacity INT,
    exerciseProgramid INT,
    groupClass BOOL,
    FOREIGN KEY (trainerId) REFERENCES Trainer(trainerId),
    FOREIGN KEY (roomId) REFERENCES Room(roomId),
    FOREIGN KEY (exerciseProgramid) REFERENCES ExerciseProgram(exerciseProgramid)
);
CREATE TABLE Bill (
    accountId SERIAL PRIMARY KEY,
    memberId INT REFERENCES Members(memberId),
    balance DOUBLE PRECISION
);
CREATE TABLE MemberRegistrations(
    registrationId  SERIAL PRIMARY KEY,
    memberId INT REFERENCES Members(memberId),
    classId INT REFERENCES Classes(classId)
);
-- DML to INSERT data
-- Admin
INSERT INTO Admin (firstName, lastName, phoneNum, address, emailAddress)
VALUES
    ('John', 'Doe', '123-456-7890', '123 Main St, Anytown, USA', 'johndoe@example.com'),
    ('Jane', 'Smith', '987-654-3210', '456 Elm St, Another Town, USA', 'janesmith@example.com');

-- ExerciseProgram
```

```sql
INSERT INTO ExerciseProgram (exerciseProgramName, rep, set, time, flexibility, strength, cardio)
VALUES
    ('Stretching', 10, 3, 30, 8.0, 50.0, 20.0),
    ('Strength', 10, 3, 30, 8.0, 50.0, 20.0),
    ('Proprioception', 10, 3, 30, 8.0, 50.0, 20.0),
    ('Cardio', 10, 3, 30, 8.0, 50.0, 20.0);

-- Members
INSERT INTO Members (firstName, lastName, phoneNum, address, emailAddress, height, weight, exerciseProgramId)
VALUES
    ('Alice', 'Johnson', '555-123-4567', '789 Oak St, Cityville, USA', 'alice@example.com', 65.5, 140, 1),
    ('Bob', 'Smith', '555-987-6543', '321 Pine St, Villagetown, USA', 'bob@example.com', 70.2, 175, 2);

-- MemberAchievement
INSERT INTO MemberAchievement (memberId, steps, heartRate, timeToComplete, persoflexibility, persostrength, persocardio)
VALUES
    (1, 8000, 120.5, 30, 0.75, 0.85, 0.45),
    (2, 10000, 130.2, 45, 0.65, 0.75, 0.85);

-- Trainer
INSERT INTO Trainer (firstName, lastName, phoneNum, address, emailAddress, specialty)
VALUES
    ('Michael', 'Brown', '333-555-9999', '555 Maple Ave, Townsville, USA', 'michael@example.com', 'Personal Training'),
    ('Sarah', 'Lee', '444-666-7777', '222 Birch St, Countryside, USA', 'sarah@example.com', 'Yoga');


-- Availability
INSERT INTO Availability (startTime, endTime, trainerId)
VALUES
    ('2024-04-01 09:00:00', '2024-04-01 09:59:00',1),
    ('2024-04-01 10:00:00', '2024-04-01 10:59:00',1),
    ('2024-04-01 11:00:00', '2024-04-01 11:59:00',2),
    ('2024-04-01 12:00:00', '2024-04-01 12:59:00',2),
    ('2024-04-01 13:00:00', '2024-04-01 13:59:00',1),
    ('2024-04-01 14:00:00', '2024-04-01 14:59:00',2),
    ('2024-04-01 15:00:00', '2024-04-01 15:59:00',1),
    ('2024-04-01 16:00:00', '2024-04-01 16:59:00',2),
    ('2024-04-01 17:00:00', '2024-04-01 17:59:00',2);

-- Room
INSERT INTO Room (roomName, maxcapacity)
VALUES
    ('Studio A', 20),
    ('Studio B', 15);

-- Equipment
INSERT INTO Equipment (equipmentName)
VALUES
    ('Treadmill'),
    ('Flying machine');

-- RoomBooking
INSERT INTO RoomBooking (roomId, startTime, endTime, bookingTime, reason, adminId)
VALUES
    (1, '2024-04-01 10:00:00', '2024-04-01 11:00:00', '2024-04-01 09:30:00', 'Yoga class', 1),
    (2, '2024-04-01 15:00:00', '2024-04-01 16:00:00', '2024-04-01 14:30:00', 'Strength training', 2);

-- EquipmentMaintenance
INSERT INTO EquipmentMaintenance (equipmentId, requestTime, reason, adminId)
VALUES
    (1, '2024-04-01 09:30:00', 'Maintenance', 1),
    (2, '2024-04-01 14:30:00', 'Maintenance', 2);

-- Classes
```

```
INSERT INTO Classes (className, trainerId, roomId, startTime, endTime, price, maxcapacity, groupClass)
VALUES
    ('Yoga Class', 1, 1, '2024-04-01 10:00:00', '2024-04-01 11:00:00', 15.0, 20, TRUE),
    ('Strength Training', 2, 2, '2024-04-01 15:00:00', '2024-04-01 16:00:00', 20.0, 15, FALSE);

-- MemberRegistrations
INSERT INTO MemberRegistrations (memberId, classId)
VALUES
    (1, 1),
    (2, 2);

-- Bill
INSERT INTO Bill (memberId, balance)
VALUES
    (1, 100.0),
    (2, 75.0);
```

Member functions:

User Registration to add a new user:

```
INSERT INTO Members (firstName, lastName, phoneNum, address, emailAddress, height, weight, exerciseProgramId, achievementid,
strength, flexibility, cardio) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

Profile Management:

Updating personal information:

```
UPDATE Members SET firstName = ?, lastName = ?, phoneNum = ?, address = ?, emailAddress = ? WHERE memberId = ?;
```

Updating fitness goals (i.e., changing the exercise program):

```
UPDATE Members SET heartrate =?, persostrength =? , persoflexibility =? ,persocardio =?, steps =? , timetocomplete =? WHERE
memberId = ?;
```

Updating health metrics (i.e., strength, flexibility, cardio):

```
UPDATE Members SET weight = ? WHERE memberId = ?;
UPDATE Members SET height = ? WHERE memberId = ?;
```

Dashboard Display:
Displaying exercise program:

```
SELECT * FROM ExerciseProgram
WHERE exerciseProgramId = (SELECT exerciseProgramId FROM Members WHERE memberId = ?);
```

Displaying fitness achievements:

```
SELECT * FROM MemberAchievement WHERE memberId = ?;
```

Displaying health statistics:

```
-- calculate the BMI using the height and weight on the front end
SELECT height, weight FROM Members WHERE memberId = ?;
```

Schedule Management:
Display classes member registered in:

```
SELECT c.classId, c.currentcapacity, c.startTime, c.endTime, c.trainerId, c.roomId, c.groupClass
FROM MemberRegistrations mr
JOIN Classes c ON mr.classId = c.classId
WHERE mr.memberId = ?;
```

Register personal training session:

```
--delete the available time slot from the trainer's availability
DELETE FROM Availability
WHERE availabilityId = (SELECT availabilityId FROM Trainer WHERE trainerId = ?);
--create a new class for personal session but do not book a room
INSERT INTO Classes (trainerId, memberId, roomId, startTime, endTime, price, maxcapacity, exerciseProgramid, group)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, false);
-- increase the bill by the class value
UPDATE Bill SET balance = balance + ?  WHERE accountId = ?;
-- register member
```

```
INSERT INTO MemberRegistrations (memberId, classId) VALUES (?,?);
```

Registering in a group fitness classes:

```
-- Check if member already registered in class or not
SELECT * FROM MemberRegistrations WHERE memberId = ? AND classId = ?;
-- Get class capacity and price
SELECT price, currentcapacity, maxcapacity FROM Classes WHERE classId = ?;
-- if can register then insert
INSERT INTO MemberRegistrations (memberId, classId) VALUES (?,?);
– increase the balance of member using the price from above
UPDATE Bill SET balance = balance + ? WHERE accountId = ? ;
```

Delete a Class (group or pero training session) from Member’s registrations:

```
DELETE FROM MemberRegistrations WHERE memberId = ? AND classId = ?;
-- update the class details
UPDATE Classes SET currentcapacity = currentcapacity - 1 WHERE classId = ?;
```

Trainer Functions:

Schedule Management (Trainer can set the time for which they are available in the Availability table.)

```
INSERT INTO Availability (startTime, endTime, trainerId) VALUES (?, ?,?);
```

Trainer can See all the available Trainers at a specific time

```
SELECT Availability.availabilityid, Availability.starttime, Availability.endtime, Trainer.firstName, Trainer.lastName, Trainer.trainerId
FROM Trainer JOIN Availability
ON Trainer.trainerId = Availability.trainerId
WHERE startTime >= ? AND endTime <= ?;
```

Trainer can Search by Member’s name to see all their attributes + their exercise program

```
SELECT Members.*, ExerciseProgram.*
FROM Members JOIN ExerciseProgram ON Members.exerciseProgramId = ExerciseProgram.exerciseProgramId
```

WHERE firstName = ? AND lastName = ?;

Administrative Staff Functions:
Room Booking (book a room for a period of time and set the reason)

```
INSERT INTO RoomBooking (roomId, startTime, endTime, bookingTime, reason, adminId)
VALUES (?, ? , ?, NOW(), ?, ?);
```

View the history (i.e., timestamp + reason) of the bookings for a room

```
SELECT bookingid, bookingTime, reason, adminId
FROM RoomBooking
WHERE roomId = ?;
```

Equipment Maintenance Monitoring (create an entry in EquipmentMaintenance and add as reason “Maintenance”)

```
INSERT INTO EquipmentMaintenance (equipmentId, requestTime, reason, adminId)
VALUES (?, NOW(), 'Maintenance', ?);
```

See all the maintenance history from the EquipmentMaintenance table

```
SELECT *
FROM EquipmentMaintenance
WHERE equipmentId = ?;
```

Class Schedule Updating:
To create a new group class assuming Trainer said they are free and room is free

```
-- first check if room available by doing a SELECT * FROM RoomBooking WHERE roomId = ? AND (? BETWEEN startTime AND endTime
OR ? BETWEEN startTime AND endTime)

-- then add room booking if room available
INSERT INTO RoomBooking (roomId, startTime, endTime, bookingTime, reason, adminId)
VALUES (?, ? , ?, NOW(), ?, ?);

-- finally create the group class
```

```
INSERT INTO Classes (startTime, endTime, className, trainerId, maxcapacity, price, roomId, groupClass, exerciseProgramId)
VALUES (?, ?, ?, ?, ?, ?, ?, 'yes', ?);
```

To get all classes for the next 24 hours

```
SELECT * FROM Classes
WHERE startTime >= ? AND endTime <= (? + INTERVAL '24 hours');
```

To unregister a member from a class if needed

```
DELETE FROM MemberRegistrations
WHERE classId = ?;
```

Pay Member Bill:

```
UPDATE Bill SET balance = 0.0 WHERE accountid = ?;
```

View Member Balance:

```
SELECT balance FROM Bill WHERE memberId = ?;
```

This project was done using PostgreSQL + Express + React + NodeJs + Bootstrap 5

Bonus:
Spotify integration using https://developer.spotify.com/
UI done using Bootstrap5 and React
More info about how to use the PERN stack can be found online
(this video helped me understand how promises work and how to connect to the DB):
https://youtu.be/ldYcgPKEZC8?si=PYujIiHKhR8m4n-s
More info about how to use Bootstrap5 can be found at https://www.w3schools.com/bootstrap5/