1. Overview

In this lab, you will:

- Decide what data you will use for your AngularJS project.
- Learn (or review) the basics about databases by studying (or skimming) a MySql Workbench Tutorial (MySql Workbench is an open source GUI for the MySql database management System).
- Using MySQL Workbench, design, implement, & populate a database to support your AngularJS project.
- Write some SQL SELECT Statements.
- Create and publish a word document that has screen captures of your database work in this lab, then add a blog (about what you learned in this lab) to your labs page and link to the word document from that blog.

2. Lab Requirements

A. Create a document that will hold screen captures as you work through this lab. I will highlight all the screen captures that you shall put in the word document.

B. Create a database that consists of two tables, as follows. Follow Database table naming conventions:

- Do not any SQL keywords in your table design (table name, field names) because it will cause unwanted database exceptions later in the semester accessing your database from java code. Some SQL keywords to avoid: role, user, password, state (google "SQL Keywords" for complete list)
- Table names should be singular, not plural, e.g. "student" not "students".
- Table names should not include space – you can use underscore if you have a multi-word name, or you can use camel case since the SQL in mysql is case sensitive.

- A User Table with at least these fields:
  - User Id (primary key, auto-increment)
  - User email (they will use as their log on name, must be unique)
  - User Password (null not allowed)
  - User Nickname (null not allowed)
  - Role Name (e.g., ADMIN or MEMBER, null not allowed)

  Do NOT name this table "user" (or any upper/lower case version of this SQL keyword).

- An Other Table that your AngularJS project will access and modify in future labs. Name this table with a self documenting name that represents what you will store (you cannot name it "other").
  - tableName_id (primary key, auto-increment)
  - some sort of descriptor or identifier (must be unique)
  - A field that will hold a long URL to an image.
  - If you wish, you can also have a field that will hold a URL to a website.
  - At least 2 other fields, at least one of which is non-character (either date, integer, or decimal) and nullable.

  Populate this table with at least 20 records (realistic looking data). At least one record shall have all fields populated. At least record shall have null in all null-able fields.
C. **Get a screen capture** from MySql Workbench of the table design of each of your two tables (and paste it into your document). To see the table design from MySql Workbench, right click on the table name (left pane) and select "Alter Table".

![Screen capture of table design in MySql Workbench](image)

Click here if you can’t see enough of the "columns" area.

To get a good (legible) screen capture, you can do an Alt-PrtSc (copies the active window into the clipboard), paste into a simple program like MSPaint, then select just the part you need which should look similar to this:

![Table design in MSPaint](image)
D. **Insert user Records.** Once you start writing your web application, your server side (java) code will be attempting to insert, update, and delete records. Therefore you need to know enough SQL to accomplish this. Also, the DBMS (database management system) will always protect your data and not allow anything to happen that would violate the rules you set up when you designed your tables – and this type of database protection will throw java exceptions that you must handle. So, I’m asking you to use MySQL Workbench to learn about SQL insert/update/delete commands and the database exceptions these operations may cause.

- Right click your user table (from the database navigator area) and “Select Rows” from your (empty) user table. Insert a user record using the grid and click the Apply button. Before you click Apply again, MySql WB will show you the SQL command that it intends to execute, which should look something like this:

  ```sql
  INSERT INTO 'FA12_2308_tuc51856'.'web_user' ('user_email', 'user_password', 'user_role') VALUES ('sallyk', 'pw', '1');
  ```

  Click through the process to insert the record and hopefully the record will be successfully inserted.

- Paste the SQL INSERT command into the Query area and then edit it so that it looks more like this (delete the unnecessary “back tick” single quotes and add the USE command so you don’t have to precede the table name with the database name). Click on the lightning bolt icon to run the SQL commands.

  Get a screen capture that shows a successful insert – the output area should say “1 row(s) affected”.

- Try to execute the same INSERT statement again and you should get an error message because this insert would violate the email address uniqueness constraint that you added to your user table. Get a screen capture that shows the database “throwing an exception” and not allowing the insert to occur.

  Your server side java code might experience this type of database exception and it has to know how to communicate the problem back to the client side so that your JavaScript code can tell the user what the issue is.

- Populate your user table with at least 5-7 records of realistic looking data.

E. **Update user records.** Using the grid, update a user record. Do the same trick (learning SQL commands from MySql Workbench) – copy the UPDATE command before clicking “Apply” the second time. Paste the UPDATE command and edit it (remove back ticks and remove the database prefix before the table name). Run this update command by clicking on the lightning bolt. Get a screen capture showing that you successfully ran your own UPDATE command on your user table.
F. **Delete user records.** Using the grid, click to the left of a data row, then right click and select “delete”. Do the same trick (learning SQL commands from MySql Workbench) – copy the DELETE command before clicking “Apply” the second time. Paste the DELETE command and edit it (remove back ticks and remove the database prefix before the table name). Run this update command by clicking on the lightning bolt. Get a screen capture showing that you successfully ran your own DELETE command on your user table.

G. **Insert “other” records.**

- Populate your “other” table with around 20 records of realistic looking data. Make sure that at least one record has all fields filled out and at least one record has null in all nullable fields.
- Note: Eventually (before the next lab, the Web API / ajax lab), you will need your image URL field to hold actual (fully qualified) URLs to images on the web, but for now, you can leave that column blank.

H. Write and execute the following three SQL SELECT statements, then get a screen capture of each and paste this screen capture into your document. Each screen capture shall include: SQL Query area, Result Set area, plus Output area, as shown below. To get a good (legible) screen capture, size the areas of MySqlWorkBench as small as possible showing only the data you need to show, then Alt-PrtSc as mentioned before, and select out just the part you need to show.

NOTE: substantial deductions will be taken if we see syntax errors in the query window or in the output area.

- **First select statement:** All the fields of all records in your User table, ordered by email address. Select the column names individually (don't use SELECT *) beginning with the email address which is the “ORDER BY” column.
• **Second select statement:** All the fields of all records in your Other table, ordered by the unique descriptor field. Select the column names individually (don't use SELECT *) beginning with the descriptor field which is the ORDER BY column. Order the other columns as you think your users would want to see the data on the web page you will be creating later in the semester. If you can't get all 20+ records to show on a single screen capture, just get the first records on one screen capture and the last records on another screen capture. BUT we do need to see the two records mentioned above: at least one record with all fields populated and at least one record with null in all nullable fields. If your screen capture is not wide enough to show all columns, just put the more important columns first and show these.

• **Third select statement:** The same columns and order as the second select statement, but having an additional condition in the WHERE clause that uses the SQL LIKE keyword and % for wild card match. Choose your WHERE clause so that you get at least several rows in your result set (but not more than can fit in one screen capture). Here is an example using the LIKE SQL keyword that returns all records where user_email starts with "S":

```
SELECT * FROM web_users WHERE user_email LIKE 'S%';
```

### 3. Submission

The document with your screen captures shall be named with your last name in it and it shall include the following:

- **Table Designs (from mySqlWorkBench):**
  - Screen capture your user table design.
  - Screen capture of your other table design.

- **INSERT/UPDATE/DELETE user:**
  - Screen capture of successful insert (after you typed in the insert command to the Query Area).
  - Screen capture of insert that failed due to uniqueness constraint (with you typing the SQL into the Query Area again).
  - Screen capture of successful update (you typing SQL into Query Area)
  - Screen capture of successful delete (you typing SQL into Query Area)

- **SQL select statements:** Screen capture of each of the 3 SQL select statements listed above.

To your labs.html page, add a blog that (1) describes what you did in this lab and (2) links to the document described just above. You link to the document just like you would like to a html page, for example:

```
Click <a href="smithDataBase.doc">here</a> to see my database work.
```

As always, publish your work and submit a zip file of your whole project into blackboard.