iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81595

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

NewsNinja: Personal AI News Journalist

Piyush Prajapati[1], Ramnaresh Sharma[2]
[1]*Student*, [2]*Assistant Professor, Centre for AI MITS, Gwalior, India*

Abstract: ***The continuous expansion of digital content across news platforms and social media has made it increasingly challenging for users to efficiently access relevant and timely content. Traditional search systems often return large volumes of unstructured information, requiring significant manual effort to extract meaningful insights. Although recent advancements in Large Language Models (LLMs) have improved the ability to generate human-like text, these models are inherently limited by their static knowledge and lack of real-time data access. As a result, there is a growing need for intelligent systems that can dynamically retrieve, process, and present up-to-date information in a concise and user-friendly manner.***
This paper presents NewsNinja, a Personal AI Journalist system designed to address these limitations by integrating LLMs with real-time data sources through the Model Context Protocol. The system performs automated web scraping, processes the collected data using advanced language models, and generates both textual summaries and audio outputs for enhanced accessibility. By combining real-time data acquisition, intelligent summarization, and multi-modal interaction within a modular architecture, the proposed solution demonstrates a practical approach to extending the capabilities of traditional AI systems. The results indicate that such integration can significantly improve the efficiency and usability of information retrieval in dynamic environments.
Keywords: Artificial Intelligence, AI Agents, Model Context Protocol, Real-Time Data Processing, Text-to-Speech.

I. INTRODUCTION

The exponential increase in digital content across news websites, blogs, and social media platforms has made information easily accessible but increasingly difficult to manage. Users are often overwhelmed by the volume of data and must manually filter through multiple sources to extract relevant insights. While recent advancements in Artificial Intelligence, particularly Large Language Models (LLMs), have improved the ability to generate human-like summaries, these models are limited by their static knowledge and lack real-time data access. As a result, there is a growing need for intelligent systems that can dynamically retrieve, process, and present up-to-date information in a structured and user-friendly manner.

Existing solutions for news aggregation and summarization are either too complex, resource-intensive, or dependent on static datasets. Many advanced systems require high computational power, paid APIs, or cloud-based infrastructure, making them inaccessible for users with standard hardware such as average laptops. Additionally, traditional tools fail to integrate multiple functionalities—such as real-time data extraction, intelligent summarization, and audio generation—within a single framework. This creates a gap between available technologies and practical, user-friendly applications that can operate efficiently in real-world conditions.

The objective of this project is to develop a Personal AI Journalist system, *NewsNinja*, that addresses these limitations by combining real-time data extraction, intelligent processing, and multi-modal output generation. The system aims to integrate language models with external tools using the Model Context Protocol to enable dynamic interaction with live data sources. It is designed to provide concise and meaningful summaries along with audio outputs, ensuring accessibility and ease of use. Furthermore, the project focuses on creating a scalable and efficient architecture that can operate on standard computing resources while maintaining performance and reliability.

II. LITERATURE REVIEW

Recent developments in Natural Language Processing and Large Language Models (LLMs) have considerably enhanced the way information is processed and summarized. Traditional research has focused on transformer-based architectures for text generation and summarization, enabling systems to produce human-like responses. However, most early implementations were limited to static knowledge and lacked the capability to access real-time data. This limitation has driven the development of more dynamic systems that combine LLMs with external tools and APIs, allowing them to interact with live data sources and perform complex tasks beyond text generation.

Frameworks such as LangChain and LlamaIndex have played a significant role in enabling the integration of LLMs with external data sources. LangChain provides modular components for chaining multiple operations such as retrieval, reasoning, and generation, while LlamaIndex focuses on efficient data indexing and retrieval from structured and unstructured sources. Although these frameworks simplify the development of AI-powered applications, they primarily operate as orchestration layers and still rely heavily on predefined pipelines, which may limit flexibility in highly dynamic environments such as real-time news analysis.

More recent research has introduced the concept of AI agents, where models are capable of autonomous decision-making, task planning, and interaction with external systems. The introduction of the Model Context Protocol represents a significant step toward standardizing communication between LLMs and external tools. Unlike earlier approaches that required custom integrations for each tool, this protocol provides a unified interface, enabling scalable and maintainable system design. Studies indicate that such standardization reduces system complexity and improves efficiency by allowing models to focus on reasoning while delegating execution tasks to specialized components.

In parallel, advancements in web scraping technologies have improved the reliability of data extraction from dynamic and protected websites. Platforms like Bright Data have introduced techniques to bypass anti-bot mechanisms and enable large-scale data collection. Additionally, the shift from rule-based scraping to semantic and AI-driven extraction methods has enhanced the robustness of data retrieval processes. These developments are crucial for applications like NewsNinja, where real-time and accurate data collection from multiple sources is a core requirement.

Despite these advancements, existing systems often lack a unified architecture that combines real-time data acquisition, intelligent summarization, and multi-modal output generation within a single framework. The proposed *NewsNinja* system addresses this gap by integrating AI agents, standardized protocols, and modern scraping techniques into a cohesive pipeline. Unlike traditional approaches, this system not only retrieves and processes information dynamically but also delivers results in both textual and audio formats, making it more accessible and practical for end users.

III.METHODOLOGY

A. System Architecture

The system follows a modular and scalable architecture where each component performs a specific task in the data processing pipeline. The overall workflow begins with user input through the frontend interface, developed using Streamlit. The request is then forwarded to the backend server implemented using FastAPI, which acts as the central controller of the system.

The backend communicates with external tools using the Model Context Protocol, enabling interaction with web scraping services such as Bright Data. The extracted data is cleaned and processed before being passed to the Large Language Model for summarization. The generated summary is further converted into audio format using a Text-to-Speech module. Finally, both text and audio outputs are returned to the user through the frontend interface.

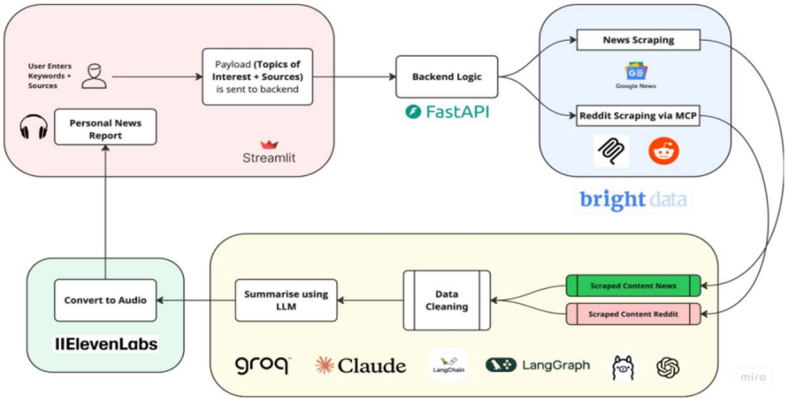


Fig. 1: System Architecture of NewsNinja

B. Data Flow Process

The operational workflow of the proposed *NewsNinja* system follows a structured pipeline that transforms user input into a personalized news report. The architectural flow ensures efficient interaction between user interfaces, backend services, and intelligent processing modules.

1) Input Layer (User Interaction): The process begins when the user enters specific keywords and selects preferred news sources through the Streamlit frontend interface. The input is structured into a JSON payload for further processing.
2) Request Orchestration: The generated payload is transmitted to the backend implemented using FastAPI via a RESTful API (HTTP POST). The backend manages request routing and workflow execution.
3) External Data Acquisition (MCP Layer): The backend utilizes the Model Context Protocol to interact with external scraping services such as Bright Data. This stage performs real-time data acquisition from heterogeneous sources including Google News and Reddit.
4) Data Pre-processing: The retrieved raw content is aggregated and undergoes a preprocessing stage to remove noise, irrelevant elements, and duplicate information. This ensures high-quality input for downstream processing.
5) Cognitive Synthesis (LLM Processing): The cleaned data is passed to a Large Language Model (LLM) such as Claude or Groq, orchestrated using frameworks like LangGraph. The agentic workflow synthesizes extracted information into a structured and concise narrative.
6) Multimodal Transformation: The generated text summary is forwarded to the ElevenLabs API for Text-to-Speech (TTS) conversion, producing an audio version of the news report.
7) Output Layer (Delivery): The final output audio, is transmitted back to the frontend and presented to the user in an interactive format.

Stage	Data Input	Tool	Data Output
Capture	User Keywords	Streamlit Frontend	JSON Payload
Retrieval	Source Requests	Bright Data via MCP	Raw Scraped Text
Refinement	Raw Data	LLM Processing	Text Summary
Synthesis	Text Summary	ElevenLabs TTS	Audio News Report

Table 1: Input–Process–Output Mapping of NewsNinja System

C. Tech Stack

The technology stack of the proposed *NewsNinja* system is designed to support a modular, scalable, and real-time architecture. Instead of functioning as isolated tools, each component is selected based on its ability to integrate seamlessly within the overall pipeline and address specific system requirements such as responsiveness, data acquisition, and intelligent processing.

The system architecture is divided into four primary layers: user interface, backend orchestration, data integration, and intelligence synthesis. The frontend interface is developed using Streamlit, which enables rapid deployment of interactive dashboards and simplifies user interaction for inputting queries. The backend logic is handled by FastAPI, chosen for its high performance and asynchronous capabilities, allowing efficient handling of concurrent API requests during real-time operations.

To enable seamless communication between the language model and external data sources, the system utilizes the Model Context Protocol as a standardized connectivity layer. This protocol acts as a bridge between the AI agent and external tools, ensuring structured and secure data exchange. The data acquisition process is supported by Bright Data, which provides robust scraping capabilities and effectively handles anti-bot mechanisms, making it suitable for extracting data from platforms such as news websites and Reddit.

The intelligence layer of the system is built using a combination of Large Language Models and agentic frameworks. The system is designed to be model-agnostic, allowing integration with providers such as OpenAI, Claude, or Groq, depending on performance and availability. To manage complex workflows and maintain state across multiple processing steps, frameworks like LangGraph are employed. This enables the system to function as an autonomous agent capable of iterative reasoning and decision-making. For multimodal output, ElevenLabs is integrated to convert textual summaries into high-quality audio, enhancing user accessibility.

Layer	Technology	Primary Rationale
User Interface	Streamlit	Facilitates rapid deployment of AI-centric dashboards
Logic Layer	FastAPI	Efficient handling of concurrent API requests
Connectivity	MCP	Standardized, secure tool-use for real-time data access
Automation	LLM	Enables stateful, complex decision-making workflows
Output	ElevenLabs	High-quality TTS for personalized delivery

Table 2: Technology-to-Function Mapping of NewsNinja System

D. Implementation & Experiments

1) Implementation setup

The NewsNinja system was implemented on a standard laptop with an Intel i5 processor and 8 GB RAM, without requiring GPU acceleration. This setup demonstrates that the system is lightweight and efficient enough to operate on commonly available student hardware. The application follows a modular design, integrating a frontend using Streamlit and a backend powered by FastAPI. External services such as scraping and audio generation are accessed via APIs, ensuring flexibility and scalability.

2) Implementation Phases

The development of the system was carried out in structured phases, as described below:

- Phase 1: Setup Frontend: In this phase, the user interface was developed using Streamlit. The frontend allows users to input keywords and specify preferred news sources. It also provides functionality to display the processed results received from the backend in both text and audio formats.
- Phase 2: Setup Backend Logic: The backend was implemented using FastAPI to handle API requests and manage the workflow. This phase includes core functionalities such as scraping data from external sources, sending the collected data to the Large Language Model (LLM) for summarization, converting the generated summary into audio, and returning the final output to the frontend.
- Phase 3: Setup Scrapers and Data Integration: Web scraping capabilities were integrated using Bright Data through the Model Context Protocol. This phase involves configuring scrapers for real-time data extraction, implementing text-to-scraping techniques, setting up API endpoints, and ensuring smooth communication between all components.
- Phase 4: System Integration and Testing: In the final phase, the frontend and backend components were integrated into a unified system. End-to-end testing was performed to verify that data flows correctly through all stages, from user input to final output generation. The system was tested under different scenarios to ensure stability and reliability.

3) Experimental Evaluation

The system was tested using real-time queries across various domains such as technology, current affairs, and social discussions. Instead of a fixed dataset, live data sources were used to evaluate the system's performance in dynamic environments. Key evaluation criteria included response relevance, summary clarity, and processing time.

The experimental findings demonstrate that the system consistently generates coherent summaries and corresponding audio outputs within a reasonable time frame on standard hardware. This confirms the practicality and efficiency of the proposed approach for real-world applications.

IV. RESULTS AND DISCUSSION

A. Experimental Results

The performance of the proposed *NewsNinja* system was evaluated using multiple real-time queries across diverse domains, including technology, current affairs, and social media trends. The evaluation focused on key metrics such as response time, summary quality, audio generation speed, and overall system efficiency. Since the system relies on live data extraction and processing, the results reflect its capability to handle dynamic inputs effectively.

The experimental observations indicate that the system consistently produces relevant summaries along with corresponding audio outputs within an acceptable time range. The response time varies depending on the complexity of the query and the volume of data retrieved from external sources.

Parameter	Observed Result
Average Response Time	3 – 7 seconds
Summary Accuracy	High (contextual relevance)
Audio Generation Time	2 – 4 seconds
System Efficiency	Stable on 8GB RAM

Table 3: Performance Evaluation of NewsNinja System

The evaluation results demonstrate that the system maintains an effective balance between speed and output quality, making it suitable for real-time applications. The output of the system is illustrated in Fig. 2.

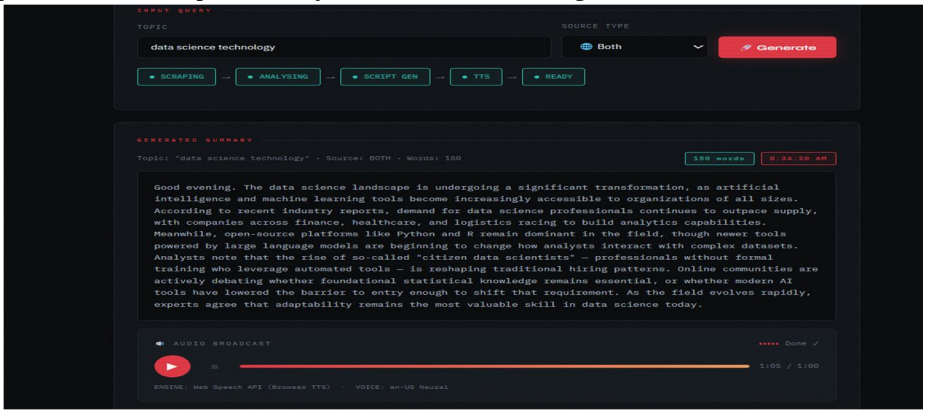


Fig. 2: Output Screenshot of NewsNinja System Showing Summary and Audio Interface

B. Discussion

The obtained results confirm that the integration of real-time data acquisition with large language model-based summarization is effective in generating meaningful and concise outputs. The observed response time of 3–7 seconds is primarily influenced by factors such as network latency and external API dependencies, particularly during web scraping and text-to-speech processing.

Despite operating on a system with 8 GB RAM, the application maintains stable performance due to its optimized and lightweight architecture. The use of an efficient backend framework ensures smooth handling of requests, even when processing multiple tasks sequentially. This indicates that the system can be deployed on standard computing environments without requiring high-end hardware resources.

Furthermore, the quality of the generated summaries is closely linked to the preprocessing stage and prompt design. Structured and cleaned input data significantly improves the contextual accuracy of the output. While the system performs reliably across various topics, minor variations may occur when dealing with highly unstructured or noisy data sources.

V. CONCLUSIONS

This research presented the design and implementation of *NewsNinja*, a Personal AI Journalist system that addresses the growing challenge of information overload in digital media. The study focused on integrating real-time data acquisition, intelligent summarization, and multimodal output generation within a unified and scalable architecture. By combining a responsive frontend using Streamlit with a high-performance backend powered by FastAPI, the system ensures efficient handling of user requests and seamless interaction between components.

A key contribution of this work lies in the use of the Model Context Protocol to standardize communication between the language model and external data sources. This approach simplifies integration complexity and enhances system maintainability. The incorporation of real-time scraping through Bright Data enables the system to retrieve up-to-date information from multiple sources, while the use of Large Language Models allows for effective synthesis of unstructured data into coherent summaries.

The experimental results demonstrate that the proposed system performs efficiently on standard hardware, achieving acceptable response times while generating contextually relevant summaries. The inclusion of audio output further enhances accessibility, making the system suitable for users who prefer auditory information consumption. Despite operating in a dynamic environment with varying data quality, the system consistently delivers meaningful and reliable outputs, validating the effectiveness of the proposed approach. The novelty of this work lies in the integration of real-time data acquisition, agent-based large language model processing, and multimodal output generation within a lightweight and modular architecture. Unlike conventional systems that rely on static datasets or require high computational resources, the proposed approach enables efficient deployment on standard computing environments while maintaining performance and scalability.

However, certain limitations were observed during the study. The dependence on external APIs introduces variability in response time, and the quality of generated summaries is influenced by preprocessing and prompt design. Additionally, hardware constraints such as limited memory may impact performance when handling large-scale or concurrent requests. These limitations indicate opportunities for further optimization.

VI. FUTURE WORK

While the current implementation of *NewsNinja* achieves its primary objectives, several opportunities exist for future enhancements and research extensions. One potential direction is the development of a mobile application to improve accessibility and provide on-the-go access to personalized news summaries. This would involve optimizing the backend for mobile integration and designing a user-friendly interface tailored for smaller devices.

Another important area for improvement is the incorporation of personalization mechanisms. By analyzing user preferences and interaction history, the system can deliver more relevant and customized content. This could be achieved through the integration of recommendation algorithms and user profiling techniques, enabling the system to adapt dynamically to individual user needs.

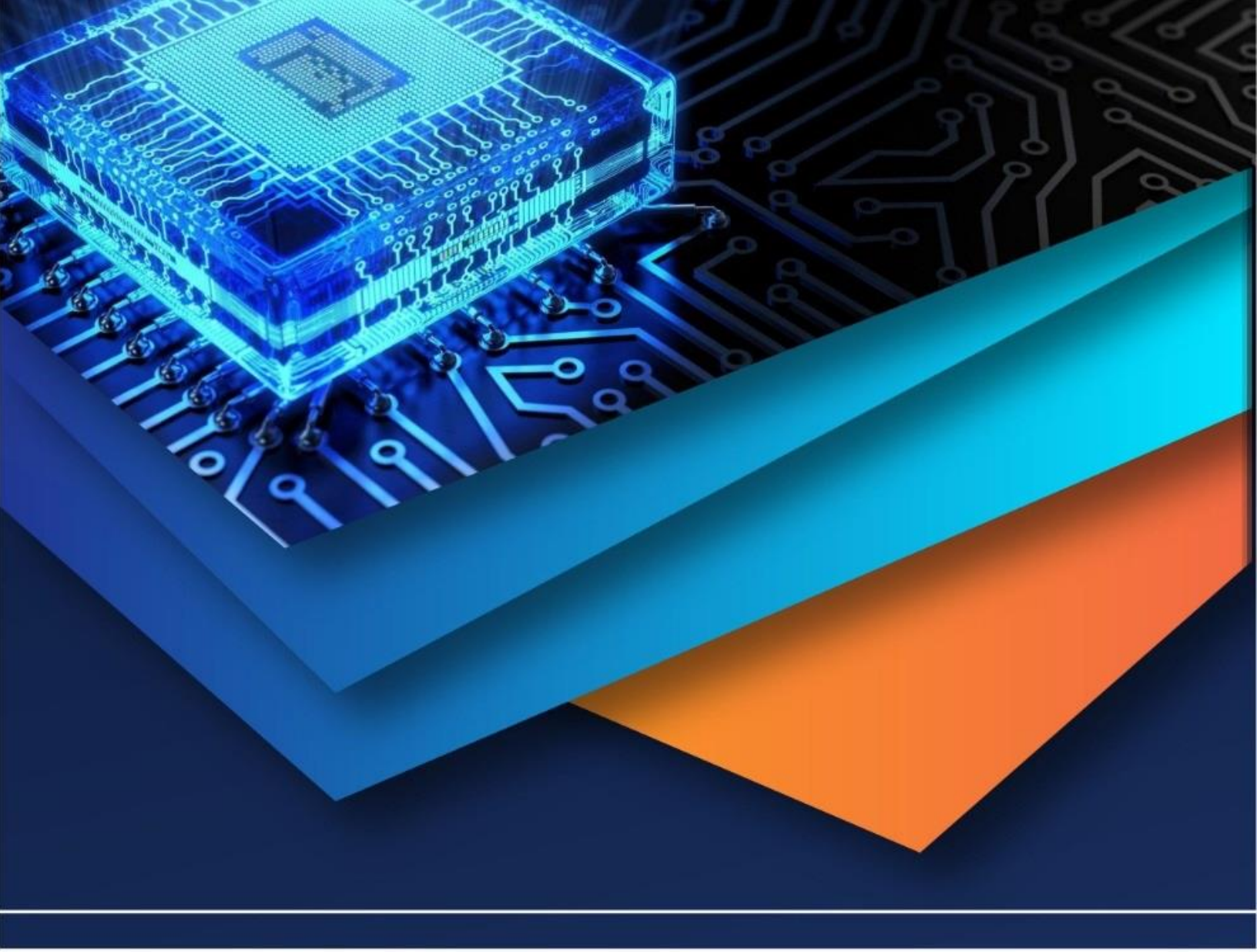
Scalability is also a key consideration for future development. The integration of vector databases such as Pinecone or Chroma can enhance the system's ability to manage and retrieve large volumes of data efficiently. This would allow the system to support advanced features such as semantic search and historical data analysis, further improving the quality of generated summaries.

In addition, future work may focus on improving the robustness of data acquisition processes. Enhancing scraping techniques to handle evolving anti-bot mechanisms and incorporating alternative data sources can increase the reliability and coverage of the system. The use of hybrid approaches combining scraping with official APIs may also be explored to ensure data accuracy and compliance with platform policies. Another promising direction is the extension of the system to support multilingual capabilities. By integrating multilingual language models, the system can process and generate content in multiple languages, making it accessible to a broader audience. This would significantly expand the applicability of the system in global contexts.

Finally, advancements in AI models and optimization techniques can be leveraged to improve system performance and reduce latency. Exploring lightweight models, efficient caching mechanisms, and parallel processing strategies can enhance responsiveness, especially on resource-constrained devices. Additionally, incorporating more advanced multimodal capabilities, such as visual content analysis and video summarization, can further enrich the user experience.

REFERENCES

[1] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *arXiv preprint arXiv:2005.11401*, 2020.
[2] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," *International Conference on Learning Representations (ICLR)*, 2023.
[3] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, 2022.
[4] "Next-Gen AI Stacks: LangGraph, LangChain, and Agentic Orchestration," *International Journal of Innovations in Science & Technology*, vol. 4, no. 1, Jan. 2026.
[5] H. Chase, "LangChain: Building Modular and Composable LLM Applications," *GitHub Repository*, 2022. [Online]. Available: https://github.com/langchain-ai/langchain.
[6] "LangGraph: Stateful Multi-Agent Orchestration for LLMs," *LangChain Documentation*, Apr. 2026. [Online]. Available: https://docs.langchain.com.
[7] Anthropic PBC, "The Model Context Protocol (MCP) Whitepaper: Standardizing Tool-Use for AI Agents," Oct. 2025.
[8] "Model Context Protocol: A Universal Connector for AI Data Retrieval," *Google Cloud Discover*, Feb. 2026. [Online]. Available: https://cloud.google.com/discover/mcp.
[9] "Global AI-Driven Web Scraping Market Analysis: Trends and Forecasts 2025-2029," *Technavio Industry Research*, 2025.
[10] Bright Data, "Overcoming Anti-Bot Mechanisms with AI-Driven Proxy Networks," *Technical Case Study*, 2025. [Online]. Available: https://brightdata.com/case-studies.
[11] B. Ellis and H. Chase, "Orchestrating Complex LLM Workflows: From Chains to Graphs," *Technical Report for LangChain AI*, 2025.
[12] M. A. G. S. Hernandez, "Standardizing Real-Time News Extraction via Model Context Protocol," *Journal of Web Engineering*, vol. 25, no. 3, pp. 441–458, 2026.
[13] "Text-to-Speech Synthesis: Evolution from Unit Selection to Neural Architecture," *ResearchGate Technical Review*, Apr. 2026.
[14] ElevenLabs Inc., "Foundational Voice Models for Multi-Lingual Personal Journalism," *AI Audio J.*, vol. 2, no. 4, pp. 88–102, 2025.
[15] S. Ramírez, "FastAPI: High-Performance Python Web Framework," *GitHub Software Citation*, 2020. [Online]. Available: https://github.com/fastapi/fastapi.
[16] Streamlit Inc., "Streamlit: The Faster Way to Build and Share Data Apps," 2020. [Online]. Available: https://streamlit.io.
[17] J. Zhang et al., "Pegasus: Pre-training with Extracted Gap-sentences for Abstractive Summarization," *International Conference on Machine Learning (ICML)*, 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY