

**FOP 2 Project Report**

**ME 15 Section B**

**Submitted to:**

Sir Saqib Nazir

**Submitted by:**

|  |  |
| --- | --- |
| **NAME** | **CMS** |
| Muhammad Owais | 461359 |
| Muhammad Shahzeb Khan | 479677 |
| Muhammad Bin Akhtar | 467463 |
| Abdul Moiz | 464834 |
| Abdur Rehman Kazmi | 460951 |

RSS Feed Filter Project Report

**Introduction to the Project**

In today's fast-paced world, staying updated with the latest news is crucial. However, the sheer volume of news articles can be overwhelming, making it difficult to filter out the noise and focus on stories that matter most to an individual. This project addresses this challenge by creating a program that automatically retrieves and processes news articles from multiple RSS feeds, filters them based on user-defined criteria, and displays the relevant articles in a user-friendly graphical user interface (GUI). By leveraging customizable triggers, users can specify exactly what types of news stories they are interested in, ensuring they receive timely updates on topics of importance without sifting through irrelevant content. This application not only enhances productivity by saving time but also ensures that users stay informed about the news that truly impacts them. The project showcases the power of combining web scraping, string manipulation, and GUI programming to create a practical and efficient news filtering system.

**Our Approach to the Code**

Our approach focuses on modular design and extensibility. We implemented various classes to represent news stories and triggers, allowing the filtering logic to be easily extended with new types of triggers. The program fetches news feeds periodically, processes them into a structured format, and applies the triggers to filter relevant stories. The filtered stories are then displayed in a GUI. We ensured that each component of the system, from data fetching to display, could be easily maintained and updated, making the application robust and scalable. This approach allows for future enhancements, such as adding more sophisticated triggers or integrating additional news sources, without requiring significant changes to the existing codebase.

**Our Libraries**

* **feedparser`:** Parses RSS feeds and extracts news stories.
* **string`:** Provides string manipulation utilities.
* **time`:** Used for implementing delays in the polling mechanism.
* **threading`:** Facilitates running the feed polling in a separate thread.
* **project\_util`:** Contains utilities for translating HTML entities.
* **mtTkinter`:** A thread-safe version of Tkinter for creating the GUI.
* **datetime` and `pytz`:** Handle date and time parsing, including timezone management.

**Techniques We Used**

* Object-Oriented Programming (OOP): We created various classes (`NewsStory`, Trigger`, `PhraseTrigger`, etc.) to encapsulate related data and functionality.
* Regular Expression and String Manipulation: Used in `PhraseTrigger` to detect phrases in text.
* Date and Time Handling: Managed with `datetime` and `pytz` libraries to handle publication dates and time-based triggers.
* GUI Development: Implemented using `mtTkinter` to create an interactive interface.
* Multithreading: Ensured the GUI remains responsive while fetching news feeds in the background.

**Explanation of the Code**

* **Muhammad Shahzeb Khan**

**Class Definitions**

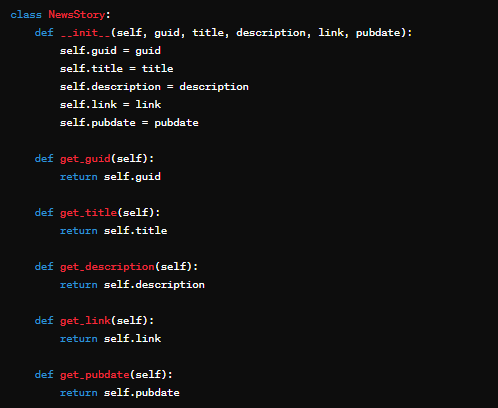
* **News Story Class**

**Purpose:** Represents a news story with attributes like GUID, title, description, link, and publication date.

**Methods:**

* + `\_\_init\_\_(self, guid, title, description, link, pubdate)`: Initializes a NewsStory object with given attributes.
  + `get\_guid(self)`: Returns the GUID of the story.
  + `get\_title(self)`: Returns the title of the story.
  + `get\_description(self)`: Returns the description of the story.
  + `get\_link(self)`: Returns the link to the full story.
  + `get\_pubdate(self)`: Returns the publication date of the story.

**Explanation:** This class encapsulates the details of a news story, allowing easy access to its components through getter methods. It serves as the primary data structure for storing news items.



* **Trigger Class**

**Purpose:** Abstract base class for all triggers.

**Methods:**

* evaluate(self, story): Abstract method to evaluate if a trigger condition is met for a given story.

**Explanation**: As an abstract base class, Trigger defines the interface for all types of triggers. Subclasses must implement the `evaluate` method to specify their conditions.

A screen shot of a computer error

Description automatically generated

* **Abdul Moiz**

**Phrase Triggers**

* **PhraseTrigger Class (subclass of Trigger)**

**Purpose:** Checks if a specific phrase is present in a given text.

**Methods:**

- `\_\_init\_\_(self, phrase)`: Initializes with a phrase.

- `is\_phrase\_in(self, text)`: Returns `True` if the phrase is found in the text.

**Explanation:** This class normalizes text by converting it to lowercase and removing punctuation, then checks for the presence of a specified phrase. It forms the basis for other phrase-based triggers.

A computer screen shot of text

Description automatically generated

* **TitleTrigger and DescriptionTrigger Classes (subclasses of PhraseTrigger)**

**Purpose:** Check if a phrase is present in the title or description of a story.

**Methods:**

- `evaluate(self, story)`: Returns `True` if the phrase is found in the title (TitleTrigger) or description (DescriptionTrigger).

**Explanation:** These subclasses leverage `PhraseTrigger` to focus on specific parts of a news story. They override the `evaluate` method to apply the phrase check to the title or description, making them specialized triggers.

A screen shot of a computer program

Description automatically generated

* **Muhammad Owais**

**Time Triggers**

* **TimeTrigger Class (subclass of Trigger)**

**Purpose:** Abstract base class for time-based triggers.

**Methods:**

- `\_\_init\_\_(self, time\_str)`: Initializes with a specific time.

**Explanation:** This class converts a time string into a `datetime` object. It provides a foundation for triggers that need to compare story publication times to a specific moment.

A black screen with white text

Description automatically generated

* + **BeforeTrigger and AfterTrigger Classes (subclasses of TimeTrigger)**

**Purpose:** Check if a story's publication date is before or after a specified time.

**Methods:**

- `evaluate(self, story)`: Returns `True` if the story's publication date is before (BeforeTrigger) or after (AfterTrigger) the specified time.

**Explanation:** These classes compare the publication date of a news story to the specified time. They are useful for filtering stories based on their recency relative to a given date and time.

A screen shot of a computer program

Description automatically generated

* **Abdur Rehman Kazmi**

**Composite Triggers**

* **NotTrigger Class (subclass of Trigger)**

**Purpose**: Inverts the result of another trigger.

**Methods:**

- `\_\_init\_\_(self, trigger)`: Initializes with another trigger.

- `evaluate(self, story)`: Returns the opposite of the wrapped trigger's evaluation.

**Explanation:** This trigger negates the result of another trigger. It's useful for excluding stories that meet certain conditions defined by another trigger.

A screen shot of a computer program

Description automatically generated

* **AndTrigger and OrTrigger Classes (subclasses of Trigger)**

**Purpose:** Combine two triggers using logical AND or OR.

**Methods:**

- `\_\_init\_\_(self, trigger1, trigger2)`: Initialize with two triggers.

- `evaluate(self, story)`: Returns `True` if both triggers evaluate to `True` (AndTrigger) or if either trigger evaluates to `True` (OrTrigger).

**Explanation**: These classes combine two triggers to form more complex conditions. `AndTrigger` requires both conditions to be met, while `OrTrigger` requires at least one to be met. They enable the creation of flexible and powerful trigger combinations.

A screen shot of a computer program

Description automatically generated

* **Muhammad Bin Akhtar**

**Filtering and Configuration**

* **filter\_stories Function.**

**Purpose:** Filters a list of `NewsStory` objects based on a list of triggers.

**Methods:**

- \*\*Parameters\*\*: `stories` (list of `NewsStory`), `triggerlist` (list of `Trigger`).

- \*\*Returns\*\*: A list of `NewsStory` objects that match any of the triggers.

**Explanation:** This function iterates through a list of news stories and applies each trigger from the trigger list. It collects and returns the stories that match any of the triggers, effectively filtering out irrelevant news.



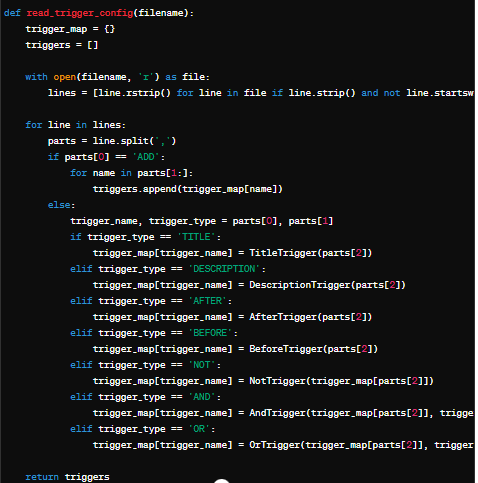
* **read\_trigger\_config Function.**

**Purpose:** Reads a configuration file to create triggers based on user specifications.

**Parameters:** `filename` (name of the configuration file).

- \*\*Returns\*\*: A list of trigger objects specified in the configuration file.

**Explanation**: This function parses a configuration file to set up triggers. It supports various trigger types and their combinations, allowing users to customize the news filtering process through a simple configuration file.



* **process Function**

**Purpose:** Retrieves and parses RSS feeds.

- \*\*Parameters\*\*: `url` (RSS feed URL).

- \*\*Returns\*\*: A list of `NewsStory` objects.

**Explanation**: This function uses `feedparser` to fetch and parse news from an RSS feed. It extracts relevant details and creates `NewsStory` objects for each entry. The function handles different date formats and ensures the publication date is correctly parsed.

A computer screen shot of a program

Description automatically generated

* **Main Thread Function**

**Purpose:** Main loop that periodically fetches news stories, filters them, and updates the GUI.

- \*\*Parameters\*\*: `master` (Tkinter master widget).

- \*\*Functionality\*\*:

- Initializes some sample triggers.

- Sets up the GUI components.

- Polls news feeds, processes and filters stories, and updates the GUI.

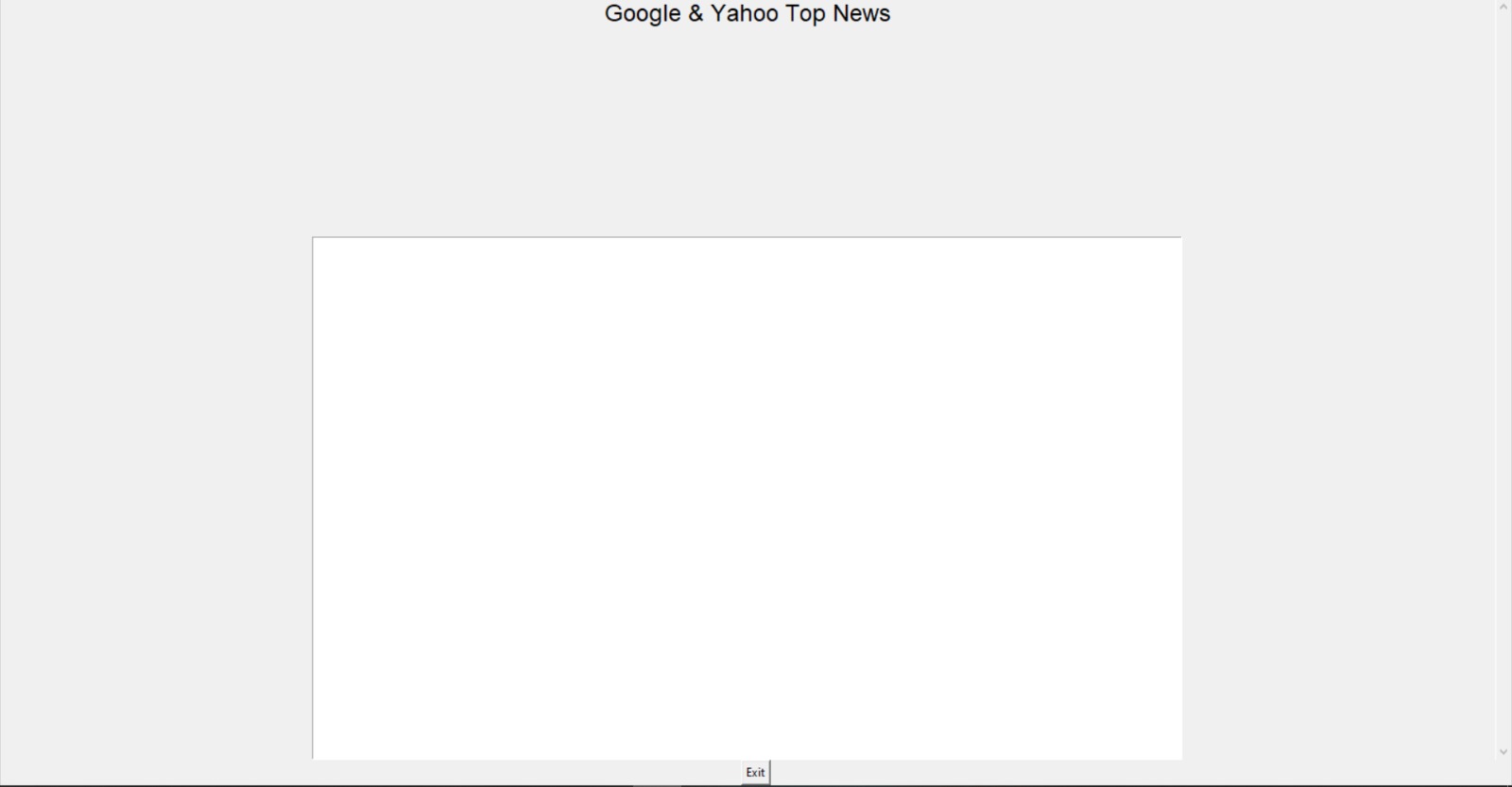
**Explanation:** This function runs the core logic of the application. It initializes the GUI, sets up the triggers, and continuously fetches and processes news feeds. The filtered stories are displayed in the GUI, and the function ensures the interface remains responsive by using a separate thread for fetching and processing news.

# **Output**

The program outputs a GUI

displaying filtered news stories based on the defined triggers. The GUI includes:

* + A title at the top.
  + A text area displaying the titles and descriptions of the filtered news stories.
  + A scrollbar for navigating through the stories.
  + An exit button to close the application.



**Conclusion**

This project demonstrates a comprehensive approach to retrieving, processing, and displaying news stories based on user-defined criteria. By utilizing object-oriented programming, multithreading, and GUI development, we created a responsive and extensible application. Users can easily add new triggers or modify existing ones, making the application adaptable to various needs and preferences. This project serves as a solid foundation for more advanced news filtering and notification systems. The modular design ensures that new features and enhancements can be integrated seamlessly, providing a robust tool for staying informed with relevant news updates.