University of Leeds **School of Computing**

COMP3011, 2023-2024

Web Services and Web Data

A RESTful API for News Aggregation

By

Brooklyn Mcswiney

201540315 - ed20b3m@leeds.ac.uk

Date: March 2024

1. Introduction

All of the feature requirements of the API and the client have been implemented within my coursework. My Django server code has been written in Python 3 and uploaded to Python anywhere and can be found at http://ed20b3m.pythonanywhere.com. My API and client have been thoroughly tested through the manual use of HTTP requests in the client and a web browser.

Login for http://ed20b3m.pythonanywhere.com.

- username: ammar
- password: password1234

2. The Database

The news API uses a relational database with two tables. The first table is the users table which comes built-in with Django admin systems. The second database stores the data required for each news article, including a foreign key to the users database for the author field.

3. The APIs

The API was implemented utilising the views class within Django to create different paths for the API. The first path for the API implemented was the login function. This function takes in a POST request containing a username and password within the request body. This function then authenticates these login details and should this authentication pass this user is marked as logged in through Django sessions. Should this authentication not pass the function returns a message that the login failed with status code 503. The next function implemented was the ability to log out of your current session, this works by checking if the current user session is logged in and if it is the function will mark this user as logged out instead of logged in.

The next set of functions implemented are with regards to making GET, POST, and DELETE requests to the “api/stories/” URL. This function begins by checking which type of request it has received, should this request be a get request it will call the “get_story” function. The “get_story” function will filter the story database based on 3 categories that are encoded within the request URL, these categories being: “story_cat”, “story_region”, and “story_date”. Should any of these be a “*” symbol the corresponding filter returns all the data. This request is then responded to with a JSON object containing the database search results and the status code 200. Should the “/api/stories/” URL get a POST request with the correct data in the body a new story will be added to the database if the current user session is marked as logged in the API returns the message “success” with the response code 200. Finally, should the “/api/stories/<id>” URL get a DELETE request with an existing story the API will delete this story from the database if the user session is marked as “logged in” and return the message “OK” with the response code 200. Should the user not be logged in the API will return “user not logged in” with the response code 503 and should the id not be found the API will return “Story not found” with the response code 503.

4. The Client

The client uses a “while true” loop to take user inputs to interact with the API. The first two commands to discuss are the simplest commands “help” which prints all available commands and what they do and “exit” which closes the program. The next command is that of the “login <URL>” command which asks the user to provide a URL to the API service they wish to login to. Should this URL be provided the client will then prompt the user for login information to be placed into the request body for authentication with the API. If this request returns successfully the URL that the client has logged into will be saved and used again if the user wishes to use the “post” and “delete”. The delete request works by checking if the user is logged in and if the user is it will send a delete request to the current service corresponding to the user-specified ID. The post command works similarly to the delete request in that it checks the user is logged in before continuing. Should the user be logged in the client will prompt the user to enter data corresponding to each field required for a news article and then send a post request to the current API.

The list command performs a get request to the newssites API and lists all available news services. The news command performs a get request to any specified news API using the IDs found in the newssites API and filters these requests by user-specified filters. Should the user not provide a specific ID to get articles from the client will get all the news articles from 20 different APIs and output them.