# CMPS 460 CAPSTONE:

## UL-LAFAYETTE GYM DATABASE

Gabe Marcus (C00058952)

Nick Barreca (C00117074)

Robbie Indorf (C00205928)

# I. Project Requirements

The University of Louisiana at Lafayette's (ULL) gym has many uses for many different people. These uses range from students who wish to lift weights, to teachers who teach courses there, to students who wish to relax by the pool on a hot day. The objective is to design a database that will keep track of the total people entering the gym, where they go in the gym, and at which times the gym has the most people. The gym is open from 8:00 in the morning, until 8 o'clock at night every day.

The database for this gym will be responsible for keeping track of the people who enter the gym and different sub sections of the gym. Firstly, the database must keep track of how many people enter the gym. The people allowed to enter the gym are only those with a valid ULID. Those with this ID are students and faculty of the University. Secondly, the database will keep track of which sub section of the gym these people are going to.

The following interactions are expected from the database system:

1: List of all people who entered the gym on any given day. This list will differentiate between teachers/front desk workers/ trainers/ and students.

2: List of people in each sub section of the gym on any given day

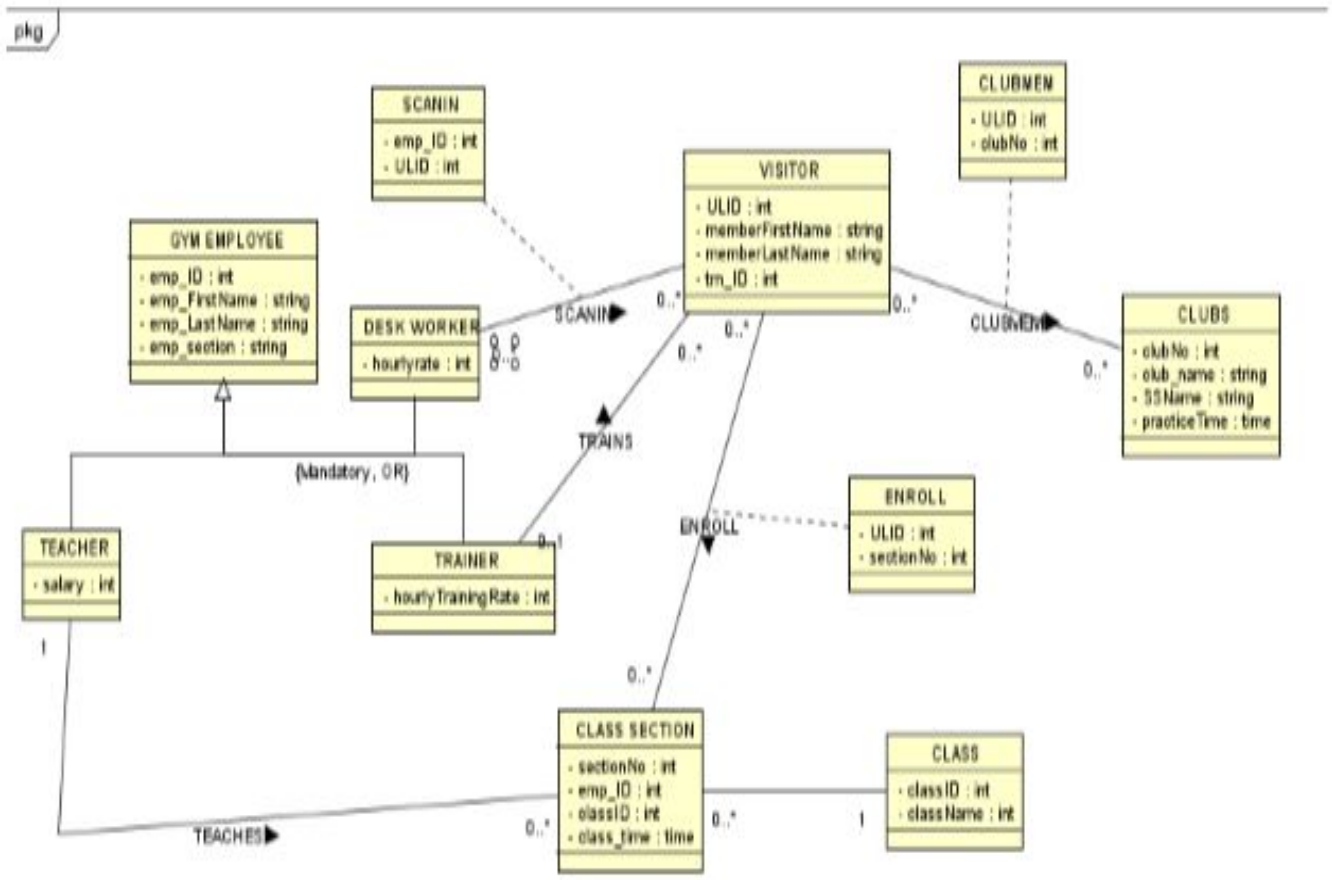
3: Keep track of the times in which people enter the gym, in order to represent busy times vs slower times

4: List of all clubs registered into the system who use the gym

5: Show class courses and allow enrollment into those courses for tracking purposes

# II. Data Modelling

The Gym Employee can be three types. They can be a deskworker, a teacher, or a trainer. Each has their own interactions with the visitors of the gym. Firstly, and the main part of the database is the deskworker. The deskworker interacts with visitors by scanning them into the gym and by scanning them into subsections. This means that we will know exactly what time the visitor entered the gym, and where he went via the "scanin" class. Other features in the database include classes and clubs available. Teachers sign into the gym to teach the students. Different clubs will be using the gym as well and their practice time is recorded so the gym knows when certain clubs will be using certain rooms.




Teacher(**emp_ID**,emp_fName,emp_lName,emp_section,salary)

DeskWorker(**emp_ID**,emp_fName,emp_lName,emp_section,hourlyrate)

Trainer(**emp_ID**,emp_fName,emp_lName,emp_section,hourlyTrainingRate)

Visitor(**ULID**,memberFirstName,memberLastName,trn_ID{FK})

Class_Section(**sectionNo**,emp_ID{FK},classID{FK},class_time)

Class(**classID**,className)

Enroll(**ULID{FK},sectionNo{FK**})

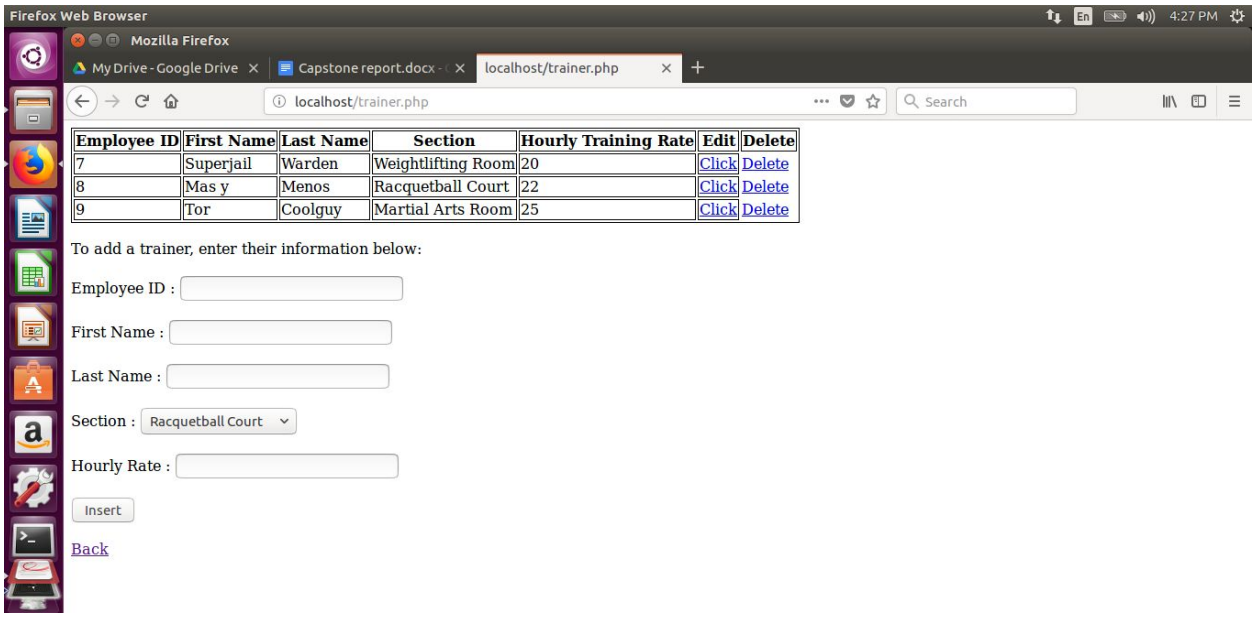
ScanIn(**emp_ID{FK},ULID{FK**})

Clubs(**clubNo**,club_name,SSName,practiceTime)

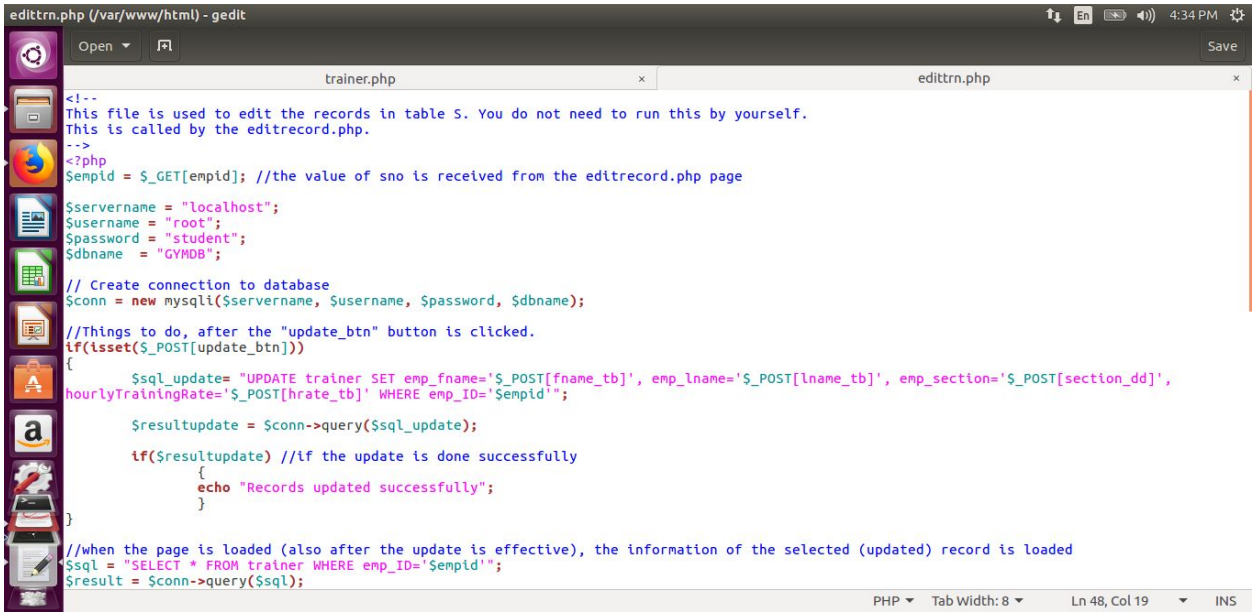
ClubMem(**ULID{FK},clubNo{FK**})

# III. Implementation

The actual implementation of the database went smoothly. The professor of the course this Capstone project was made for was kind enough to provide ready-made PHP examples that displayed how to properly use and display the results of three major types of SQL queries: INSERT, UPDATE, and DELETE. Code was simply copied from those three examples and modified to include the relevant variables to the GYMDB database.




This is a picture taken from the trainer.php page, which displays the data in the trainer table. Clicking on one of the buttons in the “Edit” column opens a separate page that allows you to edit the parameters relevant to that row of data (except the primary key, which in this case is the employee ID). Clicking one of the “Delete” buttons deletes that row from the table. Below the table is where one can insert a new trainer into the table by inserting the relevant information upon clickage of the “Insert” button. The “Back” link takes the user back to the database index page.




This is a code snippet from the “edittrn.php” file. This page is linked to from the trainer.php page, with the url being localhost/edittrn.php?empid='.$row['emp_ID'].'. '.$row['emp_ID'].' is obtained while the table on the trainer.php page is being built. ‘emp_ID’ is the internal name of the employee ID variable in the trainer table. For example, if the employee ID is 6, then the URL would be localhost/edittrn.php?empid=6. The first line of PHP code grabs this value from the URL and stores it into the $empid variable, which is used in the SELECT statement to obtain data from that row only and in the UPDATe statement to make sure only that row is updated upon request. Before SQL queries can be made, a connection must be made. To accomplish that, the $servername, $username, $password, and $dbname variables are set to the correct values, then a new connection (stored into the $conn variable) is made. The format for creating and sending SQL queries is as follows:

$sql=SQL Query (example: SELECT * FROM trainer);

$result=$conn->query($sql);

if($result) [if the SQL request is successful] {

echo ‘Records retrieved’;

}

The query is created in line 1 then executed in line 2. If the query is successful (line 3), a message is outputted (line 4).

The UPDATE code is within an if statement. “if(isset($_POST[update_btn]))” means to run the code within the if statement when the button with the name “update_btn” is clicked.

NOTE: to run the database yourself, open a terminal in the project folder and type “source scenario2.sql”. Running this file creates the database GYMDB and populates the tables with example data.