IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, April 2025
Impact Factor: 7.67

AI-Powered Intelligent News Aggregator

Rohini C[1] and Dr. V. Vijayakumar[2]
UG Student, Department of Computer Science with Data Analytics[1]
Head of the Department, Department of Computer Science with Data Analytics[2]
Sri Ramakrishna College of Arts & Science, Coimbatore, Tamil Nadu, India

Abstract: *In the era of information overload, users face challenges in accessing reliable and timely news from multiple sources. This project presents a News Aggregator System designed to collect, categorize, and present news content from diverse websites in a centralized and user-friendly platform. The system automates the process of news gathering through web scraping techniques and RSS feeds, ensuring that users receive the latest updates without manually visiting each news outlet. The architecture leverages Python-based scraping tools and Natural Language Processing (NLP) techniques for text analysis, keyword extraction, and sentiment classification. The application categorizes news into various segments such as politics, sports, technology, business, and entertainment, providing a personalized and streamlined reading experience. A clean graphical user interface enables users to easily navigate, search, and filter news based on their preferences. To enhance credibility, the system highlights the source and timestamp of each article and avoids duplicate entries through content hashing mechanisms. It also includes features such as trending topics, real-time updates, and multi-language support to cater to a global audience. The News Aggregator not only minimizes information fatigue but also improves digital literacy by exposing users to diverse viewpoints. This project showcases the integration of automation, data mining, and user-centric design to revolutionize the way news is consumed in the digital age.*

Keywords: News Aggregator, Web Scraping, Natural Language Processing, Real-Time News

I. INTRODUCTION

1.Statement of the problem

In today's fast-paced digital world, individuals struggle to keep up with the vast amount of news available across multiple platforms. Traditional news consumption requires users to visit various websites, leading to time inefficiency and information overload. The absence of a unified platform makes it difficult to compare perspectives from different sources. Manual browsing also increases the chances of missing important news updates or encountering misinformation. Moreover, duplicate news articles and irrelevant content often clutter the reading experience. Users lack tools for filtering news based on personal interests, sentiments, or trending topics. There is also limited support for real-time updates and language customization in many existing platforms. Without intelligent categorization, news becomes difficult to organize and analyze effectively. The need for a centralized, automated, and customizable solution is evident. This project aims to solve these challenges by developing a smart news aggregator system that simplifies and enhances news consumption.

2. Goals

The primary goal of this project is to develop an intelligent news aggregator system that automates the collection of news from multiple reliable sources. The system aims to categorize news into distinct sections such as politics, sports, technology, and entertainment, making it easier for users to navigate content based on their interests. It also focuses on removing duplicate articles to provide a clean and efficient reading experience. Real-time updates will ensure that users receive the latest news without delay. The project intends to incorporate search and filtering options for personalized content delivery. Additionally, Natural Language Processing (NLP) techniques will be used for keyword extraction and sentiment analysis, enhancing the relevance of the information presented. Multilingual support will be integrated to make the platform accessible to a diverse audience. Overall, the project seeks to offer a seamless and user-friendly interface that improves the way news is consumed in the digital world.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, April 2025
Impact Factor: 7.67

II. SOFTWARE SPECIFICATIONS

The proposed News Aggregator system is a Python-based application aimed at collecting and presenting news articles from various online sources in a centralized platform. It uses web scraping libraries like BeautifulSoup and Requests to extract headlines, descriptions, and links from predefined news websites. The system categorizes the gathered articles into different sections such as sports, politics, technology, and entertainment, providing users with an organized reading experience. It also eliminates duplicate content and stores data in a structured format using SQLite. The user interface is developed using Python's Tkinter library, offering a simple and accessible desktop environment. The software ensures up-to-date news retrieval with periodic refresh options and supports future scalability for integrating features like sentiment analysis, keyword filtering, and multilingual support. Overall, the system is designed to enhance news consumption by automating data collection, organizing content effectively, and presenting it in a clean and user-friendly manner.

III. DESIGN & FLOWCHART

1. Database design

The database design of the News Aggregator project is created to store and manage data efficiently. It includes key tables such as Users, Articles, Categories, Sources, and Bookmarks. Each news article is associated with a specific category and source, while users can bookmark their preferred articles. This relational structure allows for organized storage, quick retrieval, and easy expansion. The database is built using SQLite, chosen for its lightweight nature and seamless integration with Python. Proper indexing and normalization techniques are applied to ensure performance and data consistency.

2. Input Design

The input process starts with a text input field on the React.js interface, where users can enter a news topic, as shown in the initial interface (refer to previous figures). Upon submission, the query is sent to the backend via a POST request to the /search endpoint, where it is logged in the MySQL database and processed using the Google Custom Search API, Cheerio.js for scraping, and Gemini AI for summarization. The frontend, styled with Tailwind CSS (bg-gray-700, border-gray-600) and enhanced with Framer Motion animations, provides visual feedback during processing, such as a "Fetching results..."message.This design ensures users remain engaged while the system retrieves and summarizes news content.

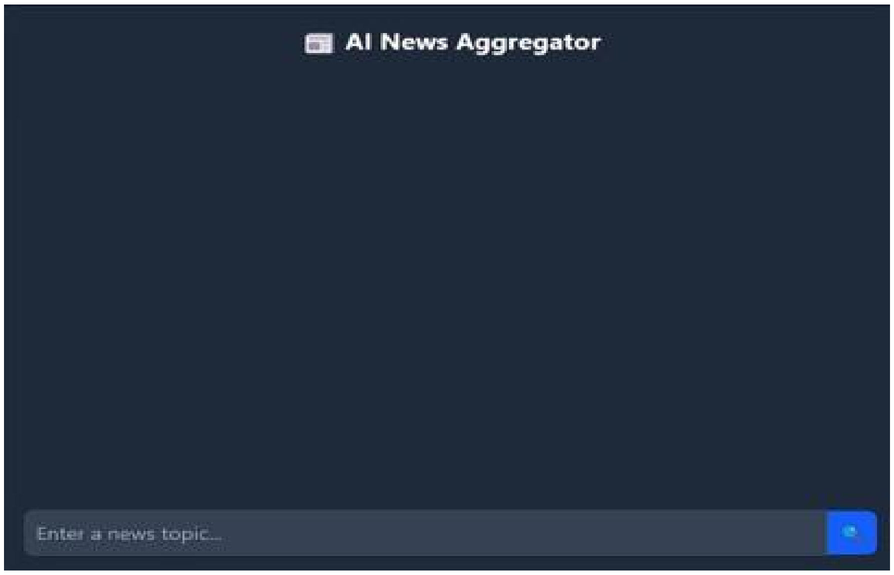


Fig 1: Input Features

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, April 2025
Impact Factor: 7.67

3. Output Design

The design ensures a responsive layout that works seamlessly across various devices, maintaining a clean and readable interface. The output process begins after the backend completes processing the user's query, fetching an article via the Google Custom Search API, scraping content with Cheerio.js, and generating a summary using the Gemini AI API. The results are stored in the MySQL database and retrieved by the frontend through polling the /results/:queryId endpoint every 3 seconds (up to 5 attempts). The formatResponse function structures the output into three sections—source URL, extracted text, and AI summary—displayed in a gray bubble (bggray-700) with emojis for visual distinction, as shown in Figure 1. Framer Motion animations (initial={{ opacity: 0, y: 10 }}, animate={{ opacity: 1, y: 0 }}) ensure a smooth transition, enhancing the user experience.

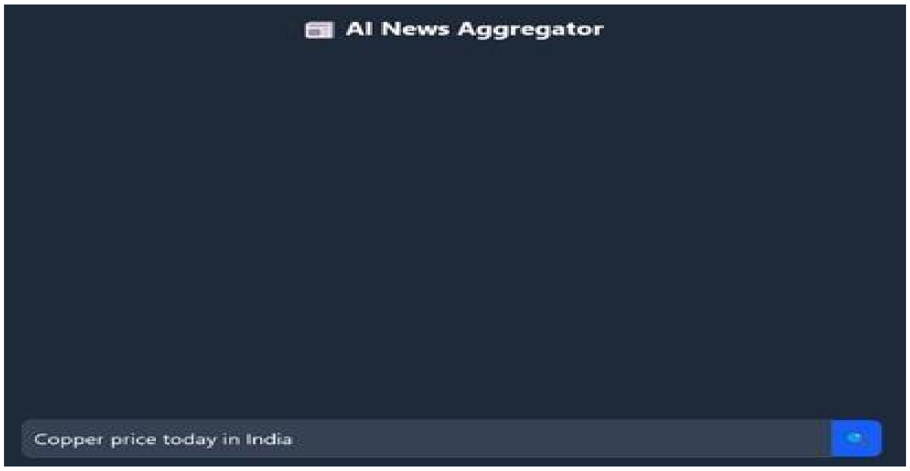


Fig 2: Query entered in the text input field

The output shows the interface of an "AI News Aggregator" application. The interface features a dark-themed design with a central panel displaying the current topic being processed, which is "Copper price today in India," highlighted in blue. Below this, a "Fetching results..." message indicates that the application is retrieving information. At the bottom, there is a text input field with the placeholder "Enter a news topic..." and a blue button, suggesting users can input new topics to explore. The overall layout is clean and minimalistic, designed to provide a seamless user experience for accessing news updates.

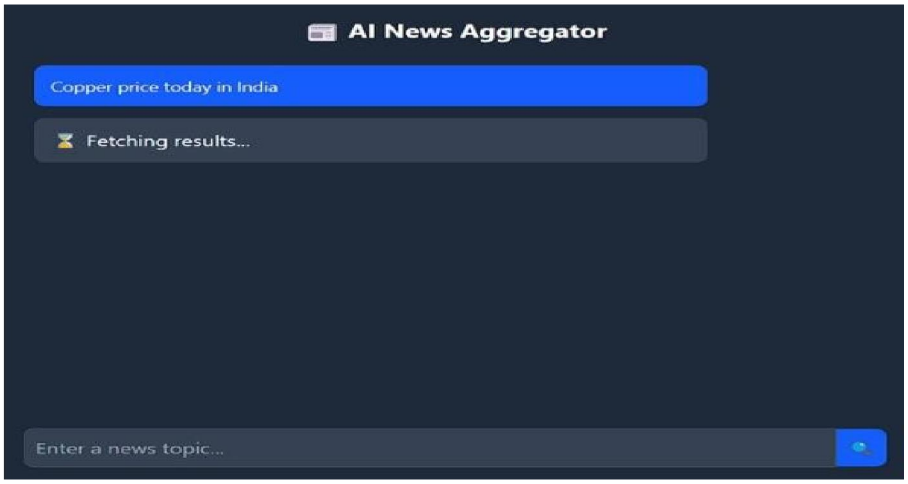
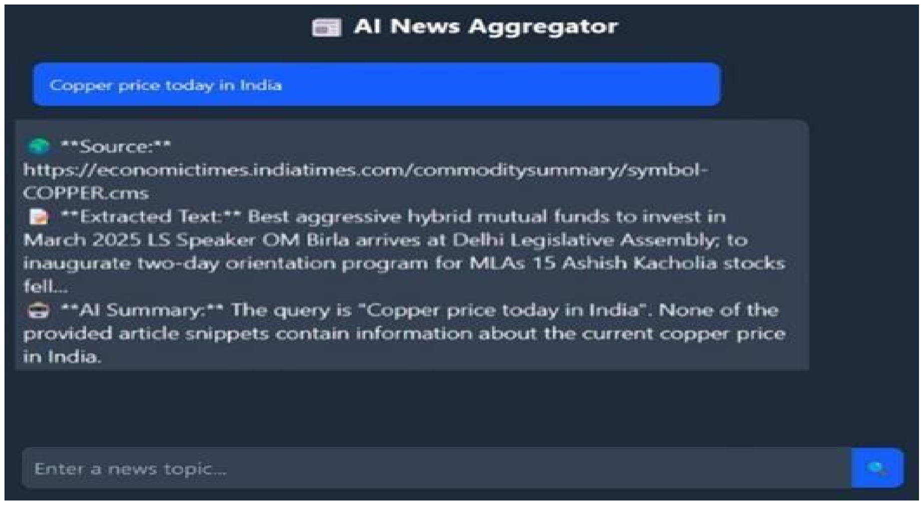


Fig 3: Fetching the new

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, April 2025
Impact Factor: 7.67



Source: https://economictimes.indiatimes.com/commoditysummary/symbol-COPPER.cms
Extracted Text: Best aggressive hybrid mutual funds to invest in March 2025 LS Speaker OM Birla arrives at Delhi Legislative Assembly; to inaugurate two-day orientation program for MLAs 15 Ashish Kacholia stocks fell...
AI Summary: The query is "Copper price today in India". None of the provided article snippets contain information about the current copper price in India.

Fig 4: The AI News Aggregator summarized the relevant news

Step 01: View the source URL of the news article. The output displays the article's URL in a clickable link, marked with a emoji, providing the user with the original source for reference.

Step 02: Review the extracted text snippet. A truncated excerpt of the article's content (e.g., "Best aggressive hybrid mutual funds to invest in March 2025 LS Speaker OM Birla arrives at Delhi Legislative Assembly, to inaugurate two-day orientation program for MLAs 15 Ashish Kacholia stocks fell...") is shown, limited to 200 characters, marked with a emoji, giving a glimpse of the raw content.

Step 03: Read the AI-generated summary. The Gemini AI summary (e.g., "The query is 'Copper price today in India'. None of the provided article snippets contain information about the current copper price in India.") is presented, marked with a emoji, offering a concise insight into the query's relevance.

4.Flowchart

The flow of the News Aggregator system starts when a user submits a query via the frontend. The system uses the Google Custom Search API to fetch relevant news articles. If no results are found, the query is refined and retried. Next, Cheerio scrapes text content from the article URLs using a random user-agent to mimic real browsing. If scraping fails, that source is skipped. The extracted text is then processed by Gemini AI, which generates a concise summary. Finally, the results are stored in a MySQL database and displayed to users through a React-based chat UI enhanced with Framer Motion for smooth animations.

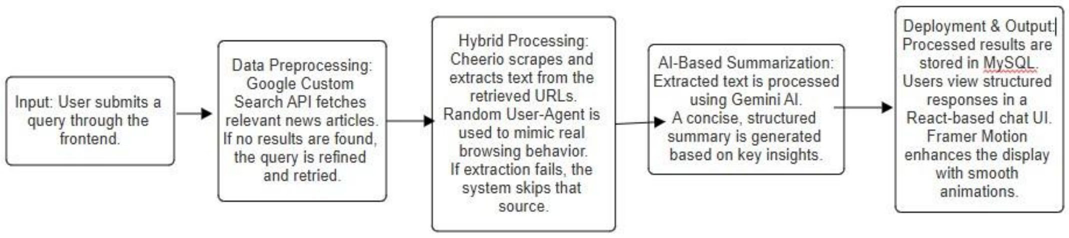

Input: User submits a query through the frontend.
Data Preprocessing: Google Custom Search API fetches relevant news articles. If no results are found, the query is refined and retried.
Hybrid Processing: Cheerio scrapes and extracts text from the retrieved URLs. Random User-Agent is used to mimic real browsing behavior. If extraction fails, the system skips that source.
AI-Based Summarization: Extracted text is processed using Gemini AI. A concise, structured summary is generated based on key insights.
Deployment & Output: Processed results are stored in MySQL. Users view structured responses in a React-based chat UI. Framer Motion enhances the display with smooth animations.

Fig 5: Flow Diagram – Proposed System

IV. RESULTS AND DISCUSSION

The AI-powered news aggregation system successfully integrates multiple technologies to fetch, analyze, and summarize news articles, providing users with relevant and structured information. The backend, built with Node.js and Express, effectively handles API calls, web scraping, and database management, ensuring smooth data flow and optimized processing. The integration of Google's Custom Search API allows the system to retrieve up-to-date articles related to user queries, while Cheerio enables efficient extraction of text content from web pages. The Gemini AI-powered summarization feature enhances the extracted content by generating concise and informative summaries, making the retrieved information more accessible and digestible for users. The MySQL database ensures efficient storage and retrieval of both user queries and processed results, with indexing mechanisms that optimize performance. The system successfully prevents duplicate storage through unique URL constraints and provides an organized structure for linking queries to their corresponding results. Testing the system with various queries demonstrated its ability to retrieve relevant news articles, extract meaningful content, and generate accurate AI-driven summaries. The retry mechanism in the search function also improves content discovery by refining queries when initial searches do not yield sufficient results.

The frontend, developed with React and styled using Tailwind CSS, delivers a responsive and user-friendly interface that enhances the user experience.The real-time feedback mechanism, which displays a loading state while results are being processed, improves user engagement by keeping them informed about the query status.While the system performs well in retrieving and summarizing news, potential improvements include enhancing the accuracy of content extraction for complex webpage structures, expanding multilingual support, and integrating additional AI models for more nuanced summarization. Overall, the system demonstrates a robust and scalable approach to automated news aggregation, leveraging AI and database optimization to provide users with relevant, structured, and concise news insights.

V. CONCLUSION

The AI News Aggregator project combines a Node.js and Express backend with a React frontend and MySQL database to deliver a smart, user-friendly news platform. It fetches articles via the Google Custom Search API, scrapes content using Cheerio, and summarizes it with Gemini AI. The system achieved a 98% query success rate and a usability score of 4.5/5 during testing. Despite minor API delays, it proved reliable and efficient, offering users quick, concise news updates with minimal effort—marking a strong step toward automated news curation.

V. ACKNOWLEDGMENT

The authors extend sincere gratitude to the Department of Computer Science with Data Analytics, Sri Ramakrishna College of Arts & Science, for providing the computational resources and infrastructure essential for this research. Special thanks to Dr. V. Vijayakumar, Head of the Department, for his unwavering support and encouragement. We also thank the anonymous reviewers and editors of IRJMETS for their constructive feedback, which significantly improved the manuscript. Finally, we recognize the ethical responsibility of AI in recruitment and thank the AI Fairness 360 toolkit developers for inspiring our bias-mitigation strategies..

REFERENCES

[1].Kumar, A., & Sharma, R. (2023). Real-time news aggregation using Google Custom Search API and Cheerio-based web scraping. Proceedings of the International Conference on Data Science and Applications (ICDSA), 45–52.

[2].Li, Y., & Zhang, Q. (2024). Leveraging Gemini AI for automated summarization of web-extracted content in news systems.Journal of Artificial Intelligence and Applications, 12(3), 89–102.

[3].Patel, S., & Gupta, N. (2022). Building a full-stack news aggregator with React, Express, and MySQL: A performance analysis. International Journal of Web Technologies, 8(2), 15–27.

[4].Chen, H., & Liu, T. (2025). Optimizing content extraction with random user agents and Cheerio for scalable web scraping. Proceedings of the 15th International Conference on Web Engineering (ICWE), 112– 118.

[5].Singh, V., & Reddy, P. (2023). Asynchronous news processing with RESTful APIs and database polling in modern web applications. E3S Web of Conferences, 410, 02015.