University of Leeds **School of Computing**
Web Services and Web Data COMP3011
Semester 2, 2019-2020

Coursework 1 (30 marks = 30% of total module marks)

Important Note: This is an individual project, NOT a team project. Each student must implement their own web API and client application.

Submission Deadline: **13/03/2020 at 10:00 am**

1. The Brief

In this project, you will use the Django framework to implement a RESTful web API for rating professors. You will host your website on www.pythonanywhere.com (for free). You will also write a simple command line application for interacting with the API.

2. The Web Service

We want to develop a RESTful web service to allow students to rate the teaching of professors in various modules on a scale from 1 to 5. The service must provide all the required web API's to allow client applications to offer the functionality described below.

The service maintains data about professors, modules, and the rating of professors by different users (students). Information about modules and professors are manually added by the admin of the service using the admin site. The admin site is automatically created by the Django web development framework.

A module may be:

- Taught by different professors in different academic years.
- Taught by different professors in different semesters.
- Taught by more than one professor at the same time (for example each teaching some part of the module).

Since an academic year spans two calendar years (e.g. 2018-19), an academic year will be given by its first year only (hence 2018-19 is given as 2018).

Users of the service (students) can rate professors but cannot add or change module information. Before they can rate professors, users must register by providing a username, email, and password. Users can only rate professors when they are logged in to the service. The overall rating of a professor is the average of the professor's rating by all users across all module instances taught by this professor. A module instance is a module taught in a certain year and semester by one or more professors. Any decimal fraction in the average is rounded to the nearest integer.

A client application connected to the service will provide the user with the following options:

Option 1. View a list of all module instances and the professor(s) teaching each of them. Here is an example of a possible client application output for this option.

Code	Name	Year	Semester	Taught by
CD1	Computing for Dummies	2017	1	JE1, Professor J. Excellent VS1, Professor V. Smart
CD1	Computing for Dummies	2018	2	JE1, Professor J. Excellent
PG1	Programming for the Gifted	2017	2	TT1, Professor T. Terrible

Note that modules and professors are given unique identifiers in the web service to avoid any possible mix-up between names.

Option 2. View the rating of all professors. Here is an example of a possible client application output for this option.

```
The rating of Professor J. Excellent (JE1) is *****
The rating of Professor T. Terrible (TT1) is *
The rating of Professor V. Smart (VS1) is **
```

Option 3. View the average rating of a certain professor in a certain module:

```
The rating of Professor V. Smart (VS1) in module Computing for Dummies (CD1) is ***
```

Option 4. Rate the teaching of a certain professor in a certain module instance.

Note that all filtering of data and calculations should be done by the server, i.e. the client application does not process incoming data; the application simply displays the data returned by the service in a human readable format.

You are required to:

1. Design and implement a suitable database for the above service. You must specify all required tables, the fields of each table and their data types, and the relationships between the tables.
2. Design and implement all web API's that must be provided by the service to allow a client application to perform the functionality described in Options 1 – 4. For each API, you must specify: the purpose of the API, a URL, an HTTP method, the request data, and the response data.

3. The Client Application

You are also required to write a command line client application to interact with the API. The application should be written in Python 3 and should be able to send requests to, and process responses from, the web service. The client must support the following commands:

register
This is used to allow a user to register to the service using a username, email and a password. When the command is invoked, the program prompts the user to enter the username, email, and password of the new user. The syntax for this command is:

```
register
```

login
This command is used to log in to the service. The syntax for this command is:

```
login url
```

where:

url is the address of the service. Since you will be hosting your web service at www.pythonanywhere.com, this should be something like 'xxxxxx.pyhtonanywhere.com', where xxxxxx is your university username.

Invoking this command will prompt the user to enter a username and password which are then sent to the service for authentication.

logout
This causes the user to logout from the current session. The syntax for this command is:

logout

list
This is used to view a list of all module instances and the professor(s) teaching each of them (Option 1 above). The syntax for this command is:

list

view
This command is used to view the rating of all professors (Option 2 above). The syntax for this command is:

view

average
This command is used to view the average rating of a certain professor in a certain module (Option 3 above). The syntax of the command is:
average *professor_id module_code*
where:
professor_id is the unique id of a professor, and
module_code is the code of a module.

rate
This is used to rate the teaching of a certain professor in a certain module instance (Option 4 above). It has the following syntax:
rate *professor_id module_code year semester rating*
where*:*
professor_id is the unique id of a professor, e.g. JE1,
module_code is the code of a module, e.g. CD1,
year is a teaching year, e.g. 2018,
semester is a semester number, e.g. 2*, and*
rating is a numerical value between 1-5.

4. Implementation Guidelines

To simplify system development, you can divide your work into individual Work Packages (WP) as follows:

WP1: implement the database model in Django, activate the admin site, test your model manually and refine it as needed.

WP2: implement the web API, one service at time, and in parallel write the client application code to request services and test each service as you go.

WP3: complete and finalize the client application.

WP4: test and debug your entire system on the local host.

WP5: upload the web API to pythonanywhere.com and test everything.

5. Submission Instructions

1. Once you are satisfied that your API is working properly on the local host, open a free account and upload the server (API) code to pythonanywhere.com. Detailed instructions on how to upload a Django project to pythonanywhere.com are available here https://help.pythonanywhere.com/pages/DeployExistingDjangoProject
2. When you register for a free account at www.pythonanywhere.com, please use your university username when you are prompted for an account name. Hence, if your university username is sc15xyz, your root domain address would be sc15xyz.pythonanywhere.com. This will make it much easier for me to know who owns an account during coursework assessment.
3. Make sure that you have properly uploaded your server code to pythonanywhere.com and tested it thoroughly.
4. Make sure that your client was written in Python 3 and fully tested on one of the DEC-10 machines.
5. Nominate me as your teacher on pythonanywhere.com. This will enable me to access and view your API code directly on the website. You can do this from the 'teacher' tab in your 'account' page. The username you have to use is 'ammarsalka'.
6. Using your admin site, add me as an another admin user to your web service, use ‘ammar’ for username and invent a simple password for this account.
7. Prepare a plain text file called readme.txt containing:
    I. instructions on using the client (like a quick help page with a list of all commands and how to use them).
    II. the name of your pythonanywhere domain.
    III. the password I have to use to login to my admin account on your service.
    IV. any other information I need in order to use your client.
8. Put the readme.txt file and your client program in one directory called 'myclient'
9. Bundle your Django project directory (the outer one) and the 'myclient' directory into a single directory. The name of this directory should be your university email address (without the @leeds.ac.uk)
10. Compress the directory with Zip, and upload to Minerva.

As part of your submission, you should also submit a brief report (around 5 pages excluding the title page) that clearly, yet briefly, describes how you implemented the database, the APIs, and the client application. The report should also include brief instructions on how to invoke and use the client application. Please do NOT fill your report by copying text from online resources as I am only interested to know what you have done yourself. The clarity of your report will affect the marks you are awarded for the relevant aspects of the mark scheme. A template for the report will be provided in due course. **The report should be uploaded to Minerva and a print copy handed in to the SSO.**

Marking Scheme

The database of the web service was designed and implemented correctly	(10 marks)
All APIs were designed and implemented correctly	(10 marks)
The client application works correctly	(10 marks)

Good Luck
Dr Ammar Alsalka