The Fitness Center

Group 6
CIS 3400 - EMWA

Eun Joo Choi (eunjoo.choi1@baruchmail.cuny.edu)
Jennifer Huang (jennifer.huang@baruchmail.cuny.edu)
Jason Mishkin (jason.mishkin@baruchmail.cuny.edu)
Yiping Xie (yiping.xie@baruchmail.cuny.edu)

1. **Description**

   Our company has owned and operated a fitness center in midtown Manhattan for 5 years. Recently, the fitness center has been renovated and expanded larger in space. Therefore, we decided to extend the choices of fitness programs and the membership plans. As we have been using an old and simple version of the Database system which only stores customer information, there is a strong need to upgrade our current database to be compatible with our business operation.

   In our business, members are signed up for a membership plan where they could register for either personal, group training or other class sessions such as yoga and Pilates. For each class time, there are different classes available. We would need to keep track of information such as address, contacts, and gender for the members. We also need to store the addresses, contact information and pay rate for our instructors.

   Our goal is to provide high-quality services to our customers through program expansion and by introducing a new application system, the efficiency of our fitness center will be improved.

2. **Identification of the information needs**
   - Member information: Member_ID, Member_FirstName, Member_LastName, Member_Street, Member_City, Member_State, Member_ZipCode, Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type
   - Instructor information: Instructor_ID, Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_City, Instructor_State, Instructor_ZipCode, Instructor_PhoneNumber, Rate_of_Pay
   - Membership information: Membership_ID, Plan_Description, Membership_Start_Date, Membership_End_Date, Membership_Rate, Payment_Due_Date
   - ClassSessions information: Session_ID, Session_Start_Time, Session_End_Time
   - ClassOfferings information: Class_ID, Class_Name, Class_Description

3. **Initial list of entities**
   - Members
   - Instructors
   - Membership
   - ClassSessions
   - ClassOfferings

4. **Distribution of duties**
   - Documentation writer: Eun Joo Choi, Jason Mishkin
   - Documentation editor/reviewer: Jennifer Huang, Yiping Xie
   - Systems analyst: Jason Mishkin, Jennifer Huang
   - Application developer: Yiping Xie, Eun Joo Choi

Relational Model

Instructors (Instructor_ID(key), Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_City, Instructor_State, Instructor_ZipCode, Instructor_PhoneNumber, Rate_of_Pay)

ClassOfferings (Class_ID(key), Class_Name, Class_Description)

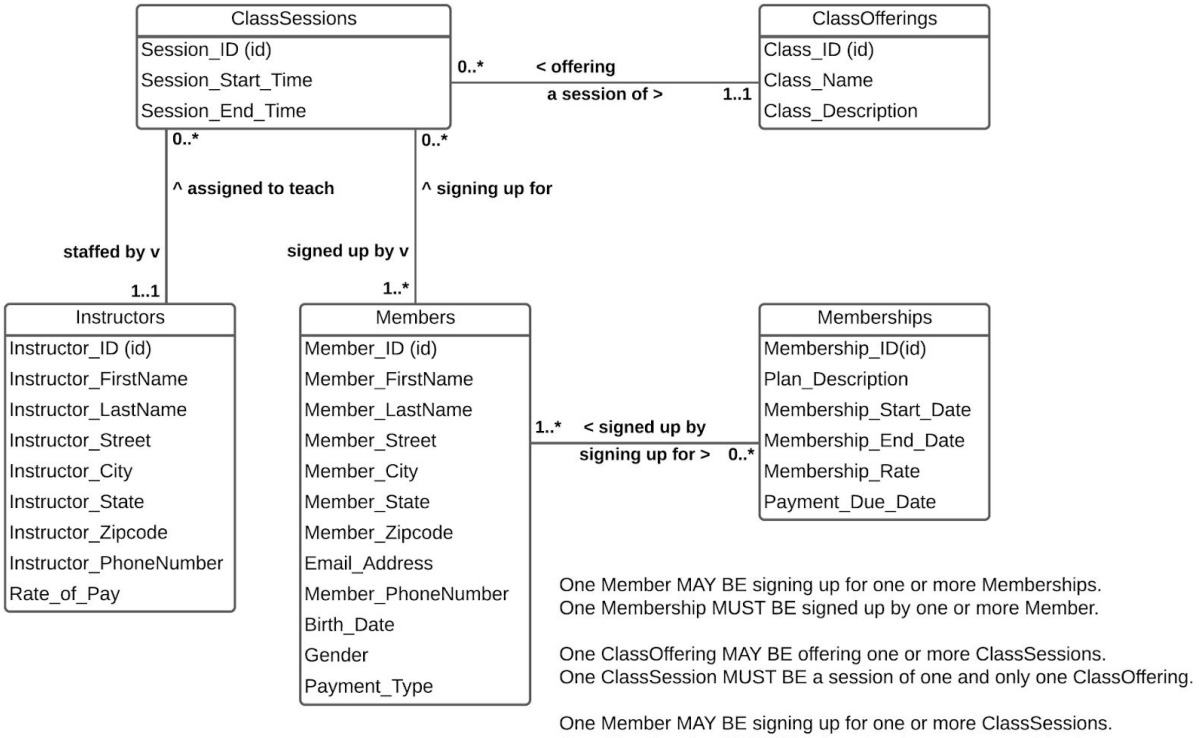
Members (Member_ID(key), Member_FirstName, Member_LastName, Member_Street, Member_City, Member_State, Member_ZipCode, Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type)

Memberships (Membership_ID(key), Plan_Description, Membership_Start_Date, Membership_End_Date, Membership_Rate,-Payment_Due_Date)

ClassSessions (Session_ID(key), Session_Start_Time, Session_End_Time, Instructor_ID(FK), Class_ID(FK))

Member_ClassSessions (Member_ID(FK)(key), Session_ID(FK)(key))

Member_Memberships (Member_ID(FK)(key), Membership_ID(FK)(key))



One Member MAY BE signing up for one or more Memberships.
One Membership MUST BE signed up by one or more Member.

One ClassOffering MAY BE offering one or more ClassSessions.
One ClassSession MUST BE a session of one and only one ClassOffering.

One Member MAY BE signing up for one or more ClassSessions.
One ClassSession MUST BE signed up by one or more Member.

One Instructor MAY BE assigned to teach one or more ClassSessions.
One ClassSession MUST BE staffed by one and only one Instructor.

Normalization

1NF: Does it match the definition of a relation? YES

Instructors (Instructor_ID(key), Instructor_FirstName, Instructor_LastName,
Instructor_Street, Instructor_City, Instructor_State, Instructor_ZipCode,
Instructor_PhoneNumber, Rate_of_Pay)
Key: Instructor_ID
FD1: Instructor_ID → Instructor_FirstName, Instructor_LastName, Instructor_Street,
Instructor_City, Instructor_State, Instructor_ZipCode, Instructor_PhoneNumber,
Rate_of_Pay
FD2: Instructor_Zipcode -> Instructor_City, Instructor_State

ClassOfferings (Class_ID(key), Class_Name, Class_Description)
Key: Class_ID
FD1: Class_ID → Class_Name, Class_Description

ClassSessions (Session_ID(key), Session_Start_Time, Session_End_Time,
Instructor_ID(FK), Class_ID(FK))
Key: Session_ID, Instructor_ID, Class_ID
FD1: Session_ID, Instructor_ID, Class_ID → Session_Start_Time, Session_End_Time

Members (Member_ID(key), Member_FirstName, Member_LastName, Member_Street,
Member_City, Member_State, Member_ZipCode, Email_Address,
Member_PhoneNumber, Birth_Date, Gender, Payment_Type)
Key: Member_ID
FD1: Member_ID → Member_FirstName, Member_LastName, Member_Street,
Member_City, Member_State, Member_ZipCode, Email_Address,
Member_PhoneNumber, Birth_Date, Gender, Payment_Type
FD2: Member_Zipcode -> Member_City, Member_State

Member_ClassSessions (Member_ID(FK)(key), Session_ID(FK)(key))
Key: Member_ID, Session_ID

Memberships (Membership_ID, Plan_Description, Membership_Start_Date,
Membership_End_Date, Membership_Rate, Payment_Due_Date)
Key: Membership_ID
FD1: Membership_ID -> Membership_Plan, Plan_Description, Membership_Start_Date,
Membership_End_Date, Membership_Rate, Payment_Due_Date

Member_Memberships (Member_ID(FK)(key), Membership_ID(FK)(key))
Key: Member_ID, Membership_ID

2NF: Are all non-key attributes functionally dependent on ALL of the keys? YES

3NF: Does it meet 2NF and are there no transitive dependencies? NO, **Members** and **Instructors** have transitive dependencies.

Instructors (Instructor_ID(key), Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_City, Instructor_State, Instructor_ZipCode, Instructor_PhoneNumber, Rate_of_Pay)
Key: Instructor_ID
FD1: Instructor_ID → Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_City, Instructor_State, Instructor_ZipCode, Instructor_PhoneNumber, Rate_of_Pay
FD2: Instructor_Zipcode -> Instructor_City, Instructor_State

Instructor_Location (Instructor_Zipcode(key), Instructor_City, Instructor_State)
Key: Instructor_Zipcode
FD1: Instructor_Zipcode -> Instructor_City, Instructor_State

Instructor_Information (Instructor_ID(key), Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_ZipCode(FK), Instructor_PhoneNumber, Rate_of_Pay)
Key: Instructor_ID
FD1: Instructor_ID → Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_ZipCode, Instructor_PhoneNumber, Rate_of_Pay

Members (Member_ID(key), Member_FirstName, Member_LastName, Member_Street, Member_City, Member_State, Member_ZipCode, Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type)
Key: Member_ID
FD1: Member_ID → Member_FirstName, Member_LastName, Member_Street, Member_City, Member_State, Member_ZipCode, Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type
FD2: Member_Zipcode -> Member_City, Member_State

Member_Location (Member_Zipcode(key), Member_City, Member_State)
Key: Member_Zipcode
FD1: Member_Zipcode -> Member_City, Member_State

Member_Information (Member_ID(key), Member_FirstName, Member_LastName, Member_Street, Member_ZipCode(FK), Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type)
Key: Member_ID
FD1: Member_ID → Member_FirstName, Member_LastName, Member_Street, Member_ZipCode, Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type

Final Set of Relations:

Instructor_Location (Instructor_Zipcode(key), Instructor_City, Instructor_State)
Key: Instructor_Zipcode
FD1: Instructor_Zipcode -> Instructor_City, Instructor_State

Instructor_Information (Instructor_ID(key), Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_ZipCode(FK), Instructor_PhoneNumber, Rate_of_Pay)
Key: Instructor_ID
FD1: Instructor_ID → Instructor_FirstName, Instructor_LastName, Instructor_Street, Instructor_ZipCode, Instructor_PhoneNumber, Rate_of_Pay

ClassOfferings (Class_ID(key), Class_Name, Class_Description)
Key: Class_ID
FD1: Class_ID → Class_Name, Class_Description

ClassSessions (Session_ID(key), Session_Start_Time, Session_End_Time, Instructor_ID(FK), Class_ID(FK))
Key: Session_ID, Instructor_ID, Class_ID
FD1: Session_ID, Instructor_ID, Class_ID → Session_Start_Time, Session_End_Time

Member_Location (Member_Zipcode(key), Member_City, Member_State)
Key: Member_Zipcode
FD1: Member_Zipcode -> Member_City, Member_State

Member_Information (Member_ID(key), Member_FirstName, Member_LastName, Member_Street, Member_ZipCode(FK), Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type)
Key: Member_ID
FD1: Member_ID → Member_FirstName, Member_LastName, Member_Street, Member_ZipCode, Email_Address, Member_PhoneNumber, Birth_Date, Gender, Payment_Type

Member_ClassSessions (Member_ID(FK)(key), Session_ID(FK)(key))
Key: Member_ID, Session_ID

Memberships (Membership_ID, Plan_Description, Membership_Start_Date, Membership_End_Date, Membership_Rate, Payment_Due_Date)
Key: Membership_ID
FD1: Membership_ID → Membership_Plan, Plan_Description, Membership_Start_Date, Membership_End_Date, Membership_Rate, Payment_Due_Date

Member_Memberships (Member_ID(FK)(key), Membership_ID(FK)(key))
Key: Member_ID, Membership_ID

Database Implementation

<CREATE TABLE>

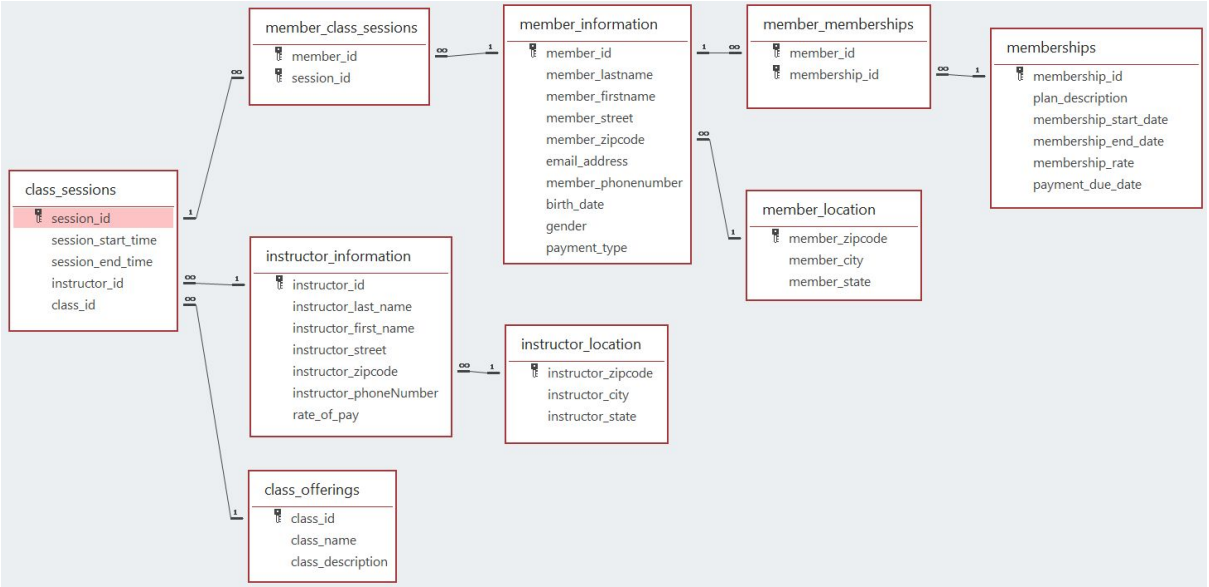
```
CREATE TABLE instructor_location(
     instructor_zipcode         VARCHAR(10) NOT NULL
     CONSTRAINT pk_instructor_location PRIMARY KEY,
     instructor_city            VARCHAR(20) NOT NULL,
     instructor_state           VARCHAR(10) NOT NULL
);

CREATE TABLE instructor_information(
     instructor_id              VARCHAR(10) NOT NULL
     CONSTRAINT pk_instructor_information PRIMARY KEY,
     instructor_last_name       VARCHAR(20) NOT NULL,
     instructor_first_name      VARCHAR(20) NOT NULL,
     instructor_street          VARCHAR(20) NOT NULL,
     instructor_zipcode         VARCHAR(10) NOT NULL,
     instructor_phoneNumber     VARCHAR(15) NOT NULL,
     rate_of_pay                CURRENCY
);

CREATE TABLE class_offerings(
     class_id                        VARCHAR(10) NOT NULL
     CONSTRAINT pk_class_offerings PRIMARY KEY,
     class_name      VARCHAR(20) NOT NULL,
     class_description          VARCHAR(20) NOT NULL
);

CREATE TABLE class_sessions(
     session_id                      VARCHAR(10) NOT NULL
         CONSTRAINT pk_class_sessions PRIMARY KEY,
     session_start_time         DATE NOT NULL,
     session_end_time           DATE NOT NULL,
     instructor_id              VARCHAR(10),
     class_id                   VARCHAR(10)
);

CREATE TABLE member_location(
     member_zipcode             VARCHAR(10) NOT NULL
     CONSTRAINT pk_member_location PRIMARY KEY,
     member_city                     VARCHAR(20) NOT NULL,
     member_state               VARCHAR(10) NOT NULL
);

CREATE TABLE member_information(
     member_id                  VARCHAR(10) NOT NULL
           CONSTRAINT pk_member_information PRIMARY KEY,
     member_lastname            VARCHAR(20) NOT NULL,
     member_firstname           VARCHAR(20) NOT NULL,
     member_street              VARCHAR(20) NOT NULL,
     member_zipcode             VARCHAR(10) NOT NULL,
     email_address              VARCHAR(50) NOT NULL,
     member_phonenumber         VARCHAR(15) NOT NULL,
     birth_date                 DATE,
     gender                     VARCHAR(10),
```

```
     payment_type               VARCHAR(20) NOT NULL,
);

CREATE TABLE member_class_sessions(
     member_id                  VARCHAR(10) NOT NULL,
     session_id                 VARCHAR(10) NOT NULL,
     CONSTRAINT pk_member_class_sessions PRIMARY KEY(member_id,
session_id)
);

CREATE TABLE memberships(
     membership_id              VARCHAR(10) NOT NULL
         CONSTRAINT pk_memberships PRIMARY KEY,
     plan_description           VARCHAR(100) NOT NULL,
     membership_start_date      DATE,
     membership_end_date        DATE,
     membership_rate            CURRENCY,
     payment_due_date           DATE,
);

CREATE TABLE member_memberships (
     member_id                  VARCHAR(10) NOT NULL,
     membership_id              VARCHAR(10) NOT NULL,
     CONSTRAINT pk_member_memberships PRIMARY KEY(member_id,
membership_id)
);
```

<ADD FK by using ALTER>

```
ALTER TABLE instructor_information
     ADD CONSTRAINT fk_instructor_location
     FOREIGN KEY (instructor_zipcode)
     REFERENCES instructor_location(instructor_zipcode)

ALTER TABLE class_sessions
     ADD CONSTRAINT fk_class_offerings
     FOREIGN KEY (class_id)
     REFERENCES class_offerings(class_id)

ALTER TABLE class_sessions
     ADD CONSTRAINT fk_instructor_information
     FOREIGN KEY (instructor_id)
     REFERENCES instructor_information(instructor_id)

ALTER TABLE member_information
     ADD CONSTRAINT fk_member_location
     FOREIGN KEY (member_zipcode)
     REFERENCES member_location(member_zipcode)

ALTER TABLE member_class_sessions
     ADD CONSTRAINT fk_member_class_sessions
     FOREIGN KEY (member_id)
        REFERENCES member_information (member_id)
```

```
ALTER TABLE member_class_sessions
   ADD CONSTRAINT fk_member_class_sessions_id
     FOREIGN KEY (session_id)
        REFERENCES class_sessions (session_id)

ALTER TABLE memberships
     ADD CONSTRAINT fk_member_information
     FOREIGN KEY (member_id)
     REFERENCES member_information(member_id)

ALTER TABLE member_memberships
   ADD CONSTRAINT fk_member_id
     FOREIGN KEY (member_id)
        REFERENCES member_information (member_id)

ALTER TABLE member_memberships
   ADD CONSTRAINT fk_membership_id
     FOREIGN KEY (membership_id)
        REFERENCES memberships (membership_id)
```



<ADD Data to the Tables>

instructor_location

```
INSERT INTO instructor_location
     VALUES ('11225', 'Brooklyn', 'NY');

INSERT INTO instructor_location
     VALUES ('10010', 'New York', 'NY');

INSERT INTO instructor_location
     VALUES ('11223', 'Brooklyn', 'NY');

INSERT INTO instructor_location
     VALUES ('11375', 'Maspeth', 'NY');

INSERT INTO instructor_location
     VALUES ('00000', 'Disappeared', 'NY');
```

instructor_information

```
INSERT INTO instructor_information
     VALUES ('I101', 'Choi', 'Eun Joo', '1234 Houston Street',
     '11225', '347-456-7890', 15.50);

INSERT INTO instructor_information
     VALUES ('I102', 'Mishkin', 'Jason', '653 Beverly Avenue',
     '10010', '636-558-2662', 15);

INSERT INTO instructor_information
     VALUES ('I103', 'Huang', 'Jennifer', '27 Stillwell Avenue',
     '11223', '718-998-0065', 17.25);

INSERT INTO instructor_information
     VALUES ('I104', 'Xie', 'Yiping', '8447 85th Avenue', '11375',
     '646-881-7952', 18.75);

INSERT INTO instructor_information
     VALUES ('I105', 'Saintleger', 'Nadine', '000 Absent Street',
     '00000', '000-000-0000', 0);
```

class_offerings

```
INSERT INTO class_offerings
     VALUES ('C101', 'Yoga', 'Improve Flexibility');

INSERT INTO class_offerings
     VALUES ('C102', 'Spin', 'Improve Cycling');

INSERT INTO class_offerings
     VALUES ('C103', 'Strength Training', 'Improve Strength');

INSERT INTO class_offerings
     VALUES ('C104', 'Dance', 'Full Body Workout');

INSERT INTO class_offerings
     VALUES ('C105', 'Cardio Fitness', 'Get Heart Rate Up');
```

class_sessions

```
INSERT INTO class_sessions
     VALUES ('S101', '9:30 AM', '10:30 AM', 'I101', 'C101');

INSERT INTO class_sessions
     VALUES ('S102', '11:00 AM', '12:00 PM', 'I102', 'C102');

INSERT INTO class_sessions
     VALUES ('S103', '12:30 PM', '01:30 PM', 'I103', 'C103');

INSERT INTO class_sessions
     VALUES ('S104', '3:00 PM', '5:00 PM', 'I104', 'C104');

INSERT INTO class_sessions
     VALUES ('S105', '4:00 PM', '5:30 PM', 'I101', 'C105');
```

member_location

```
INSERT INTO member_location
     VALUES ('11377', 'Woodmere', 'NY');

INSERT INTO member_location
     VALUES ('11205', 'Lawrence', 'NY');

INSERT INTO member_location
     VALUES ('10001', 'Cedarhurst', 'NY');

INSERT INTO member_location
     VALUES ('10451', 'Hewlett', 'NY');

INSERT INTO member_location
     VALUES ('10004', 'Inwood', 'NY');

INSERT INTO member_location
     VALUES ('10025', 'Springfield', 'NY');
```

member_information

```
INSERT INTO member_information
     VALUES ('M101', 'Williams', 'Robert', '881 52th Street',
     '11205', 'robert102@gmail.com', '917-256-5675', '8/12/1990',
     'M', 'credit card');

INSERT INTO member_information
     VALUES ('M102', 'Smith', 'Jane', '552 67th Street', '11377',
     'jane101@gmail.com', '347-444-2121', '2/3/1956', 'F', 'credit
     card');

INSERT INTO member_information
     VALUES ('M103', 'Johnson', 'Jack', '5656 Lanyard Boulevard',
     '10001', 'jack103@gmail.com', '347-727-1992', '4/5/1977',
     'M', 'debit card');

INSERT INTO member_information
     VALUES ('M104', 'Miller', 'Davis', '99 Canyon Street',
     '10451', 'davis104@gmail.com', '917-877-4563', '8/12/1981',
     'M', 'credit card');

INSERT INTO member_information
     VALUES ('M105', 'Brown', 'Summer', '30 Water Street',
     '10004', 'summer105@gmail.com', '917-377-2342', '5/12/1979',
     'F', 'debit card') ;

INSERT INTO member_information
     VALUES('M106', 'Stevens', 'Ryan', '215 Spring Street',
     '10025', 'ryan106@gmail.com', '347-123-4568', '12/7/1963',
     'M', 'credit card');
```

member_class_sessions

```
INSERT INTO member_class_sessions
     VALUES ('M101', 'S101');

INSERT INTO member_class_sessions
     VALUES ('M101', 'S104');

INSERT INTO member_class_sessions
     VALUES ('M102', 'S102');

INSERT INTO member_class_sessions
     VALUES ('M102', 'S105');

INSERT INTO member_class_sessions
     VALUES ('M103', 'S102');

INSERT INTO member_class_sessions
     VALUES ('M103', 'S103');

INSERT INTO member_class_sessions
     VALUES ('M104', 'S103');

INSERT INTO member_class_sessions
     VALUES ('M105', 'S105');

INSERT INTO member_class_sessions
     VALUES ('M106', 'S101');

INSERT INTO member_class_sessions
     VALUES ('M106', 'S104');
```

memberships

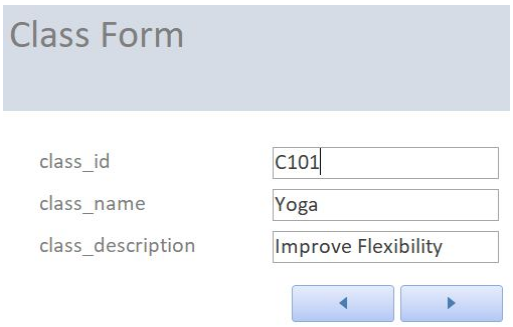
```
INSERT INTO memberships
     VALUES ('P101', '1st Quarter', '1/1/2020', '3/31/2020', 60,
     '1/1/2020' )

INSERT INTO memberships
     VALUES ('P102', '2nd Quarter', '4/1/2020', '6/30/2020', 60,
     '4/1/2020' )

INSERT INTO memberships
     VALUES ('P103', '3rd Quarter', '7/1/2020', '9/30/2020', 60,
     '7/1/2020' )

INSERT INTO memberships
     VALUES ('P104', '4th Quarter', '10/1/2020', '12/31/2020', 60,
     '10/1/2020' )
```

member_memberships

```
INSERT INTO member_memberships
     VALUES ('M101', 'P101')

INSERT INTO member_memberships
     VALUES ('M101', 'P102')

INSERT INTO member_memberships
     VALUES ('M101', 'P103')

INSERT INTO member_memberships
     VALUES ('M101', 'P103')

INSERT INTO member_memberships
     VALUES ('M102', 'P102')

INSERT INTO member_memberships
     VALUES ('M103', 'P103')

INSERT INTO member_memberships
     VALUES ('M103', 'P104')

INSERT INTO member_memberships
     VALUES ('M104', 'P104')

INSERT INTO member_memberships
     VALUES ('M105', 'P102')

INSERT INTO member_memberships
     VALUES ('M105', 'P103')

INSERT INTO member_memberships
     VALUES ('M106', 'P101')
```

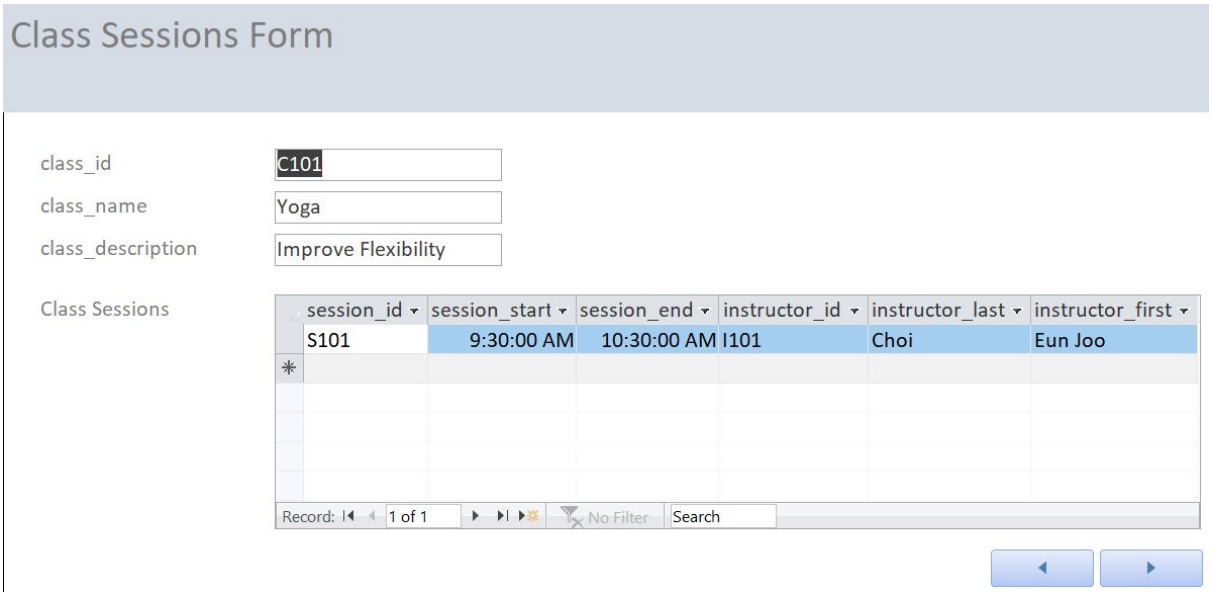
Application Implementation

Class Form



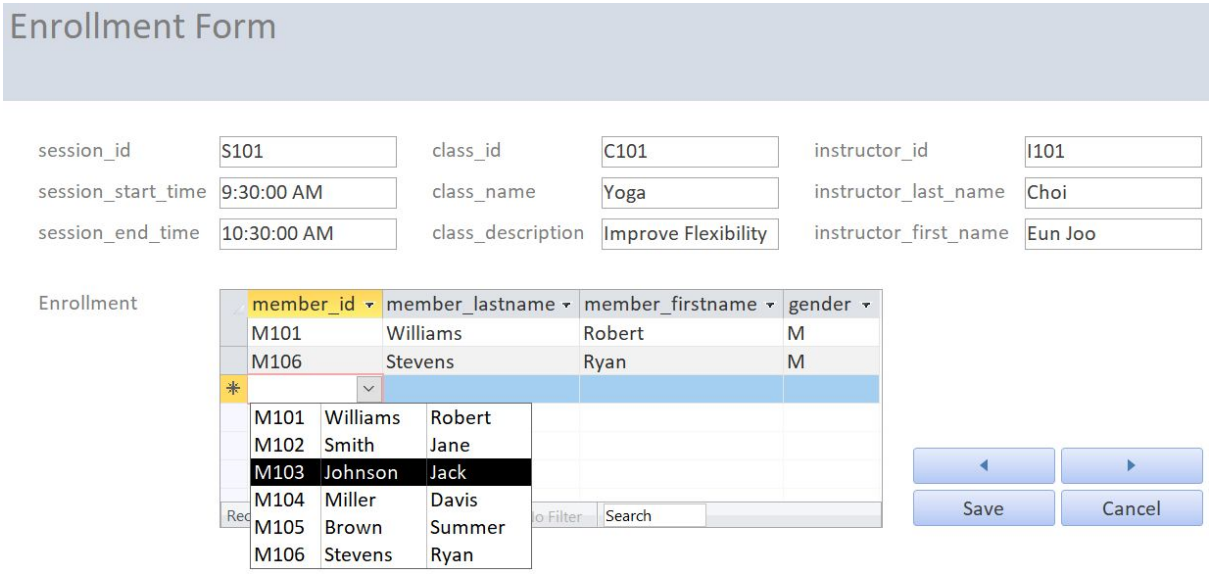
The class form is used to look up, update, and add different types of classes that our gym offers or will offer.

Class Sessions Form



The purpose of the class sessions form is to show the sessions each class has, the time it begins and ends as well as the name of the instructor that's teaching that particular session.

Enrollment Form



The enrollment form is to record the class sessions each member signed up for. We can also add members to a particular section using this form.

Instructor Form

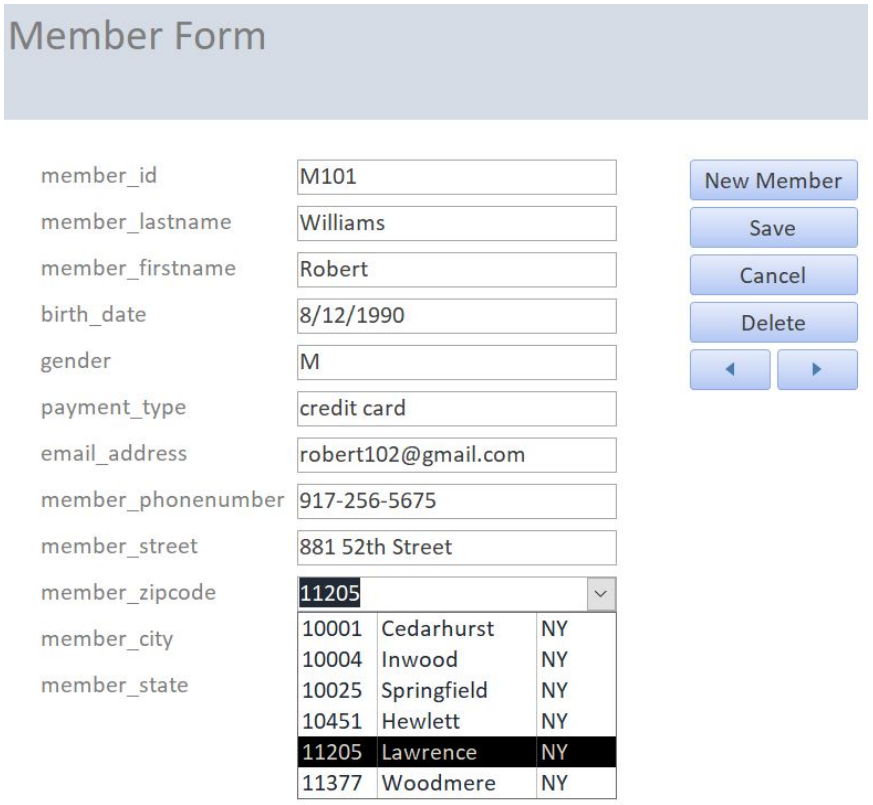
The instructor form shows detailed information for each instructor. The user can add an instructor by clicking the “New Instructor” button. City and State fields are automatically filled in after selecting a zipcode from the combo box. Additionally, changes could be entered into the form and recorded by clicking the “Save” button. This form also has VBA code to automatically capitalize the first initials of the FirstName and LastName.

<VBA code>

```
Private Sub instructor_firstname_AfterUpdate()
instructor_firstname = StrConv(instructor_firstname, vbProperCase)
End Sub
-----------------------------------------------------------------
--
Private Sub instructor_lastname_AfterUpdate()
instructor_lastname = StrConv(instructor_lastname, vbProperCase)
```

```
End Sub
```

Member Form

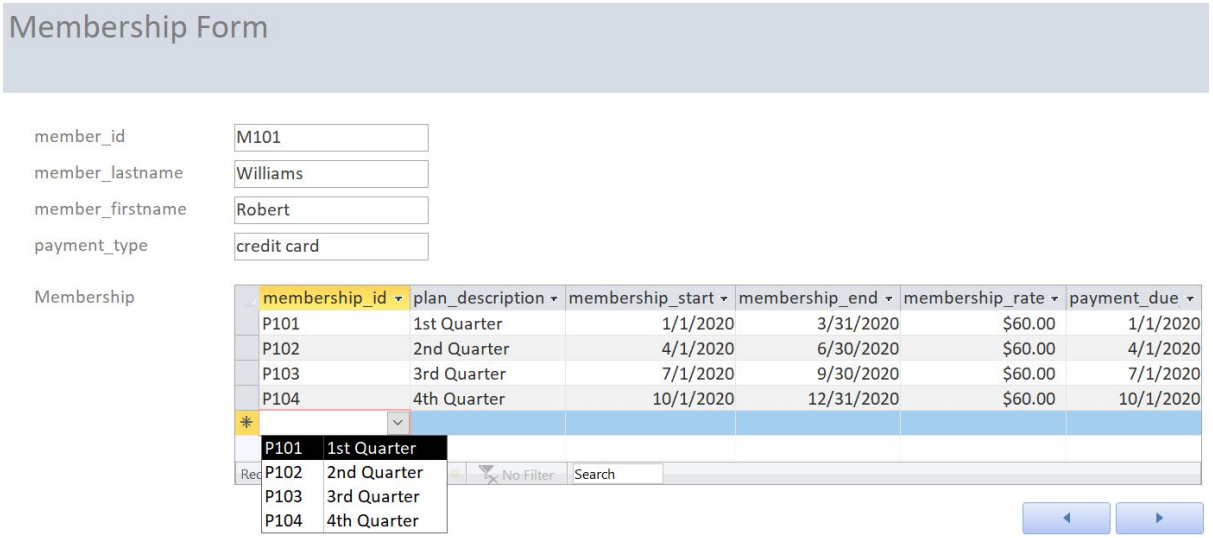


The member form is used to look up current members and to add new member information. The city and state field are automatically populated by selecting the zipcode from the combo box. The Member Data Entry Form has VBA code to automatically capitalize the initials of the FirstName and LastName.

<VBA code>

```
Private Sub member_firstname_AfterUpdate()
member_firstname = StrConv(member_firstname, vbProperCase)
End Sub
-----------------------------------------------------------------
--
Private Sub member_lastname_AfterUpdate()
member_lastname = StrConv(member_lastname, vbProperCase)
End Sub
```

Membership Form



The purpose of the membership form is to manage the memberships signed up by each member and to keep track of the status of the membership whether it needs to be renewed and the payment due date for each membership. We can add new memberships that the member signs up for via the dropdown box in the table. It will autofill the rest of the details in the table once selected.

MemberCount Report

<Query for this Report>

```
SELECT c.class_id, c.class_name, c.class_description,
     COUNT(m.member_id) AS MemberCount
FROM member_class_sessions AS m, class_offerings AS c,
     class_sessions AS s
WHERE (((m.session_id)=s.session_id) And
((s.class_id)=c.class_id))
GROUP BY c.class_id, c.class_name, c.class_description;
```

This query and report shows the total number of members enrolled in each class. The query selects the class id, name, description, and counts the number of members in each type of class. The purpose is to find out which classes do members come to our gym for. With this knowledge, we can find out which classes we need to increase sessions for and get rid of unpopular classes.

TotalPayment Report

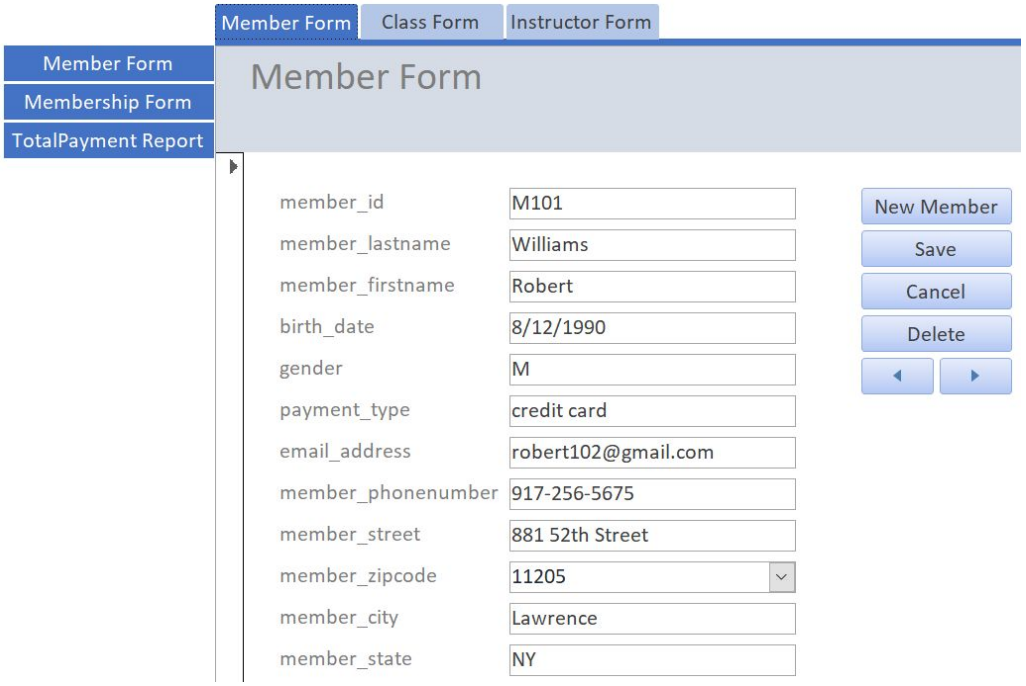
TotalPayment Report

member_id	member_lastname	member_firstname	payment_type	membership_id	plan_description	payment_due_date	membership_rate
M101	Williams	Robert	credit card	P101	1st Quarter	1/1/2020	$60.00
				P102	2nd Quarter	4/1/2020	$60.00
				P103	3rd Quarter	7/1/2020	$60.00
				P104	4th Quarter	10/1/2020	$60.00
Sum							$240.00
	Summary for 'member_id' = M101 (4 detail records)						
M102	Smith	Jane	credit card	P102	2nd Quarter	4/1/2020	$60.00
Sum							$60.00
	Summary for 'member_id' = M102 (1 detail record)						
M103	Johnson	Jack	debit card	P103	3rd Quarter	7/1/2020	$60.00
				P104	4th Quarter	10/1/2020	$60.00
Sum							$120.00
	Summary for 'member_id' = M103 (2 detail records)						
M104	Miller	Davis	credit card	P104	4th Quarter	10/1/2020	$60.00
Sum							$60.00
	Summary for 'member_id' = M104 (1 detail record)						
M105	Brown	Summer	debit card	P102	2nd Quarter	4/1/2020	$60.00
				P103	3rd Quarter	7/1/2020	$60.00
Sum							$120.00
	Summary for 'member_id' = M105 (2 detail records)						
M106	Stevens	Ryan	credit card	P101	1st Quarter	1/1/2020	$60.00
Sum							$60.00
	Summary for 'member_id' = M106 (1 detail record)						
Grand Total							$660.00

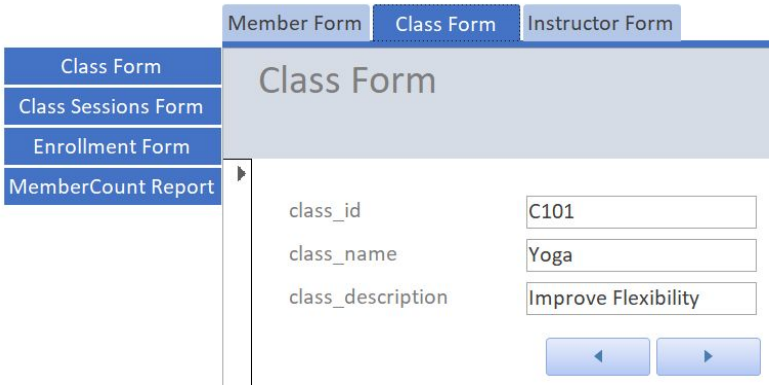
The purpose of this form is to find out and keep track of the total payment each member needed to pay based on the types and number of memberships they have this year.

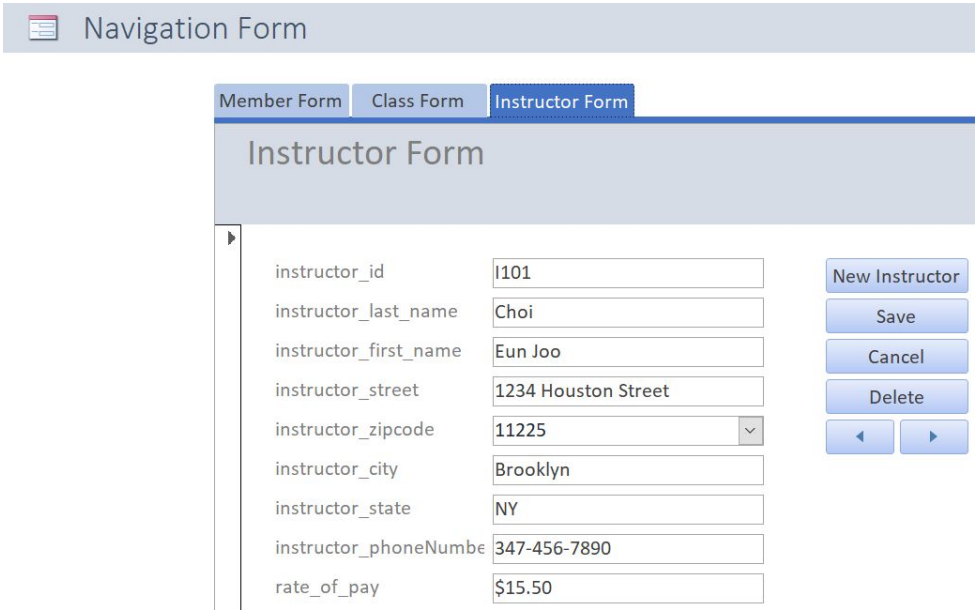
Navigation Form

Navigation Form



Navigation Form





This navigation form is split into three sections. Forms and reports are sorted into the Member Tab, Class Tab, and Instructor Tab. The Member Tab contains all the forms that are needed to enroll a new member and assign memberships. The report keeps a track of all the members, their memberships, their payment dates, and how much they owe in total to the gym. The Class Tab contains all the forms regarding classes: what classes and sessions our gym offers, and which members are enrolled in which session of which class. The report in the Class Tab keeps track of classes and how many are enrolled in that class. Lastly, the Instructor Tab simply contains the form regarding the instructor's information.

Conclusion

We mainly communicated through email and an app called "Discord" while working on the project. We worked together through chat, voice call, and screen share. The easiest part of this project was identifying the entities that played a role in our gym because it was similar to a college setup. As we progressed through the final phase, we were able to find problems with the database model we designed earlier. We had to discuss several times and test different methods to process the data more efficiently. It took quite a bit of time to understand it; however, through this process, we learned how important it is to logically express and implement the schema of the database.

For our group, we would say the easiest part was setting up times to meet up. Despite the random assembly of group members, our group members replied quickly and were active often to answer each other's questions. A difficulty we faced was being unable to edit the Access file at the same time. There were times where different members were working on the file at the same time on their own computer and it was hard to merge the changes together into one file. If possible, next time we should set up a status on the file to inform members that someone is already working on it or not.

With this new system, our fictional gym would be better at keeping track of members and their classes. It can use the system to identify classes that are not popular and cut down on losses from employing instructors for running empty classes. They can also increase the number of sessions for popular classes. The gym can now also keep track of seasonal trends, such as “the summer body” where many customers will want to be fit during the summer. They can also use the system to keep track of the members’ memberships to know who to collect money from, and prevent unauthorized entry and use from members whose memberships have expired.

This project was a really good way to gain experience on how to build a database. It was really interesting how the steps we took in class--from creating an entity relationship model, to converting it to a relational model, to then normalizing the relations, etc.--all came back in a full circle. And then how we were able to create our own tables inside Microsoft Access using SQL to supply the entities with information for our database. Overall, this project of creating a database was very intuitive, and kept us on our toes throughout the entire process.