



6/5/2024

GYM MANAGEMENT SYSTEM

- SUBJECT: DATABASE
- CLASS: BS (SE) 2nd
- SUBMITTED TO: SIR ABDUL GHANI
- SUBMISSION DATE:05-06-2024

GROUP MEMBERS

- MUHAMMAD ATTAULLAH (233665)
- MUHAMMAD AWAIS (233680)
- AWAIS ALI (233696)

1. Project Overview

1.1. Scope of the Gym Management System

The Gym Management System aims to streamline the operations of a gym by providing a comprehensive platform for managing memberships, classes, trainers, and equipment.

1.2. Types of Services


- Membership Plans
- Personal Training Sessions
- Group Classes (Yoga, Pilates, Zumba, etc.)
- Equipment Management

1.3. Key Functionalities

- User Registration and Profile Management
- Membership Management
- Class Scheduling and Booking
- Trainer Management
- Equipment Management

2. Database Used In Project:

2.1. Database Creation and Usage

```
 create database GymManagement;  
use GymManagement;
```

2.2. Table Definitions

➤ **Users Table:**

```
---User Table
CREATE TABLE Users (
    UserID INT IDENTITY(1,1) PRIMARY KEY,
    FullName Text Not Null,
    Address TEXT,
    PhoneNmbr int,
    Email VARCHAR(100) NOT NULL,
    Username VARCHAR(50) NOT NULL,
    Password VARCHAR(255) NOT NULL,
);
```

➤ Memberships Table:

```
---Membership Table
CREATE TABLE Memberships (
    MembershipID INT IDENTITY(1,1) PRIMARY KEY,
    UserID INT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    Status VARCHAR(10) NOT NULL,
    PaymentDetails TEXT,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    CHECK (Status IN ('Active', 'Expired', 'Pending'))
);
```

➤ Classes Table:

```
---Classes Table
CREATE TABLE Classes (
    ClassID INT Identity(1,1) PRIMARY KEY,
    ClassName VARCHAR(100) NOT NULL,
    Description TEXT,
    Schedule DATETIME NOT NULL,
    TrainerID INT NOT NULL,
    FOREIGN KEY (TrainerID) REFERENCES Users(UserID)
);
```

➤ Trainers Table:

```
---Trainer Table
CREATE TABLE Trainers (
    TrainerID INT Identity(1,1) PRIMARY KEY Not null,
    Name VARCHAR(100) NOT NULL,
    Specialization VARCHAR(100),
    Availability varchar(50)
);
```

➤ Equipment Table:

```
---Equipment Table
CREATE TABLE Equipment (
    EquipmentID INT Identity(1,1) PRIMARY KEY,
    EquipmentName VARCHAR(100) NOT NULL,
    Description TEXT,
    MaintenanceDate DATE,
    Status VARCHAR(10) NOT NULL,
    Check (Status IN ('Available', 'Under Maintenance'))
);
```

2.3. SQL Queries for Data Retrieval

```
select * from Users;
select * from Memberships;
select * from Classes;
select * from Equipment;
select * from Trainers;
```

2.4. Stored Procedures

➤ Register Member Procedure

```
---Procedure 1
CREATE PROCEDURE RegisterMember
AS
(
    @ username VARCHAR(50)
    @ password VARCHAR(255)
    @ email VARCHAR(100)
    @ address TEXT
    @ phone INT
    @ fullname TEXT
)
BEGIN
    INSERT INTO Users (Username, Password, Email, Address, PhoneNمبر, FullName)
    VALUES (username, password, email, address, phone, fullname);
END ;
```

➤ **Schedule Class Procedure:**

```
---Procedure 2
CREATE PROCEDURE ScheduleClass
    @className VARCHAR(100),
    @description TEXT,
    @schedule DATETIME,
    @trainerID INT
AS
BEGIN
    INSERT INTO Classes (ClassName, Description, Schedule, TrainerID)
    VALUES (@className, @description, @schedule, @trainerID);
END;
```

2.5. Triggers

➤ **After Membership Insert Trigger**

```
---Triggers
CREATE TRIGGER after_membership_insert
ON Memberships
AFTER INSERT
AS
BEGIN
    -- Assuming there's a stored procedure to send an email
    DECLARE @UserID INT, @MembershipID INT;
    SELECT @UserID = UserID, @MembershipID = MembershipID FROM inserted;
    EXEC SendConfirmationEmail @UserID, @MembershipID;
END;
```

2.6. Data Insertion

➤ Insert Statement for User

```
---Insert Statement for user
insert into Users(FullName,Address,PhoneNmbr,Email,Username>Password) values
('MAttaullah','Jahanian',123467,'233665@stds.au.edu.pk','atta123','qwerty8317');
```

3. Database Management System (DBMS)

3.1. Selection of RDBMS

- Chosen RDBMS: MySQL for its robustness, ease of use, and wide community support.

3.2. Installation and Configuration

- Installed MySQL on a local development environment.

3.3. SQL Utilization

- Created tables, defined relationships, and managed data using SQL.

4. Database Design

4.1. Detailed Documentation

- Project Overview:** Describes the scope, services, and key functionalities of the Gym Management System.
- Database Schema:** Provides a detailed structure of the database including table definitions, stored procedures, triggers, and data insertion statements.

- **Entity Descriptions:** Explains each table and its attributes.

4.2. Additional Diagrams

- **Description of ERD:**

1. Users

- UserID (PK)
- Full Name
- Address
- PhoneNmbr
- Email
- Username
- Password

2. Memberships

- MembershipID (PK)
- UserID (FK to Users)
- StartDate
- EndDate
- Status
- PaymentDetails

3. Classes

- ClassID (PK)
- ClassName
- Description
- Schedule
- TrainerID (FK to Trainers)

4. Trainers

- TrainerID (PK)
- Name

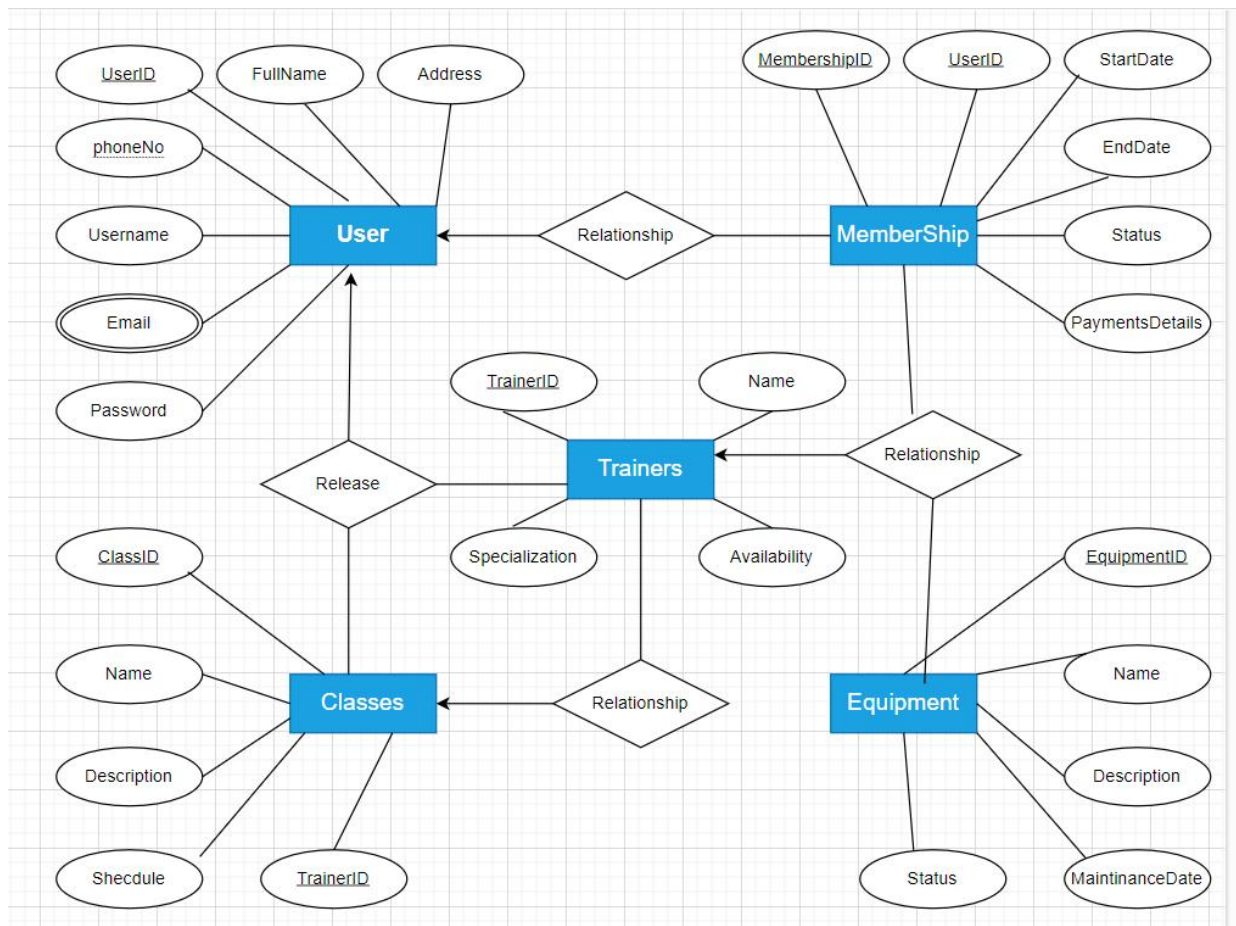
Gym Management System

- Specialization
- Availability

5. Equipment

- EquipmentID (PK)
- EquipmentName
- Description
- MaintenanceDate
- Status

5. ER Diagram



- **4.2. Deployment Instructions**
- Install MySQL on the development environment.

- Execute the provided SQL script to create and populate the database.
- Use SQL tools or command-line interface to interact with the database.

5. Testing and Validation

5.1. Test Cases

- User Registration
- Membership Management
- Class Scheduling
- Trainer Management
- Equipment Management

5.2. Comprehensive Testing

- Ensured data accuracy and system performance.
- Verified error handling mechanisms.

5.3. Automated Testing

- Used applicable testing frameworks for automated testing.

6. Documentation

6.1. Detailed Documentation

- Provided detailed documentation, including project overview, database schema, and entity descriptions.
- **Project Overview:** Describes the scope, services, and key functionalities of the Gym Management System.
- **Database Schema:** Provides a detailed structure of the database including table definitions, stored procedures, triggers, and data insertion statements.
- **Entity Descriptions:** Explains each table and its attributes.

6.2. Additional Diagrams

- Included data dictionaries, ER diagrams, SQL scripts, and other supporting diagrams.

6.3. Deployment Instructions

- Clear instructions on how to deploy and interact with the database system.

7. Presentation and Demonstration

7.1. Structured Presentation

- Created a structured presentation highlighting objectives, design choices, implementation details, and testing outcomes.

7.2. Demonstration

- Demonstrated key functionalities of the gym management system.

7.3. Discussion

- Discussed challenges faced, solutions implemented, and lessons learned.

8. Future Work

- **Enhance User Interface**

Develop a user-friendly web and mobile interface for easy interaction with the system.

- **Advanced Analytics**

Integrate advanced analytics to provide insights into membership trends, class popularity, and equipment usage.

- **AI and Machine Learning Integration**

Implement AI-driven recommendation systems for personalized class and trainer suggestions.

- **Automated Notifications**

Develop automated email and SMS notifications for class reminders, membership renewals, and equipment maintenance schedules.

- **Extended Payment Options**

Integrate multiple payment gateways to offer more flexibility in payment options.

- **Multi-location Support**

Expand the system to manage multiple gym locations with centralized data management.

- **Integration with Wearable Devices**

Sync with wearable fitness devices to track member activities and progress.

- **Enhanced Security Features**

Gym Management System

Implement advanced security measures like multi-factor authentication and encryption for data protection.

- **Social Features**

Add social features such as member forums, class reviews, and sharing of achievements.

- **API Development**

Develop APIs for third-party integration and to allow other developers to build apps on top of the system.

