Smart Content Summarizer

M.KASTHURI #1, KARUNYA S #2, LAVANYA SREE V #3, MARIA SNOWBE S #4,
#1 Assistant Professor, Adhiyamaan College of Engineering(An Autonomous Institution), Hosur
#2,3,4 UG Students, Adhiyamaan College of Engineering(An Autonomous Institution), Hosur

Abstract: The Smart Content Summarizer (SCS) is an intelligent web-based system designed to automatically extract key information from lengthy content and generate concise, meaningful summaries. In the modern digital era, users often face information overload from online articles, research papers, and documents. To address this problem, SCS provides a multi-input summarization platform that supports direct text input, URL-based content extraction, and PDF file summarization. The system processes the input content, identifies important sentences, removes redundant information, and produces a structured summary that preserves the original context. The application includes features to save, view, update, and delete generated summaries, allowing users to maintain an organized personal knowledge library. With a simple, clean, and responsive user interface, SCS ensures smooth interaction and enhances user productivity. Technically, the system is developed using HTML, CSS, and JavaScript for the frontend and Node.js with Express.js for backend processing. MongoDB is used to store user summaries securely, and content extraction logic is implemented to handle text parsing from web pages and PDF documents. The project demonstrates real-time content processing, efficient data handling, and scalable full-stack development practices. The Smart Content Summarizer serves as a practical productivity tool for students, researchers, and professionals, enabling faster understanding of large-scale content and supporting efficient learning and decision-making in the digital age

I. INTRODUCTION

In the modern digital era, information is growing at an unprecedented rate across multiple online and offline platforms. Students, researchers, and professionals encounter large volumes of text daily through academic articles, digital study materials, web pages, technical documents, and PDF files. While this availability of information is an advantage, processing it manually becomes highly time-consuming and mentally demanding. The need for fast, accurate, and meaningful content extraction has therefore become extremely important. The Smart Content Summarizer (SCS) system is developed as a solution to overcome this increasing information-overload challenge. SCS is a web-based tool designed to automatically extract and convert lengthy content into short, meaningful summaries without losing context or important ideas. Unlike many simple text summarizers, our system supports three input modes: direct text input, web URL extraction, and PDF file upload. This multi-input capability enables users to summarize information from diverse sources, making it highly useful in real-world academic and professional environments. The system uses text-processing logic to analyze the content, identify the most relevant sentences, and produce a clear summary. It is designed with a clean, user-friendly interface that allows users to easily interact with the system. To enhance usability, the platform also includes a summary-management module where users can save, view, edit, and delete previously generated summaries. This helps in building a personal knowledge library for future reference. Technically, the system is built using modern full-stack technologies, ensuring real-time processing, smooth performance, and secure data management. The frontend is developed using HTML, CSS, and JavaScript, while the backend uses Node.js and Express.js to manage the logic and API processing. The database is implemented using MongoDB to store user summaries efficiently along with metadata such as timestamp and input type. The Smart Content Summarizer is not only a mini-project but also a practical productivity tool that reflects real-time problem solving, software design principles, and modern web development practices. By reducing reading time and helping users focus on key ideas, the system contributes to efficient learning, quick decision-making, and enhanced digital productivity. This project demonstrates both technical knowledge and user-centric design, making it a valuable academic and real-world solution.

II. LITERATURE SURVEY

The rapid growth of digital content across web platforms, documents, and research databases has created a strong demand for automated summarization tools. Text summarization, a key branch of Natural Language Processing (NLP), focuses on condensing lengthy information into concise and meaningful summaries without losing essential context. Early studies in extractive summarization, such as Luhn (1958) and Edmundson (1969), focused on selecting significant sentences based on word frequency and position. With the evolution of machine learning, statistical and graph-based models like TextRank improved summary coherence by considering sentence connectivity and relevance.

Recent advancements in deep learning and transformer-based architectures have revolutionized text summarization. Models such as Google's BERT, OpenAI's GPT, and Mistral 7B enable abstractive summarization, generating human-like paraphrased summaries rather than direct sentence extraction. Research has also emphasized multi-document and cross-domain summarization, addressing challenges like redundancy, context retention, and semantic accuracy. Tools like OpenAI's ChatGPT and Hugging Face APIs have further simplified integration of large-scale language models into real-time applications.

However, most existing systems focus primarily on text inputs, while real-world users often require summarization of web articles (URLs) and PDF documents. The proposed system, Smart Content Summarizer, bridges this gap by combining content extraction techniques such as Cheerio and Extractus for web data, pdf-parse for document processing, and OpenRouter AI for generating high-quality summaries. This integration enables users to summarize diverse content formats in a unified web platform. The literature highlights that hybrid approaches—combining extraction, AI inference, and human-readable design—enhance both usability and performance. Thus, Smart Content Summarizer contributes to advancing accessible, intelligent summarization systems suitable for students, researchers, and professionals in the digital age.

III. PROPOSED SYSTEM

The Smart Content Summarizer (SCS) system has been designed to overcome the drawbacks of existing manual and limited-function summarization tools. The proposed system provides an automated, efficient, and user-friendly web-based platform that summarizes both direct text and online content through URLs. It focuses on simplifying the process of information extraction while maintaining accuracy, readability, and data control for the user. This system uses a structured architecture where the frontend handles user interaction, the backend manages processing and summarization, and the database securely stores user-generated summaries. The design emphasizes speed, simplicity, and reusability, ensuring that users can easily access and manage their summarized information. Key Features of the Proposed System:

1. Automated Summarization: Generates concise summaries instantly from either plain text or URL-based content using a backend summarization logic.

2. Dual Input Option: Accepts both direct text input and valid URLs, allowing flexibility in summarizing various types of content.

3. Responsive Interface: Developed using HTML, CSS, and JavaScript to ensure a clean, fast, and mobile-friendly user experience.

4. CRUD Functionality: Users can create, view, update, and delete their saved summaries stored in the MongoDB database.

5. Backend Efficiency: Built using Node.js and Express.js for faster request handling, validation, and connection to the summarization engine.

6. Data Storage and Management: MongoDB stores summaries securely with timestamps and unique identifiers, ensuring easy retrieval and management.

7. Error Handling and Validation: Input validation and meaningful alert messages help prevent errors and improve system reliability.

8. Real-time Communication: API integration allows real-time interaction between frontend and backend without page reloads.

9. Scalability and Extensibility: The modular structure allows future improvements like AI-based summarization or user authentication. Advantages Over Existing System:

- Reduces manual reading time and effort.
- Produces more accurate, readable, and context-aware summaries.
- Provides full CRUD functionality for summary management.
- Offers better data security and accessibility.
- Responsive, modern, and easy-to-use interface.

The proposed system thus provides a complete and reliable summarization platform that is fast, secure, and user-centric, making it a valuable solution for efficiently managing and understanding digital information.

IV. PROPOSED SOLUTION

Smart Content Summarizer, aims to develop an intelligent and user-friendly web application capable of summarizing content from multiple sources — including plain text, URLs, and PDF documents. The system addresses the growing need for quick, accurate, and accessible summarization in an era where users are constantly exposed to large volumes of information. Existing summarization tools are often limited to single formats or rely on basic extraction methods that fail to capture semantic meaning. This project proposes a hybrid solution integrating advanced Natural Language Processing (NLP) and deep learning models to generate meaningful summaries efficiently.

The web tool is built using HTML, CSS, and JavaScript for the front-end interface, ensuring simplicity and responsiveness. The backend is powered by Node.js and Express.js, which manage user requests, handle input data, and interact with the summarization engine. When users enter text, a URL, or upload a PDF, the system intelligently identifies the input type. For web URLs, content extraction is handled through libraries such as Cheerio and Extractus, while pdf-parse efficiently processes PDF content.

The extracted or input text is then sent to the OpenRouter AI API, which uses the Mistral-7B-Instruct model to perform high-quality abstractive summarization. In cases of API failure, a fallback summarization method ensures reliability by producing a basic extractive summary. The summarized content is stored in a MongoDB database for easy retrieval and history tracking.

By combining multi-source input handling, AI-based summarization, and a simple interactive interface, Smart Content Summarizer offers a unique, all-in-one platform for users such as students, researchers, and professionals. This proposed solution ensures scalability, accuracy, and accessibility, making it an effective and practical tool for quick content understanding and knowledge extraction.

V. MODULES

The **Smart Content Summarizer** system is divided into several interconnected modules, each handling a specific function to ensure smooth operation and accurate summarization.

1. **User Interface Module:**
This module provides an interactive front-end where users can enter text, paste a URL, or upload a PDF file. It is developed using HTML, CSS, and JavaScript, ensuring simplicity, responsiveness, and ease of use.

2. **Input Processing Module:**
This module identifies the type of input (text, URL, or PDF) and prepares it for processing. For URLs, it extracts article content using Extractus and Cheerio libraries, while PDF files are read and converted into text using pdf-parse.

3. **Summarization Module:**
The core module communicates with the OpenRouter API, utilizing the **Mistral-7B-Instruct** model for generating high-quality summaries. It performs both extractive and abstractive summarization, depending on content type and length.

4. **Database Module:**
This module stores all inputs and summaries in a **MongoDB** database. It also supports retrieval, search, and deletion of past summaries.

5. **History & Management Module:**
This module allows users to view, search, and manage their previous summaries easily.

Together, these modules ensure seamless integration, reliability, and efficient summarization across multiple content formats.

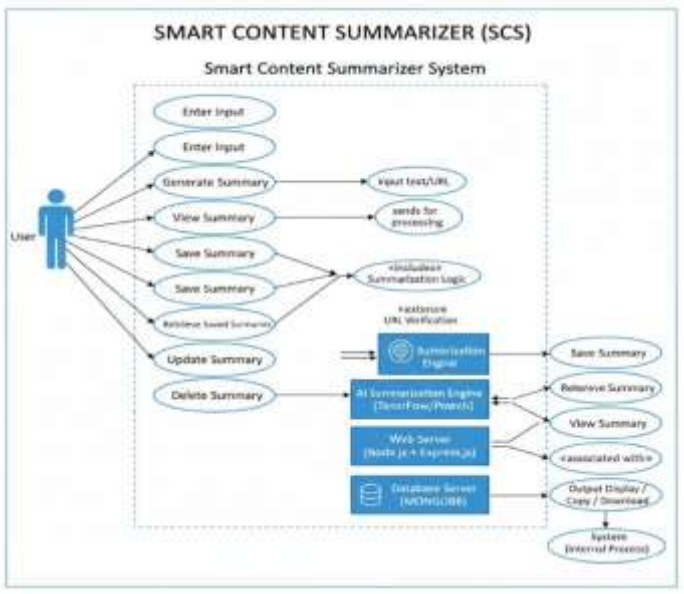

Figure 5.1

VI. IMPLEMENTATION

The System Workflow of the Smart Content Summarizer (SCS) explains how data moves through the system from the moment a user enters input until the final summary is generated and displayed. It represents the overall process of interaction between the frontend, backend, and database, highlighting each step involved in the summarization process.

The workflow follows a sequential pattern, ensuring that every operation — from input validation to summary generation and CRUD management — takes place efficiently and accurately. Workflow Steps:

1. User Input: The user enters text directly or provides a valid URL through the user interface.

2. Input Validation: The frontend checks whether the provided text or URL is valid and non-empty. Invalid inputs trigger alert messages.

3. Data Transmission: The validated input is sent to the backend server using an HTTP POST request through APIs.

4. Processing in Backend: The backend (Node.js + Express.js) receives the input and passes it to the Summarization Engine Module for processing.

5. Summarization: The engine analyzes the content, identifies key points, and generates a concise summary while maintaining context.

6. Response to Frontend: The generated summary is sent back to the frontend, where it is displayed on the screen.

7. CRUD Operations: The user can choose to save, update, or delete the summary using backend API routes connected to MongoDB.

8. Data Storage and Retrieval: MongoDB stores all summary records securely, allowing users to access them later.

Advantages of This Workflow:

- Ensures real-time summary generation.
- Provides efficient communication between all modules.
- Maintains smooth data flow and modular processing.
- Supports easy debugging and scalability for future improvements.

The System Workflow ensures seamless interaction between all components, resulting in a reliable and user-friendly summarization process that operates quickly and efficiently.

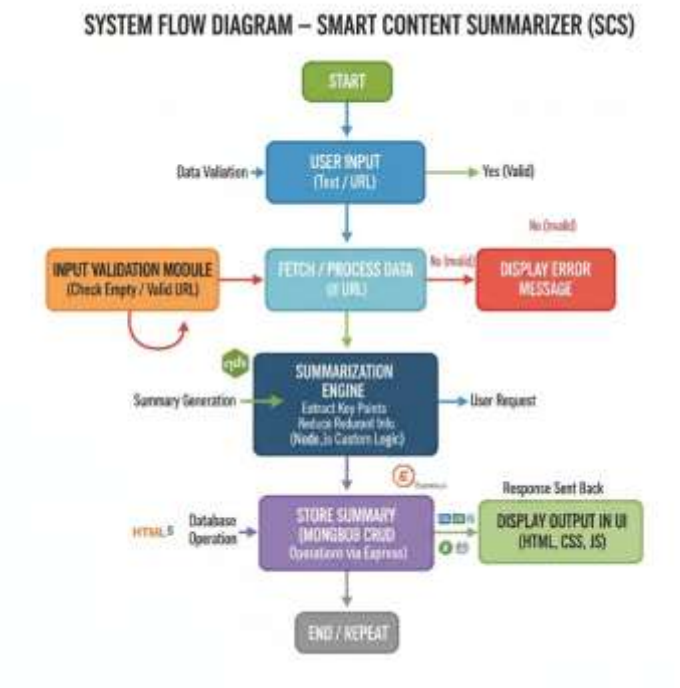

Figure 6.1

The Smart Content Summarizer (SCS) system provides a clean and user-friendly interface where users can interact easily, summarize content, and manage their results effectively. The output screens display each stage of user interaction — from input to result generation and CRUD operations.

Each screen has been designed using HTML, CSS, and JavaScript to ensure responsive layout, clarity, and accessibility across different devices

.

Homepage / Input Screen

- This is the main interface where users enter their text or paste a valid URL.
- It contains a large input area, a URL field, and a “Summarize” button.
- Once the user submits the content, the request is sent to the backend for processing.
- The design ensures readability with a simple and minimal layout.

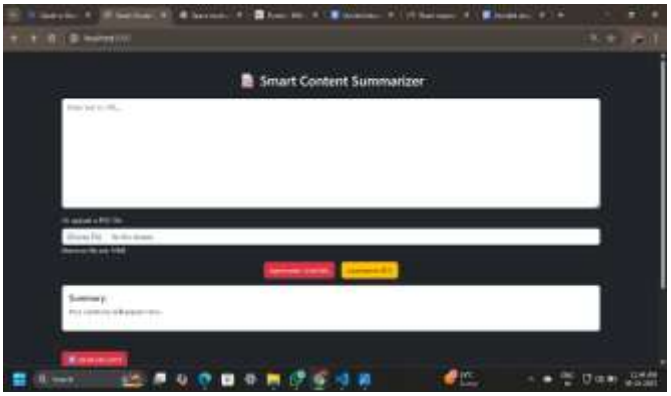

Figure 6.2 Home page

Summary Display Screen

- After processing, the summarized content is displayed instantly below the input field.
- The interface highlights the key points of the original text in a neatly formatted box.

- The screen also includes buttons to Save, Edit, and Delete summaries

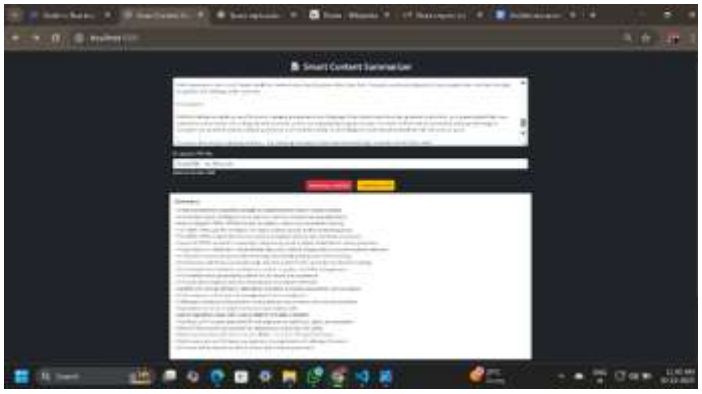

Figure 6.3 Text summarization

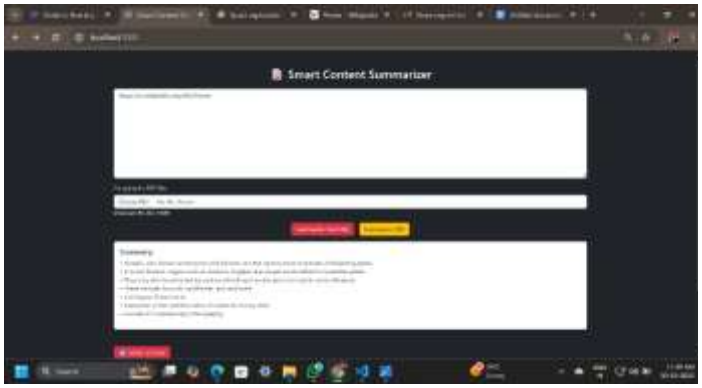

Figure 6.4 URL Summarization

Figure 6.5 PDF Summarization

VII. CONCLUSION

The Smart Content Summarizer (SCS) is a fully functional and efficient web application developed to simplify the process of extracting meaningful information from lengthy online content or textual data. The system successfully integrates frontend, backend, and database components to deliver a smooth and reliable summarization experience for users.

Through this project, the problem of manual summarization and time-consuming content reading has been effectively addressed. The system allows users to input text or a valid URL, processes it through the summarization engine, and instantly provides a concise summary while maintaining the original context. Additionally, the inclusion of CRUD operations enables users to store, edit, and manage their summaries in an organized way.

Developed using HTML, CSS, and JavaScript for the frontend and Node.js, Express.js, and MongoDB for the backend, the application demonstrates the power and versatility of full-stack development using modern web technologies. The modular architecture ensures easy maintenance, scalability, and integration of future improvements.

The project also served as a great learning experience, providing hands-on knowledge of how web technologies communicate in real time through APIs, how data is stored and retrieved efficiently, and how user-centric design improves usability.

Overall, the Smart Content Summarizer meets its intended objectives successfully — it reduces human effort, saves time, and enhances productivity by providing an automated way to summarize digital information effectively. The system not only offers practical benefits to users but also showcases the potential of simple, well-structured web development in solving real-world information management challenges.

VIII.REFERENCES

1. Nenkova, A., & McKeown, K. (2011). Automatic summarization. Foundations and Trends in Information Retrieval, 5(2–3), 103–233. https://doi.org/10.1561/1500000015

2. Gambhir, M., & Gupta, V. (2017). Recent automatic text summarization techniques: A survey. Artificial Intelligence Review, 47(1), 1–66. [https://doi.org/10.1007/s10462-016-9475-9](https://doi.org/10.1007/s10462-016- 9475-9)

3. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). Text summarization techniques: A brief survey. International Journal of Advanced Computer Science and Applications (IJACSA), 8(10), 397–402. [https://doi.org/10.14569/IJACSA.2017.081052](https://doi.org/10.14569/IJACSA. 2017.081052)

4. Dong, Y., & Zhang, M. (2020). Deep learning for text summarization: A review. IEEE Access, 8, 150360–150378. [https://doi.org/10.1109/ACCESS.2020.3015950](https://doi.org/10.1109/ACCES S.2020.3015950)

5. Gupta, V., & Lehal, G. S. (2010). A survey of text summarization extractive techniques. Journal of Emerging Technologies in Web Intelligence, 2(3), 258–268. [https://doi.org/10.4304/jetwi.2.3.258-268](https://doi.org/10.4304/jetwi.2.3.258-2 68)

6. See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (pp. 1073–1083). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1099

7. Paulus, R., Xiong, C., & Socher, R. (2018). A deep reinforced model for abstractive summarization. In the International Conference on Learning Representations (ICLR). [https://openreview.net/forum?id=HkAClQgA-](https://openreview.net/forum?id=H kAClQgA-)

8. Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 3721–3731).

Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1387

9. Narayan, S., Cohen, S. B., & Lapata, M. (2018). Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1797–1807).

10. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. [https://proceedings.mlr.press/v119/zhang20ae.html] and (https://proceedings.mlr.pres s/v119/zhang20ae.html)