ISAT 340 – Software Development
Dr. Anthony A. Teate
Fall 2016

Data Driven Web Application
Mini-Project & Final Exam

ISAT 340 – Software Development
HTML, SQLITE and FLASK
Data Driven Web Applications
(Mini-Project & Final Exam)

Due Date: ____Wednesday - December 14, 2016_________

Objectives:

- To develop a data-driven web application using HTML, Python FLASK and SQLITE3.
- To create a database and tables that drive the website
- To create and use HTML Forms with a variety of controls on a login and registration form.
- To read from and write to a sqlite database using a web page
- To load data and images from a remote server into the web site
- To become familiar with using Python in the FLASK micro-framework to process data on a data driven web page

Deliverables: (teams of two)

1. Demonstration of your completed and working web app
2. Submission of the application code and database to Canvas.
3. An e-copy of the answers to the attached worksheet

Problem Background

Many website applications require that users create an account that requires a login name and other valid information for re-entry into the website. In this mini-project we will create the simple forms for users to complete in order to log into the website and update their membership information which is stored in a database. We will also allow members to update other tables in the database in order to keep web content current.
COMMEMT: This project is designed to reinforce the steps we walked-through in class. Please thoroughly read it and carefully follow the steps.

Instructions:

In this mini-project we will create a data-driven website application that allows an authorized end-user to update content in a database that will be displayed on the website. The user(s) will also be required to fill in certain fields before the web form can be submitted and processed. The data that is entered will be displayed in another webpage with a simple (*but elegant*) HTML markup.

Pre-Project Preparation-Important!

Downloading Flask libraries and other code and preparing the project folder structure

Before you began any coding:

1. Create a folder called *FlaskApp* on your desktop.

2. Download the Python code, the HTML file (*login.htm*) and the flask libraries from Canvas and place them in your FlaskApp folder

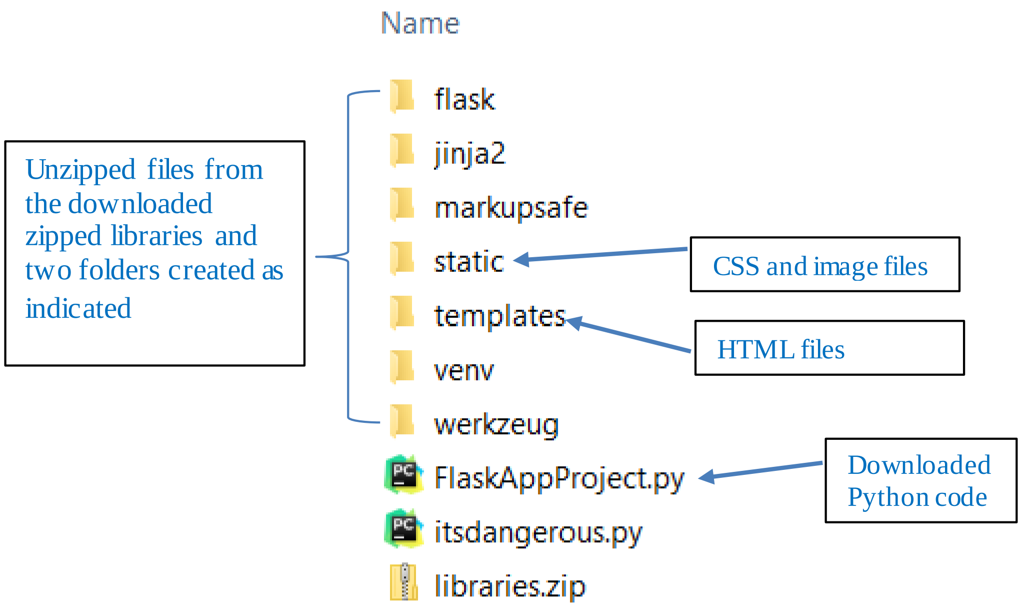
3. Unzip the flask libraries into the FlaskApp folder (*use 7-zip and click extract here*)

4. Create TWO more folders within the main FlaskApp folder called:

    a. templates
        i. *Place the login.htm file in this folder*
    b. static
        i. You will place images and CSS files in this folder

5. The structure of files in this folder is important so be sure your folder structure is as indicated in the instructions that follow (see below).

At this point, the files in you FlaskApp folder should look like this:



Part I- The SQLite Database

Step 1: You are to begin this project by using the python code we developed in class (appropriately modified) to create a sqlite3 database with the following properties:

- The name of the database should be celebrities.db. Save it in the above folder.
- Now using your other python code (appropriately modified), create TWO tables in the database with the following names and fields:
    - A table called **"celebs"** with seven (7) fields as indicated below:
        - celebID, firstname, lastname, age, email, photo and bio. All fields are of data type 'text' except the age field which is of type 'integer'. Make the celebID field the PRIMARY key with data type integer.
    - A table called **"members"** with six (6) fields as indicated below:
        - memberID, firstname, lastname, age, email, and bio. All fields are of data type 'text' except the age field which is of type 'integer'. Make the memberID field the PRIMARY key with data type integer.

Step 2: Use Python to INSERT INTO the **celebs table** the data shown below. You must insert the exact data shown **<u>but</u>** youmay search the web and find any short **<u>realistic</u>** sentences to enter for the bio values (*no more than five sentences per celebrity*):

celebID	**firstname**	**lastname**	**age**	**email**	**photo**	**bio**
1	Angelina	Jolie	40	angie@hollywood.us	http://www.nphinity.com/pics/aj.jpg	
2	Brad	Pitt	51	brad@hollywood.us	http://www.nphinity.com/pics/bp.jpg	
3	Snow	White	21	sw@disney.org	http://www.nphinity.com/pics/sw.jpg	
4	Darth	Vader	29	dv@darkside.me	http://www.nphinity.com/pics/dv.jpg	
5	Taylor	Swift	25	ts@1989.us	http://www.nphinity.com/pics/ts.jpg	
6	Beyonce	Knowles	34	beyonce@jayz.me	http://www.nphinity.com/pics/bk.jpg	
7	Selena	Gomez	23	selena@hollywood.us	http://www.nphinity.com/pics/sg.jpg	
8	Stephen	Curry	27	steph@golden.bb	http://www.nphinity.com/pics/sc.jpg	

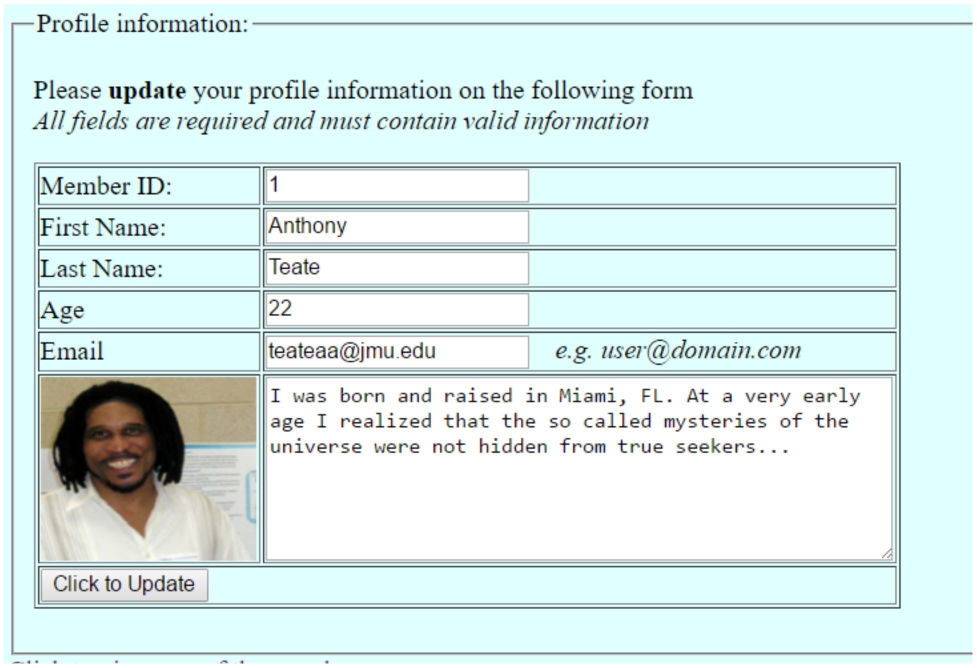
Step 3: Use Python to INSERT INTO the **members table** your personal information. This information should be exactly the information that you entered HTML Forms assignment. *Please note that there is no field in the table for a photo URL but a field called memberID has been added!*

Step 2: Use your Python code from the earlier labs to retrieve the data you wrote to the database in order to make sure that it was correctly inserted.

Part II-a – Modifying the layout of your HTML form

Step 1: You created an HTML page that was a simple profile information page.

- Make a copy of the html file and rename it "*profile.htm*". (NOTE: the code we will write (later) uses the .htm file extension but .html would also work if used consistently!)
- You need to modifiy the layout of your form so that it looks similar to mine (see below). Note that I have updated the text and have also removed the field and label that referred to the photo URL.



Profile information:

Please **update** your profile information on the following form
All fields are required and must contain valid information

Member ID:	1	
First Name:	Anthony	
Last Name:	Teate	
Age	22	
Email	teateaa@jmu.edu	*e.g. user@domain.com*
	I was born and raised in Miami, FL. At a very early age I realized that the so called mysteries of the universe were not hidden from true seekers...	
Click to Update		

- You should take a headshot of yourself and save it as a jpeg to be loaded into the image tag. My image tag is in a <td>…</td> cell and is:
  - <img src="teate.jpg" width="130" height="110" >
- Later, we will modify this page and will also change the location of the image file. For now, it is important that you just get your layout completed and browse it to make sure it displays properly.

Part II-b – Enabling your HTML form to receive data from the database

- Now, place the *profile.htm* file you created above into the ***templates*** *folder*
- Also, place the jpeg headshot image of yourself into the ***static*** *folder*

We will now modify the profile.htm file so it can receive data from the database.

Step 1: **Use Notepad++ to modify your *profile.htm* by entering the following markup**.

NOTE: Below, I am *only* showing a snippet of the HTML markup you will need. *You must complete the HTML file by entering the rest of the HTML 'code', contiinuing with the pattern I have started, until all of the fields in the members table of the database have been accounted for.*

Also, your HTML page needs to be complete with all tags required for it to properly render in the browser (<!doctype html>, <html>, <head>, <title> etc.). Don't forget the closing tags!

```
<!doctype html/>

<html>

<head>

<title> </title>
<style type="text/css">body { background-color: lightcyan;}</style>
</head>

<body>
<fieldset>
        <legend>Profile information:</legend>
<form action="" method="post">
<Table Border=1>

<p> Please <strong>update</strong> your profile information on the following form <br>
        <em>All fields are required and must contain valid information </em>
<p>
<tr>
        <td> Member ID:</td> <td><input type="text" name="memberID" value="{{memberID}}" readonly /></td>
</tr>

<tr>
        <td> First Name:</td> <td><input type="text" name="firstname" value="{{firstname}}"/></td>
</tr>
.
.
.
</html>
```

IMPORTANT:

- In Flask, the image tag which will display the picture of your headshot uses a special format. ALL images must be placed in the *static folder* and referenced as indicated below:

```
<img src="{{ url_for('static', filename = 'teate.jpg') }}" width="130"  height="110">
```

- Save the modified *profile.htm* webpage into the *templates folder*.
- We will further modify the *profile.htm* file later…

Statements in the *profile.htm* webpage like:

```
        <td><input  type="text"  name="firstname"  value="{{firstname}}"/></td>
```

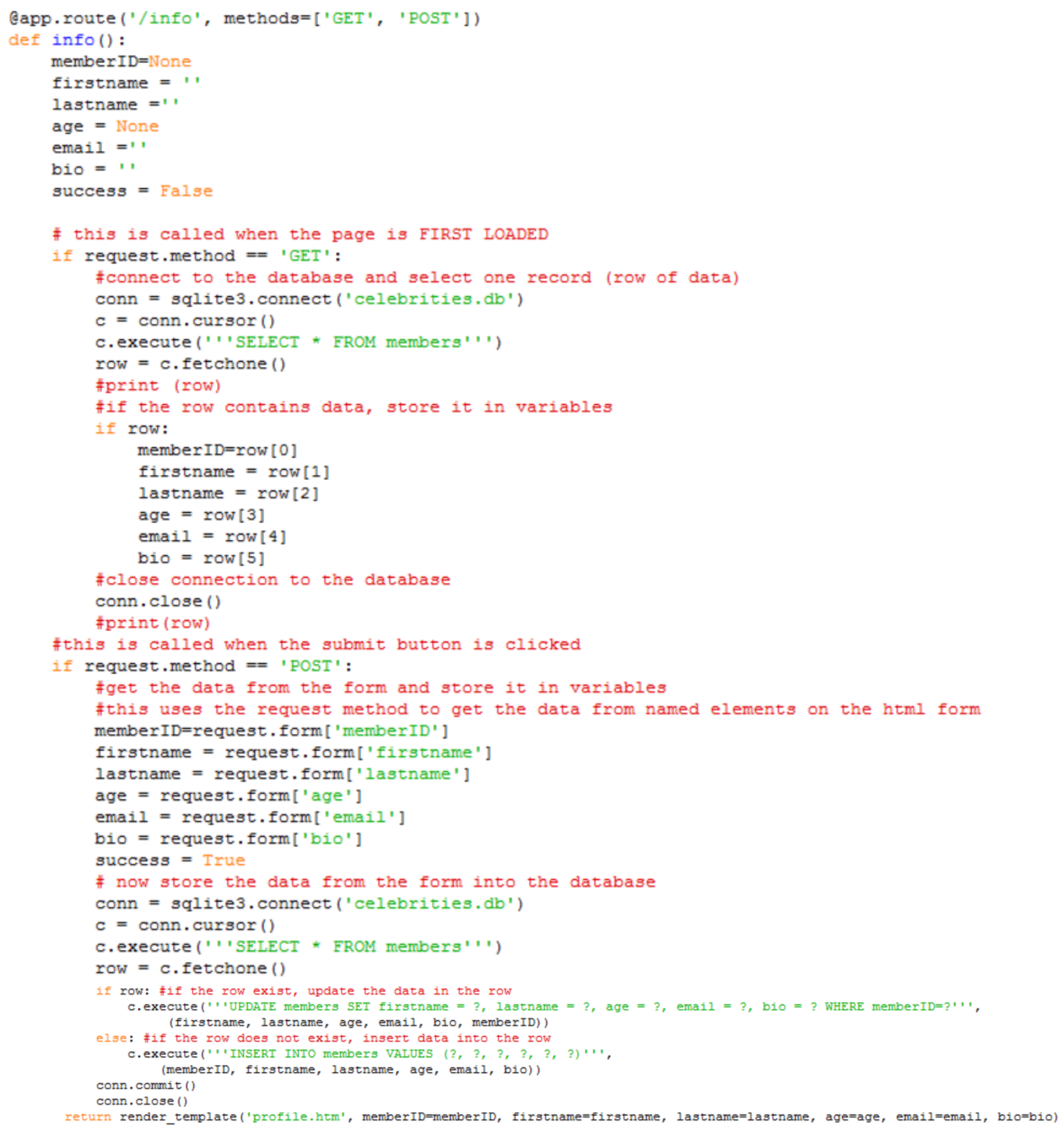
with the name of an input type (e.g., textbox) from the form enclosed in in double braces is flask's way of displaying information that is received **from** the database and sent **to** the *profile.htm after the user logs in*.

Part III – Python and Flask

Step 1: We will now create code to add to the python file that you downloaded. Before we do this, open and run the python file that you downloaded, *FlaskAppProject.py*. Point your browser to http://localhost:5000/ and you should see a welcome message and login page. *The user name is **admin** and the password is also **admin***. Try to log into the site by clicking the login button. It doesn't do anything yet (in fact it gives a server error!). Let's create code to enhance the website in python using flask to interface to the database and to HTML pages. Kill the server by closing IDLE.

I have extensively commented the code below so that you understand what is occurring when the code is executed. *You will need to create similar code to complete the project/final exam.* Using the IDLE editor, open the *FlaskAppProject.py* file and *carefully* enter the following python code. This code should be entered right after the statement:

```
return render_template('login.htm', error=error)
```



```
@app.route('/info', methods=['GET', 'POST'])
def info():
    memberID=None
    firstname = ''
    lastname =''
    age = None
    email =''
    bio = ''
    success = False

    # this is called when the page is FIRST LOADED
    if request.method == 'GET':
        #connect to the database and select one record (row of data)
        conn = sqlite3.connect('celebrities.db')
        c = conn.cursor()
        c.execute('''SELECT * FROM members''')
        row = c.fetchone()
        #print (row)
        #if the row contains data, store it in variables
        if row:
            memberID=row[0]
            firstname = row[1]
            lastname = row[2]
            age = row[3]
            email = row[4]
            bio = row[5]
        #close connection to the database
        conn.close()
        #print(row)
    #this is called when the submit button is clicked
    if request.method == 'POST':
        #get the data from the form and store it in variables
        #this uses the request method to get the data from named elements on the html form
        memberID=request.form['memberID']
        firstname = request.form['firstname']
        lastname = request.form['lastname']
        age = request.form['age']
        email = request.form['email']
        bio = request.form['bio']
        success = True
        # now store the data from the form into the database
        conn = sqlite3.connect('celebrities.db')
        c = conn.cursor()
        c.execute('''SELECT * FROM members''')
        row = c.fetchone()
        if row: #if the row exist, update the data in the row
            c.execute('''UPDATE members SET firstname = ?, lastname = ?, age = ?, email = ?, bio = ? WHERE memberID=?''',
                  (firstname, lastname, age, email, bio, memberID))
        else: #if the row does not exist, insert data into the row
            c.execute('''INSERT INTO members VALUES (?, ?, ?, ?, ?, ?)''',
                 (memberID, firstname, lastname, age, email, bio))
        conn.commit()
        conn.close()
    return render_template('profile.htm', memberID=memberID, firstname=firstname, lastname=lastname, age=age, email=email, bio=bio)
```

Step 2: Make sure all files are in their proper folders.

- Your *profile.htm* is in the *templates folder*
- Your image is in the *static folder*.
- Your python code and database is in the main *FlaskApp folder*

Step 3: Run your python code.

- Point your browser to http://localhost:5000/
- You should see the login page load. Enter the user name and password and click the *login* button.
    - IF there were no errors, you should now see your profile page.
- Change some of the information on your profile page and click the update button.
    - Again, if there are no errors, the page will "flicker" as the updated data is written to the database (We will enhance this later to give the user a message indicating the data was successfully written to the database)
    - **As a check:** Run your python code from the SQLite assignment to verify that your updated data was written to the members table in the database.
- If there are errors, debug your code until they are corrected and the profile page loads as expected when you browse it and it correctly updates the members table in the database.

If you have you have successfully reached this point, take a break and review the project and make sure you understand the python code and the html and how they are interfacing to the database.

- This is the HALFWAY POINT.
- The remaining parts of the project will be posted on canvas shortly.