Gym Management System

A Java-based application for managing gym operations including users, memberships, and workout classes.

Table of Contents

Introduction

The Gym Management System allows users to perform various tasks including adding new users, logging in, purchasing gym memberships, and managing workout classes based on user roles.

Requirements

1. Ensure you have an IDE (Integrated Development Environment) on your local computer device. Could be VS Code or IntelliJ for example.
2. Download all files from the repository or clone the repository in your IDE or clone the repository by running this command into your IDE's terminal: "git clone https://github.com/AlecBProxy/JavaFinalSprint".
3. Ensure you have the Java JDK installed to assist in compiling. If you do not, here is a link: https://www.oracle.com/ca-en/java/technologies/downloads/
4. Install Maven installed in your IDE. If you do not, here is a link: https://maven.apache.org/download.cgi and type "mvn install" in your IDE terminal to get the necessary dependencies.

5. Install Postgresql and run the scripts found in the file scripts.sql to establish and populate the tables.
6. Update the database connection URL, username, and password in your application configuration in the Database.java file to reflect your's.

Installation

1. Clone the repository:

```
git clone https://github.com/AlecBProxy/JavaFinalSprint
```

2. Install Maven dependencies:

```
mvn install
```

3. Run the PostgreSQL scripts found in `scripts.sql` to establish and populate the database tables
4. Update the database connection URL, username, and password in the `Database.java` file to match your configuration

Running the Application

Once you have downloaded the files or cloned the repository to your local computer and have met the necessary requirements above, to run the program type these commands into your IDE terminal:

```
mvn compile
mvn exec:java -Dexec.mainClass="org.keyin.GymApp"
```

Here the menu options include:

1. Add a new user
2. Login as a user
3. Exit

Features

- User management with different roles (Admin, Trainer, Member)

- Secure authentication with password hashing
- Membership management
- Workout class management
- Role-specific interfaces and privileges

Menu Options

Adding a New User

If you choose to add a new user you will be prompted to enter:

- Username
- Password
- Phone Number
- Address
- Role (Choose between admin, trainer or member)

Now upon completion you will have added your credentials to the database and you can login in the next choice.

Login as a User

When you login, you will be prompted to enter your username and password.

Based on your role, you will have a different menu; admin, member, and trainer

Admin Menu

Here you have the options:

1. View all users
2. Delete a user
3. View all memberships and total expenses
4. Exit

Trainer Menu

Here you have the options:

1. View assigned workout classes

2. Create a new workout class
3. Update an existing workout class
4. Delete a workout class
5. Purchase a gym membership
6. Exit

Member Menu

Here you have the options:

1. View workout classes
2. Purchase membership
3. View total membership expenses
4. Exit

Project Structure

```
JAVAFINALSPRINT/
├── .idea/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   └── org/
│   │   │       └── keyin/
│   │   │           ├── common/
│   │   │           │   └── CrudService.java
│   │   │           ├── database/
│   │   │           │   └── DatabaseConnection.java
│   │   │           ├── memberships/
│   │   │           │   ├── Membership.java
│   │   │           │   ├── MembershipDAO.java
│   │   │           │   └── MembershipService.java
│   │   │           ├── menus/
│   │   │           │   ├── AdminMenuHandler.java
│   │   │           │   ├── MemberMenuHandler.java
│   │   │           │   └── TrainerMenuHandler.java
│   │   │           ├── user/
│   │   │           │   ├── childclasses/
│   │   │           │   ├── User.java
│   │   │           │   ├── UserDAO.java
│   │   │           │   └── UserService.java
│   │   │           ├── workoutclasses/
│   │   │           │   ├── WorkoutClass.java
```

```
│   │   │           │   ├── WorkoutClassDAO.java
│   │   │           │   └── WorkoutClassService.java
│   │   │           └── GymApp.java
│   │   └── resources/
│   │       └── scripts.sql
│   └── test/
│       └── java/
├── target/
├── .gitignore
├── Documentation.txt
├── package-lock.json
└── pom.xml
```

Class Overview

- **User**: Base class containing general user information including username, hashed password, email, phone number, address, and role.
- **Admin**: Extends User, providing administrative privileges like managing user accounts and viewing revenue.
- **Trainer**: Extends User, allows trainers to manage workout classes and memberships.
- **Member**: Extends User, enabling members to interact with workout classes and manage their memberships.
- **UserDAO, MembershipDAO, WorkoutClassDAO**: Manage direct interactions with the database, handling create, read, update, and delete operations for users, memberships, and workout classes respectively.
- **UserService, MembershipService, WorkoutClassService**: Facilitate interactions between the user interface and DAO classes, encapsulating business logic.
- **DatabaseConnection**: Manages the PostgreSQL database connection.
- **Membership**: Represents membership details like membership ID, user ID, type, description, cost, and duration.
- **WorkoutClass**: Manages workout class details including class ID, type, description, and associated trainer ID.
- **AdminMenuHandler, TrainerMenuHandler, MemberMenuHandler**: Handle menu interactions and functionality specific to each user role.

Dependencies

Maven is used for dependency management.

Dependencies include PostgreSQL JDBC Driver, BCrypt, and JUnit for testing.

Class Diagram

A detailed class diagram has been created and can be located in the main project directory. It is titled "FinalSprintClassDiagram.png".

Documentation

To view generated Javadocs, run:

```
open target/reports/apidocs/index.html
```

Team Contributions

Alex Russell

For this project I setup the initial Trello board and team meetings. I also worked on setting up the database connection and testing that to ensure it's working. Next I worked on the CLI, creating the sub-menus and making sure they were styled consistently and ready to work with the other members classes. Finally I fixed a few errors and added some scripts.

Regarding challenges, the initial one was trying to learn the concepts needed for this project that weren't discussed during class. But I would say the main challenge in the project was trying to understand the requirements versus what a full gym application would likely have as options and functions.

Abdul Zahiri

For this project, I created the Workout Classes feature and built the Trainer Menu, which includes options like creating, updating, and deleting workout classes. I also worked closely with my teammate on the Membership Classes and the Member Menu, helping with both functionality and testing to make sure everything worked smoothly.

I was also responsible for setting up the Trainer Menu interface, making sure it was clear and easy to use. Throughout the project, we used Trello to stay organized, and I actively collaborated with the team to plan and divide tasks.

Alexander Barry

I was primarily responsible for designing and implementing the User superclass, along with its supporting classes: UserDAO and UserService. My contributions included developing the logic for user authentication, including password encryption and login functionality.

As the command-line interface (CLI) began to take shape, I led the refactoring of our menu system into dedicated MenuHandler classes, which are coordinated through a switch statement in the main GymApp class. To enhance the user experience, I also added dynamic tables that populate at various stages of CLI operation.

For testing and development purposes, I wrote the loadDefaultUsers function, which allowed the team to validate program features with preloaded data. Lastly, I contributed to the creation of our class diagram to support the final project documentation.

The main challenge experienced in the completion of this project was setting up Maven for the project. It was a new and unfamiliar workflow for me and took some time.

Noah Hickey

I was responsible for creating the Membership, Membership DAO, and Membership Services files. Tested the database functionality in pgAdmin and tested the menu functionality. Revised the membership associated files with a teammate. Ran through the menu systems for our video and generated Javadocs for our project. I completed the Java documentation.

The biggest challenge was understanding Maven and getting a handle on content not explicitly covered in class. Sometimes we would realize things needed to be simplified, so we would have to review project requirements and focus on the main project objectives and attempt to succeed in completing them.