University of Leeds **School of Computing**
COMP3011, 2021-2022
Web Services and Web Data

A RESTful API for Rating Professors

By
MARK STEPHEN MULILA MUINDE
STUDENT ID 201374599
Email: sc19msmm@leeds.ac.uk

Date: 22nd March 2022

1. Introduction

COMP3011 coursework 1 was a challenge for me as all my prior web development projects were done with Flask as a framework, but to my pleasant surprise I found Django to be more convenient for web development as most of what I needed was already in the documentation. I have implemented all other features, but the *login*() and logout() function aren't perfect, therefore my functions don't demand that a user log in, but otherwise they all work perfectly. I have uploaded the Django server code to pythonanywhere.com and I have written the client in Python 3 and thoroughly tested it.

1. The URL of my pythonanywhere account is http://sc19msmm.pythonanywhere.com/
2. Superuser usernamename: 'ammar'
   Superuser email: 'ammar@email.com'
   Superuser password: 'ammar'

2. The Database

The professors are saved in the *Professor* table that has two fields: a unique *CharField* labelled *professor_id* and a *CharField* labelled *professor_name.*

The modules are saved in the *Module* table that has five fields: a unique *CharField* labelled *module_code,* a *CharField* labelled *module_name,* a *ManyToManyField* labelled *module_professor* relating to the *Professor* table*,* a *CharField* labelled *year* and a *PositiveIntegerField* labelled *semester.*

Kindly note that when creating a module instance, even if it is the same module taught in different years, please give them unique names for my implementation to work. The convention I have chosen is to label them by years, for example 'CD1 taught in 2017' as per the coursework brief becomes 'CD17 taught in 2017' in my implementation, 'CD1 taught in 2018' as per the coursework brief becomes 'CD18 taught in 2018' in my implementation, 'PG1 taught in 2017' as per the coursework brief becomes 'PG17 taught in 2017' in my implementation and 'PG19 taught in 2019' as per the coursework brief becomes 'PG19 taught in 2017' in my implementation. Whenever I called view() and average() with the module codes given in the coursework brief I couldn't get them to work as the .get() method returned two module instances and so the server crashed. With the unique module codes my implementations work perfectly.

Ratings of professors are saved in the *Rating* table which has three fields: a *ForeignKey* field labelled *professor* relating to the *Professor* table, a *PositiveIntegerField* labelled *rating* for the rating value and a *ForeignKey* field labelled *module* relating to the *Module* table
Use of *ForeignKey* fields was straightforward as the COMP3011 lecture notes provided enough guidance and coincidentally they helped me avoid a bug in one of the APIs where one can rate a professor who doesn't teach a module.

3. The APIs

a. Register

Registers a user
URL: http://sc19msmm.pythonanywhere.com/api/register/ . **HTTP method:** POST

Request Data: Registration data to register a user .**Response Data:** Confirmation of user creation via response code 201 (CREATED).

b. Login

Logs a user in

URL: http://sc19msmm.pythonanywhere.com/api/login/ . **HTTP method:** POST

Request Data: Login credentials to log in a user. **Response Data:** Confirmation of user logged in via HTTP response code 200 (OK).

c. Logout

Logs a user out

URL: http://sc19msmm.pythonanywhere.com/api/logout/ . **HTTP method:** POST

Request Data: Log a user out. **Response Data:** Confirmation of user logged out via HTTP response code 200 (OK).

d. List

Lists all modules and professors

URL: http://sc19msmm.pythonanywhere.com/api/list/ . **HTTP method:** GET

Request Data: All *Module* and *Professor* instances. **Response Data:** JSON payload of a *dict* with the data to be parsed into and displayed by a *pandas* dataframe. Confirmation of success via HTTP response code 200 (OK).

e. View

View the average ratings of professors.

URL: http://sc19msmm.pythonanywhere.com/api/view/ . **HTTP method:** GET

Request Data: All *Rating* instances gotten from a *QuerySet* of *Module* and *Professor* instances. **Response Data:** JSON payload of a *dict* with the professor, module, and rating data, to be parsed into and displayed by a *pandas* dataframe. Confirmation of success via HTTP response code 200 (OK).

f. Average

View the average ratings of professors.

URL: http://sc19msmm.pythonanywhere.com/api/average/ . **HTTP method:** POST

Request Data: Professor's ID and module ID sent via JSON. **Response Data:** JSON payload of a *dict* with the professor, module, and average rating data. Confirmation of success via HTTP response code 200 (OK).

g. Rate

Rate a professor.

URL: http://sc19msmm.pythonanywhere.com/api/rate/ . **HTTP method:** POST

Request Data: Professor's ID, module ID, module year, module semester and rating value sent via JSON . **Response Data:** JSON payload of a *dict* with the professor, module, and average rating data. Rating creation success via HTTP response code 201 (CREATED).

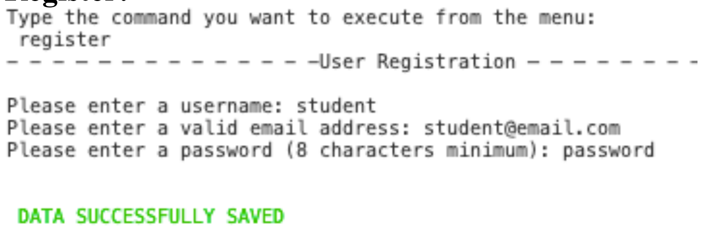
4. The Client

The client operates via a command line interface where the user inputs commands as described on the menu with fail-safes implemented in case of client error. Each function relates to an API on the server.

5. Testing

Screenshots taken from the client with success messages when commands are invoked:

Register:

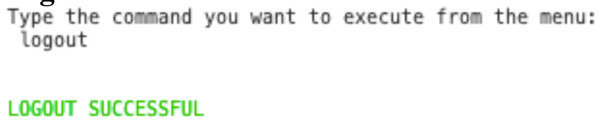


```
Type the command you want to execute from the menu:
 register
- - - - - - - - - - - - - -User Registration - - - - - - - -

Please enter a username: student
Please enter a valid email address: student@email.com
Please enter a password (8 characters minimum): password


 DATA SUCCESSFULLY SAVED
```

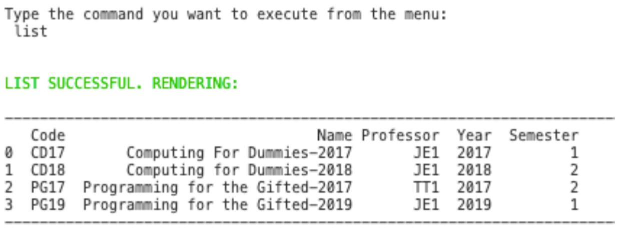
Logout:



```
Type the command you want to execute from the menu:
 logout


LOGOUT SUCCESSFUL
```

List:

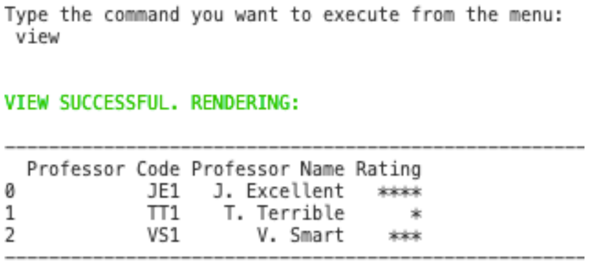


```
Type the command you want to execute from the menu:
 list

LIST SUCCESSFUL. RENDERING:

----------------------------------------------------------------
   Code                         Name Professor  Year  Semester
0  CD17       Computing For Dummies-2017       JE1  2017         1
1  CD18       Computing for Dummies-2018       JE1  2018         2
2  PG17  Programming for the Gifted-2017       TT1  2017         2
3  PG19  Programming for the Gifted-2019       JE1  2019         1
----------------------------------------------------------------
```

View:

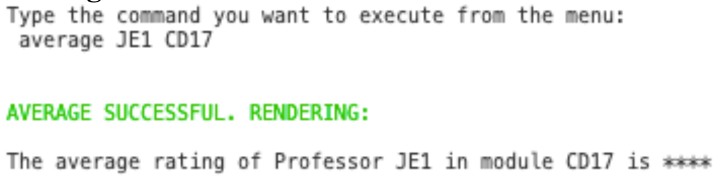


```
Type the command you want to execute from the menu:
 view

VIEW SUCCESSFUL. RENDERING:

------------------------------------------------------
  Professor Code Professor Name Rating
0            JE1    J. Excellent   ****
1            TT1     T. Terrible      *
2            VS1        V. Smart    ***
------------------------------------------------------
```

Average:

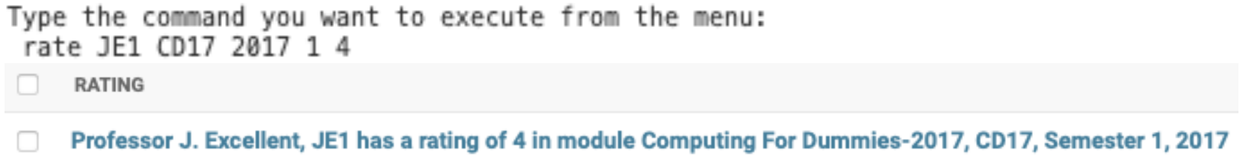


```
Type the command you want to execute from the menu:
 average JE1 CD17

AVERAGE SUCCESSFUL. RENDERING:

The average rating of Professor JE1 in module CD17 is ****
```

Rate



```
Type the command you want to execute from the menu:
 rate JE1 CD17 2017 1 4
```

☐	RATING
☐	Professor J. Excellent, JE1 has a rating of 4 in module Computing For Dummies-2017, CD17, Semester 1, 2017

6. Using the client

If you do not have python installed, from the links below paste the link that relates to your device in a browser and install it before doing step A:

a) Windows: https://www.python.org/ftp/python/3.9.2/python-3.9.2.exe

OR

https://www.python.org/ftp/python/3.9.2/python-3.9.2-amd64.exe

b) MacOS: https://www.python.org/ftp/python/3.9.2/python-3.9.2-macosx10.9.pkg

Assuming you already have python installed, open a terminal, cd into the root directory containing the client application and activate the virtual environment provided named *Django* as follows:

```
source django/bin/activate
```

Once the virtual environment is activated, run the following:

```
python3 client.py
```

To use the client follow the following instructions:
1. Type 'register' to register a new user
2. Type 'login https://sc19msmm.pythonanywhere.com/' to log into your account
3. Type 'logout' to log out of your session
4. Type 'list' to get a list of all modules and their corresponding professors
5. Type 'view' to view ratings for all the professors
6. Type 'average <professor_id> <module_code>' with the required arguments to get the average rating of a professor in a given module
7. Type 'rate <professor_id> <module_code> <year> <semester> <rating>' with the required arguments to rate a professor in a given module

Kindly note that for my implementation to work each module must have a unique code as explained in section 2 .

Kindly note that *login*() and *logout*() are not completely implemented as explained in section 1, but otherwise everything else works as expected.

Appendix A: Testing Protocol

To test your service, the assessor will login to your admin site using the superuser name and password that you have provided. The assessor should then be able to use the admin site to:

1. Add the following two modules:
    a) Computing for Dummies.
    b) Programming for the Gifted.
2. Add the following three professors:
    a) J. Excellent.
    b) V. Smart.
    c) T. Terrible.
3. Create the following module instances:
    a) Computing for Dummies, Semester 1, Year 2017, taught by J. Excellent and V. Smart
    b) Computing for Dummies, Semester 2, Year 2018, taught by J. Excellent.
    c) Programming for the Gifted, Semester 2, Year 2017, taught by T. Terrible.
    d) Programming for the Gifted, Semester 1, Year 2019, taught by J. Excellent.

The assessor will then use your client to:

1. Register two new users (students).
2. Login in as the first user and use the *list* command to see a list of all module instances.
3. Use the *rate* command to:
    i. Give professor J. Excellent a rating of 2 stars in module instance a), and a rating of 5 stars in module instance b).
    ii. Give professor V. Smart a rating of 3 stars in module instance a).
    iii. Give professor T. Terrible a rating of 1 star in module instance c).
    iv. Give professor J. Excellent a rating of 4 stars in module instance d).
4. Logout, then login as the second user, and use the *rate* command to:
    i. Give professor J. Excellent a rating of 3 stars in module instance a), and a rating of 4 stars in module instance b).
    ii. Give professor V. Smart a rating of 2 stars in module instance a).
    iii. Give professor T. Terrible a rating of 1 star in module instance c).
    iv. Use the *view* command to see the rating of all professors. For the above data, professor J. Excellent would have a rating of 4, V. Smart 3, and T. Terrible 1. Note that fractions are rounded to nearest integers.
    v. Use the *average* command to see the rating of professor J. Excellent in the Computing for Dummies module. For the above data this would be 4.