**GYM MANAGEMENT SYSTEM  
  
USER DOCUMENTATION**

**1. Overview of the System**

The Gym Management System is a console-based Java application developed to help manage gym operations across three user types: administrators, trainers, and members. The system allows users to register and log in, and then access a menu of features tailored to their role. It supports class management, membership purchasing, and administrative tasks.

This system is designed for small to mid-sized gyms or training centers seeking a lightweight software solution for handling basic operations and data tracking without needing a full graphical interface.

**2. Application Features**

Once launched, the program has a simple text-based menu that presents users with three options: register a new account, log in to an existing account, or exit the program. When you login, the application detects the user's role (admin, trainer, or member) and presents a role-specific submenu with relevant options to choose from.

**Administrator Features**

Admins have access to full oversight capabilities. They can view a list of all users, delete accounts by username, view all the existing memberships, and review the total revenue generated by membership purchases.

**Trainer Features**

Trainers can manage workout classes. This includes adding new classes, updating existing ones, viewing their own classes, and deleting them as needed. They can also purchase gym memberships and view their own total membership expenses.

**Member Features**

Members can view all available workout classes, buy memberships, and check their total membership spending.

**3. How the System Works**

The Gym Management System is built around major features being handled by dedicated classes. These classes communicate through service layers that perform logic and validation before interacting with an underlying PostgreSQL database. Users interact with the system through text menus in the console.

**Main Control Flow**

* **GymApp.java**  
  This is the entry point of the program. It handles the main menu, user input, and sends the users to the correct menu based on their role, either admin, trainer, or member.

**User System**

* **UserService.java**  
  Handles user registration, login, and deletion. When a new user is created, their password is hashed using the jBCrypt library before it is saved to the database. When logging in, the system checks the entered password against the hashed one using the same library. This improves security so that passwords are not stored in plain text.
* **Login and Roles**  
  Users are assigned a role when they register. The system then loads different menus based on the role to allow them to access the appropriate features.

**Workout Classes**

* **WorkoutClassService.java**  
  Trainers can add, update, delete, and view their own workout classes. Members can view all available classes. This service handles the logic to interact with the workout\_classes table in the database.

**Membership System**

* **MembershipService.java**  
  Users can buy gym memberships. Admins can see all memberships and the total revenue. Members can view how much they have spent. The membership data is stored in the memberships table.

**Data and Database**

* **Domain Model Classes**  
  The User, WorkoutClass, and Membership classes define the structure of the data.
* **Database**  
  All user, classes, and membership data is saved in a PostgreSQL database. A SQL script is included in the project to set everything up. It contains commands to create the tables and add sample data.

**Input Handling**

* **Validation and Error Handling**  
  If a user enters the wrong input (like a letter instead of a number), the system will not crash; it asks the user to try again instead.

A diagram of a service layer

AI-generated content may be incorrect.

Figure 1: UML Class Diagram showing the core data models (User, WorkoutClass, Membership), business logic services, and data access objects. Arrows represent dependencies and associations between layers.

**4. Using the Application**

To begin using the system, launch the application by running the GymApp.java file from an IDE or command line. You will be prompted with a numbered menu. Enter the number that corresponds to your desired action and follow the on screen instructions.

**Example Workflow**

1. Select "Register new user" to create an account.
2. Enter your details, including your role (Admin, Trainer, or Member).
3. Once registered, select "Login as user" and input your username and password.
4. After successful login, the system will detect your role and load the appropriate menu.

Each role menu includes clear prompts for each available action. For example, trainers will see an option to "Add Workout Class," while admins will see options like "View Total Revenue." All interactions are performed via keyboard input and confirmation messages are displayed after each action.

**5. System Purpose and Goals**

This system was developed to simulate a functional gym management experience with minimal interface complexity.