Project Template: Database

In this guide we will:

- Connect to a database
- Initialise information in the database
- Display data from the database
- Save form input into the database

Before we start

The current project is split into 4 different types of files.

- Python files (flaskblog.py, forms.py)
    - Creates the instance of flask.
    - Stores the data to be displayed.
    - Performs routing.
    - Creates form functionality.
    - **Saves form input to the database**
- HTML files (blog.html, home.html, layout.html and register.html)
    - Handles the webpage's structure and contents.
    - Creates form elements.
- CSS files (main.css)
    - Handles the webpage's styling.
- SQL files (database.sql)
    - **Used for writing SQL queries**

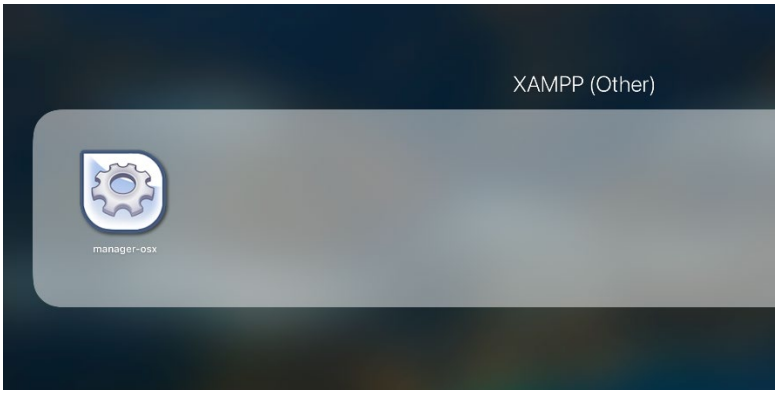
Run the code in '**flaskblog.py'** and then make sure you can view both the web page and the code.

Project Setup

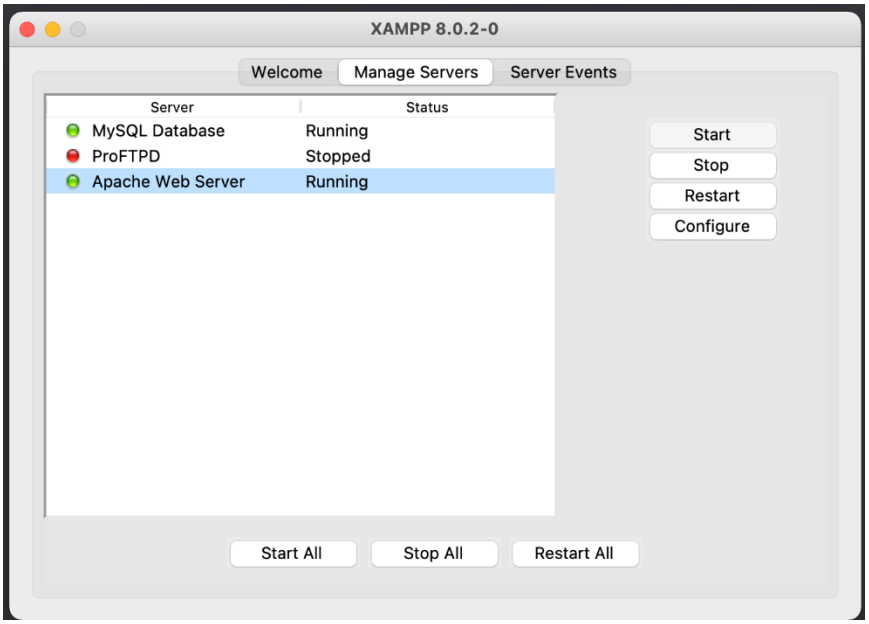
There are **two options** to run the project. Option 1 uses your own computer, so you'll need to install mysql and phpMyAdmin on your machine Option 1 is good for developing your model e.g. your project. Option 2 uses the currently installed software on your own zone. You can use it to host your developed project (e.g. develop an app on your own system and then move it to your zone for display):

1. **Option 1**: Installing XAMPP on your **local machine**. This XAMPP will help you to manage your MySQL data using PHPMyadmin (I really recommend you to do this, so you can do more experiments using MySQL database on your local machine)
    a. To download XAMPP: Windows https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/8.0.2/

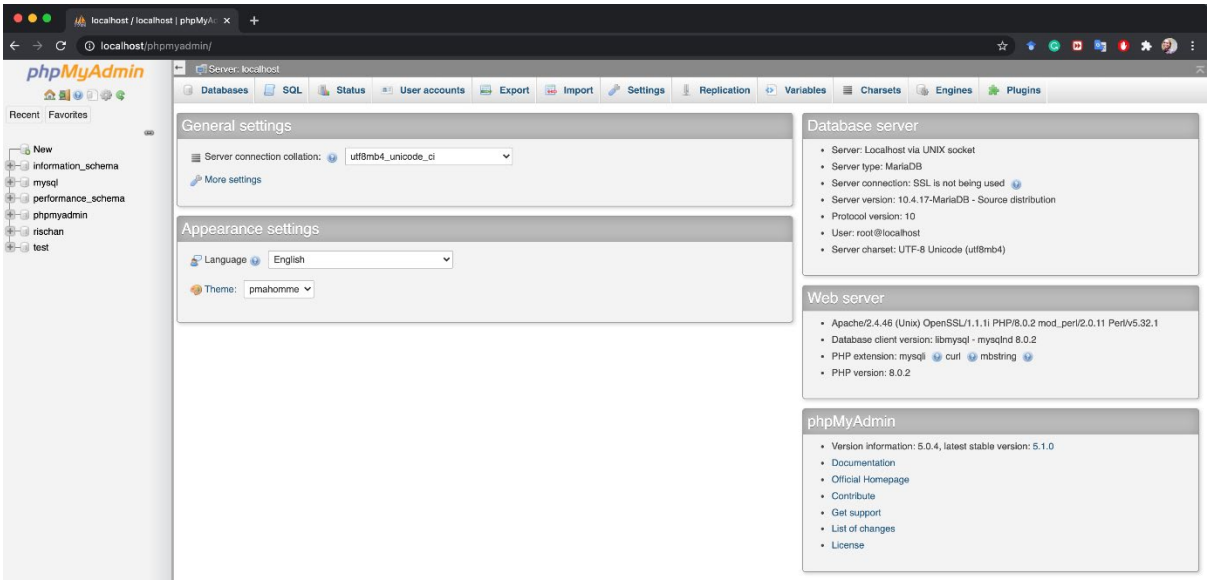
b. MAC OS **[DOWNLOAD INSTALLER NOT VM]:**
https://sourceforge.net/projects/xampp/files/XAMPP%20Mac%20OS%20X/8.0.2/



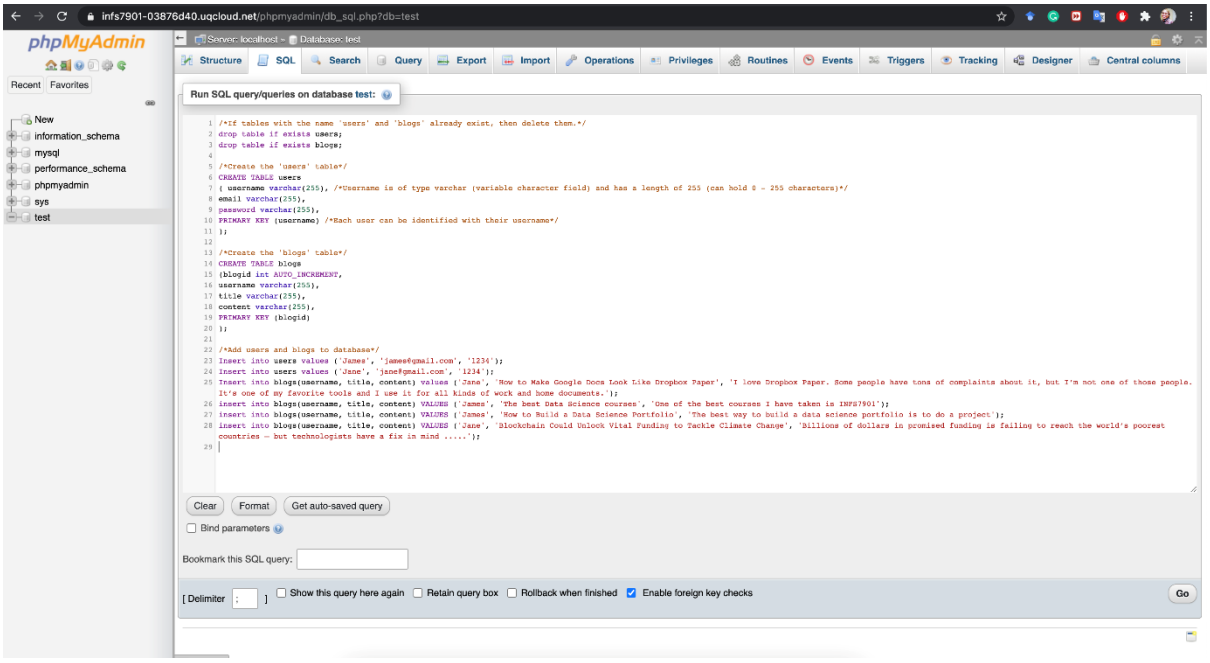
After done installation, you need to run XAMPP and run MYSQL and Apache Web Service



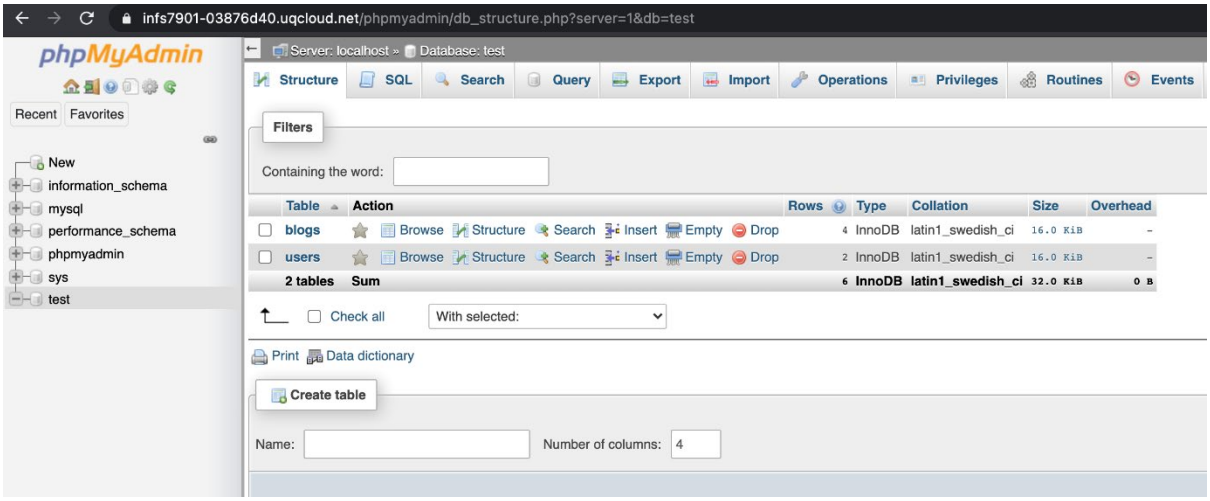
You can go to your browser and open url: http://localhost/phpmyadmin



2. **Option 2**: Using UQ zone
   Go to https://infs7901-YOURZONENAME.uqcloud.net/phpmyadmin
   Create a new user, grant all privilege, and create a new database.
   Click your new database, go to SQL, open database.sql using your editor then copy and paste to your SQL phpmyadmin



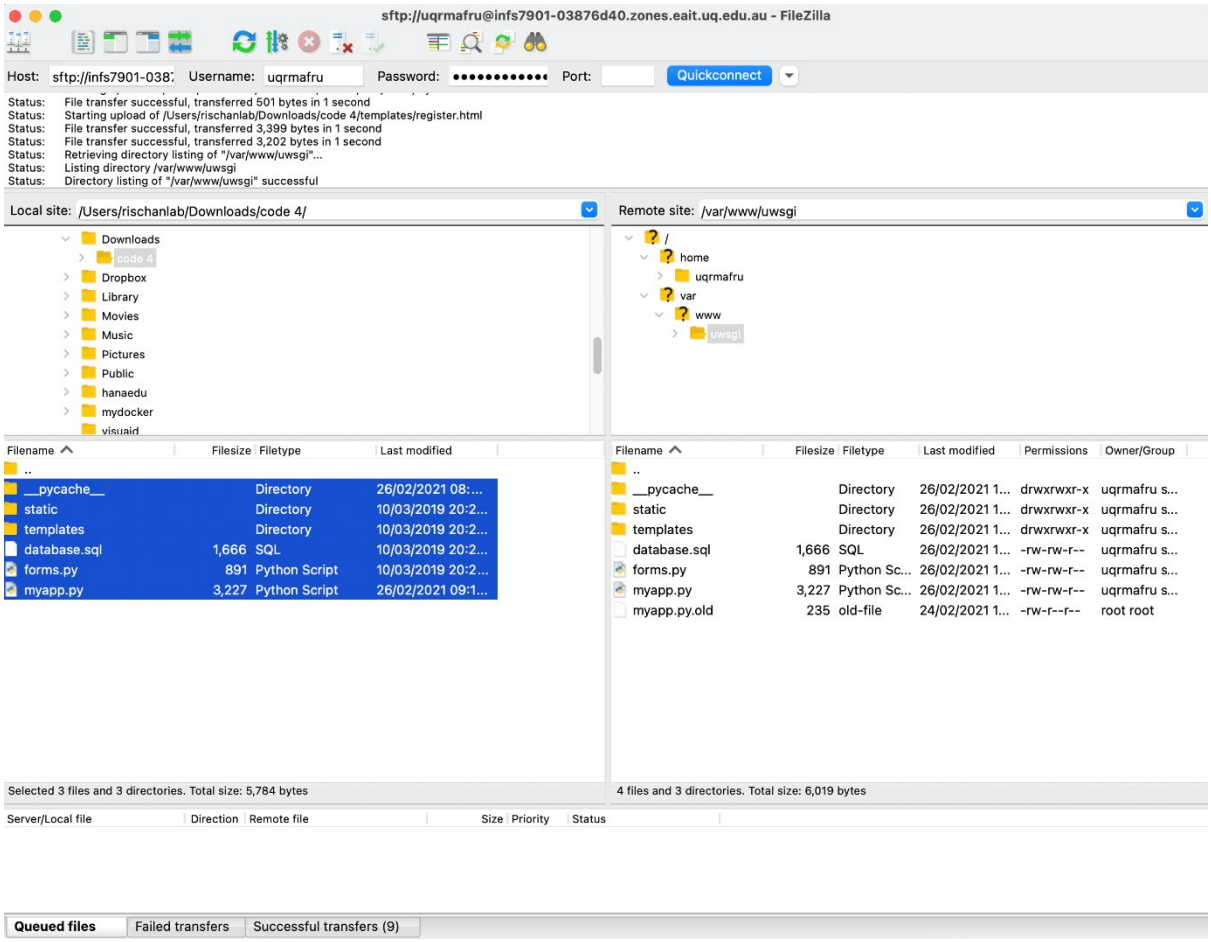
Now you should have two tables (users and blogs) in your database



Change your MySQL setting in the `flaskblog.py` using your MySQL username, password, and database name

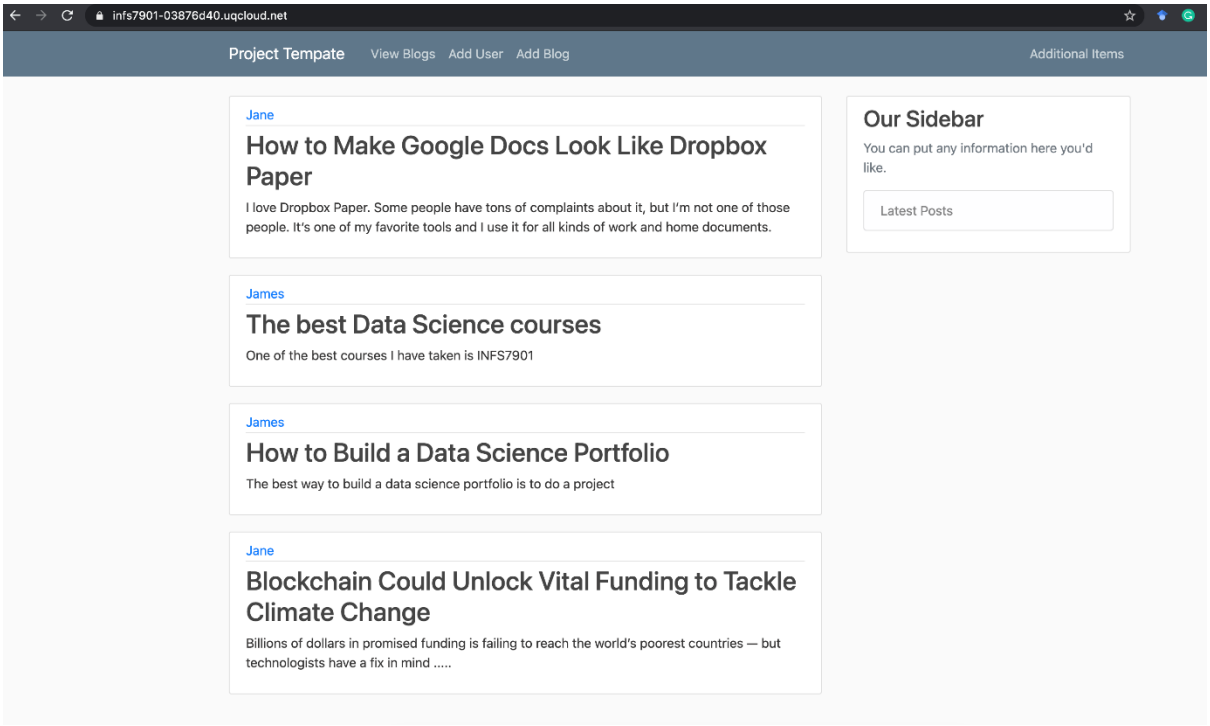
```
conn = pymysql.connect(host='localhost',
                       user='rischan',
                       password='mafrur',
                       database='test',
                       charset='utf8mb4',
                       cursorclass=pymysql.cursors.DictCursor
                       )
```

Copy myrpject from your local machine to your UQ zone using Filezilla.



RENAME `flaskblog.py` to myapp.py

SSH to your UQ zone, go to uwsgi directory (cd uwsgi), check whether your code have been copied correctly to your zone (learn basic linux commands: cd (go into a directory), ls (list directory), mv (rename)). Finally, run **sudo systemctl restart uwsgi.service** to take effect your changes. Now you can check your live website: https://infs7901-YOURZONENAME.uqcloud.net/


infs7901-03876d40.uqcloud.net
Project Tempate
View Blogs
Add User
Add Blog
Additional Items
Jane
How to Make Google Docs Look Like Dropbox Paper
I love Dropbox Paper. Some people have tons of complaints about it, but I'm not one of those people. It's one of my favorite tools and I use it for all kinds of work and home documents.
James
The best Data Science courses
One of the best courses I have taken is INFS7901
James
How to Build a Data Science Portfolio
The best way to build a data science portfolio is to do a project
Jane
Blockchain Could Unlock Vital Funding to Tackle Climate Change
Billions of dollars in promised funding is failing to reach the world's poorest countries — but technologists have a fix in mind .....
Our Sidebar
You can put any information here you'd like.
Latest Posts

Step 1: Connect to a database

In this example we will be using MySQL. The key line to connect to the MySQL database is the following code in **flasblog.py:**

```
conn = pymysql.connect(host='localhost',
                       user='your_mysql_username',
                       password='your_mysql_password',
                       database='your_mysql_db',
                       charset='utf8mb4',
                       cursorclass=pymysql.cursors.DictCursor
                       )
```

The variable, **conn** (short for connection), can now be used when referring to the database.

Step 2: Initialise information in the database

Next let's look at the code in **database.sql**. Writing SQL queries is a concept that will be explored later in the course, so for now, focus on what the code is doing rather than how it's doing it.

Key Functionality:

- If tables with the name 'users' and 'blogs' already exist, delete them.
- Create tables 'users' and 'blogs'.
- Add users and blogs into the created tables.

Exercises

1. For each key piece of functionality, identify the relevant code in **database.sql.**
2. Consider why we first delete any tables with the names 'users' and 'blogs'?
3. Use the following link to learn about different variable types e.g. varchar:

   https://www.w3schools.com/sql/sql_datatypes.asp

   https://www.w3schools.com/sql/sql_autoincrement.asp

Step 3: Display data from the database

When you view the **home** page, you should see several posts are displayed. The information for each of these posts is stored in the database.

The code that display these posts is in the function, **home()**, in **flaskblog.py**.

```
if form.validate_on_submit():
    c = conn.cursor()

    #Add the new blog into the 'blogs' table
    query = 'insert into users VALUES (' + "'" + form.username.data + "',"
    + "'" + form.email.data + "'," + "'" + form.password.data + "'" + ')' #Build the query
    c.execute(query) #Execute the query
    conn.commit() #Commit the changes
```

Let's deconstruct the steps for accessing the database and displaying the blogs:

1. Connect to the database.
2. Change the database results into a dictionary format rather than a list / array.
3. Create a cursor object.
    - A cursor is like a temporary copy of the database where data can be retrieved or manipulated.
4. Execute a query to select all blogs.
5. Set a variable, **posts,** to be equal to all the selected blogs in a dictionary format.
    - In the **home.html**, **posts** will refer to this variable.

Exercises

1. Open the **Add Blog** page in your browser and identify the Username field dropdown.

   Now look at the function **blog() in** flaskblog.py and identify the code which displays this username field.

   Compare this code to the code for displaying the blogs. What are the similarities and differences?

Step 4: Save form input into the database

To save data from a form input into the database we need to adjust the form code in **flaskblog.py.**

The following code snippet is in the function, **register().**

```
def home():

    conn.row_factory = dict_factory
    c = conn.cursor()
    c.execute("SELECT * FROM blogs")
    posts = c.fetchall()
```

Let's deconstruct the steps for saving a new user.

1. Connect to the database.
2. Create a cursor object.
3. Create an SQL query (in the form of a string) to create a new entry in the **users** table. Notice that within the query are the form fields.
4. Run the query and commit the changes.

Exercises

1. Compare this code to the code for saving a new blog (in **blog()**) . What are the similarities and differences?