GYM MANAGEMENT SYSTEM

Final Sprint
Semester3

Walid Jerjawi

1. User Documentation

Overview

The Gym Management System is a user-friendly, console-based application designed to help gym administrators, trainers, and members manage all core operations of a gym.

This includes:

• User registration and login
• Membership management
• Workout class management
• Role-based access and menus

The system uses a PostgreSQL database for data persistence and is built using Java, with Maven for dependency management. Users interact via a simple text menu, and all passwords are encrypted using BCrypt for secure storage.

Explanation of Classes and Their Interactions

User & Subclasses:

• User – Base class for all user types. Includes fields like username, passwordHash, email, phone number, address, and role.
• Admin, Trainer, Member – Extend from User. Their role is automatically assigned and used to determine permissions.

UserDAO / UserService:

• UserDAO – Handles all database interactions related to users.
• UserService – Provides logic for registration, login, and role-based instantiation.

Membership:

• Membership – Contains membershipType, description, cost, and userId.
• MembershipDAO – Handles CRUD operations.
• MembershipService – Business logic to purchase and view memberships.

WorkoutClass:

• WorkoutClass – Represents a gym class with fields like ID, type, description, trainerID.
• WorkoutClassDAO – Database operations for workout classes.
• WorkoutClassService – Add, update, delete, view classes.

GymApp:

• Entry point. Manages user interaction and role-specific actions.

UML Class Diagram Explanation

User (Super Class)

```
-  id:  int
-  username: String
- passwordHash: String
- email: String
- phoneNumber: String
- address: String
- role: String
---
+  User()
+ User(id: int,  username: String, passwordHash:
        String, email: String,  phoneNumber: String,
        address: String, role: String )
+ User(username: String, passwordHash:
        String, email: String,  phoneNumber: String,
        address: String, role: String )
+ User (User other)
+  getId(): int
+ getUsername(): String
+ getEmail():String
+ getPhoneNumber(): String
+ getAddress(): String
+ getRole(): String
+ setId(int): void
+ setUsername(String): void
+ setEmail(String): void
+ setPhoneNum(String): void
+toString: String
```

Admin (SubClass)

```
+  Admin()
+ Admin(username: String, passwordHash:
        String, email: String,  phoneNumber:
        String, address: String, role: String )
+  Admin(User user)
+ toString: String
```

Member (SubClass)

```
+  Member()
+Member(username: String, passwordHash:
        String, email: String,  phoneNumber:
        String, address: String, role: String )
+  Member(User user)
+ toString: String
```

Trainer (SubClass)

```
+   Trainer (()
+  Trainer (username: String, passwordHash:
        String, email: String,  phoneNumber:
        String, address: String, role: String )
+  Trainer (User user)
+ toString: String
```

Membership

```
-  membershipID:  int
-  membershipType: String
- membershipDescription: String
- membershipCost: double
- phoneNumber: String
- memberID: int
---
+  Membership()
+ Membership( membershipType: String,
membershipDescription:String,  membershipCost: double,
memberID: int)
+ Membership(membershipID: int,membershipType: String,
membershipDescription:String,  membershipCost: double,
memberID: int)
+  getMembershipID(): int
+ getMembershipType(): String
+ getMembershipDescription(): String
+ getMemberID(): int
+ setMembershipID(int): void
+ setMembershipType(String): void
+ setMembershipDescriptionl(String): void
+ setMembershipCost(String): void
+ setMemberID(int): void
+toString: String
```

WorkoutClass

```
-  workoutID:  int
-  type: String
- description: String
- trainerID: int
---
+  WorkoutClass()
+ WorkoutClass(type: String, description:  String, trainerID:
                  int)
+ WorkoutClass(workoutID: int, type: String, description:
                  String, trainerID:  int)
+ getWorkoutID(): int
+ getType(): String
+ getDescription(): String
+ getTrainerID(): int
+ setWorkoutID (int): void
+setType(String): void
+ setDescription(String): void
+ setTrainerID( int): void
+ toString: String
```

UserService

- userDAO: UserDAO

+ UserService()
+ register(User user): void
+ login(String username, String password): User
+ deleteUser(int userID): void
+ printAllUsers(): void

MembershipService

- membershipDAO: MembershipDAO

+ MembershipService()
+ purchaseMembership(Membership m): void
+ viewMyMemberships(int memberID): void
+ viewAllMemberships(): void
+ viewTotalRevenue(): void
+ viewTotalExpenses(int memberID): void

WorkoutClassService

- workoutClassDAO: WorkoutClassDAO

+ WorkoutClassService()
+ createWorkoutClass(WorkoutClass wc): void
+ viewMyWorkoutClasses(int trainerID): void
+ viewAllWorkoutClasses(): void
+ updateWorkoutClass(WorkoutClass updated): void
+ deleteWorkoutClass(int workoutID, int trainerID): void

UserDAO

- dbConnection: DBConnection

+ registerUser(User user): void
+ getUserByUsername(String username): User
+ getAllUsers(): List<User>
+ deleteUserById(int id): Boolean
+ isUsernameOrEmailTaken(String username, String email): boolean

MembershipDAO

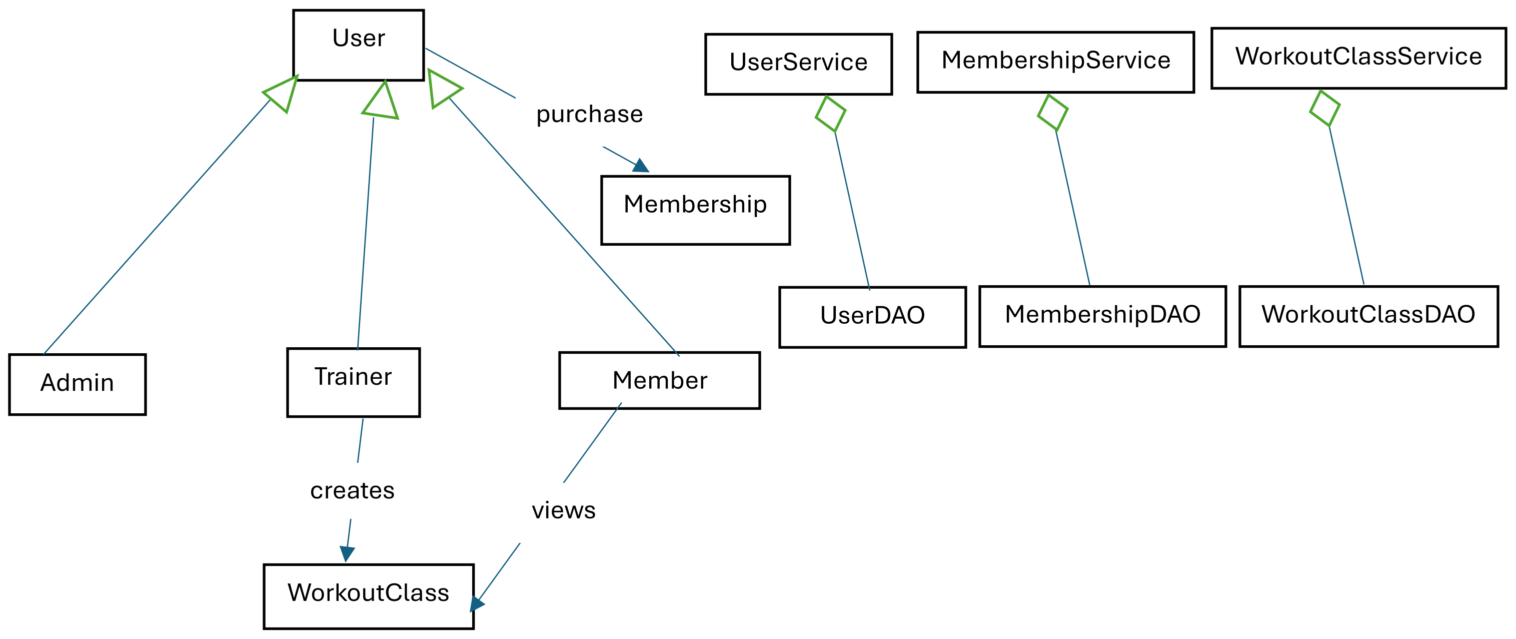
- dbConnection: DBConnection

+ addMembership(Membership membership): void
+ getMembershipsByMemberID(int id): List<Membership>
+ getAllMemberships(): List<Membership>
+ viewAllMemberships(): void
+ getTotalRevenue(): double
+ getTotalExpensesByMember(int memberId): double

WorkoutClassDAO

- dbConnection: DBConnection

+ addWorkoutClass(WorkoutClass wc): void
+ getWorkoutClassesByTrainer(int id): List<WorkoutClass>
+ getAllWorkoutClasses(): List<WorkoutClass>
+ updateWorkoutClass(WorkoutClass wc): void
+ deleteWorkoutClass(int workoutId, int trainerId): void



Explanation of UML Elements

1. User (Base Class):

- Attributes: id, username, passwordHash, email, phoneNumber, address, role
- Methods: constructors, getters/setters, toString()

2. Admin, Trainer, Member (Child Classes):

- Inherit from User
- Have their own constructors and toString()

3. Membership:

- Represents a gym membership
- Attributes: membershipID, type, description, cost, memberID
- Associated with User via memberID (aggregation)

4. WorkoutClass:

- Represents a gym class
- Attributes: workoutID, type, description, trainerID
- Associated with User (trainer) via trainerID (aggregation)

UML notations

1. Inheritance (Triangle Arrow):
- Admin, Trainer, and Member inherit from User, representing an "is-a" relationship. Each subclass shares the properties of User.

2. Aggregation (White Diamond):
- Services (UserService, MembershipService, WorkoutClassService) use DAOs via aggregation. This shows a has-a relationship without strong ownership.

3. Association (Plain Line):
- Represents relationships between objects like `User` purchasing a `Membership`, `Trainer` creating WorkoutClass, and Member viewing WorkoutClass.

How to Start and Use the System

Requirements:

• Java 21

• PostgreSQL

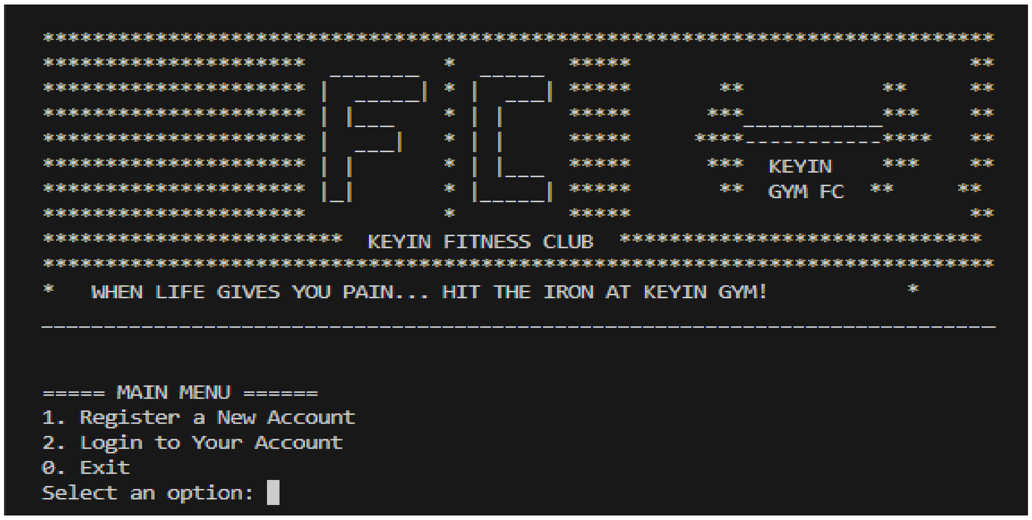
• Maven

• Git

Setup Instructions:

1. Clone the repo
2. Setup PostgreSQL and run schema.sql
3. Configure DBConnection.java with credentials
4. Run: `mvn clean compile exec:java`

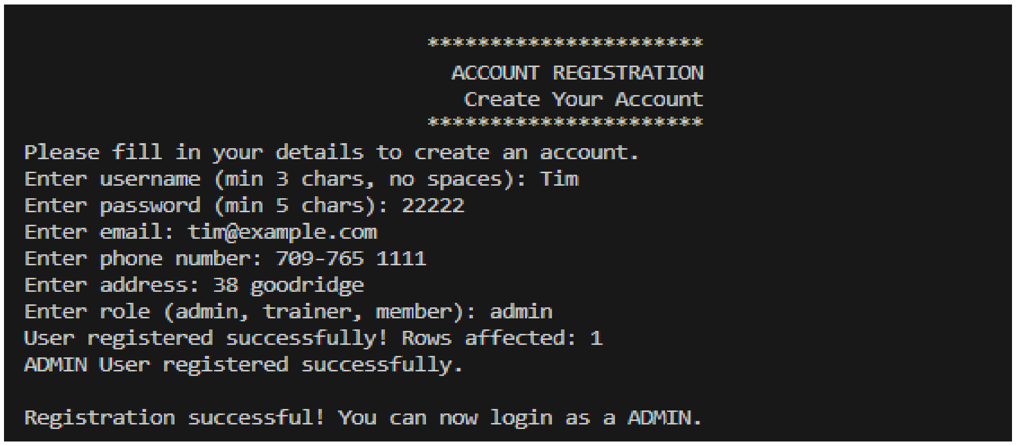
Main Menu



```
****************************************************************************
*********************   _______   *   ______   *****                     **
********************* |  _____| * |  ____| *****      **           **     **
********************* | |___     * | |       *****     ***___________***    **
********************* |  ___|    * | |       *****    ****___________****   **
********************* | |        * | |___    *****     ***  KEYIN     ***    **
********************* |_|        * |_____| *****       **   GYM FC  **      **
*********************            *           *****                          **
************************  KEYIN FITNESS CLUB  *****************************
****************************************************************************
*   WHEN LIFE GIVES YOU PAIN... HIT THE IRON AT KEYIN GYM!          *
----------------------------------------------------------------------------


===== MAIN MENU ======
1. Register a New Account
2. Login to Your Account
0. Exit
Select an option:
```

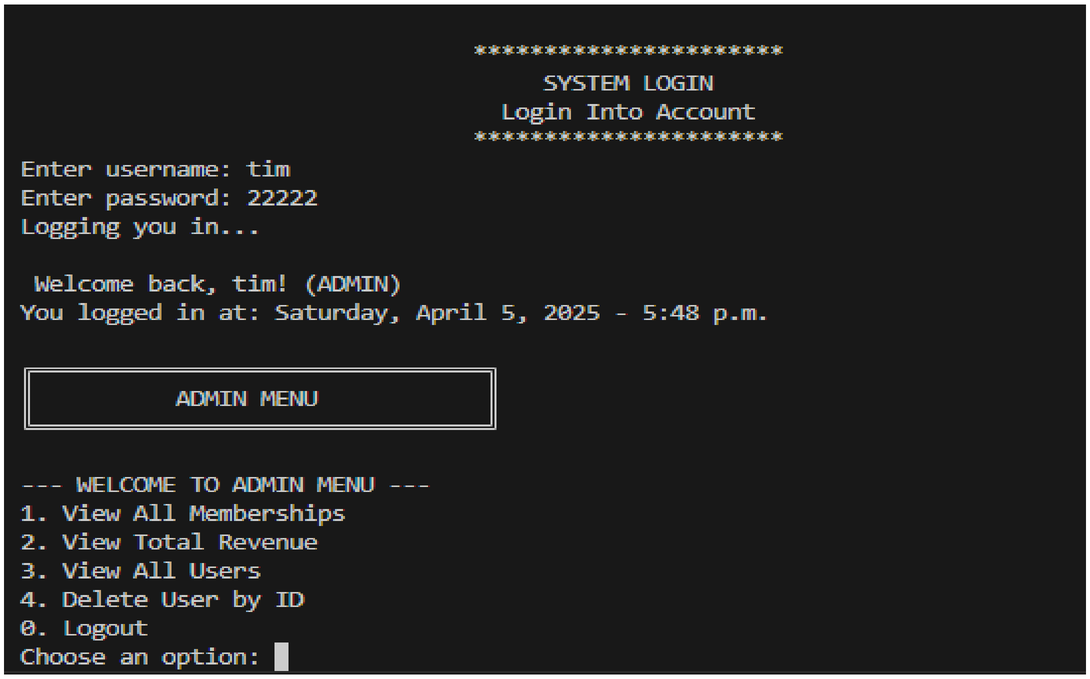
2- Registration Menue



```
                                ***********************
                                  ACCOUNT REGISTRATION
                                   Create Your Account
                                ***********************
Please fill in your details to create an account.
Enter username (min 3 chars, no spaces): Tim
Enter password (min 5 chars): 22222
Enter email: tim@example.com
Enter phone number: 709-765 1111
Enter address: 38 goodridge
Enter role (admin, trainer, member): admin
User registered successfully! Rows affected: 1
ADMIN User registered successfully.

Registration successful! You can now login as a ADMIN.
```

3- Login Menu (Admin)



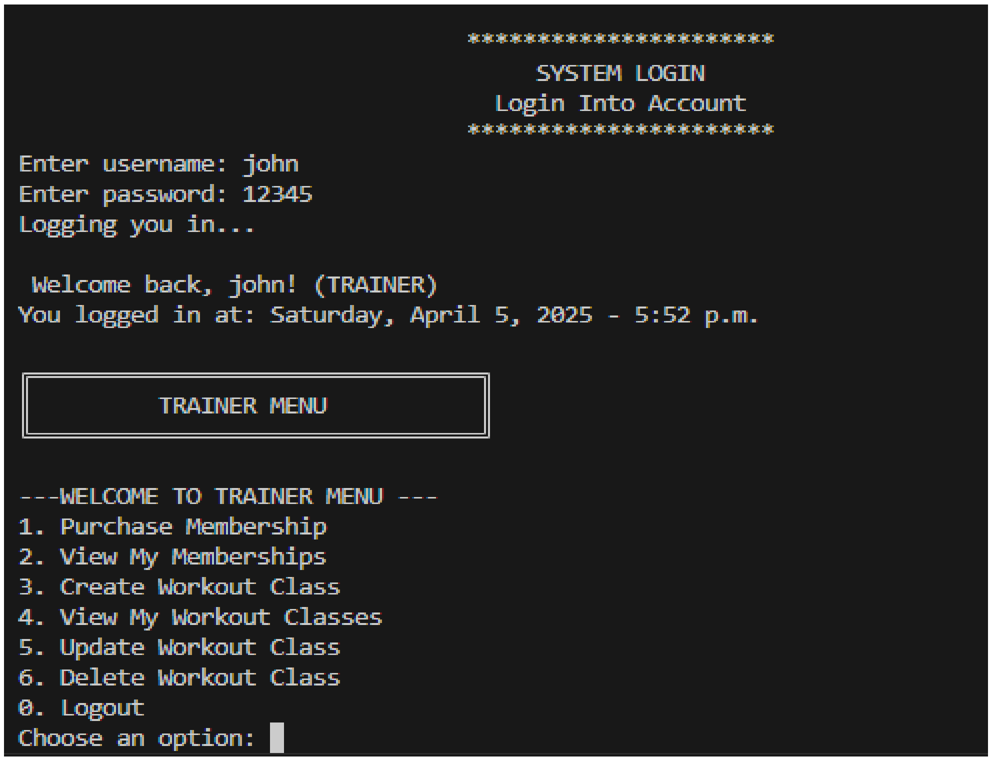
```
                                 ***********************
                                      SYSTEM LOGIN
                                   Login Into Account
                                 ***********************
Enter username: tim
Enter password: 22222
Logging you in...

 Welcome back, tim! (ADMIN)
You logged in at: Saturday, April 5, 2025 - 5:48 p.m.

 ╔═════════════════════════════════╗
 ║           ADMIN MENU            ║
 ╚═════════════════════════════════╝

--- WELCOME TO ADMIN MENU ---
1. View All Memberships
2. View Total Revenue
3. View All Users
4. Delete User by ID
0. Logout
Choose an option:
```

- Login Menu (Trainer)



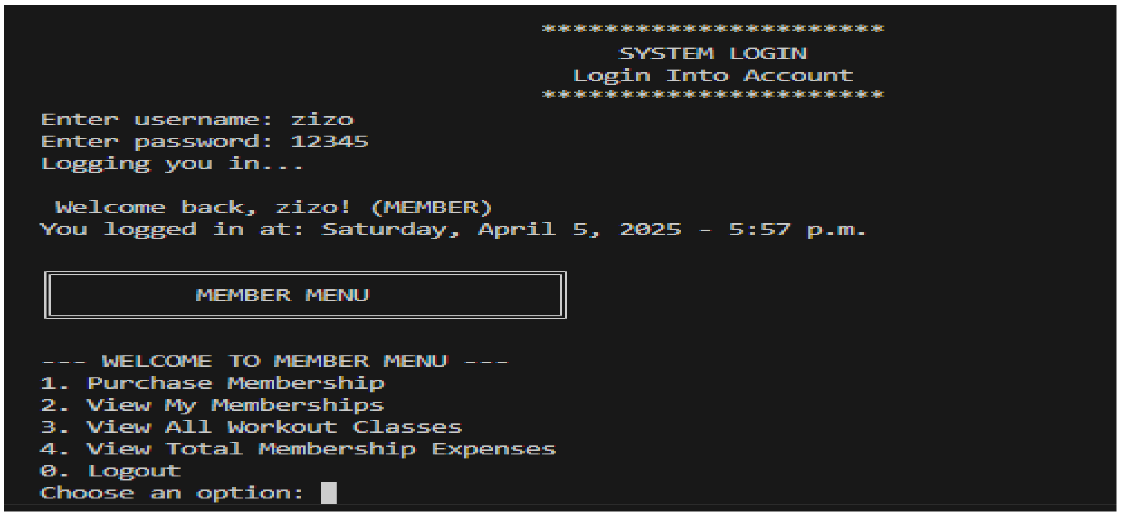
```
                                 ***********************
                                      SYSTEM LOGIN
                                   Login Into Account
                                 ***********************
Enter username: john
Enter password: 12345
Logging you in...

 Welcome back, john! (TRAINER)
You logged in at: Saturday, April 5, 2025 - 5:52 p.m.

 ╔═════════════════════════════════╗
 ║          TRAINER MENU           ║
 ╚═════════════════════════════════╝

---WELCOME TO TRAINER MENU ---
1. Purchase Membership
2. View My Memberships
3. Create Workout Class
4. View My Workout Classes
5. Update Workout Class
6. Delete Workout Class
0. Logout
Choose an option:
```

- Login Menu (Member)



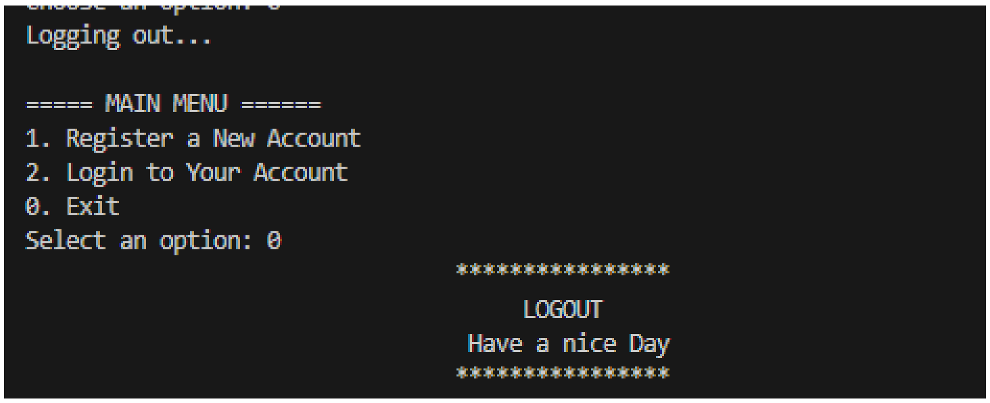
```
                                        ************************
                                             SYSTEM LOGIN
                                          Login Into Account
                                        ************************
Enter username: zizo
Enter password: 12345
Logging you in...

 Welcome back, zizo! (MEMBER)
You logged in at: Saturday, April 5, 2025 - 5:57 p.m.

╔══════════════════════════════════╗
║           MEMBER MENU            ║
╚══════════════════════════════════╝

--- WELCOME TO MEMBER MENU ---
1. Purchase Membership
2. View My Memberships
3. View All Workout Classes
4. View Total Membership Expenses
0. Logout
Choose an option:
```

4- Logout



```
Logging out...

===== MAIN MENU ======
1. Register a New Account
2. Login to Your Account
0. Exit
Select an option: 0
                             *****************
                                  LOGOUT
                               Have a nice Day
                             *****************
```

For More details about the menu watch the Video

2. Development Documentation

Javadoc Documentation

- All major classes (User, UserService, Membership, WorkoutClass) include Javadoc comments.
- Every public method (e.g., registerUser, login, purchaseMembership, viewAllMemberships) has accompanying descriptions for purpose and parameters.

Project directory structure

```
gym-management-system/
│
├── pom.xml                # Maven configuration file
└── src/
    └── main/
        └── java/
            └── com/
                └── gym/
                    ├── GymApp.java              # Main application (console UI and flow)
                    │
                    ├── database/
                    │   └── DBConnection.java       # Manages PostgreSQL DB connections
                    │
                    ├── user/
                    │   ├── User.java             # Base class for all users
                    │   ├── UserDAO.java            # Handles DB operations for users
                    │   ├── UserService.java         # Business logic for users
                    │   └── childclasses/
                    │       ├── Admin.java          # Admin subclass of User
                    │       ├── Trainer.java         # Trainer subclass of User
                    │       └── Member.java         # Member subclass of User
                    │
                    ├── memberships/
                    │   ├── Membership.java         # Model for gym memberships
                    │   ├── MembershipDAO.java       # DB operations for memberships
                    │   └── MembershipService.java    # Business logic for memberships
                    │
                    └── workoutclasses/
                        ├── WorkoutClass.java        # Model for workout classes
                        ├── WorkoutClassDAO.java      # DB operations for workout classes
                        └── WorkoutClassService.java   # Business logic for workout classes
```

Build Process

- Build Tool: **Maven**
- Java Version: 21
- Packaging: jar
- Compiler Plugin:

```xml
<plugin>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.10.1</version>
  <configuration>
    <source>21</source>
    <target>21</target>
  </configuration>
</plugin>
```

- Exec Plugin (for console launch):

```xml
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>3.1.0</version>
  <configuration>
    <mainClass>com.gym.GymApp</mainClass>
  </configuration>
</plugin>
```

Dependencies

- org.postgresql:postgresql:42.5.0
- org.mindrot:jbcrypt:0.4
- junit:junit:4.11

Setting Up Database for Development

1. Start PostgreSQL and create a new database DB1.
2. Execute this script:

```
CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone_number VARCHAR(20),
    address TEXT,
    role VARCHAR(20) NOT NULL
);

CREATE TABLE memberships (
    membership_id SERIAL PRIMARY KEY,
    membership_type VARCHAR(50),
    membership_description TEXT,
    membership_cost NUMERIC(10,2),
    member_id INT REFERENCES users(user_id
);

CREATE TABLE workout_classes (
    workout_id SERIAL PRIMARY KEY,
    workout_class_type VARCHAR(100) NOT NULL,
    workout_class_description TEXT,
    trainer_id INT REFERENCES users(user_id)
);
```

How to Clone and Run the Project

```
git clone https://github.com/your-username/gym-management-system.git
```

cd gym-management-system

```
mvn clean compile exec:java
```

The application will launch in your terminal with a menu interface.

3. Individual Report

Contribution

As the sole developer of the Gym Management System project, I was responsible for designing, implementing, and delivering all components of the system. My work included:

- Complete Application Design & Development
- Authentication & Role-Based Access
  - ➢ Implemented secure user registration and login.
  - ➢ Handled role-based redirection to Admin, Trainer, and Member interfaces.
- Database Integration
  - ➢ Designed and created the PostgreSQL schema.
  - ➢ Wrote SQL scripts and ensured foreign key relationships.
- Secure Password Handling
  - ➢ Integrated BCrypt for password hashing to enforce secure user data storage.
- CRUD Operations
  - ➢ Developed DAO and Service layers for:
    - ✓ Users
    - ✓ Memberships
    - ✓ Workout Classes

- Console User Interface
    - Built a clean and user-friendly CLI with color-coded feedback and structured menus for each role.
- Maven & Project Setup
    - Configured project with Maven for dependency management.
    - Ensured smooth build and execution with plugins like exec-maven-plugin.
- Documentation
    - Wrote extensive in-code Javadoc comments.
    - Produced full user and developer documentation with UML and structure overviews.

Challenges Faced

- PostgreSQL Setup & Connectivity
    - Initial hurdles with connecting PostgreSQL to Java using JDBC.
- Password Encryption
    - Learned to securely hash passwords using BCrypt and verify them on login.
- Maven Configuration
    - Faced and resolved plugin compatibility and dependency versioning issues.