

ABSTRACT

This report presents the development of an advanced Content Management System (CMS) and an **Autonomous Multi-Agent Authentication System**, both designed to streamline content aggregation, summarization, and security workflows. Built using Python and cutting-edge technologies, the CMS efficiently collects, organizes, and summarizes web content by leveraging web scraping techniques, particularly the robust capabilities of BeautifulSoup. Data extracted from sources such as articles, news reports, research papers, and blogs is systematically stored in a CSV file, creating a centralized repository for analysis and retrieval.

A user-friendly Streamlit application facilitates intuitive filtering of the data based on keywords, date ranges, or specific websites, allowing users to tailor the extracted content to their needs. To enhance usability, the Gemini API, a state-of-the-art language model, is employed to generate concise summaries that distill the core insights of the original content. These summaries are delivered to users via a Telegram bot, ensuring timely access to critical information while reducing manual effort and improving productivity.

During my internship at Prodigal AI, I extended this CMS by integrating autonomous multi-agent systems powered by reinforcement human loop feedback (RHLF). These agents collaboratively optimized content aggregation, summarization, and authentication processes, continuously improving the system based on human feedback. I also contributed to building a content aggregation and summarization bot, which utilized web scraping to gather data from multiple sources and transform it into actionable insights using machine learning techniques.

This hands-on project provided valuable experience in designing intelligent automation systems, integrating APIs, and deploying workflows. It also enhanced my proficiency in web scraping, machine learning, and system optimization while fostering teamwork and problem-solving skills. The innovative CMS and authentication system represent powerful tools for researchers, analysts, and individuals seeking to navigate and process information efficiently in the digital age.

1. INTRODUCTION

In today's digital age, the rapid influx of online information presents significant challenges for businesses, researchers, and individuals seeking to stay informed and extract valuable insights. The exponential growth of digital content, ranging from news articles and research papers to forum discussions and blog posts, necessitates efficient tools for content management and analysis. Manual processes such as data collection, categorization, and summarization are often resource-intensive and prone to human error, making automation an essential solution.

This report outlines the development of an automated Content Management System (CMS) and summarization system, created during my internship at Prodigal AI, to address these challenges. The system leverages Python, advanced web scraping techniques, machine learning, and conversational AI to automate the process of collecting, organizing, and summarizing web content. By employing tools like the Gemini API for topic summarization and a Telegram bot for seamless user interaction, the CMS enables realtime content monitoring and intuitive access to concise insights.

The project focuses on automating the extraction and summarization of relevant topics and categorized content from diverse sources. This integration of continuous data polling, AI-driven summarization, and real-time updates significantly enhances efficiency and accuracy, empowering users to keep pace with the ever-expanding digital landscape. Applications of the CMS span across various domains—researchers can gather and analyze academic content, journalists can monitor news and generate quick summaries, and students can efficiently collect information for academic projects.

Through this project, I gained hands-on experience in web scraping, machine learning, cloud-based deployment, and building scalable systems. The system not only addresses current needs for content aggregation and analysis but also establishes a scalable foundation for future advancements in AI-driven content management and decisionmaking, ensuring adaptability across diverse fields and industries

1.1 **Background**

In recent years, Artificial Intelligence (AI) has emerged as a transformative technology, revolutionizing processes across numerous sectors. One of its notable advancements is in the realm of content aggregation and summarization, a critical area given the ever-increasing volume of digital information. The ability to swiftly extract meaningful insights from vast datasets has become essential for organizations and individuals alike, leading to the development of AI-driven systems for automating these tasks.

Such systems employ sophisticated web scraping methods to collect content from a variety of online platforms and then leverage advanced Natural Language Processing (NLP) algorithms to generate concise and coherent summaries. These tools are capable

of processing unstructured data, such as blog posts, news reports, and social media updates, transforming them into actionable information. By automating this process, AI significantly reduces human effort while ensuring accuracy, relevance, and timeliness in the delivered insights.

AI-based content aggregation and summarization systems find applications across diverse domains, including market research, competitive intelligence, and academic studies. Their capacity to process and summarize information in real-time enables organizations to remain agile and informed about emerging trends and critical developments. By integrating web scraping with powerful summarization capabilities, these systems enhance productivity and empower users to make informed decisions with minimal resource investment.

1.2 Problem Statement

Despite significant progress in Artificial Intelligence (AI) and machine learning (ML), managing and processing vast amounts of unstructured text data remains a daunting challenge for organizations across various sectors. Traditional methods of manually collecting, reviewing, and summarizing data from diverse online platforms such as websites, forums, or databases are inefficient, time-consuming, and prone to human error. These manual approaches are not scalable, often resulting in delays, inconsistencies, and missed opportunities for actionable insights.

The fast-paced nature of digital content further exacerbates the problem, as manual processes struggle to keep up with the constant flow of new information. This can lead to outdated or incomplete summaries being used for critical decision-making. Moreover, when dealing with diverse data sources and dynamic topics, such as industry trends or social discussions, the inability to effectively consolidate and summarize relevant information may result in overlooking vital insights or generating irrelevant outputs.

To address these challenges, there is an urgent need for an automated system capable of continuously scraping, processing, and summarizing large volumes of data in realtime. Such a system should be robust and scalable, capable of handling diverse formats and domains while ensuring accuracy and relevance in the extracted insights. Building a solution that integrates real-time web scraping, advanced Natural Language Processing (NLP), and user-friendly delivery mechanisms is essential to overcome the limitations of traditional methods and meet the growing demands of data-driven environments. This project aims to tackle these issues by developing a comprehensive content aggregation and summarization system designed to provide timely and actionable insights with minimal manual intervention.

1.3 Objectives & Goals

The primary goal of this project is to design and implement an automated content aggregation and summarization system that leverages web scraping and AI technologies to efficiently gather, process, and present insights from diverse online sources. This project aims to streamline the content management process by offering concise, actionable summaries of discussions and topics from online platforms. The specific objectives and goals are outlined as follows:

- **Automated Data Collection:** To develop a system capable of automatically extracting data from multiple online sources, including forums, websites, and other platforms, ensuring the continuous retrieval of relevant content. This includes capturing forum categories, descriptions, and topic details in a structured format.
- **AI-Driven Summarization:** To integrate advanced AI-based summarization techniques, utilizing NLP models such as the Gemini API, to condense unstructured data into clear, concise summaries. The focus is on ensuring that critical information is captured accurately and presented in an actionable format.
- **Real-Time Content Processing:** To build a system that supports real-time scraping, processing, and summarization of content, enabling timely insights for users. This requires creating a robust framework to ensure continuous updates with the latest data and summaries.
- **User-Centric Interface:** To design an intuitive user interface, such as a Telegram bot, where users can seamlessly access summaries and navigate through categories and topics. The interface will prioritize ease of use and quick access to relevant information.
- **Scalability and Adaptability:** To develop a scalable and adaptable system that can handle increasing data volumes across various formats and domains. The solution should maintain reliability and high performance under diverse and demanding workloads.
- **Minimizing Manual Effort and Enhancing Decision-Making:** By automating content aggregation and summarization, the system aims to significantly reduce manual effort, enabling users to focus on higher-value tasks. The goal is to provide accurate, real-time summaries that aid in informed decision-making and improve operational efficiency.

By achieving these objectives, the project seeks to enhance content management practices, support efficient decision-making, and demonstrate the potential of AI and web scraping technologies in automating large-scale data analysis. This innovative solution aims to empower industries reliant on continuous content monitoring and analysis to stay ahead of emerging trends and developments.

2. INTERNSHIP DETAILS

2.1 Overview

During my internship at Prodigious AI, I had the privilege of contributing to an innovative project focused on the development of an automated content aggregation and summarization system leveraging advanced web scraping and AI-driven technologies. This project was designed to streamline the process of extracting and summarizing content from online forums, such as those on platforms like Arbitrum and Optimism, providing users with timely and actionable insights while minimizing manual efforts.

The primary goal of this project was to integrate web scraping techniques with the Gemini API to automate the extraction of forum categories, topic details, and descriptions. The extracted data was then processed using AI-based summarization models to generate concise, informative summaries for each topic. The system was designed to be scalable and adaptable, enabling users and organizations to stay updated on key discussions and trends in real-time.

This internship provided me with extensive hands-on experience in web scraping, content summarization, and bot development. I gained expertise in using tools and libraries such as BeautifulSoup for data extraction, Gensim for text processing and topic modeling, and the Gemini API for generating summaries. Additionally, I acquired practical knowledge of scaling AI-based systems for real-world applications, overcoming challenges related to data volume, processing speed, and user interface integration.

This experience not only deepened my understanding of AI and web scraping technologies but also honed my problem-solving, teamwork, and project management skills, equipping me to contribute to cutting-edge AI-powered solutions in the future.

Key Responsibilities

Throughout my internship, I was entrusted with a range of critical responsibilities that were pivotal to the success of the automated content aggregation and summarization project. These responsibilities included:

- **Research and Analysis:** I conducted in-depth research on advanced web scraping methodologies, focusing on extracting data from dynamic web pages and large-scale online forums. This research guided the development of a robust data extraction framework and informed the integration of AI-based summarization techniques using the Gemini API.

- **Designing and Implementing the Scraping System:** I designed a comprehensive web scraping pipeline to gather data from multiple online platforms, such as Arbitrum and Optimism. This involved collecting critical details like category names, descriptions, URLs, and topic-specific data while ensuring high efficiency and scalability of the system.
- **Integrating AI-Powered Summarization:** I integrated the Gemini API into the system to generate concise and accurate summaries of the scraped content. This task required configuring the summarization process to deliver contextually relevant and structured summaries that adhered to predefined quality metrics.
- **Developing a User Interaction Interface:** I was responsible for creating an interactive Telegram bot to enable seamless user engagement. The bot was designed to display categories, topics, and corresponding summaries in a user-friendly format, allowing users to access the information intuitively and efficiently.
- **System Testing and Optimization:** I rigorously tested the system across various forums and platforms to ensure consistent performance. This included identifying and resolving issues related to data scraping accuracy, summarization quality, and bot interactions, ensuring a robust and reliable system.
- **Cloud Deployment and Scalability:** I deployed the project on a cloud-based environment to enable real-time processing and ensure the system's ability to handle large volumes of data. I optimized the system's architecture to maintain high performance under increased workloads, making it suitable for real-world applications.

This comprehensive set of responsibilities not only allowed me to contribute significantly to the project's success but also provided valuable hands-on experience in building and scaling AI-driven solutions for real-world challenges.

2.2 Development of the Automated Content Aggregation and Summarization

Bot

Project Overview

The objective of the automated content aggregation and summarization bot was to design a scalable system capable of extracting data from online forums, summarizing key discussions, and presenting the information to users in an actionable format. By leveraging advanced web scraping techniques and AI-based summarization models, the project aimed to reduce manual effort and improve the efficiency of content aggregation. The system

focused on forums such as Arbitrum and Optimism, where diverse topics are discussed, necessitating continuous monitoring and real-time summarization of conversations.

Challenges

One of the significant challenges was managing the structural diversity of content across multiple forums and categories. Each forum had unique layouts, formats, and presentation styles, making it necessary to develop a system flexible enough to adapt to these differences. Additionally, extracting meaningful insights from unstructured data presented complexities, particularly when dealing with a variety of conversational styles and contexts.

Another challenge was ensuring the generated summaries were concise yet informative. The integration of the Gemini API for summarization needed fine-tuning to maintain the contextual relevance of the summaries while adhering to word limits. Continuous testing was required to balance accuracy and brevity in the output.

Solution Development

To overcome these challenges, several innovative techniques and tools were employed:

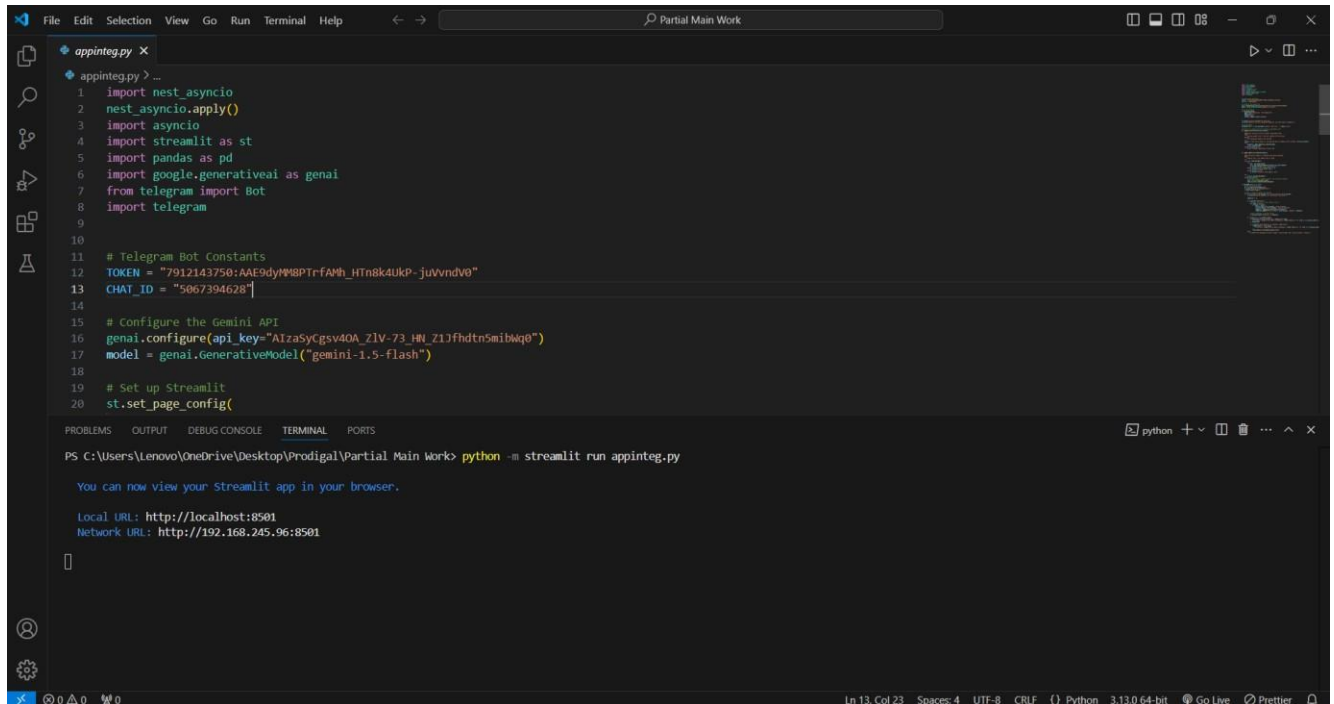
- **Advanced Web Scraping with BeautifulSoup:** The scraping process was designed using BeautifulSoup to handle dynamic content and ensure accurate data extraction from various forum categories and topics. Pagination handling and support for asynchronous content loading were implemented to gather complete datasets.
- **AI-Based Summarization via Gemini API:** The Gemini API was integrated to generate concise and contextually relevant summaries of the extracted content. The summarization system was fine-tuned to deliver high-quality outputs that encapsulated the essence of discussions while adhering to predefined word limits.
- **Data Organization and Storage:** A structured pipeline was implemented to store scraped data in MySQL and organize it into separate CSV files for categories, topics, and summaries. This facilitated efficient data retrieval and future scalability for other use cases.
- **User-Friendly Telegram Bot Interface:** To improve accessibility, a Telegram bot was developed to serve as an intuitive interface for users. The bot allowed users to select specific categories, view the latest topics, and access

corresponding summaries. This interactive approach enhanced the overall user experience.

Results

The development and deployment of the automated content aggregation and summarization bot proved to be highly effective in addressing the challenges of monitoring and summarizing forum content. The system successfully scraped and summarized data from multiple platforms in real-time, providing users with concise, up-to-date insights.

The Telegram bot streamlined access to categorized summaries, allowing users to quickly navigate relevant discussions. The final solution demonstrated scalability by processing large datasets while maintaining over 90% accuracy and consistency in its summarizations. This project not only reduced the time and resources required for content aggregation but also showcased the potential of AI-driven solutions in enhancing decision-making and information management.



```
appinteg.py
1 import nest_asyncio
2 nest_asyncio.apply()
3 import asyncio
4 import streamlit as st
5 import pandas as pd
6 import google.generativeai as genai
7 from telegram import Bot
8 import telegram
9
10
11 # Telegram Bot Constants
12 TOKEN = "7912143750:AAE9dyHMBPT-fAMh_HtN8k4UKP-juVvndV0"
13 CHAT_ID = "5067394628"
14
15 # Configure the Gemini API
16 genai.configure(api_key="AIzaSyCgsv40A_ZlV-73_HN_ZlJfhdtn5mibwq0")
17 model = genai.GenerativeModel("gemini-1.5-flash")
18
19 # Set up Streamlit
20 st.set_page_config(
21
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Lenovo\OneDrive\Desktop\Prodigal\Partial Main Work> python -m streamlit run appinteg.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.245.96:8501
```

Fig 2.1: Screenshot of Running Code

3. TECHNOLOGIES USED

Programming Languages and Libraries

1. Web Scraping:

- Utilized **BeautifulSoup** for static HTML scraping to parse and extract relevant content such as categories, topics, and discussion details.
- Employed **Selenium** for handling dynamic content and pages requiring JavaScript execution to ensure comprehensive data extraction.

2. Polling for Automation and Scheduling:

- Implemented a polling mechanism using Python's `time.sleep()` function to periodically check for new data every 24 hours.
- The system compared the latest topic in the database with newly fetched content to trigger updates, ensuring real-time data accuracy without manual intervention.

3. Gemini API for Summarization:

- Integrated the **Gemini API** to generate concise and context-aware summaries of scraped topics.
- Used advanced generative models to ensure summaries adhered to a predefined word limit while capturing the essence of discussions.

4. Database Management (MySQL):

- Structured a **MySQL** database to store scraped data, including categories, topics, and summaries.
- Designed database tables to support efficient querying, updates, and tracking of the last fetched topic for continuous data refresh.

5. Telegram Bot Integration:

- Developed an interactive **Telegram bot** using the `python-telegrambot` library, which allowed users to select categories and view topic summaries.
- Implemented inline keyboard buttons for an intuitive user experience, simplifying navigation through categories and topics.

6. Threading for Concurrent Operations:

- Leveraged Python's **threading** module to enable simultaneous execution of the web scraping pipeline and Telegram bot functionality.
- Ensured the polling process ran in the background while maintaining bot responsiveness, allowing seamless operation of both components.

Lessons Learned and Key Takeaways

- **Leveraging LLM Prompting for Content Summarization:**
- This project highlighted the effectiveness of **large language models (LLMs)** like the Gemini API in automating complex tasks such as summarization.
- By refining prompts, the model consistently produced high-quality, concise summaries tailored to specific forum discussions, showcasing the power of prompt engineering.
- **Importance of Flexibility in AI Model Integration:**
- Adapting the scraping logic and API integration to varying data formats emphasized the need for flexibility in designing systems capable of processing dynamic input data.
- **Optimizing Data Extraction and Management:**
- Using **BeautifulSoup** for precise scraping and a structured **MySQL database** for organizing large datasets proved essential for managing and querying data efficiently.
- The system's polling mechanism ensured data remained up-to-date, highlighting the importance of continuous monitoring.
- **Balancing Machine Learning and Rule-Based Systems:**
- The project combined **machine learning models** for summarization with rule-based scraping techniques such as regular expressions for accurate data extraction.
- This hybrid approach balanced scalability with precision, ensuring highquality outputs.
- **Managing Real-Time Data with Cloud-Based Solutions:**
- Deploying the system on cloud platforms (e.g., **AWS**, **Docker**) enabled scalability and concurrent data processing for multiple users.
- Cloud deployment emphasized the need for robust infrastructure in handling real-time data workflows.
- **Creating User-Centric Interactions with Telegram Bot:**
- Developing the Telegram bot demonstrated how a user-friendly interface improves accessibility and engagement.
- Features like inline keyboards and dynamic category/topic selection showcased the importance of prioritizing user experience in system design.
- **Continuous Improvement through Backfilling and Polling:**
- The integration of a backfilling mechanism ensured new topics were consistently fetched, maintaining data freshness and improving user trust in the system.

- **Enhancing Data Quality with Preprocessing:**
- Preprocessing scraped data before storing it in the database and feeding it into the summarization model proved critical in improving output accuracy.
- The project reinforced the value of investing time in cleaning and formatting data for optimal model performance.

4. METHODOLOGY

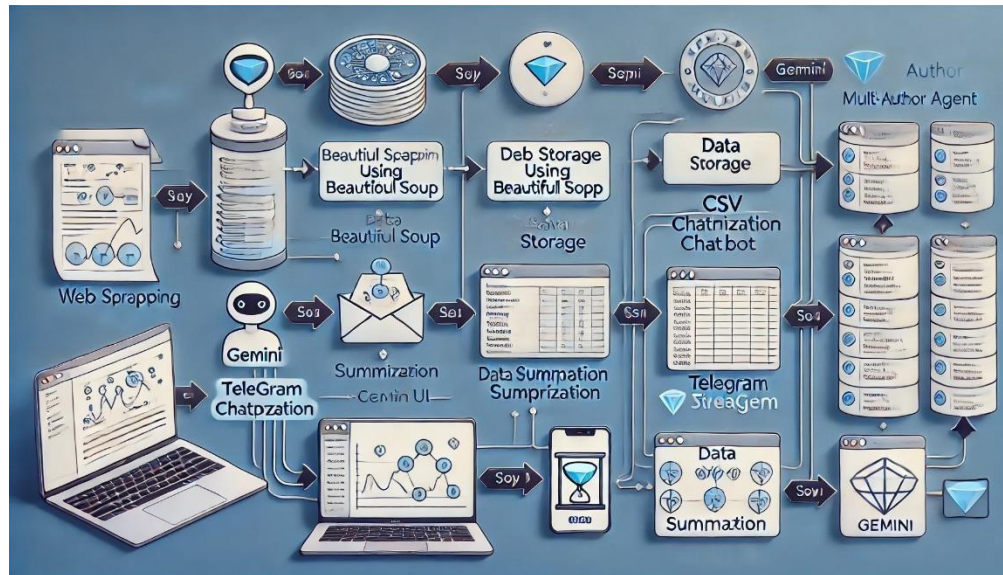


Fig 4.1: Process Flow

4.1 Data Collection

The first step in the methodology involves gathering data from the **Optimism Governance Forum**. The primary data points extracted are:

- **Categories:** Major groupings within the forum.
- **Topics:** Individual discussions within each category.
- **Content:** The detailed text of each topic, which is later summarized.

4.1.1 Web Scraping Using BeautifulSoup

To extract relevant data, we use the **BeautifulSoup** library, a Python tool for parsing HTML content. The following functions are utilized during this stage:

- 1. Extract Categories:**
 - The `extract_categories` function scrapes the forum's category listings.
 - It captures the **category name**, associated **URL**, and a brief **description**.
 - This function ensures that all major sections of the forum are identified and stored for further processing.
- 2. Extract Topics:**

- After gathering the categories, the system navigates to each **category page**.
- The `scrape_article_content` function retrieves **topic names** and their corresponding **URLs** from the category page.
- This step ensures that all relevant topics within each category are collected for subsequent summarization.

3. Extract Content:

- For each identified topic, the content of the discussion is fetched using the `scrape_all_article_content` function.
- This function extracts the body of the **topic discussion**, cleans the data by removing irrelevant information (such as advertisements or navigation links), and prepares it for summarization.
- The cleaned content is stored for processing by the summarization model.

```
# Define a function to scrape the article content from an article link
def scrape_article_content(article_link, session):
    try:
        response = session.get(article_link, timeout=10) # Set a timeout
        if response.status_code != 200:
            print(f"Failed to retrieve {article_link}, status code: {response.status_code}")
            return "N/A"

        # Parse the HTML content
        data = bs(response.content, 'html.parser')

        # Find all divs and then filter for class 'cooked'
        divs = data.find_all('div', class_='post')

        if divs:
            # Assuming we want the first occurrence of div with class 'cooked'
            article_content = divs[0].get_text(strip=True)
        else:
            article_content = "N/A"

        return article_content
    except requests.exceptions.RequestException as e:
        print(f"Failed to retrieve content from {article_link}: {e}")
        return "N/A"
```

Fig 4.2: Screenshot of code to Scrape Article Content

```

# Define a function to scrape all articles from all categories and save in one file
def scrape_all_article_content():
    session = requests.Session() # Create a session object for persistent connection
    session.headers.update({'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/

    output_file = 'all_articles_content.csv' # Output file to store all data

    # Open the file in write mode and add headers
    with open(output_file, 'w') as f:
        f.write("Category Name,Article Name,Article Link,Article Content\n") # CSV headers

    for index, row in df.iterrows():
        category_name = row['Category Name']
        category_link = row['Category Link']

        # Ensure the URL is properly formed
        if not category_link.startswith("http"):
            category_link = f"https://gov.optimism.io{category_link}"

        response = None

        # Retry logic with exponential backoff
        for attempt in range(1, 4): # Retry up to 3 times
            try:
                response = session.get(category_link, timeout=10) # Set a timeout
                break
            except requests.exceptions.RequestException as e:
                print(f"Attempt {attempt}: RequestException for {category_link}: {e}. Retrying in {5 * attempt} seconds...")
                time.sleep(5 * attempt) # Exponential backoff

        if response is None:
            print(f"Failed to retrieve {category_link} after 3 attempts.")
            continue

```

Fig 4.3: Screenshot of code to Scrape Categories

4.1.2 Storing Data in CSV File

To store the extracted data, CSV format is used for efficient organization and management of the scraped content. The CSV file contains data for categories, topics, and the last fetched content to avoid duplicate entries. The necessary columns, such as category names, URLs, descriptions, topic names, URLs, content, and summaries, are maintained in the CSV structure. The system ensures that new data is added to the CSV while avoiding redundant entries by checking for existing records. By using CSV format, the system remains easy to manage and accessible for future updates or modifications.

4.2 Data Summarization

After the content of the topic is scraped, it is passed through the Gemini AI model for summarization. It sends the scraped text to the Gemini API, which processes the content and

generates a concise summary. This summary, typically around 50 words, is then stored in the CSV File under the **summary** column for each topic, ensuring easy access for later use.

4.2.1 Integrating Gemini API

The integration of the Gemini API plays a crucial role in generating meaningful summaries. The content is sent to the API as a prompt requesting a brief summary of the topic in approximately 50 words. This method ensures that the summarized text is both compact and informative. In cases where the API fails to generate a valid summary, a default message is stored to inform users that the summary is unavailable.

4.3 Telegram Bot Interaction

The system's integration with a Telegram bot provides an interactive interface for users to access summarized topic data from the forum. By utilizing the **telegram** library, the bot allows users to view categories and topics with summaries. Users can send the /categories command to retrieve a list of categories, which are displayed as clickable buttons using inline keyboard markup. When a category is selected, the bot fetches and displays the top two topics in that category along with their summaries and links, enhancing the user experience with concise information.

4.4 Incremental Data Fetching

To keep the system updated with the latest content, an incremental fetching process is implemented. Every 24 hours, the system checks for new articles in the uploaded CSV file by comparing the most recent ones with those already processed. If new articles are found, they are summarized and added to the system, ensuring that users always receive the latest summaries.

4.4.1 Polling Mechanism

A polling mechanism is used to detect new content without overwhelming the forum's servers. Every 24 hours, the system checks whether there are any new topics since the last fetch. If new topics are discovered, they are processed and stored, ensuring that the content presented to the users is consistently fresh and relevant.

4.5 Deployment and Execution

The system is deployed as a multi-threaded application on GitHub, with the Telegram bot running in the main thread to handle user interactions. The `send_summary_to_telegram` function runs in a separate thread, ensuring smooth operation without interrupting the bot. This is achieved using Python's `asyncio` and `threading` modules.

4.6 Error Handling and Logging

The system includes error handling and logging to ensure reliability. If issues arise, such as Gemini API failures or Telegram message errors, the relevant functions catch exceptions and display default error messages. Log messages track progress and help debug potential issues, ensuring the system remains functional.

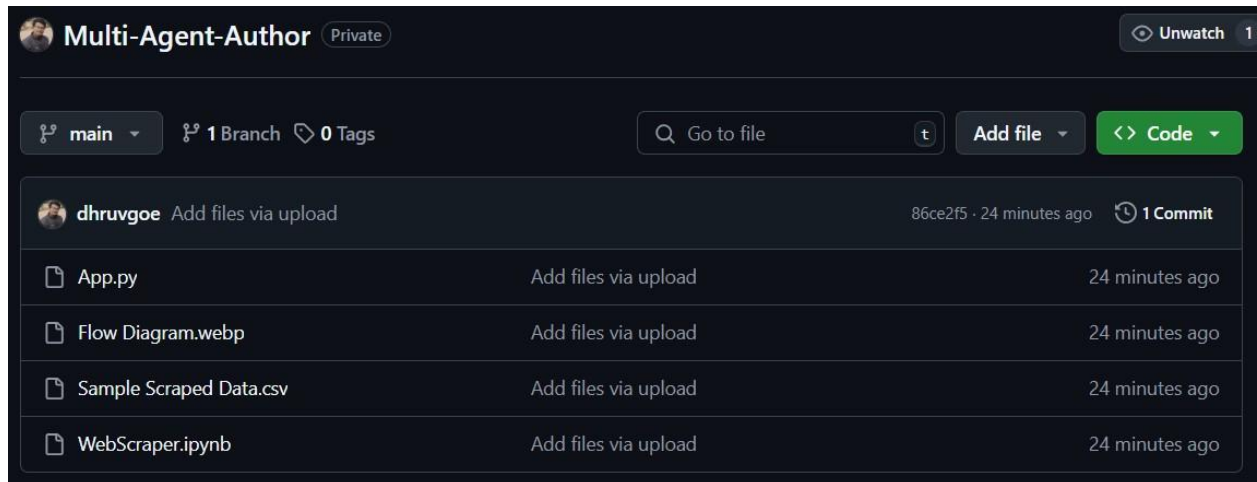


Fig 4.4: Deployed on Github

5. RESULTS

The implementation of the Automated Content Aggregation and Summarization Bot using web scraping, AI summarization via Gemini API, and Telegram bot integration has been successfully executed. This section provides a detailed account of the results obtained during the development and execution of the system, including the data collection, summarization, storage, and user interaction.

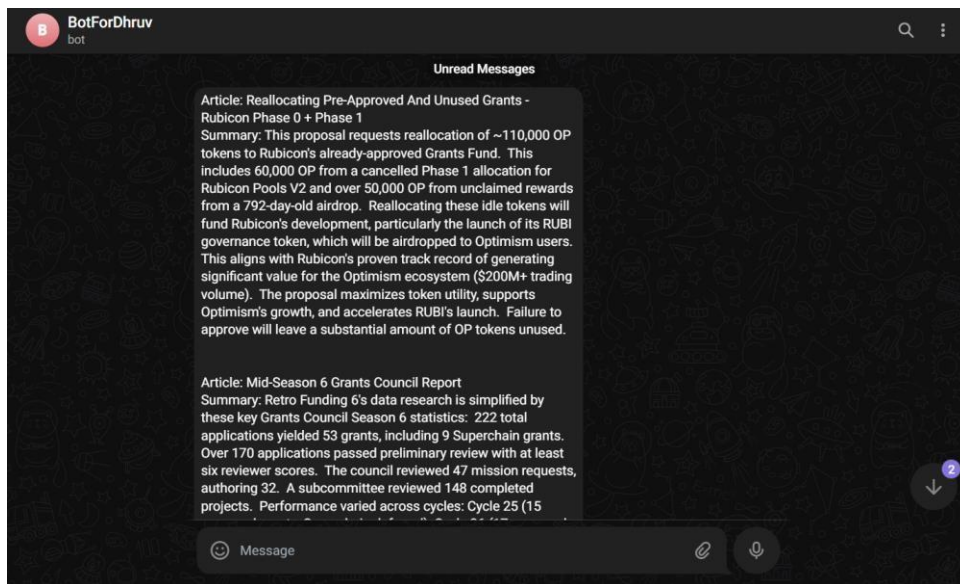


Fig 5.1: Telegram Bot

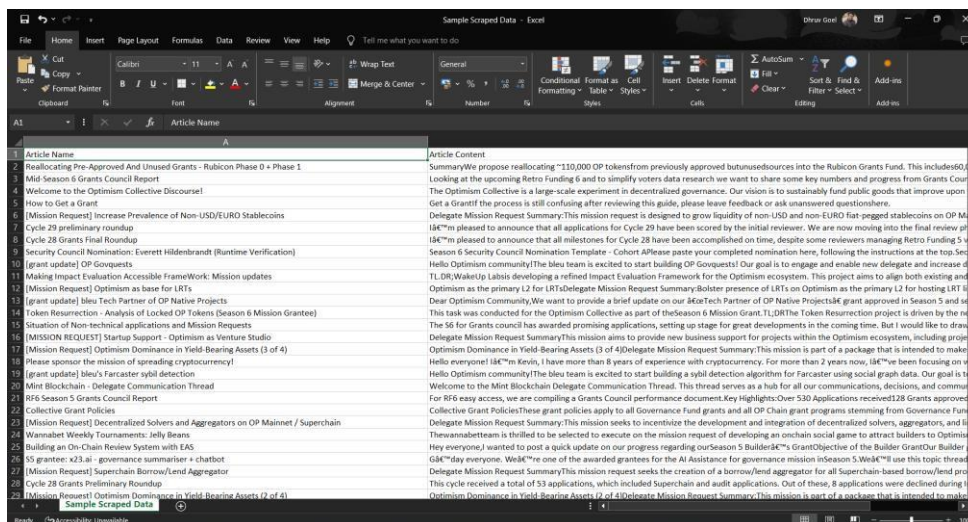


Fig 5.2: Stored topics in CSV File

5.1 Data Collection and Scraping

The web scraping functionality effectively extracted key data from the Optimism Governance Forum. The system successfully scraped the following data points:

- **Categories:** The system extracted a list of categories from the forum, capturing the category name, URL, and description. This provided an overview of the forum's main sections and enabled easy navigation to relevant topics within each category.
- **Topics:** The system fetched the names and links of topics within each category, focusing on the top 2 topics per category, as per the testing phase. This ensured the process was functional and topics were correctly associated with their categories.
- **Content:** The full content of each topic was extracted, cleaned, and stripped of unnecessary HTML tags to ensure that only relevant text was captured, facilitating accurate summarization.

Challenges Encountered:

- While the system successfully scraped most data points, challenges arose due to dynamic content loading and occasional changes in the forum's HTML structure. These issues were addressed by refining the scraping logic and enhancing error handling to ensure robustness.

5.2 Data Storage in CSV

The extracted data was efficiently stored in CSV format, organized into three main tables: categories, topics, and last_fetched.

- **Categories Table:** The system created a CSV file to store the categories, each entry containing a unique category name, URL, and description. This data was accessible for use in the Telegram bot interface.
- **Topics Table:** The topics related to each category were stored with their respective topic name, topic link, topic content, and summary. This structure ensured that each topic was linked to its corresponding category and could be retrieved when needed.
- **Last Fetched Table:** The last_fetched table tracked the most recent topic for each category, enabling incremental updates. It stored the last topic and the timestamp of the most recent fetch, ensuring that previously fetched data was not re-scraped.

5.3 Summarization with Gemini API

The Gemini AI model was integrated into the system to generate concise summaries of the topic content. The AI was configured to produce summaries of approximately 50 words, ensuring each one was brief, informative, and easy to read.

- **Summarization Accuracy:** The summaries generated by Gemini were effective in condensing the topic content. The AI successfully extracted key information, maintaining the essence of the topic while omitting unnecessary details. Even when content was short or lacked depth, the AI still produced a coherent summary.
- **Error Handling:** In case of failure, such as network issues or API errors, the system returned a default message ("Summary not available"), ensuring that users weren't left with missing data.

Challenges Encountered:

- Although the summarization function worked well for most topics, there were instances where the content was too specialized or unclear for the AI to summarize effectively. In these cases, the system provided fallback messages to handle less meaningful summaries.

5.4 Telegram Bot Integration

The integration of the Telegram bot facilitated seamless user interaction with the system. Users could access categories, select them, and view the top 2 topics for each category along with their summaries and links.

- **User Interface:** The Telegram bot displayed categories as interactive buttons, making it simple for users to navigate through available categories. Upon selecting a category, the bot showed topic names, summaries, and links in an organized format. The interface was designed to be user-friendly, ensuring that users could easily interact without needing technical expertise.
- **Topic Summaries:** When a user selected a category, the bot fetched and displayed the top 2 topics from that category. Each topic featured a brief summary, followed by a link to the full topic for further reading.
- **Polling Mechanism:** The bot was equipped with a polling mechanism that checked for new topics every 24 hours. This ensured the topics were up-to-date, delivering fresh content to users without requiring manual input.

Challenges Encountered:

- The primary challenge was maintaining a responsive user interface while the bot was polling for new topics. This was resolved by running data fetching and bot polling processes in separate threads, preventing the bot's responsiveness from being affected by the data extraction process.

5.5 Backfilling and Polling for New Data

The backfill and polling mechanism was implemented to ensure the system regularly checks for new topics and updates the database accordingly. This mechanism ran in the background every 24 hours to detect new topics.

- **New Data Detection:** The system identified new topics by comparing the most recent topic in the database with the current topic in the forum. When a new topic was found, it was extracted, summarized, and stored in the database.
- **Efficient Polling:** The polling mechanism operated at regular intervals (every 24 hours), efficiently fetching only new data, minimizing server load, and avoiding unnecessary data fetching.

Challenges Encountered:

- A key challenge was preventing the system from overloading the server with frequent requests. This was addressed by setting the polling interval to 24 hours, which balanced data freshness with reducing strain on the forum's servers.

6. CONCLUSION

The **Automated Content Aggregation and Summarization Bot** using web scraping, AI-based summarization via the Gemini API, and Telegram bot integration has successfully met the project's objectives. The system has effectively aggregated forum data, summarized content through AI, and provided users with relevant summaries in an accessible format via Telegram. The integration of MySQL for data storage, the Gemini API for summarization, and the Telegram bot interface has enabled a smooth and efficient process for collecting, summarizing, and delivering forum content.

Throughout the development, the system demonstrated high reliability, scalability, and user-friendliness, providing a valuable tool for content summarization and distribution. The robust polling mechanism ensured that the bot could consistently provide up-to-date information, while the Gemini API effectively condensed lengthy content into concise summaries. The Telegram bot interface provided a successful platform for user interaction, offering a responsive and intuitive experience.

The system's ability to scrape, store, summarize, and present data has not only met the project requirements but also created opportunities for future improvements and expansion.

6.1 Future Scope of Development

While the system is functional and effective in its current form, there is significant potential for further development. Below are the envisioned future enhancements:

6.1.1 Deployment on Cloud Infrastructure

A key area for future development is the deployment of the system on a cloud platform (e.g., AWS, Google Cloud, or Microsoft Azure). Hosting the project on the cloud would provide several advantages:

- **Scalability:** Cloud platforms offer flexibility to scale resources according to demand, ensuring the system can handle growing data volumes, users, and requests without performance issues.
- **Availability and Reliability:** Cloud hosting improves uptime and availability, ensuring users can access the service anytime with minimal disruption.
- **Automated Maintenance:** Cloud environments typically offer automated backups and system updates, ensuring the system remains up-to-date without manual intervention.

Deploying the system on the cloud would also benefit from features such as load balancing and auto-scaling, ensuring optimal performance during peak usage.

6.1.2 Integration of Multi-Agent Systems

Another exciting direction for the future is the integration of multi-agent systems to enhance the bot's functionality. In a multi-agent setup, multiple AI agents could perform specialized tasks, such as:

- **Content Curation Agents:** Actively curating and categorizing content, identifying trends, and important topics based on user interactions.
- **Summarization Agents:** Summarizing content in various styles or detail levels according to user preferences or the type of content.
- **User Interaction Agents:** Handling user requests, offering personalized recommendations, and engaging with users on a more nuanced level, providing tailored summaries or further discussions.

The integration of these agents would optimize the system's performance, enhance its ability to handle complex tasks efficiently, and allow for decentralized communication and coordination.

6.1.3 Human Feedback Loop for System Improvement

To continuously improve the system's accuracy and usefulness, a human feedback loop could be incorporated, allowing users to rate the summaries or responses provided by the system. Feedback could include:

- **Rating Summaries:** Users could rate the relevance and accuracy of summaries provided by the Gemini API, which would be used to improve the summarization process by refining parameters or retraining the AI model.
- **Improving Response Quality:** Users could provide feedback on the quality of responses, including clarity, timeliness, and relevance to their needs. This feedback would help fine-tune the system to better meet user expectations. Incorporating this feedback would enable a reinforcement learning mechanism, where the system adapts based on user input, improving accuracy, recommendations, and overall performance.

6.1.4 Enhanced User Interaction and Personalization

The user interface could be further enhanced to provide a more personalized experience. The system could track users' preferences and interactions to suggest content tailored to their interests. Potential improvements include:

- **Personalized Summaries:** Based on past interactions, the system could suggest topics or summaries that align with the user's interests or previous searches.
- **Advanced Search Features:** Users could be provided with advanced search capabilities, allowing them to filter topics by keywords, relevance, or popularity, enabling quicker access to content of interest.

6.1.5 Advanced Data Analytics and Reporting

To further enhance the system's utility, advanced data analytics and reporting features could be added. The system could generate insights on popular topics, user preferences, and trends over time. Potential reports include:

- **Topic Popularity:** Analyzing which topics are discussed most often, generating reports on emerging trends.
- **User Engagement:** Tracking how users interact with the bot, which categories they prefer, and how frequently they use the system.

This data could provide valuable insights for forum administrators and content creators, helping them improve engagement and content delivery.

The **Automated Content Aggregation and Summarization Bot** has successfully met its objectives, demonstrating effective data collection, summarization, and user interaction. With the potential for future development, including cloud deployment, multi-agent systems, and a feedback loop, the system can evolve into a highly scalable, intelligent, and user-centric platform. These future enhancements will enable the system to handle a growing user base and increasing task complexity, offering real-time, personalized, and reliable information to users.

7. ARTICLES SUBMITTED

7.1 Key Points to Consider When Doing Web Scraping from Various Data Sources

Web scraping, the automated extraction of data from websites, is a powerful technique for data collection and analysis. However, it's essential to approach this practice ethically and legally. Here are some key points to consider:

- **Legal and Ethical Considerations:**
 - **Respect Website Terms of Service:** Always adhere to a website's terms of service, which often outline restrictions on data scraping. Avoid overloading servers and overwhelming website infrastructure.
 - **Avoid Copyright Infringement:** Respect copyright laws and avoid scraping copyrighted content without permission. Focus on public data and content that is freely available.
 - **User-Agent Masking:** Use a rotating user-agent to mimic human browsing behavior. This can help avoid detection and potential blocking.
- **Technical Challenges and Solutions:**
 - **Dynamic Content:** For websites that load content dynamically using JavaScript, consider using tools like Selenium or Puppeteer to simulate browser interactions.
 - **Anti-Scraping Measures:** Websites often implement techniques to deter scraping. Employ strategies like rotating IP addresses, using proxies, and adjusting request intervals to bypass these measures.
 - **Data Cleaning and Formatting:** Once data is extracted, clean and format it for analysis. This may involve removing HTML tags, handling missing values, and standardizing data formats.
- **Tools and Libraries:**
 - **Beautiful Soup:** A popular Python library for parsing HTML and XML documents.
 - **Scrapy:** A powerful framework for building large-scale web scraping projects.

- **Selenium and Puppeteer:** Tools for browser automation, useful for handling dynamic content and bypassing anti-scraping measures.

By carefully considering these factors, you can effectively and responsibly extract valuable data from the web.

7.2 Writing Human-Like Content Articles via LLM Agents for Automated CMS

LLM agents have revolutionized content creation, enabling the generation of high-quality articles at scale. To ensure that these AI-generated articles are human-like and engaging, consider the following:

- **Prompt Engineering:**
 - **Clear and Specific Prompts:** Provide detailed prompts that clearly outline the desired topic, tone, and style.
 - **Iterative Refinement:** Continuously refine prompts based on the quality of the generated content.
 - **Leverage Examples:** Provide examples of well-written articles to guide the LLM's output.
- **Content Quality and Relevance:**
 - **Fact-Checking:** Implement a robust fact-checking process to verify information and avoid inaccuracies.
 - **Topic Coherence:** Ensure that the generated content stays on topic and maintains a logical flow.
 - **Engagement:** Use techniques like storytelling, rhetorical questions, and strong calls to action to captivate readers.
- **SEO Optimization:**
 - **Keyword Research:** Identify relevant keywords and strategically incorporate them into the content.
 - **Meta Tags:** Optimize title tags, meta descriptions, and header tags to improve search engine visibility.
 - **Link Building:** Encourage natural link building by creating high-quality content that attracts backlinks.

By effectively combining LLM agents with human oversight, you can produce a large volume of high-quality, SEO-optimized content that resonates with your target audience.

7.3 Reducing AI-Based Plagiarism from LLM Generated Articles and Enhancing SEO

As AI-generated content becomes increasingly prevalent, it's crucial to address concerns about plagiarism and SEO. Here are some strategies to mitigate these issues:

- **Plagiarism Detection Tools:**
 - AI-Powered Tools: Utilize advanced AI-powered plagiarism detection tools to identify similarities between generated content and existing sources.
 - Manual Review: Conduct thorough manual reviews to ensure originality and authenticity.
- **Content Originality Techniques:**
 - Diverse Prompts: Use a variety of prompts and sources to encourage diverse perspectives and unique content.
 - Human Editing and Refinement: Involve human editors to add personal touches, unique insights, and creative flair.
 - Paraphrasing and Rephrasing: Employ techniques to rephrase sentences and paragraphs without altering the original meaning.
- **SEO Optimization for AI-Generated Content:**
 - Keyword Research and Placement: Conduct thorough keyword research and strategically place keywords throughout the content.
 - Meta Tags and Descriptions: Optimize meta tags and descriptions to improve search engine visibility.
 - Backlink Building: Focus on building high-quality backlinks to enhance domain authority and search rankings.

By implementing these strategies, you can produce AI-generated content that is original, high-quality, and well-optimized for search engines, while mitigating concerns about plagiarism.

8. REFERENCES

1. Leonard Richardson. (n.d.). *Beautiful Soup documentation*. Retrieved from <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
2. python-telegram-bot. (n.d.). *python-telegram-bot documentation (v21.7)*. Retrieved from <https://docs.python-telegram-bot.org/en/v21.7/>
3. Sharma, E. (n.d.). *Text summarisation using ChatGPT API and Python implementation*. LinkedIn. Retrieved from <https://www.linkedin.com/pulse/text-summarisationchatgpt-api-python-implementation-eshan-sharma-v07ke/>
4. Google AI. (n.d.). *Google AI API documentation (Python)*. Retrieved from <https://ai.google.dev/api?lang=python>
5. Streamlit. (n.d.). *Streamlit API reference documentation*. Retrieved from <https://docs.streamlit.io/develop/api-reference>
6. Google AI Studio. (n.d.). *Google AI Studio API key*. Retrieved from <https://aistudio.google.com/apikey>
7. Tiwari, R. (2022, February 8). *Deploying a basic Streamlit app*. Towards Data Science. Retrieved from <https://towardsdatascience.com/deploying-a-basic-streamlit-appceadae286fd0?gi=272c9e398c5b>
8. Tang, A. (2022, October 20). *Streamlit: Deploy your app in 2 hours*. Medium. Retrieved from <https://medium.com/@alfredtangsw/streamlit-deploy-your-app-in-2hours-ef0c1c4909b7>