ISAT 340 Mini-Project and Final Exam (Phase III)

FULL-STACK DATA DRIVEN WEB APPLICATION

Flask | Python | SQLite | HTML | CSS |GitHub |AWS

REMAINING TASKS = 60 POINTS

WELCOME BACK! THIS IS THE FINAL PHASE OF THE PROJECT!

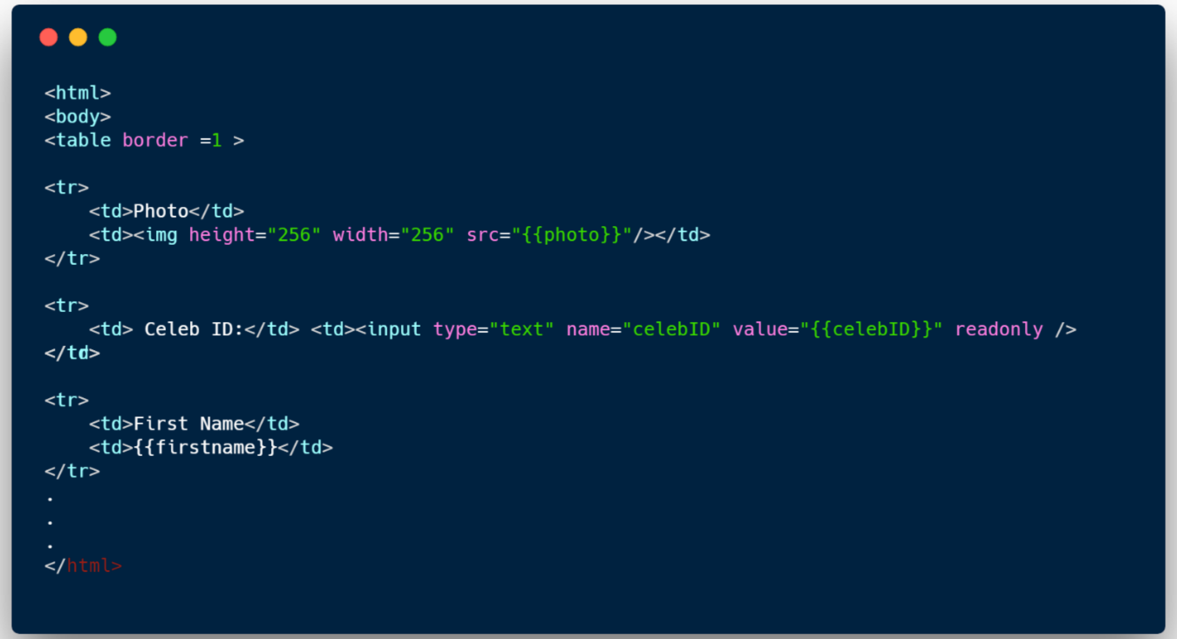
AT THIS POINT, YOU SHOULD HAVE SUCCESSFULLY BEEN ABLE TO CONNCT TO THE SERVER AND PULL ALL OF THE ASSOCIATED IMAGES FROM IT THAT COMPLEMENT THE DATA IN YOUR DATABASE.

IN THE PREVIOUS TASKS I GAVE YOU A LOT OF PYTHON CODE AND HTML MARKUP IN ORDER TO SUCCCEFULLY ACCOMPLISH THOSE TASKS. IN THE REMAINING TASKS YOU MAY FIND THAT PYTHON CODE AND HTML MARKUP USEFUL…

THE REMAINING TASKS ARE:

1. (5pts) **view one celebrity page.** Create an HTML page called view_one_celeb.htm and enter the html markup that will display the information about *one of the celebrities*.

You should be able to use code from your previous task that allowed you to view all of the celebrities in the database(noting that for this task your html code does not have to loop over the entire table in the DB). As before, 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 **celebs table** of the database have been accounted for.



```
<html>
<body>
<table border =1 >

<tr>
    <td>Photo</td>
    <td><img height="256" width="256" src="{{photo}}"/></td>
</tr>

<tr>
    <td> Celeb ID:</td> <td><input type="text" name="celebID" value="{{celebID}}" readonly />
</td>

<tr>
    <td>First Name</td>
    <td>{{firstname}}</td>
</tr>
.
.
.
</html>
```

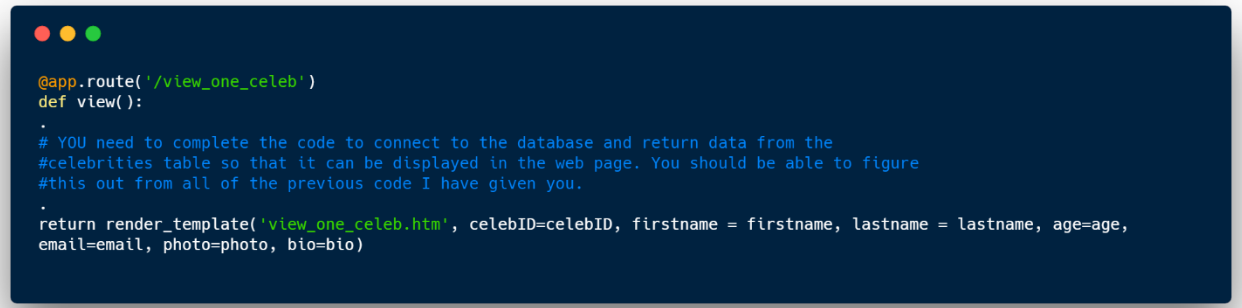
IMPORTANT:

- When you have completed entering the html, save the page with the name *view_one_celeb.htm* and make sure to put it into the *templates folder*.

COMMENT: You will notice (if you got this working!) that it always shows the first celebrity in the celebs table from the database. See BONUS below that asks you to modify your SQL code so that it displays any celebrity from the database that one randomly picks.

The Python Code:

A very small snippet of the beginning and ending part of the corresponding python code would look something like (NOTE THE COMMENTS IN THE SNIPPET OF YOUR TASK):



```
@app.route('/view_one_celeb')
def view():
.
# YOU need to complete the code to connect to the database and return data from the
#celebrities table so that it can be displayed in the web page. You should be able to figure
#this out from all of the previous code I have given you.
.
return render_template('view_one_celeb.htm', celebID=celebID, firstname = firstname, lastname = lastname, age=age,
email=email, photo=photo, bio=bio)
```

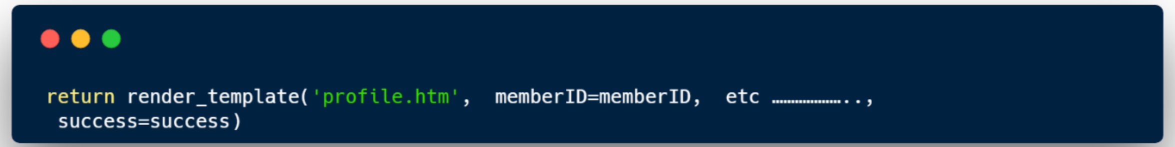
REMINDER(IMPORTANT):

- Don't forget to modify your *profile.htm* webpage and add a **second link** to the page so that when it is clicked it will take you to the *view_one_celeb.htm. Also, don't forget to enhance the HTML view_one_celeb page with CSS to make it look professional!*

2. (5 points) **Modification of the *profile.htm* page.** It would be nice to let the user know that their updated profile information has been submitted to the database by displaying a message after they click the submit button. One way to do this is to add a **Boolean variable** to your python code and set it equal to *False*. Then find the place in the python code that is inserting data to the database and set the variable to *True* when that part of the code executes (NOTE: In my code, I called the Boolean variable *success,* but you can

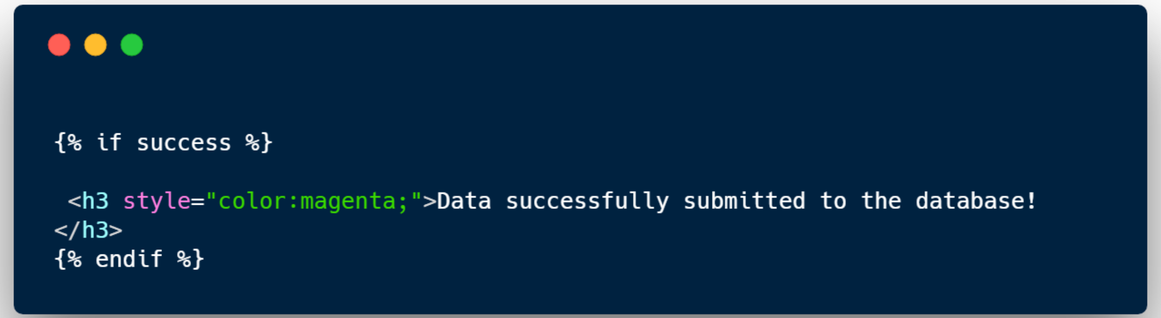
call it anything you want to such as *posted* so long as you use the same variable name everywhere you see it used below)

   a. You then need to pass this value and variable to the *profile.htm* page by modifying the appropriate python code **render_template** so that *it passes this **additional** Boolean variable and its value.*



```
return render_template('profile.htm',  memberID=memberID,  etc ………………..,
 success=success)
```

   b. You also need to insert the code below into your profile.htm page



```
{% if success %}

 <h3 style="color:magenta;">Data successfully submitted to the database!
</h3>
{% endif %}
```

3. (10 points total) **Enhance the *login* security of the website** Examine the FlaskAppProject.py file using Notepad++. As you have seen earlier, the username ('admin') and the password('admin') have been hardcoded into the python code. This is bad! We want to increase the security of the site by having the username and password of all members stored in the database. You will do this based on your knowledge of reading data from the database as you have done earlier. Proceed by:
   a. Creating a new table in the celebrities database and call it *member_login* with three (3) fields as indicated below:
      i. memberID, username, password. All fields are of data type 'text' except the memberID field. Make the memberID field the PRIMARY key with data type integer.
   b. Use python to insert two rows of data not the table for your and your teammate (i.e., make up a username, and password for each of you. Your memberID should

be the same as that in the members table, either “1” or “2” since there are only two persons per team).

You should be able to use python code form your earlier work pull this data from the table and compare the username and password from the from to those retrieved from the database. If they match, allow the member to proceed to the profile page. If not, give an error message saying the credentials are incorrect.

COMMENT: You will notice (if you got this working!) that it always shows only one of the profile pages located in the templates folder. You will modify this in Task 6 below

4. (5 points total) **Enhance the appearance of the *login.htm* webpage.** Examine the login.htm web page using Notepad++. You will notice that I have commented out a link to a non-existent CSS page (I called it ***project_stylesheet.css***). Use your knowledge of HTML and CSS to spruce up the login web page!
   a. Create a simple CSS page (with the given name) so that the login page looks more welcoming and professional. *To receive full credit, all styling (fonts, colors, etc.) MUST be done in the CSS file*.
   b. REMEMBER to un-comment the link I put in the login page and save the CSS file to the **static folder**.

5. (10 points). **Enhance the appearance of the *view_all_celebs.htm* webpage.** The display of the data and images using the current HTML in the the *view_all_celebs.htm* HTML page is boring and bland. The pictures are too big; some columns are too small; some columns are too wide; the layout could be improved; and the fonts, text, etc. could look better.
   a. Use your knowledge of HTML and CSS to spruce up the web page so it looks more professional when presenting the data.
   b. This CSS page should also be stored in the static folder and given a different name than the one previously used in task (3)

6. (5 points total) **Modification of the *login.htm* webpage to allow different member’s profile page to be displayed.** Examine the login.htm web page using Notepad++. You will note that the page only requests the username and password from the person that is attempting to log in to the site.
   a. Enhance the code in the *login.htm* page so that it also *requests* the memberID.
   b. We then want to use this information along with the username and password to:
      i. Load the proper profile page of the member trying to log into the website.

There are several ways that this can be accomplished but all of them will involve some usage of the require that the code that “calls” the profile.htm page

```
return render_template('profile.htm', memberID=memberID, firstname=firstname, lastname=lastname,
age=age,                         email=email, bio=bio, success=success)
```

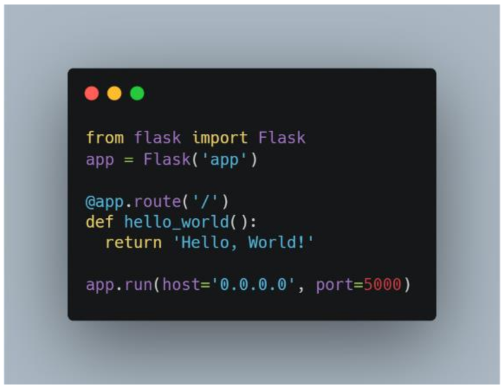
and using the memberID variable. The code may also involve some simple decision structure.
HINT:

```
if you want to pass a value from a template in a url, you can use a query parameter:

<a href="{{ url_for('data_page', my_var='my_value') }}">Send my_value</a>
will produce the url:

/data_page?my_var=my_value
which can be read from data_page:

@app.route('/data_page')
def data_page():
    my_var = request.args.get('my_var', None)
```

7. (10 points). **Deployment to GitHub**. One (or both) of the team members should get a GitHub account and create a ***public*** repository and name it *isat340_miniproject*. You should learn how to push all the code and folders in your project to the repository. NOTE: *In addition to your code, the files you push must include the complete flask framework files including all necessary files and folder to run your app (sqlite database, templates folder, static folder, all html files and image files, etc.).*
    a. I **MUST** be able to browse to your GitHub folder, download (or clone) your files/folders and run them on my local machine without them throwing an exception (error). *If an error occurs, then you get zero points for this part.*
    b. ***I will expect you to have completely 'pushed' your final project to GitHub before the final exam start time and NOT touch it again until after the exam time is over.***
    c. I have put links below to three tutorial files on that will help you get up to speed with GitHub and Git and you can use any other resources you want in order to learn how to 'store' projects on GitHub. **Don't underestimate the importance of what I am trying to do here!** *This is an invaluable skill that you should be able to put on your resume AND use as part of your portfolio after you know how to manage your projects and their versions on GitHub*!

    d. https://guides.github.com/activities/hello-world/
    e. *https://realpython.com/python-git-github-intro/*
    f. *http://rogerdudler.github.io/git-guide/*
    g. *https://guides.github.com/activities/hello-world/*

8. (10 points). **Deployment to AWS**. In the real world, you would deploy your web app to a server and make it available to users via a browser. So, the final phase of this project will consist of you deploying your Flask app to a web server and making it available for me to browse and use. The app on the server must function as expected (showing the celebrities, allowing updates to the sqlite database via the web form, etc.). During the exam, in addition to being graded on the above tasks, I must be able to browse your deployed app and use it to update you profile information on the profile page. This means that you must supply me with:
   a. The address of your server
   b. The login credentials to your celebrities app (username and password)
   c. *I will browse your website and click through the various links and grade you on parts 1-6 above. If I can successfully 'use' your app, you get full credit for this part.*



```python
from flask import Flask
app = Flask('app')

@app.route('/')
def hello_world():
  return 'Hello, World!'

app.run(host='0.0.0.0', port=5000)
```

Simple code to run on server to test if it can be viewed on port 5000 of your amazon web server instance. Note the address host ='0.0.0.0'. This makes the app visible to the outside world when they try and browse it at your server's address on port 5000

Again, deploying a micro web framework app (flask) to a server is an invaluable skill that you should be able to put on your resume AND use as part of your portfolio after you know how create an AWS Instance and deploy your flask project on it.

https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/EC2_GetStarted.html

Bonus (5 points)

Use the html **select element** to allow a user to arbitrary pick and display any celebrity info page from the database using the dropdown menu.

SUMMARY

There is much, much more that we could do with this mini-project such as:

- searching for members;
- displaying a specific member based on their email address or some other unique identifier;
- doing some statistics such as the average age of the members,
- etc.;

but since we are out of time, I hope that what we have done gives you an idea of the power of dynamic data-driven websites!

Reminder- for the final exam:

- Your complete project with the correct file structure **must** be posted on github in a public repo!
- you **must** supply me with the github address to the repo
- your complete project **must** be deployed to an amazon server
- your **must** supply me with *both the address to the server (flask website) and the login credentials* so that i can test your website.
- good luck!