**University of Leeds** **School of Computing**

**COMP3011, 2023-2024**
**Web Services and Web Data**

# A RESTful API for News Aggregation

By
Abhinav Ramakrishnan
201517997 sc21a2r@leeds.ac.uk

**Date:** 18th March 2024

# 1. Introduction

I have been able to implement all the coursework. On the server side, this includes user authentication, allowing authors to log in and log out, meaning only authorized authors can access other functionalities. I have also implemented the ability for users to post stories, get all stories or stories based on filters, and the ability to delete stories based on a key they provide.

I've implemented the entire specification for the client-side application. This application provides a command-line interface for users to interact with news service providers who use the common api's. It covers all required functionalities, including logging in to news services using provided URLs, posting news stories, retrieving news stories based on various criteria such as ID, category, region, and date, listing available news services, and deleting specific news stories.

I have uploaded the Django server code to pythonanwhere.com. The client application is written in python 3.11.5 and has been thoroughly tested.

**URL**
sc21a2r.pythonanywhere.com
**Superuser Credentials**
username: ammar
password: ammar123

# 2. The Database

The Django database model for the news service comprises two main tables: Author and Story.
The Author table represents the authors who can post news stories. This table is linked to Django's built-in User model via a one-to-one relationship, enabling seamless integration with Django's authentication system. It allows the author field to utilize the username and password fields from the User model, along with an additional 'name' field for storing the author's name.
The Story table stores the news stories posted on the service. It includes fields for the headline, category, region, author, date, time, and story details. The category and region fields are implemented as choices to restrict inputs to predefined options. For the category field, the options are: politics (pol), art, technology (tech), and trivia. For the region field, the options are: uk (United Kingdom), eu (Europe), and w (World). The author field establishes a many-to-one relationship with the Author table, linking each story to its respective author. One author can write many stories, but each story can only be written by one author. The date field stores the date of the story, while the time field records the timestamp of when the story was posted. Finally, the details field provides information about the story, limited to 128 characters.

## 3. The APIs

**Log In:**

- The Login view accepts POST requests.
- It checks the provided username and password from the request payload.
- If the credentials are correct, it authenticates the user and logs them in using Django's authentication system.
- It responds with a 200 OK status and a welcome message if login is successful.
- If the login fails (due to incorrect credentials or other reasons), it responds with an appropriate status code and explanation.

**Log Out:**

- The Logout view accepts POST requests.
- Checks to see if any user is logged in first.
- It logs out the currently authenticated user.
- It responds with a 200 OK status and a goodbye message upon successful logout.
- If the logout fails, it responds with an appropriate status code and explanation.

**Post a Story:**

- The Stories view accepts POST requests.
- It expects data in JSON format containing story details such as headline, category, region, and details.
- It verifies if the user is logged in before allowing story posting.
- If the user is authenticated, it creates a new story object with the provided details and saves it to the database.
- It responds with 201 CREATED if the story is successfully added to the database.
- If the story cannot be added (e.g., unauthenticated author), it responds with 503 Service Unavailable along with an explanation.

**Get Stories:**

- The Stories view accepts GET requests.
- It expects query parameters for story category, region, and date.
- It retrieves stories from the database based on the provided filters.
- It formats the retrieved stories into a JSON response as per the API specifications.
- It responds with 200 OK and the list of stories if stories are found.
- If no stories are found, it responds with 404 Not Found along with an explanation.

**Delete Story:**

- The Delete view accepts DELETE requests with the story key as part of the URL.
- It verifies if the user is logged in before allowing story deletion.
- It retrieves the specified story from the database and deletes it.
- It responds with 200 OK if the story is successfully deleted.
- If the deletion fails (e.g., story not found), it responds with 503 Service Unavailable along with an explanation.

## 4. The Client

The client is a command-line interface that allows users to interact with a news service API. Implemented in Python, the application uses the requests module to handle HTTP requests to various API endpoints. It offers a range of functionalities including logging in, logging out, posting news stories, fetching news, listing news agencies, and deleting news stories.

On start-up, I defined some global variables. The session object from the request's module ensures a persistent session across multiple requests, facilitating authentication and maintaining user state throughout the session. Additionally, base_url stores the base URL of the news service API, while authenticated keeps track of the user's login status.

To handle user interaction, I implemented some functions, each responsible for a specific task. For instance, the login function prompts the user for their username and password, sends a POST request to the login API endpoint, and sets the authenticated flag to True upon successful authentication. The logout function terminates the user session by sending a POST request to the logout API endpoint and setting authenticated to False.

Users can perform various actions such as posting news stories, fetching news, listing agencies, and deleting stories. These functionalities are encapsulated in respective functions that construct appropriate HTTP requests and handle responses from the API endpoints.

The main function is the entry point. It greets the user with a welcome message, enters a loop to continuously prompt for user input, interprets commands, and executes corresponding functions. In case of errors or invalid commands, it provides informative feedback to the user. Upon completion, the application closes the session, ensuring proper resource management.