

# 471 Database Management Systems

## My Gym Booking

### Final Report

<https://github.com/VictorTDong/471-MyGymBooking>

# Table of Contents

<b>Abstract</b>	<b>5</b>
<b>Introduction</b>	<b>5</b>
What problem does My Gym Booking solve?	5
How does My Gym Booking address the problem?	5
<b>Project Design</b>	<b>6</b>
<b>EER Diagram</b>	<b>11</b>
<b>Implementation</b>	<b>13</b>
<b>Relational Model</b>	<b>13</b>
SQL Queries	14
Queries for Class	14
Create a new class	14
Get classes with facility information	14
Updating a class	14
Deleting a class	14
Updating class participants	15
Queries for Shifts	15
Get shifts with information from facility	15
Updating shift information	15
Creating a shift	15
Deleting a shift	15
Get the trainer' shift	15
Queries for Facility	15
Deleting a class	15
Queries for Schedule	16
Get schedule of a client	16
Deleting a schedule	16
Queries for Account	16
Creating an account	16
Get account info from a member	16
Updating account password	16
Queries for Membership	16
Creating a membership	16
<b>API</b>	<b>16</b>
GYM API	17
Introduction	17
Overview	17

Error Codes	17
API Endpoints	17
Account	17
GET Get all accounts	17
Sample request:	17
Sample response:	18
GET Get an account	20
Sample request:	20
Sample response:	20
POST Update account	20
Sample request:	21
Sample response:	21
POST Create account	21
Sample request:	22
Sample response:	22
DEL Delete account	22
Sample request:	22
Sample response:	22
Class	22
GET Get all classes	22
Sample request:	22
Sample response:	22
POST Create class	23
Sample request:	24
Sample response:	24
DEL Delete a class	24
Sample request:	24
Sample response:	24
GET Get single class	24
Sample request:	25
Sample response:	25
POST Update class	25
Sample request:	25
Sample response:	25
Facility	25
GET Get all facilities	25
Sample request:	25
Sample response:	25
GET Get a facility	26
Sample request:	26
Sample response:	26

<b>Membership</b>	<b>27</b>
GET Get all memberships	27
Sample request:	27
Sample response:	27
GET Get a membership	27
Sample request:	27
Sample response:	27
POST Create membership	28
Sample request:	28
Sample response:	28
<b>Schedule</b>	<b>28</b>
GET Get all bookings	28
Sample request:	28
Sample response:	28
GET Get a booking	29
Sample request:	29
Sample response:	29
DEL Delete booking	29
Sample request:	30
Sample response:	30
POST Create booking	30
Sample request:	30
Sample response:	30
<b>Shift</b>	<b>30</b>
GET Get all shifts	30
Sample request:	31
Sample response:	31
GET Get a shift	33
Sample request:	33
Sample response:	33
DEL Delete shift	33
Sample request:	34
Sample response:	34
POST Update shift	34
Sample request:	34
Sample response:	34
POST Create shift	34
Sample request:	35
Sample response:	35
<b>User guide</b>	<b>35</b>

Setup	35
System requirements:	35
Pre-login	37
Client	40
Manager	45
Janitor	61
<b>References:</b>	<b>67</b>
<b>Appendix</b>	<b>69</b>

# Abstract

My Gym booking is an interactive website that seeks to solve a problem apparent in most gyms and fitness centers, overcrowding. Our website easily allows gym owners to implement a flexible client reservation system. By using our website the gym owners can easily display class times and member participation. This information allows users to choose a time to workout when they feel safe or reserve space during a more popular time. The current system used is outdated; compared to current systems implemented like the University of Calgary's booking system, ours offer an easier interface. We use an SQL based database thus allowing us to easily create, read, update and delete data as need be. We are able to provide an interface that allows users to be more efficient in the booking process.

## Introduction

### What problem does My Gym Booking solve?

The COVID-19 pandemic has made apparent that many of the world's systems aren't up to standard. My Gym Booking attempts to target one of these systems. Reserving time in a gym became the best way to guarantee space in a business with limited capacity. As well having a database for occupancy allows the staff of the facility to save time by not having to manually track each person entering and leaving the facility.

### How does My Gym Booking address the problem?

My Gym Booking allows paying members to pre-book time in the gym, ensuring that they will not be turned around at the door. They are also allowed to view the number of people registered for a time slot. With this information they are better allowed to make an informed decision based on their personal risk tolerance. After booking a time slot, the user is allowed to modify their plans based on their needs at the moment.

Employees of the gym are also offered a variety of services based on their role in the company. Trainers (or other class leaders) are able to view their upcoming class schedule to better prepare for their lessons and gauge upcoming participation. Janitors are able to view an automatically generated cleaning schedule for the different facilities of the gym. Management is able to modify account information for users and other employees. They are also able to input the upcoming classes for the day.

# Project Design

Members are the clients of the gym. Before logging in they are able to view the schedules for both classes and facilities (fig 1.1 and fig 1.2). They are also able to log in/out of their account (fig 2). When logged into their account, the user will then be able to register for a class (fig 3), and be able to view their schedule (fig 4). While logged in the user's password can also be changed (fig 5).

## View Available Classes

Hot Yoga Time: 09:00:00 Room: Gym 2 Capacity 25 / 50	Hot Yoga Time: 09:00:00 Room: Gym 1 Capacity 24 / 50	Power Lifting Time: 09:00:00 Room: Weight Room Capacity 7 / 50	Cycling Time: 09:00:00 Room: Gym 2 Capacity 3 / 50	Zumba Time: 15:00:00 Room: Gym 1 Capacity 0 / 50
---------------------------------------------------------------	---------------------------------------------------------------	-------------------------------------------------------------------------	-------------------------------------------------------------	-----------------------------------------------------------

fig 1.1

## View Facilities

			
Facility Name: Gym 1 Room Number: 1 Max Capacity: 50	Facility Name: Gym 2 Room Number: 2 Max Capacity: 50	Facility Name: Gym 3 Room Number: 3 Max Capacity: 50	Facility Name: Weight Room Room Number: 4 Max Capacity: 50

fig 1.2

## Log in

Please enter your login information.

Username

Password

fig 2

## View Available Classes

Hot Yoga Time: 09:00:00 Room: Gym 2 Capacity 25 / 50	Hot Yoga Time: 09:00:00 Room: Gym 1 Capacity 24 / 50	Power Lifting Time: 09:00:00 Room: Weight Room Capacity 7 / 50	Cycling Time: 09:00:00 Room: Gym 2 Capacity 3 / 50	Zumba Time: 15:00:00 Room: Gym 1 Capacity 0 / 50
<input type="button" value="Join Class ID: 8"/>	<input type="button" value="Join Class ID: 9"/>	<input type="button" value="Join Class ID: 10"/>	<input type="button" value="Join Class ID: 13"/>	<input type="button" value="Join Class ID: 15"/>

fig 3

## Your Bookings

Class ID	Booking ID	Class Name	Time	Room Number	Actions
9	126	Hot Yoga	09:00:00	1	

fig 4

## Account information

First Name	Last Name	Birthdate	Reoccuring payment date	Membership ID
Tony	Kers	2022-04-12	2022-04-13	123

Change your password

fig 5

Trainers are employees of the gym that teach a class. They are able to view the class and facility schedule before they log in (fig 1.1 and fig 1.2). When they log in they are able to view their work schedule (fig 6).

## My work Schedule

Facility Name	Date	Start Time	End Time
Gym 2	2022-04-15	09:00:00	17:00:00
Gym 2	2022-04-16	09:00:00	17:00:00
Gym 2	2022-04-17	09:00:00	17:00:00
Gym 3	2022-04-18	09:00:00	13:30:00

fig 6

Janitors are employees of the gym that clean after a class ends. They are able to view the class and facility schedule before they log in (fig 1.1 and fig 1.2). When they log in they are able to view their cleaning schedule (fig 7).

## Today's Cleaning Schedule

Facility Name	Room Number	Time	Complete
---------------	-------------	------	----------

## Room's already cleaned

Facility Name	Room Number
Gym 1	1
Gym 2	2

fig 7

Managers are employees who are able to control the more administrative duties for the gym. They are able to view the class and facility schedule before they log in (fig 1.1 and fig 1.2).

While logged in they are able to manage client information(fig 8), register members(fig 9), manage classes(fig 10), and manage shifts (fig 11).

## Manage Clients

Username	First Name	Last Name	Password	Vaccination Status	
client2	Kevin	Tree	password	0	
client4	Carmen	Cong	password	0	
client1	Tony	Kers	password	0	
client3	Deane	Gale	password	0	

fig 8

## Register member

SSN	First Name	Last Name	Birthdate
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> mm/dd/yyyy
Username	Password	Vaccination status	
<input type="text"/>	<input type="text"/>	<input type="text"/>	
<input type="button" value="Add Member"/>			

fig 9

## Manage Classes

Class ID	Name	Time	Number of Participants	Room Number	Facility Name	
8	Hot Yoga	09:00:00	25	2	Gym 2	 
9	Hot Yoga	09:00:00	24	1	Gym 1	 
10	Power Lifting	09:00:00	7	4	Weight Room	 
13	Cycling	09:00:00	3	2	Gym 2	 
15	Zumba	15:00:00	0	1	Gym 1	 

Class ID	Name	Time	Room Number
<input type="text"/>	<input type="text"/>	<input type="text"/> --:-- --	<input type="text"/>
<input type="button" value="Add Class"/>			

fig 10

# Manage Shifts

SSN	Employee First Name	Employee Last Name	Date	Time	Facility	Room Number		
321	Brendan	Bob	2022-04-14	12:00:00-17:00:00	Gym 2	2		
321	Brendan	Bob	2022-04-14	09:00:00-12:00:00	Gym 3	3		

SSN	Date	Start Time	End Time	Room Number
<input type="text"/>	<input type="text"/> mm/dd/yyyy	<input type="text"/> --:-- --	<input type="text"/> --:-- --	<input type="text"/>

fig 11

# EER Diagram

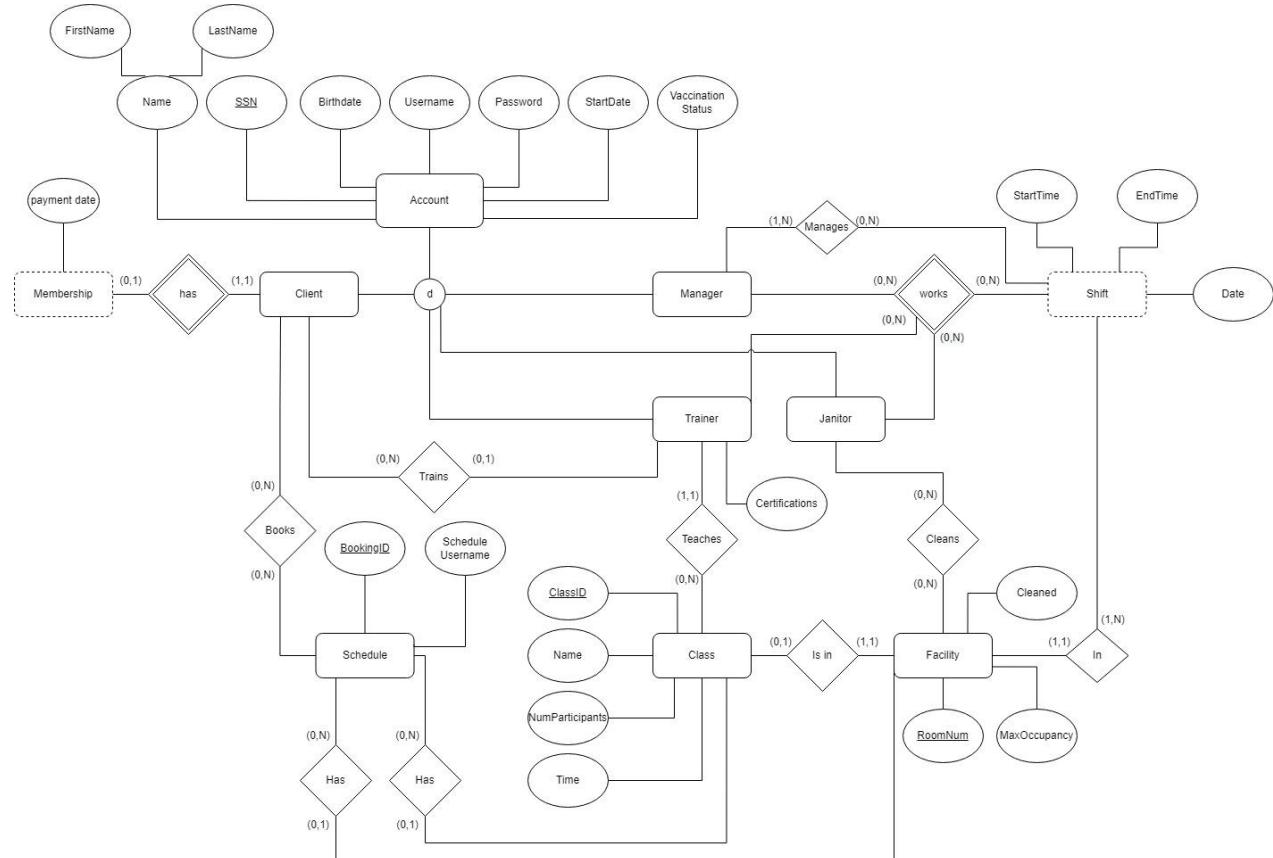
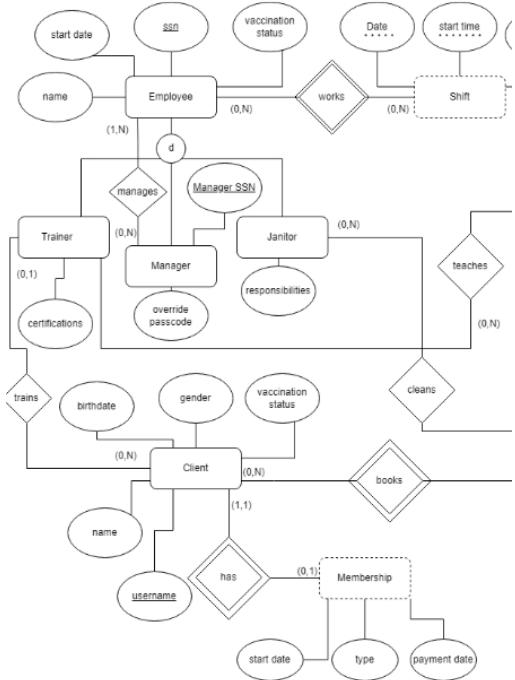


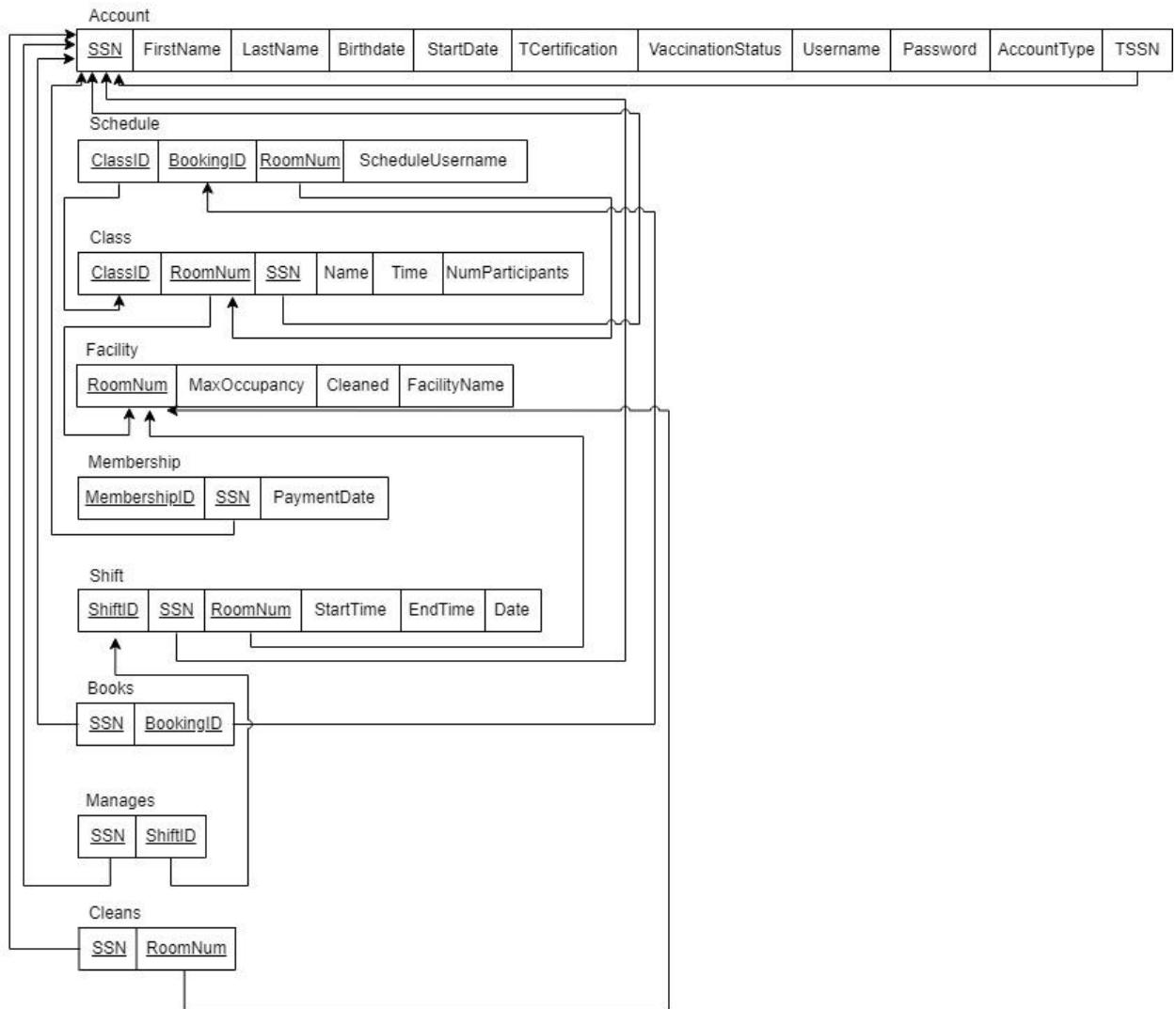
fig 12

For the final results of our project, we made some changes including adding another entity called Account because based on the implementation of our project it made sense to split the super class Employee into 3 separate accounts since the entity Client shares the same attributes and will require the same log in process. The difference is shown between fig 12 and fig 13. Further detail of the old EER is provided in the appendix.



# Implementation

## Relational Model



We had to make some changes so it aligns with our new EER diagram. The previous versions of the EER and RM will be included in the appendix.

The program has changed since the presentation. The website is now fully functional. During the presentation, there were some issues with PHPMYADMIN which didn't let us export our database and the problem has since been fixed by reinstalling a previous version of XAMPP with an older version of php. The API's are fully functional and during the presentation we had issues with them but that is due to our website storing session cookies. In the API section we have Postman results from our website as referenced in the link provided. These results were all generated with the correct session cookies for the specified user and as well as API's for our project with full documentation and examples (More details provided in the appendix). Please refer to the Postman link for requests from our website.

For our project, we have used PHPMYADMIN for our database management system. This system provided an easy to use interface that allowed us to implement our database and run the SQL statements. In each of our functionalities of the system, we have used various SQL statements including the SELECT, DELETE, UPDATE and INSERT to help with the collection and manipulating of data in our system.

## SQL Queries

### Queries for Class

Create a new class

```
INSERT INTO `class` (`ClassID`, `SSN`, `Name`, `Time`, `NumParticipants`, `RoomNum`)
VALUES ('$ClassID', '$SSN', '$ClassName', '$Time', '0', '$RoomNum');
```

Get classes with facility information

```
SELECT *
FROM `class` as c, `facility` as f
WHERE f.RoomNum = c.RoomNum ORDER BY c.ClassID ASC;
```

```
SELECT ClassID, Name, Time, RoomNum, NumParticipants, MaxOccupancy,
FacilityName
FROM class NATURAL JOIN facility
ORDER BY NumParticipants DESC;
```

Updating a class

```
UPDATE `class`
SET SSN = '$SSN', Name = '$ClassName', Time = '$Time', RoomNum = '$RoomNum'
WHERE ClassID = '$updateClassID';
```

Deleting a class

```
DELETE FROM `class`
WHERE ClassID = '$ClassID';
```

## Updating class participants

```
UPDATE `class` Set `NumParticipants` = `NumParticipants` - 1  
WHERE classID = $deleteClassID;
```

## Queries for Shifts

### Get shifts with information from facility

```
SELECT a.FirstName, a.LastName, a.SSN, `Date`, `StartTime`, `EndTime`,  
f.FacilityName, f.RoomNum  
FROM `shift` as s, `facility` as f, `account` as a  
WHERE s.SSN = a.SSN AND s.RoomNum = f.RoomNum ORDER BY Date ASC;
```

### Updating shift information

```
UPDATE `shift`  
SET Date = '$insertDate', StartTime = '$insertStart_time', EndTime = '$insertEnd_time',  
RoomNum = '$insertFacility'  
WHERE SSN = '$updateSSN' AND Date = '$updateDate' AND StartTime =  
'$updateStart_time' AND EndTime = '$updateEnd_time';
```

### Creating a shift

```
INSERT INTO `shift` (`SSN`, `Date`, `StartTime`, `EndTime` , `RoomNum`)  
VALUES ('$insertSSN','$insertDate','$insertStart_time','$insertEnd_time','$insertFacility');
```

### Deleting a shift

```
DELETE FROM shift  
WHERE SSN = '$delete_SSN' AND Date = '$date' AND StartTime = '$start_time' AND  
EndTime = '$end_time';
```

### Get the trainer' shift

```
SELECT s.SSN, s.Date, s.StartTime, s.EndTime, f.RoomNum, f.FacilityName  
FROM `shift` as s, `facility` as f  
WHERE s.RoomNum = f.RoomNum AND SSN = '$trainerSSN'  
ORDER BY Date ASC;
```

## Queries for Facility

### Deleting a class

```
SELECT RoomNum, MaxOccupancy, FacilityName  
FROM facility  
ORDER BY RoomNum ASC;
```

## Queries for Schedule

Get schedule of a client

```
SELECT c.Name, `BookingID`, s.ClassID, c.Time, c.RoomNum  
FROM `schedule` as s, `Class` as c  
WHERE c.ClassID = s.ClassID AND ScheduleUsername = '$memberUsername';
```

Deleting a schedule

```
DELETE FROM schedule  
WHERE BookingID = '$deleteBookingID';
```

## Queries for Account

Creating an account

```
INSERT INTO `account` (`Username`, `FirstName`, `LastName`, `Startdate`,  
`AccountType`, `VaccinationStatus`, `BirthDate`, `Certification`, `SSN`, `Responsibilities`,  
`Password`)  
VALUES ('$insertUsername','$insertFirstName','$insertLastName', '$date',  
'0','$insertVaccination','$insertBirthdate',NULL,'$insertSSN',NULL,'$insertPassword');
```

Get account info from a member

```
SELECT * FROM `account` as s, `membership` as m  
WHERE m.SSN = s.SSN AND s.Username = '$memberUsername';
```

Updating account password

```
UPDATE account  
SET Password = '$updatePassword'  
WHERE UserName = '$username';
```

## Queries for Membership

Creating a membership

```
INSERT INTO `membership` (`MembershipID`, `SSN`, `PaymentDate`)  
VALUES (DEFAULT,'$insertSSN', '$date');
```

# API

## GYM API

### Introduction

This API collection holds all the back end API methods for interacting with accounts, classes, facilities, memberships, schedules, and shifts

### Overview

To run these API's you will need to open the XAMPP apache and my sql server with the gym database

### Error Codes

Users can expect HTTP response error codes 404 indicating that the request was unsuccessful and HTTP response code 200 indicating that the request was successful

Note full documentation for the GYM API can be acquired from  
<https://documenter.getpostman.com/view/20196276/Uyr5nyTD>

Note Postman results for the request from the website can be acquired from  
<https://documenter.getpostman.com/view/20196276/Uyr5nyXV#intro>

Note all sample request will be requested with Curl based on Postman documentation

### API Endpoints

#### Account

**GET** Get all accounts

<http://localhost/471Project/api/account/account.php>

Get all accounts in the database

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/account/account.php'
```

Sample response:

```
{  
  "body": [  
    {  
      "username": "trainer1",  
      "firstname": "Victor",  
      "lastname": "Bill",  
      "startdate": "2022-04-14",  
      "accounttype": "1",  
      "vaccination": "1",  
      "birthdate": "1990-04-12",  
      "certification": "CPR",  
      "ssn": "123",  
      "password": "password"  
    },  
    {  
      "username": "janitor1",  
      "firstname": "Brendan",  
      "lastname": "Bob",  
      "startdate": "2022-04-14",  
      "accounttype": "2",  
      "vaccination": "1",  
      "birthdate": "2005-04-12",  
      "certification": "CPR",  
      "ssn": "321",  
      "password": "password"  
    },  
    {  
      "username": "client2",  
      "firstname": "Kevin",  
      "lastname": "Cool",  
      "startdate": "2022-04-14",  
      "accounttype": "0",  
      "vaccination": "1",  
      "birthdate": "2022-04-12",  
      "certification": "",  
      "ssn": "654",  
      "password": "password"  
    },  
    {  
      "username": "manager1",  
      "firstname": "Samantha",  
      "lastname": "House",  
      "startdate": "2022-04-14",  
    }  
  ]  
}
```

```
        "accounttype": "3",
        "vaccination": "1",
        "birthdate": "2001-04-12",
        "certification": "CPR",
        "ssn": "741",
        "password": "password"
    },
{
    "username": "client1",
    "firstname": "Anthony",
    "lastname": "Pit",
    "startdate": "2022-04-14",
    "accounttype": "0",
    "vaccination": "0",
    "birthdate": "0200-04-12",
    "certification": "",
    "ssn": "789",
    "password": "password"
},
{
    "username": "client3",
    "firstname": "Deane",
    "lastname": "Gale",
    "startdate": "2022-04-14",
    "accounttype": "0",
    "vaccination": "0",
    "birthdate": "1999-12-25",
    "certification": "",
    "ssn": "856",
    "password": "password"
},
{
    "username": "client4",
    "firstname": "Kim",
    "lastname": "Possible",
    "startdate": "2022-04-16",
    "accounttype": "0",
    "vaccination": "1",
    "birthdate": "2005-05-29",
    "certification": null,
    "ssn": "4567",
    "password": "password"
}
],

```

```
        "ItemCount": 7  
    }
```

## GET Get an account

<http://localhost/471Project/api/account/account-single.php?SSN=>

Get a specified account from the database

Sample request:

```
curl --location --request GET  
'http://localhost/471Project/api/account/account-single.php?SSN=123'
```

Sample response:

```
{  
    "body": [  
        {  
            "username": "trainer1",  
            "firstname": "Billy",  
            "lastname": "Smith",  
            "startdate": "2022-04-14",  
            "accounttype": "1",  
            "vaccination": "1",  
            "birthdate": "2022-04-12",  
            "certification": "CPR",  
            "ssn": "123",  
            "password": "password"  
        }  
    ],  
    "ItemCount": 1  
}
```

## POST Update account

<http://localhost/471Project/api/account/account-update.php?SSN=&Firstname=&Lastname=&VaccinationStatus=&Password=>

Updates an account

PARAMS

SSN: The account SSN (INT)

Firstname: First name of account holder (STRING)

Lastname: Last name of account holder (STRING)

VaccinationStatus: Vaccination status of account holder (0 for false, 1 for true)

Password: Password of account holder (STRING)

Sample request:

```
curl --location --request POST  
http://localhost/471Project/api/account/account-update.php?SSN=123&Firstname=Victor&Lastname=Bill&VaccinationStatus=1&Password=password'
```

Sample response:

```
[  
 {  
   "username": "trainer1",  
   "firstname": "Victor",  
   "lastname": "Bill",  
   "startdate": "2022-04-14",  
   "accounttype": "1",  
   "vaccination": "1",  
   "birthdate": "2022-04-12",  
   "certification": "CPR",  
   "ssn": "123",  
   "password": "password"  
 }  
]
```

**POST** Create account

<http://localhost/471Project/api/account/account-create.php/>

Creates an account

BODY Raw

```
{  
   "Username": "",  
   "FirstName": "",  
   "LastName": "",  
   "AccountType": "",  
   "VaccinationStatus": "",  
   "BirthDate": "",  
   "Certification": "",  
   "SSN": ""  
}
```

```
    "Responsibilities": "",  
    "password": ""  
}
```

Sample request:

```
curl --location --request POST  
http://localhost/471Project/api/account/account-update.php?SSN=123&Firstname=Victor&Lastname=Bill&VaccinationStatus=1&Password=password'
```

Sample response:

```
{  
  "msg": "Account created",  
  "status": true  
}
```

## DEL Delete account

<http://localhost/471Project/api/account/account-delete.php?SSN=625>  
Deletes an account

Sample request:

```
curl --location --request DELETE  
http://localhost/471Project/api/account/account-delete.php?SSN=625'
```

Sample response:

```
{  
  "message": "Account with SSN: 625 deleted",  
  "status": true  
}
```

## Class

### GET Get all classes

<http://localhost/471Project/api/class/class.php>

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/class/class.php'
```

Sample response:

```
{
```

```

"body": [
  {
    "ClassID": "8",
    "Name": "Hot Yoga",
    "Time": "09:00:00",
    "NumParticipants": "25",
    "RoomNum ": "2"
  },
  {
    "ClassID": "9",
    "Name": "Hot Yoga",
    "Time": "09:00:00",
    "NumParticipants": "23",
    "RoomNum ": "1"
  },
  {
    "ClassID": "10",
    "Name": "Power Lifting",
    "Time": "09:00:00",
    "NumParticipants": "7",
    "RoomNum ": "4"
  },
  {
    "ClassID": "13",
    "Name": "Cycling",
    "Time": "09:00:00",
    "NumParticipants": "3",
    "RoomNum ": "2"
  },
  {
    "ClassID": "15",
    "Name": "Zumba",
    "Time": "15:00:00",
    "NumParticipants": "0",
    "RoomNum ": "1"
  }
],
"itemCount": 5
}

```

**POST** Create class

<http://localhost/471Project/api/class/class-create.php>

BODY raw

```
{  
    "Name": "",  
    "Time": "",  
    "NumParticipants": "",  
    "RoomNum": ""  
}
```

Sample request:

```
curl --location --request POST 'http://localhost/471Project/api/class/class-create.php' \  
--data-raw '{  
    "Name": "Rowing",  
    "Time": "12:00:00",  
    "NumParticipants": "0",  
    "RoomNum": "2"  
}'
```

Sample response:

```
{  
    "msg": "Class created",  
    "status": true  
}
```

## DEL Delete a class

<http://localhost/471Project/api/class/class-delete.php?ClassID=>

PARAMS

ClassID : Class ID of class to be deleted (INT)

Sample request:

```
curl --location --request DELETE  
http://localhost/471Project/api/class/class-delete.php?ClassID='
```

Sample response:

```
{  
    "message": "Class with ClassID : 16 deleted",  
    "status": true  
}
```

## GET Get single class

<http://localhost/471Project/api/class/class-single.php?ClassID=>

PARAMS

ClassID : Class ID of class to acquire (INT)

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/account/account.php'
```

Sample response:

### POST Update class

<http://localhost/471Project/api/class/class-update.php?ClassID=&Name=&Time=&RoomNum=>

PARAMS

ClassID: Class ID of class to update (INT)

Name: Name of class (STRING)

Time: Time of class (TIME)

RoomNum: Room number for the class (INT)

Sample request:

```
curl --location --request POST  
'http://localhost/471Project/api/class/class-update.php?ClassID=8&Name=HotYoga&Time=10:00:00&RoomNum=3'
```

Sample response:

```
{  
  "message": "Class updated",  
  "status": true  
}
```

## Facility

### GET Get all facilities

<http://localhost/471Project/api/facility/facility.php>

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/facility/facility.php'
```

Sample response:

```
{  
  "body": [  
    {  
      "RoomNum": "1",  
      "MaxOccupancy": "50",  
      "Cleaned": "1",  
      "FacilityName": "Gym 1"  
    }  
  ]  
}
```

```

},
{
  "RoomNum": "2",
  "MaxOccupancy": "50",
  "Cleaned": "0",
  "FacilityName": "Gym 2"
},
{
  "RoomNum": "3",
  "MaxOccupancy": "50",
  "Cleaned": "0",
  "FacilityName": "Gym 3"
},
{
  "RoomNum": "4",
  "MaxOccupancy": "50",
  "Cleaned": "0",
  "FacilityName": "Weight Room"
}
],
"itemCount": 4
}

```

**GET** Get a facility

<http://localhost/471Project/api/facility/facility-single.php?RoomNum=>

Sample request:

```
curl --location --request GET
http://localhost/471Project/api/facility/facility-single.php?RoomNum=3
```

Sample response:

```
{
  "body": [
    {
      "RoomNum": "3",
      "MaxOccupancy": "50",
      "Cleaned": "0",
      "FacilityName": "Gym 3"
    }
  ],
  "itemCount": 1
}
```

## Membership

**GET** Get all memberships

<http://localhost/471Project/api/membership/membership.php>

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/membership/membership.php'
```

Sample response:

```
{
  "body": [
    {
      "MembershipID": "123",
      "SSN": "789",
      "PaymentDate": "2022-04-13"
    },
    {
      "MembershipID": "365",
      "SSN": "654",
      "PaymentDate": "2022-04-15"
    },
    {
      "MembershipID": "2357",
      "SSN": "678",
      "PaymentDate": "2022-04-15"
    }
  ],
  "itemCount": 3
}
```

**GET** Get a membership

<http://localhost/471Project/api/membership/membership-single.php?MembershipID=123>

Sample request:

```
curl --location --request GET
http://localhost/471Project/api/membership/membership-single.php?MembershipID=123'
```

Sample response:

```
{
  "body": [
    {
      "MembershipID": "123",
    }
  ]
}
```

```
        "SSN": "789",
        "PaymentDate": "2022-04-13"
    },
],
"itemCount": 1
}
```

**POST** Create membership

<http://localhost/471Project/api/membership/membership-create.php>

BODY Raw

```
{
    "SSN": ""
}
```

Sample request:

```
curl --location --request POST
'http://localhost/471Project/api/membership/membership-create.php' \--data-raw '{
    "SSN": "856"}'
```

Sample response:

```
{
    "msg": "Membership created",
    "status": true
}
```

## Schedule

**GET** Get all bookings

<http://localhost/471Project/api/schedule/schedule.php>

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/schedule/schedule.php'
```

Sample response:

```
{
    "body": [
        {
            "BookingID": "48",
            "ClassID": "10",
            "RoomNum": "1",
            "ScheduleUsername": "client3"
        },
    ]
```

```
{
  "BookingID": "49",
  "ClassID": "10",
  "RoomNum": "1",
  "ScheduleUsername": "client2"
},
{
  "BookingID": "110",
  "ClassID": "15",
  "RoomNum": "1",
  "ScheduleUsername": "client2"
}
],
"itemCount": 3
}
```

## GET Get a booking

<http://localhost/471Project/api/schedule/schedule-single.php?BookingID=>

PARAMS

BookingID: Booking ID of booking to get (INT)

Sample request:

curl --location --request GET

['http://localhost/471Project/api/schedule/schedule-single.php?BookingID=48'](http://localhost/471Project/api/schedule/schedule-single.php?BookingID=48)

Sample response:

```
{
  "body": [
    {
      "BookingID": "48",
      "ClassID": "10",
      "RoomNum": "1",
      "ScheduleUsername": "client3"
    }
  ],
  "itemCount": 1
}
```

## DEL Delete booking

<http://localhost/471Project/api/schedule/schedule-delete.php?BookingID=>

PARAMS

BookingID: Booking ID of booking to get (INT)

Sample request:

```
curl --location --request DELETE  
http://localhost/471Project/api/schedule/schedule-delete.php?BookingID=111'
```

Sample response:

```
{  
  "message": "Booking with Booking ID: 111 deleted",  
  "status": true  
}
```

**POST** Create booking

<http://localhost/471Project/api/schedule/schedule-create.php>

BODY Raw

```
{  
  "ClassID": "",  
  "RoomNum": "",  
  "ScheduleUsername": ""  
}
```

Sample request:

```
curl --location --request POST  
http://localhost/471Project/api/schedule/schedule-create.php' \  
--data-raw '{  
  "ClassID": "15",  
  "RoomNum": "1",  
  "ScheduleUsername": "client2"  
}'
```

Sample response:

```
{  
  "msg": "Booking created",  
  "status": true  
}
```

Shift

**GET** Get all shifts

<http://localhost/471Project/api/shift/shift.php>

Sample request:

```
curl --location --request GET 'http://localhost/471Project/api/shift/shift.php'
```

Sample response:

```
{
  "body": [
    {
      "ShiftID": "1",
      "SSN": "123",
      "Date": "2022-04-15",
      "StartTime": "09:00:00",
      "EndTime": "17:00:00",
      "RoomNum": "2"
    },
    {
      "ShiftID": "5",
      "SSN": "123",
      "Date": "2022-04-16",
      "StartTime": "09:00:00",
      "EndTime": "17:00:00",
      "RoomNum": "2"
    },
    {
      "ShiftID": "6",
      "SSN": "123",
      "Date": "2022-04-17",
      "StartTime": "09:00:00",
      "EndTime": "17:00:00",
      "RoomNum": "2"
    },
    {
      "ShiftID": "7",
      "SSN": "456",
      "Date": "2022-04-17",
      "StartTime": "09:00:00",
      "EndTime": "17:00:00",
      "RoomNum": "2"
    },
    {
      "ShiftID": "10",
      "SSN": "321",
      "Date": "2022-04-18",
      "StartTime": "09:00:00",
      "EndTime": "16:30:00",
      "RoomNum": "2"
    }
  ]
}
```

```
"RoomNum ": "3"
},
{
  "ShiftID ": "11",
  "SSN": "321",
  "Date": "2022-04-14",
  "StartTime": "09:00:00",
  "EndTime": "12:00:00",
  "RoomNum ": "3"
},
{
  "ShiftID ": "12",
  "SSN": "321",
  "Date": "2022-04-15",
  "StartTime": "09:00:00",
  "EndTime": "12:00:00",
  "RoomNum ": "1"
},
{
  "ShiftID ": "13",
  "SSN": "321",
  "Date": "2022-04-16",
  "StartTime": "09:00:00",
  "EndTime": "12:00:00",
  "RoomNum ": "1"
},
{
  "ShiftID ": "14",
  "SSN": "321",
  "Date": "2022-04-15",
  "StartTime": "12:00:00",
  "EndTime": "17:00:00",
  "RoomNum ": "2"
},
{
  "ShiftID ": "15",
  "SSN": "321",
  "Date": "2022-04-16",
  "StartTime": "12:00:00",
  "EndTime": "17:00:00",
  "RoomNum ": "2"
},
{
  "ShiftID ": "18",
```

```

        "SSN": "123",
        "Date": "2022-04-18",
        "StartTime": "09:00:00",
        "EndTime": "13:30:00",
        "RoomNum ": "3"
    }
],
"itemCount": 11
}

```

**GET** Get a shift

<http://localhost/471Project/api/shift/shift-single.php?ShiftID=>

PARAMS

ShiftID: Shift ID of shift to get (INT)

Sample request:

```
curl --location --request GET
'http://localhost/471Project/api/schedule/schedule-single.php?BookingID=48'
```

Sample response:

```
{
  "body": [
    {
      "ShiftID ": "5",
      "SSN": "123",
      "Date": "2022-04-16",
      "StartTime": "09:00:00",
      "EndTime": "17:00:00",
      "RoomNum ": "2"
    }
  ],
  "itemCount": 1
}
```

**DEL** Delete shift

<http://localhost/471Project/api/shift/shift-delete.php?ShiftID=>

PARAMS

ShiftID: Shift ID of shift to get (INT)

Sample request:

```
curl --location --request DELETE  
http://localhost/471Project/api/shift/shift-delete.php?ShiftID=20'
```

Sample response:

```
{  
  "message": "Booking with Booking ID: 111 deleted",  
  "status": true  
}
```

### POST Update shift

<http://localhost/471Project/api/shift/shift-update.php?ShiftID=&Date=&StartTime=&EndTime=&RoomNum=>

PARAMS

ShiftID: Shift ID of shift to update (INT)

Date : Date of shift (DATE)

StartTime: Start time of shift (TIME)

EndTime: End time of Shift (TIME)

RoomNum: Room number of the shift which the employee works at (INT)

Sample request:

```
curl --location --request POST  
http://localhost/471Project/api/shift/shift-update.php?ShiftID=20&Date=2022-04-18&StartTime=09:00:00&EndTime=16:30:00&RoomNum=3'
```

Sample response:

```
{  
  "message": "Shift updated",  
  "status": true  
}
```

### POST Create shift

<http://localhost/471Project/api/shift/shift-create.php>

BODY Raw

```
{  
  "SSN":"123",  
  "Date":"2022-04-30",  
  "StartTime":"09:00:00",  
  "EndTime":"17:00:00",  
  "RoomNum":"4"  
}
```

Sample request:

```
curl --location --request POST 'http://localhost/471Project/api/shift/shift-create.php' \
--data-raw '{
    "SSN": "123",
    "Date": "2022-09-05",
    "StartTime": "09:00:00",
    "EndTime": "17:00:00",
    "RoomNum": "3"
}'
```

Sample response:

```
{
    "msg": "Shift created",
    "status": true
}
```

## User guide

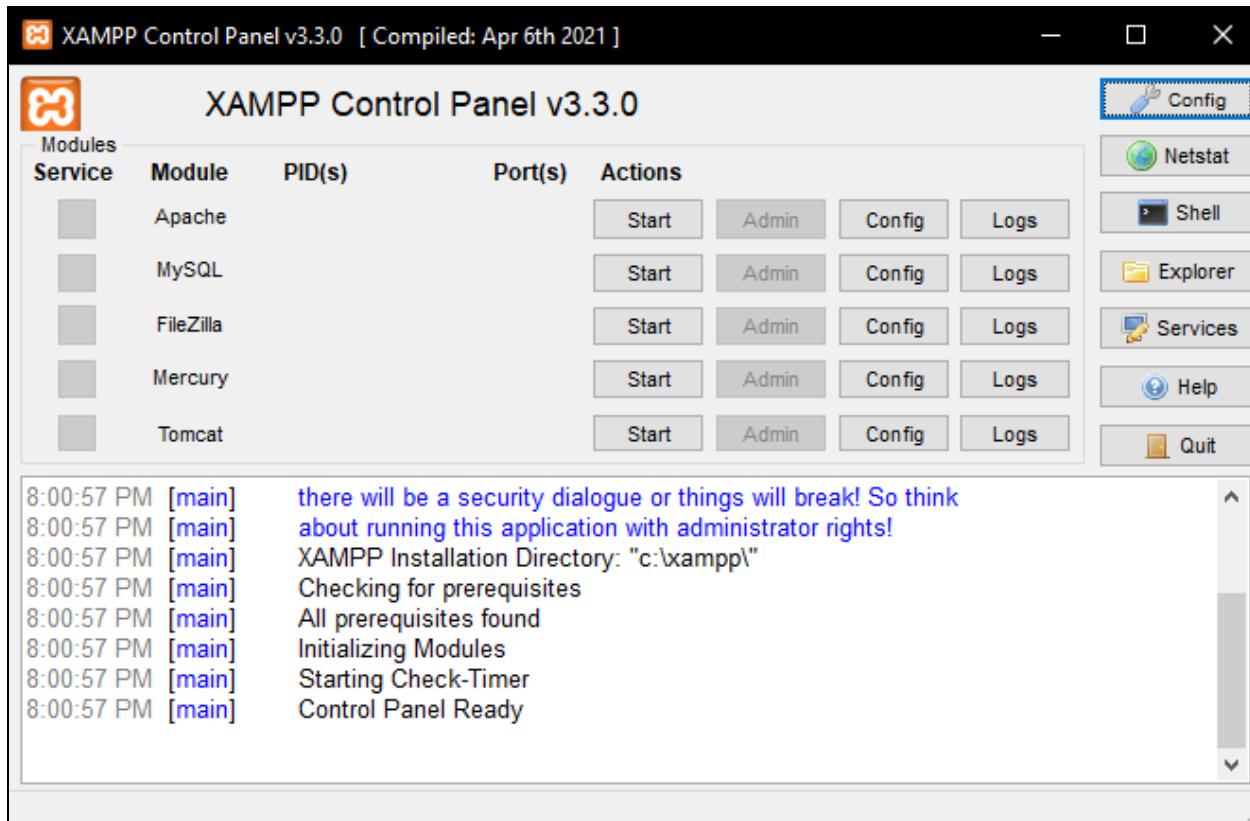
### Setup

#### System requirements:

Users will need to have XAMPP installed on their computer. There has been issues with the newest version of XAMPP so version 7.4.28 is preferred and can be found through the link XAMPP'S download page below.

<https://www.apachefriends.org/download.html>

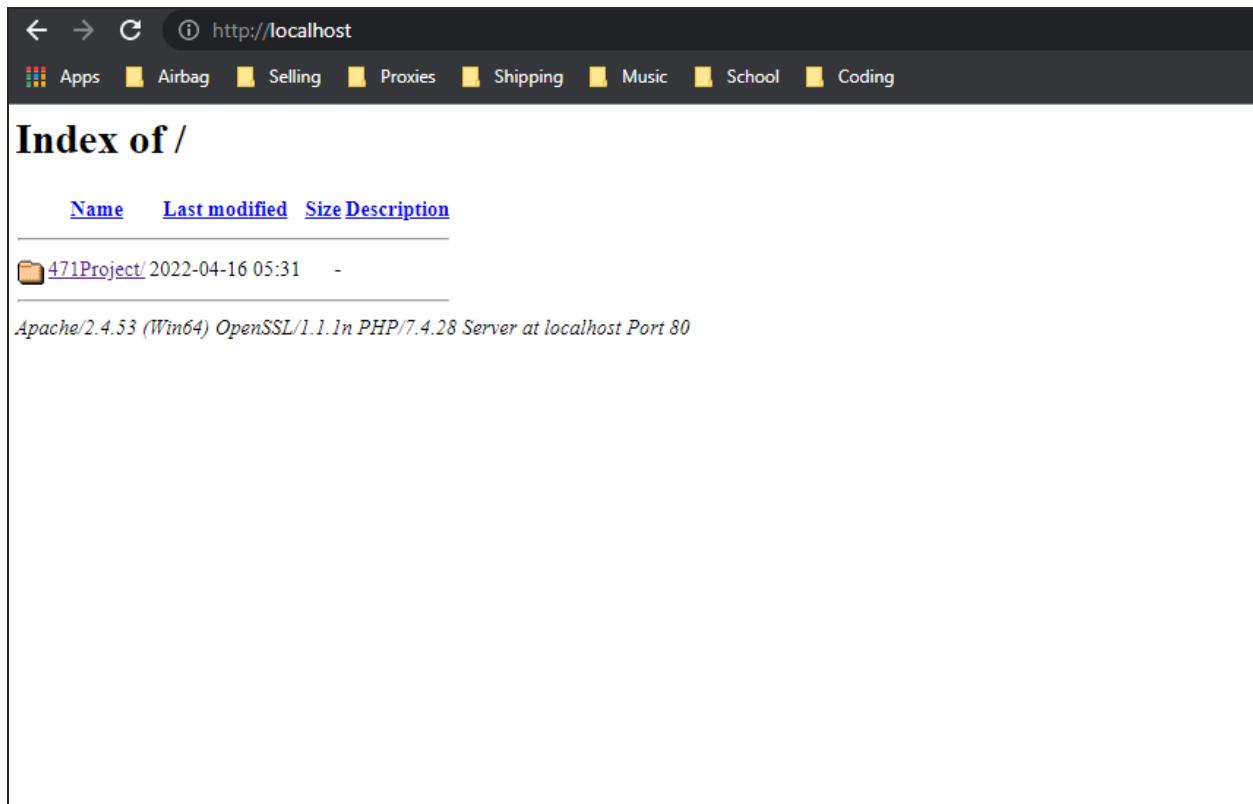
Upon installing XAMPP users will need to start the Apache server and MySql server by clicking on the start button



Users can now open their browser and go <http://localhost/phpmyadmin/> in which they can import the gym database by creating a new database located on the left side of the page and then clicking import and selecting the database file

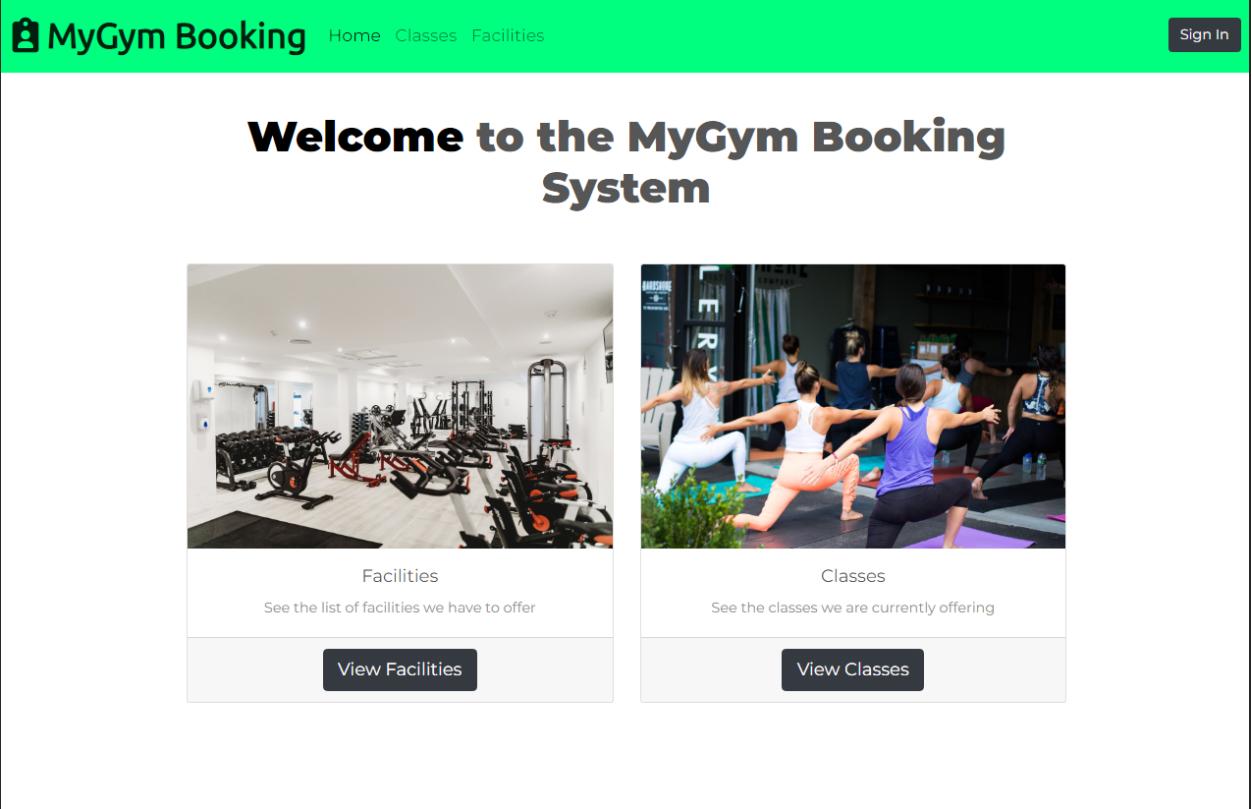
After uploading the database, users will need to place all the folder containing their source code in XAMPP's htdocs folder located at C:\xampp\htdocs (path can vary)

After placing the folder there, the user can access the website through the link <http://localhost/> and needs to select the folder in which it is located in.



The user will now be greeted with the home page of the website and can learn to navigate the website through the guide below

## Pre-login



The screenshot shows the homepage of the MyGym Booking system. At the top, there is a green header bar with the logo 'MyGym Booking' and navigation links for 'Home', 'Classes', and 'Facilities'. On the far right of the header is a 'Sign In' button. Below the header, the main title 'Welcome to the MyGym Booking System' is displayed in a large, bold, black font. The page is divided into two main sections: 'Facilities' on the left and 'Classes' on the right. Each section contains a small image, a title, a brief description, and a 'View Facilities' or 'View Classes' button.

**Facilities**  
See the list of facilities we have to offer  
[View Facilities](#)

**Classes**  
See the classes we are currently offering  
[View Classes](#)

Upon logging in, users will be greeted with the landing of the website and can navigate through the various options. The users can view available classes by clicking view classes. The "Classes" on the navigation bar and the "View Classes" will both lead you to the view available classes as shown below

## View Available Classes

### Cycling

Time: 09:00:00  
Room: Gym 2  
Capacity 9 / 50

### Power Lifting

Time: 09:00:00  
Room: Weight Room 1  
Capacity 8 / 50

### Hot Yoga

Time: 17:00:00  
Room: Gym 3  
Capacity 8 / 50

### Zumba

Time: 09:00:00  
Room: Gym 2  
Capacity 7 / 50

### Cross Fit

Time: 15:00:00  
Room: Weight Room 1  
Capacity 4 / 50

### Swimming

Time: 15:00:00  
Room: Swimming pool  
Capacity 1 / 50

The user can view classes that are currently offered by the gym and notice that the user has no option to join a class as that feature will require the user to log in.

## View Facilities



Facility Name: Gym 1  
Room Number: 1  
Max Capacity: 50



Facility Name: Gym 2  
Room Number: 2  
Max Capacity: 50



Facility Name: Gym 3  
Room Number: 3  
Max Capacity: 50

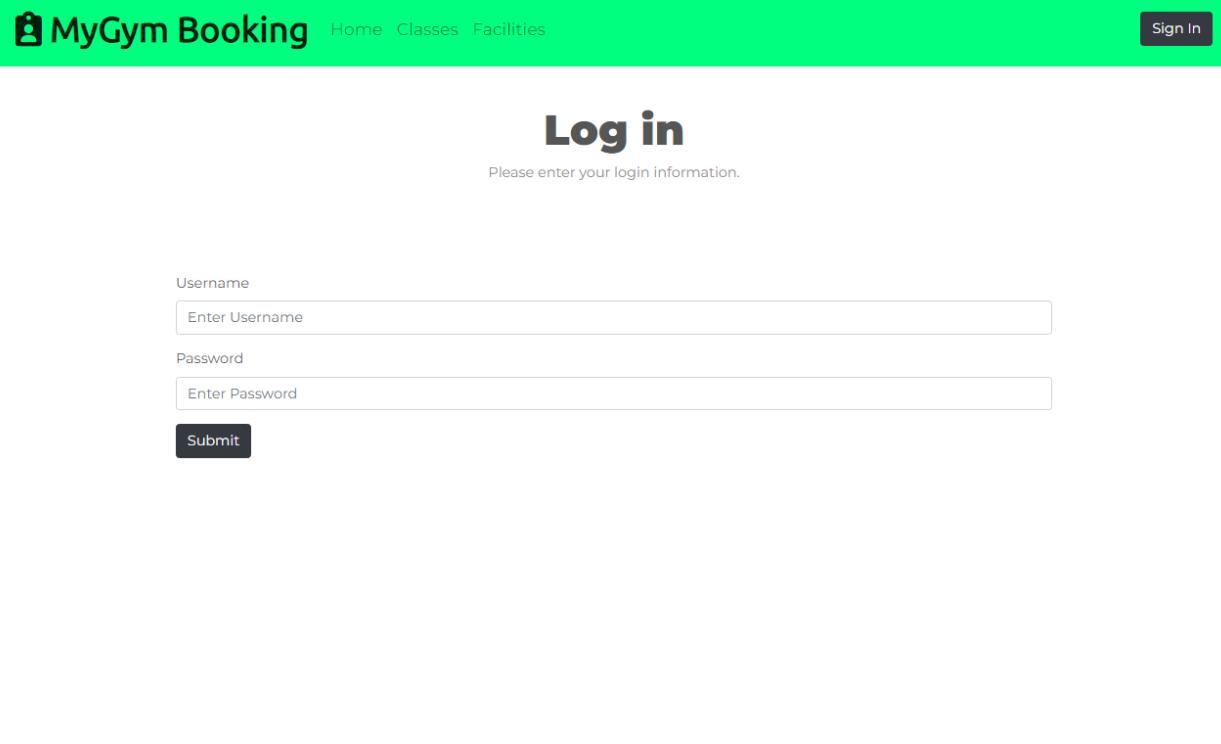


Facility Name: Weight Room 1  
Room Number: 4  
Max Capacity: 50



Facility Name: Swimming pool  
Room Number: 6  
Max Capacity: 50

Users can click “Facilities” on the navigation bar to view available facilities. Note this button will direct you to this page which is the same as the “View Facilities” button on the home page.



The screenshot shows the login page for 'MyGym Booking'. At the top, there's a green header bar with the logo 'MyGym Booking' and navigation links for 'Home', 'Classes', and 'Facilities'. On the right side of the header is a 'Sign In' button. Below the header, the main content area has a title 'Log in' and a sub-instruction 'Please enter your login information.' There are two input fields: 'Username' (with placeholder 'Enter Username') and 'Password' (with placeholder 'Enter Password'). A 'Submit' button is located below the password field.

Users can click the “Sign In” button which will lead you to the sign in page in which users can log in. This application will have 4 different types of users that they can sign in as. These different type of users are: Client, Manager, Janitor and Trainer.

The user will have 4 accounts to log in with:

User: client1 Password: password

User: client2 Password: password

User: client3 Password: password

User: client4 Password: password

The manager can log in with:

User: manager1 Password: password

The Janitor can log in with:

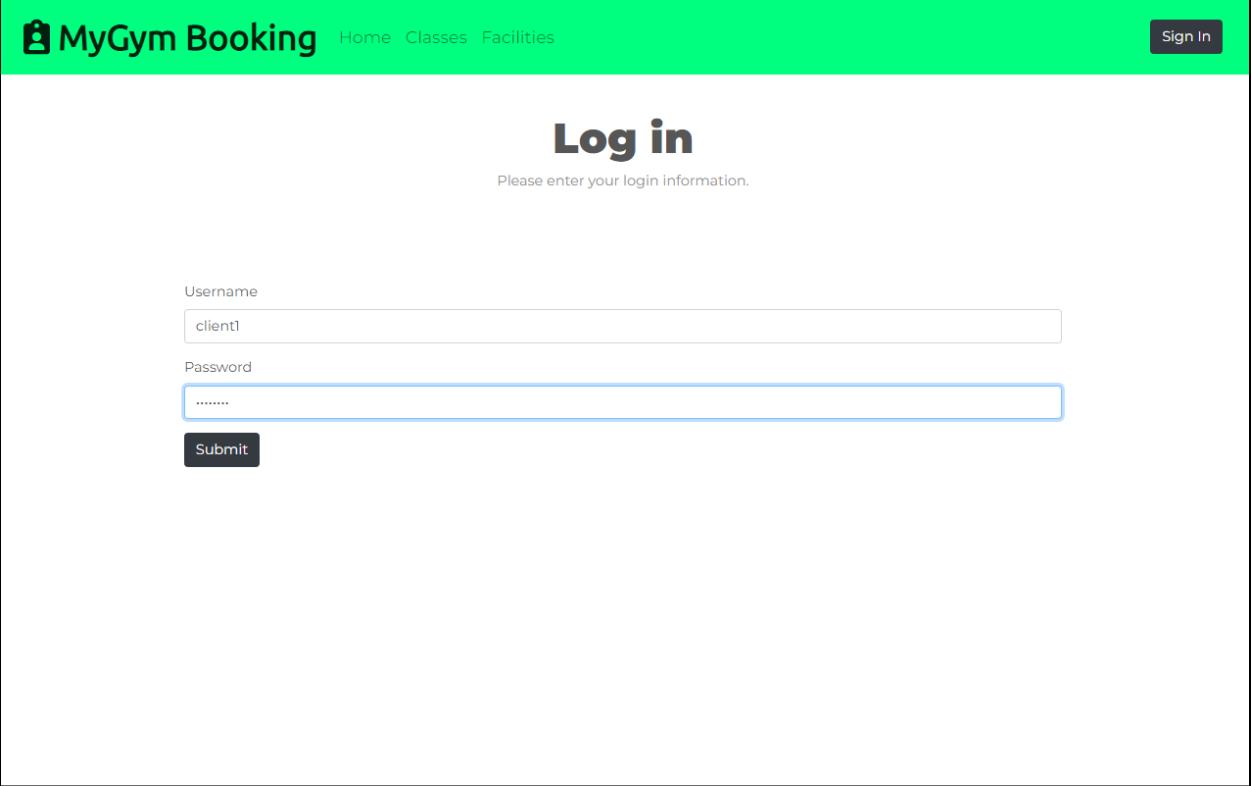
User: janitor1 Password: password

The Trainer can log in with:

User: trainer1 Password: password

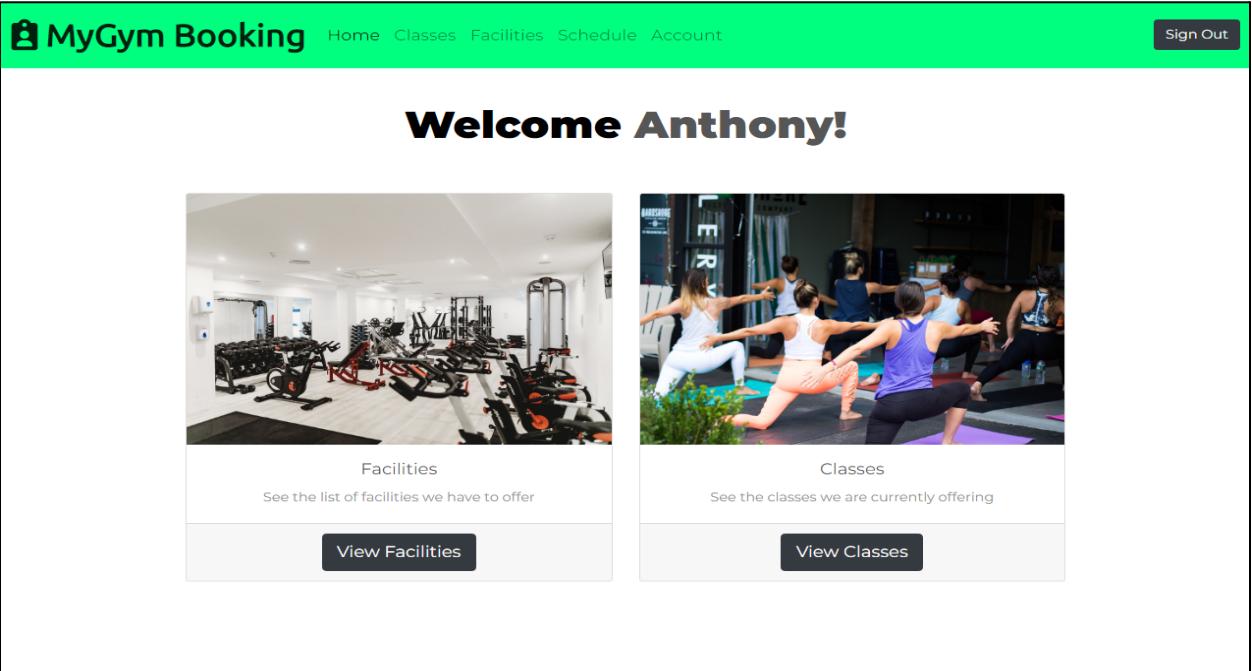
Demonstrated below is the process that each user can go through with the various account types.

## Client



The screenshot shows the 'Log in' page of the MyGym Booking website. At the top, there is a green header bar with the logo 'MyGym Booking' and navigation links for 'Home', 'Classes', and 'Facilities'. On the right side of the header is a 'Sign In' button. Below the header, the page title 'Log in' is displayed in large, bold, black font. A sub-instruction 'Please enter your login information.' is shown in smaller text. The form contains two input fields: 'Username' with the value 'client1' and 'Password' with the value '.....'. A 'Submit' button is located below the password field.

Users can log in with one of the accounts list above.



The screenshot shows the landing page of the MyGym Booking website after logging in. The top navigation bar includes the logo, 'Home', 'Classes', 'Facilities', 'Schedule', 'Account', and a 'Sign Out' button. The main heading 'Welcome Anthony!' is prominently displayed. Below the heading are two cards: 'Facilities' (showing a gym interior) and 'Classes' (showing a group fitness class). Each card has a 'View Facilities' or 'View Classes' button respectively.

Upon logging in they will be greet with the landing page again. The user can click facilities to view the available facilities offered by the gym.

## View Facilities



Facility Name: Gym 1  
Room Number: 1  
Max Capacity: 50



Facility Name: Gym 2  
Room Number: 2  
Max Capacity: 50



Facility Name: Gym 3  
Room Number: 3  
Max Capacity: 50



Facility Name: Weight Room 1  
Room Number: 4  
Max Capacity: 50



Facility Name: Swimming pool  
Room Number: 6  
Max Capacity: 50

Users can see information for each facility along with the capacity and room number.

## View Available Classes

Cycling  
Time: 09:00:00  
Room: Gym 2  
Capacity 8 / 50

Join Class ID:

Power Lifting  
Time: 09:00:00  
Room: Weight Room 1  
Capacity 7 / 50

Join Class ID:

Zumba  
Time: 09:00:00  
Room: Gym 2  
Capacity 6 / 50

Join Class ID:

Cross Fit  
Time: 15:00:00  
Room: Weight Room 1  
Capacity 3 / 50

Join Class ID:

Swimming  
Time: 15:00:00  
Room: Swimming pool  
Capacity 0 / 50

Join Class ID:

Hot Yoga  
Time: 17:00:00  
Room: Gym 1  
Capacity 0 / 50

Join Class ID:

Users can now click the “Classes” button or navigate back to the home page and click “View Classes” in which they will be greeted with the available classes page.

The screenshot shows a user interface for managing bookings. At the top, there is a navigation bar with the logo 'MyGym Booking' and links for 'Classes', 'Facilities', 'Schedule', and 'Account'. On the far right of the bar is a 'Sign Out' button. Below the navigation bar, the main title 'Your Bookings' is displayed in a large, bold, dark font. Underneath the title, there is a header row with six columns: 'Class ID', 'Booking ID', 'Class Name', 'Time', 'Room Number', and 'Actions'. The rest of the page is currently empty, indicating no bookings have been made.

Users can click “Schedule” to view the classes they are currently enrolled in.

The screenshot shows a list of available classes. At the top, there is a navigation bar with the logo 'MyGym Booking' and links for 'Home', 'Classes', 'Facilities', 'Schedule', and 'Account'. On the far right of the bar is a 'Sign Out' button. Below the navigation bar, the main title 'View Available Classes' is displayed in a large, bold, dark font. The page lists six classes in a grid format:

Cycling	Power Lifting	Zumba	Cross Fit	Swimming	Hot Yoga
Time: 09:00:00 Room: Gym 2 Capacity 9 / 50	Time: 09:00:00 Room: Weight Room 1 Capacity 8 / 50	Time: 09:00:00 Room: Gym 2 Capacity 7 / 50	Time: 15:00:00 Room: Weight Room 1 Capacity 4 / 50	Time: 15:00:00 Room: Swimming pool Capacity 1 / 50	Time: 17:00:00 Room: Gym 1 Capacity 1 / 50
Join Class ID: <a href="#">13</a>	Join Class ID: <a href="#">10</a>	Join Class ID: <a href="#">655</a>	Join Class ID: <a href="#">214</a>	Join Class ID: <a href="#">456</a>	Join Class ID: <a href="#">9</a>

Users can now navigate back to the available classes page and join the classes they desire by clicking on the class ID. In this case, our users joins all the classes.

## Your Bookings

Class ID	Booking ID	Class Name	Time	Room Number	Actions
13	274	Cycling	09:00:00	2	
10	275	Power Lifting	09:00:00	4	
655	276	Zumba	09:00:00	2	
214	277	Cross Fit	15:00:00	4	
456	278	Swimming	15:00:00	6	
9	279	Hot Yoga	17:00:00	1	

Users can now click "Schedule" to view the classes they enrolled in.

## Your Bookings

Class ID	Booking ID	Class Name	Time	Room Number	Actions
10	275	Power Lifting	09:00:00	4	
655	276	Zumba	09:00:00	2	
214	277	Cross Fit	15:00:00	4	
456	278	Swimming	15:00:00	6	
9	279	Hot Yoga	17:00:00	1	

Users can unenroll in a class by clicking on the red trash can button.

## Account information

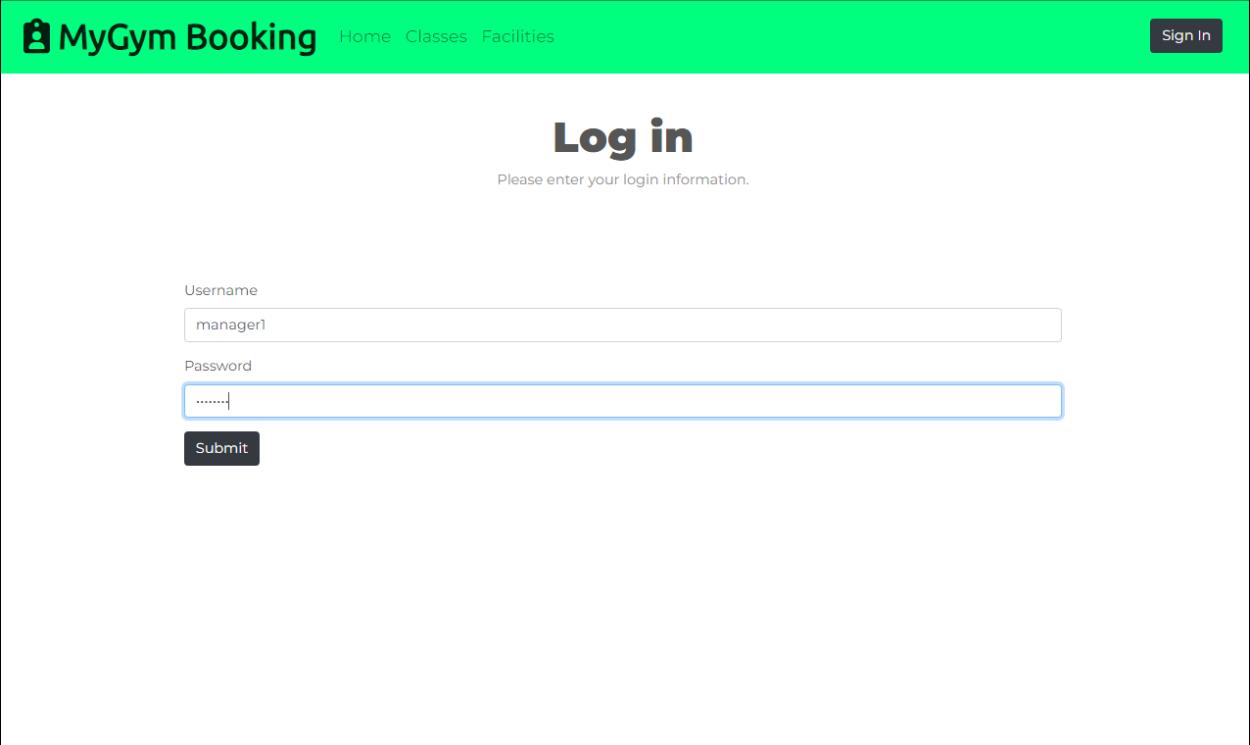
First Name	Last Name	Birthdate	Reoccuring payment date	Membership ID
Anthony	Pit	0200-04-12	2022-04-16	2359

Change your password

 passwordSubmit

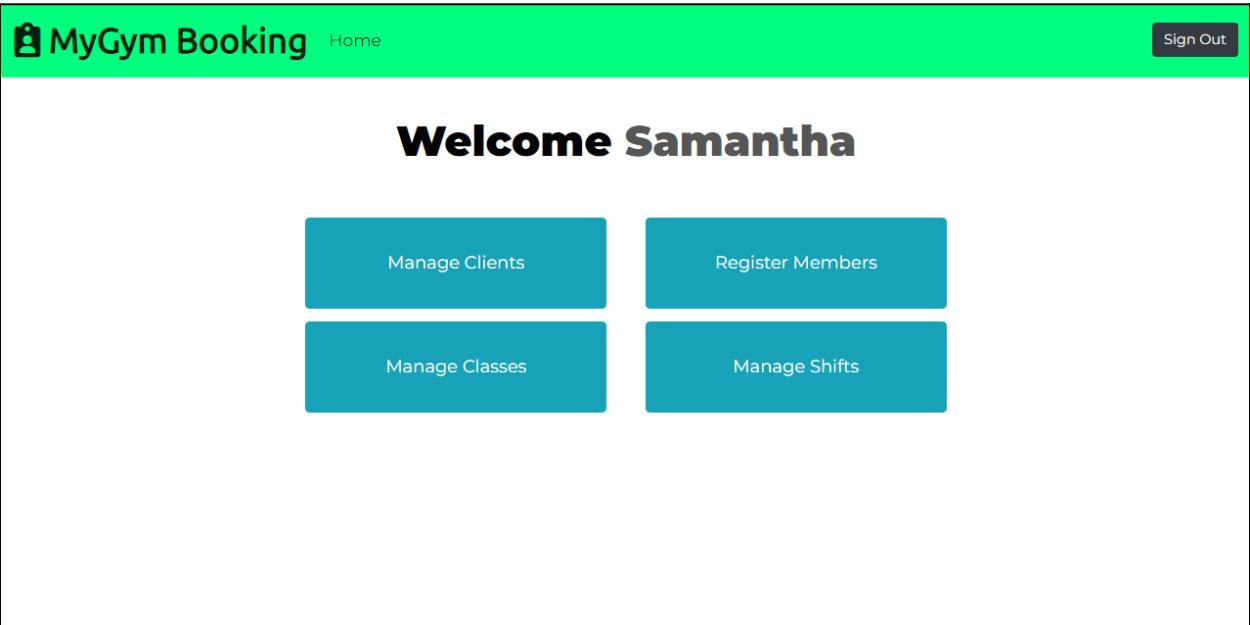
Users can navigate to view their account information by clicking on the “Account” button on the navigation bar in which they can change only their password as they do not have admin privileges. Editing other information will require a manager account.

## Manager



The screenshot shows the MyGym Booking manager login page. At the top, there is a green header bar with the logo "MyGym Booking" and navigation links for "Home", "Classes", and "Facilities". On the right side of the header is a "Sign In" button. Below the header, the main content area has a title "Log in" and a sub-instruction "Please enter your login information." There are two input fields: "Username" containing "manager1" and "Password" containing a masked value. A "Submit" button is located below the password field.

The manager can log in with the manager log in.



The screenshot shows the MyGym Booking manager landing page. At the top, there is a green header bar with the logo "MyGym Booking", a "Home" link, and a "Sign Out" button. Below the header, the main content area features a large, bold "Welcome Samantha" message. It includes four teal-colored buttons arranged in a 2x2 grid, each labeled with a management task: "Manage Clients", "Register Members", "Manage Classes", and "Manage Shifts".

Upon logging in, they will be greeted with the manager landing page and can select the various options.

The screenshot shows the "Manage Clients" page of the MyGym Booking application. At the top, there is a navigation bar with the title "MyGym Booking" and a "Home" link. On the right side of the navigation bar is a "Sign Out" button. Below the navigation bar, the main title "Manage Clients" is centered. The page contains two tables. The first table is a list of clients with columns: Username, First Name, Last Name, Password, and Vaccination Status. Each row has a blue edit icon in the last column. The second table is a form for updating client information, with fields for User Name, First Name, Last Name, Password, and Vaccination Status. The "Update" button is located at the bottom right of this form.

Username	First Name	Last Name	Password	Vaccination Status
client2	Kevin	Cool	password	1
client1	Anthony	Pit	password	0
client3	Deane	Gale	password	0
client4	Kim	Possible	password	1

User Name	First Name	Last Name	Password	Vaccination Status
<input type="text"/>				

**Update**

If the manager selects the “Manage Clients” button, they will be greeted with a screen in which they can update client information and can expedite the process by clicking on the blue edit button to auto fill the information into the field below

This screenshot shows the "Manage Clients" page with the client information from the previous screenshot now populated into the update form. The "User Name" field contains "client2", the "First Name" field contains "Kevin", the "Last Name" field contains "Cool", the "Password" field contains "password", and the "Vaccination Status" field contains "1". The "Update" button is visible at the bottom right of the form.

Username	First Name	Last Name	Password	Vaccination Status
client2	Kevin	Cool	password	1
client1	Anthony	Pit	password	0
client3	Deane	Gale	password	0
client4	Kim	Possible	password	1

User Name	First Name	Last Name	Password	Vaccination Status
client2	Kevin	Cool	password	1

**Update**

Now the user can change client information and by inputting the data into the fields and in this case, they are changing the vaccination status of Kevin Cool. Note that the SSN, Birthdate, Start date and account type cannot be changed as those are assigned during member registration and act as identifiers for the client.

**MyGym Booking** Home Sign Out

## Manage Clients

Username	First Name	Last Name	Password	Vaccination Status
client2	Kevin	Cool	password	0
client1	Anthony	Pit	password	0
client3	Deane	Gale	password	0
client4	Kim	Possible	password	1

User Name First Name Last Name Password Vaccination Status

client2

Upon hitting update, the user information will be updated

**MyGym Booking** Home Sign Out

## Welcome Samantha

Manage Clients Register Members

Manage Classes Manage Shifts

The user can navigate back to the manager landing page by clicking on the “Home” button on the navigation bar.

**MyGym Booking** Home Sign Out

## Register member

---

SSN	First Name	Last Name	Birthdate
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> mm/dd/yyyy <input type="button" value=""/>
Username	Password	Vaccination status	
<input type="text"/>	<input type="text"/>	<input type="text"/>	

**Add Member**

The user can navigate to the register member page by clicking on the “Register Members” button and be greeted with a screen to register new members of the gym. Here they can input the user information.

**MyGym Booking** Home Sign Out

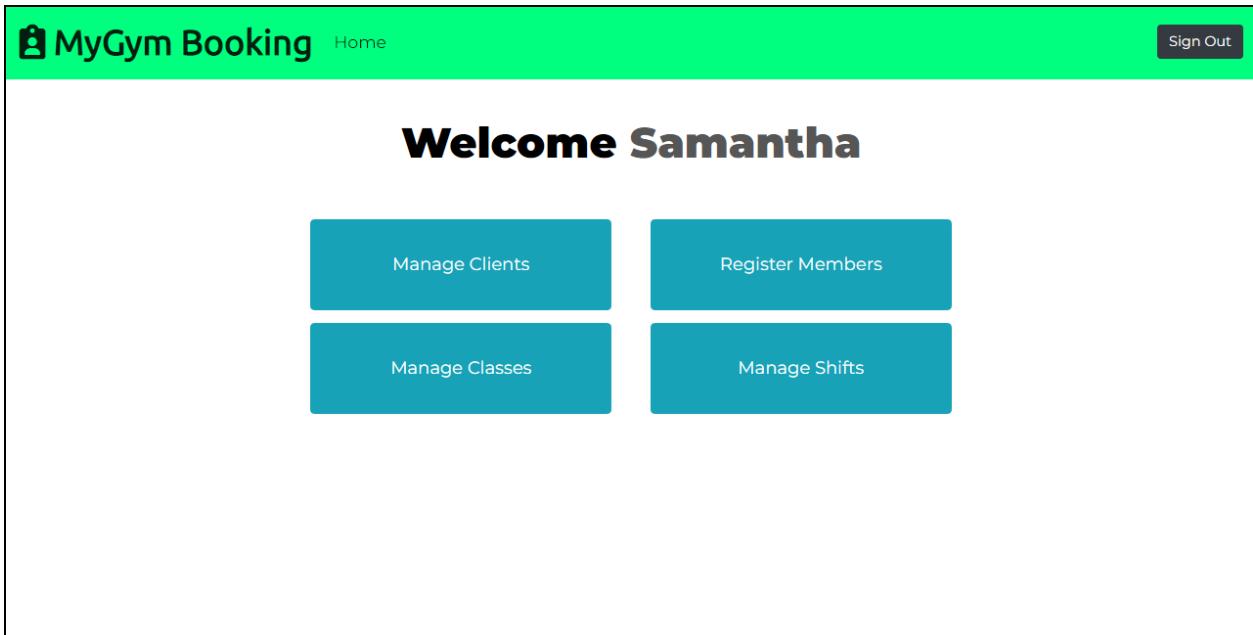
## Register member

---

SSN	First Name	Last Name	Birthdate
<input type="text"/> 213	<input type="text"/> Victor	<input type="text"/> Bell	<input type="text"/> 12/31/2022 <input type="button" value=""/>
Username	Password	Vaccination status	
<input type="text"/> client5	<input type="text"/> password	<input type="text"/> 0	

**Add Member**

The manager will now input the new clients details and hit add.

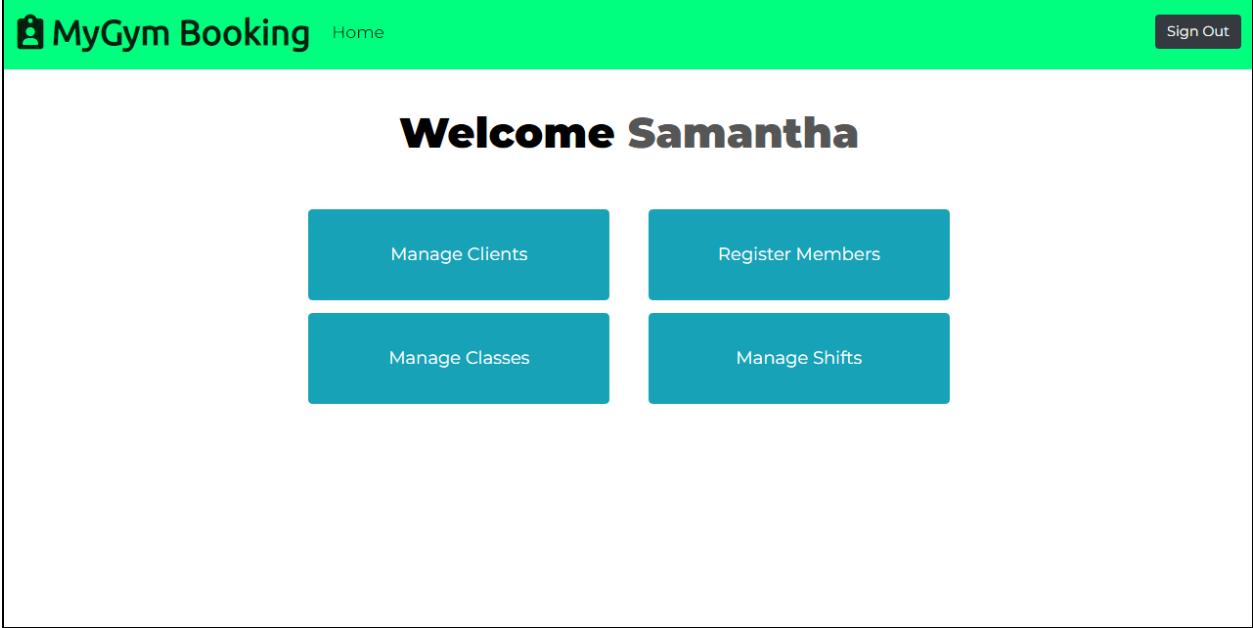


The user can navigate back to the manager landing page by clicking on the “Home” button on the navigation bar.

This screenshot shows the 'Manage Clients' page. The top navigation bar includes the 'MyGym Booking' logo, a 'Home' link, and a 'Sign Out' button. The main title 'Manage Clients' is centered above a table. The table has columns for 'Username', 'First Name', 'Last Name', 'Password', and 'Vaccination Status'. It lists five clients: client5, client2, client1, client3, and client4. Each row contains an edit icon. Below the table is a form with input fields for 'User Name', 'First Name', 'Last Name', 'Password', and 'Vaccination Status', followed by an 'Update' button.

Username	First Name	Last Name	Password	Vaccination Status
client5	Victor	Bell	password	0
client2	Kevin	Cool	password	0
client1	Anthony	Pit	password	0
client3	Deane	Gale	password	0
client4	Kim	Possible	password	1

They can navigate back to the manage clients page to view the new user they have added.



The user can navigate back to the manager landing page by clicking on the “Home” button on the navigation bar.


Home
Sign Out

## Manage Classes

Class ID	Name	Trainer SSN	Time	Number of Participants	Room Number	Facility Name	
9	Hot Yoga	123	17:00:00	8	3	Gym 3	 
10	Power Lifting	123	09:00:00	8	4	Weight Room 1	 
13	Cycling	123	09:00:00	9	2	Gym 2	 
214	Cross Fit	123	15:00:00	4	4	Weight Room 1	 
456	Swimming	123	15:00:00	1	6	Swimming pool	 
655	Zumba	123	09:00:00	7	2	Gym 2	 

Class ID	SSN	Name	Time	Room Number
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> - : - : -	<input type="text"/> 

Add Class

The manager can navigate to the manage classes page in which they view all the classes they currently offer by clicking on the “Manage Classes” button on the manager home page.

⌚ MyGym Booking Home
Sign Out

## Manage Classes

Class ID	Name	Trainer SSN	Time	Number of Participants	Room Number	Facility Name	
9	Hot Yoga	123	17:00:00	1	3	Gym 3	<span style="color: blue;">edit</span> <span style="color: red;">delete</span>
10	Power Lifting	123	09:00:00	8	4	Weight Room 1	<span style="color: blue;">edit</span> <span style="color: red;">delete</span>
13	Cycling	123	09:00:00	8	2	Gym 2	<span style="color: blue;">edit</span> <span style="color: red;">delete</span>
214	Cross Fit	123	15:00:00	4	4	Weight Room 1	<span style="color: blue;">edit</span> <span style="color: red;">delete</span>
456	Swimming	123	15:00:00	1	6	Swimming pool	<span style="color: blue;">edit</span> <span style="color: red;">delete</span>
655	Zumba	123	09:00:00	7	2	Gym 2	<span style="color: blue;">edit</span> <span style="color: red;">delete</span>

Class ID	SSN	Name	Time	Room Number
9	123	Hot Yoga	05:00 PM <input type="button" value="🕒"/>	<input type="button" value="2"/> <span style="color: green; border: 1px solid green; padding: 2px 5px;">Update Class</span>

The manager can click the blue edit button to update a class in which the green submit button will change to say update class and the information will be autofilled. Our manager wishes to change the facility to gym 2.

## Manage Classes

Class ID	Name	Trainer SSN	Time	Number of Participants	Room Number	Facility Name	
9	Hot	123	17:00:00	8	2	Gym 2	 
10	Power Lifting	123	09:00:00	8	4	Weight Room 1	 
13	Cycling	123	09:00:00	9	2	Gym 2	 
214	Cross Fit	123	15:00:00	4	4	Weight Room 1	 
456	Swimming	123	15:00:00	1	6	Swimming pool	 
655	Zumba	123	09:00:00	7	2	Gym 2	 

Class ID	SSN	Name	Time	Room Number
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> --::-- <input type="text"/>	<input type="text"/>

[Add Class](#)

Upon clicking the update button, the class will be updated and the “Add class” button will reappear.

## Manage Classes

Class ID	Name	Trainer SSN	Time	Number of Participants	Room Number	Facility Name	
10	Power Lifting	123	09:00:00	8	4	Weight Room 1	 
13	Cycling	123	09:00:00	8	2	Gym 2	 
214	Cross Fit	123	15:00:00	4	4	Weight Room 1	 
456	Swimming	123	15:00:00	1	6	Swimming pool	 
655	Zumba	123	09:00:00	7	2	Gym 2	 

Class ID	SSN	Name	Time	Room Number
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> : <input type="text"/> : <input type="text"/>	<input type="text"/>

A Manager can also delete a class by clicking on the red trash can button. The page will auto refresh with the new list of classes available. In this case, the manager has deleted the class with ID 9.

⌚ MyGym Booking Home
Sign Out

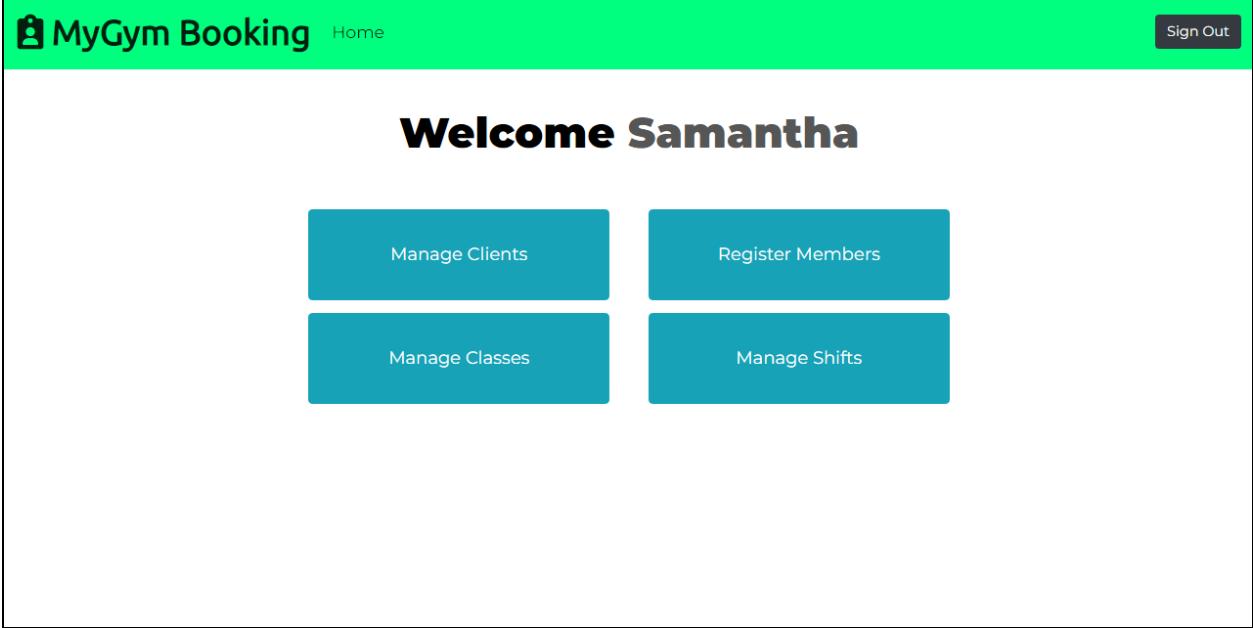
## Manage Classes

Class ID	Name	Trainer SSN	Time	Number of Participants	Room Number	Facility Name	
10	Power Lifting	123	09:00:00	8	4	Weight Room 1	<span style="color: blue; font-size: 1.5em;">✎</span> <span style="color: red; font-size: 1.5em;">✖</span>
13	Cycling	123	09:00:00	8	2	Gym 2	<span style="color: blue; font-size: 1.5em;">✎</span> <span style="color: red; font-size: 1.5em;">✖</span>
214	Cross Fit	123	15:00:00	4	4	Weight Room 1	<span style="color: blue; font-size: 1.5em;">✎</span> <span style="color: red; font-size: 1.5em;">✖</span>
456	Swimming	123	15:00:00	1	6	Swimming pool	<span style="color: blue; font-size: 1.5em;">✎</span> <span style="color: red; font-size: 1.5em;">✖</span>
655	Zumba	123	09:00:00	7	2	Gym 2	<span style="color: blue; font-size: 1.5em;">✎</span> <span style="color: red; font-size: 1.5em;">✖</span>

Class ID	SSN	Name	Time	Room Number	
<input type="text" value="231"/>	<input type="text" value="123"/>	<input type="text" value="Gymnastics"/>	<input type="text" value="12:00 PM"/> <input type="button" value="🕒"/>	<input type="text" value="3"/>	<input type="button" value="Add Class"/>

The manager can also add new classes to the gym by inputting the data into the fields. The manager will have to input the class ID, SSN of the trainer teaching it, name of the class, time and the room number for which it will be held in.



The user can navigate back to the manager landing page by clicking on the “Home” button on the navigation bar.

**Manage Shifts**

SSN	Employee First Name	Employee Last Name	Date	Time	Facility	Room Number	
321	Brendan	Bob	2022-04-14	12:00:00-15:00:00	Gym 2	2	
123	Victor	Bill	2022-04-14	09:00:00-17:00:00	Swimming pool	6	
741	Samantha	House	2022-04-14	09:00:00-17:00:00	Weight Room 1	4	
321	Brendan	Bob	2022-04-14	09:00:00-12:00:00	Gym 1	1	
741	Samantha	House	2022-04-15	09:00:00-17:00:00	Gym 2	2	
321	Brendan	Bob	2022-04-15	09:00:00-12:00:00	Gym 1	1	
321	Brendan	Bob	2022-04-15	12:00:00-15:00:00	Gym 2	2	
321	Brendan	Bob	2022-04-15	15:00:00-	Gym 3	3	

The manager can navigate to the manage shifts page by clicking on the “Manage Shifts” button. The manager can choose to update, delete or add shifts to a trainer or janitor.

SSN	Employee First Name	Employee Last Name	Date	Time	Facility	Room Number	
123	Victor	Bill	2022-04-14	09:00:00-17:00:00	Gym 1	1	
741	Samantha	House	2022-04-14	09:00:00-17:00:00	Weight Room 1	4	
321	Brendan	Bob	2022-04-15	12:00:00-15:00:00	Gym 2	2	
321	Brendan	Bob	2022-04-15	15:00:00-17:00:00	Gym 3	3	
741	Samantha	House	2022-04-15	09:00:00-17:00:00	Gym 2	2	
321	Brendan	Bob	2022-04-15	09:00:00-12:00:00	Gym 1	1	
741	Samantha	House	2022-04-15	09:00:00-17:00:00	Swimming	6	

If the manager clicks the red trash can button, the shift will be deleted from the system as shown above with the employee Brendan Bob. They can also click on the blue button and update a shift.

SSN	Date	Start Time	End Time	Room Number
123	04/14/2022	09:00 AM	05:00 PM	2

**Update Shift**

The manager has clicked on the edit button for Victor Bull's April 14th shift and the information is auto filled and the update shift button will appear. The manager has chosen to change the location of the shift to room 2.

SSN	Employee First Name	Employee Last Name	Date	Time	Facility	Room Number	
123	Victor	Bill	2022-04-14	09:00:00-17:00:00	Gym 2	2	
741	Samantha	House	2022-04-14	09:00:00-17:00:00	Weight Room 1	4	
321	Brendan	Bob	2022-04-15	15:00:00-17:00:00	Gym 3	3	
321	Brendan	Bob	2022-04-15	12:00:00-15:00:00	Gym 2	2	
321	Brendan	Bob	2022-04-15	09:00:00-12:00:00	Gym 1	1	
741	Samantha	House	2022-04-15	09:00:00-17:00:00	Gym 2	2	
741	Samantha	House	2022-04-16	09:00:00-17:00:00	Swimming pool	6	

As seen in the picture above, Victor Bull's shift has been updated

321	Brendan	Bob	2022-04-30	12:00:00-15:00:00	Gym 2	2	
123	Victor	Bill	2022-04-30	09:00:00-17:00:00	Weight Room 1	4	
321	Brendan	Bob	2022-04-30	09:00:00-12:00:00	Gym 1	1	

SSN	Date	Start Time	End Time	Room Number
123	05/01/2022	09:00 AM	05:00 PM	1

**Add Shift**

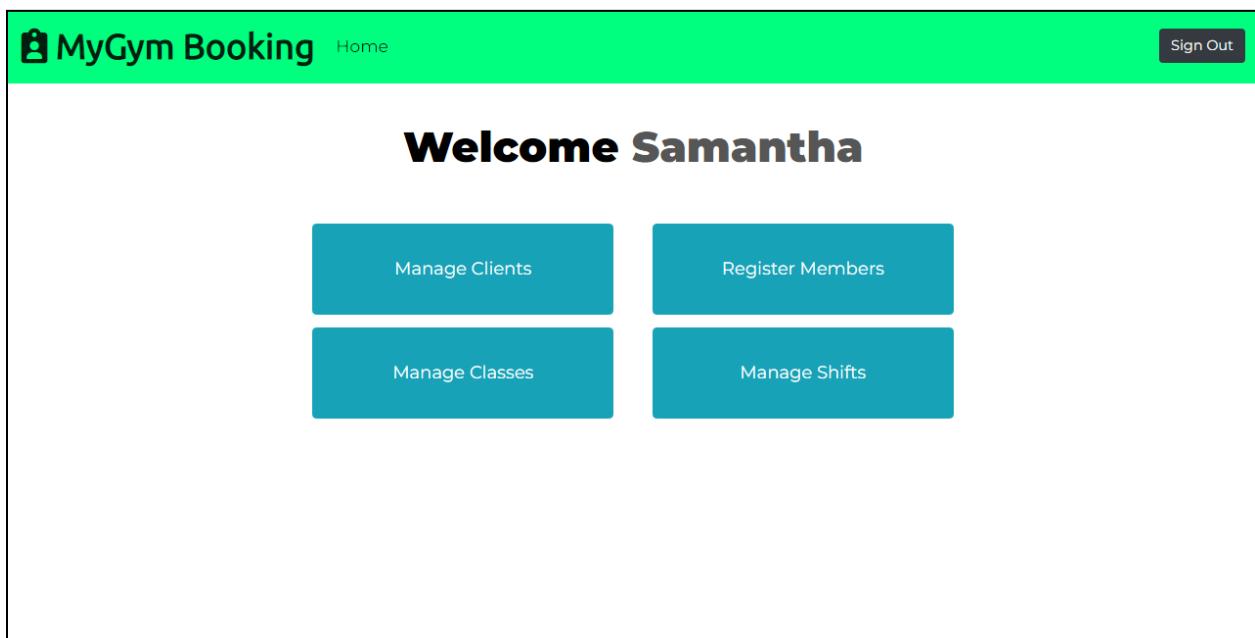
Managers also have the option of adding a shift to by scrolling to the bottom and inputting the data and hitting “Add Shift”

321	Brendan	Bob	2022-04-30	15:00:00-17:00:00	Gym 3	3	 
321	Brendan	Bob	2022-04-30	12:00:00-15:00:00	Gym 2	2	 
123	Victor	Bill	2022-05-01	09:00:00-17:00:00	Gym 1	1	 

SSN	Date	Start Time	End Time	Room Number
<input type="text"/>	<input type="text"/> mm/dd/yyyy 	<input type="text"/> --:-- -- 	<input type="text"/> --:-- -- 	<input type="text"/>
<a href="#">Add Shift</a>				

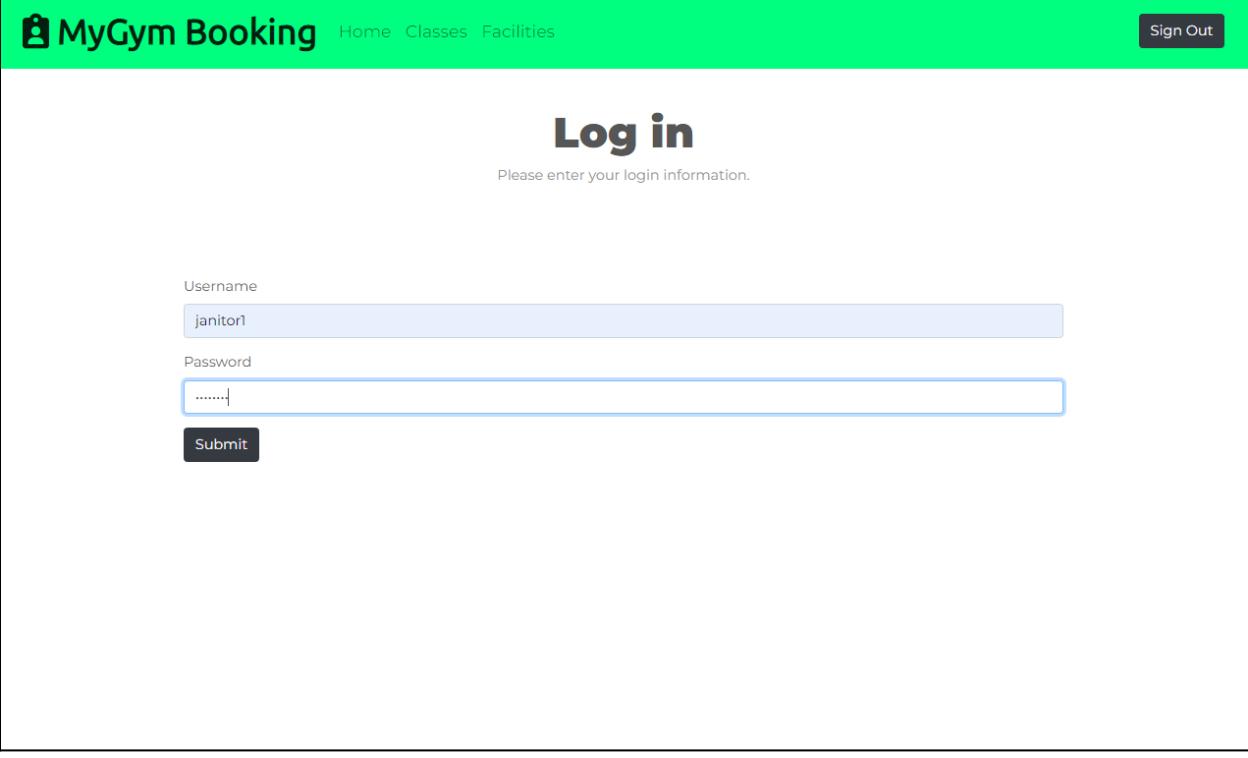
As seen, the shift has now been added.



The screenshot shows the MyGym Booking application's home page. At the top, there is a navigation bar with the logo "MyGym Booking", a "Home" link, and a "Sign Out" button. The main heading "Welcome Samantha" is displayed prominently. Below the heading are four teal-colored buttons arranged in a 2x2 grid, each labeled with a management function: "Manage Clients", "Register Members", "Manage Classes", and "Manage Shifts".

The user can navigate back to the manager landing page by clicking on the “Home” button on the navigation bar.

## Janitor



The screenshot shows the 'MyGym Booking' login page. At the top, there is a green header bar with the logo 'MyGym Booking' and navigation links for 'Home', 'Classes', and 'Facilities'. On the right side of the header is a 'Sign Out' button. Below the header, the main content area has a title 'Log in' and a subtitle 'Please enter your login information.' In the center, there are two input fields: one for 'Username' containing 'janitor1' and another for 'Password' containing several dots. A 'Submit' button is located below the password field.

A Janitor can log in with the details provided above.

 MyGym Booking

Home Cleaning

Sign Out

## Welcome Brendan



View Cleaning Schedule

Upon logging in, the janitor will be greeted with their employee specific landing page.

 MyGym Booking

Home Cleaning

Sign Out

## Today's Cleaning Schedule

Facility Name	Room Number	Time	Complete
Gym 2	2	12:00:00	<input type="checkbox"/>
Gym 3	3	15:00:00	<input type="checkbox"/>

Submit

## Room's already cleaned

Facility Name	Room Number
Gym 1	1

A janitor can click the “View Cleaning Schedule” button or click the “Cleaning” button on the navigation bar to navigate to today's cleaning schedule screen.

 MyGym Booking [Home](#) [Cleaning](#) [Sign Out](#)

## Today's Cleaning Schedule

Facility Name	Room Number	Time	Complete
Gym 2	2	12:00:00	<input checked="" type="checkbox"/>
Gym 3	3	15:00:00	<input type="checkbox"/>

[Submit](#)

## Room's already cleaned

Facility Name	Room Number
Gym 1	1

A janitor can hit the check boxes then submit to indicate that a room has been cleaned.

 MyGym Booking [Home](#) [Cleaning](#) [Sign Out](#)

## Today's Cleaning Schedule

Facility Name	Room Number	Time	Complete
Gym 3	3	15:00:00	<input type="checkbox"/>

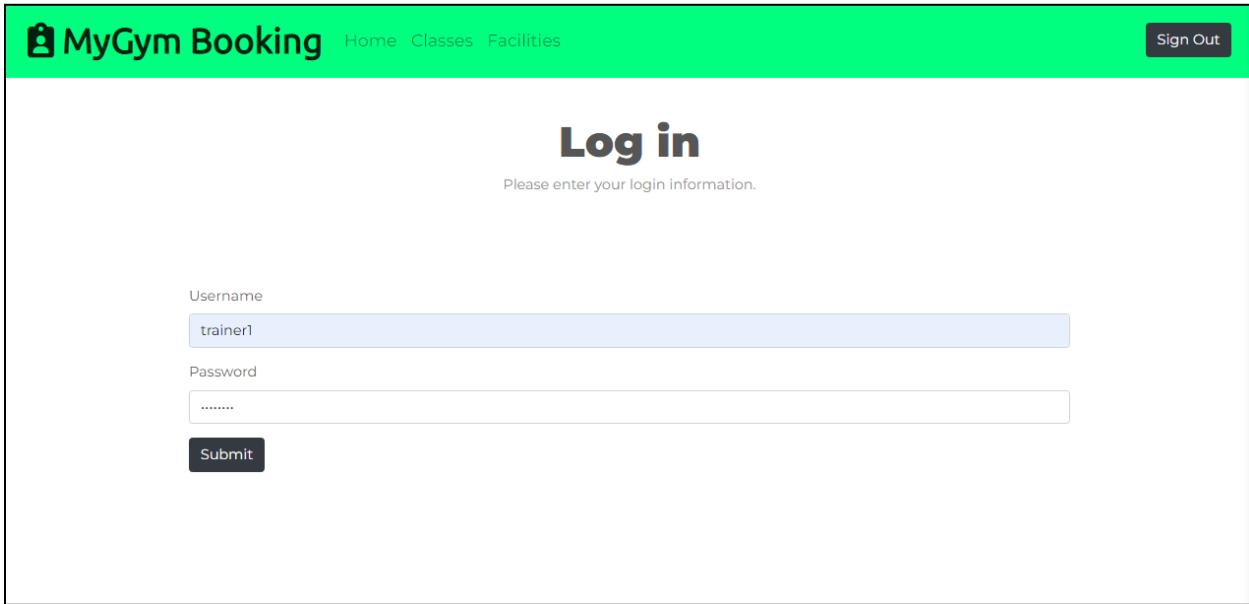
[Submit](#)

## Room's already cleaned

Facility Name	Room Number
Gym 1	1
Gym 2	2

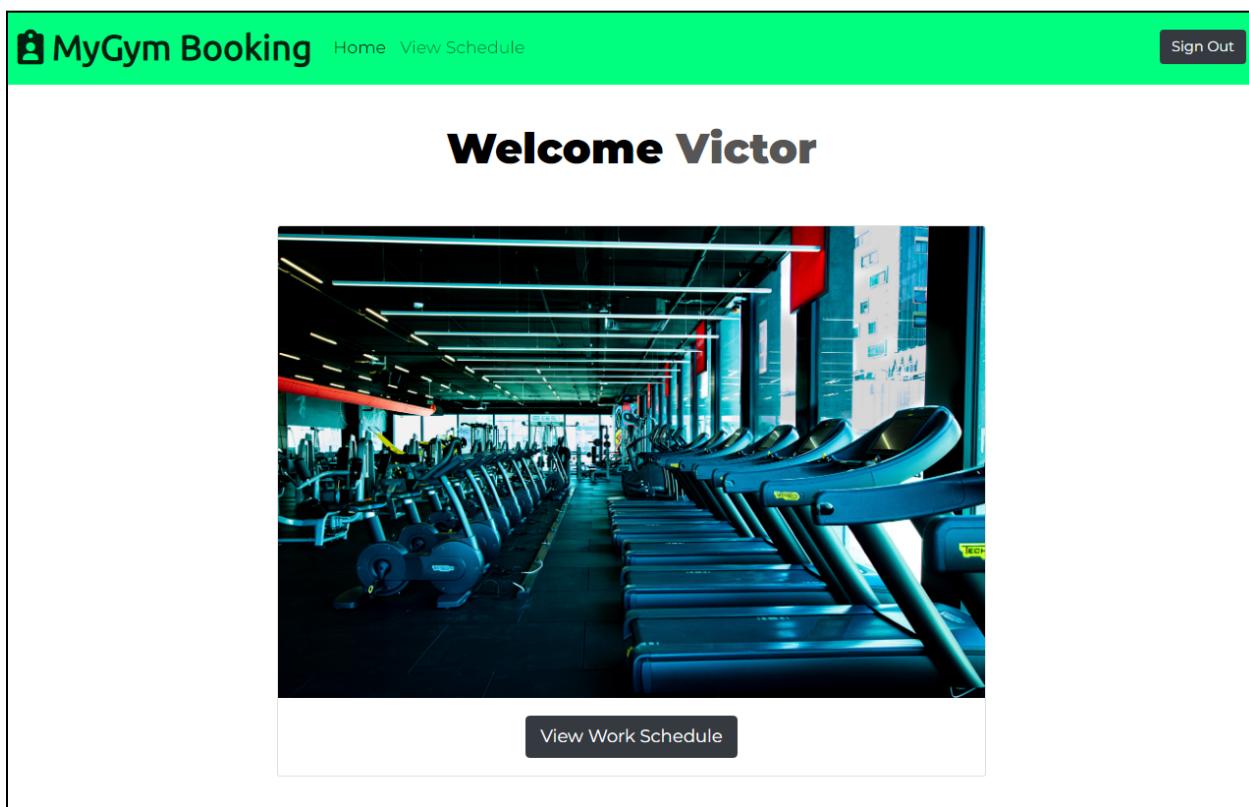
As seen above, the page has been updated to reflect these changes.

## Trainer



The screenshot shows the 'Log in' page of the MyGym Booking application. At the top, there is a green header bar with the logo 'MyGym Booking' and navigation links for 'Home', 'Classes', and 'Facilities'. On the right side of the header is a 'Sign Out' button. Below the header, the main content area has a title 'Log in' and a sub-instruction 'Please enter your login information.'. There are two input fields: 'Username' containing 'trainer1' and 'Password' containing '.....'. A 'Submit' button is located below the password field.

A trainer can log in with the information provided in the Pre-login section.



The trainer will be greeted with the landing page and click on the “View Work Schedule” or “View Schedule” button on the navigation bar.

## My work Schedule

Facility Name	Date	Start Time	End Time
Gym 2	2022-04-14	09:00:00	17:00:00
Weight Room 1	2022-04-16	09:00:00	17:00:00
Gym 3	2022-04-17	09:00:00	17:00:00
Gym 2	2022-04-18	09:00:00	17:00:00
Weight Room 1	2022-04-19	09:00:00	17:00:00
Swimming pool	2022-04-20	09:00:00	17:00:00
Gym 1	2022-04-21	09:00:00	17:00:00
Gym 2	2022-04-22	09:00:00	17:00:00
Weight Room 1	2022-04-23	09:00:00	17:00:00
Gym 3	2022-04-25	09:00:00	17:00:00
Gym 3	2022-04-26	09:00:00	17:00:00
Gym 1	2022-04-27	09:00:00	17:00:00
Gym 2	2022-04-28	09:00:00	17:00:00
Swimming pool	2022-04-29	09:00:00	17:00:00
Weight Room 1	2022-04-30	09:00:00	17:00:00
Gym 1	2022-05-01	09:00:00	17:00:00

When the trainer clicks on one of the two buttons, they will be directed to a page in which they can view their schedule.

## References:

Digamber, D. I. am, & Digamber. (2022, April 10). *Create PHP 8 CRUD REST API with MySQL & PHP PDO*. positronX.io. Retrieved April 16, 2022, from <https://www.positronx.io/create-simple-php-crud-rest-api-with-mysql-php-pdo/>

The University of Calgary. (n.d.). *Fitness Center Booking*. Retrieved January 31, 2022, from <https://iac01.ucalgary.ca/CamRecWebBooking/default.aspx>

ToThePointCode. (2021, May 19). PHP Crud Application FT mysql, JQuery. YouTube. Retrieved April 16, 2022, from <https://www.youtube.com/playlist?list=PLk8gdrb2DmCjGHFH1iQxdJUTWsxyceSO0>

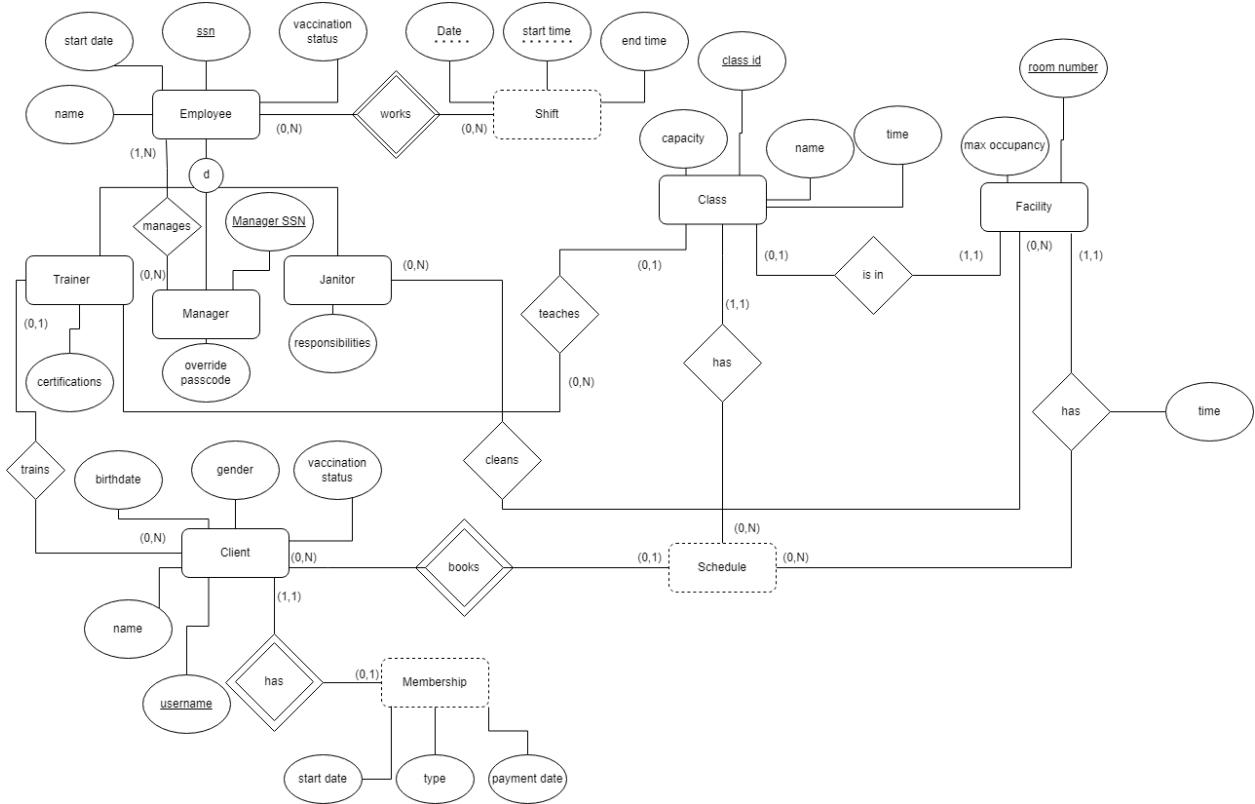
PHP tutorial. (n.d.). Retrieved April 16, 2022, from <https://www.w3schools.com/php/default.asp>

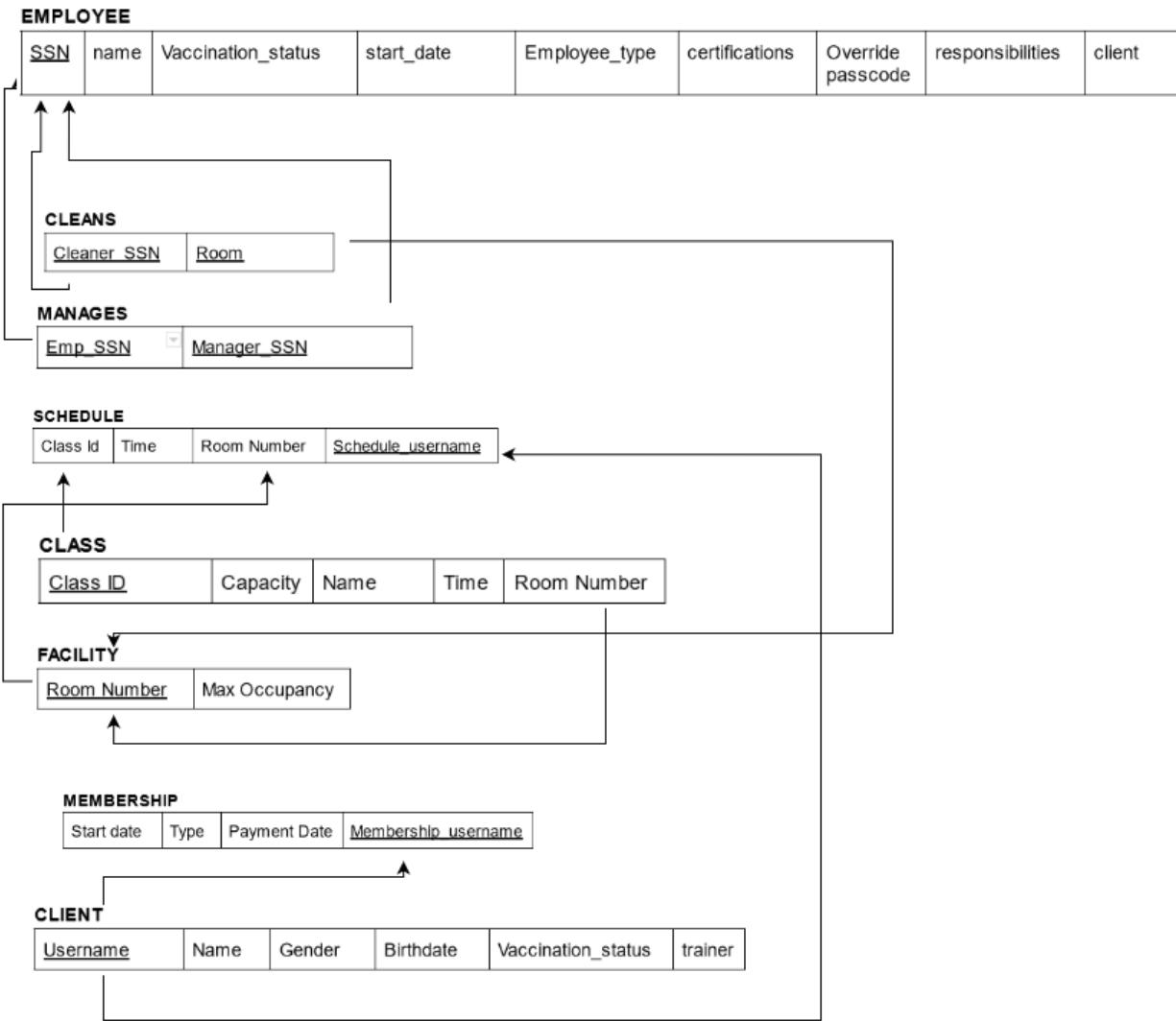
Some code was referenced from notes on D2L and Zoom videos

Tutorial\_W09\_API\_Winter2022

Tutorial\_W08\_PHP\_SQL\_Winter2022

# Appendix





Our project has changed since the presentation as we had some bugs in our code and issues with our api endpoints. They have since been fixed and the program is now fully functional along with our API end points. During our presentation we had some issues with the the functionality regarding the entity classes and have now been fixed.