**The (Failed) Creation of a Language Change Simulator**

The goal of this project is to create a software that simulates language change. Below is a list of steps I have to take towards completing the project. Of course, I was wayyyy too overly optimistic when I first envisioned the project…

**Describing the language**

In order to simulate language change, we must first describe the current state of the language. Therefore, our software needs to be able to read in and store the grammar of the language. The user should be able to input descriptions of an existing language, or they could use the software to create their own!

**1 Phonemic inventory**

The most basic component of a (spoken) language is a phoneme – the smallest unit of sound. Every language has a phonemic inventory, which is a collection of sounds that a speaker of that language perceives as distinctive. Each phoneme can be described using a feature matrix – a set of minimally necessary binary features that distinguishes it from all other phonemes in the language.

Thus, our software must be able to (1) store a phonemic inventory, (2) associate each phoneme to its set of features, and (3) allow the user to access phonemes via feature matrixes, and vice versa. As a bonus, we could also incorporate an algorithm that outputs a feature matrix given a list of feature values, and a system that allows the user to describe or create a phonetic-based writing system and associate each phoneme to an orthographical element.

**2 Phonology**

Every language has phonological rules that govern how phonemes are realized in speech. These rules are encoded using the same feature matrixes described above.

Thus, our software must be able to (1) allow the user to input and store phonological rules, and (2) correctly apply these rules to phonemic representations and yield the appropriate phonetic forms.

**Time out…**

So by then I realized that even getting that far is a bit of a stretch… And I was right…

**An account of my unfortunate struggles**

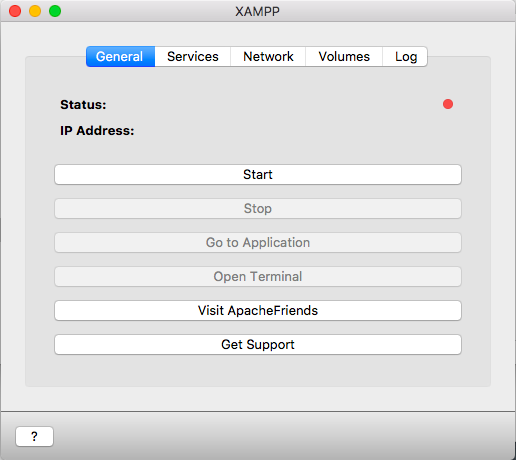
**1 Creating a user interface**

When I first started the project, I decided that the first thing I needed to do was to create a user interface. I found the *Qt5 C++ GUI Programming Cookbook* (URL: https://culturalengineerassociation.weebly.com/uploads/8/6/7/7/86776910/qt5\_c\_\_\_gui\_programming\_cookbook.pdf) and followed instructions. I learned to use stylesheets, layout options, spacers, and various clicking reactions.

**2 Setting up a database**

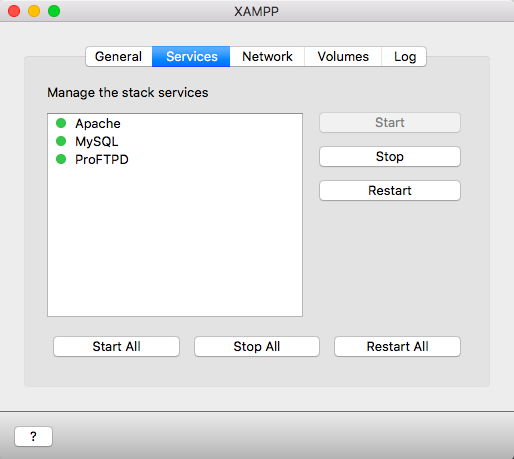
Following the Qt Cookbook, I attempted to create and connect to a database. I downloaded XAMPP, which is a package that includes Apache and MySQL, which I need in order to create a database. Since the version of XAMPP that I downloaded has a very different user interface than the one described in the cookbook, I had to figure out how to make it work. Below is a list of steps (with screenshots!) that I took to set up a simple database for any unfortunate soul to follow:

(1) Launch XAMPP and click “Start”



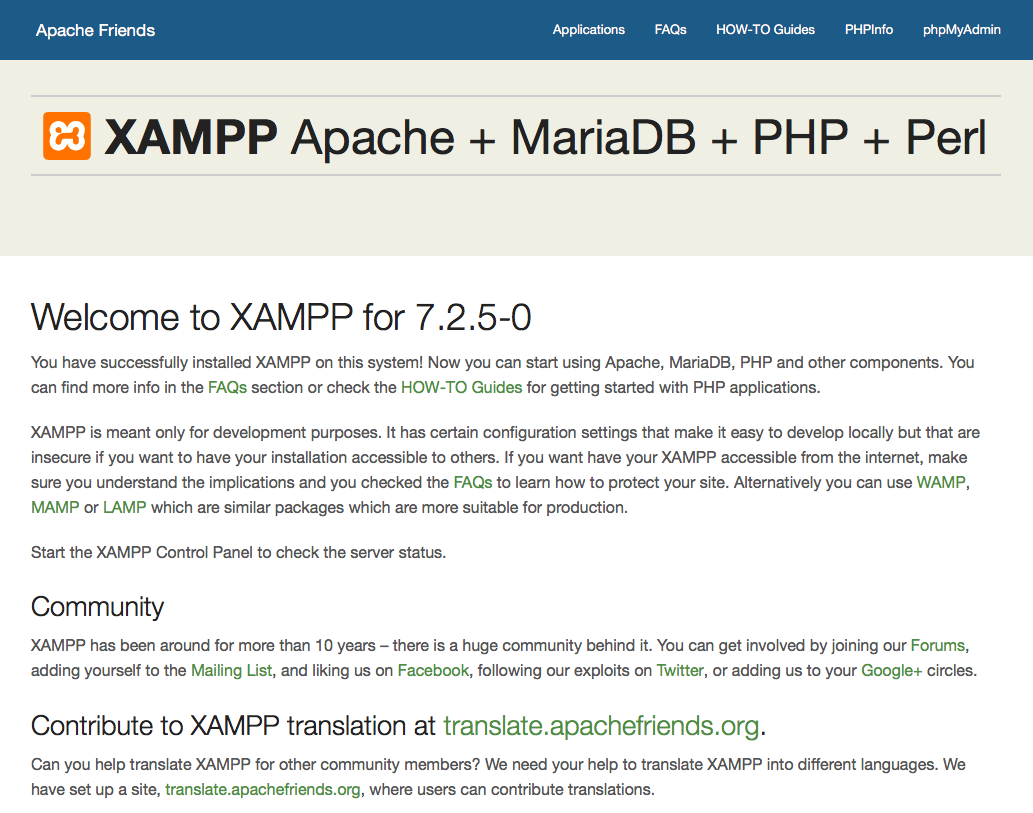
(2) Click on “Services”. If the circles next to Apache and MySQL are red, click on each and

click on start. If they are both green, you’re good to go!



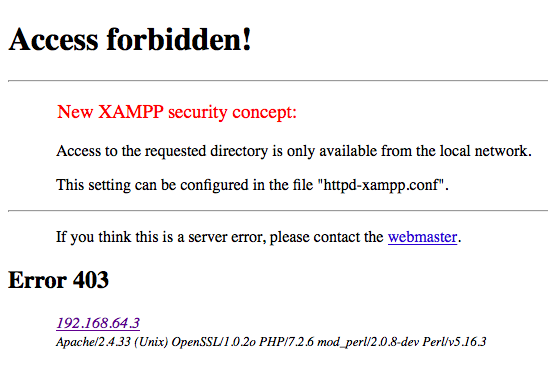
(3) Go back to “General” and click on “Go to Application”. It should take a page that looks

like this:



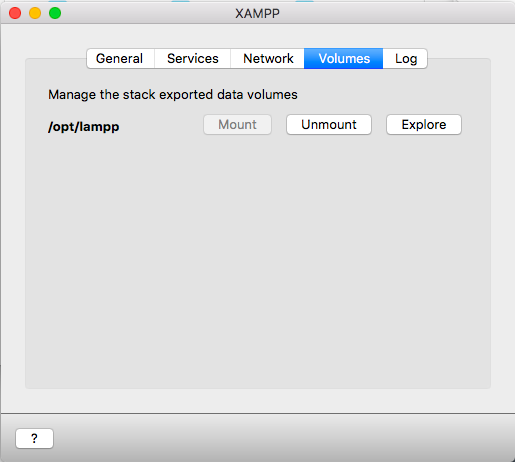
(4) Click “phpMyAdmin” at the top right corner of the page. You will most likely see this

error message:

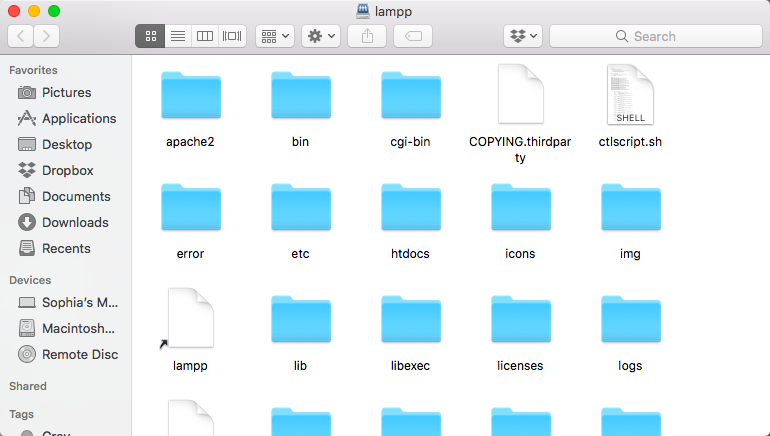


Note: It took me foreeever to figure out how to fix this. You’re welcome!

(5) Go back to the XAMPP interface and click on “Volumes”. Then, click on “Mount”.

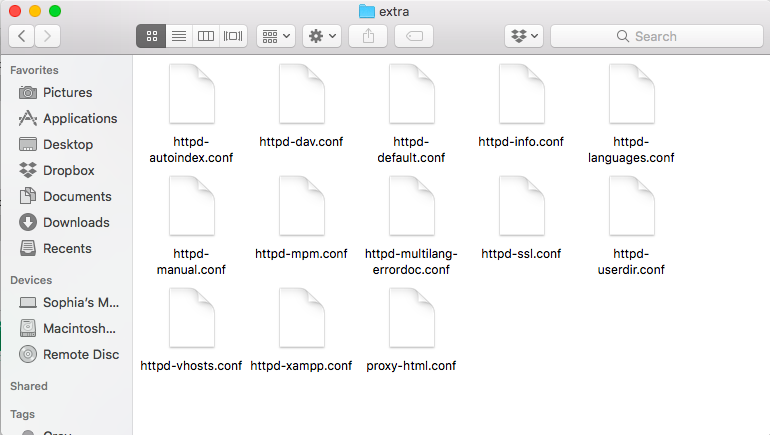


(6) Click on “Explore”, and you’ll be taken to a Finder window that looks like this:

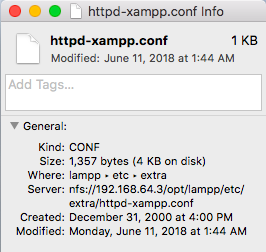


(7) Click into the folder “etc”, then into another folder “extra”, where you’ll see a file

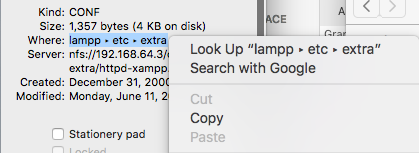
named “httpd-xampp.conf” (second one in last row). We need to modify this file in order to fix the Access Forbidden error.



(8) Right click on the file and select “Get Info”. A window like this will pop up:



(9) Select the information after “Where” and copy it.



(10) Open Terminal, type in **cd**, space, and paste the address you just copied. Hit enter. Your

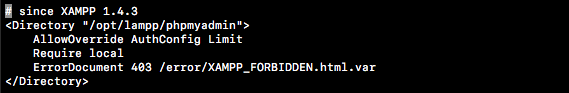
Terminal should look something like this:



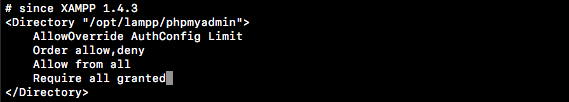
(11) Type **vim httpd-xampp.conf** and hit enter. You will see something like this:



(12) Hit ‘i’ (stands for “insert”) so we can modify the file. Find the following section:

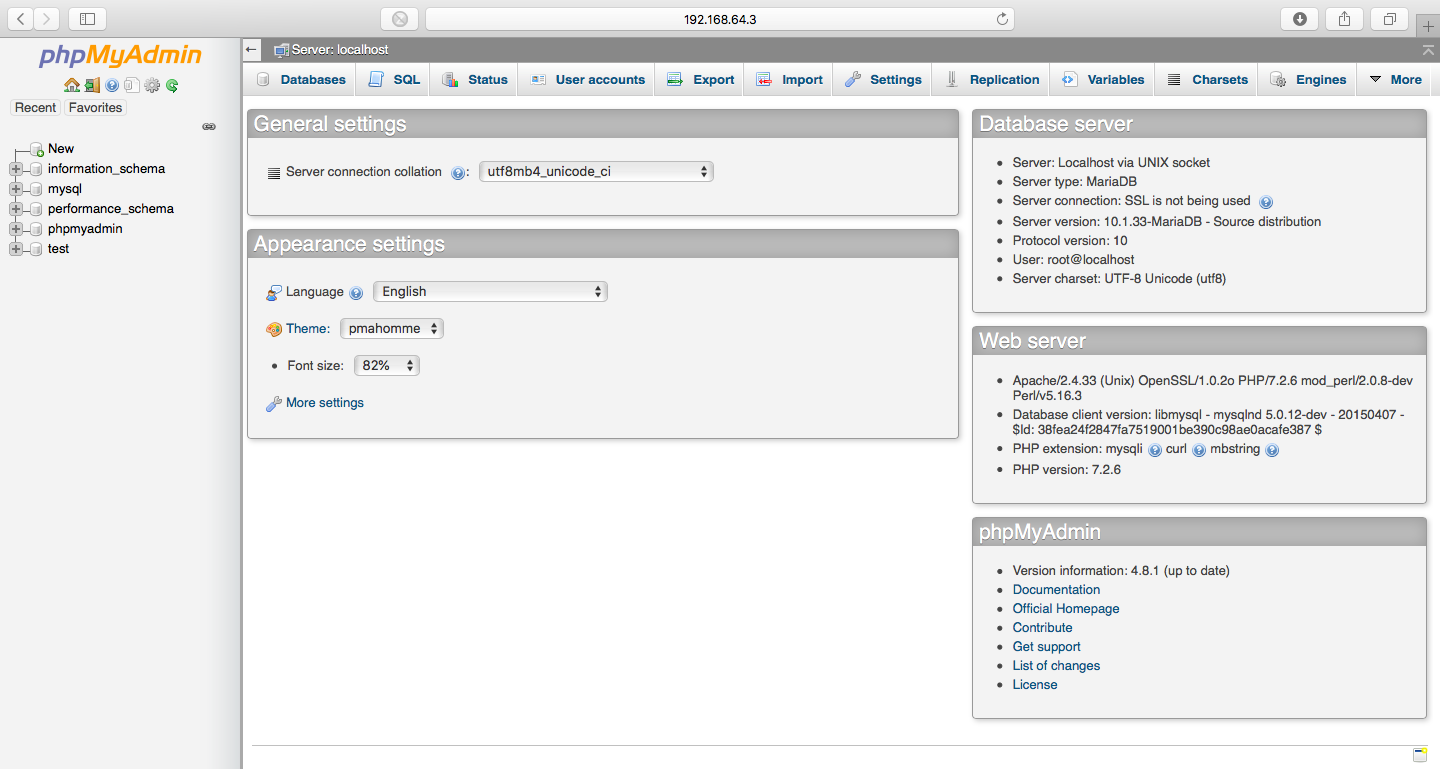


(13) Modify this section as follows:



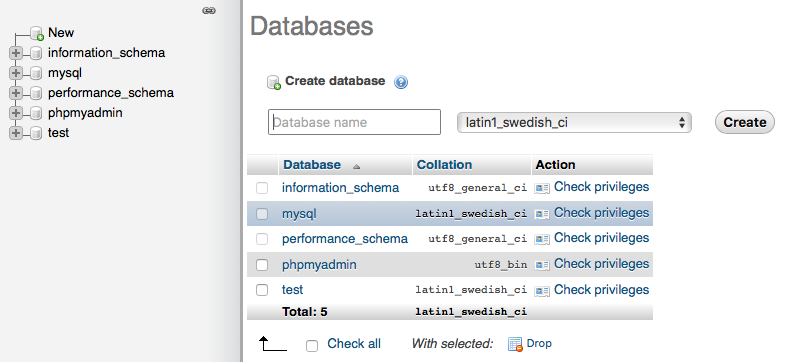
(14) Hit **esc**, type **wq**, then hit enter to save and quit. Now, restart XAMPP following steps (1)

to (4). This time, you should be taken to an interface like this:



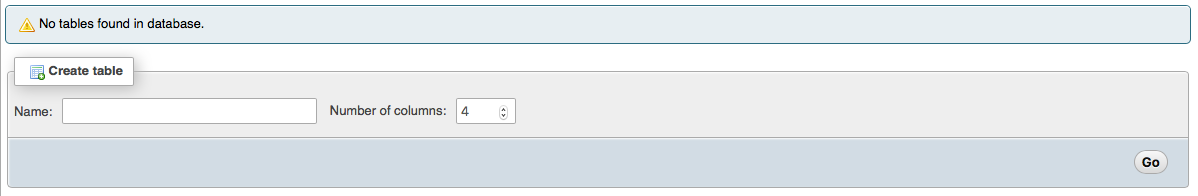
We’re finally ready to start creating a database!

(15) Click on “New” at the top of the left side bar, and you will see something like this:



(16) Type in a database name of your choosing (I will use “Tutorial”) and click on “Create”.

You will then see this:

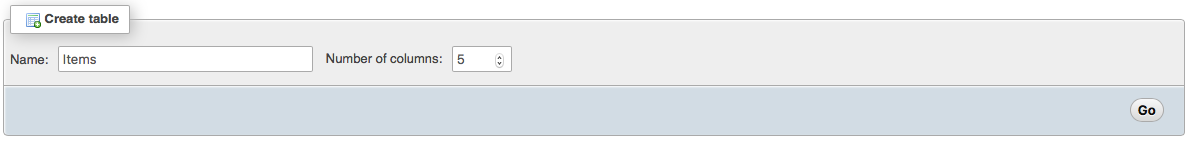


(17) Now, suppose that, in our database, we want to store some items and their colors –

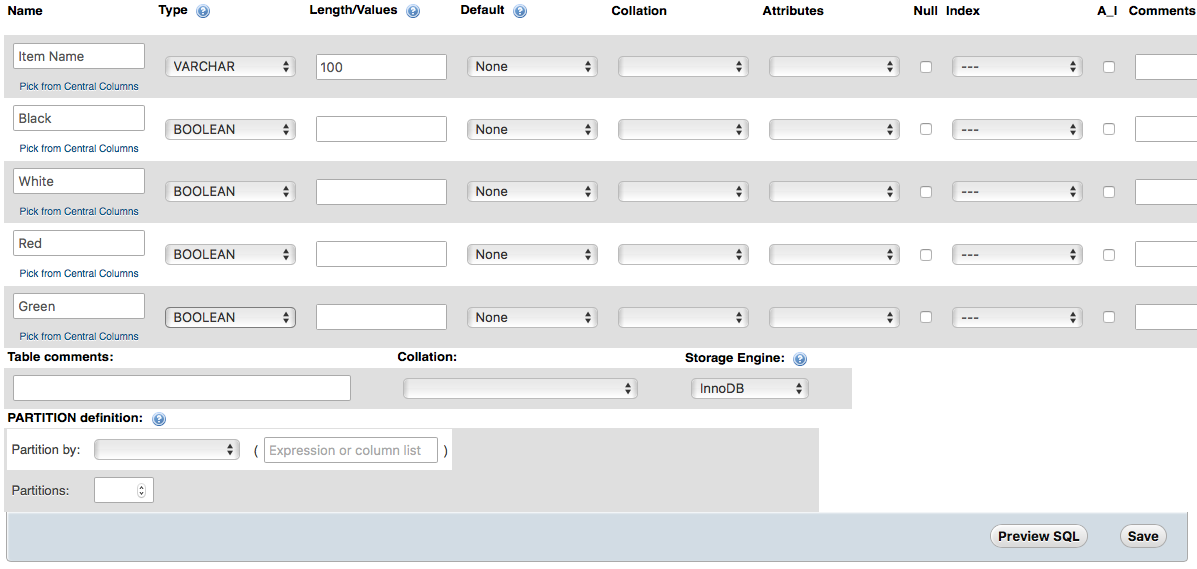
whether they are black, white, red, green, or some of the above, or all of the above. For

this purpose, I will create a tabled named “Items”, and give it five columns. You create

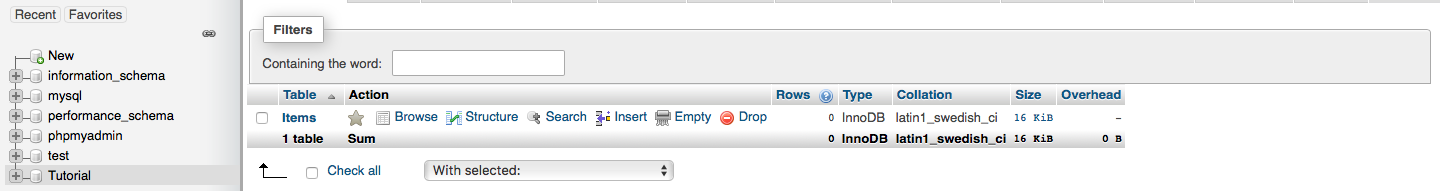
any table of your choosing. Once you have entered the name and the number of columns you want, click on “Go”.



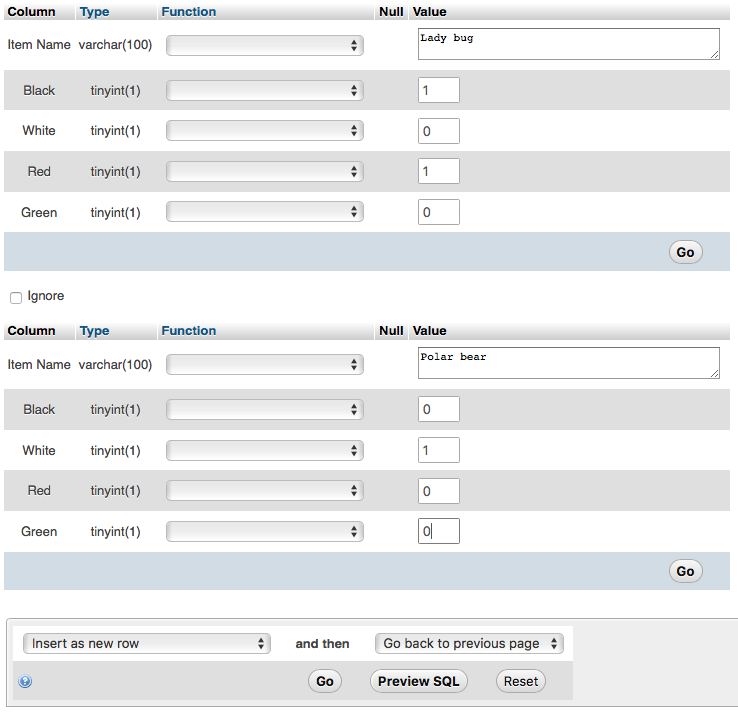
(18) Now, you will have the chance to set up your table. When you’re ready, click on “Save”. Here is how I set up mine:



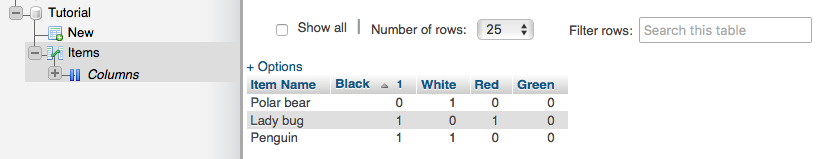
(19) In order to add entries into our database, click on the database name in the left sidebar (mine is “Tutorial”), and click on “Insert” under “Action” at the center of the page.



(20) Add each item to the form, and click “Go” at the bottom of the page when you’re done. Here is an example:



(21) Now, click on “Items” in the left sidebar, and you should see a table with items we entered:



We did it! Hooray!!!

**3 Connecting Qt to a database**

So God knows how many hours I have spent looking at tutorials and forums trying to figure this out, but it just *would not work*.

The code that various tutorials and forums said to put in **main.cpp** are pretty much all the same, and look something like this:

|  |
| --- |
| // creating a database connection  QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");  db.setHostName("localhost"); |
| db.setUserName("root"); // default mysql username for xampp is 'root' |
| db.setPassword(""); // xampp mysql has no password in default |
| db.setDatabaseName("qtdatabase"); |
|  |
| // lets test the connection |
| if(db.open()){ |
| cout << "Database connected" << endl; |
| } |
| else{ |
| cout << "Database connect failed" << endl; |
| } |

It never works.

Following tutorials and forums, I have tried to

(1) Copy and paste the folder “sqldrivers” in …/Qt/5.11.0/clang\_64/plugins into the debug

location of my Qt project. It didn’t work.

(2) Download MySQL Connector/C++ (download link here:

https://dev.mysql.com/downloads/connector/cpp/) and add the library files in my Qt

project. Since all the tutorials I could find were done on Windows OS, the library files are slightly different. Nonetheless, I tried including, one, some combination, or all of the library files in my project, and it DOES NOT WORK.

Moral of the story: Maybe I should try Python instead…