3MillionDance Academy

Oanh Le, Dan Nguyen, Augusta Major

University of South Florida

COP 4710 - 001

July 2021

I. Description

This project demonstrates an enterprise full-stack information system that assists customers with registration for a wide range of dance classes from 3MillionDance Academy by utilizing technologies such as API, PostgreSQL, NodeJS, and ReactJS to create a seamless and efficient environment which enhances user navigability. Users are able to view instructors and class schedules to proceed with registering for courses.

II. Schema and Database Initialization Queries

1. **Customers**(<u>CustID</u>, Fname, Lname, DOB, Phone, Address, Email, Password)

   Customers records information about customers, including their login information which they use to reserve classes. Upon account creation/sign up, users are assigned a customer ID. Their role as a customer determines some of their viewing and editing privileges.

```
CREATE TABLE Customers (
      CustID CHAR(9) PRIMARY KEY,
      Fname VARCHAR(20),
      Lname VARCHAR(20),
      DOB DATE,
      Address VARCHAR(200),
      Email VARCHAR(50),
      Password VARCHAR(50),
      Phone VARCHAR(20)
);
```

2. **Instructors**(<u>InstructorID</u>, Fname, Lname, Email, Password, DOB, Phone)

   Instructors records information about instructors, including their contact information for people. Upon account creation/sign up, users are assigned an instructor ID. Their role as an instructor determines some of their viewing and editing privileges.

```
CREATE TABLE Instructors (
      InstructorID VARCHAR(20) PRIMARY KEY,
      Fname VARCHAR(20),
      Lname VARCHAR(20),
      Email VARCHAR(50),
      Password VARCHAR(50),
      Phone VARCHAR(20),
      DOB DATE,
);
```

3. **Rooms**(<u>RoomID</u>, Floor, Capacity)

   Rooms records individual dance rooms, the floor they are located on, and the room capacity.

```
CREATE TABLE Rooms (
        RoomID CHAR(3) PRIMARY KEY,
        Floor INTEGER,
        Capacity INTEGER
);
```

Classes(ClassID, Name, Type, Level, Genre, Availability, DateTime, Duration, Price, Individual, Capacity, RoomID)

- FK: RoomID references Rooms

Classes records details relevant to what type of course it is (in-person or online), who teaches it, the room in which the class is held, methods of instruction, and availability with each entry in the table representing a single class.

```
CREATE TABLE Classes (
        ClassID CHAR(4) PRIMARY KEY,
        Name TEXT,
        Type VARCHAR(10) check(Type in ('virtual', 'in-person')),
        Level VARCHAR(15) check(Level in ('beginner',
        'intermediate', 'master')),
        Genre TEXT,
        Availability VARCHAR(5) check(Availability in ('TRUE',
        'FALSE')),
        DateTime TIMESTAMP(2),
        Duration INTEGER,
        Price REAL,
        Individual VARCHAR(8) check(Individual in ('Public',
        'Private')),
        Capacity INTEGER,
        Held CHAR(3) NOT NULL REFERENCES Rooms
);
```

4. **Teach**(PK: (ClassID, InstructorID), Evaluation)
    - FK: ClassID references Classes, FK: InstructorID references Instructors

    Teach describes the relationship between *Classes* and *Instructors* and includes information on instruction evaluations.

```
CREATE TABLE Teach (
        ClassID CHAR(4) REFERENCES Classes,
        InstructorID CHAR(20) REFERENCES Instructors,
        Evaluation TEXT,
        CONSTRAINT teach_pk PRIMARY KEY (ClassID, InstructorID)
);
```

5. **Orders**(OrderID), CustID, OrderedDate, PaymentType, Status)
    - FK: CustID references Customers

Orders records orders that *Customers* made. One entry is stored for every order that the user places. *Orders* can only be placed if the user logins with an account as order entries require a customer ID. It also includes data about each individual class's booking details (where Status refers to if the reservation was cancelled).

```
CREATE TABLE Orders (
      OrderID VARCHAR(10) PRIMARY KEY,
      CustID CHAR(9) REFERENCES Customers,
      OrderedDate TIMESTAMP(2),
      PaymentType VARCHAR(10),
      Status VARCHAR(20)
);
```

6. **Place_Order** (PK: OrderID, CustID), OrderedDate)
   - FK: OrderID references Order, CustID references Customers

   Place_Order describes the relationship between *Order* and *Customers* with a date and timestamp of when the order was placed.

```
CREATE TABLE Place_Order (
      OrderID VARCHAR(10) REFERENCES Orders,
      CustID VARCHAR(9) REFERENCES Customers,
      OrderedDate TIMESTAMP(2),
      CONSTRAINT placeorder_pk PRIMARY KEY (OrderID, CustID)
);
```

7. **Order_List** (PK: (OrderID, ClassID), Quantity)
   - FK: OrderID references Order, ClassID references Classes

   Order_List describes the relationship between *Order* and *Classes* with the quantity of each class for each transaction in the *Order* table.

```
CREATE TABLE Order_List (
      OrderID VARCHAR(10) REFERENCES Orders,
      ClassID CHAR(4) REFERENCES Classes,
      Quantity INTEGER,
      CONSTRAINT orderlist_pk PRIMARY KEY (OrderID, ClassID)
);
```

8. **Take**(PK: (ClassID, CustID), Attendance)
   - FK: ClassID references Classes, FK: CustID references Customers

   Take describes the relationship between *Classes* and *Customers* and records customer class attendance (whether they showed up).

```
CREATE TABLE Take (
      ClassID VARCHAR(4) REFERENCES Classes,
      CustID VARCHAR(9) REFERENCES Customers,
      Attendance BOOLEAN,
                CONSTRAINT take_pk PRIMARY KEY (ClassID, CustID)
        );
```

II. EER Diagram

The following EER diagram depicts the conceptual design of our website's database where **Customers**, **Instructors**, **Rooms**, and **Classes** are entities with attributes and their own primary keys while **Teach**, **Orders**, **Place_Order**, **Order_List**, and **Take** are the relationships between the various entities.

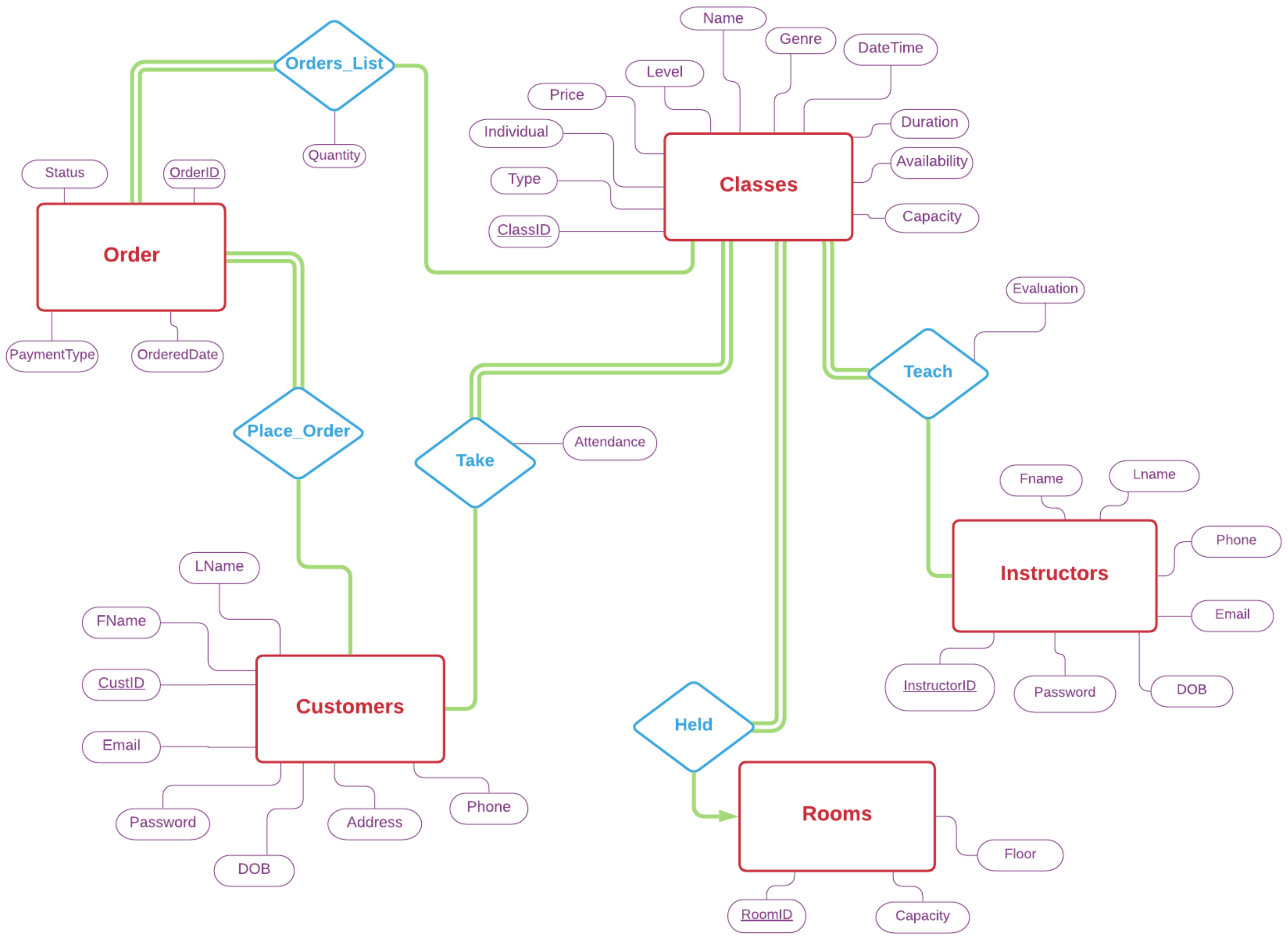


Figure 1. EER Diagram

III. SQL Queries

1. **Show profile of a specific user**
   ```
   SELECT * FROM Customers WHERE CustID = 'value';
   ```
2. **Login**
   ```
   SELECT * FROM Customers WHERE email = 'value' AND password =
   'value';
   ```

```
SELECT * FROM Instructors WHERE email = 'value' AND password =
'value';
```

3. **Create new user and add them to database**
```
INSERT INTO Customers (CustID, Fname, Lname, DOB, Phone, Address,
Email, Password) VALUES (value1, value2, …) RETURNING *;
```
4. **Booking**
```
INSERT INTO Orders (OrderID, CustID, OrderedDate, PaymentType,
Status) VALUES (value1, value2, …) RETURNING *;
INSERT INTO Place_Order (OrderID, CustID) VALUES (value1, value2)
RETURNING *;
INSERT INTO Order_List (OrderID, ClassID, Quantity) VALUES (value1,
value2, …) RETURNING *;
INSERT INTO Take (ClassID, Custid) VALUES (value1, value2) RETURNING
*;
```
5. **Modify database when user update their profile**
```
UPDATE Customers SET column1 = value1, column2 = value2, ... WHERE
CustID = 'value';
UPDATE Instructors SET column1 = value1, column2 = value2, ... WHERE
InstructorID = 'value';
```
6. **Show all the classes a customer has booked for, order by recently**
```
SELECT * FROM Orders O, Order_List L WHERE CustID = 'value' and
O.OrderID = L.OrderID ORDER BY OrderedDate DESC;
```
7. **Show detail of a specific class that a customer has booked for (info, price, instructors, rooms, ...)**
```
SELECT classes.classid, classes.name, classes.type, classes.level,
classes.genre, classes.datetime, classes.duration, classes.price,
classes.individual, R.roomid, R.floor, take.attendance, I.fname,
I.lname, I.email
FROM Classes natural join Take natural join teach AS T, Rooms R,
Instructors I
WHERE classes.held = R.roomid and T.instructorid = I.instructorid
and take.custid = 'value'AND classes.classid = 'value';
```
8. **Return all classes teached by a specific instructor, count how many students booked for each class**
```
SELECT T.classid, MAX(C.name),MAX(C.type),MAX(C.level),
MAX(C.genre), MAX(C.availability), MAX(C.datetime), MAX(C.duration),
MAX(C.individual), MAX(R.roomID), MAX(R.floor), Max(R.capacity),
COUNT(T.custid)
FROM Teach natural join Classes as C, Rooms R, Take T
WHERE instructorid ='value' and C.held = R.roomid and T.classid =
C.classid GROUP BY T.classid;
```
9. **Return all students information who took or will take a specific class**
```
SELECT * FROM Teach NATURAL JOIN Take AS T, Customers C WHERE
classid = 'value' AND instructorid ='oliver0129' AND C.custid =
T.custid;
```
10. **Show all available instructors.**

```
SELECT Fname, Lname FROM Instructors;
```

11. **Show all the upcoming classes and its availability.**
```
SELECT max(classes.capacity) - count(take.classid) AS availability
FROM Classes natural join Take
WHERE datetime > now() group by classid, name, type, level, genre,
datetime, duration, price, individual;
```
12. **Reschedule classes.**
Example: Reschedule classes between July 26 and 30 to one week later:
```
UPDATE Classes SET DateTime = DateTime + 7 WHERE DateTime BETWEEN
'07-26-2021' AND '07-30-2021';
```

IV. Client Functionality

How our user interface will work and how customers will view and be able to interact with our website is partially based on if they are (1) an anonymous user or (2) a logged-in user. Most functions and features are kept available even to anonymous users to maintain information accessibility to a wider range of people and to increase user-website interaction. The following overview describes the two types of database users:

1. Anonymous Users
Users who initially connect with and navigate through our website maintain their anonymous status until they create an account and log in. They are able to view most website pages - such as the homepage, Instructors page, and Schedule page - but they will not have a personal profile to view their own personal details, such as their contact information, order history, or teaching schedule. They will also be prevented from booking classes or editing class details unless they create an account as they will be prompted to log in or sign up.

2. Logged-in Users
Logged-in users retain all the viewing privileges and functionality of anonymous users, but they are also able to have a personal profile with previously inputted login information. Their profile holds data they have inputted for login and classes they have booked; account creation is primarily for booking and organization on the system's behalf. All users with accounts have unique IDs attached to their account in our system's database which was assigned upon account creation. Logged-in users are divided into two categories: customers and instructors. While customers have customer IDs (CustID), instructors have instructor IDs (InstructorID), and users are prompted to enter their role (customer or instructor) on the login page as per this distinction. Both customers and instructors will have the options to view the Instructors, Schedule, and profile pages and to log out, but customers will be able to view their order history while instructors will be able to view their personal class schedule of the courses they teach.

In particular, anonymous users are able to view the following pages:

1. Homepage
2. Instructors page
3. Schedule page
4. Login page

5. Signup page

   Logged-in users are also able to view the following pages:
6. Booking page
7. My Profile page
8. (1) My Orders
   (2) My Schedule

Regardless, at the top of every page is a menu bar which allows all users easy access to certain pages in the system. These links lead to the Instructors page and Schedule page. If the user is not signed in, then every page's menu bar will also have links leading to the Login page and the Signup page. If the user is signed in, then they will have options to view their profile or to log out instead of to login or signup.

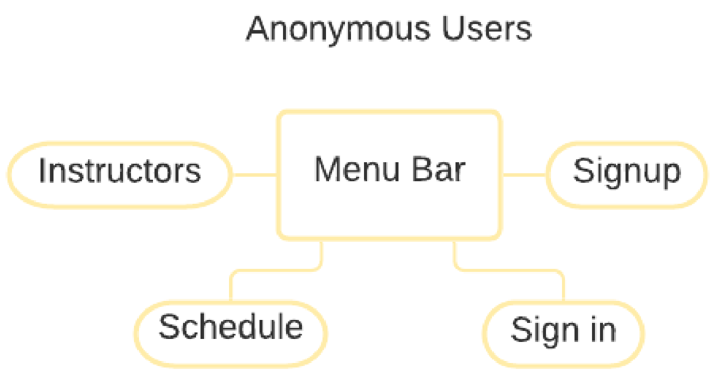


Figure 2. View for Anonymous users

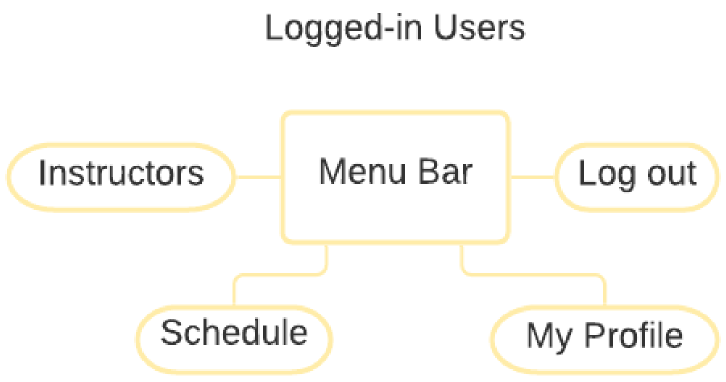


Figure 3. View for logged-in users

1. **Anonymous Users**
   I. Homepage

All users are greeted by the home page when first connecting to the website, which features the menu bar at the top as well as additional navigation links to them as the user scrolls down and through the page. Our home page simply gives customers a glimpse at what 3MillionDance offers as a business (such as various dance programs), how the rest of the website can help deliver the service (via information about our instructors, dance studio, and company beliefs), and where we can be reached (through social media, in-person bookings, etc.).

II. Instructors page

Users are able to "meet" all of 3MillionDance Academy's instructors by viewing their virtual profiles on this page, which presents the instructors' names and photos.

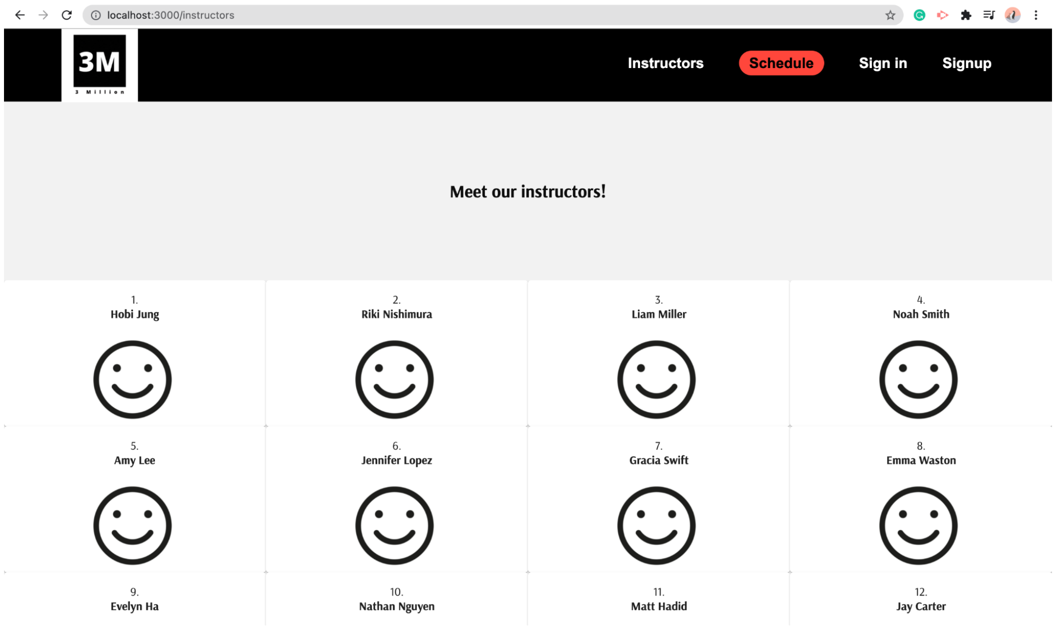
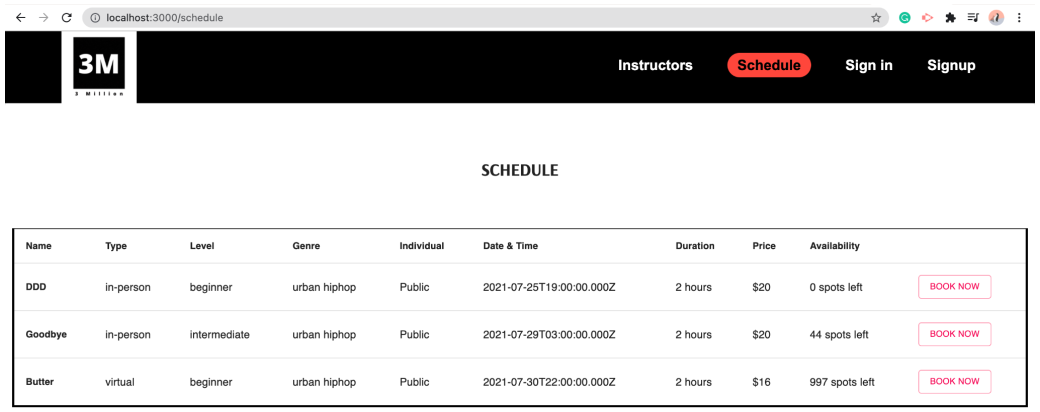


Figure 4. Instructors page

III. Schedule page

The schedule page allows users to view a schedule filled with all of the future classes of all instructors. Users can view the dance class song names, type, level, genre, public or private status, date and time, duration, price, and the number of available spots. They will also be given an option to book for each class.



SCHEDULE

Name	Type	Level	Genre	Individual	Date & Time	Duration	Price	Availability	
DDD	in-person	beginner	urban hiphop	Public	2021-07-25T19:00:00.000Z	2 hours	$20	0 spots left	BOOK NOW
Goodbye	in-person	intermediate	urban hiphop	Public	2021-07-29T03:00:00.000Z	2 hours	$20	44 spots left	BOOK NOW
Butter	virtual	beginner	urban hiphop	Public	2021-07-30T22:00:00.000Z	2 hours	$16	997 spots left	BOOK NOW

Figure 4. Schedule page

IV. Login page
The Login page, upon successful user login using their previously created account credentials (email, password, and role - customer or instructor), turns user status from anonymous to logged-in. They will also be informed that their login form has been submitted. Otherwise, users will be notified that there was an error and that they need to try again. After logging in, users will also have an option to log out or to view their profile at the top right of their menu bar after finishing their session. These options replace the login and signup options which were previously there. On the backend, a successful login message and verification of the credentials can be seen.

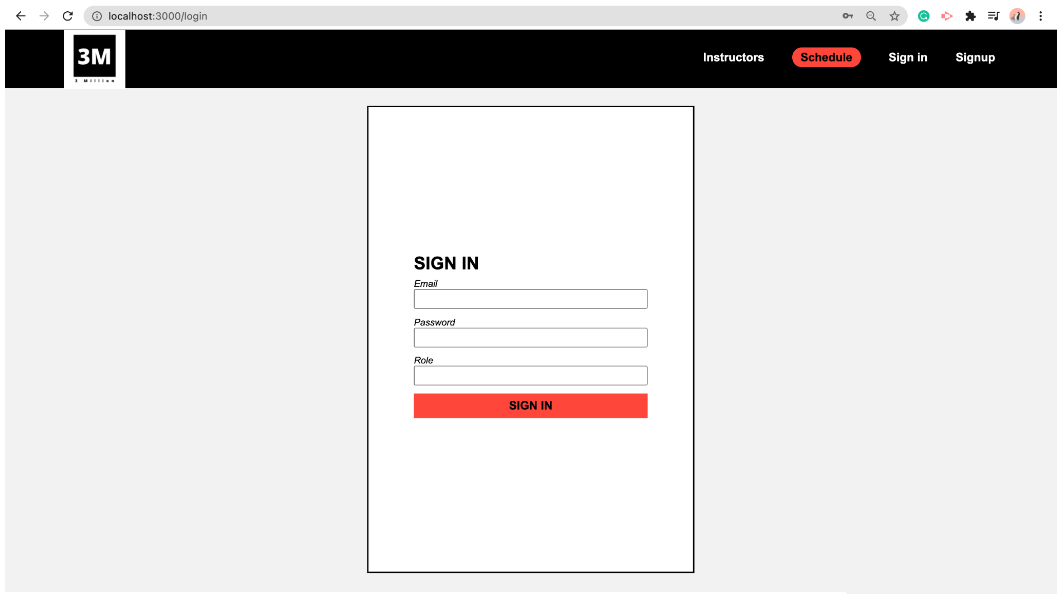


Figure 5. Sign in page

V. Signup page
The Signup page allows users to create an account on our website which holds their personal information.

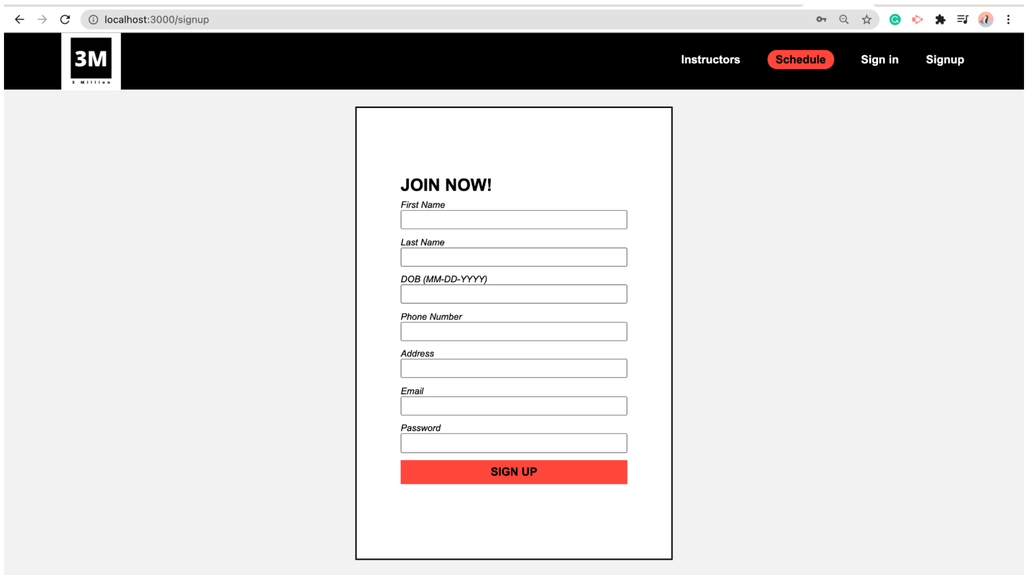


Figure 6. Sign up page

Users can sign up by inputting their first name, last name, date of birth, email, and address; they will also be required to create a password for their account. Within the system, users are assigned with unique IDs (customer IDs and instructor IDs) and their inputted information is added to the system database. Users are asked to reenter information if any of the fields are missing or invalid (for example, if their password length is less than five characters, then users are notified that their password is too short). Users are notified of successful account creation as they are directed to continue the registration process through confirming their email address via a verification link. After signing up, they may log in and proceed as a logged-in user.

2. Logged-in Users

I. Booking page

The booking page prompts users for the desired class, quantity, and payment type. Upon submitting the form, they will be registered for the class; this means that the class's available spots will update with the appropriate number of decrements (according to quantity) and their submission will be added to Orders within the database as well as the user's order history.

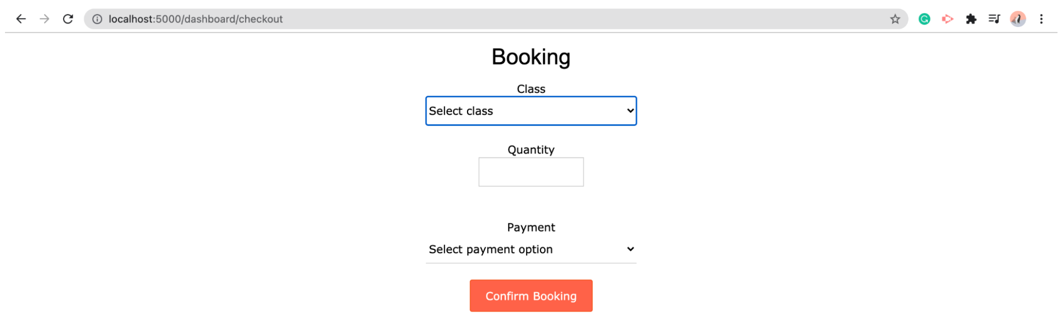
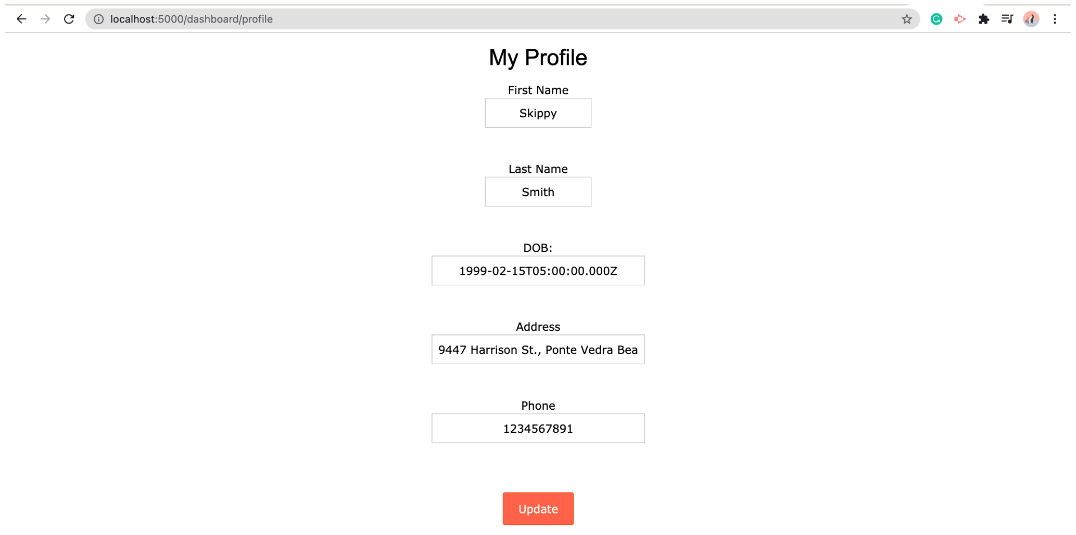


Figure 7. Booking page

II. My Profile page

Logged-in users will have a personal profile which shows their details, such as their first and last name, address, phone number, and date of birth. Users are also able to update their information by simply changing the data within the fields and clicking "update."


localhost:5000/dashboard/profile

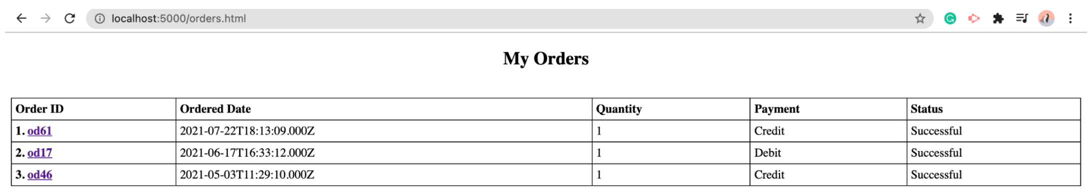
My Profile

First Name	Skippy
Last Name	Smith
DOB:	1999-02-15T05:00:00.000Z
Address	9447 Harrison St., Ponte Vedra Bea
Phone	1234567891

Figure 8. My profile page

III. (1) My Orders

If the user's role (as indicated by their signup/login credentials) is that of a customer, the user will be able to view their order history of both inactive (classes which have ended) and active courses.

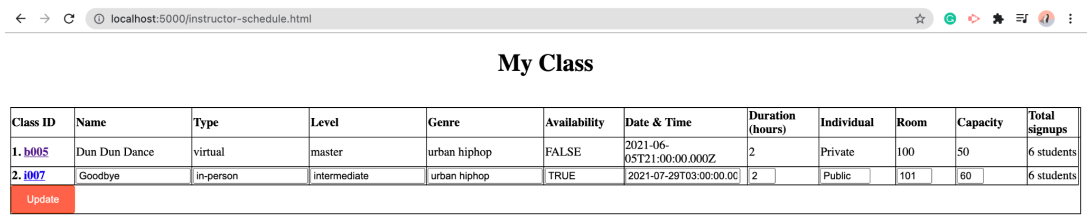

localhost:5000/orders.html

My Orders

Order ID	Ordered Date	Quantity	Payment	Status
1. od61	2021-07-22T18:13:09.000Z	1	Credit	Successful
2. od17	2021-06-17T16:33:12.000Z	1	Debit	Successful
3. od46	2021-05-03T11:29:10.000Z	1	Credit	Successful

Figure 9. My orders page

(2) My Schedule
If the user's role (as indicated by their signup/login credentials) is that of an instructor, the user will be able to view classes that they have previously taught and classes that they will be teaching in the future. For future classes, they are able to update nearly all of the fields as desired, such as the name, type, level, genre, availability, duration, and capacity. Some fields, such as the class ID, cannot be altered because these are values automatically assigned or calculated by the database system.



My Class

Class ID	Name	Type	Level	Genre	Availability	Date & Time	Duration (hours)	Individual	Room	Capacity	Total signups
1. b005	Dun Dun Dance	virtual	master	urban hiphop	FALSE	2021-06-05T21:00:00.000Z	2	Private	100	50	6 students
2. i007	Goodbye	in-person	intermediate	urban hiphop	TRUE	2021-07-29T03:00:00.00	2	Public	101	60	6 students
Update											

Figure 10. My schedule page

V. Links to Assignments

1. Milestone 1
   https://docs.google.com/document/d/1kAjofyItIIUUMa5bXw4FTV7ZY3hq9JjhvYRQHEcDAWg/edit?usp=sharing
2. Link to diagram
   https://lucid.app/lucidchart/invitations/accept/inv_3fa69acf-94c2-475e-beee-59421a19772a?viewport_loc=26%2C143%2C1908%2C1101%2C0_0
3. Google sheet to database tables
   https://docs.google.com/spreadsheets/d/1k612XW-7bn-yE6OM9_jrVqSM-r-jar7NQdecrkPtbPM/edit?usp=sharing
4. Github Repository
   https://github.com/dnguyen231/dance-academy

VI. Sources

1. Connecting node-postgres (backend - database)
   https://node-postgres.com/features/connecting
2. Connecting node-react (backend - frontend)
   https://www.geeksforgeeks.org/how-to-connect-nodejs-with-reactjs/
3. Postgresql - express- react
   https://dev.to/andrewbaisden/creating-react-node-apps-that-connect-to-postgresql-and-harperdb-41h3