International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 03 | March - 2024 SJIF Rating: 8.176 ISSN: 2582-3930

An Article Summarizer Application using OpenAI's GPT Model

[1] Mr. Rajat Raj , [2] Mr. Prakash Kumar Sharma, [3] Mr. Rohit kumar ,
[4] B. Yamini [5] Dr.D.Usha ,[6] Dr.T. Kumanan,

[1][2][3] Student, B. Tech CSE, Department of CSE, Dr. M.G.R. Educational and Research Institute
[4]Professor, Department of CSE, Dr. M.G.R. Educational and Research Institute
[5] Professor, Department of CSE, Dr. M.G.R. Educational and Research Institute

ABSTRACT

Due to the growing amount of online content, automated article summarization using artificial intelligence (AI) has received a lot of interest lately.This study proposes a novel method to automate the process of article summarization by combining Generative AI approaches with React JS, a well-liked JavaScript toolkit for creating user interfaces. React JS integration offers a user- friendly and interactive platform that allows user to submit articles and instantly receive succinct summaries. Neural networks and deep learning architectures are two examples of generative AI models that are used to extract important information and produce cogent summaries that encapsulate the main ideas of the original articles.The system's technological implementation— which includes data preparation, model training, and the creation of real-time summaries—is covered in this paper. The proposed method has a number of benefits, such as a faster summarizing process, better information accessibility, and an improved user experience. ReactJS and Generative AI.summarization system, making it possible to effectively extract important information from articles in a variety of domains.

Index Terms - React JS, Generative AI, Neural Networks, RTK Query .

INTRODUCTION

Since digital content is growing so quickly, people now have access to an abundance of information. In this age of information overload, automated article summarizing with artificial intelligence (AI) has become a practical way to extract key insights and provide concise summaries of articles. To automate the process of summarizing articles, this study This presents an innovative strategy that combines Generative AI techniques with React JS, a popular JavaScript library for building user interfaces. Because manual publishing summaries need human readers to carefully evaluate and summarize the main concepts of lengthy texts, they have traditionally required a significant investment of time and energy.

There are several benefits to using React JS in the article summary process. Thanks to React JS, users can submit articles and receive real-time summaries on an easy-to-use and entertaining platform. Because of its component-based architecture, which also guarantees a seamless user experience, summarizing is quick and easy. Meanwhile, the use of generative AI techniques an important part in extracting the most important information from articles and creating logical summaries. These methods make use of deep learning architectures and neural networks to extract semantic and contextual information from massive datasets. The system can extract important information, condense it, and provide summaries while preserving the original context by using Generative AI models.

The technological implementation of the suggested system, including data preparation, model training, and real-time summarization creation, is the focus of this research study. In order to make sure that the system generates accurate and significant results, it also goes over the assessment criteria that are used to gauge the quality and correctness of the generated summaries. While React JS integration and Although generative AI has many benefits, it is important to recognize the drawbacks of this methodology. It may be difficult for generative AI models to capture subtle contextual information, and they will need ongoing training in order to adjust to changing text patterns. Additionally, creating coherent summaries is significantly hampered by the need to strike a balance between brevity and the preservation of important details.

LITERACTURE SURVEY

The sheer volume of information available online in the modern digital age can be debilitating. It might be difficult for users to stay up to speed with the steady stream of news updates, research papers, blog entries, and articles. One way to solve this problem is by automated article summarizing, which reduces long articles into brief summaries so that readers may rapidly understand the essential ideas and select the articles they want to read. are worthy of more investigation. Manually summarizing.

React JS: User interfaces are made with this popular JavaScript package. It was developed by Facebook and is widely used to make dynamic, interactive web applications. React JS's component-based architecture enables developers to manage state changes and produce reusable user interface elements. It uses a virtual Document Object Model (DOM) for rendering. Updates to the user interface that boost performance and experience Quickly Vite is a state-of-the-art tool for developing web applications. Its primary focus is on development and builds, which makes it an ideal choice for React JS projects. [1].

Vite is a state-of-the-art tool for developing web applications. Its primary focus is on development and builds, which makes it an ideal choice for React JS projects. Vite leverages native ES modules to enable faster development server startup times and better, faster production builds. It also provides features like hotmodule to enable more frequent code updates during development.[2].

Tailwind CSS: Tailwind CSS facilitates rapid user interface design. It is a utility-first framework. Tailwind CSS, together with the utility CSS framework, offers a collection of classes that may be used to produce unique designs, in contrast to frameworks that rely on pre-designed parts. There are several prebuilt utility classes available for responsive designs, stylistic elements, and other purposes. Tailwind CSS encourages a mobile-first strategy and makes it easier to create UI/UX that is aesthetically pleasing and responsive[2].

RTK Query: RTK Query is a state management library built on top of Redox Toolkit. It makes data fetching and caching in React apps simpler by providing a robust and perceptive API for managing remote data. RTK Query helps with managing API calls, caching results, and handling loading and error statuses. It functions well with React .

Local Storage : Users can save data locally on their device by utilising this web browser function. It provides cross-browser session data persistence along with an easy-to-use key-value storage mechanism. The project can keep user history, such as user preferences and previously summarised articles, in local storage. By saving data even when the browser is closed or the page is reloaded, it enables a seamless user experience.

PROBLEM STATEMENT

Digital Information Overflow Online world is full of information-packed articles. Getting important insights quickly is tough. Reading long articles takes a lot of time and energy :People avoid reading due to the effort needed Lengthy articles can be discouraging. Lack of User-Friendly Interfaces Existing platforms lack intuitive designs and interactive tools. A solution is needed to make reading easier. We need a way to make content consumption smoother.

The problem statement addressed in this research is the growing challenge posed by the overwhelming volume of online content, leading to information overload for users. In response to this, the research focuses on the need for efficient and automated article summarization using Artificial Intelligence (AI). Despite the potential benefits of AI-based summarization, there are acknowledged issues, including the limitations of Generative AI models in capturing nuanced contextual information and the requirement for continuous model training to adapt to evolving text patterns. The paper aims to develop a solution by proposing a novel system that integrates React JS with Generative AI to streamline the article summarization process, enhance user experience, and address the challenges associated with information overload in the digital era.

PROPOSED SYSTEM

The proposed system introduces an AI-powered article summarizer, integrating React JS and Generative AI techniques. Users can leverage an intuitive platform, entering articles to receive real-time concise summaries. Generative AI, employing neural networks and deep learning, ensures the extraction of salient information and coherent summaries. Technical aspects include data pre-processing, model training, and real-time summarization. Evaluation metrics ensure reliable results, acknowledging challenges like nuanced contextual capture and continuous model training. Future research directions explore hybrid summarization approaches and multi-modal inputs. The UI/UX design prioritizes consistency, simplicity, and visual hierarchy, incorporating the contemporary Glass Morphism trend. Advanced API requests are implemented using RTK Query, enabling pagination, filtering, sorting, and custom queries. User history is stored in local storage for seamless data retrieval. Testing involves functional, performance, and robustness evaluations, comparing AI-generated summaries with human-written counterparts. The system aims to revolutionize information accessibility and processing in the digital era[4]

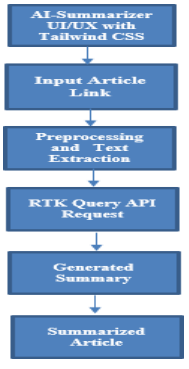

Figure 1: Conceptual Representation

UI/UX DESIGN :

Digital product design requires the creation of an intuitive and visually appealing user interface (UI) and user experience (UX). UI design is focused on the look and feel of the user interface, whereas UX design focuses on enhancing user satisfaction and usefulness by ensuring as mooth and intuitive user experience A number of principles are used in UI/UX design to create efficient and interesting interfaces. These guidelines consist of: Consistency Users will have a more seamless experience if interface design components like colors, fonts, and layouts are kept consistent. Simplicity To facilitate user comprehension, improve usability, and reduce cognitive load, designers should strive for minimalism and simplicity. Burden Visual Hierarchy Organizing and prioritizing interface elements to guide user attention and facilitate easy navigation. User feedback To improve interaction clarity and help users identify their actions, provide them with interactive and visual clues such as button animations or hover effects.

RTK QUERY API INTEGRATION:

RTK Query is a powerful data retrieval and caching module provided by Redux Toolkit (RTK). It simplifies API integration and state management by generating API request hooks and automatically managing data caching, invalidation, and re-fetching.[8]

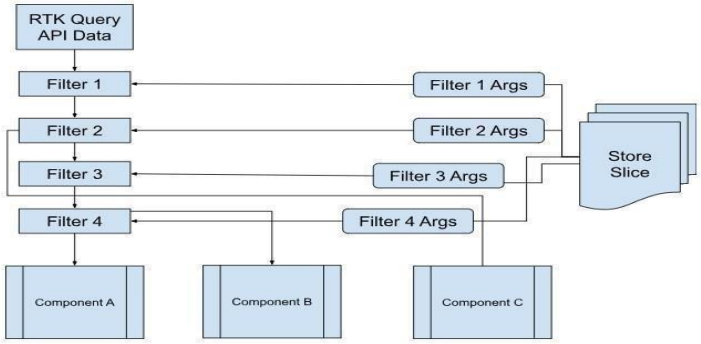

Figure 2 : RTK Query Architecture

We applied the required criteria in use Effect and use Memo, and we used query in every component. This indicates that at least two filters are repeated n times, which is bad After utilizing the create Slice and extra Reducers options and waiting for the query to finish, we execute the filtering procedure. This is fantastic. since I can use the filter parameters in the reducer, but I don't believe there is a method to repeat the process with new arguments once the filter arguments change but the query data doesn't. After publishing data to a slice following each related filter stage, we subscribed one component, then subscribe each component to the associated data. This is how I presently have it set up, however it is not ideal because it couples components together that I want to avoid doing, bloats one component that was sort of chosen at random, and results in a lot of huge state activities that slow down my application. After raising the query subscription to the common ancestor component, we passed data as props. It is not ideal that these components are at different depths with respect to their common parent; I imagine that prop drilling would be required for some of the components.

To share the outcomes of the first two filter processes with the respective components, we also used React Context. I haven't given this much thought yet; would a query subscription work with it? It seems sense to me that the best solution would be a callback that acts as a mediator between the component's subscribed data and the API response. I am conscious of the change. Although the response option is defined in the API slice, I don't think it's appropriate or feasible for this situation

Implementing API Requests using RTK Query:

Advanced API requests, like pagination, filtering, sorting, and custom queries, can be easily implemented with RTK Query versatile syntax for defining these advanced features is offered by RTK Query Pagination In the endpoint setup, it specifies pagination settings such as page size and current page number. RTK Inquiry Pagination is handled automatically, and methods and hooks relating to pagination are provided to let users to browse through the paginated data. Filtering and Sorting It adds query parameters to the endpoint setup in order to filter and sort data. With RTK Query, you may specify quer parameters and dynamically change them to retrieve data that has been sorted and filtered Custom Queries RTK Query supports custom queries where you can define specific parameters and structures required by your API. You can create custom endpoints and hooks to fetch data using your own query configurations

Generative AI Algorithm

subset of artificial intelligence known as "generative AI" is dedicated to creating new data instances that bear similarities to a given dataset. Generative AI models seek to discover and replicate the underlying patterns and structures of the training data in order to generate new and distinct data instances, in contrast to other AI techniques that concentrate on categorization or prediction tasks. Artificial Intelligence Algorithms make use of statistical methods and probabilistic models to comprehend and record the distribution of data. By assimilating the patterns, correlations, and dependencies found in the training data, these models are able to produce fresh data samples with comparable traits.[7]

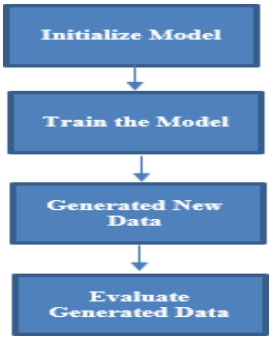

Figure 2: Generative AI Architecture

In the figure 2 below we made a simplified flowchart to explain the basic steps involved in a generative AI algorithm Initialize Model In this stage, the generative AI model, which could be a Generative Adversarial Network (GAN), a Variation Auto encoder (VAE), or any other generative model, is initialized. At this point, the parameters and architecture of the model are configured. Model Training The model is trained using a dataset including actual data samples. The model's parameters need to be optimized during the training phase in order to comprehend the underlying structures and patterns seen in the training data. This makes it easier for the model to replicate the data distribution of the training set and generate new, comparable data instances. Generate New Data By sampling from the learnt distribution, the trained model can produce new data instances. Although they are not exact duplicates, these created instances might have traits in common with the training set.

The model generates a variety of outputs by using latent variables or random noise as input. Evaluate Generated Data A variety of evaluation criteria or human judgment can be used to determine the generated data's quality and fidelity. This stage aids in determining if the generated data satisfies the intended criteria and how effectively the generative model has learned and whether the generated data aligns with the desired criteria. Digital product design requires the creation of an intuitive and visually appealing user interface (UI) and user experience (UX). UI design is focused on the look and feel of the user interface, whereas UX design focuses on enhancing user satisfaction and usefulness by ensuring as mooth and intuitive user experience A number of principles are used in UI/UX design to create efficient and interesting interfaces

HISTORY AND LOCAL STORAGE MANAGEMENT

Web apps can save data locally on a user's device thanks to a feature of web browsers called local storage. It provides a simple key-value storage mechanism that may be used to store and retrieve data when a user closes their browser or leaves a web page. You can take the following actions to use local storage to store and manage user history Record the History of Users: Take note of the pertinent information or event details whenever a big event happens in your application that you want to monitor as part of the user's history such as page navigation or action completion Serialize and Store Data[9]

To make sure the recorded data is compatible with local storage, convert it into a serialized format, like JSON. To store the serialized data, use the web browser's local Storage API. The history data that has been stored can be identified using a special key. Retrieve User History Use the key you were given to access the data from local storage in order to retrieve the user history. To prepare the data for additional processing or presentation, desterilize it from its serialized form and return it to its original format. Update and Manage History Keep recording and updating the user's history information as they engage with your application. If necessary, you can erase the entire history or add new occurrences to the one that already exists.

RESULTS AND EVALUATION

Functional Testing To make sure the AI Summarizer carries out its primary functions as intended, it thoroughly tests the system. Examine a range of input articles with varying lengths and levels of complexity to confirm the precision and caliber of the summaries that are produced. Performance testing It assesses the AI Summarizer's effectiveness by monitoring resource consumption and reaction time. To evaluate its effectiveness and scalability, test it with various workloads and input sizes. Determine any performance problems or bottlenecks, then adjust the infrastructure or algorithm as necessary.

Comparing the AI Summarizer's output with human-generated summaries or pre-existing benchmark datasets is the third step in the process. To determine the efficacy of the AI-generated summaries, compare their overall quality, coherence, and resemblance to those authored by humans4. Robustness Testing: To assess how the AI Summarizer responds to different edge circumstances, such unclear or ill-structured articles, it conducts robustness testing. We additionally assess its capacity to manage unexpected sources and generate insightful insights.

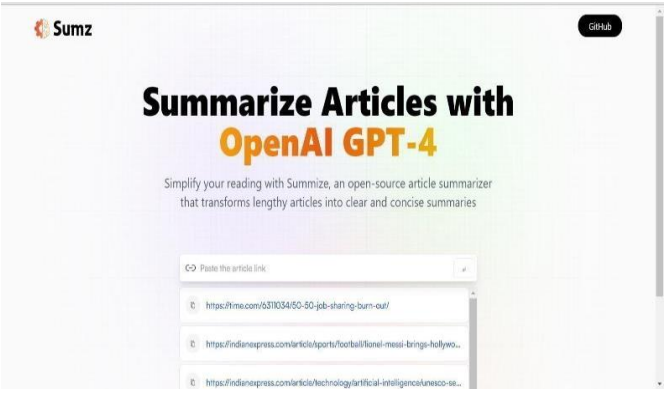

Figure 4: UI/UX of AI Article Summarizer

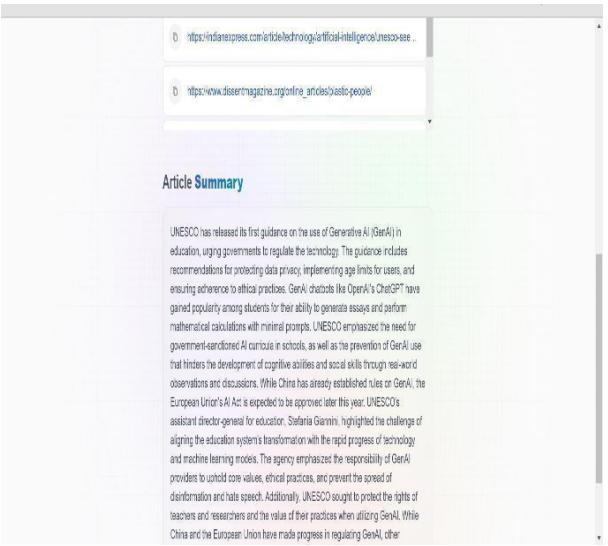

Figure 5: Summary of an Article

CONCLUSION

In conclusion, starting this paper is a great chance to learn a lot about a variety of topics while also creating a thorough application. You will gain knowledge about establishing a ReactJS project utilizing Vite, a cutting-edge and effective tooling system, by working on this project We will also get a taste of UI/UX design through making responsive and visually appealing interfaces and discover how to provide our application a contemporary and slick appearance by using the well- liked glass morphism design trend using Tailwind CSS. The paper also focuses on enhancing functionality by implementing advanced RTK query API requests. You will gain experience in creating queries that trigger based on specific conditions, allowing for dynamic data retrieval and customization. Additionally, this work emphasizes the importance of data management and user experience.

FUTURE SCOPE

In future, our algorithm more congenial with more features such as:-
Multilingual Support: Enhance the AI Summarizer to support multiple languages, making it more accessible to a global audience and accommodating diverse content sources Customization Options: Introduce user preferences for summarization styles, allowing users to customize the level of brevity, inclusion of specific details, or emphasis on certain content aspects. User Feedback Mechanism: Integrate a feedback system for users to provide input on the quality of summaries, enabling iterative improvements based on real-world usage and user preference Integration of Audio and Visual Summarization: Explore the integration of multi-modal inputs, such as audio and images, to provide more comprehensive summarization, accommodating diverse content formats

REFERENCE

1 Agarwal, H. 2022. ReactJS Components. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/reactjs- components/. Accessed 17 December 2022.

1 Anthony, A., Nathaniel, M., & Lerner, A. 2017. Fullstack React: The Complete Guide to ReactJS and Friends.Fullstack.IO.

[3] Borusiuk, Y. 2022. Top 10 Benefits of React.js forDevelopment. Application Online. NCube. Available at: https://ncube.com/blog/top-10-benefits-of-react-js-for-application-development. Accessed 03 March 2022.

[4] hi, C. 2021. "A Beginner's Guide to Data Flow Diagrams". Blog.Hubspot.Available at: https://blog.hubspot.com/marketing/data-flow-diagram. Accessed June 10, 2023.

[5] Didacus O. 2018. System Design in Software Development. Online. Medium. Available at: https://medium.com/the-andelaway/system-design-in-software-development- f360ce6fcbb9. Accessed 21 March 2023.

[6] Document Object Model (ODM), 2023. Tips and Tricks. Meklit. http://www.meklit.com/index.php/documentobject- model-odm. Accessed 05 December 2022.

[7] Eygi, C. 2020. React.js for Beginners — Props and State Explained. Online. freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/react-js-for-beginners- props-state-explained/. Accessed 29 January 2023.

[8] Hammad, M. 2020. "Use Case Diagram for Library Management System". Preprint. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/use-case-diagram-for- library-management-system/. Accessed June 13, 2023.

[9]Javatpoint. 2021. ReactJS State Vs Props. Online. Javatpoint. Available at: https://www.javatpoint.com/react- state-vs-props. Accessed 10 February 2023.