# Gym Management System NickPow

## 1. User Documentation

### Overview

The Gym Management System is a console-based application written in Java using Maven, connected to a PostgreSQL database. It simulates the management of a gym where users can register and interact with the system based on their roles: \*\*Admin\*\*, \*\*Trainer\*\*, or \*\*Member\*\*. Each user role has specific features based on access level.

### How It Works

- Users can register or log in.  
- Upon logging in, they see a role-specific menu with options.  
- Admins manage users and revenue, trainers manage classes, and members view classes and buy memberships.  
- All actions are performed using simple numbered menus through the terminal.

### Class Interactions

- `User` is the abstract superclass for Admin, Trainer, and Member.  
- `UserDAO` handles registration, login, and user-related queries.  
- `WorkoutClassDAO` and `MembershipDAO` handle database operations.  
- `MembershipService` contains business logic for memberships.  
- `Main.java` handles the user interface and routing.  
- `PasswordUtils` handles password hashing and verification using BCrypt.

### Step-by-Step Usage

1. Open the terminal in the project root folder.  
2. Run the following commands:  
 mvn clean compile  
 mvn exec:java -Dexec.mainClass="com.gym.Main"  
3. Choose to register or log in.  
4. After login, use the menu to perform actions based on your role.

## 2. Development Documentation**:**

### Project Structure**:**

JavaFinalSprintS3/

├── src/

│ └── main/

│ └── java/

│ └── com/

│ └── gym/

│ ├── dao/ - Handles database operations (UserDAO, MembershipDAO, WorkoutClassDAO)

│ ├── models/ - Java class models (User, Admin, Member, Trainer, WorkoutClass, Membership)

│ ├── services/ - Business logic (e.g., MembershipService)

│ ├── utils/ - Helper classes like PasswordUtils for hashing passwords

│ └── Main.java - Entry point that manages role-based menus

├── docs/ - Word documentation and testing checklist

│ ├── GymManagementSystem\_Upgraded\_Documentation.docx

│ ├── Gym\_Testing\_Checklist.docx

│ └── UML\_Class\_Diagram.png

├── schema.sql - PostgreSQL schema for creating tables

├── seed.sql - Seed data with valid BCrypt-hashed accounts

├── pom.xml - Maven build configuration file

├── README.md - Project summary and setup guide for GitHub

Build Process:

I began the project by setting up the Maven structure in Visual Studio Code, creating the base folders for models, dao, services, and utils. I used the pom.xml to manage dependencies for PostgreSQL and BCrypt. The first part I implemented was the User model and its subclasses. I followed with UserDAO and login/registration logic. I tested this manually using print statements and login prompts until I could reliably create new users and authenticate them securely using hashed passwords. Once user authentication was working, I built out the menu system in Main.java, using role-based logic to display different menu options for Admins, Trainers, and Members. I then created the Membership and WorkoutClass models, followed by their DAO classes. I used a scanner-based interface for input and manually tested adding, updating, and deleting classes and memberships from the console. During development, I frequently reset the database using the schema.sql and seed.sql scripts to keep test data consistent. As I finalized the project, I added improved input validation, and example prompts. Once everything worked as expected, I tested each role's features and commited to github.

### Build & Run Instructions for development**:**

- Clone the repository.   
- Set up the database in PostgreSQL:  
 CREATE USER gymuser WITH PASSWORD 'gympass';  
 CREATE DATABASE gymdb;  
 GRANT ALL PRIVILEGES ON DATABASE gymdb TO gymuser;  
- Run:  
 psql -U gymuser -d gymdb -f schema.sql  
 psql -U gymuser -d gymdb -f seed.sql  
- Run the app:  
 mvn clean compile  
 mvn exec:java -Dexec.mainClass="com.gym.Main"

## 3. Challenges**:**

- Time management issues.