1. ABSTRACT

The Search Engine Summarizer is an innovative application that provides a fast and efficient way for users to access information on any given topic. It is a unique search engine that stands out from traditional search engines by delivering real-time results of the latest and most relevant information from various sources including articles, blogs, forums, and more. This application employs NLP techniques like parsing, tokenizing, stemming and summarizing to transform the content into clear and concise summaries. To gather the latest news articles, the Search Engine Summarizer uses the newsapi, a popular news aggregator that collects news articles from various sources. Then, it employs sumy, a Python library for text summarization, to create summaries of the articles. The process starts with parsing, which breaks down the text into its constituent parts. The parsed text is broken down into small chunks called tokens. These tokens are then stemmed by stripping words to thier basic components. This makes comparisons between words easier. Finally, summarizing condenses the content into a short, readable summary. The resulting summaries are presented to the user, providing fast and easy access to the most important information on their desired topic. This approach eliminates the need for users to manually browse through large amounts of content to find what they need. The Search Engine Summarizer is an ideal tool for researchers, students, and anyone looking to stay informed on current events. By delivering fast, efficient summaries of the latest information from a range of sources, the application offers a significant advantage over traditional search engines. The Search Engine Summarizer offers a user-friendly interface that allows seamless navigation and interaction. Furthermore, the API provided will make it easier for developers and advanced users to integrate the application into their own projects. With its unique combination of features, the Search Engine Summarizer is a powerful tool that can be used in a variety of ways. Whether it is used to stay informed on current events or conduct research, the application provides a fast and efficient way to access the latest information on any given topic.

Keywords: Search Engine, Summarization, Text Summarization, Natural Language Processing, Information Retrieval

2. PROBLEM IDENTIFICATION

The abundance of information available on the internet can be overwhelming for every-day users, who may not have the time or expertise to sift through large amounts of data to find what they need.

Search engines and other tools often require users to manually sort through lists of webpages to find relevant information, which can be time-consuming and frustrating. Different types of documents (e.g., academic papers, news articles, and blog posts) may present information in different ways, using different formats, structures, and languages, making it hard for users to derive insightful conclusions from them. Users who lack prior knowledge or expertise in a particular topic may find it difficult to understand the information presented in online content, which may use technical language or assume a certain level of background knowledge. When users are under time pressure to find information, they may not have the luxury of reading through multiple documents to find what they need, which can lead to incomplete or inaccurate results.

The quality of information available online can vary widely, with some sources being unreliable, biased, or inaccurate, which can make it difficult for users to determine what information is trustworthy and accurate. Non-expert users find it challenging to understand a material when its language and terminology are too technical or complex, like content related to science, technology, etc. The lack of personalized content presentation can hinder users from quickly accessing information that aligns with their specific interests, preferences, or needs. Accessing information from devices with small screens, such as smartphones may be difficult for some forms of content, such as long-form articles or academic papers, which may require users to scroll through multiple pages or zoom in on text to read it clearly. The accessibility of online content poses challenges for users with disabilities, such as visual impairments or cognitive limitations. Many existing search engines and content sources may lack proper accessibility features, which can makes it challenging for users with disabilities to access and understand information.

3. NEED FOR A SOLUTION

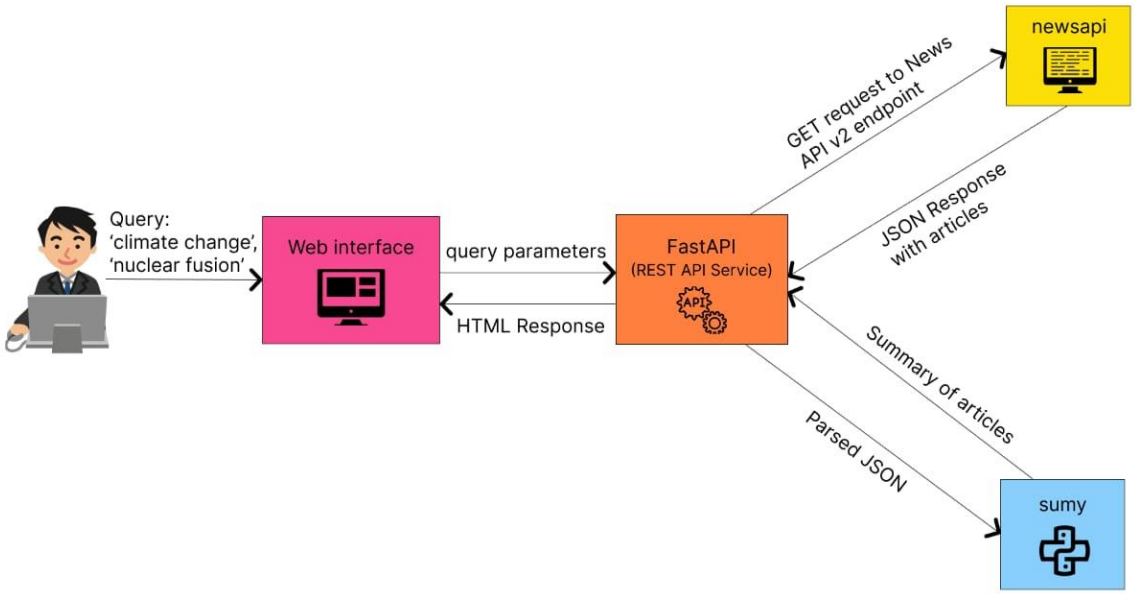
- People consume a lot of information from the internet on a daily basis. Therefore, having a system that can assist them in quickly finding the information they need is essential.
- Users with accessibility issues need more than just a search engine. The results should be accessible and sometimes a small summary of the information is enough for the user.
- Students and educators have special needs. Note-making, creating presentations, etc. are some of the tasks the system aims to help with.
- Confusing perspectives and opinions can be a problem for users and make them lose interest or get frustrated.

4. OBJECTIVES

- To provide an easy-to-use interface that allows a user to enter a query and optional arguments and receive a result set containing concise summaries/key information.
- To provide an API for advanced users to access the system's functionality programmatically. This will allow developers to integrate the system into their own applications.
- To present material online in a way that is user-friendly, clear, and accessible.
- To provide users with a personalized experience that aligns with their specific interests, preferences, or needs.
- To provide users with a set of tools to assist them in their research, including a citation generator, a note-taking tool, and a presentation generator.

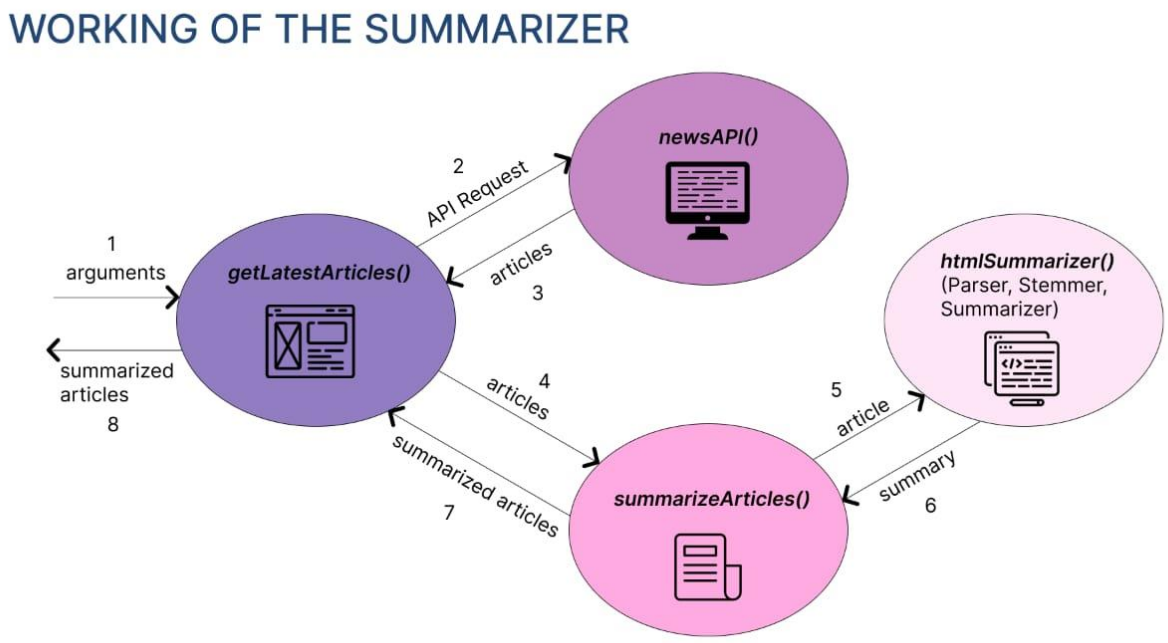
5. POSSIBLE SOLUTION

Our proposed system is implemented as an API that simplifies the process of fetching and summarizing relevant content based on user queries. When a query is provided as a parameter, the system interacts with the NewsAPI to retrieve the corresponding content from various sources. The received response, in the form of a JSON, is then parsed to extract the necessary information about the articles. To create concise summaries, the system utilizes a summarizer module that condenses the article content into key points. The summarized information is then returned as the output.



Architecture Diagram

The "summarizer" module consists of several functions that work together to retrieve and summarize the latest articles based on user-defined arguments. The first function, "getLatestArticles," takes arguments as input and interacts with the NewsAPI. It sends an API request to "newsAPI" using the provided arguments and receives a response containing a list of articles. Next, the "summarizeArticles" function takes the list of articles as input. It iterates through each article and applies the "htmlSummarizer" function to generate a summary for each article. The "htmlSummarizer" function takes an article as input and returns a concise summary based on the article's content. Finally, the "summarizeArticles" function returns a list of summarized articles, where each article is associated with its corresponding summary.



Working

Real-time summarization is another important feature of the proposed system. As new information becomes available, the system will continuously update the summaries, providing users with the latest and most accurate information. This real-time capability ensures that users stay informed about current events and developments in their chosen topic. To offer a personalized experience, the proposed system will provide options for user customization and preference settings. Users will be able to specify the length and level of detail they prefer in the summaries. This flexibility allows users to adjust the system's output according to their specific requirements and reading preferences.

6. CONSTRAINTS AND RESTRICTION

I. Materials

Hardware

- Disk space: 128GB
- Processor: 2-core CPU
- Memory: 4GB
- Display: Any display that supports 1024x768 resolution.

Software

Frontend

- React.js (v16)
- Node.js (v12)

Backend

- FastAPI (v0.96)
- Python (v3.10)
- Uvicorn (v0.22)
- sumy (v0.11)
- nltk (v3.8.1)
- newsapi-python (v0.2.7)

II. COST

A virtual machine capable of running the software above cost around Rs. 1000 per month. News articles are fetched from the News API, which is free for 100 requests per day. The cost of the hardware is therefore Rs. 1000 per month.

III. PRODUCTION PROCESS

A. Preparing Environment

An API key is obtained from the News API website. Python 3.10 is installed and the required libraries such as fastapi, newsapi, sumy, nltk, uvicorn, etc., are installed using pip. Node.js is installed and the required libraries such as react, react-dom, react-scripts, etc., are installed using npm.

B. Data Retrieval

The first step in implementing search engine summarization is to retrieve content from the web. news articles are retrieved from relevant sources. The collected data can be in the form of raw text, HTML, or XML. It is important that the data be relevant to the search query and is retrieved from reliable sources. This step is realized by using the News API.

C. Development

A virtual environment is created for the project. Django is used for the backend and React.js is used for the frontend. An endpoint is created for the backend to receive the search query from the frontend. The search query along with parameters such as number of articles, date etc., are passed to the newsapi client to retrieve the news articles. The news articles are then passed to the summarization module. The summarization module implements Sumy's parser with an English tokenizer. After removing stop words, summarization is done. The summary is then passed to the frontend. The search function calls the backend API with the search query and other parameters.

D. Testing and Deployment

The application is first tested by running it locally. Various search queries are tested, and the results are verified. The API was iteratively improved to provide better results. After much testing and getting satisfactory results, the API is deployed as a web service on Render.com.

7. BEST SOLUTION

- The proposed system is a very able and efficient system. Taking in a query string and a list of optional arguments, it can return a list of relevant results.
- The API proves to be a useful tool for customization and integration with other systems.
- The system can handle lots of data at once and is able to return results in a reasonable amount of time.
- The simple web interface provided can be used by anyone with a basic knowledge of computers and the internet.
- It does not require much computing power and can be run on a simple computer.
- Both the development and deployment process is very simple and requires very little time and effort.
- Also, the cost of development and deployment is very low.

8. DETAILED ENGINEERING ANALYSIS

I. Empathize

Our first step while developing the system was to empathize with the user. We laid out a few questions for ourselves to answer:

1. When searching in a search engine, what do we expect to see?

2. How much information do we expect to gain?

3. How much time do we spend on a typical search?

4. What are the most important factors that we consider while searching?

And then the idea of having a system that can provide us with the most relevant information in the least amount of time seemed like a good idea. To understand the problem better, we conducted a survey to understand the user's perspective. The survey was conducted among a few students of our college. We found out that most of the students spend around 5-10 minutes on a typical search and that they want to get the most relevant information in the least amount of time. Presenting the information in a concise manner is one of the best ways to achieve this. From the insights gained from the survey, we made a list of features that we wanted to implement in our system.

II. Define

In this phase, we defined the problem statement and the scope of our project. The user wants a system that will take in a query and produces a list of summaries. The manner in which the summaries are presented is also important. The user wants the summaries to be concise and relevant. A developer wants to be able to customize and integrate tools available to him. So, an API needs to be developed. This API can be used as a backend in our system and can also be used by other developers to integrate it into their systems.

Customization and personalization are important factors for a user. So, we decided to implement a feature that will allow the user to customize the summaries according to his needs. Sorting based on relevance, date, length, etc., length of the summary, etc., are some of the features that we decided to implement.

III. Ideate

With a firm grasp of user expectations and insights from the survey, we started the ideation phase. We started by listing out the features that we wanted to implement. We then started to think about the architecture of the system. We decided to use an API based architecture. This will allow us to integrate other tools and allow other developers to integrate our tool into their systems. The system should also be modular so that it can be easily extended. The system will be a simple web search interface with a RESTful API as backend and a Web interface as frontend.

We chose to use Python as our programming language. Python is a very powerful language and has a lot of libraries that can be used to implement our system. We also decided to use FastAPI as our API framework. FastAPI is a very powerful and lightweight framework that is easy to use and has a lot of features. We will also be using NewsAPI to get the news articles. NewsAPI is a very powerful API that provides news articles from a lot of sources. For the frontend, we decided to go with ReactJS.

IV. Prototype

After the architecture was decided, we started to develop the system. We started by developing the API. It is a minimal web application. It has a single endpoint that takes in a query and returns a list of summaries. The frontend consisted of a search bar with a search button. It also had an "Advanced Search" button that allowed the user to customize the search. The user can choose the order of articles, category, location, etc.,

V. Test

In the testing phase, we mainly focused on the following:

1. Functionality: Queries of varying complexities were fed into the system to assess its responsiveness, accuracy, and ability to generate concise and relevant summaries.

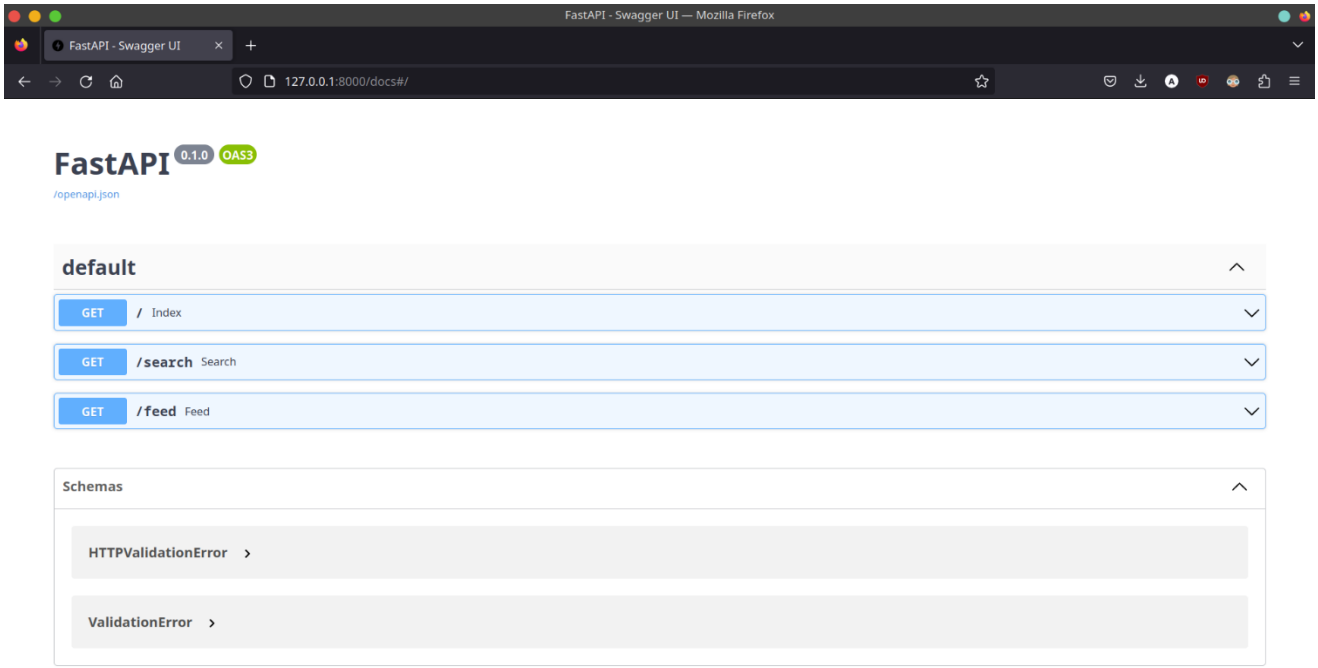
2. Integration: The integration of FastAPI, Python libraries, and external APIs such as NewsAPI was tested to ensure the system handles the workload efficiently and provide the user with the most relevant information in the least amount of time.

3. User experience: Our goals of usability, accessibility, and time efficiency were tested by having users interact with the system and provide feedback on their experience.

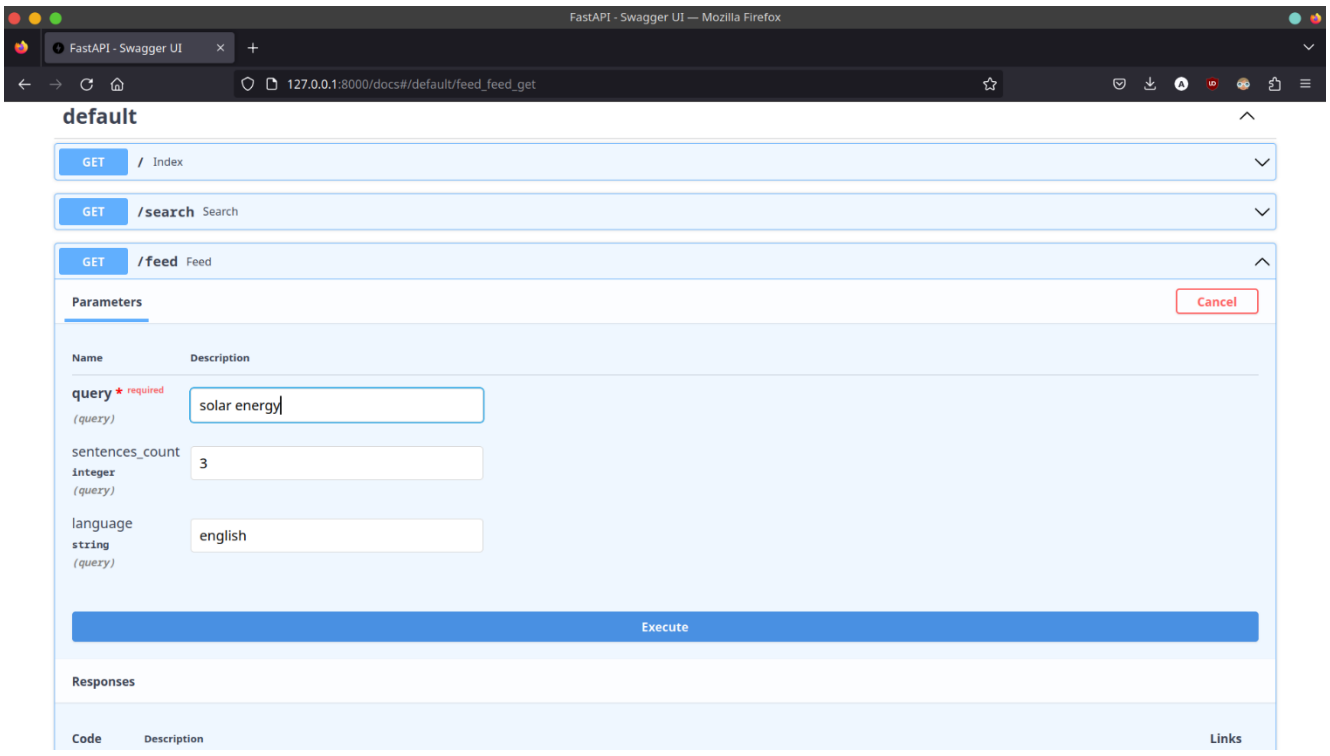
4. Performance: We put the system under varying levels of load to test its performance and scalability.

9. TEST RESULTS

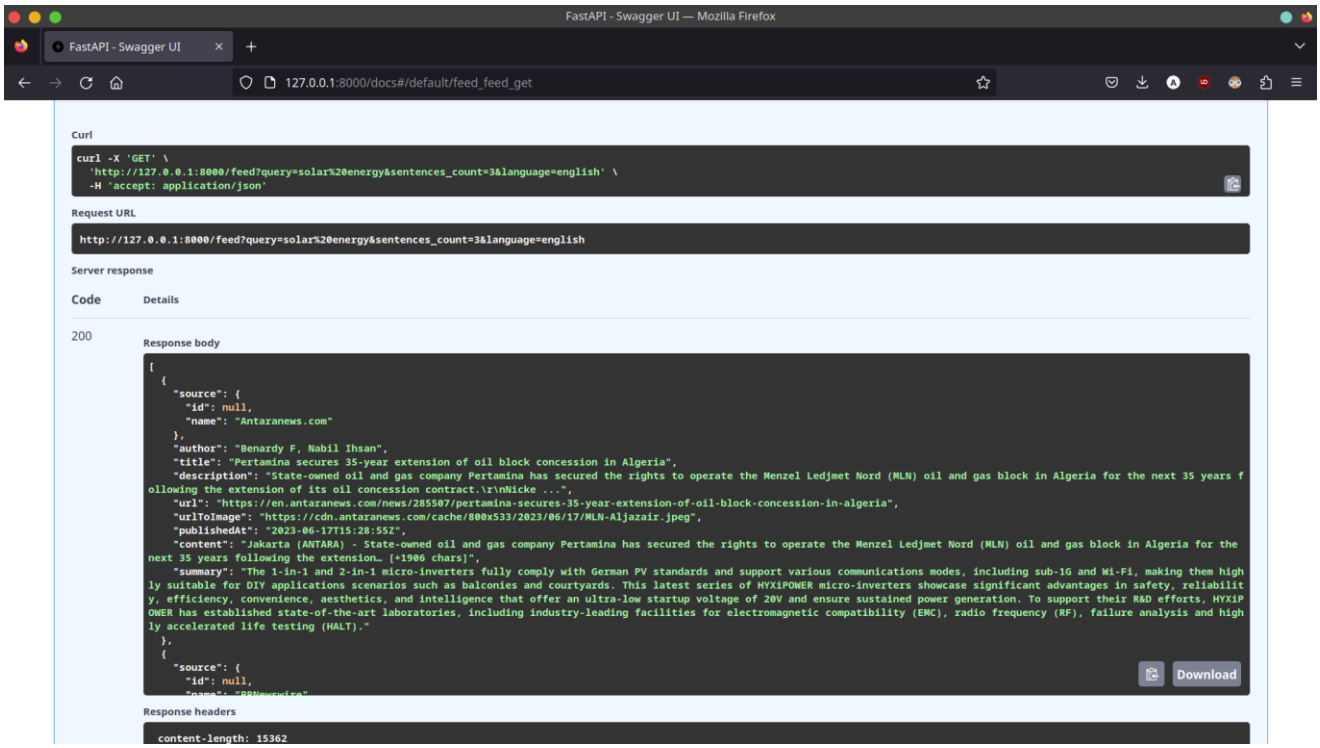
I. API



API

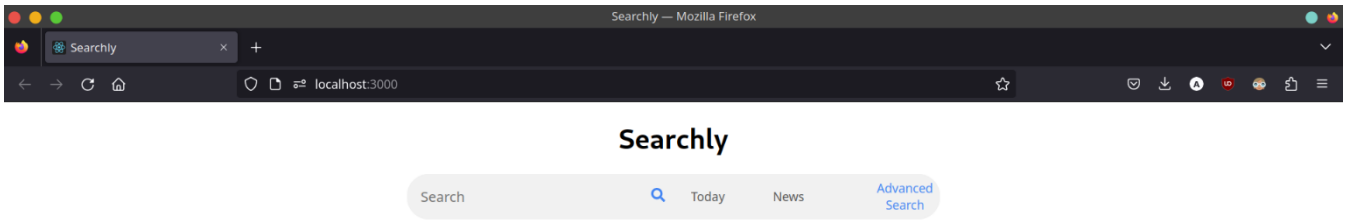


Query

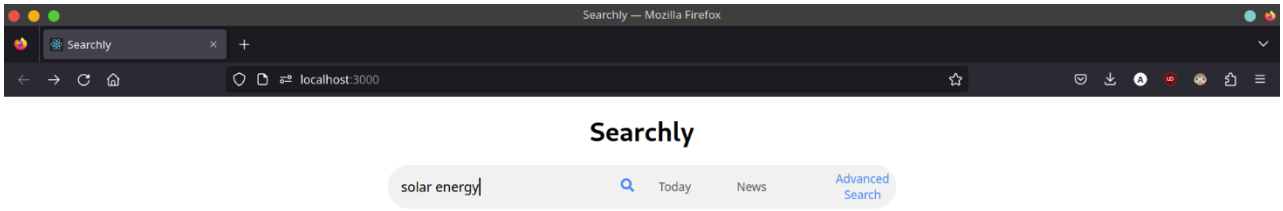


API result

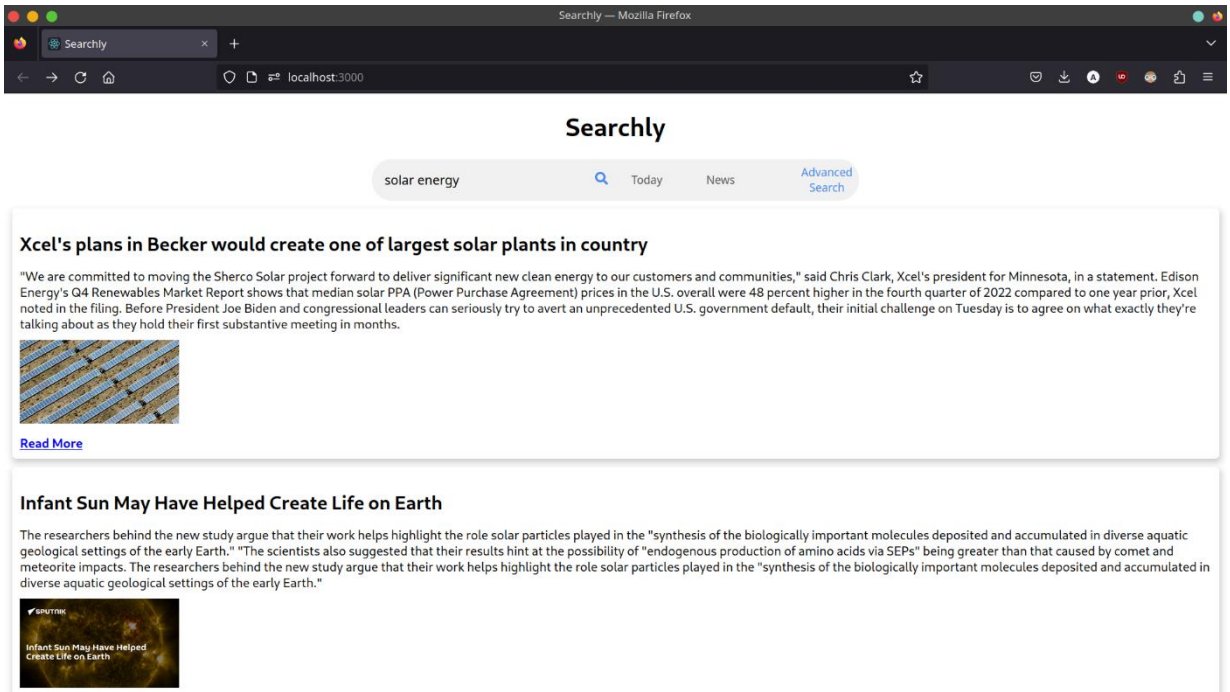
II. UI



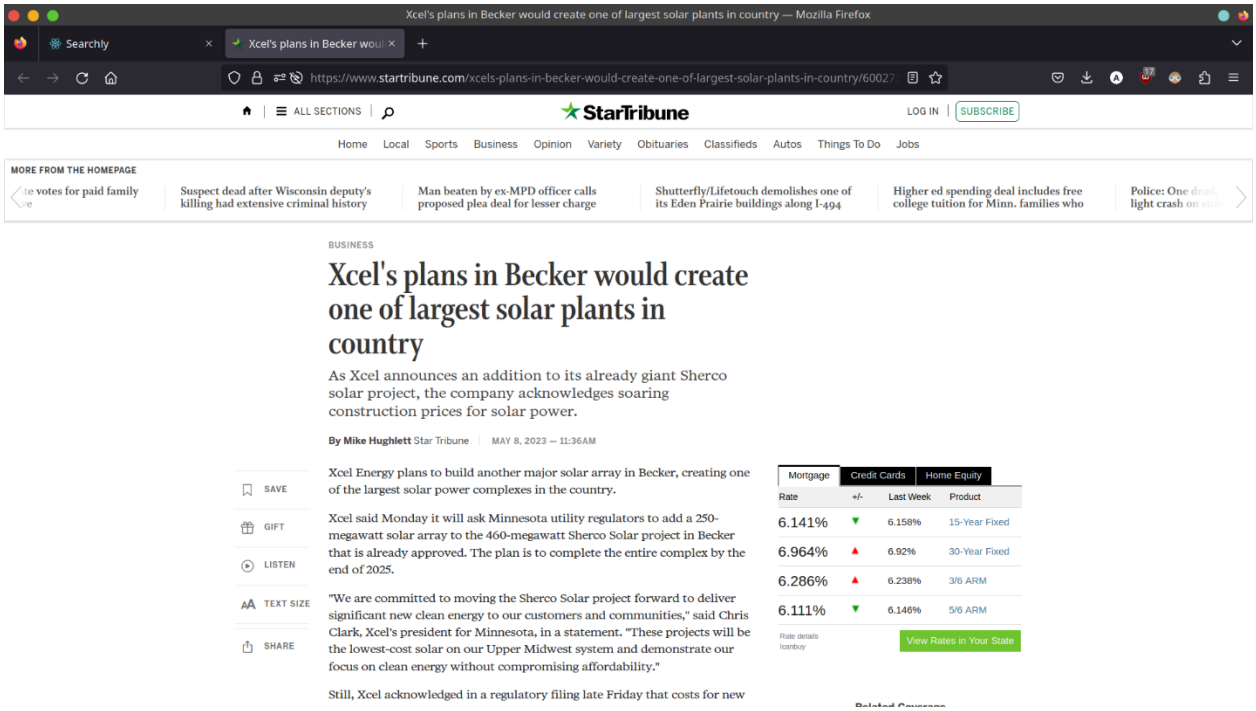
UI



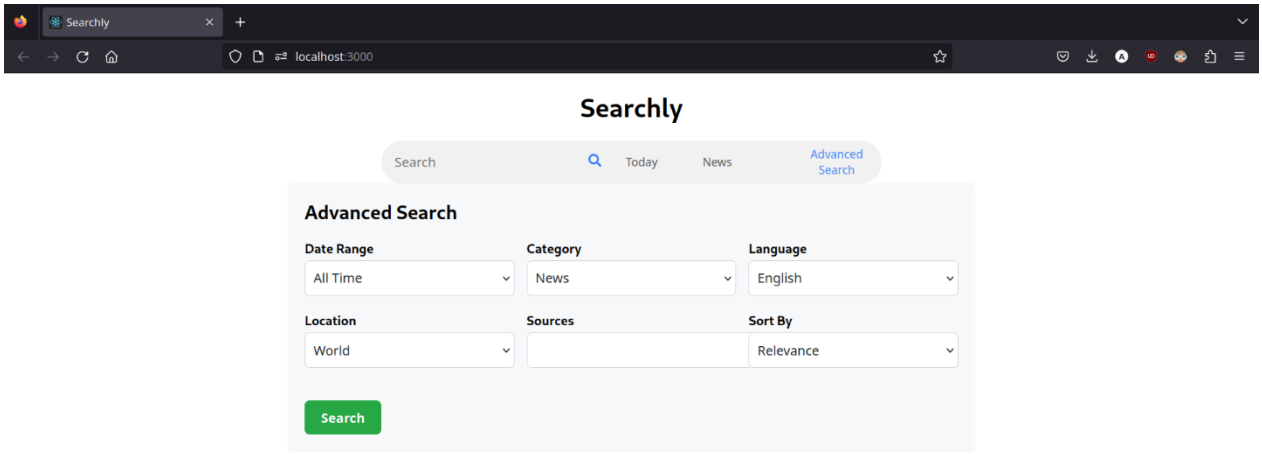
Query



List of summaries as a result of the query



Original article



Advanced search

10. CONCLUSION

The proposed system represents an innovative approach to the problem of summarizing search results. This can provide users with a quick and efficient overview of the most important information related to their search query and save them time and effort in going through long lists of search results. Moreover, the system is scalable and integration with existing search engines can be done easily, providing a reliable and efficient solution to summarize search results across a variety of domains and applications. The proposed system can potentially transform the way traditional search engines work and enhance the overall user experience.

11. PUBLICATIONS

Our project won the 3rd place in Schlumberger's New Year Hackathon held in January 2023 at Shaastra, IIT Madras.

Schlumberger Hackathon

12. REFERENCES

1. Mihalcea, R. and Tarau, P., 2004, July. Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).
2. Dragomir R. Radev and Weiguo Fan. 2000. Automatic summarization of search engine hit lists. In ACL-2000 Workshop on Recent Advances in Natural Language Processing and Information Retrieval, pages 99–109, Hong Kong, China. Association for Computational Linguistics.
3. H. Suo, W. Zhang and Y. Zhang, "Research on Automatic Summarization Based on Search Engine Result," 2009 International Conference on Web Information Systems and Mining, Shanghai, China, 2009, pp. 74-77, doi: 10.1109/WISM.2009.23.
4. Steinberger J, Jezek K. Using latent semantic analysis in text summarization and summary evaluation. Proc. ISIM. 2004 Apr;4(93-100):8.
5. Hal Daumé III and Daniel Marcu. 2006. Bayesian Query-Focused Summarization. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pages 305–312, Sydney, Australia. Association for Computational Linguistics.
6. H. Gupta and M. Patel, "Method Of Text Summarization Using Lsa And Sentence Based Topic Modelling With Bert," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 2021, pp. 511-517, doi: 10.1109/ICAIS50930.2021.9395976.
7. Mashechkin, Igor & Petrovskiy, Mikhail & Popov, D. & Tsarev, Dmitry. (2011). Automatic Text Summarization Using Latent Semantic Analysis. Programming and Computer Software. 37. 299-305. 10.1134/S0361768811060041.
8. Ou, Shiyan & Khoo, Christopher & Goh, Dion. (2007). Automatic multidocument summarization of research abstracts: Design and user evaluation. Journal of the American Society for Information Science and Technology. 58. 10.1002/asi.20618.
9. Chandra, Munehs & Gupta, Vikrant & Paul, Santosh. (2011). A Statistical Approach for Automatic Text Summarization by Extraction. Proceedings - 2011 International Conference on Communication Systems and Network Technologies, CSNT 2011. 268 - 271. 10.1109/CSNT.2011.65.
10. Zhong, Ming & Liu, Pengfei & Chen, Yiran & Wang, Danqing & Qiu, Xipeng & Huang, Xuanjing. (2020). Extractive Summarization as Text Matching. 6197-6208. 10.18653/v1/2020.acl-main.552.

11. Sharma, Grishma & Sharma, Deepak. (2022). Automatic Text Summarization Methods: A Comprehensive Review. SN Computer Science. 4. 10.1007/s42979-022-01446-w.
12. Cajueiro, Daniel & Gomes Nery, Arthur & Tavares, Igor & Melo, Maisa & Reis, Silvia & Weigang, Li & Celestino, Victor. (2023). A comprehensive review of automatic text summarization techniques: method, data, evaluation and coding. 10.48550/arXiv.2301.03403.