**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

# Introduction

Explain how much of the coursework you have been able to implement (for example, have you been able to implement all features or just the database).

Clearly state that you have uploaded the Django server code to pythonanywhere.com and that you have written the client in Python 3.x and thoroughly tested it.

Also, state the following:

1. The URL of your pythonanywhere account.
2. A superuser name and password that the assessor of your coursework can use to access the admin site of your service.

(Restrict this section to a maximum of half a page)

# 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.

# The APIs

**Log In**

The /api/login endpoint handles POST requests for user authentication. It extracts the username and password from the request payload and attempts authentication using Django's authenticate function. If successful, the user is logged in; otherwise, appropriate error responses are returned. Error handling ensures correct HTTP status codes and error messages are returned according to the specifications.

**Log Out**

The logout API in Django is implemented at `/api/logout`. It handles POST requests, checks the user's authentication status, and logs out the user if authenticated using Django's logout function. Successful logout returns a `200 OK` response with a goodbye message, while failed logout attempts return a `403 Forbidden` response.

**Post a Story**

The API endpoint /api/stories handles POST requests for posting stories. It checks user authentication, processes incoming JSON data for story details, and stores them in the database. Error handling manages exceptions during database operations, providing appropriate responses. Successful postings return HTTP status 201 (CREATED), while errors return status 503 (Service Unavailable).

Clearly yet briefly explain how you implemented your APIs.

(Restrict this section to a maximum of one page)

# The Client

Clearly but briefly explain how you implemented the client.

(Restrict this to a maximum of one page)