Fitness Management System

Project Step 5 Draft Version

Prepared For:

Danielle Safonte and Michael Curry

CS340

Oregon State ECampus

Fall 2022

Contributors : Jarrod Saechao / James Lee

Executive Summary:

Our project is based on a fitness management system. The main focus of the fitness management system was to help organize and store information about training sessions in a standard gym. After analysis, we decided that clients, trainers, sales, payments and sessions were the most important factors to track. This was done by creating hypothetical scenarios where information about those entities would be needed to keep the session running smoothly. For example, a trainer can not be at more than one session at a given time, so we would need to keep track of when the trainer is at a session so they do not get double booked. At this point we started drawing out entity relationships diagrams and schemas. This helped map out how each entity interacts or is influenced by one another (Their relationships). Creating these visualizations transitioned into creating SQL commands to define the data we needed to track. During this step, we needed to back track to restructure our tables and relationships for simplification. Once this step was completed , we started creating data manipulation SQL commands for each entity and testing them on the backend. We had our layout and were ready to create the application.

Our application followed the node starter app tutorial provided from the course module. From the tutorial we were able to connect the application to the database, render and display dynamic data and incorporate most CRUD functions for each entity. When an issue occurred we would backtrack to previous steps of the tutorial, find the inconsistencies or bugs then retest the step. Time constraints were an issue. We did not have enough time to finish the create and edit functions for the payments entity due to the use of pre-filling forms based on selections. Aside from that the application is fully functional and running on the OSU server.

Overview

The fitness management system will be used for a gym with personal trainers. There are 20 personal trainers working at the facility and over 300 clients. There are 10 sessions available each day for clients to sign up for. Each session has one instructor. Clients can attend multiple sessions and each session can have multiple clients. Clients can optionally create a membership. Sales are recorded for each client attending a session and the charge is based on the trainer's pay rate. Clients have the flexibility to pay using multiple payment methods to cover the cost of a session.

Outcome

The web application will help organize sessions and entities involved in those sessions. Trainers will no longer attend or miss specific sessions and will not be double booked causing time conflicts. Clients will be able to pay online by storing card information and know who will be teaching the session, how many people will be attending the session and what day/time their session is. Recording sales and invoices can help with financial management such as keeping track of who has paid, what cards need to be updated and track financial performance.

Database Outline:

- **Clients: Records information about the client**
    - **Attributes**
        - clientID: int, auto_increment, unique, not NULL
        - firstName: varchar, not NULL
        - lastName: varchar, not NULL
        - clientEmail: varchar
        - clientPhone: bigint, not NULL
        - memberCard: int, FK references Memberships(memberID)
        - sessionDetailsSessionID: int, FK references Sessions(sessionID)
    - **Relationships**

        - **M:M** between **Clients and Sessions** with SessionDetails as an intersection table. Clients can attend multiple sessions and each session can have more than one client.
            - **Clients and Sessions** intersection table is **SessionDetails.**
        - **1:M** between **Clients and Sales**. As clients pay for each session, there will be many sales associated with one client.
        - **1:M** between **Clients and Payments**. Each payment is associated with one client.
        - **1:1** between **Clients and Memberships**. Nullable relationship. Members can join sessions without a membership.
- **Trainers: Records information about the trainer**
    - **Attributes**
        - trainerID: int, auto_increment, unique, unsigned, not NULL, PK
        - firstName: varchar, not NULL
        - lastName: varchar, not NULL
        - trainerEmail: varchar, not NULL
        - trainerPhone: bigint, not NULL
    - **Relationships**
        - **1:M** between **Trainers and Sessions**. Each trainer will instruct in multiple sessions while working at the fitness center.
        - **1:M** between **Trainers and Sales**. Each sale needs to be associated with the trainer teaching that session so pay can be directed to that trainer.
- **WorkoutSessions: Records date, trainer, and clients attending a session**
    - **Attributes**
        - sessionID: int, auto_increment, unique, unsigned, not NULL, PK
        - sessionDate: date, not NULL, PK
        - sessionDetailsClientID: int, unsigned, not NULL, FK references Clients(clientID)
            - **Note**: this attribute is part of ***SessionDetails*** intersection table
            - ON DELETE SET NULL
        - workoutSessionsTrainerID: int, unique, FK references Trainers(trainerID)

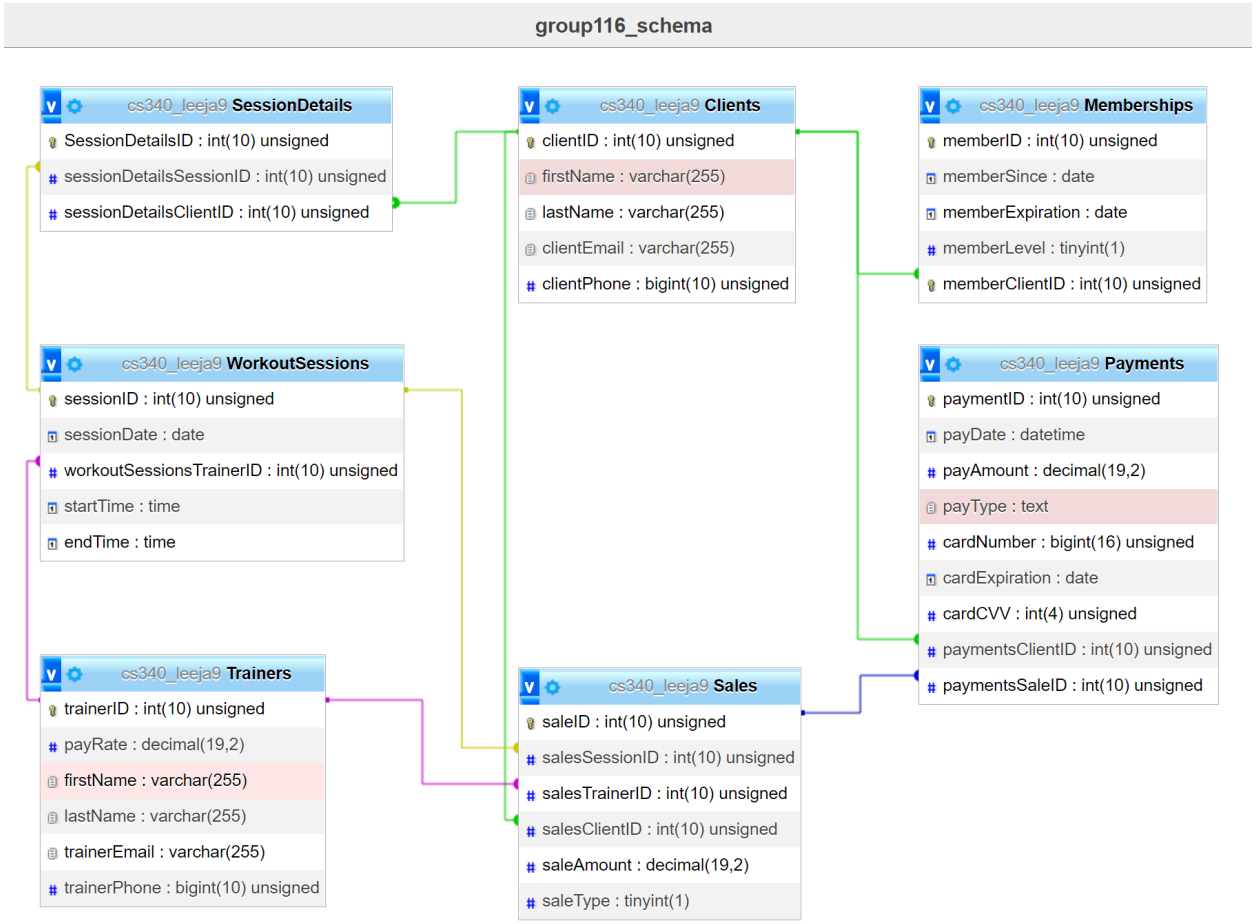
            - ON DELETE CASCADE
        - startTime: time, not NULL
        - endTime, not NULL
    - **Relationships**
        - **M:M** between **WorkoutSessions and Clients** as described in the Clients section.
            - **WorkoutSessions and Clients** intersection table is **sessionDetails.**
        - **M:1** between **Sessions and Trainers** as described in the Trainers section.
        - **1:M** between **Sessions and Sales**. For each session, there can be multiple clients attending, which means there can be multiple sales per session.
- **Payments: Stores payment transaction information**
    - **Attributes**
        - paymentID: int, auto_increment, unique, unsigned, not NULL, PK
        - payDate: datetime, not NULL
        - payAmount: double, not NULL
        - cardNumber: bigint, unsigned
        - cardExpiration: date
        - cardCVV: int, unsigned
        - paymentsClientID: int, unsigned, FK references Clients(clientID)
            - ON DELETE SET NULL
        - paymentsSaleID: int, unsigned, FK references Sales(saleID)
            - ON DELETE SET NULL
    - **Relationships**
        - **M:1** between **Payments and Sales**. There can be multiple payments for a single sale if a client makes partial payments.
        - **M:1** between **Payments and Clients**. Many payments can be associated with a single client.
- **Sales: Records sale for each client attending a session**
    - **Attributes**
        - saleID: int, auto_increment, unique, unsigned, not NULL, PK
        - salesSessionID: int, unsigned, FK references workoutSessions(sessionID)

        - ON DELETE RESTRICT
        - **Note**: we want to prevent deleting sessions that have sales associated with them.
    - salesTrainerID: int, unsigned, FK references Trainers(trainerID)
        - ON DELETE SET NULL
    - salesClientID: int, unsigned, FK references Clients(clientID)
        - ON DELETE SET NULL
    - saleAmount: decimal(19,2)
    - saleType: tinyint, default 0
        - 0 = workout session
        - 1 = merchandise
        - 3 = refund
- **Relationships**
    - **M:1** between **Sales and Clients** as described in the Clients section.
    - **M:1** between **Sales and Trainers** as described in the Trainers section.
    - **M:1** between **Sales and Sessions** as described in the Sessions section.
    - **1:M** between **Sales and Payments** as described in the Payments section.

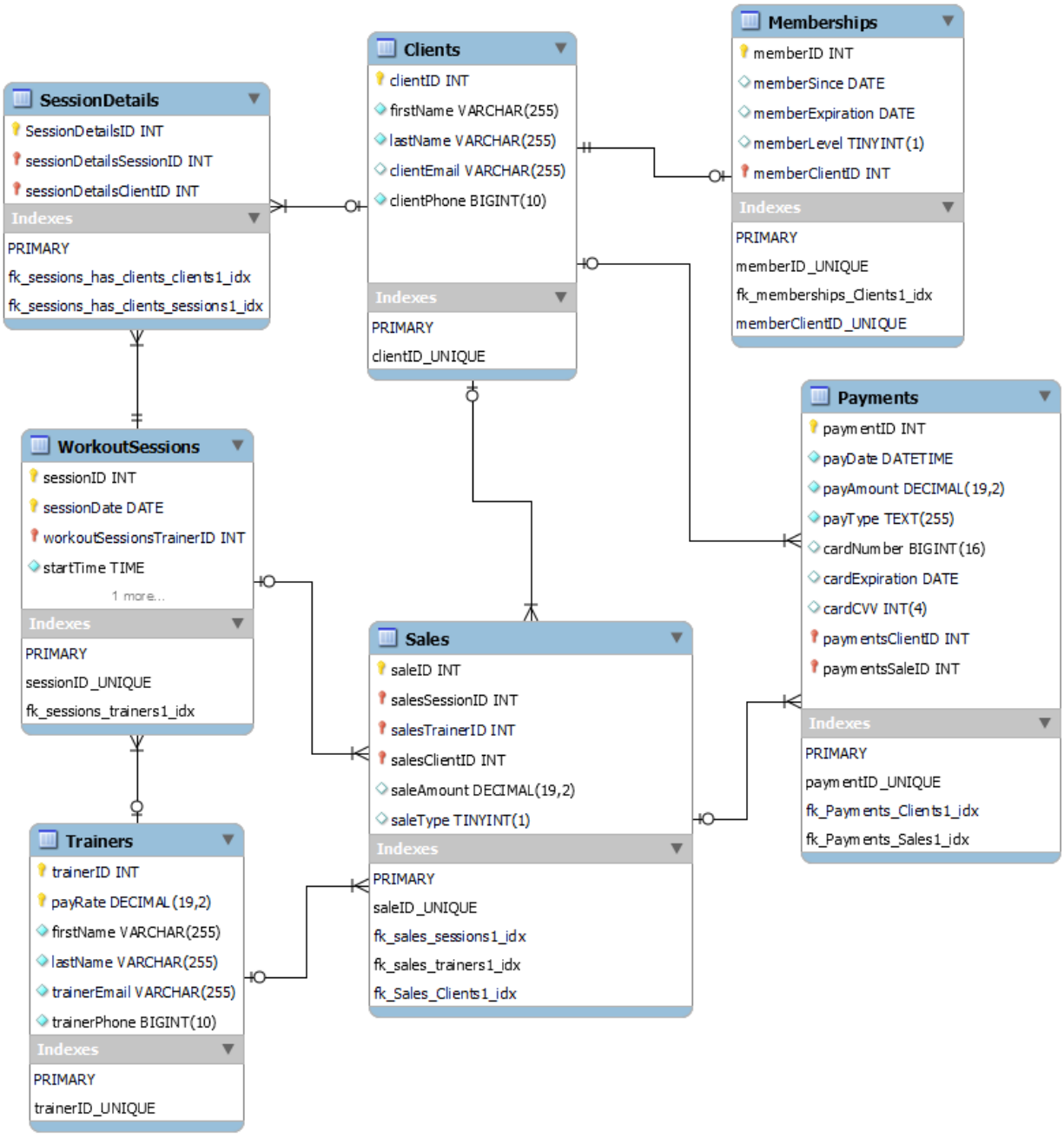
Memberships: Keeps track of who has active memberships

- **Attributes**
  - memberID: int, auto increment, NOT NULL
  - memberSince: date, NOT NULL
  - memberExpiration: date, NOT NULL
  - memberClientID: int, unique, NOT NULL FK references Clients(clientID)
    - ON DELETE CASCADE
- **Relationships**
  - **1:1** between **Clients and Memberships**. Membership is optional for clients.

Schema:



Entity-Relationship Diagram:



Sample Data:

Clients Table

clientID	firstName	lastName	clientEmail	clientPhone
1	John	Smith	johnsmith@fake.com	2221110001
2	Jane	Smith	janesmith@fake.com	2221110002
3	Michael	Jackson	heehee@fake.com	2221110003
4	The	Dude	myrug@fake.com	2221110004

Trainers Table

trainerID	payRate	firstName	lastName	trainerEmail	trainerPhone
1	85.50	Arnold	Schwarzenegger	illbeback@fake.com	2221110005
2	23.90	Scooby	Doo	ruffworkouts@fake.com	2221110006
3	75.50	Thor	Odinson	justbash@fake.com	2221110007
4	20.00	Shirley	Temple	meanerthanilook@fake.com	2221110008

WorkoutSessions Table

sessionID	sessionDate	workoutSessionsTrainerID	startTime	endTime
1	2022-10-20	2	15:00:00	15:55:00
2	2022-10-20	2	10:00:00	10:55:00
3	2022-10-20	4	11:00:00	11:55:00
4	2022-10-20	1	12:00:00	12:55:00

SessionDetails Table

SessionDetailsID	sessionDetailsSessionID	sessionDetailsClientID
1	1	1
2	1	3
3	2	2

Sales Table

saleID	salesSessionID	salesTrainerID	salesClientID	saleAmount	saleType
1	1	2	1	23.90	0
2	1	2	3	23.90	0
3	2	2	2	23.90	0

Payments Table

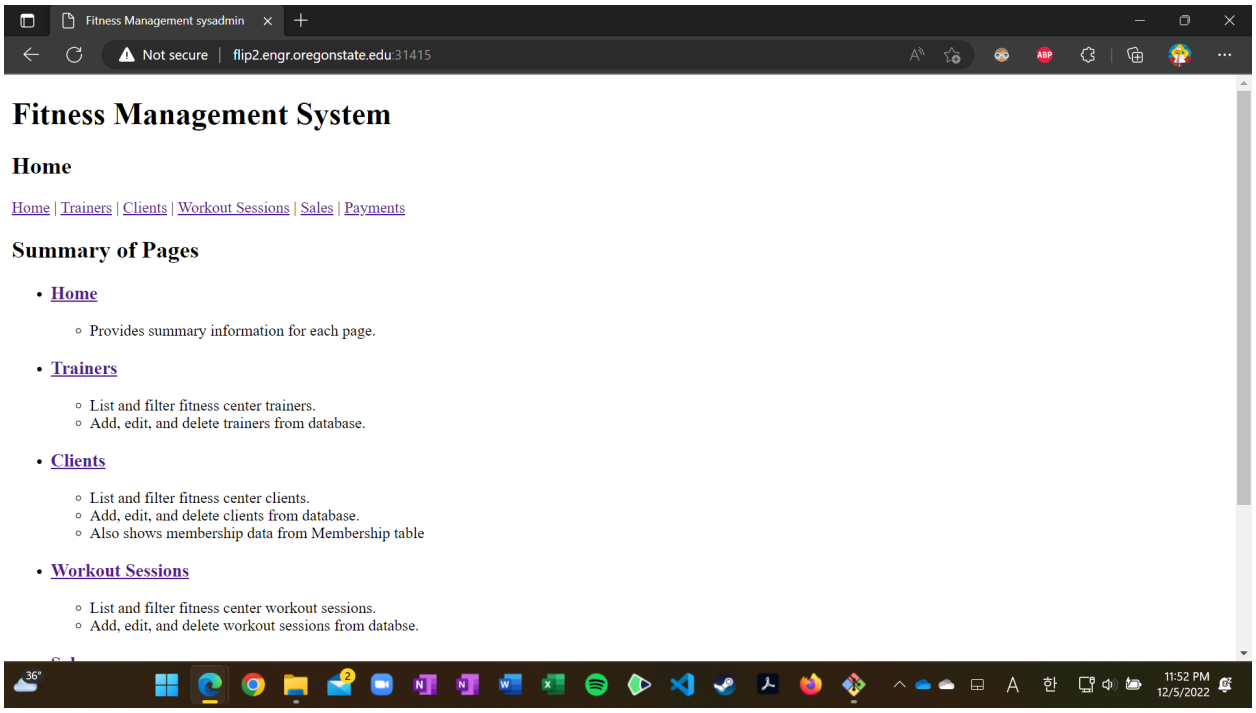
paymentID	payDate	payAmount	payType	cardNumber	cardExpiration	cardCVV	paymentsClientID	paymentsSaleID
1	2022-10-20 00:00:01	23.90	CASH	NULL	NULL	NULL	1	1
2	2022-10-20 00:00:02	23.90	VISA	1111222233334444	2025-01-01	123	3	2
3	2022-10-20 00:00:03	23.90	MASTERCARD	1111222233334444	2025-01-01	123	2	3

Memberships Table

memberID	memberSince	memberExpiration	memberLevel	memberClientID
1	2022-01-01	2023-01-01	0	1
2	2022-01-01	2023-01-01	1	2
3	2022-01-01	2023-01-01	2	3

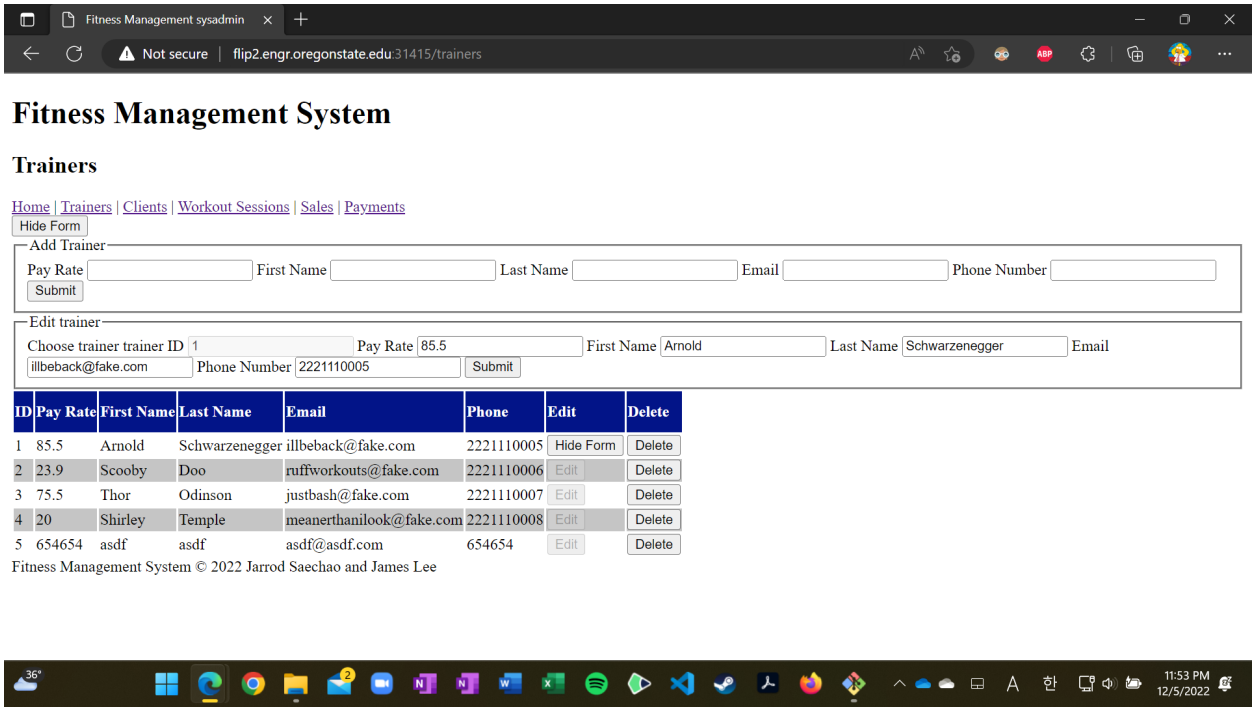
Screenshots

Home Page



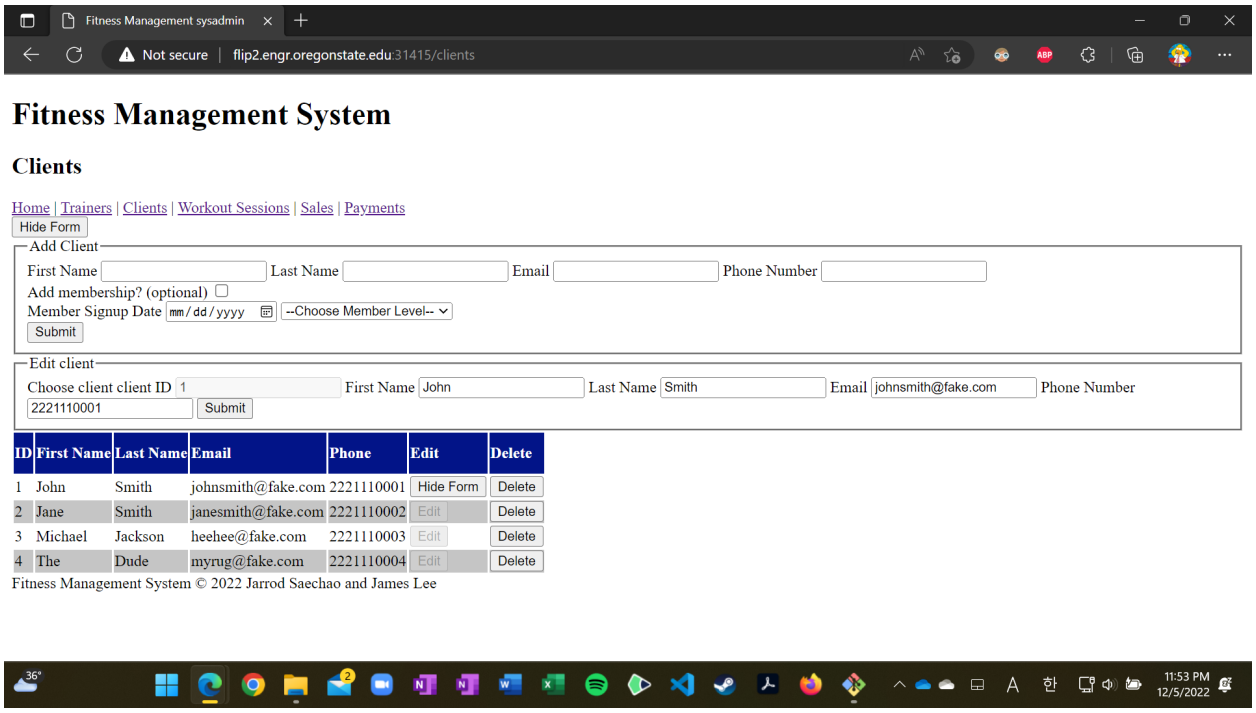
Trainers Page

Buttons toggle form elements. Each row can be edited and deleted.

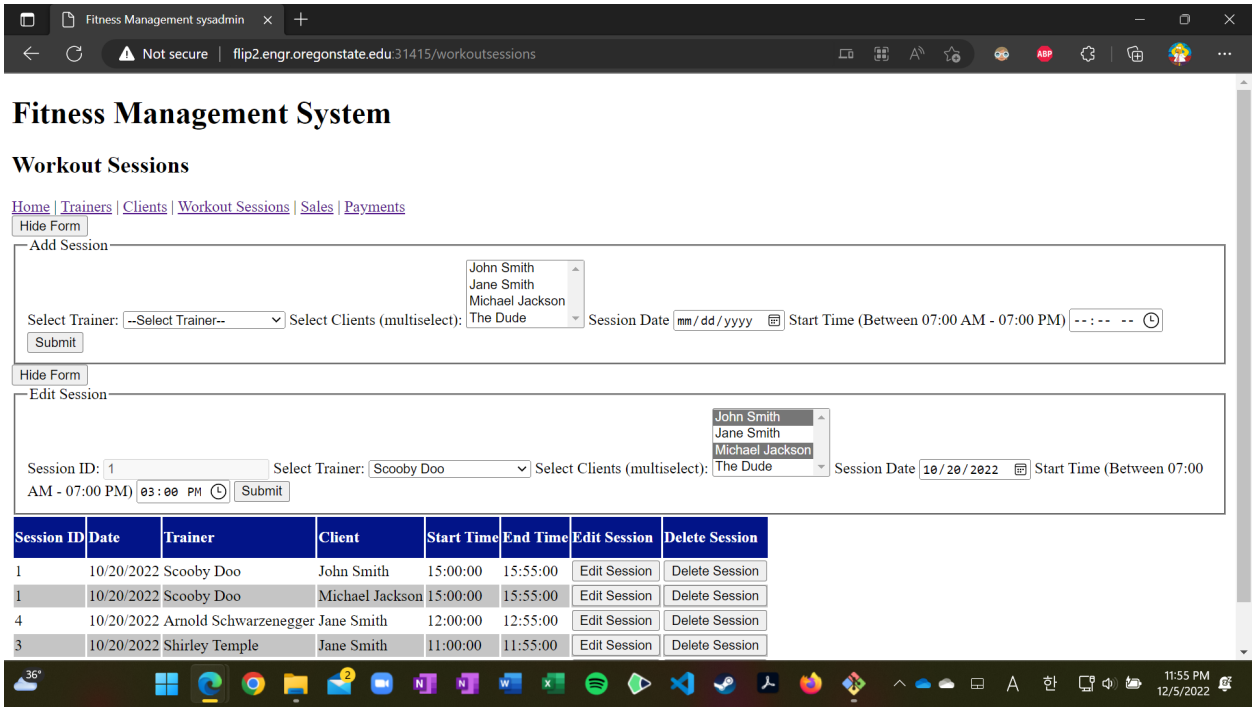


Clients Page

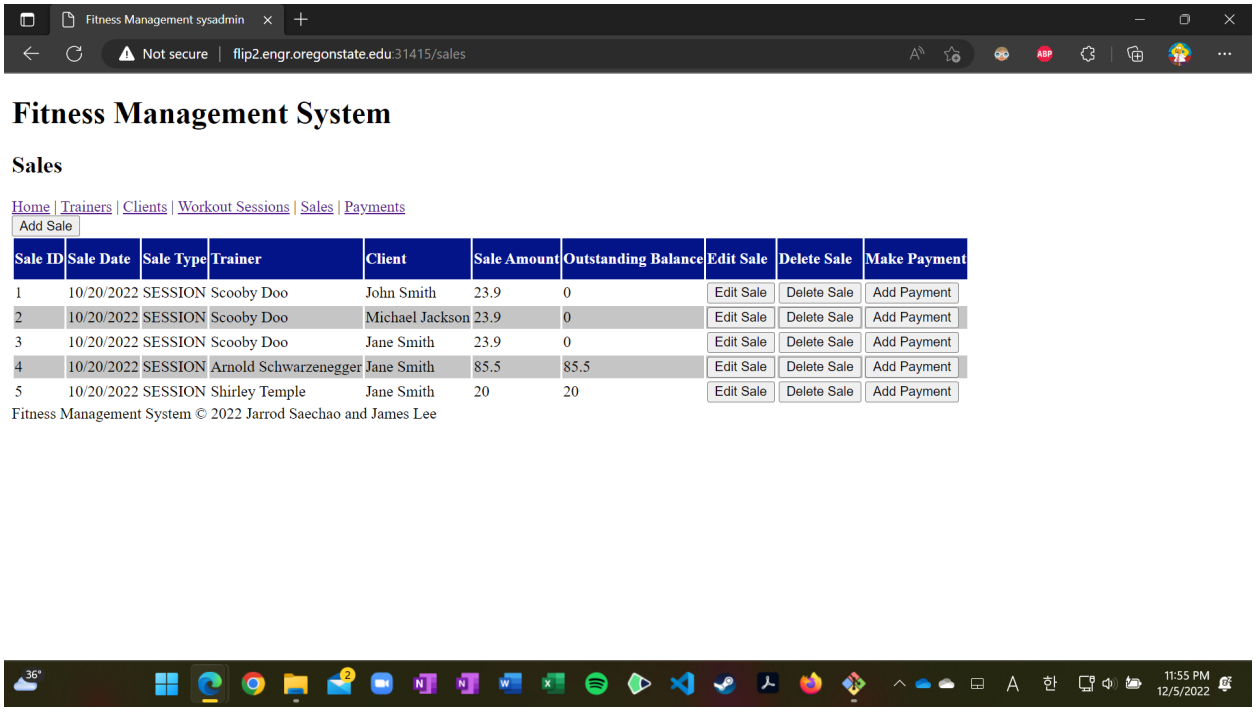
Optional membership data doesn't work. The intent was to have the check box toggle optional form elements that would implement CRUD functions for the optional Memberships table.



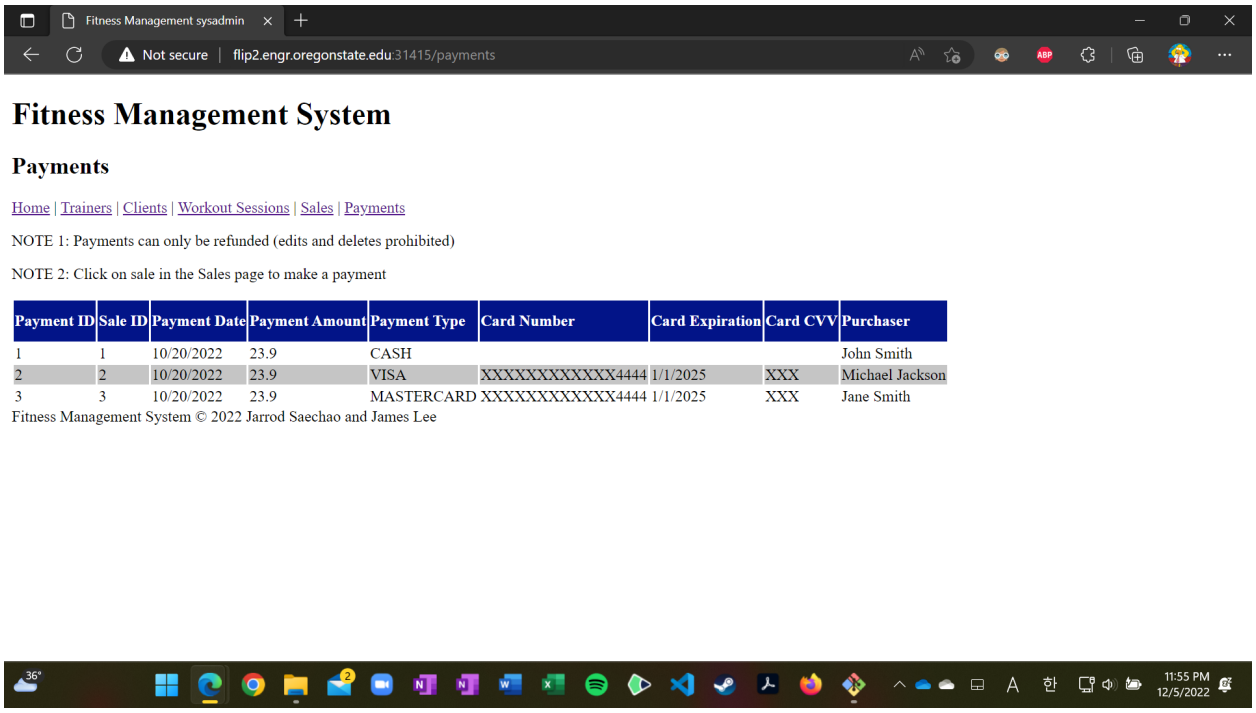
Workout Sessions



Sales



Payments



Citations:

Curry, M (Oct 2022) Nodejs-starter-app, Frame structure of application
https://github.com/osu-cs340-ecampus/nodejs-starter-app

Mozilla (Oct 2022) HTML Web API Javascript, Tutorials for implementation
https://developer.mozilla.org/

MIT (Oct 2022) Handlebars guide, Implementation
https://handlebarsjs.com/guide/