159.352 2023/S1 – Assignment 1 Brief

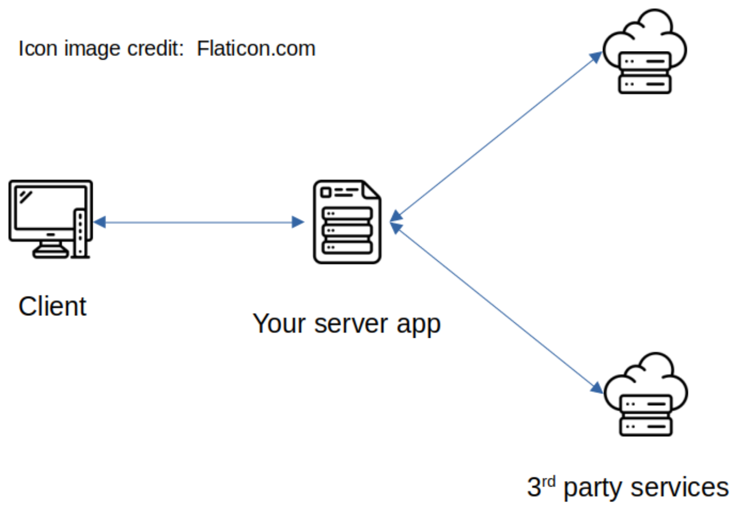
Due date: 2023 April 6, 11:55pm.

This assignment is worth 25 marks (25% towards your final grade).

Online Psychological Profiling

In this assignment, you will extend the minimalistic Web servers developed in the exercises from the lectures. Here you will develop a Web application to generate an online (and not necessarily serious) psychological profile of the user.

Your application will function both as a server to the end user and as a client itself in order to consume 3rd party Web services via RESTful APIs—as in the following schematic.



A key aim of this assignment is to gain hands-on experience with HTTP fundamentals. Do not use high level frameworks (Django, Flask, Nodejs, etc) as they abstract the low level HTTP functionality. You will get a chance to use these frameworks in the 2nd assignment. Here you are being asked to implement your own micro-framework.

Requirements

Authentication

Use basic HTTP authentication to protect your site. Implement this in your Python server so that without the correct login credentials, none of the resources will be accessible.

Use your 8-digit student ID for both the user name and password, e.g.. when your browser asks for the credentials put in :

```
User Name:  12345678
Password:   12345678
```
(replace `12345678` with your own ID)

[4 marks]

The back end

Design your server to respond to the following URI paths. Also add other path definitions as you see fit. Any undefined path should result in a `404 NOT FOUND` response.

`/`

The default/empty path should deliver the content of the "landing page" `index.html` (or otherwise). This will function as the "front end" as below.

`/form`

Deliver the content of the file `psycho.html` to the user. The content here is a classic form using vanilla HTML. Your browser should then display this appropriately.

`/analysis`

This is the "action" upon submitting the form data from the browser. You will first need to (slightly) modify `psycho.html`.

This URI should action the following tasks at the server-side:

1. Parse the input form data and store at the server side in a suitable format.
2. Analyze the input data to generate a "psychological profile". The results should be as follows:
   - an assessment of the users suitability for their selected preferred career
   - a recommendation of movies the user might like—fetch the relevant data from a 3rd party site via a RESTful API (see below)
3. Fetch a random image, from a 3rd party site (see also below), for each pet that the user selected in the check boxes. Store these at the server side.

For step 2 you can analyze the form data in any way you see fit—be creative and have fun!

The result of actioning this URI should be the psychological profile data and image files stored at the server. These data should NOT be be delivered to the client. The server response should just be a simple message in a suitable format.

The delivery of actual data is to be handled by the `view` URI paths below.

`/view/input`

This URI delivers the input data to the client. This should be delivered in a suitable serialization format. Do not include HTML. The visualization of the data is to be handled by the front end.

`/view/profile`

Deliver the psychological profile data to the user for display in the front end. Again, use a suitable serialization format.

[11 marks]

The front end

This deals with the presentation and visualization of the data generated at the server. Design a suitable front end in `index.html` to interact with server. You will need to add JavaScript functionality. This should have the following functionality:

1. Fetch the input data form `psycho.html` by actioning the appropriate back end URI. The content of this HTML will need to be displaced in a separate browser window.
2. Fetch the serialized input data and display in the browser in a suitable human viewable form.
3. Fetch the serialized psychological profile data together with the pet images (if any) and display them in the browser in a human readable format.

Results should be displayed in a manner you would expect to see in a browser window after appropriately parsing the serialization text. Do not just dump this raw text in the display document.

[7 marks]

Deployment

Package your application as a Docker image and get it working in a Docker container.

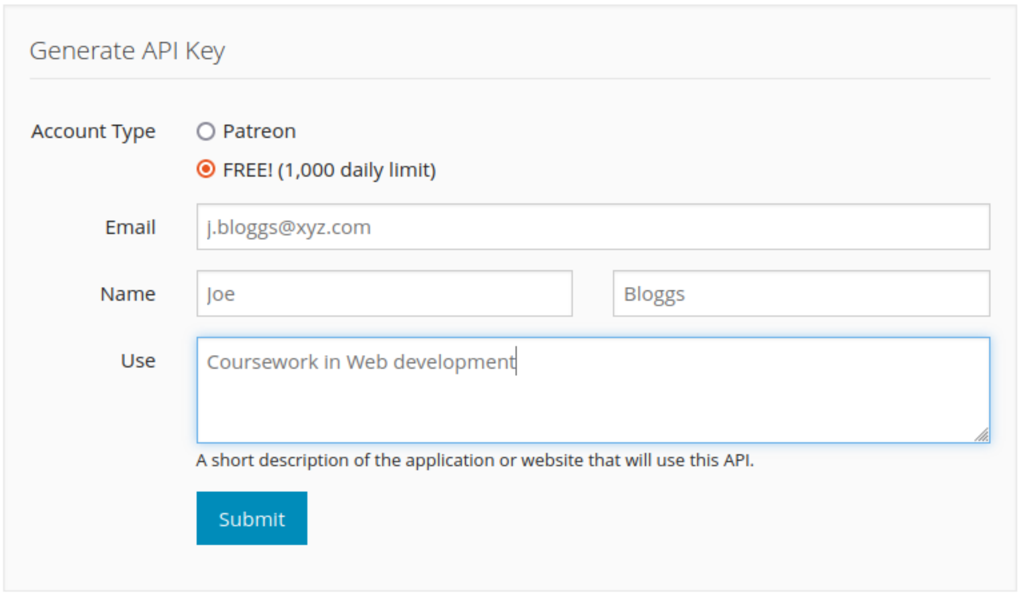
Deployment and submission instructions will be provided closer to the due date.

[3 marks]

Appendix: 3rd Party Services

Movie Database

Visit `http://www.omdbapi.com/apikey.aspx`. Click on the “FREE!” radio button and enter your details, e.g.:



Generate API Key

Account Type	( ) Patreon [✓ FREE! (1,000 daily limit)]
Email	j.bloggs@xyz.com
Name	Joe / Bloggs
Use	Coursework in Web development

A short description of the application or website that will use this API.

Your API key will be sent to your e-mail. To get details on a particular movie title, try the URI (replace **`abcdefgh`** with your own key)

`http://www.omdbapi.com/?apikey=abcdefgh&t=alien`

and see what comes back. See the documentation for other types of API calls.

Random Animal Pictures

The following URIs will fetch image metadata for animal image files. No API keys are needed. Note: the metadata will be in JSON format which will contain the URL for the actual image file—which you also need to action.

To understand what is going here, use **`curl`**, **`wget`**, or your own Python test script to action these URIs. Then just look at the JSON text.

Random Dog

`https://dog.ceo/api/breeds/image/random`

Random Cat

`https://api.thecatapi.com/v1/images/search`

Random Duck

`https://random-d.uk/api/v2/random`