

Fundamentals of Database Technologies

Coursework Assignment 5

Date: 25 September 2020

Name: Gopal Juneja

CID: 01246827

Introduction:

The aim of this assignment was to gain hands on approach into web application development and learn about the various tools that are used to optimize processes in the development lifecycle. Keeping simplicity in mind, the task at hand was to use Python's Flask server to create the application and using SQLAlchemy to implement object relational mapping on the module database to extract data using Python. The final task was about deploying the application on Heroku. Version control was an important part of this assignment, as it helped gain thorough understanding of making repositories on GitHub as well as linking it to the IDE used and Heroku.

Object-relational Mapping & its Importance:

In essence, object-relational mapping (ORM) is a programming technique to write database queries in the form of the object-oriented paradigm of a programming language. ORM is used to bridge the gap between relational databases and object-oriented programming languages by converting data between incompatible type systems. The importance of ORM lies in its capability to abstract database querying, so that the code could be changed around the convenience of the programmer. Moreover, ORM forces programmers to follow good programming practices such as reducing unnecessary redundancies in code (Don't Repeat Yourself (DRY) principle).^[1]

For instance, there exists a database table called Humans, which stores information regarding names, phone numbers, email addresses, home address and genders of all the users.

The following SQL statement queries information about all male users from Humans^[1]:

```
SELECT * FROM Humans WHERE gender == 'Male';
```

Using an ORM, it could be implemented in an object-programming language as:

```
var orm = require('orm-library');
```

```
var humans = orm("Humans").where({gender: 'Male'});
```

One way this could be seen is through the Model-View-Controller software pattern where programmers would map the model into a code (e.g. Python) to represent the task at hand. The ORM would map this model to a database by automatically generating relational database queries (e.g. PostgreSQL), to update the database or extract information based on the objects of the model being fed in^[2]. The outputs, if any can be seen by users through the View.

The Flask Server:

Flask is a lightweight WSGI web application framework of Python, developed by Armin Ronacher on April 1, 2010^[3].

It lacks the need for programmers to be verbose in web application development (minimalistic boilerplate code) and is suitable for beginners to learn about web development frameworks. Some of its features include a built-in development server, flexibility in terms of lightweight and modular design, URL routing, Jinja2 Templating (used to create markup formats(e.g.HTML) that are presented to users through HTTP requests), development server and debugger etc.^[4]

Originally, Flask was supposed to be nothing more than a joke, towards another web framework called Bottle, which is another simple WSGI micro web-framework for Python. Ronacher named his framework Flask just to prove he could make a better version of Bottle^[5]. Popular applications of Flask include Reddit, Trivago, LinkedIn and Pinterest.

The SQLAlchemy ORM (Python library):

SQLAlchemy is an open-source toolkit and object-relational mapper for Python, written by Michael Bayer, and released on February 14, 2006.

It supports a variety of databases like PostgreSQL, SQLite, Firebird, Oracle, etc. in implementing “unit of work” pattern mapping complex SQL queries to objects.^[6]

SQLAlchemy can be divided into two parts^[6]:

1. The Core:
 - a. SQL abstraction toolkit which has Database API implementations, and SQL expression language.
2. Object Relational Mapper (ORM):
 - a. It allows object-oriented classes to manipulate databases in multiple ways.
 - b. The “unit of work” system aids the ORM in sorting operations that manipulate the database such as insert/update/delete, by providing a “topological dependency” to reduce deadlocks (two transactions blocking each other by retaining resources the other transaction needs).

Popular open-source projects for SQLAlchemy are Flask SQLAlchemy, which adds SQLAlchemy support to Flask. It is also used as a database toolkit for Go in the form of Qb.^[7]

Heroku Hosting Platform:

Founded by James Lindenbaum, Adam Wiggins and Orion Henry in 2007, Heroku is an application of Cloud Computing, which allows developers to build and operate applications entirely in the cloud. Falling under the umbrella of Platform as a Service (PaaS), it's integrated data services for running applications are based on a managed container system. Heroku supports application in Ruby, Node.js, Java, Python, Clojure, Scala, Go and PHP. ^[8]

Some of its main features include app-centric delivery approach, eliminating the need to ponder about infrastructure or server requirements. The applications built by users of Heroku are run in containers called dynos, which provide a runtime environment for code written in the aforementioned languages. Heroku also allows its users to extend its application by facilitating build packs, to launch applications with buttons instead of the command line or terminal.

Some popular companies that use Heroku are Accenture, StackShare, Craftbase etc. ^[9]

The name "Heroku" being a combination of "heroic" and "haiku", is an acknowledgement to Yukihiro "Matz" Matsumoto, the designer of Ruby. The platform was initially developed for supporting projects that were compatible with Ruby programming platform called Rack.

Git Version Control Tool

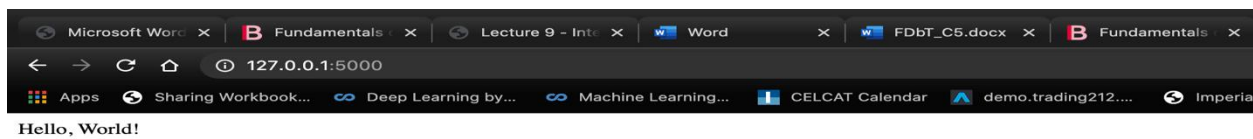
Git is an open source distributed version control system designed to keep record of changes in projects of any scales. Created by Linus Torvalds in 2005 for the development of Linux Kernel, Git was born due to the conclusion of the relationship between the community that developed Linux kernel and the commercial company that developed BitKeeper(distributed version control system) as the company started charging users of BitKeeper(<https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>). Git could be seen as a content tracker usually for programming content, because of its speed in terms of non-linear development, simplistic design and fully distributed nature. Git works in the form of two kinds of repositories, one stored in a server called remote repository and the other stored in the computer of each developer called local repository. This makes the code exists in multiple locations minimizing losses in times of server outages. A repository can be divided into multiple branches, which are pointers to the latest update of the code in Git repository. Several developers can use the same repository at once by creating branches and committing their code updates in these branches. ^[10]

Popular companies that use Git are Netflix, Robinhood, Reddit, Lyft etc^[10]. The name Git means unpleasant person in British English slang, hinting towards the nature of its creator, Linus Torvalds. (<https://www.quora.com/Why-is-Git-called-Git>)

Question 4:

There wasn't a need to use set or export at this stage. 'pip3' was used to install Flask in the home directory, and the following screenshots shows live Flask server:

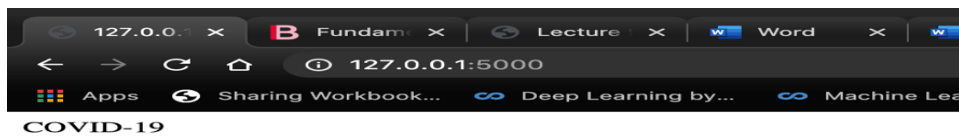
```
gopaljuneja@Gopals-MBP server % flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [22/Oct/2020 12:28:09] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Oct/2020 12:28:09] "GET /favicon.ico HTTP/1.1" 404 -
```



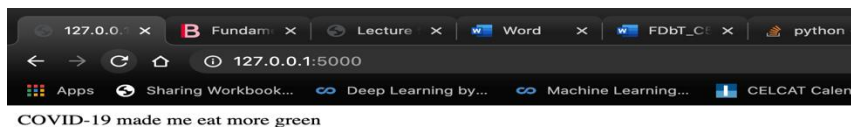
Question 5:

“export FLASK_DEBUG=1” was used to turn on Debug mode in Flask. This resulted in the changes being displayed on the webpage as they were happening in the code.

```
gopaljuneja@Gopals-MBP server % export FLASK_DEBUG=1
gopaljuneja@Gopals-MBP server % flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 331-000-518
```

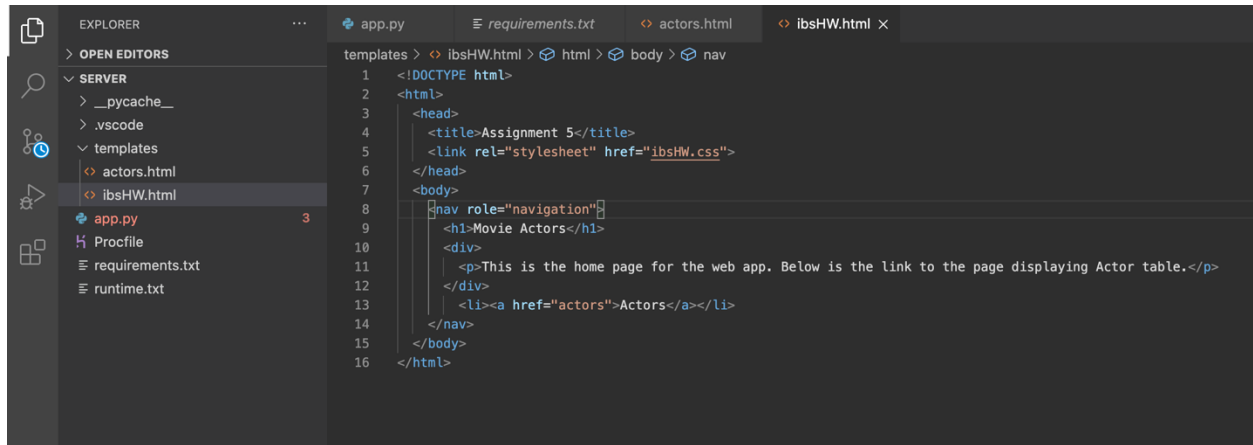


```
gopaljuneja@Gopals-MBP server % flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 663-112-721
127.0.0.1 - - [22/Oct/2020 12:47:22] "GET / HTTP/1.1" 200 -
* Detected change in '/Users/gopaljuneja/Desktop/Assignment5/server/app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 663-112-721
127.0.0.1 - - [22/Oct/2020 12:47:35] "GET / HTTP/1.1" 200 -
* Detected change in '/Users/gopaljuneja/Desktop/Assignment5/server/app.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 663-112-721
127.0.0.1 - - [22/Oct/2020 12:48:33] "GET / HTTP/1.1" 200 -
```

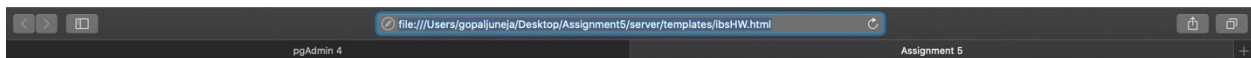


Question 6:

It was slightly tricky at first as the .html files weren't being accessed simply because they were not present in a folder named "templates". Once this was done, the web page was successfully hosted on the Flask server.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Assignment 5</title>
5     <link rel="stylesheet" href="ibsHW.css">
6   </head>
7   <body>
8     <nav role="navigation">
9       <h1>Movie Actors</h1>
10      <div>
11        <p>This is the home page for the web app. Below is the link to the page displaying Actor table.</p>
12      </div>
13      <li><a href="actors">Actors</a></li>
14    </nav>
15  </body>
16 </html>
```



Movie Actors

This is the home page for the web app. Below is the link to the page displaying Actor table.

- [Actors](#)

Below shows majority of interaction, when linking local git repository to the remote repository. “git config” was used to link the email of the remote repository to the local repository, which made committing to git fairly easy.

```
Procfile      __pycache__  app.py        l05SHW.html   Requirements.txt
gopaljuneja@Gopals-MBP server % git init .
Reinitialized existing Git repository in /Users/gopaljuneja/Desktop/Assignment5/server/.git/
gopaljuneja@Gopals-MBP server % git add .
gopaljuneja@Gopals-MBP server % git commit -m "Added procfile and requirements"
[master ebe8cf3] Added procfile and requirements
Committer: Gopal Juneja <gopaljuneja@Gopals-MBP.home>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

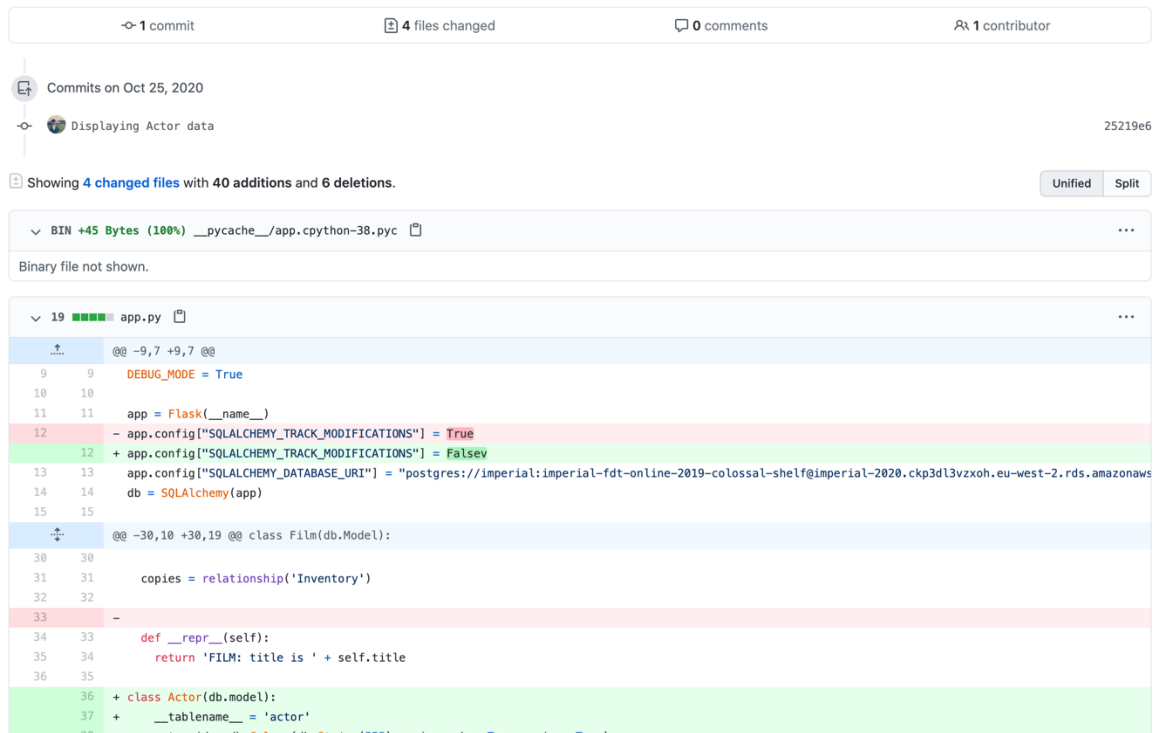
    git commit --amend --reset-author

5 files changed, 1 insertion(+), 3 deletions(-)
rename requirments.txt => requirements.txt (100%)
gopaljuneja@Gopals-MBP server % git config --global edit
error: key does not contain a section: edit
gopaljuneja@Gopals-MBP server % git config --list
gopaljuneja@Gopals-MBP server %
gopaljuneja@Gopals-MBP server % "git -c user.name="Gopal Juneja" -c user.email=gopal.juneja03@gmail.com commit --amend --reset-author"
zsh: command not found: git -c user.name=Gopal
gopaljuneja@Gopals-MBP server % git config --global user.name "Gopal Juneja"
gopaljuneja@Gopals-MBP server % git config --global user.email "gopal.juneja03@gmail.com"
gopaljuneja@Gopals-MBP server % git config user.email
gopal.juneja03@gmail.com
gopaljuneja@Gopals-MBP server %
gopaljuneja@Gopals-MBP server % git commit -m "Added procfile and requirments"
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
gopaljuneja@Gopals-MBP server % git push
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.89 KiB | 969.00 KiB/s, done.
Total 12 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/gjuneja0080/FDbt_A5.git
230d3a0..ebe8cf3 master -> master
gopaljuneja@Gopals-MBP server % touch runtime.txt
gopaljuneja@Gopals-MBP server % python3 -version
Unknown option: -e
usage: /usr/local/bin/python3 [option] ... [-c cmd | -m mod | file | -] [arg] ...
Try 'python -h' for more information.
gopaljuneja@Gopals-MBP server %
gopaljuneja@Gopals-MBP server % python3 -V
Python 3.8.5
```


In order to deploy the project, a runtime.txt file was also needed specifying the version of Python being used.

To deploy projects automatically on Heroku, remote repository was linked to Heroku, and if changes were to be reviewed in Heroku, it would redirect to Git showing a window like the one in screenshot below:



1 commit 4 files changed 0 comments 1 contributor

Commits on Oct 25, 2020

Displaying Actor data 25219e6

Showing 4 changed files with 40 additions and 6 deletions. Unified Split

BIN +45 Bytes (100%) __pycache__/app.cpython-38.pyc

Binary file not shown.

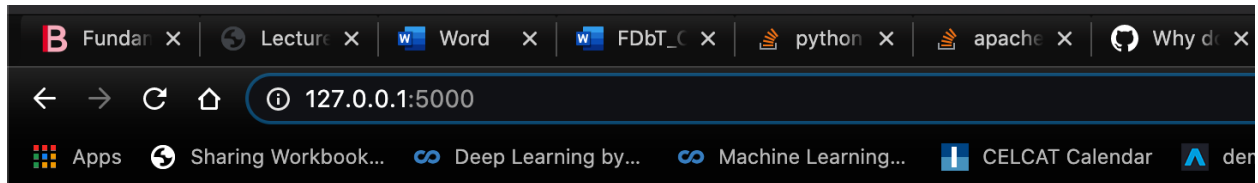
19 app.py

```
@@ -9,7 +9,7 @@
9 9  DEBUG_MODE = True
10 10
11 11  app = Flask(__name__)
12 - app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = True
12 + app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
13 13  app.config["SQLALCHEMY_DATABASE_URI"] = "postgres://imperial:imperial-fdt-online-2019-colossal-shelf@imperial-2020.ckp3dl3vzxoh.eu-west-2.rds.amazonaws.com:5432/imperial"
14 14  db = SQLAlchemy(app)
15 15

@@ -30,10 +30,19 @@ class Film(db.Model):
30 30
31 31  copies = relationship('Inventory')
32 32
33 -
34 33  def __repr__(self):
35 34  return 'FILM: title is ' + self.title
36 35

+ class Actor(db.Model):
37 +  __tablename__ = 'actor'
38 +  actor_id = db.Column(db.String(75), primary_key=True, unique=True)
```

The local version of the web app had this as a beginning page. Using “Heroku local web”, the web app was run locally.



Web Page for Assignment 5

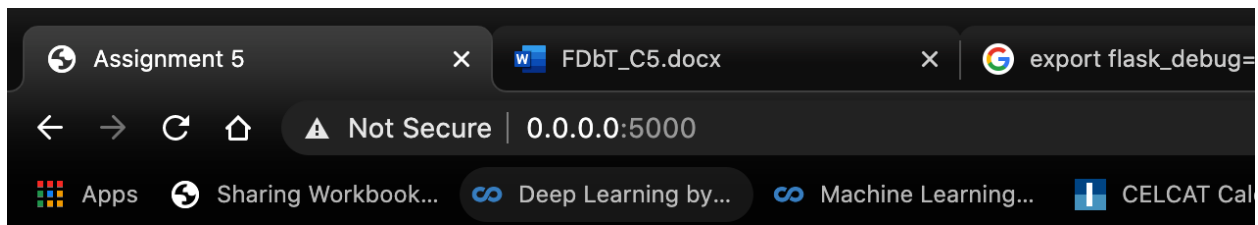
```
[gopaljuneja@Gopals-MBP server % heroku local web
10:09:14 web.1 | [2020-10-25 10:09:14 +0000] [39059] [INFO] Starting gunicorn 20.0.4
10:09:14 web.1 | [2020-10-25 10:09:14 +0000] [39059] [INFO] Listening at: http://0.0.0.0:5000 (39059)
10:09:14 web.1 | [2020-10-25 10:09:14 +0000] [39059] [INFO] Using worker: sync
10:09:14 web.1 | [2020-10-25 10:09:14 +0000] [39061] [INFO] Booting worker with pid: 39061
10:10:00 web.1 | [2020-10-25 10:10:00 +0000] [39059] [CRITICAL] WORKER TIMEOUT (pid:39061)
10:10:00 web.1 | [2020-10-25 10:10:00 +0000] [39061] [INFO] Worker exiting (pid: 39061)
10:10:00 web.1 | [2020-10-25 10:10:00 +0000] [39100] [INFO] Booting worker with pid: 39100
^C[WARN] Interrupted by User
[DONE] Killing all processes with signal SIGINT
10:10:30 web.1 | [2020-10-25 10:10:30 +0000] [39059] [INFO] Handling signal: int
```

An issue encountered when ending processes using “control + Z” key combination was that the zombie states of the processes were still running, which resulted in the following

```
gopaljuneja@Gopals-MBP server % heroku local web
09:44:23 web.1 | [2020-10-25 09:44:23 +0000] [38411] [INFO] Starting unicorn 20.0.4
09:44:23 web.1 | [2020-10-25 09:44:23 +0000] [38411] [ERROR] Connection in use: ('0.0.0.0', 5000)
09:44:23 web.1 | [2020-10-25 09:44:23 +0000] [38411] [ERROR] Retrying in 1 second.
09:44:24 web.1 | [2020-10-25 09:44:24 +0000] [38411] [ERROR] Connection in use: ('0.0.0.0', 5000)
09:44:24 web.1 | [2020-10-25 09:44:24 +0000] [38411] [ERROR] Retrying in 1 second.
09:44:25 web.1 | [2020-10-25 09:44:25 +0000] [38411] [ERROR] Connection in use: ('0.0.0.0', 5000)
09:44:25 web.1 | [2020-10-25 09:44:25 +0000] [38411] [ERROR] Retrying in 1 second.
09:44:26 web.1 | [2020-10-25 09:44:26 +0000] [38411] [ERROR] Connection in use: ('0.0.0.0', 5000)
09:44:26 web.1 | [2020-10-25 09:44:26 +0000] [38411] [ERROR] Retrying in 1 second.
09:44:27 web.1 | [2020-10-25 09:44:27 +0000] [38411] [ERROR] Connection in use: ('0.0.0.0', 5000)
09:44:27 web.1 | [2020-10-25 09:44:27 +0000] [38411] [ERROR] Retrying in 1 second.
09:44:28 web.1 | [2020-10-25 09:44:28 +0000] [38411] [ERROR] Can't connect to ('0.0.0.0', 5000)
[DONE] Killing all processes with signal SIGINT
09:44:28 web.1 | Exited with exit code null
```

Using “ps aux”, the Process ID’s of all the processes using port 5000 were determined and using “kill -9 [PID]”, those processes were killed successfully. This helped in realizing that processes should be exited using “control + c” key combination instead.

Finally using “heroku local web” the application could be accessed locally as seen in the screenshot below

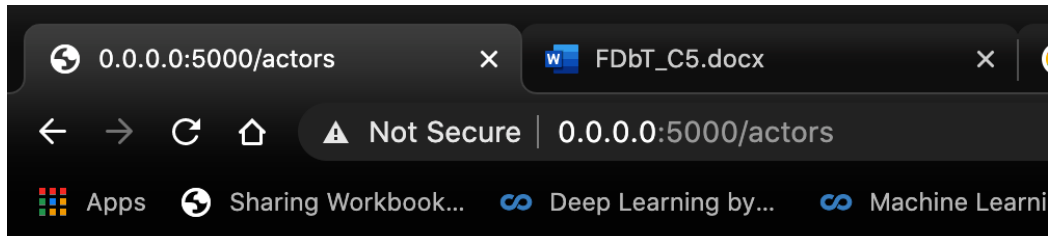


Movie Actors

This is the home page for the web app. Below is the link to the page displaying Actor table.

- [Actors](#)

By adding “/actors” at the end of the URL, the user can redirect to the page containing information about actors. This process is somewhat similar to what has been done in the web application once it’s been deployed on Heroku.



Actor Table Data:

- [Home](#)

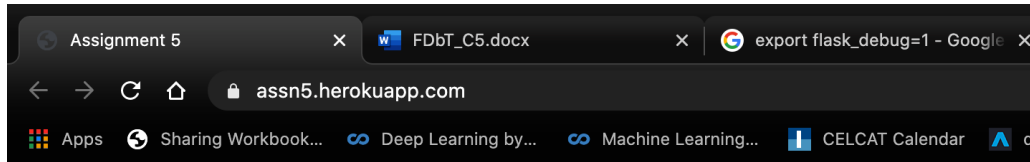
Actor ID	First Name	Last Name	Last Updated
1	Penelope	Guinness	2013-05-26 14:47:57.620000
2	Nick	Wahlberg	2013-05-26 14:47:57.620000
3	Ed	Chase	2013-05-26 14:47:57.620000
4	Jennifer	Davis	2013-05-26 14:47:57.620000
5	Johnny	Lollobrigida	2013-05-26 14:47:57.620000
6	Bette	Nicholson	2013-05-26 14:47:57.620000
7	Grace	Mostel	2013-05-26 14:47:57.620000
8	Matthew	Johansson	2013-05-26 14:47:57.620000
9	Joe	Swank	2013-05-26 14:47:57.620000
10	Christian	Gable	2013-05-26 14:47:57.620000
11	Zero	Cage	2013-05-26 14:47:57.620000
12	Karl	Berry	2013-05-26 14:47:57.620000
13	Uma	Wood	2013-05-26 14:47:57.620000
14	Vivien	Bergen	2013-05-26 14:47:57.620000
15	Cuba	Olivier	2013-05-26 14:47:57.620000

Once the runtime.txt file was specified, the requirements.txt file was made using

“pip3 > freeze requirements.txt” in the server folder on the terminal.

Finally, the app was deployed on Heroku server as follows:

The home page has a link called “Actors” which the user of the web application can click on to see a table of actors with all their details.

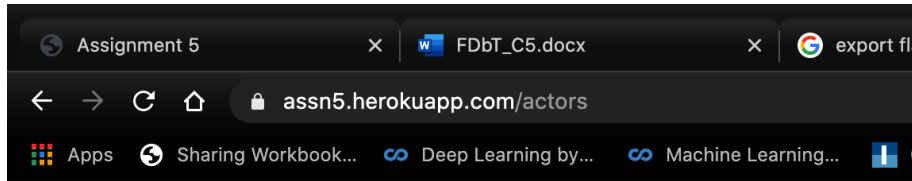


Movie Actors

This is the home page for the web app. Below is the link to the page displaying Actor table.

- [Actors](#)

Once the user is on the Actors page, they can select the Home button on this page to go back to the previous page.



Actor Table Data:

- [Home](#)

Actor ID	First Name	Last Name	Last Updated
1	Penelope	Guinness	2013-05-26 14:47:57.620000
2	Nick	Wahlberg	2013-05-26 14:47:57.620000
3	Ed	Chase	2013-05-26 14:47:57.620000
4	Jennifer	Davis	2013-05-26 14:47:57.620000
5	Johnny	Lollobrigida	2013-05-26 14:47:57.620000
6	Bette	Nicholson	2013-05-26 14:47:57.620000
7	Grace	Mostel	2013-05-26 14:47:57.620000
8	Matthew	Johansson	2013-05-26 14:47:57.620000
9	Joe	Swank	2013-05-26 14:47:57.620000
10	Christian	Gable	2013-05-26 14:47:57.620000
11	Zero	Cage	2013-05-26 14:47:57.620000
12	Karl	Berry	2013-05-26 14:47:57.620000
13	Uma	Wood	2013-05-26 14:47:57.620000
14	Vivien	Bergen	2013-05-26 14:47:57.620000
15	Cuba	Olivier	2013-05-26 14:47:57.620000
16	Fred	Costner	2013-05-26 14:47:57.620000
17	Helen	Voight	2013-05-26 14:47:57.620000
18	Dan	Torn	2013-05-26 14:47:57.620000
19	Bob	Fawcett	2013-05-26 14:47:57.620000
20	Lucille	Tracy	2013-05-26 14:47:57.620000
21	Kirsten	Paltrow	2013-05-26 14:47:57.620000
22	Elvis	Marx	2013-05-26 14:47:57.620000
23	Sandra	Kilmer	2013-05-26 14:47:57.620000
24	Cameron	Streep	2013-05-26 14:47:57.620000
25	Kevin	Bloom	2013-05-26 14:47:57.620000
26	Rip	Crawford	2013-05-26 14:47:57.620000
27	Julia	Mcqueen	2013-05-26 14:47:57.620000

Actors.html is the file that queries actor data from the Imperial PostgreSQL database, whilst **ibsHW.html** is the file that generates the home page for the web application.

templates > <> actors.html > nav

```
1  <style>
2    body{
3      font-family: 'Times New Roman';
4    }
5  </style>
6  <h1>Actor Table Data: </h1>
7    <nav role="navigation">
8      <li><a href="home">Home</a></li>
9    </nav>
10   <table border="1">
11     <tr>
12       <th>Actor ID</th>
13       <th>First Name</th>
14       <th>Last Name</th>
15       <th>Last Updated</th>
16     </tr>
17     {% for actor in actors %}
18       <tr>
19         <td>{{ actor.actor_id }}</td>
20         <td>{{ actor.first_name }}</td>
21         <td>{{ actor.last_name }}</td>
22         <td>{{ actor.last_update }}</td>
23       </tr>
24     {% endfor %}
25   </table>
```

app.py

actors.html

ibsHW.html X

templates > <> ibsHW.html > html > body > nav

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Assignment 5</title>
5      <link rel="stylesheet" href="ibsHW.css">
6    </head>
7    <body>
8      <nav role="navigation">
9        <h1>Movie Actors</h1>
10       <div>
11         <p>This is the home page for the web app. Below is the link to the page displaying Actor table.</p>
12       </div>
13       <li><a href="actors">Actors</a></li>
14     </nav>
15   </body>
16 </html>
```

```
app.py X  <> actors.html  <> ibsHW.html
app.py > ...
1  from flask import Flask, render_template
2  import os
3  from flask_sqlalchemy import SQLAlchemy
4  from sqlalchemy.orm import relationship
5  from sqlalchemy import Table, Column, Integer, ForeignKey
6
7  HOST = '0.0.0.0'
8  PORT = int(os.environ.get('PORT', 5000))
9  DEBUG_MODE = True
10
11 app = Flask(__name__)
12 app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = True
13 app.config["SQLALCHEMY_DATABASE_URI"] = "postgres://imperial:imperial-fdt-online-2019-colossal-shelf@imperial-2020.ckp3dl3vzxoh.eu
14 db = SQLAlchemy(app)
15
16 class Actor(db.Model):
17     __tablename__ = 'actor'
18     actor_id = db.Column(db.String(255), primary_key=True, unique=True)
19     first_name = db.Column(db.String(255), index=True)
20     last_name = db.Column(db.String(255), index=True)
21     last_update = db.Column(db.DateTime)
22
23     def __repr__(self):
24         return 'Actor ID: ' + self.actor_id, 'First Name: ' + self.first_name, 'Last Name: ' + self.last_name, 'Last Updated: ' + s
25
26 @app.route('/')
27 @app.route('/home')
28 def ibsHW():
29     return render_template("ibsHW.html")
30
31 @app.route('/actors')
32 def actors():
33     actors = Actor.query.all()
34     return render_template('actors.html', actors = actors)
35
36 if __name__ == '__main__':
37     app.run(debug=DEBUG_MODE, host=HOST, port=PORT)
```

Alternatively, if users do not want to click on links on the web application can also edit the URL of the web application and put “/actors” at the end to access actors table page or just a “/” or “/home” to access the home page. This done by the “@app.route()” functionality seen in above screenshot.

Conclusion:

This assignment helped gain familiarity towards HTML, SQLAlchemy, Flask and Object Relational Mapping. Combination of these tools can be extremely powerful and is prominently used in industry in today’s world. The use of version control is deemed extremely important as well as scale of such projects is usually exponentially higher than what’s been implemented in this assignment. Given more time, emphasis would be put on enhancing the look of the web page and adding more data into the web page from the Imperial database.

References:

1. Hoyos, M. 2018. "What is an ORM and Why Should You Use It". BitsPieces. Available at: <https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a>
2. Barnwell, Neil. 2009. "In MVC, does ORM represent the Model". Stack Overflow. Available at: <https://stackoverflow.com/questions/836929/in-mvc-does-an-orm-represent-the-model#:~:text=No%2C%20the%20ORM%20is%20the,your%20database%20and%20vice%20versa.&text=If%20you'd%20like%20to,Nhibernate%20and%20the%20repository%20pattern.>
3. 2020. "Flask_ (Web Framework)". Wikipedia. Available at: [https://en.wikipedia.org/wiki/Flask_\(web_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))
4. Gill, J. 2019. "A Complete Introduction to Python Flask Framework". XenonStack. Available at: <https://www.xenonstack.com/blog/python-flask-framework/>
5. 2017. "Why is it named Flask?". Reddit. Available at: https://www.reddit.com/r/flask/comments/6cs1b2/why_is_it_named_flask/
6. Quinta Group. Available at: <https://quintagroup.com/cms/python/sqlalchemy>
7. "QB: The Database Toolkit for go". Awesome Open Source. Available at: <https://awesomeopensource.com/project/slicebit/qb>
8. Heroku Documentation. Available at: <https://www.heroku.com/platform>
9. "What is Heroku?". Available at: <https://stackshare.io/heroku>
10. Sridhar, A. 2018. "An introduction to Git: What is it, and how to use it. Free Code Camp. Available at: <https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/>