National College of Ireland

Computing Project

Software Development

2022/2023

Sean Fulton

x19518013

x19518013@student.ncirl.ie

Pressfeed

Technical Report

Contents

Executive Summary

This document outlines the implementation of Pressfeed, a social media focused news aggregation web application built using the Python based web framework Django. Throughout this document I will outline the requirements specification of the application's development, UML diagrams, explanations of core functionality code snippets, testing and continuous integration approaches. All technologies included are listed with clear reasons for why they are implemented in the application and what purpose they provide to the application.

The forefront of Pressfeeds goals are to deliver a platform for users to curate the news they would like to keep up to date on, offering features such as a direct message system to allow discourse on articles recommended by the system, focusing on a younger demographic Pressfeed aims to attract frequent users of social media in which they would rely on to stay up to date on news, Pressfeed provides a way to keep the features of social media platforms but deliver real world news sources to its users.

The aim of this document is to showcase the methodologies and approaches taken to implement such an application while also looking at the future of the product.

1.0 Introduction

1.1. Background

I chose to create Pressfeed as I feel there is a growing need for real news sources to be available to people through the medium of a social media platform. A lot of the news people are consuming today are coming from unreliable user accounts that frequent the feeds of Twitter, Facebook and Instagram and can even spread misinformation on topics.

In 2020 a survey was conducted by the Trusted Web Foundation on a group of 1000 people in Europe, with ages ranging from 16 to over 54:

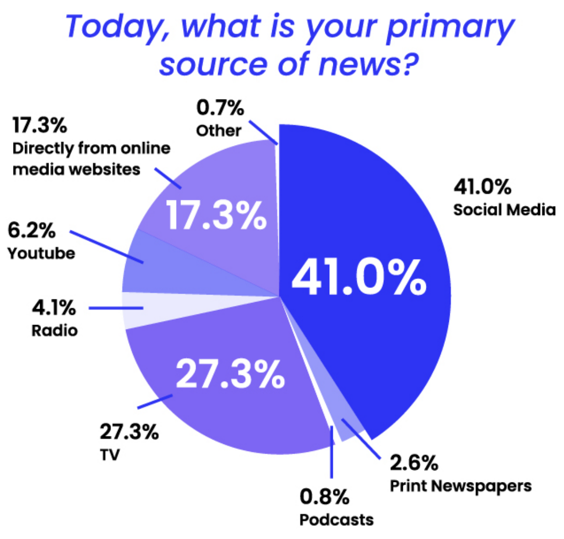


Figure 1 – Survey of users primary news source

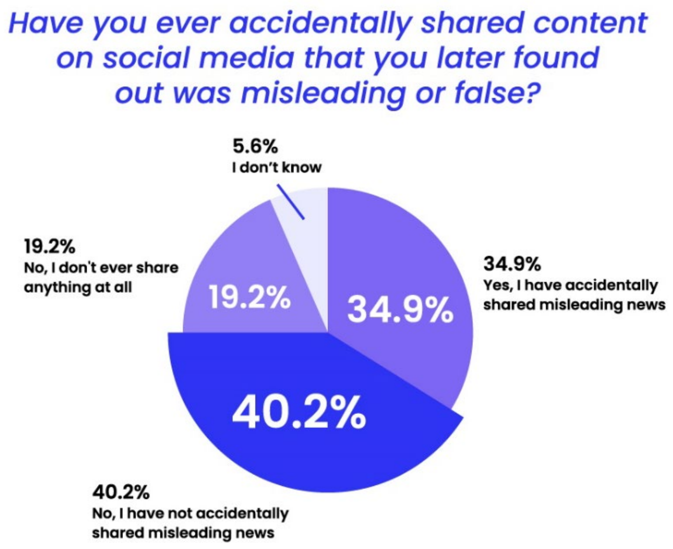


Figure 2 - Survey of accidental sharing of misinformation,

Figures 1 and 2 are pie charts on the results of a study that concluded people of all ages even the young demographic of users of social media have difficulty identifying misleading news through their use of various social media platforms. Roughly 38.2% of people claim they have reshared false information on social media. (Trusted Web Foundation, 2021)

A core goal for Pressfeed is to mitigate the amount of misinformation being shared by people online without removing the social discourse functionality provided by current social media platforms.

1.2. Aims

I believe a solution such as Pressfeed can lead to more accurate information being available to users by providing direct sources to users while still providing social media features on other platforms to allow for the discourse of topics users of Pressfeed can subscribe to.

1.3. Technologies

Software / Service	Technology
Development Environment (IDE)	Visual Studio Code
Programming Language	Python 3.11.0
Web Framework	Django 4+
Relational Database / Persistent Storage	PostgreSQL
Application Programming Interfaces	NewsAPI
Hosting	Fly.io
Front-End Technologies	Bootstrap 5
	JavaScript
Version Control	Git
	GitHub

Django - Python rest web framework:

Django is a fast web framework for the Python programming language it offers a fast development cycle for developers to get web applications from design to code efficiently. I chose Django to quickly develop Pressfeed's and easily interconnect both backend and frontend functionality.

Python (3.11.0):

Python is an object oriented programming language. I will be using Python version 3.11 to implement the majority of the application. This is a great choice for me as a developer due to Python natively supporting a Windows 11 environment which I am using for Pressfeeds development. Python excels in backend development which will be a core focus of handling the logic for my PostgreSQL database models with the purpose of the delivery of news data to Pressfeeds users.

NewsAPI:

NewsAPI advertised as a replacement for the deprecated Google News API is the primary news article aggregator implemented into Pressfeed, this API returns metadata on articles in a JSON format, which is easily manipulated using Python and Django. This allows Pressfeed to seamlessly deliver news data in a well-structured format alongside the Bootstrap CSS framework.

PostgreSQL:

PostgreSQL is an open source object-relational database that uses and extends the SQL language. I have chosen PostgreSQL as the database system for Pressfeed due to both Django and my deployment platform (Fly.io) supporting PostgreSQL databases. My application requires a way to retain news data from NewsAPI , and also User authentication data. My solution to this is to design a database structure utilising PostgreSQL allowing me to perform fast and effective queries on Pressfeeds data due to PostgreSQL incorporating a vast number of indexing algorithms.

Git, GitHub & GitHub Actions (Version Control)

Pressfeeds implementation is strictly managed using version control software

1. Git:

   Git is a distributed version control system which in the development of Pressfeed I am using to track the changes of its implementation and to easily revert to older version of the application

2. GitHub:

   GitHub is a web-based hosting platform for Git repositories. Since I am the sole developer of Pressfeed the main features of GitHub I use is for backup purposes. In the event of losing the application from my local development environment I can always retrieve my last pushed commit to my remote repository on GitHub. Branche Pull Requests are also a very useful feature to run checks (builds, test plans) on new features before merging to my main branch

3. GitHub Actions

   GitHub Actions is the Continuous Integration & Deployment platform of Pressfeed. GitHub Actions automates the workflow jobs I have defined for both the build criteria and Deployment steps for Pressfeed.

1.4. Structure

The structure of this document is to detail the key headings below:

- 2.0 System – The System section contains all implementation and design models of the application

    - 2.1 Requirements – outlines the functional and non-functional requirements of the system
        - 2.1.6 Design & Architecture - details the design processes undertaken to discover the best implementation of the application.
        - 2.1.7 Implementation – Documents the final implementation of the application and its core systems
        - 2.1.8 – Graphical User Interface – Screenshots of the final implementation of the application
        - 2.1.9 – Testing – Documents the testing techniques undertaken to ensure the stability of the application through its development lifecycle
        - 2.1.10 – Evaluation – A final evaluation of the application's performance and the more efficient approach to news aggregation.

- 3.0 Conclusions – Conclusions outline the analysis of the final implementation of application and the overall concepts the document has demonstrated

- 4.0 Further Development or Research – Outlines the paths Pressfeed can take on further development of the application and what areas were researched to improve the applications implementation.

- 5.0 References – Contains all cited sources which are used in the application and document.

- 6.0 Appendices

    - Project Proposal – includes the updated application proposal for Pressfeed.
    - Ethics Approval Application – outlines the ethical aspects of the application and approval from external sources.
    - Reflective Journals – includes all reflective journals which keep track of Pressfeeds development on a monthly basis

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1. Use Case Diagram

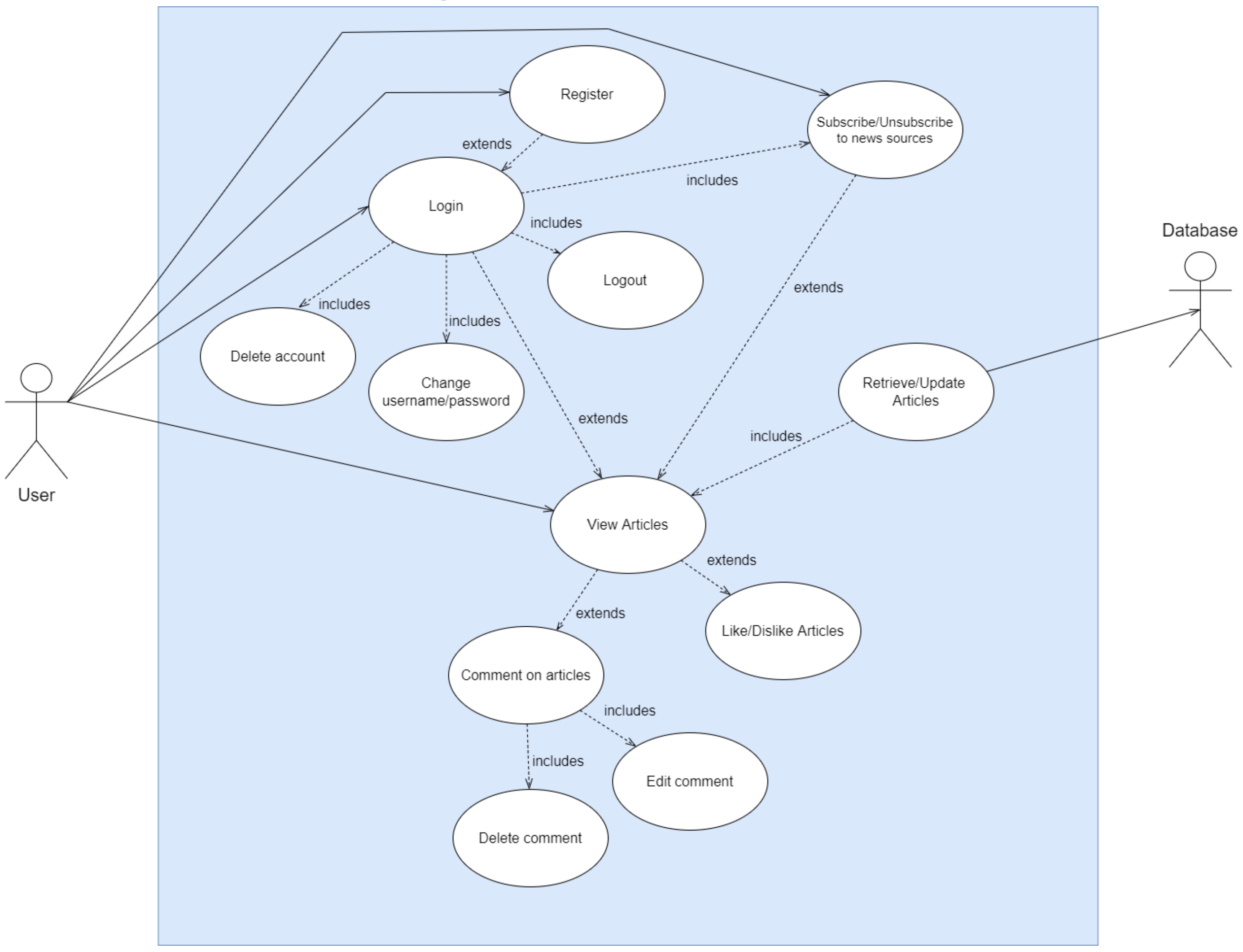


Figure 3 – HIGH LEVEL USE CASE DIAGRAM

Figure 3 outlines the top level use case diagram of the application. The diagram outlines all Actors, which include the User, System, and Database and the main uses cases I intend to implement within the application. The purpose of this diagram is to provide a structure for user functionality and to aid in identifying the functional requirements to ensure users can effectively use the application.

2.1.1.2. Requirement 1 – User Authentication

2.1.1.1.1. Description & Priority:

[PRIORITY MEDIUM]

User Authentication should allow Users to register an account with a unique username and password, login to their accounts with those credentials, change their account details and delete their accounts if they choose to.

User Authentication should be secure and ensure the correct validation of the User’s username and password inputs.

2.1.1.1.2. Use Case – Register:

Description	This use case describes a user registering an account through the System’s frontend
Scope	Registering allows users to create a new account on the System with a username and password. It does not allow the user to login, delete their account, or change their username or password
Pre-Condition	➢ The User must be logged out
Activation	➢ The User navigates to the Register page by clicking the ‘Register’ link located on the navbar of the System.
Main Flow	1. The User clicks the ‘Register’ link on the navbar of the application 2. The System returns the Register view and template 3. The User enters a unique username 4. The User enters valid password: • Password must be no less than 8 characters • The password should not contain the Users username • It must not be entirely numeric • It should not be a commonly used password 5. The User confirms their password

	6. The User clicks the Register button 7. The System creates a new user record with the entered credentials and saves the Users password using a salting / hashing algorithm to store passwords 8. The User is logged in, redirected to the home page and a success message is displayed
Alternate Flow	N/A
Exceptional Flow	• Not unique username: 1. The user navigates to the Register view 2. The User enters an already used username 3. The User fills out both password fields correctly 4. The User clicks the Register button 5. The System redirects to the Register page and an error message is displayed indicating that the User's chosen username is already used • Passwords do not match: 1. The User navigates to the Register page 2. The User enters a unique username 3. The user enters password 4. The user does not enter the same password for the 'confirm password field' 5. The user clicks the register button 6. The System redirects to the Register page and an error message is displayed indicating that the User's enter password fields do not match.
Termination	• The system completes User registration by redirecting the User to the home page when the User clicks the Register button • User registration terminates unsuccessfully when the user enters insufficient criteria for the fields. When clicking the Register button, the System will redirect to the Register page

Post Condition	The System either successfully creates a new user record in the Database user's

2.1.1.1.2. Use Case – Login:

Description	This use case describes the User's ability to login to an existing account registered on the System
Scope	• The Login use case allows users to Login to an existing account • It does not allow users to Register, • Changing username/password is included in this use case • Deleting a user account is included in this use case • Must be logged in to Logout
Pre-Condition	• The user must be logged out
Activation	• Login begins once the User navigates to the login page when logged out of the System
Main Flow	1. The User navigates to the login page 2. The User enters their login credentials 3. The User clicks the Login button 4. The System authenticates the User 5. The user is successfully logged in
Alternate Flow	N/A

Exceptional Flow	• Invalid username 1. The User navigates to the Login page 2. The User enters an invalid username 3. The User enters their password 4. The User clicks the Login button 5. The System redirects to the Login page and displays an error messaging indicating the login credentials provided were incorrect • Incorrect password 1. The User navigates to the login page 2. The User enters an existing username 3. The User enters an incorrect password 4. The System redirects to the Login page indicating the login credentials provided were incorrect
Termination	The System successfully authenticates the user or returns an error message due to incorrect login credentials provided
Post Condition	The System has successfully activated a session with User account

2.1.1.1.3. Use Case – Logout:

Description	This use case describes the User's ability to Logout of their currently active account
Scope	• The Logout use case allows users to Logout of their logged in account and destroy their user session
Pre-Condition	The System can logout a user account when a current user session is active
Activation	• The User clicks the logout button

Main Flow	1. The User clicks the logout button 2. The System destroys the active User session
Alternate Flow	N/A
Exceptional Flow	N/A
Termination	• The User clicks the logout button
Post Condition	• The System destroys the User sessions

2.1.1.1.4. Use Case – Change Username/Password:

Description	This use case describes the editing of both the Users Username and Password
Scope	• This use case enables the functionality of editing a User's username and password
Pre-Condition	• The User has logged in and has an active user session
Activation	• The User navigates to the account page • The either updates their username or password by filling out their respective fields and clicking the change username / change password button

Main Flow	1. The User logs in 2. The User navigates to the account page 3. The User: a. opts to change their username b. opts to change their password 4. The User submits the changes 5. The System saves the updated User information
Alternate Flow	N/A
Exceptional Flow	• Updated username is not unique: 1. The username the User has chosen is not unique 2. The User attempts to submit their changes 3. The System refuses and displays an error message indicating the username was not unique • Update passwords did not match 1. The password and confirm password fields for the new password were not matching 2. The User attempts to submit their changes 3. The System refuses and displays an error message indicating the users password fields did not match
Termination	• The System updates the username or password for the User successfully • The System throws an error and displays it to the user on the frontend
Post Condition	• The Users username and/or password is updated successfully • The Users username and/or password is not updated successfully due to invalid inputs

2.1.1.1.5. Use Case – Delete Account:

Description	This use case describes the functionality for Users to be able to delete their account and remove all of their User data related to their account
Scope	• This use case completely removes all data of a User upon the users request
Pre-Condition	• The User must be logged in.
Activation	• The User navigates to the User account page
Main Flow	1. The User clicks the delete account button on the account page. 2. The User is prompt to confirm or cancel their request by The System 3. The User clicks 'Confirm' 4. The System successfully deletes the User's account data 5. Redirects the User back to the home page 6. Displays an message indicating the account has been deleted successfully
Alternate Flow	• The User cancels account deletion: 1. The User navigates to the User account page 2. The User clicks the delete account button 3. The User is prompted to Confirm or Cancel account deletion 4. The User clicks Cancel 5. The System aborts account deletion action
Exceptional Flow	N/A
Termination	• The User's account deletion process has completed after the user clicks the delete

	account button and either chooses to confirm or cancel their request
Post Condition	• The User's account has been successfully deleted by the System • The System has aborted the deletion of the User's account

2.1.1.3. Requirement 2 – News Aggregation

2.1.1.2.1 Description & Priority:

[PRIORITY HIGH]

The core functionality of Pressfeed lies within its processes of news aggregation. How I aim to achieve this will be outlined in this requirement. Pressfeed needs to deliver news articles to its user's in a streamlined manner, with it easy to navigate through the many articles and sources that will be available to users upon using the system.

2.1.1.2.2. Use Case – Retrieve/Update Articles:

Description	The ability for the System to aggregate to Article metadata from NewsAPI and store it in the Database
Scope	• The System and Database are the only actors of this use case, the User does not interact with processes of news data retrieval within the System
Pre-Condition	• The System must be running • A Database connection must be established • NewsAPI endpoints must be live
Activation	• When the System starts • Scheduled task should execute article retrieval on a regular basis throughout the day
Main Flow	1. The System is running and executes article retrieval from NewsAPI 2. Transforms JSON response into SQL Model to be persisted in the Database 3. While the System is running a scheduled task should retrieve news periodically

Alternate Flow	N/A
Exceptional Flow	• A Database connection is lost 1. The System attempts to execute retrieval of articles. 2. The Database connection cannot be established 3. The System throws an error due to it being unsuccessful of retrieval of article data
Termination	• The System is terminated
Post Condition	• The System persists the previous retrieval of news data until being executed

2.1.1.2.3. Use Case – Subscribe/Unsubscribe to news sources:

Description	This use case describes the functionality to allow Users to select which news sources available on the System they would to subscribe or subsequently unsubscribe to.
Scope	• This use case allows users to choose which sources they would like to see on their feed page • This use case dictates which articles will be shown to users in the View Articles use case
Pre-Condition	• The User must be registered and logged in to the System
Activation	• The User navigates to the Subscriptions page
Main Flow	1. The User's account is logged in and active 2. The User navigates to the Subscriptions 3. The User chooses which articles to subscribe / unsubscribe to 4. The User submits their changes 5. The System updates the Database to include which sources the User is subscribed to

Alternate Flow	N/A
Exceptional Flow	• No news sources are available in the database 1. The User navigates to the Subscriptions page 2. The User cannot subscribe nor unsubscribe to articles due to their none being available
Termination	• The User submits the updated subscription list
Post Condition	• The System persists the Users subscriptions

2.1.1.2.4. Use Case – View Articles:

Description	This use case describes the functionality for User's to be able to view the available articles from news sources they have subscribed to
Scope	• The scope of this use case is to provide Users with the functionality to view articles available to the System & Database • Comment on articles, edit comment, like/dislike articles are extensions of this use case
Pre-Condition	• The User must be logged in • The User must have subscribed to at least one news source
Activation	• The User navigates to their Feed page
Main Flow	1. The User navigates to their Feed page 2. The User can view articles from their subscribed news sources

Alternate Flow	• The user has not subscribed to any news sources 1. The User navigates to their Feed page 2. The System indicates to the user there are no articles to display due to them not being subscribed to any news sources 3. The User navigates to their Subscriptions page 4. The User updates their news sources 5. The User navigates to the Feed page 6. The User can now view articles from their newly subscribed to news sources
Exceptional Flow	N/A
Termination	• The user navigates away from the Feed page
Post Condition	• The System does not query anymore articles from the Database and serves the newly selected use case to the user

2.1.1.3. Requirement 3 – Social Functionality

2.1.1.3.1 Description & Priority:

[PRIOTITY MEDIUM]

The Pressfeeds goal to bring social discourse to accurate news sources lies within its requirement to provide similar features to popular Social Media applications. This requirement details the functionality of how Pressfeeds Users can interact with Articles provided by the System and provide a reaction through a like/dislike system, and also give the users extended functionality to provide their opinion through the ability to leave comments on articles. These features should not be forced upon User's in the case they would like to use the application for pure news aggregation purposes

2.1.1.3.2. Use Case – Like/Dislike Articles:

Description	This use case describes the ability for Users to be able to provide their reaction by leaving either a Like or Dislike on articles
Scope	• Like / Disliking allows users to select whether they Like (approve of) or Dislike (Disapprove of) an article and display their reaction to other users of the System
Pre-Condition	• The User must be logged in

Activation	• The User clicks either Like or Dislike button on an article
Main Flow	1. The User navigates to the Feed page 2. The User clicks the Like or Dislike button on an article 3. The System logs there logs there like / dislike and subsequently displays it in the pool of total likes / dislikes the article has accumulated
Alternate Flow	• User likes or dislikes from the detailed view of the article: 1. The User navigates to the Feed page 2. The User clicks the view comments button 3. The User either clicks the Like or Dislike button 4. The System saves their reaction • User removes their existing reaction 1. The User navigates to an article they have previously liked or disliked and removes their old reaction by clicking the same reaction which 2. The System removes their reaction • User changes their reaction 1. The User navigates to an article they have previously liked or disliked and changes their reaction by clicking the opposite reaction 2. The System updates their reaction
Exceptional Flow	N/A
Termination	• The User leaves the article view or Feed page after leaving a reaction, changing or removing a reaction
Post Condition	The System updates the respective action to the reacted to articles by the User

2.1.1.3.3. Use Case – Comment on articles:

Description	This use case describes the functionality for User's to comment on respective articles available to them by the System
Scope	• The scope of this use case is to allow user to leave comments on articles • Editing of comments and deleting comments use cases are included within this use case
Pre-Condition	• The User is logged in • There are Articles available to the System in the Database
Activation	• The user navigates to the comment / detailed page of the article
Main Flow	1. The User navigates to the Feed page 2. The User clicks on the comment button attached to any of the articles displayed 3. The User writes their comment in the text box under the article 4. The User clicks the send/comment button 5. The System returns the detailed view of the article 6. The user leaves a comment on the article
Alternate Flow	N/A
Exceptional Flow	N/A
Termination	The System successfully saves the comment to the Database comment model
Post Condition	The System displays the comment on the detailed article view page

2.1.1.3.4. Use Case – Edit comment:

Description	This use case describes functionality for Users to be able to Edit a previously submitted comment on an article
Scope	• This use case allows Users to edit the text of the comment they have left on an article previously • This use case does not allow you to delete a comment (Handled by Delete comment use case)
Pre-Condition	• The User must be logged • The System must have available articles to comment on • The User must have commented on an article
Activation	• The User clicks the comment button on an article on the Feed page
Main Flow	1. The User navigates to the detailed view of an article that they have previously left a comment on 2. The User clicks the edit button on one of their comments 3. The User updates their comment in the text area 4. The User submits the changes by click the check mark button 5. The System updates the comment
Alternate Flow	• The user edits a comment but cancels the action 1. The User navigates to an article they have previously left a comment on 2. The User clicks the edit button on one of their comments 3. The User changes their mind and clicks the cancel / x button on the comment box
Exceptional Flow	N/A
Termination	• The User either submits their changes or closes the edit form

Post Condition	• The System updates the edited comment and displays the original detailed view of the article

2.1.1.3.5. Use Case – Delete comment:

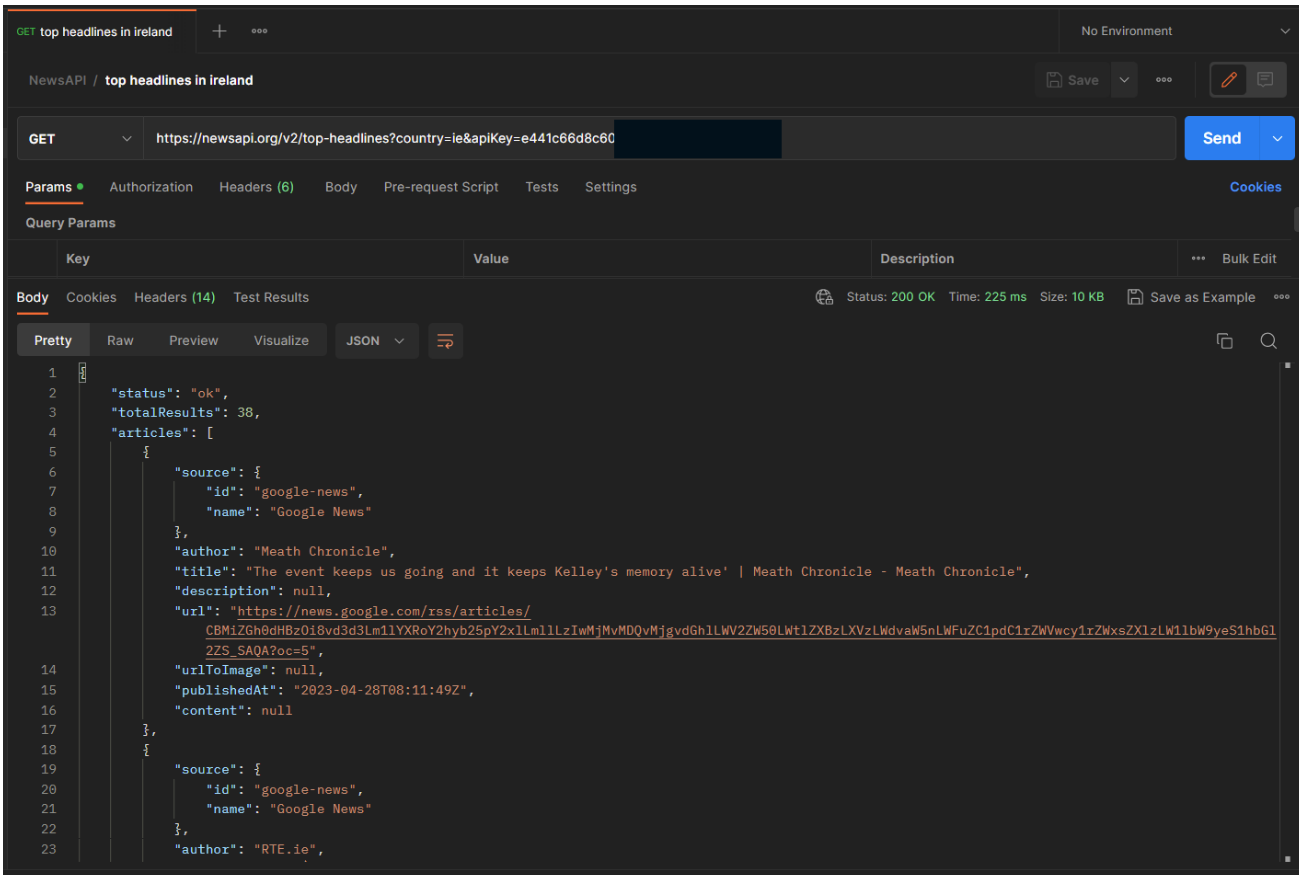
Description	This use case describes the functionality to enable Users to delete a previously left comment on an article
Scope	• The scope of this use case allows user to delete their own comments on articles
Pre-Condition	• The User must be logged in • The User must have commented on an article
Activation	• The User navigates to the comment view of an article they have commented on previously
Main Flow	1. The User navigates to a comment they have left on an article 2. The User clicks the delete comment button on their comment 3. The User is prompted to confirm or cancel the deletion process 4. The User clicks confirm 5. The System deletes the comment record from the Database 6. The page is refreshed and displays the newly updated comments on the detailed article view
Alternate Flow	• The User cancels the delete comment process 1. The User navigates to a comment they have left on an article 2. The User clicks the delete comment button on their comment 3. The User is prompted to confirm or cancel the deletion process 4. The User clicks the cancel button 5. The System aborts the delete comment process

Exceptional Flow	N/A
Termination	• The System either aborts the deletion process or deletes the comment from the Database comment model
Post Condition	• The System displays the detailed article view of the comment

2.1.2 Data Requirements

Pressfeed's implementation will require a Database Management System (DBMS). I have researched industry standard DBMS and have chosen PostgreSQL for the Systems data storage capabilities due to the Postgres being an Object-Relational SQL Database Management System. This allows me to form key relationships with the systems data. This will allow the system to accurately and efficiently query article data. Postgres offers many more indexing algorithms for data when compared to other DBMS such as MySQL. Indexing can be very useful when data needs to retrieved frequently in a specific order, the System will be delivering data most of the time based on date to display the most recent article (descending order) in the Systems database. Indexing the models in the Systems database, based on article date columns will allow the system to query articles more efficiently.

The system requires a method of collecting news metadata. There are various services and tools which can scrape news data from the internet which can come in the form of scraping libraries and API's. When I conceptualised Pressfeed I had researched the Google News API which I planned to be the primary source of the systems news data source. However this API is not available to the public for development purposes, another API I have looked into which is advertised as a replacement for Google's News API is NewsAPI. NewsAPI returns metadata on news articles in a JSON format which is updated daily.



https://newsapi.org/v2/top-headlines?country=ie&apiKey=e441c66d8c60[redacted]

```
{
    "status": "ok",
    "totalResults": 38,
    "articles": [
        {
            "source": {
                "id": "google-news",
                "name": "Google News"
            },
            "author": "Meath Chronicle",
            "title": "The event keeps us going and it keeps Kelley's memory alive' | Meath Chronicle - Meath Chronicle",
            "description": null,
            "url": "https://news.google.com/rss/articles/
                CBMiZGh0dHBzOi8vd3d3Lm1lYXRoY2hyb25pY2xlLmllLzIwMjMvMDQvMjgvdGhlLWV2ZW50LWtlZXBzLXVzLWdvaW5nLWFuZC1pdC1rZWVwcy1rZWxsZXlzLW1lbW9yeS1hbGl
                2ZS_SAQA?oc=5",
            "urlToImage": null,
            "publishedAt": "2023-04-28T08:11:49Z",
            "content": null
        },
        {
            "source": {
                "id": "google-news",
                "name": "Google News"
            },
            "author": "RTE.ie",
```

Figure 4 NewsAPI Request to top-headlines endpoint

Figure 4 is a GET request I initiated to NewsAPI's 'Top Headlines' endpoint through Postman. I have given the country parameter a value of 'ie' to indicate I would like to target articles in Ireland. The API response structure shown returns the response status, the total length of the article list 'totalResults' and the list of articles. Each article element contains a:

- Source
    - ID
    - Name
- Author
- Title
- Description
- The articles URL
- URL to the articles header image
- its published date
- Content

Understanding the structure of how NewsAPI will return news metadata based on requests to its endpoints I can begin designing the database structure and requirements to effectively persist request data from the API. This is one of the core reason the system requires a database management system. Calls to an API can be very inefficient and performance taxing to the system and can even cap requests to the API, this is the case with NewsAPI as a developer I can only request the API 100 times per day. If I use a DBMS to serve articles to Pressfeeds User's this will not incur more requests than the system requires. Scheduling a task to call the api a few times a day can maximise API request efficiency and keep the database up to date with news data.

2.1.3 User Requirements

Users of Pressfeed require a client device of some form which can receive and initiate HTTP requests. This will most often be performed by a Browser application on either a computer or mobile device. User's should be able to access the application without any setup before hand and by simple navigating the web applications URL.

Users should be first greeted with a landing page explaining the purpose of the system i.e. news aggregation functionality and social discourse features and also provide a sample of the news aggregation technology to demonstrate to users how it is achieved and delivered to the User. Users should have the ability to either register or login if they have an existing account. Upon registering / logging in the user should be authenticated and now have access to the functionality to view and manage their subscriptions to news sources, Users should be able view their feed of articles from sources they are subscribed to also. Users should be able to manage their account upon logging in

All of these functional requirements should be streamlined to the user and not challenging to understand how to interact with the system.

This leads into the system ensuring a stable performance under load of users, with quick page request times with minimal to zero lag of UI elements and page functionality to users. The system should be reliable and should be available to users at all times with minimal or if possible no downtime.

2.1.4 Environmental Requirements

Deployment:

Pressfeed's external factors for its implementation includes for where in the world the deployment platform hosts the application and database. Currently I have chosen Fly.io as the deployment platform for Pressfeed, Fly offers locations all around the world to deploy web applications, I believe hosting the application in London as this is offered by the service and it being the closest deployment location to Ireland. This should result in faster connection to Irish and UK users

Sourcing of news metadata:

Another crucial external service which the system relies on is the access to and availability of NewsAPI. This will be the applications sole source of news data as of writing. This service will need to be available during times the system requests its endpoints to keep the systems database up to date to ensure users are able to access to a steady flow of up to date news

There are options to explore in future to include fallback services / technologies which can substitute for the sourcing of news metadata for the application. Since Pressfeed is built on the Django framework, there is a Python library known as Beautiful Soup, Beautiful Soup is a HTML & XML web parser which could be implemented in the system to be developed into a fallback service if and when NewsAPI fails.

Optimising API endpoints:

NewsAPI offers various endpoints which the system can call to request news metadata, each endpoint requires a specific country code in which sources are based to return metadata on articles from, with my current developer account I must adhere to the 100 requests per day quota on requests to NewsAPI. Due to this I will have to carefully consider which countries will grab the most popular and desired news sources for the applications userbase.

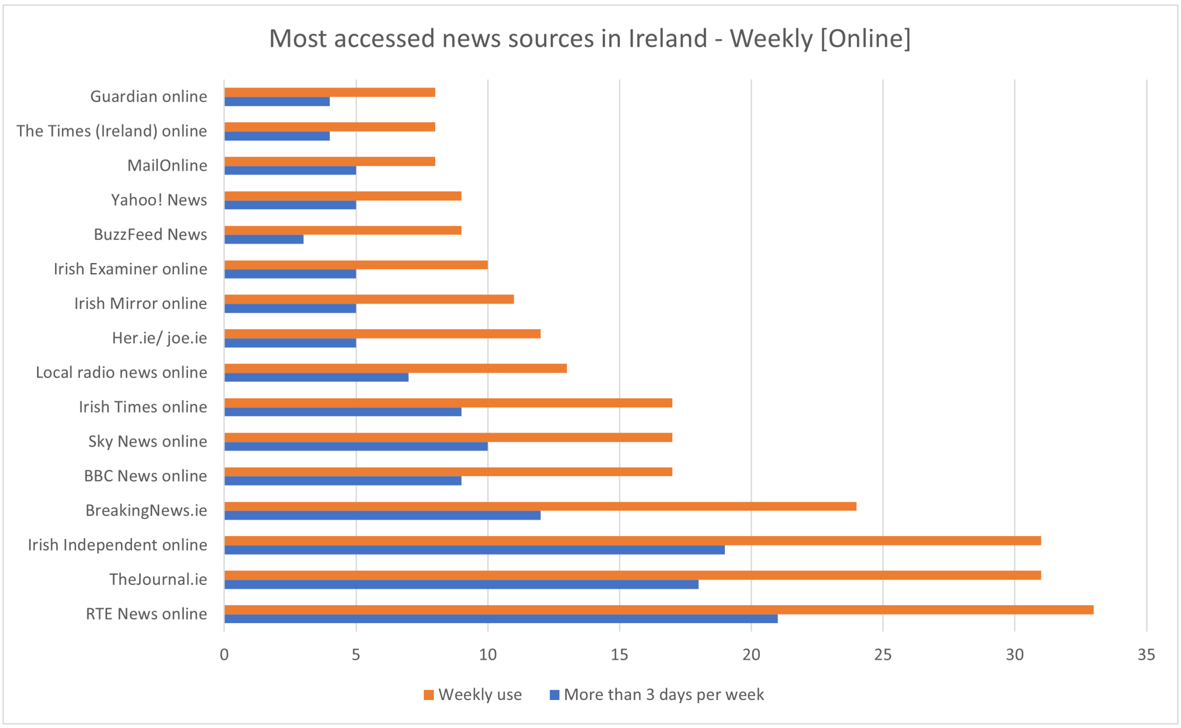


Figure 5 – Results of study on most accessed news sources in Ireland in 2020

Figure 5 is a dataset of results from a study on digital news in 2020 (Reuters Institute, 2020).

The study was conducted by Reuters Institute in collaboration with The University of Oxford, this study was conducted on the entire population of Ireland and which online news sources the Irish population most often visit on a weekly basis. The study states most often Irish citizens are most often visiting Irish news sources, with UK sources coming in second and thirdly US publications.

With these results I have plan to aggregate the 'top-headlines' endpoint of NewsAPI 3 times, with a the respective country code of Ireland, The UK, and the US.

At the moment this implementation will be very focused on the requirements of the Irish user base for the UK deployment of Pressfeed however I do intend all users from any country to be accommodated by the application. The system will be implemented in a scalable approach to easily include more endpoints and even other sources of news metadata other than NewsAPI in the future.

I believe as an Irish citizen if I can achieve an implementation which is reliable for the user of Pressfeed within my own country, I will be able to conduct the most effective evaluation of the system and apply my discoveries to a more global scale for the application in the future.

2.1.5 Usability Requirements

User Interface:

Pressfeeds user interface should enable any user, that is using a device capable of accessing a web application to be able to access and utilise the application. Since the application will include the use of Bootstrap 5 CSS framework, I can develop a responsive user interface for desktop, laptop and mobile devices to access the application.

Providing clear and concise and implementing easily accessible design patterns to the applications frontend I can ensure the application is streamlined to its users.

Error handling:

The system should indicate clearly to the user if they are trying to execute a use case or functionality of the application incorrectly. The application should return error messages which clearly indicate the reason as to why the users attempts to use the applications in the way that they are trying to may be the incorrect way of using the system.

Performance:

The application should deliver its functionality without fault and not crash from a user interacting with the system. The web server should serve the system's http responses / views to the users clients in under 5 seconds. The system should effectively execute queries to the database efficiently and not be the cause of long wait times when using the system

2.1.6 Design & Architecture

2.1.6.1. Entity Relationship Diagram

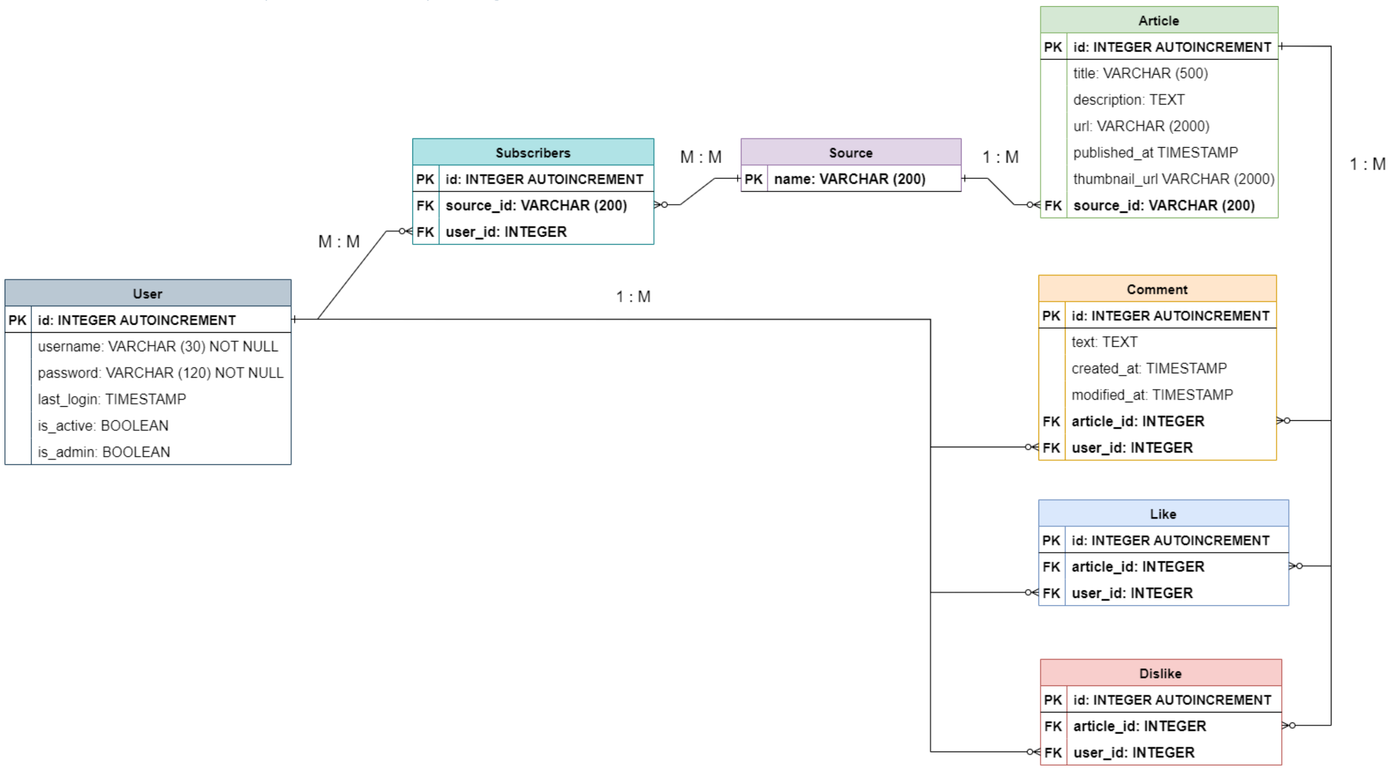


Figure 6 – Entity Relationship Diagram of the database

The Entity Relationship Diagram's purpose is to outline and identify the requirements of the database structure of the application.

I have concluded from designing the ERD that the database of the application will require these models:

1. User

   The Users table should handle the persistence of data directly related to the user i.e the user's username and password.

This model will handle the authentication of Users within the application and requires salting / hashing of passwords when storing them within DBMS.

Django can ensure this capability when forming the Users model as it utilises the PBKDF2 cryptographic algorithm with a SHA256 hash.

2. Article

   The Article model's requirements were identified with guidance of both constructing the entity relationship diagram and analysing the data requirements of the application, specifically the response content of NewsAPI.

   The purpose of the Article model is to persist the article data from the various news metadata scraping methods to give context to its data and manipulate the data in a way the application requires to deliver it to the systems users.

3. Source

   Due to the systems requirement to provide the functionality of subscribing to article sources, the database should reflect this necessity. Separating out the JSON response content from NewsAPI into two models, Source and Article, the system can easily provide the functionality of subscriptions by utilising the Source model and forming a relationship through the Subscriber model between both the User and Source models

   The system can also avoid duplicate records of sources with this structure if alternatively the system was to use a source column in the Article model.

4. Subscriber

   The purpose of the Subscriber model should be a key component when creating a relationship between Sources and which Users are subscribed to which sources through the system. The Source and User models should form the Subscriber model through a Many-to-Many relationship between User and Source models primary keys.

5. Comment

   This model was constructed to adhere to the data requirements of the comment functionality required by the system, The Comment model requires the user's and article's id's to be foreign keys in the model, this is to allow articles to HAVE comments and Users to CREATE/HAVE comments on articles.

   This models columns should contain the comments content, the TIMESTAMP it was created at, the TIMESTAMP of when the comment was modified if the user has edited one of their comments.

   The Comment model will primarily be required in the detailed view of articles.

6. Like / Dislike

   Both the Like and Dislike models purpose are to record the number of likes and dislikes an article has, each model a article id and user id column. When a user likes or dislikes an article a row is created in the respective tables to persist the number of likes an article has and to record the state of the users reaction to articles.

2.1.6.2. Class Diagram

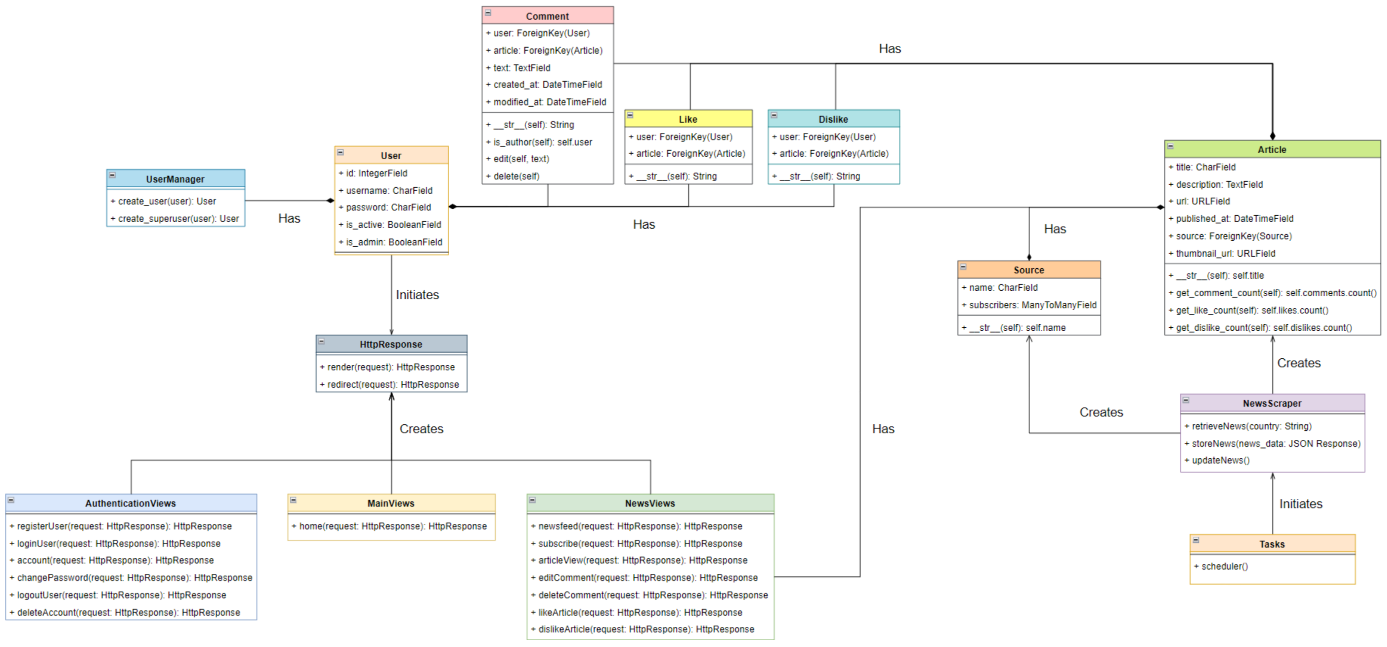

Comment
+ user: ForeignKey(User)
+ article: ForeignKey(Article)
+ text: TextField
+ created_at: DateTimeField
+ modified_at: DateTimeField
+ __str__(self): String
+ is_author(self): self.user
+ edit(self, text)
+ delete(self)
Has
Like
+ user: ForeignKey(User)
+ article: ForeignKey(Article)
+ __str__(self): String
Dislike
+ user: ForeignKey(User)
+ article: ForeignKey(Article)
+ __str__(self): String
Article
+ title: CharField
+ description: TextField
+ url: URLField
+ published_at: DateTimeField
+ source: ForeignKey(Source)
+ thumbnail_url: URLField
+ __str__(self): self.title
+ get_comment_count(self): self.comments.count()
+ get_like_count(self): self.likes.count()
+ get_dislike_count(self): self.dislikes.count()
UserManager
+ create_user(user): User
+ create_superuser(user): User
Has
User
+ id: IntegerField
+ username: CharField
+ password: CharField
+ is_active: BooleanField
+ is_admin: BooleanField
Has
Has
Source
+ name: CharField
+ subscribers: ManyToManyField
+ __str__(self): self.name
Initiates
HttpResponse
+ render(request): HttpResponse
+ redirect(request): HttpResponse
Creates
Creates
NewsScraper
+ retrieveNews(country: String)
+ storeNews(news_data: JSON Response)
+ updateNews()
Creates
Has
Initiates
AuthenticationViews
+ registerUser(request: HttpResponse): HttpResponse
+ loginUser(request: HttpResponse): HttpResponse
+ account(request: HttpResponse): HttpResponse
+ changePassword(request: HttpResponse): HttpResponse
+ logoutUser(request: HttpResponse): HttpResponse
+ deleteAccount(request: HttpResponse): HttpResponse
MainViews
+ home(request: HttpResponse): HttpResponse
NewsViews
+ newsfeed(request: HttpResponse): HttpResponse
+ subscribe(request: HttpResponse): HttpResponse
+ articleView(request: HttpResponse): HttpResponse
+ editComment(request: HttpResponse): HttpResponse
+ deleteComment(request: HttpResponse): HttpResponse
+ likeArticle(request: HttpResponse): HttpResponse
+ dislikeArticle(request: HttpResponse): HttpResponse
Tasks
+ scheduler()

Figure 7 – Class diagram of the main components of the system

Please zoom in to diagram if text is not readable!

Figure 7 outlines the proposed implementation for the core objects which will handle the operations of the system to achieve the outlined requirements.

User Management Classes:

The User and UserManager classes handle the creation and instances of User objects within the system the User class is created when the 'create_user()' or 'create_superuser' methods are called within the UserManager class, these instantiate a new User class when called and save the users credentials in the system's database User model.

The User model is used in almost every View request method inside the Views classes and is also utilised by the Comment, Like, and Dislike Model classes to assign a the correct user to each when a new record is initiate by the User

Views Classes

The 'Views' Classes (AuthenticationViews, MainViews, NewsViews) are collections of function based views which all accept a 'request' parameter this will be a HttpResponse object initiated by the Users client when the User accesses a route (URL) which is passed to the associated 'View' of that route.

For example when, a user access the home page URL of the application which is denoted by a '/' after the applications listening URL, the application will initiate an HttpResponse object, the associated view 'home' is bound to this route, the home view function based method will be executed and return a HttpResponse object with the 'template' the html file, and the context content generated by the view in a HttpResponse object to be interpreted and served to the users client by the system.

Database Model Classes:

The User, Comment, Like, Source and Article classes are the class counterparts of their respective database tables. These classes were created with the aid of the Entity Relationship Diagram as Django provides database model creation, and migrations through extending the 'Models' class which allows developers to define the database structure through the framework.

The methods included in these classes return the associated data that the system will most often be utilising. i.e the Article model has a method called get_comment_count() which returns the amount of comments that particular article object has. This method can be used to display the amount of comments an article has through a UI element.

News Aggregation Classes:

The NewsScraper and Tasks classes handle the aggregation and periodic execution of gathering news metadata from the API. The Scheduler Class calls the update_news() method in the NewsScraper class on a defined interval to ensure the application has a pool of up to date news data. This is the most integral part of the system's functionality.

2.1.6.3. Sequence Diagram

News Scraper Sequence Diagram

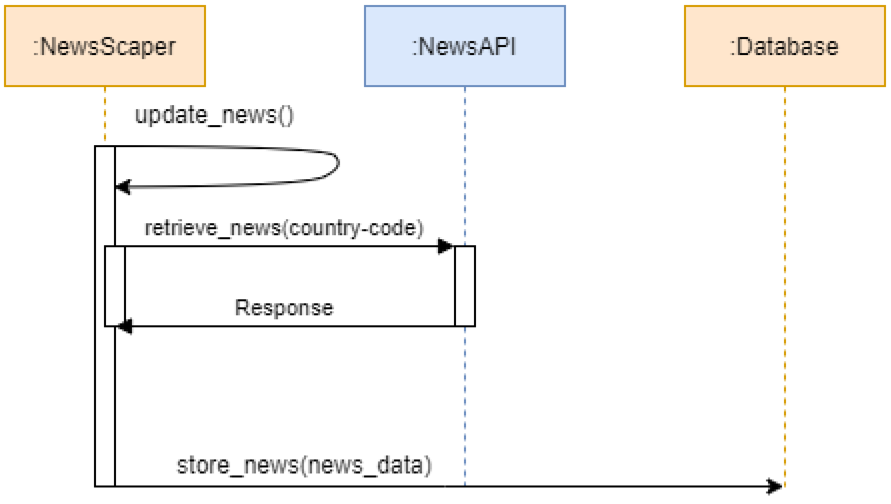


Figure 8 – Sequence Diagram

Figure 8 shows the sequence of the News Scraper class which defines three methods which handle the aggregation of news metadata:

1. Retrieve_news()
   This method accepts a country to be interpolated into the GET request method to the NewsAPI top headlines endpoint and will return a JSON object

2. Store_news()
   This method accepts a JSON object and iterates through it for every 'article' element within it. The method creates two records for each article element in it by matching up the keys to a Source and Article database record.

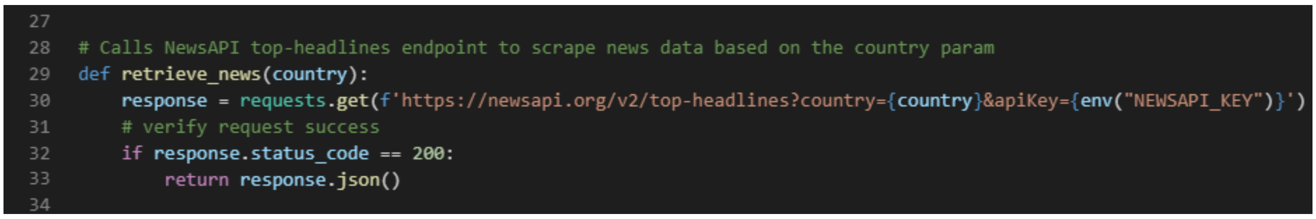
3. Update_news()

This method calls the retrieve_news() method with a specific country code to request NewsAPI with the top headlines from that country, the method then calls store_news() while passing in the news_data that was returned from the previously executed retrieve_news() method. This method allows the multiple calls of NewsAPI to gather a larger pool of news metadata by requesting the endpoint multiple times with different country codes. The application currently uses codes for Ireland, The UK and The US

2.1.7 Implementation

This section of the documentation will go over the implementation of the core functionality of system I have identified these to be the main systems to be explained over others due to their functionality being the foundation for other requirements of the system.

2.1.7.1. News Aggregation System:



```

# Calls NewsAPI top-headlines endpoint to scrape news data based on the country param
def retrieve_news(country):
    response = requests.get(f'https://newsapi.org/v2/top-headlines?country={country}&apiKey={env("NEWSAPI_KEY")}')
    # verify request success
    if response.status_code == 200:
        return response.json()

```

Figure 9 – retrieve_news method

Figure 9 is a method located in the file which contains the methods for the Systems news aggregation functionality, called News_Scraper.py.

The **retrieve_news** method was defined to allow the application to request the NewsAPI 'top-headlines' endpoint, this is to allow the system to aggregate news metadata

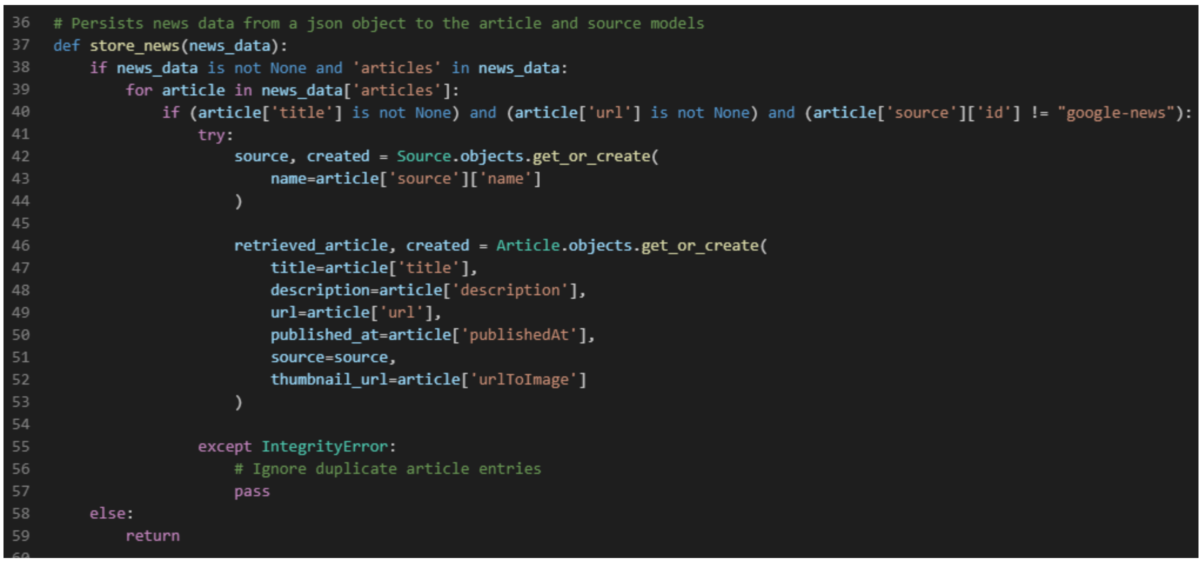
This method accepts a 'country' parameter. This parameter as proposed in my class diagram should be of a string type and is to provide a key portion of the next component of this method.

The response variable, is of a response object type and utilises the Python *requests* library, importing this package provides the application with the functionality to execute HTTP requests and here the system uses a **GET** request on the NewsAPI top-headlines endpoint.

The response variable requests the endpoint using the *.get* method of the *requests* package, this is passed an interpolated string which takes in the country parameter passed to the method when it was called, and also my personal NewsAPI key which is stored in the systems environment variables. This is to ensure that I do not commit sensitive information to the projects public GitHub repository as environment variables stored in a separate collection and ignored from the Git commit history.

The next component of this method is the return type of the method,

The method checks if the response variables status code equals '200' to indicate the request was successful, if so the method returns the response in a JSON format using the *.json* method of the *Response* object.



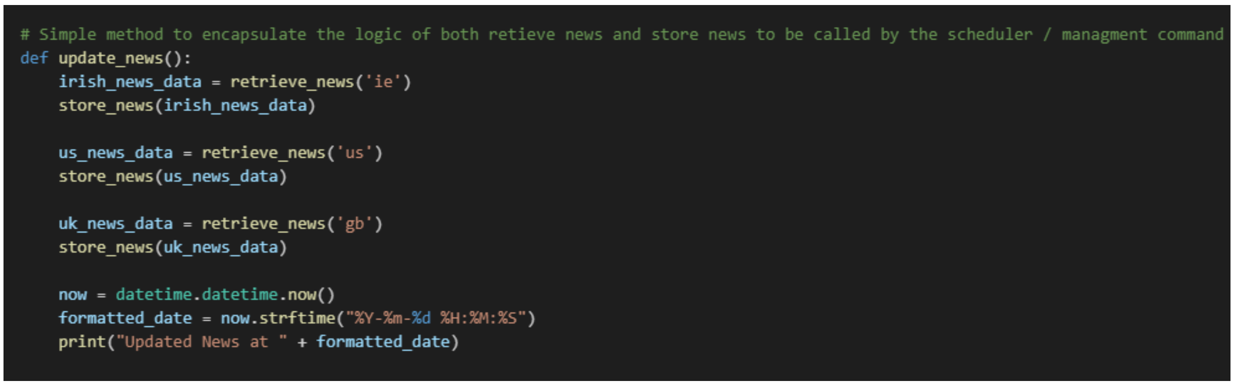
```python
# Persists news data from a json object to the article and source models
def store_news(news_data):
    if news_data is not None and 'articles' in news_data:
        for article in news_data['articles']:
            if (article['title'] is not None) and (article['url'] is not None) and (article['source']['id'] != "google-news"):
                try:
                    source, created = Source.objects.get_or_create(
                        name=article['source']['name']
                    )

                    retrieved_article, created = Article.objects.get_or_create(
                        title=article['title'],
                        description=article['description'],
                        url=article['url'],
                        published_at=article['publishedAt'],
                        source=source,
                        thumbnail_url=article['urlToImage']
                    )

                except IntegrityError:
                    # Ignore duplicate article entries
                    pass
    else:
        return
```

Figure 10 – store_news method

Figure 10 is the **store_news** method which fulfils the purpose of transforming the JSON objects into Source & Article database model records

The method accepts a **news_data** parameter which is of a JSON object. To validate if the JSON object passed to method is in the correct format of the usual returned format of a NewsAPI top-headlines response structure, The method checks if the object is first not empty and then checks if contains a key of 'articles' if these are true the method continues to loop through each article element within the article list in the JSON object and create both a Source and Article object for each article element, as long as that article has a value for a title and URL of the article and its source is not Google News this is due to Google News requiring a google account and not all users of Pressfeed should need to have to create a google account to view news.

The method then executes the JSON to model logic next which is in a try block that uses the *get_or_create* **Object** method when creating a new record in both the Source and Article models, this is to ensure duplicate articles from response objects are not entered into the database by catching the *IntegrityError* in an except clause and which will just pass over duplicate records.



```python
# Simple method to encapsulate the logic of both retieve news and store news to be called by the scheduler / managment command
def update_news():
    irish_news_data = retrieve_news('ie')
    store_news(irish_news_data)

    us_news_data = retrieve_news('us')
    store_news(us_news_data)

    uk_news_data = retrieve_news('gb')
    store_news(uk_news_data)

    now = datetime.datetime.now()
    formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
    print("Updated News at " + formatted_date)
```

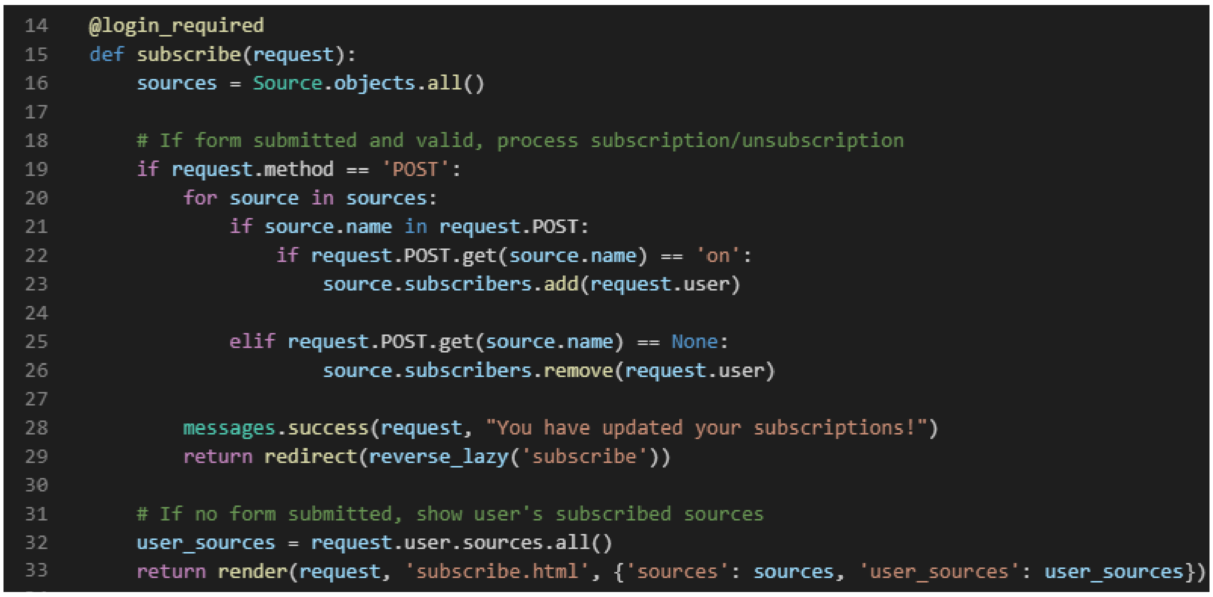
Figure 11 – Update news method

Figure 11 demonstrates that final piece of the scraping functionality of the system by combining the functionality of both the **retrieve_news** and **store_news** methods in the **update_news** method, **update_news** is used by the scheduler of the system which calls this method periodically which is set to every 4 hours per day. It begins by defining an '*irish_news_data'* variable which is a JSON object which is returned by the **retrieve_news** method with the country parameter set to the irish country code of NewsAPI ‘ie’, **store_news** is then called with *irish_news_data* passed as the **news_data** parameter to persist the response in the systems database.

update_news repeats this for both US and UK news. Once the method has initiated the logic for aggregation of news sources from all country codes it outputs the timestamp it was called with a print statement for news update logging.

2.1.7.2. Application’s Main Views / Templates

Subscribe



```python
@login_required
def subscribe(request):
    sources = Source.objects.all()

    # If form submitted and valid, process subscription/unsubscription
    if request.method == 'POST':
        for source in sources:
            if source.name in request.POST:
                if request.POST.get(source.name) == 'on':
                    source.subscribers.add(request.user)

            elif request.POST.get(source.name) == None:
                    source.subscribers.remove(request.user)

        messages.success(request, "You have updated your subscriptions!")
        return redirect(reverse_lazy('subscribe'))

    # If no form submitted, show user's subscribed sources
    user_sources = request.user.sources.all()
    return render(request, 'subscribe.html', {'sources': sources, 'user_sources': user_sources})
```

Figure 12 – Subscribe function based view

Figure 12 is a function based view is defined to deliver the functionality of the Subscriptions page through HTTP Responses.

the method accepts a request parameter as all function based views in Django do. This method is passed the HTTP request object generated from the clients GET or POST request to subscribes specific route (URL) defined in the systems *urls.py* file. Below is the URL for this view:

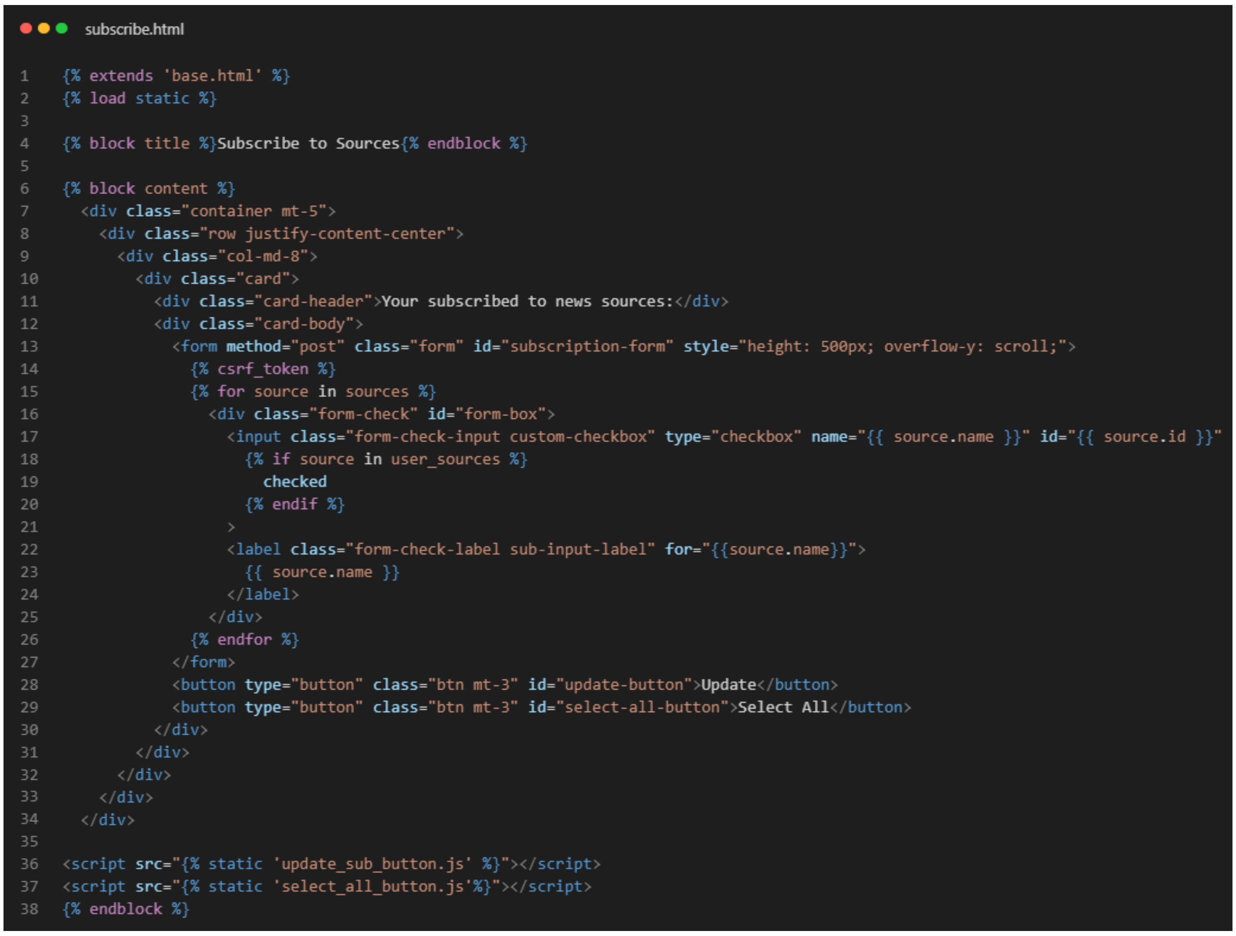
```python
path('subscribe/', views.subscribe, name="subscribe"),
```

The method first collects all the Source records from the Source database model into a ‘*sources’* variable this will be returned in the response object by the view.

If the client initiates a **GET** request to this URL the view method will return the ‘subscribe.html’ template and a dictionary passing the ‘*sources’* variable as sources and ‘*user_sources’* which is a collection of the ManyToManyField relationship formed through the request User and Sources ‘subscriptions’ column

Unless. The request method is a POST. Which is initiated through the user choosing the update news on the frontend. If the request method type is a POST the application will iterate through the checkbox forms elements to see which Sources the user has checked to subscribe to or unchecked to unsubscribe from and will update the users subscriptions accordingly

Templates:


subscribe.html

```
{% extends 'base.html' %}
{% load static %}

{% block title %}Subscribe to Sources{% endblock %}

{% block content %}
  <div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-8">
        <div class="card">
          <div class="card-header">Your subscribed to news sources:</div>
          <div class="card-body">
            <form method="post" class="form" id="subscription-form" style="height: 500px; overflow-y: scroll;">
              {% csrf_token %}
              {% for source in sources %}
                <div class="form-check" id="form-box">
                  <input class="form-check-input custom-checkbox" type="checkbox" name="{{ source.name }}" id="{{ source.id }}"
                    {% if source in user_sources %}
                      checked
                    {% endif %}
                  >
                  <label class="form-check-label sub-input-label" for="{{source.name}}">
                    {{ source.name }}
                  </label>
                </div>
              {% endfor %}
            </form>
            <button type="button" class="btn mt-3" id="update-button">Update</button>
            <button type="button" class="btn mt-3" id="select-all-button">Select All</button>
          </div>
        </div>
      </div>
    </div>
  </div>

<script src="{% static 'update_sub_button.js' %}"></script>
<script src="{% static 'select_all_button.js'%}"></script>
{% endblock %}
```

Figure 13 – the HTML template for the susbcribe page

Figure 13 is the implementation of the returned template / HTML every template in the application extends the 'base.html' template which includes the basic boilerplate setup for a HTML document, Content Delivery Network (CDN) links for Bootstrap CSS stylesheets and its JavaScript counterpart and also Font-Awesome stylesheet for stylised HTML icons.

The page specific HTML implementation of all templates other than 'base.html' relies within the *'{% block content %}'* and *'{% endblock %}'* block

This template simply renders a bootstrap styled checkbox and populates it with the retuned *'sources'* and *'user_sources'* collections from the *subscribe* view.

Newsfeed View:

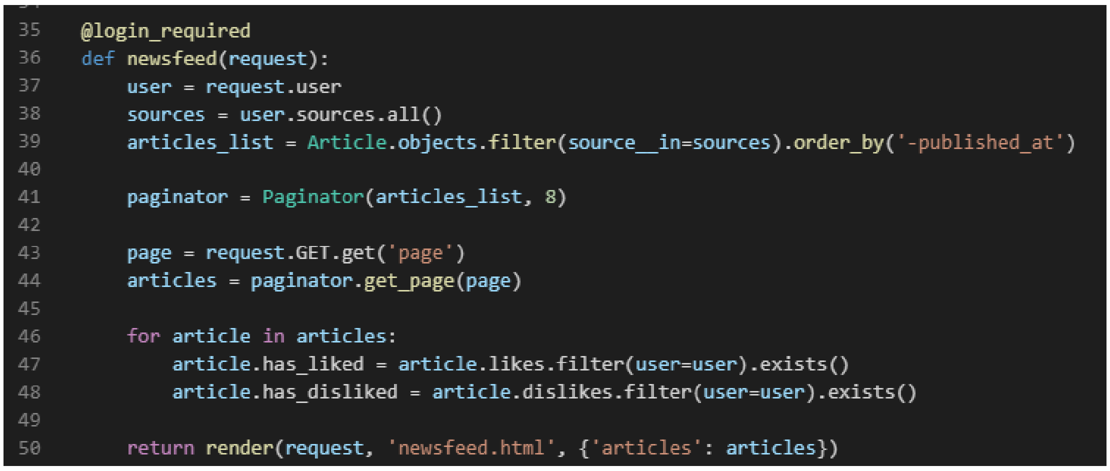

```
@login_required
def newsfeed(request):
    user = request.user
    sources = user.sources.all()
    articles_list = Article.objects.filter(source__in=sources).order_by('-published_at')

    paginator = Paginator(articles_list, 8)

    page = request.GET.get('page')
    articles = paginator.get_page(page)

    for article in articles:
        article.has_liked = article.likes.filter(user=user).exists()
        article.has_disliked = article.dislikes.filter(user=user).exists()

    return render(request, 'newsfeed.html', {'articles': articles})
```

Figure 14 – Newsfeed function based view

Figure 14 is of the newsfeed view which handles the request logic of the main functionality of the system.

The view defined the *user, sources,* and *articles_list* variables to correctly query the articles required to be returned to specific users feed page. The query used filters the Article model based on the users subscriptions and orders them by date in descending order

This view also utilises the **Paginator** class to package the articles list into a collection of pages in the event the article_list exceeds 8 articles. This is to ensure the feed page is not overwhelmed with query request time and limits the number of queries per **Paginator** page on the feed page to be 8 at a time. The user is provided with a navigation module at the bottom of the feed page to cycle through pages.

Article_view:

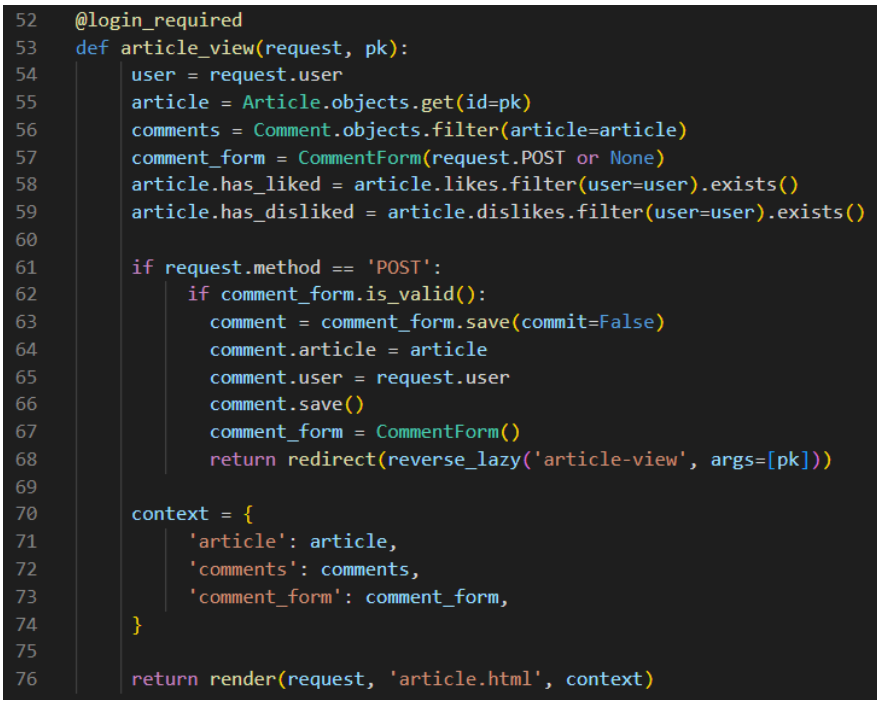

```
@login_required
def article_view(request, pk):
    user = request.user
    article = Article.objects.get(id=pk)
    comments = Comment.objects.filter(article=article)
    comment_form = CommentForm(request.POST or None)
    article.has_liked = article.likes.filter(user=user).exists()
    article.has_disliked = article.dislikes.filter(user=user).exists()

    if request.method == 'POST':
        if comment_form.is_valid():
            comment = comment_form.save(commit=False)
            comment.article = article
            comment.user = request.user
            comment.save()
            comment_form = CommentForm()
            return redirect(reverse_lazy('article-view', args=[pk]))

    context = {
        'article': article,
        'comments': comments,
        'comment_form': comment_form,
    }

    return render(request, 'article.html', context)
```

Figure 15 – Article_view function based view

The Article view allows the users to view a more detailed version of the aggregated article from their feed page. This page also enables the functionality to comment on the article.

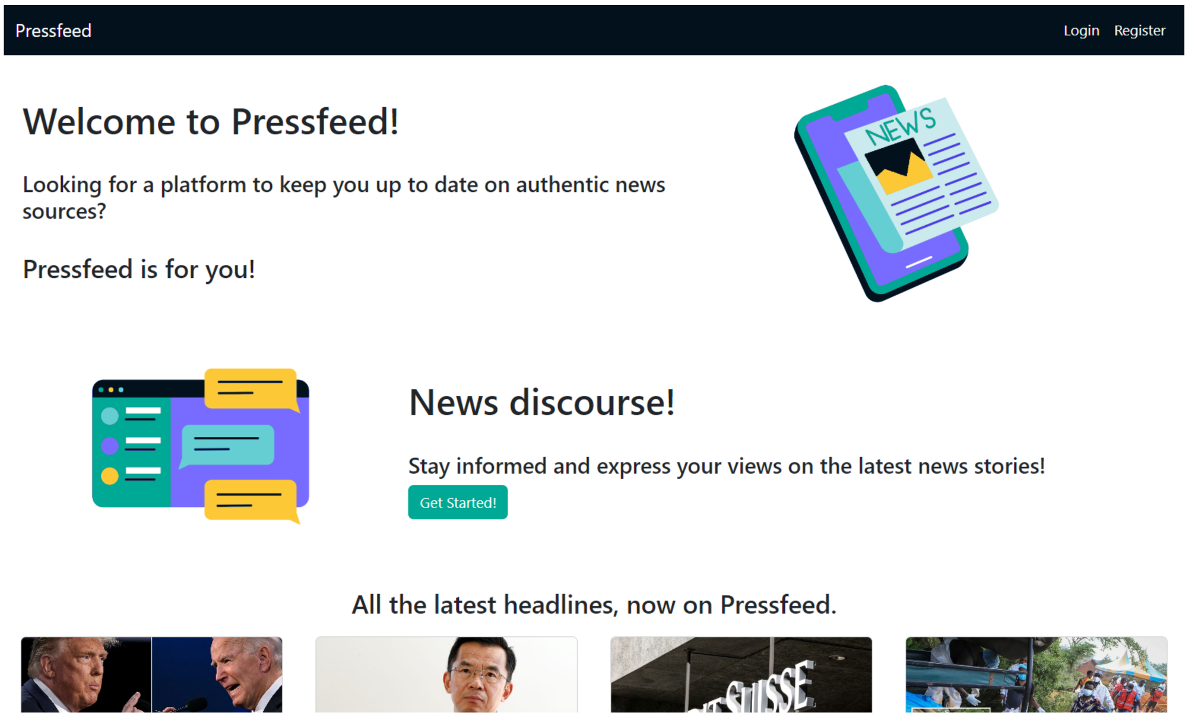
The view first defines the '*user'* variable with the user that initiates the request, the '*article'* variable with the current article that was chosen based on its id that was passed to view through the request. '*comments'* variable with the associated comments of that specific article, the article also returns the CommentForm defined in the applications **forms.py**

If the request type is a **GET** request the view simply returns the 'article.html' template and the context which includes the previously defined variables *article, comments* and the comment form '*comment_form*'

If the request type is a **POST** request the view ensures the form that initiated the **POST** was valid and saves the comment to the **Comment** model and then finally redirecting the user to the same view to update the clients page with the new comment being displayed on the Article view.

2.1.8 Graphical User Interface (GUI)

2.1.8.1. Landing Page:



Welcome to Pressfeed!

Looking for a platform to keep you up to date on authentic news sources?

Pressfeed is for you!

News discourse!

Stay informed and express your views on the latest news stories!

All the latest headlines, now on Pressfeed.

Figure 16 – Screenshot of the landing page

This is the landing page of the final implementation of the system. Every page of the application features navigation / nav bar at the top of the page. If the user is not logged in it will provide links to the login and logout pages unless the user is logged in which it will display a dropdown box with the users username when click shows options for the Feed, Subscriptions, and Account pages and also the logout button is located in the dropdown menu.

The Landing page features 3 sections:

1. Top-Card:

   The top-card features an icon of a news paper on a mobile phone and explanation of what the web application's purpose is, news aggregation.

2. Middle-Card:

   The middle-card displays an icon of a messaging system to indicate social functionality. An explanation that social discourse features on news is capable within the application. The middle card also displays a button linking to the Register page after the end of middle-cards description. Alternatively the application displays a ‘Go to your feed’ button when a user is logged in and links to their feed page.

3. Bottom-card:
   The last card queries the latest 4 article within the database and displays a sample of the news available to the user through the use of the application. This also demonstrates what the user can expect from the application

2.1.8.2. Register & Login Pages

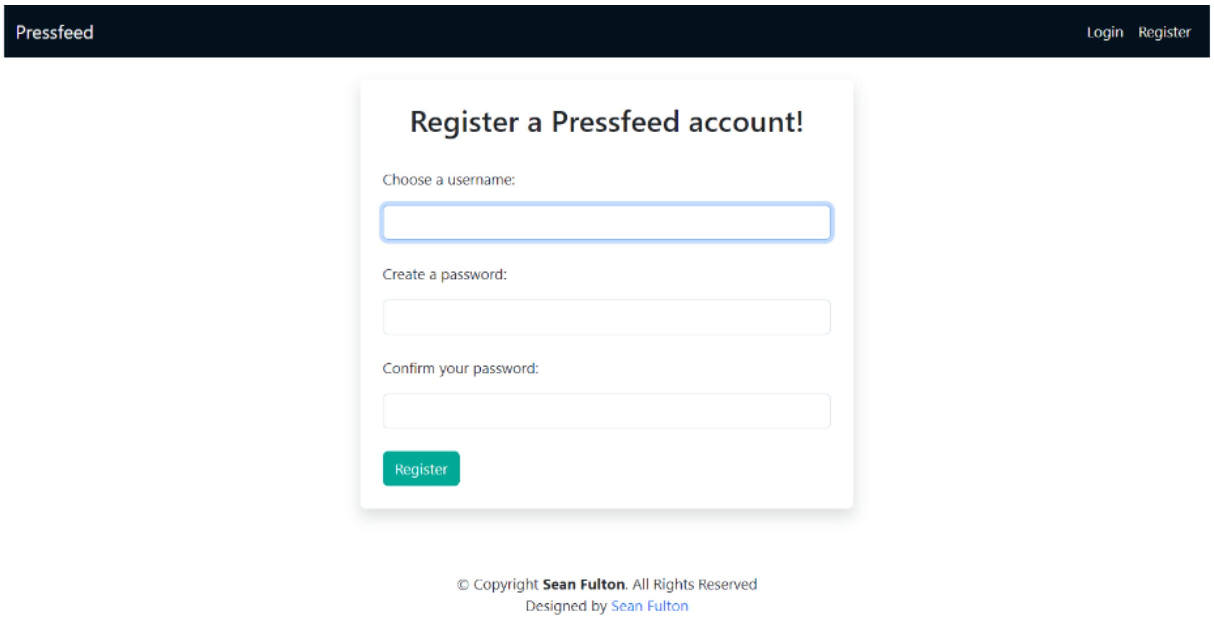
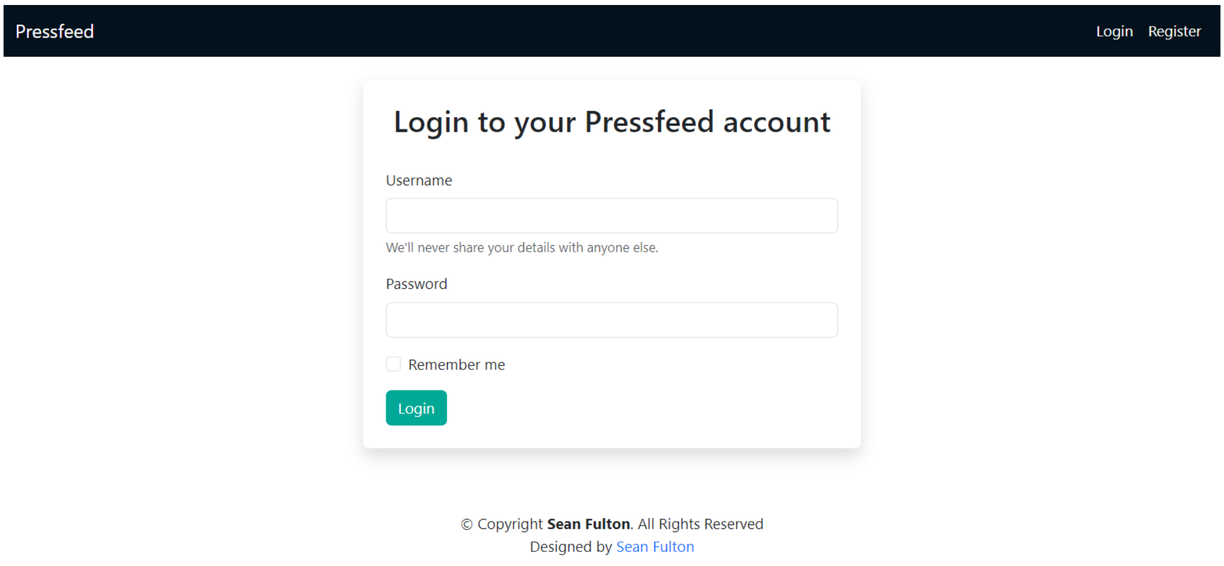


Figure 17 - Screenshot of the register page


Pressfeed

Login Register

Login to your Pressfeed account

Username

We'll never share your details with anyone else.

Password

Remember me

Login

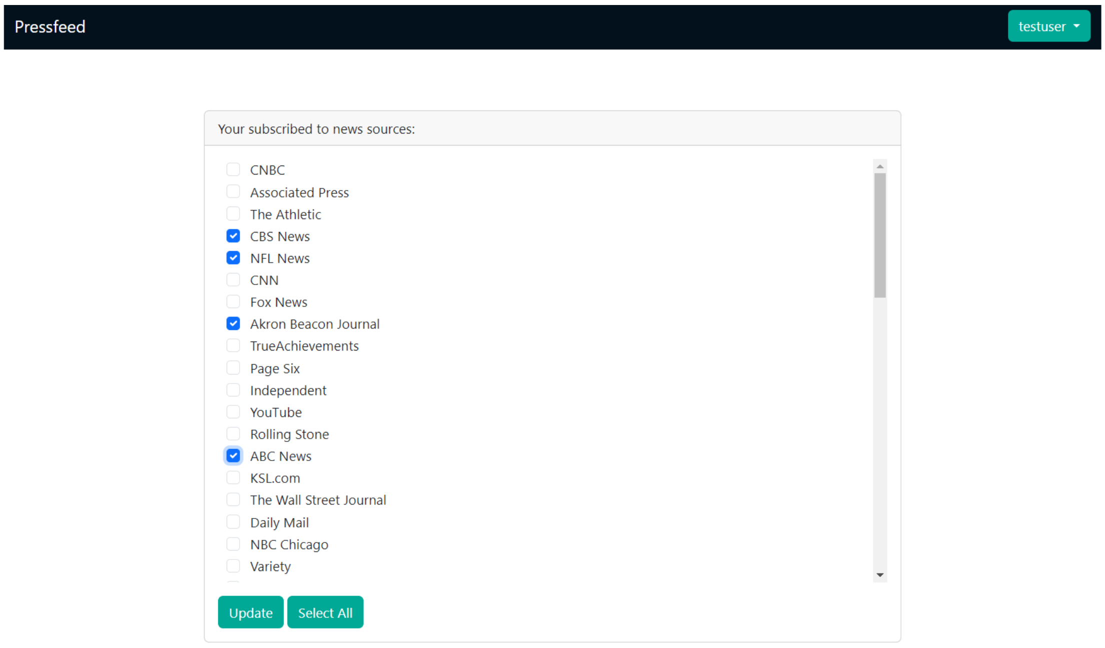
© Copyright **Sean Fulton**. All Rights Reserved
Designed by Sean Fulton

Figure 18 - Screenshot of the Login page

Both figures above show the Register and Login pages of the system. The register page provides fields for users to enter their username and password and field to confirm their password before they can click the 'Register' button below to submit a form to register a new account on the system

The login page features a field for the users username and password which are required to be filled out with the correct credentials of user before they can click the 'Login' button.

2.1.8.3. Subscriptions Page


Pressfeed

testuser

Your subscribed to news sources:

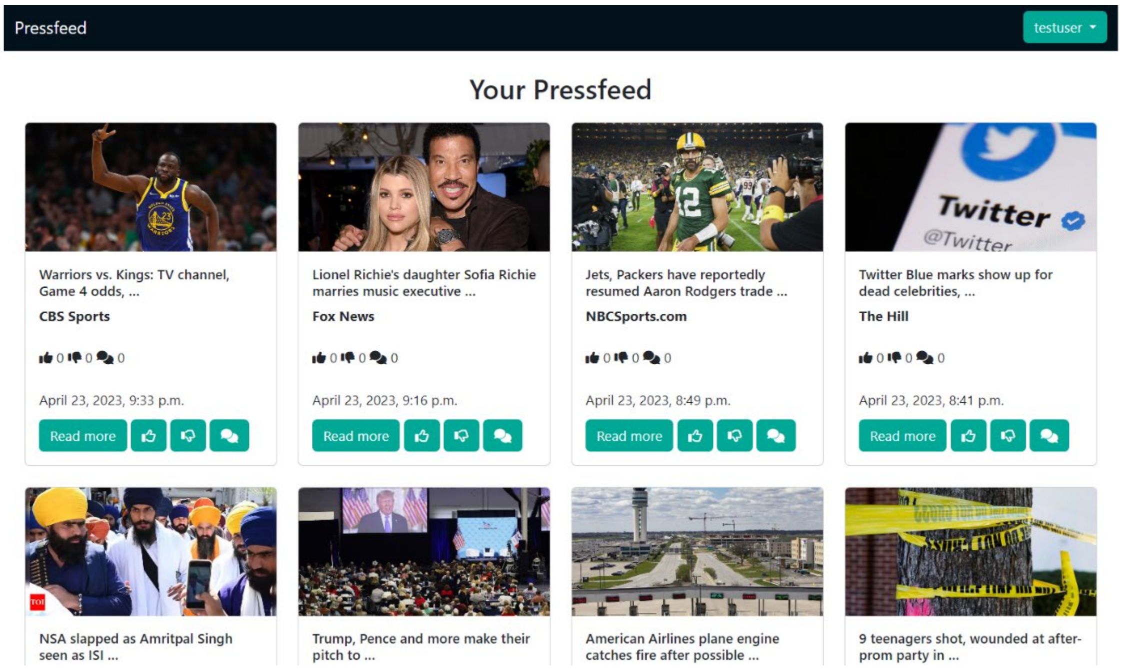
- [ ] CNBC
- [ ] Associated Press
- [ ] The Athletic
- [x] CBS News
- [x] NFL News
- [ ] CNN
- [ ] Fox News
- [x] Akron Beacon Journal
- [ ] TrueAchievements
- [ ] Page Six
- [ ] Independent
- [ ] YouTube
- [ ] Rolling Stone
- [x] ABC News
- [ ] KSL.com
- [ ] The Wall Street Journal
- [ ] Daily Mail
- [ ] NBC Chicago
- [ ] Variety

Update Select All

Figure 19 - Screenshot of the Subscriptions page

This shows the subscriptions page of the system, this page is only accessible if the user is authenticated and logged in to a user account. This page displays all the sources available to the system through the database and allows the user to subscribe/unsubscribe from sources.

2.1.8.4. Feed Page


Pressfeed testuser

Your Pressfeed

Warriors vs. Kings: TV channel, Game 4 odds, ...
CBS Sports
0 0 0
April 23, 2023, 9:33 p.m.

Lionel Richie's daughter Sofia Richie marries music executive ...
Fox News
0 0 0
April 23, 2023, 9:16 p.m.

Jets, Packers have reportedly resumed Aaron Rodgers trade ...
NBCSports.com
0 0 0
April 23, 2023, 8:49 p.m.

Twitter Blue marks show up for dead celebrities, ...
The Hill
0 0 0
April 23, 2023, 8:41 p.m.

NSA slapped as Amritpal Singh seen as ISI ...

Trump, Pence and more make their pitch to ...

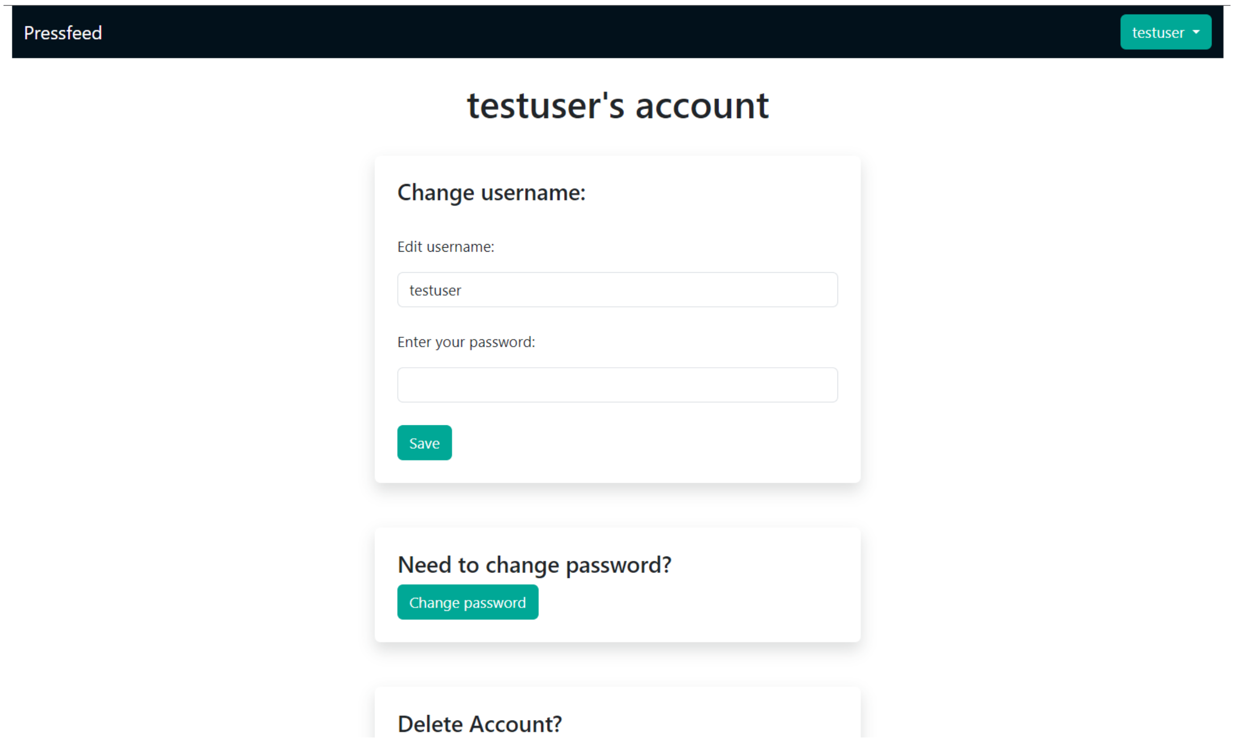
American Airlines plane engine catches fire after possible ...

9 teenagers shot, wounded at after-prom party in ...

Figure 20 - Screenshot of the Feed page

This is the users feed page which deliver Article objects from Sources in which the user is subscribed to. From this page the user can like/dislike articles, cycle through pages at the bottom of the page. And also view the comments of each individual article.

2.1.8.5. Account Page


Pressfeed testuser

testuser's account

Change username:

Edit username:

testuser

Enter your password:

Save

Need to change password?

Change password

Delete Account?

Figure 21 - Screenshot of the Account page

This figure displays the account options functionality which enables the user to change their username and password and also delete their account if they choose to do so.

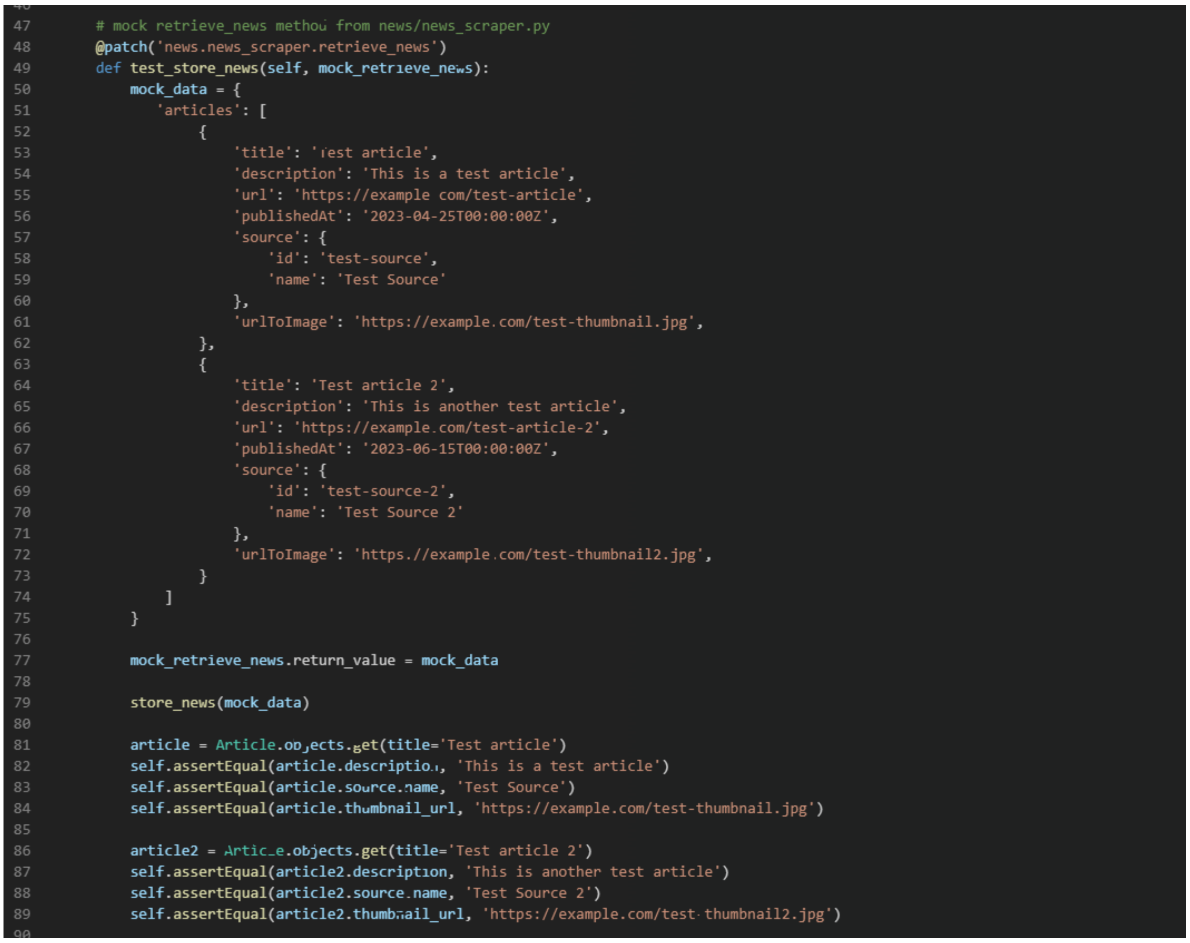
2.1.9 Testing

The applications software development lifecycle incorporated the uses of unit testing and build checks to ensure each release of the application was stable.

2.1.9.1. Unit testing

Unit tests were utilised to test individual blocks of code when implementing new features. I used Pythons built in testing library *‘unittest’.*

A testing environment was created to execute a test suite which uses an in memory database so it does not effect the production databases for the application. This is due to tests using mock and test user data which perform database operations and is flushed (erased) upon the completion of the unit test suite.



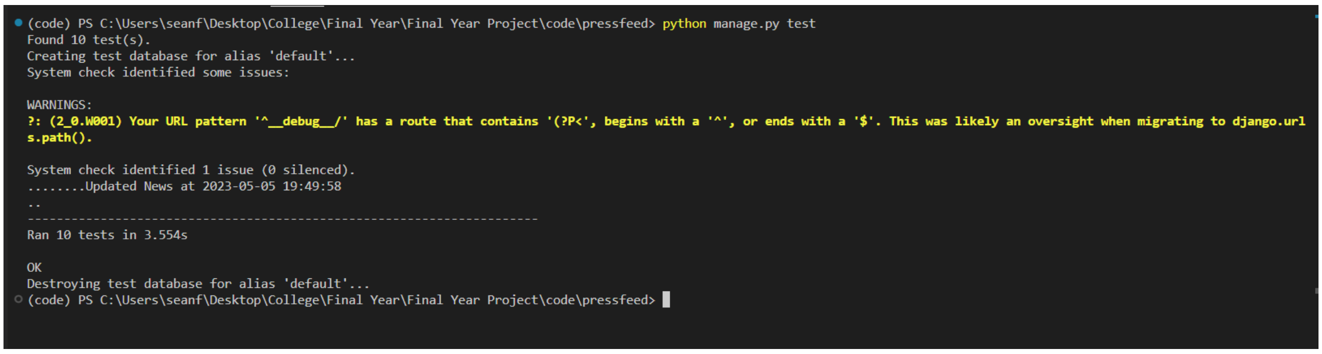
```
    # mock retrieve_news method from news/news_scraper.py
    @patch('news.news_scraper.retrieve_news')
    def test_store_news(self, mock_retrieve_news):
        mock_data = {
            'articles': [
                {
                    'title': 'Test article',
                    'description': 'This is a test article',
                    'url': 'https://example.com/test-article',
                    'publishedAt': '2023-04-25T00:00:00Z',
                    'source': {
                        'id': 'test-source',
                        'name': 'Test Source'
                    },
                    'urlToImage': 'https://example.com/test-thumbnail.jpg',
                },
                {
                    'title': 'Test article 2',
                    'description': 'This is another test article',
                    'url': 'https://example.com/test-article-2',
                    'publishedAt': '2023-06-15T00:00:00Z',
                    'source': {
                        'id': 'test-source-2',
                        'name': 'Test Source 2'
                    },
                    'urlToImage': 'https://example.com/test-thumbnail2.jpg',
                }
            ]
        }

        mock_retrieve_news.return_value = mock_data

        store_news(mock_data)

        article = Article.objects.get(title='Test article')
        self.assertEqual(article.description, 'This is a test article')
        self.assertEqual(article.source.name, 'Test Source')
        self.assertEqual(article.thumbnail_url, 'https://example.com/test-thumbnail.jpg')

        article2 = Article.objects.get(title='Test article 2')
        self.assertEqual(article2.description, 'This is another test article')
        self.assertEqual(article2.source.name, 'Test Source 2')
        self.assertEqual(article2.thumbnail_url, 'https://example.com/test-thumbnail2.jpg')
```

Figure 22 - Unit test of store_news method

Figure 22 is the unit test one of core methods which handle news aggregation of the system. Before defining the test method we provide a *‘@patch’* annotation to tell the system that we are mocking the retrieve_news method in this application.

The method defines a variable *‘mock_data’* to mock 2 article objects that would resemble a article keys from a response from NewsAPI. The application then passes this data to the **store_news** method and calls it.

The application performs a series of assertions on the database to ensure that the **store_news** method saves the objects correctly according to the specified format in the assertions.



```
(code) PS C:\Users\seanf\Desktop\College\Final Year\Final Year Project\code\pressfeed> python manage.py test
Found 10 test(s).
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
?: (2_0.W001) Your URL pattern '^__debug__/' has a route that contains '(?P<', begins with a '^', or ends with a '$'. This was likely an oversight when migrating to django.url
s.path().

System check identified 1 issue (0 silenced).
........Updated News at 2023-05-05 19:49:58
..
----------------------------------------------------------------------
Ran 10 tests in 3.554s

OK
Destroying test database for alias 'default'...
(code) PS C:\Users\seanf\Desktop\College\Final Year\Final Year Project\code\pressfeed>
```

Figure 23 – Test suite execution

Figure 23 shows the output log when executing the unit tests. Each period (.) represents a passed assertion and the 'OK' message indicates all tests were successful.

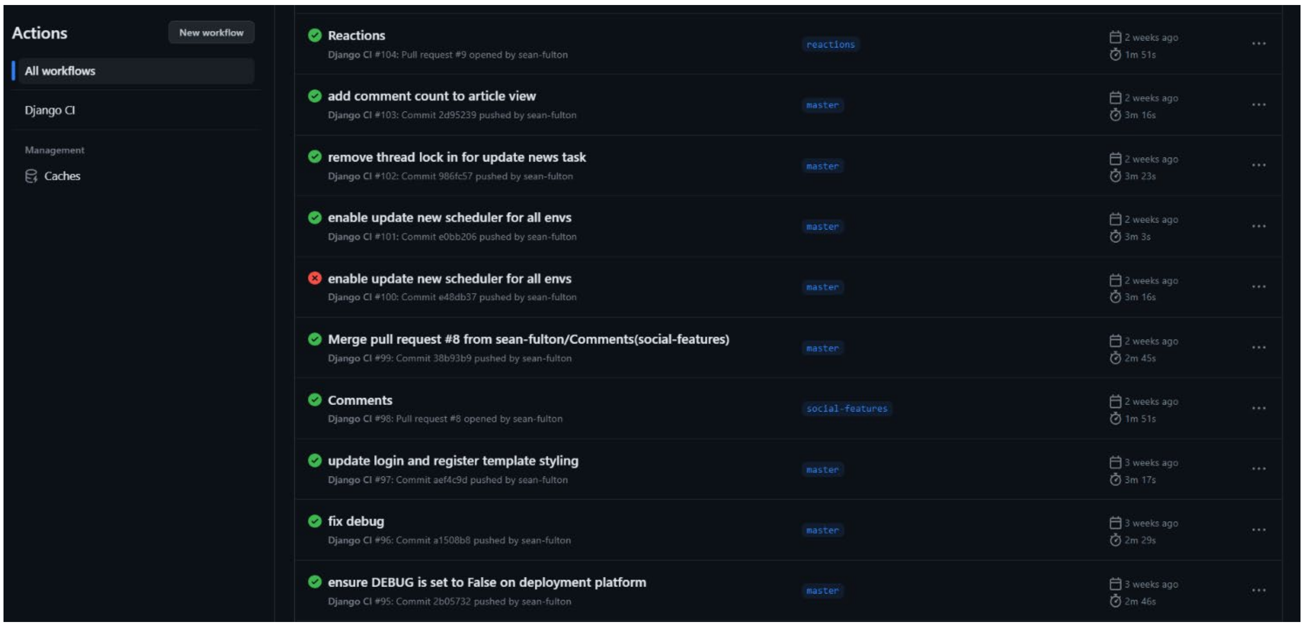
2.1.9.2. System & Acceptance Tests

These tests were executed by utilising the systems functionality through its frontend user interface and verifying the functionality is working as intended after each change

Test	Description	Outcome
User Registration	User can register an account with the system	[PASSED]
User Login	Users can login with a registered account	[PASSED]
Feed Page Display	The feed page should return all articles available to the databases from the users subscribed to sources	[PASSED]
Feed Page Pages	The feed page should only query 8 article per pages. If the articles exceed more than 8 the paginator should section out articles into pages of 8 articles per page.	[PASSED]
News Scraper (System)	News Scraper methods retrieve_news, store_news, update_news function correctly and persist news data as intended	[PASSED]
Update News Scheduler (System)	The python scheduler should start a separate threat at the upon application runtime and schedule a the update_news method to be executed every 4 hours.	[FAILED] Application unsuccessfully executes news aggregation periodically on deployment platform, however does on local development.
Change password	Can the user access their account page and change their password successfully?	[PASSED]
Change username	The user can access the account page and change their account username	[PASSED]

Logout	The user should be able to logout of their account	[PASSED]
Delete account (confirm)	The user should be able to delete their account	[PASSED]
Delete account (cancel)	The user should be able to cancel the deletion process of the account	[PASSED]
Subscriptions	The user should be able to access their subscriptions and manage them, by subscribing or unsubscribing from news sources	[PASSED]

2.1.9.3. CI/CD / Integration Tests



✓	**Reactions** Django CI #104: Pull request #9 opened by sean-fulton	`reactions`	2 weeks ago	1m 51s
✓	**add comment count to article view** Django CI #103: Commit 2d95239 pushed by sean-fulton	`master`	2 weeks ago	3m 16s
✓	**remove thread lock in for update news task** Django CI #102: Commit 986fc57 pushed by sean-fulton	`master`	2 weeks ago	3m 23s
✓	**enable update new scheduler for all envs** Django CI #101: Commit e0bb206 pushed by sean-fulton	`master`	2 weeks ago	3m 3s
✗	**enable update new scheduler for all envs** Django CI #100: Commit e48db37 pushed by sean-fulton	`master`	2 weeks ago	3m 16s
✓	**Merge pull request #8 from sean-fulton/Comments(social-features)** Django CI #99: Commit 38b93b9 pushed by sean-fulton	`master`	2 weeks ago	2m 45s
✓	**Comments** Django CI #98: Pull request #8 opened by sean-fulton	`social-features`	2 weeks ago	1m 51s
✓	**update login and register template styling** Django CI #97: Commit aef4c9d pushed by sean-fulton	`master`	3 weeks ago	3m 17s
✓	**fix debug** Django CI #96: Commit a1508b8 pushed by sean-fulton	`master`	3 weeks ago	2m 29s
✓	**ensure DEBUG is set to False on deployment platform** Django CI #95: Commit 2b05732 pushed by sean-fulton	`master`	3 weeks ago	2m 46s

Figure 24 - builds of the application on GitHub Actions

Figure 24 shows the builds ran by GitHub Actions which were defined by myself with Django.yaml workflow. These workflows run the test suite on versions of the codebase to ensure stable changes are being committed to the remote repository. If a *build* passes on the master branch the *Deploy* workflow executes the deployment steps.

Small changes would be checked locally by running the test suite before pushing to master. However bigger implementations of features I would run checks through a branch on my local dev and GitHub Actions first before merging to master, for example the '*reactions*' and '*social features'* branches as show in Figure 24.

2.1.10 Evaluation

2.1.10. Performance

Conducting tests of the systems performance, evaluating this through SQL query times and http request response times. All tests will be conducted on the local instance of the application to ensure the exact processing time it takes the system complete each request.

HTTP Request times

Test criteria - Request every page 3x and get average request time:

View / Page	Request Time 1	Request Time 2	Request Time 3	Average RT
Home (GET)	84.32ms	80.70ms	88.50ms	84.51ms
Login (GET)	5.56ms	7.08ms	5.52ms	6.05ms
Register (GET)	17.68ms	8.09ms	5.35ms	10.37ms
Login (POST)	100.90ms	113.26ms	146.29ms	120.15ms
Register (POST)	139.00ms	149.00ms	118.69ms	135.56ms
Subscribe (GET)	100.87ms	180.72ms	75.69ms	119.09ms
Subscribe (POST)	84.19ms	78.10ms	108.79ms	90.36ms
Newsfeed (GET)	255.00ms	275.68ms	246.17ms	258.95ms
Account (GET)	57.14ms	55.94ms	55.74ms	56.27ms
Account (POST) [Change Username]	56.03ms	57.99ms	159.08ms	91.03ms
Account (POST) [Delete account]	82.80ms	86.13ms	87.81ms	85.58ms
Change_password (GET)	57.28ms	65.08ms	54.13ms	58.83ms
Change_password (POST)	59.01ms	72.92ms	63.60ms	65.18ms

SQL Query times

Test criteria – Execute the queries used by the application 3x and record their execution time

Query	Query time 1	Query time 2	Query time 3	Average QT
Home page query for sample articles	13.31 ms	18.99 ms	10.35 ms	14.22ms
Newsfeed page when querying articles from article model	37.59 ms	41.05 ms	40.55 ms	39.73ms
Subscriptions page when querying sources	7.99 ms	11.01 ms	9.92 ms	9.64ms

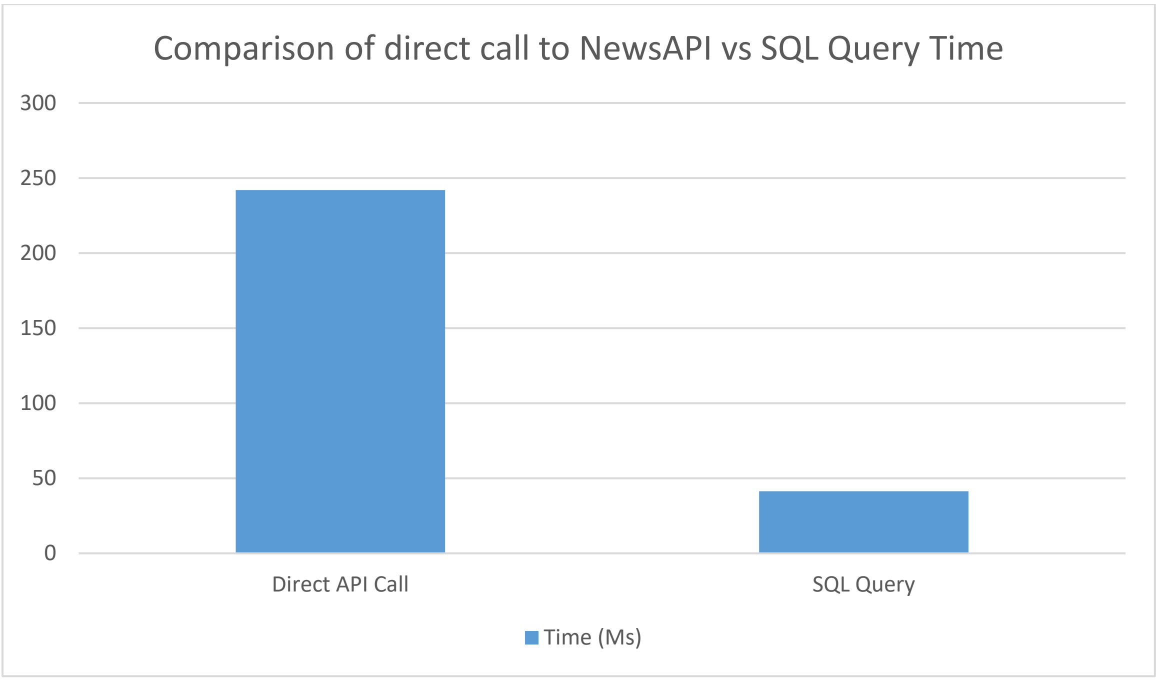


Figure 25 - NewsAPI Call vs SQL Query

Figure 25 compares the performance impact on the request time of the newsfeed page when directly calling NewsAPI each time a user accesses that page vs using a database model to allow the system to query when it needs to display news metadata. a SQL implementation is a much more efficient approach as it does not use NewsAPI requests unnecessarily and is also a much more efficient approach at delivering news metadata to users.

3 Conclusions

The implementation of the application successfully includes all the requirements outlined in the requirements section of this document. The system performs all functionality in an efficient manner as evaluated through the testing documented in the Testing / Evaluation section of this document. Demonstrating use cases requiring computation time to be longer than half a second to deliver their functionality to the user.

The system promotes a scalable approach to news aggregation by modelling the database and news scraper methods in such a way to accept a JSON object from any source to and store the elements of the object in the systems Source and Article tables. Streamlining the ability to implement another scraping library or API which can structured into a JSON and added to the news aggregation system.

The application performs all user functionality using the POST / Redirect / GET (PRG) pattern, this can be an issue for simpler functionality such as liking and disliking articles as the users page is redirected when initiating a like or dislike or even a comment.

Overall the application serves its purpose as intended and opens up many areas to further develop the application in the future.

Users can reach the deployed web application at this url: https://pressfeed.fly.dev/

4 Further Development or Research

Pressfeed has many options for its future in terms of new features and commercialisation.

1. Chat System:

   This would tie into one of the core values of Pressfeed which is its aim to provide social discourse for news, this would allow users to chat privately and directly between 2 users or a group of users. User should be able to share articles in the structure that was presented to them on their curated feed page.

2. Search and Filtering functionality:
   Allowing users to search through articles directly instead of being required to browse through articles of their subscribed sources, would enable users to use Pressfeed as a article lookup system and also even find new news sources they may be interested in.

3. Commercialising the product:

   Pressfeed could look into displaying ads or accepting a premium membership to elevate users functionality through the application. This could provide a source of income for the development to team to invest in greater sources of news metadata options such as premium API's which serve metadata

Issue with scheduled update news task on deployment platform:

The current deployment platform of Pressfeed is Fly.io when scheduling task of updating news on the deployment platform the application cannot seem to initiate a successful call ot the database when carrying out the operations of the update_news method when executed by the scheduler. This will need to be looked into in the future and could be related to the scheduler utilising multithreading which executes the scheduler in a separate thread to the core system.

5.0 References

Reuters Institute, 2020. *Reuters Institute Digital News Report,* Oxford: University Of Oxford.

Trusted Web Foundation, 2021. *State of Misinformation 2021.* [Online]
Available at: https://thetrustedweb.org/state-of-misinformation-2021-europe/
[Accessed 5 April 2023].

5 Appendices

5.1 Project Proposal

National College of Ireland

Project Proposal

Pressfeed

24/10/2022

BSHCSD4

Software Development

2022/2023

Sean Fulton

x19518013

x19518013@student.ncirl.ie

Contents

Objectives

Pressfeed sets out to bring more of a social aspect to online media & news outlets, I would like to incorporate a user defined newsfeed much like popular social media phone applications such as Instagram, Facebook & Twitter etc.. I will try to transform what these applications achieved in terms of user experience and user interface for Pressfeed.

Pressfeed will be a RESTful web application for users to curate a hub for news outlets and articles they are interested in seeing on a regular basis.

Pressfeed should include these main features by the end of development:

- A user authentication system for users to sign up or log in to Pressfeed.
- A user defined feed on which users can subscribe to news outlets, articles, keyword tracked searches they would like to see on their Pressfeed feed.
- A rating & comment system on each article localised to Pressfeed for discussion & feedback on articles.

During development I plan to expand on the capabilities of Pressfeed and further develop the application's features but for now I feel these features are the main goal of my application.

Background

I chose to create Pressfeed for my final year project as I feel there is a gap in the market for a social media based news compiler platform with features for discourse of media & news all within in the one platform. people with similar interests can discuss the topics delivered to them with the other users of Pressfeed.

How I aim to achieve these requirements is by developing a web application based on the Django web framework, Python programming language, and NewsAPI (an API to locate and deliver news articles). These 3 technologies & resources should be adequate in my goal to develop Pressfeed's main components. I aim to research and develop my ability in these web technologies to implement the required objectives I have outlined for Pressfeed's final release.

State of the Art

Similar web applications to Pressfeed already exist in basic concept. Some of these applications include Flipboard.com & Feedly.com. Pressfeed in contrast to these applications which are within the same field. Pressfeed tries to introduce a connected experience between its users to share, comment, like and dislike news articles they subscribe to. Note Pressfeed will not act as a substitute to going to the original articles being shown on the application.

Pressfeed aims to allow independent discussion of these articles on a platform separate to where the articles were originally published. This gives less of a biased discussion towards certain news publications and stops the filtering of negative comments by the author/publisher of the article.

Technical Approach

I plan to use a Hybrid approach by taking characteristics of two methodologies to carry out planning & development for Pressfeed . I would like to implement features in an Agile framework by following a timeframe of 2 week sprints to deliver and begin new features for the application. I feel a 2 week timeframe given for a new set of features will give ample time to implement a new set of features regularly and give enough development time to each feature to be implemented to my according to the requirement of the feature.

Since I am sole developer on Pressfeed I will be responsible for all tasks related to the project. This gives me the freedom to shift features in / out of sprint windows to give more development time or less for certain features which I feel are adequately implemented or others that need more time to be completed.

I would also like to combine characteristics of the Waterfall release process, since Pressfeed as a pre-determined date for which it will be submitted, I need to take into consideration the final release of Pressfeed deadline will be in May of 2023 for its assessment. This release / version of Pressfeed is the goal at which my development planning must ad-here to. This means that there will be certain fixed timeframes for the current state Pressfeed should be at to stay on track for final release in May 2022

How I plan to identify requirements for Pressfeed is by researching the implementation of a Django based web application. I can then assess how I will approach implementing my own designs for Pressfeed using Python, Django, the API's required for Pressfeed, other technologies I may decide to include in the application.

Once I am more familiar with the technologies use to implement Pressfeed plan to assess these requirements and how to implement them for the application.

- Functional Requirements
    1. Provide web service to curate articles published from news publications.
    2. Allow the discussion of posted articles.
    3. Authentication system for user logins.
    4. Provide a discover page for recommended articles based on user interests.
- User Requirements
    1. The user should be able to sign up to Pressfeed and Login.
    2. The user can search for and subscribe to articles and news publications websites.
    3. User should be able to comment, rate and share articles they see on Pressfeed.
    4. Users should to see a snippet of articles.
    5. User can add friends on Pressfeed.
- Quality of Service
    1. Users experience of Pressfeed should be smooth and have no UI issues when browsing the application.
    2. New users should be able to easily identify the full functionality of Pressfeed.
    3. Ensure profanity is filtered from discussions of articles on public posts.

- Ethics Requirements
    1. Ensure Pressfeed is not a substitute for articles from any news publications. This will avoid copyright implications.
    2. Ensure user data is stored securely on Pressfeed.
    3. Private user data should only be accessible by the user.
    4. Allow users to delete their profile on Pressfeed and remove any data relating to the user.

Technical Details

Python 3 - https://www.python.org/ :

Python will be the programming language that powers Pressfeed alongside the Django Web Framework. I choose Python to handle Pressfeed's logic as I don't have a lot of experience with Python, Pressfeed gives me the opportunity to create a new idea while improving my knowledge of new technologies and Python is a great choice for how popular the language is web application development.

Python has many libraries which make the use of API's very accessible which Pressfeed will utilize at least one API, namely NewsAPI. Python is very efficient at handling large amounts of data, which is another consideration for Pressfeed since there will need to be have technologies with scalable infrastructure to handle the amount of data delivered from API's

Other considerations: Ruby On Rails, React Native.

Django Web Framework - https://www.djangoproject.com/ :

Django is a web framework for the Python Programming language, choosing Django as framework for Pressfeed, stemmed from its popularity and its scalability, Django is an obvious choice for a web application developed with Python as it is one of the two most popular web frameworks for Python, the other being Flask. I chose Django for its exceptional performance, flexibility and security.

When I use an application the first thing I will notice is ease of use, how smooth and responsive the application feels when I use it. This is why I aim to have a fast and reliable UX for Pressfeed and I am confident Django can achieve that goal.

NewsAPI - https://newsapi.org/ :

Pressfeed backend will utilize the API "NewsAPI" to deliver articles from news publications websites to the invite Pressfeed users to view the article externally on the original website the article is published and then discuss to rate, comment and/or share on Pressfeed.

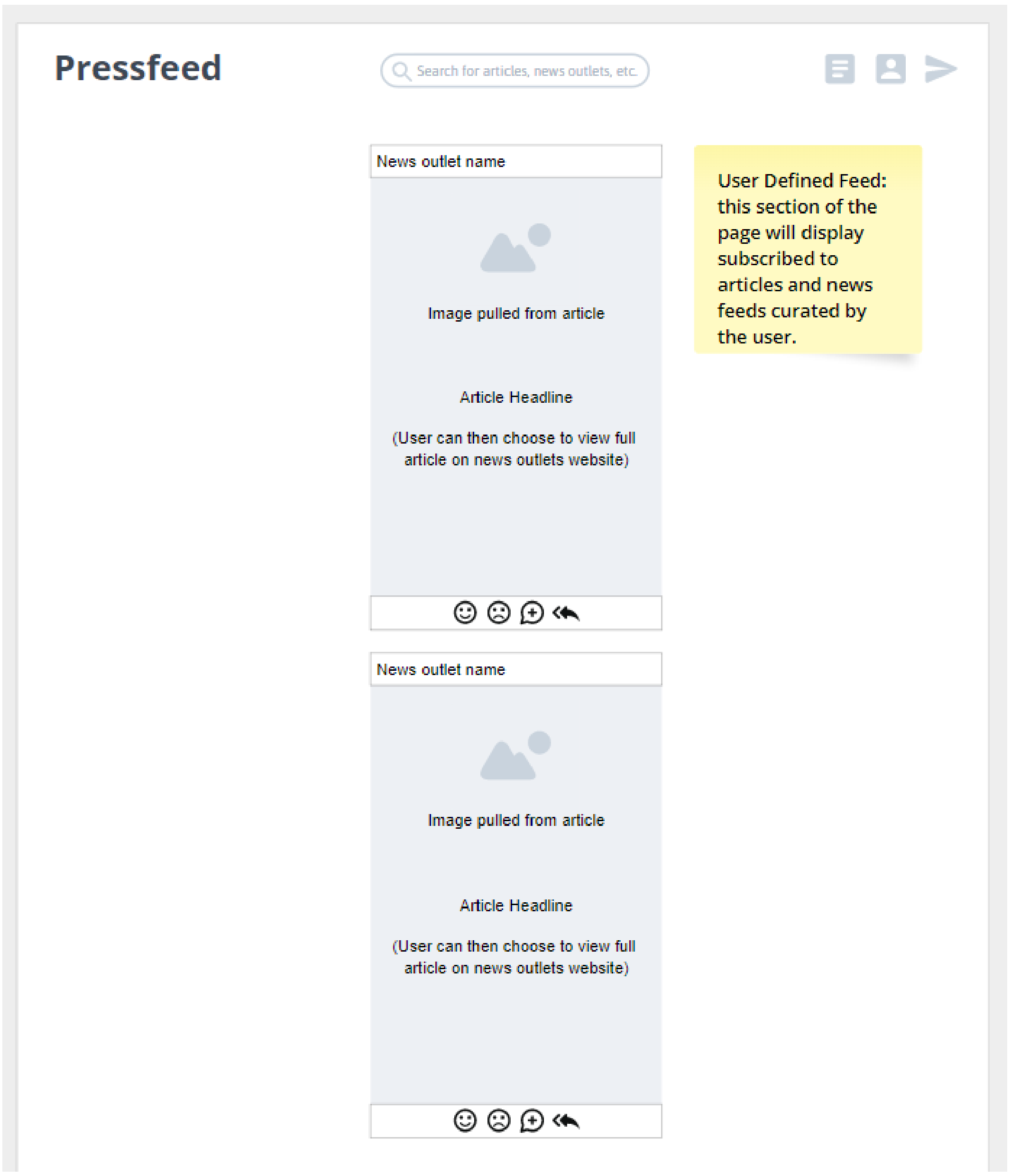
This API integrates seamlessly as it has its own library developed for Python, this should allow me to easily request data from the API to deliver to Pressfeed.

Special Resources Required

I would consider NewsAPI as a special resource for Pressfeed although it being the technology that delivers the center piece feature of the application, this API serves one function, which is to accept queries on news articles and deliver them back to the application. The main technology that will handle the rest of Pressfeed's features will be Python & the Django framework.

Project Plan

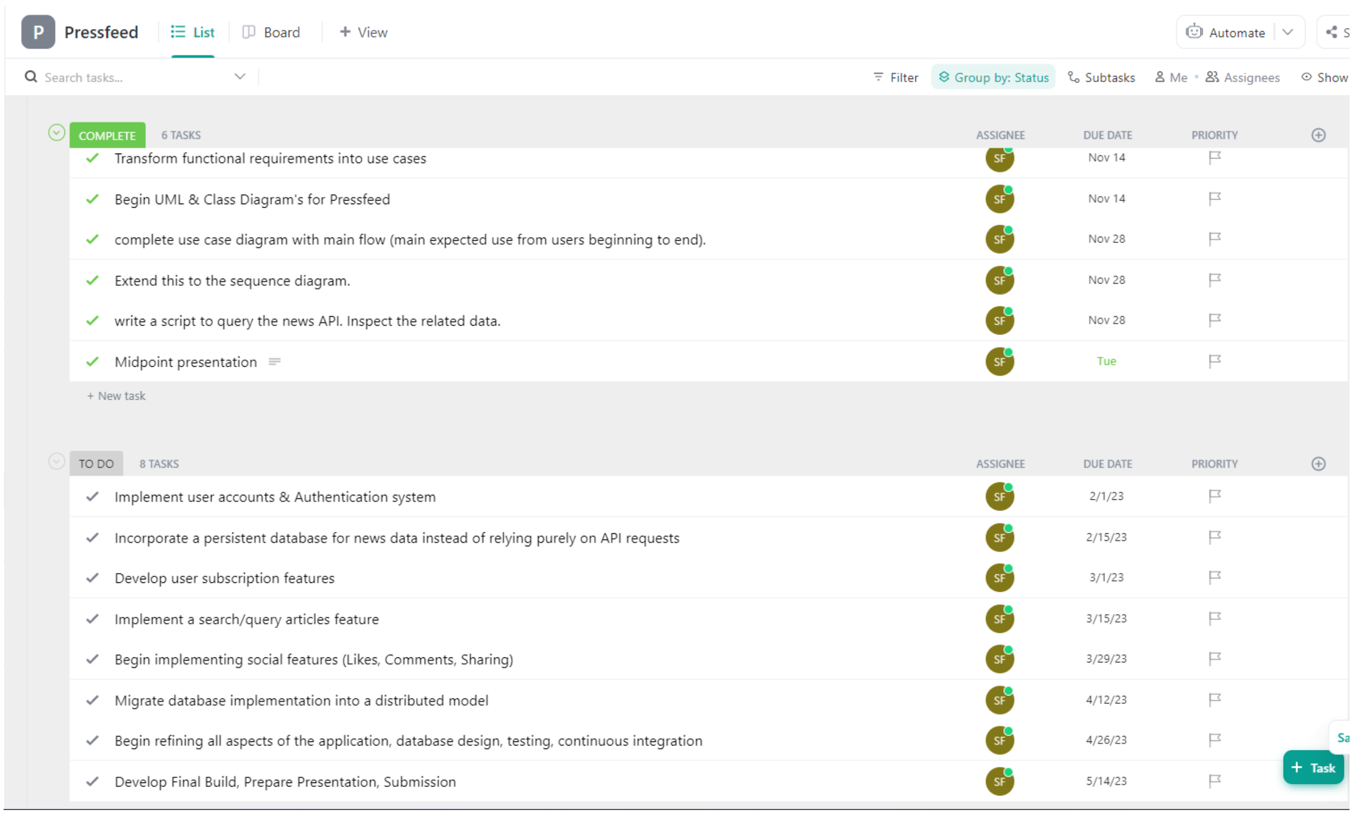
During development of Pressfeed I plan to implement all functionality of the application with the help of both UML & mock diagrams. for the moment I have created a mock diagram to represent what I plan Pressfeed's final build to look like:



Pressfeed
Search for articles, news outlets, etc.
News outlet name
Image pulled from article
Article Headline
(User can then choose to view full article on news outlets website)
User Defined Feed: this section of the page will display subscribed to articles and news feeds curated by the user.
News outlet name
Image pulled from article
Article Headline
(User can then choose to view full article on news outlets website)

Keeping track of deliverables in on a per sprint basis of 2 weeks will allow me to stay on top of development deadlines

I have separated the implementation of features into two week sprints as a general timeline to have features completed by. Although I won't be releasing features for general use nor releasing Pressfeed until May, I feel following an Agile approach for feature implementation will keep me on track to complete Pressfeed for May 2023.



COMPLETE 6 TASKS	ASSIGNEE	DUE DATE	PRIORITY
✓ Transform functional requirements into use cases	SF	Nov 14	
✓ Begin UML & Class Diagram's for Pressfeed	SF	Nov 14	
✓ complete use case diagram with main flow (main expected use from users beginning to end).	SF	Nov 28	
✓ Extend this to the sequence diagram.	SF	Nov 28	
✓ write a script to query the news API. Inspect the related data.	SF	Nov 28	
✓ Midpoint presentation	SF	Tue	

+ New task

TO DO 8 TASKS	ASSIGNEE	DUE DATE	PRIORITY
✓ Implement user accounts & Authentication system	SF	2/1/23	
✓ Incorporate a persistent database for news data instead of relying purely on API requests	SF	2/15/23	
✓ Develop user subscription features	SF	3/1/23	
✓ Implement a search/query articles feature	SF	3/15/23	
✓ Begin implementing social features (Likes, Comments, Sharing)	SF	3/29/23	
✓ Migrate database implementation into a distributed model	SF	4/12/23	
✓ Begin refining all aspects of the application, database design, testing, continuous integration	SF	4/26/23	
✓ Develop Final Build, Prepare Presentation, Submission	SF	5/14/23	

Testing

I plan to incorporate test driven development for Pressfeed by unit testing all instances of code within its implementation, this includes methods and other functionality of the codebase's implementation. Incorporating a continuous integration / continuous delivery workflow within Pressfeed's GitHub repository I can ensure my modifications keep a standard of stability which will prevent introducing bugs throughout development.

I will incorporate the Python unit-test library during development for unit testing functionality to ensure my implementation of new functionality is stable and does not affect other aspects of the codebase.

5.2 Reflective Journals

5.2.1. October:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4
Supervisor	William Clifford

Month: October 2022

What? Reflect on what has happened in your project this month? This month I conceptualised Pressfeed as an idea. I prepared and submitted the project pitch video for Pressfeed for review. I also researched how I will combine the technologies. I have chosen to develop Pressfeed with. I prepared the first draft for my project proposal and completed the ethics forms for Pressfeed.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? Pressfeed still needs to begin and complete its design phase. This will be my first task is to begin forward design of Pressfeed by creating UML diagrams of Pressfeed to give a more in depth view into the process of creating the application. My goal is to expose any design challenges I may encounter during development.
Now What? What can you do to address outstanding challenges? Continue with planning and begin the design phase for the application.

Student Signature	Sean Fulton

5.2.2. November:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4
Supervisor	William Clifford

Month: November 2022

What? Reflect on what has happened in your project this month? • I began outlining my functional requirements by creating a use case diagram for the main user flow of Pressfeed's implementation. • Started testing NewsAPI by calling GET requests to various endpoints on the API. • Created a virtual environment for the development of my Django app with the python virtual environment package 'pipenv' • Generated a new Django-project to begin implementing a prototype version of the application.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? Still early days for Pressfeed but I am happy with the progress so far biggest challenges is looking into permission to use news publication websites article metadata for aggregation to be used in my web application, also researching the Django framework is something I will need to learn as I develop the project.
Now What? What can you do to address outstanding challenges? My goal for the month of December is to complete a prototype for the midpoint presentation and complete my UML diagrams for the technical report.

Student Signature	Sean Fulton

5.2.3. December:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4

Supervisor	William Clifford

Month: December 2022

What? Reflect on what has happened in your project this month? For December I configured the technologies for Pressfeed to begin development. I began implementing a 0.1.0 version of Pressfeed for my Midpoint prototype. This included a basic aggregation of news article metadata scraped using NewsAPI and displayed on a Django web application.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? This is great alpha version of my application which will allow me to begin the implementation of many features requirements. i.e user accounts, subscriptions, liking, sharing, direct message system, distributed models. These features will require a lot of research into the various technologies I have chosen for each.
Now What? What can you do to address outstanding challenges? My plan for the month of January is to further research technologies such as RabbitMQ, Django, and Google Firebase to implement more of the backend features I have previously mentioned including subscriptions and a distributed database model for persistent data storage to alleviate over use of a API calls.

Student Signature	Sean Fulton

5.2.4. January:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4

Supervisor	William Clifford

Month: January 2023

What? Reflect on what has happened in your project this month? In January I spent a considerable amount of time researching RabbitMQ, Google Firebase, and Django to improve my understanding of how these technologies can interact with one another before I continue developing the more crucial features of Pressfeed
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? The outlying features from December are still to be implemented however I feel well equipped to implement these now with knowledge I have gained from researching these technologies.
Now What? What can you do to address outstanding challenges? For February I plan to get back on track with the development process and aim for milestones I have outlined for myself during the planning stage of Pressfeed

Student Signature	Sean Fulton

5.2.5. February:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4
Supervisor	William Clifford

Month: February 2023

What? Reflect on what has happened in your project this month? This month I implemented core functionality to Pressfeed which included: • Scheduled task to retrieve news from NewsAPI and store it within a PostgreSQL database • User Authentication (Users can create an account, login, view their feed, and manage subscriptions) • Feed view now delivers articles from the PostgreSQL implementation instead of a call to NewsAPI
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? This month has been my most productive month so far as I have implemented core functionality (user subscriptions) • User account detail editing by users is still to be implemented. • Styling of the applications views (web pages) • Messaging system to allow users to share article cards to one another
Now What? What can you do to address outstanding challenges? Documentation needs revision. I will start filling in implementation and architecture

Student Signature	Sean Fulton

5.2.6. March:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4
Supervisor	William Clifford

Month: March 2023

What? Reflect on what has happened in your project this month? For March I revised the implementation of the frontend of Pressfeed. Styling of the application required a more streamlined UI for new users to easily create an account and choose their desired news sources before showing the 'Feed' page.
So What? Consider what that meant for your project progress. What were your successes? What challenges still remain? A few more frontend features need to be implemented during April such as a more user friendly register and login views, before I begin implementing the messaging system. I can then begin testing web application codebase and its security.
Now What? What can you do to address outstanding challenges? Deployment and Documentation are still the final stages of Pressfeeds development I would like to have these completed by the last week of April.

Student Signature	Sean Fulton

5.2.7. April:

Supervision & Reflection Template

Student Name	Sean Fulton
Student Number	x19518013
Course	BSHCSD4
Supervisor	William Clifford

Month: April 2023

What? Reflect on what has happened in your project this month? This month I completed the entire functionality of all aspects of Pressfeed. Reviewing testing suite , unit tests and acceptance tests of the system

The newly features implemented in April was a reaction and comment system. This allowed users to comment on articles and also like or dislike them.

The application is also now deployed at https://pressfeed.fly.dev

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

This month was by far the most difficult for me, At the start of this month the development process of the application was very slow due to me having other modules deadlines and the Social Functionality requirement which accomplishes one of the applications main goals was still to be implemented.

I feel I have gained a considerable amount of knowledge and experience in developing a web application at as I come to the final stages of Pressfeeds development.

Now What?

What can you do to address outstanding challenges?

Since the implementation has been completed I can focus on documenting the application and preparing the video demonstration for submission in two weeks' time.

Student Signature	Sean Fulton