Project Documentation: Global News Topic Tracker

1. Introduction

Brief Overview

The Global News Topic Tracker is a web application designed to aggregate and display news articles from various sources. It provides users with a centralized dashboard to track news on specific topics, from different sources, and from several countries.

Purpose and Goals

The primary goal of this project is to offer a streamlined and efficient way for users to consume news. It aims to cut through the noise of the internet by providing filtered and relevant news content based on user preferences.

Target Users or Audience

This application is intended for a general audience, including researchers, students, professionals, and anyone interested in staying informed about specific topics or world events.

Key Features and Business Value

- **Topic-based Filtering:** Allows users to select news from predefined topics like AI, Politics, Sports, etc.
- **Source Selection:** Users can choose to see news from specific outlets like Google News, Reuters, BBC, etc.
- **Country-based Filtering:** Narrows down the news to a specific country.
- **Modern UI:** A clean and user-friendly interface built with Streamlit.
- **Summarization (Partial):** Displays a snippet of the article content.

2. Technology Stack

- **Frontend:**
  - **Streamlit:** Used as the web application framework to create the user interface.
- **Backend & Data Scraping:**
  - **Python:** The core language for the application logic.
  - **feedparser:** A Python library used to parse RSS feeds from news sources.
- **AI & Summarization:**
  - **Ollama:** Used to run the `llama2` model for article summarization.
  - **Subprocess:** Python's built-in module to execute the Ollama command-line tool.

- **Styling:**
    - **CSS:** Custom CSS is used for styling the application to provide a better user experience.

3. System Architecture

Overall Architecture

The application follows a monolithic architecture. It is a single, self-contained Python application powered by Streamlit.

Components

- **Frontend (UI):** The user interface is built with Streamlit. It consists of a sidebar for filters and a main area to display the news articles.
- **News Scraper (`news_scraper.py`):** This module is responsible for fetching news articles from Google News RSS feeds based on the selected topic.
- **Summarizer (`summarizer.py`):** This module contains the logic to summarize article content using the Ollama and Llama2 model. *Note: The summarization feature is not fully integrated into the main application flow in the current version.*
- **Styling (`styles.css`):** A separate CSS file is used to define the visual style of the application, such as colors, fonts, and layout.

Textual Diagram of Data Flow

```
User Interface (Streamlit)
      |
      |-- 1. User selects filters (Topic, Source, Country)
      |
      v
Backend Logic (app.py)
      |
      |-- 2. Calls NewsScraper with the selected topic
      |
      v
NewsScraper (`news_scraper.py`)
      |
      |-- 3. Fetches RSS feed from Google News
      |
      v
Backend Logic (app.py)
      |
      |-- 4. Receives list of articles
      |
      v
User Interface (Streamlit)
      |
      |-- 5. Displays articles in a grid layout
```

4. Functionalities in Detail

Feature: News Filtering

- **Description:** Users can apply filters from the sidebar to customize the news feed.
- **Inputs:**
    - `Topic`: A string representing the news category (e.g., "AI", "Politics").
    - `Source`: A string for the news source (e.g., "Reuters", "BBC").
    - `Country`: A string for the country (e.g., "India", "US").
- **Outputs:** A filtered list of news articles displayed on the screen.
- **Dependencies:** `NewsScraper` module.

Feature: News Scraping

- **Description:** This feature scrapes news from Google News. It can fetch general news or search for a specific topic.
- **Inputs:** `topic` (optional string).

- **Outputs:** A list of dictionary objects, where each dictionary represents a news article containing `title`, `link`, `published` date, `source`, and `content`.
- **Dependencies:** `feedparser` library.

Feature: Article Summarization

- **Description:** This feature is intended to provide a summary of a news article. It uses the Ollama library to run the `llama2` model.
- **Inputs:** `article_content` (string).
- **Outputs:** A string containing the summarized text.
- **Dependencies:** `ollama` command-line tool.
- **Note:** In the current implementation, the main application displays a truncated version of the article content, not the AI-generated summary.

5. API Overview

The application does not expose its own API. It interacts with an external data source:

- **Google News RSS Feed:** The `NewsScraper` module sends GET requests to `https://news.google.com/rss` to fetch news data. The topic is appended as a query parameter (e.g., `https://news.google.com/rss/search?q=AI`).

6. Database Design

There is no database used in this project. The application is stateless and fetches the latest news from the source every time the "Fetch News" button is clicked.

7. Security

- **Authentication and Authorization:** The application is publicly accessible and does not have any authentication or authorization mechanisms.
- **Data Protection:** The application only handles publicly available news data. No user data is collected or stored.
- **Input Validation:** Basic input validation is handled by the Streamlit widgets, which restrict user input to the provided options.

8. Deployment and Environment Setup

Local Environment Setup

1. **Prerequisites:**
    - Python 3.7+

- `pip` (Python package installer)
- Ollama installed and running with the `llama2` model pulled (`ollama pull llama2`).

2. **Installation:**

   - Clone the project repository.
   - Install the required Python packages:

     ```
     pip install streamlit feedparser streamlit_extras
     ```

3. **Running the Application:**

   - Navigate to the project's root directory.
   - Run the following command:

     ```
     streamlit run app.py
     ```

   - The application will be accessible at `http://localhost:8501`.

Production Environment

The application can be deployed to any platform that supports Streamlit hosting, such as Streamlit Community Cloud, Heroku, or a private server with Nginx as a reverse proxy.

Configuration

There are no required environment variables for the application to run.

9. Testing

- **Testing Strategy:** The project does not currently include a formal testing strategy or any test files (e.g., unit tests, integration tests).
- **Tools:** No testing frameworks are currently configured.

10. Future Enhancements

- **Full Summarization Integration:** Integrate the `Summarizer` module into the main application flow to display AI-generated summaries for each article.
- **Expanded News Sources:** Add support for more news sources beyond Google News.
- **User Preferences:** Implement user accounts to allow users to save their preferred topics, sources, and other settings.
- **Database Integration:** Add a database (e.g., SQLite, PostgreSQL) to cache news articles, reducing redundant scraping and enabling historical data analysis.

- **Improved Error Handling:** Implement more robust error handling for network requests and the summarization process.
- **Automated Testing:** Add a suite of unit and integration tests to ensure code quality and reliability.
- **Asynchronous Operations:** Use asynchronous operations for news scraping to improve performance and prevent the UI from freezing.

11. Conclusion

The Global News Topic Tracker is a functional prototype that demonstrates the potential of combining web scraping and AI for personalized news consumption. While currently limited in scope, it provides a solid foundation for a more feature-rich and robust news aggregation platform. Its simple, monolithic architecture makes it easy to understand, maintain, and extend.