FUSENA: Journal Sains and Technology
p-issn/eissn: **xxxx-cxxx/cccc-xxxx**
doi: **xxxxxxxxxxx.10**

OPTIMIZATION OF SEARCH FILTERS USING REACTJS IN WIKI-INF

Muhammad Zuhrul Ikfa
Universitas Nahdlatul Ulama Al Ghazali Cilacap, Indonesia
zuhrulikfamhd@gmail.com

Histori Manuskrip

Submit:
10 January 2026

Review:
15 January 2026

Recepted:
28 January 2026

Abstract
The rapid growth of digital information has increased the demand for efficient and responsive search systems capable of delivering accurate results in real time. Wiki-Inf, as an information management and knowledge-sharing platform, requires an optimized search mechanism to improve user experience and information retrieval efficiency. This study aims to optimize the search filter functionality in Wiki-Inf by utilizing ReactJS as the primary frontend framework. The implementation focuses on developing dynamic and interactive filtering features that enable users to search and sort information based on multiple criteria without requiring full page reloads. The optimization process employs ReactJS features such as state management, component reusability, and virtual DOM rendering to enhance application performance and responsiveness. The research adopts a development and testing approach by comparing the performance of the search system before and after optimization, considering response time, rendering efficiency, and user interaction quality. The results indicate that the implementation of optimized search filters using ReactJS significantly improves search speed, reduces unnecessary rendering processes, and enhances overall user satisfaction. The study concludes that ReactJS provides an effective solution for building scalable and high-performance search functionalities in modern web-based information systems such as Wiki-Inf.
Keywords: ReactJS, Search Filter Optimization, Wiki-Inf, User Experience, Web Application Performance, Information Retrieval.

Abstrak
Pesatnya pertumbuhan informasi digital telah meningkatkan kebutuhan akan sistem pencarian yang efisien dan responsif yang mampu menyajikan hasil pencarian secara akurat dan real-time. Wiki-Inf sebagai platform pengelolaan informasi dan berbagi pengetahuan memerlukan mekanisme pencarian yang optimal untuk meningkatkan pengalaman pengguna dan efisiensi temu kembali informasi. Penelitian ini bertujuan untuk mengoptimalkan fungsi filter pencarian pada Wiki-Inf dengan memanfaatkan ReactJS sebagai framework utama pada sisi frontend. Implementasi difokuskan pada pengembangan fitur penyaringan yang dinamis dan interaktif sehingga pengguna dapat melakukan pencarian dan penyortiran informasi berdasarkan berbagai kriteria tanpa memerlukan pemuatan ulang halaman secara keseluruhan. Proses optimasi memanfaatkan berbagai fitur ReactJS seperti manajemen state, penggunaan kembali komponen (component reusability), dan teknologi Virtual DOM untuk meningkatkan kinerja dan responsivitas aplikasi.

Penelitian ini menggunakan pendekatan pengembangan dan pengujian dengan membandingkan kinerja sistem pencarian sebelum dan sesudah dilakukan optimasi berdasarkan waktu respons, efisiensi rendering, dan kualitas interaksi pengguna. Hasil penelitian menunjukkan bahwa implementasi filter pencarian yang dioptimalkan menggunakan ReactJS mampu meningkatkan kecepatan pencarian secara signifikan, mengurangi proses rendering yang tidak diperlukan, serta meningkatkan tingkat kepuasan pengguna secara keseluruhan. Penelitian ini menyimpulkan bahwa ReactJS merupakan solusi yang efektif untuk membangun fungsionalitas pencarian yang skalabel dan berkinerja tinggi pada sistem informasi berbasis web modern seperti Wiki-Inf.
Kata Kunci: ReactJS, optimasi filter pencarian, Wiki-Inf, pengalaman pengguna, kinerja aplikasi web, temu kembali informasi.

A. Introduction

The rapid development of information technology has transformed the way users access, manage, and retrieve information through web-based applications. Modern users expect information systems to provide fast, accurate, and responsive search experiences to support their activities efficiently. Search functionality has therefore become one of the most important components in information systems, particularly in platforms that manage large volumes of data and knowledge resources (Shneiderman; 2016).

As the amount of digital information continues to grow, traditional search mechanisms often face challenges related to performance degradation and reduced usability. Users frequently experience delays in obtaining relevant information due to inefficient filtering mechanisms and repeated page reloads. Consequently, the optimization of search filters has become an essential aspect of improving both system performance and user experience in web applications (Hearst; 2009).

Wiki-Inf is a web-based information platform designed to facilitate information sharing and knowledge management among users. The increasing amount of content stored within the platform requires an effective search system capable of filtering information dynamically based on user preferences and search criteria. Without proper optimization, the search process may become slower and less responsive, ultimately reducing user satisfaction and application usability (Nielsen; 2012).

Recent advancements in frontend technologies have introduced more efficient approaches to developing interactive user interfaces. Among these technologies, ReactJS has emerged as one of the most widely adopted JavaScript libraries for building modern web applications due to its

component-based architecture and efficient rendering mechanism through the Virtual DOM (Facebook; 2023). These features allow developers to create dynamic interfaces that can update data in real time without requiring full page reloads.

The implementation of search filters using ReactJS provides several advantages, including faster rendering performance, improved state management, and enhanced component reusability. Through the use of hooks and state management techniques, ReactJS enables filtering operations to be executed efficiently while minimizing unnecessary rendering processes. As a result, users can obtain search results more quickly and interact with the system more smoothly (Banks & Porcello; 2020).

Several previous studies have discussed the implementation of search optimization techniques in web applications. Research conducted by Wieruch (2022) demonstrated that ReactJS significantly improves interface responsiveness in applications requiring frequent data updates. Similarly, another study found that optimized filtering mechanisms contribute positively to user satisfaction and application performance in information retrieval systems (Pressman & Maxim; 2020). However, studies specifically examining the optimization of search filters within the Wiki-Inf platform remain limited.

Based on these considerations, this study aims to optimize the search filter functionality in Wiki-Inf using ReactJS technology. The research focuses on improving the efficiency of information retrieval, reducing response time, and enhancing user interaction quality. The findings are expected to contribute to the development of scalable and high-performance search systems in modern web-based information platforms.

B. Theoretical Study

1. Search Filter in Information Systems

   Search filters are mechanisms used in information systems to refine and limit search results according to specific criteria determined by users. The implementation of filtering systems allows users to retrieve relevant information more efficiently, especially when dealing with large datasets. Effective search filters contribute significantly to improving information retrieval accuracy, reducing search time, and enhancing user satisfaction in web applications (Hearst; 2009).

In modern web systems, search filtering is no longer limited to simple keyword matching but includes advanced functionalities such as category filtering, sorting, range selection, and dynamic queries. These features enable users to interact with information systems more effectively and obtain results that closely match their needs (Baeza-Yates & Ribeiro-Neto; 2011).

2. ReactJS as a Frontend Library

ReactJS is an open-source JavaScript library developed by Facebook for building interactive and dynamic user interfaces. Unlike traditional web development approaches that rely heavily on direct manipulation of the Document Object Model (DOM), ReactJS utilizes a Virtual DOM mechanism that updates only the components affected by changes in application state. This approach significantly improves rendering efficiency and application performance (Facebook; 2023).

ReactJS adopts a component-based architecture, allowing developers to divide complex interfaces into reusable and independent components. This modular structure simplifies maintenance, improves scalability, and accelerates application development processes (Banks & Porcello; 2020). The introduction of React Hooks further enhances functionality by enabling state management and lifecycle operations within functional components without requiring class-based components (Wieruch; 2022).

3. State Management in ReactJS

State management is a fundamental concept in ReactJS that controls how data changes are handled and reflected in the user interface. In search filter implementation, state management plays a crucial role in storing search keywords, selected categories, sorting preferences, and filter conditions entered by users. Whenever the state changes, ReactJS automatically updates the relevant components without reloading the entire webpage (Banks & Porcello; 2020).

Several state management approaches can be implemented in ReactJS applications, including the use of useState, useReducer, Context API, Redux, and Zustand. For medium-scale applications such as Wiki-Inf, local state management using Hooks is often sufficient to ensure efficient rendering and maintain application simplicity (Wieruch; 2022).

4. Virtual DOM and Performance Optimization

One of the primary advantages of ReactJS is its Virtual DOM technology. Virtual DOM is a lightweight representation of the actual DOM that allows ReactJS to identify changes efficiently through a reconciliation process before updating the browser interface. Instead of re-rendering the entire page, ReactJS only modifies the components that experience state changes, thereby reducing computational overhead and improving performance (Facebook; 2023).

This optimization mechanism becomes particularly important in applications involving frequent data updates and user interactions, such as search filtering systems. By minimizing unnecessary rendering operations, ReactJS ensures faster response times and smoother user experiences (Pressman & Maxim; 2020).

5. Information Retrieval and User Experience

Information retrieval refers to the process of obtaining relevant information from a collection of data based on user queries or search criteria. The effectiveness of information retrieval systems is generally measured by factors such as precision, recall, response time, and user satisfaction. Efficient filtering mechanisms improve these metrics by reducing irrelevant results and facilitating easier access to desired information (Manning, Raghavan, & Schütze; 2008).

User experience is another critical factor influencing the success of web applications. Responsive interfaces, real-time updates, and intuitive interactions contribute positively to users' perceptions of application quality. Therefore, integrating optimized search filters with modern frontend technologies such as ReactJS can significantly improve usability and engagement within information platforms such as Wiki-Inf (Nielsen; 2012).

6. Wiki-Inf as an Information Management Platform

Wiki-Inf is an information and knowledge management platform that facilitates the storage, organization, and dissemination of information among users. As the volume of stored information grows, the need for efficient search and filtering capabilities becomes increasingly important. An optimized search system allows users to locate relevant information quickly and supports the overall effectiveness of the platform in managing digital knowledge resources (Shneiderman; 2016).

The integration of ReactJS into Wiki-Inf's search architecture offers opportunities to improve responsiveness, scalability, and maintainability. Through dynamic filtering and

efficient rendering mechanisms, the platform can provide better performance while maintaining a high-quality user experience.

C. Research Methodology

This study employed a Research and Development (R&D) approach to optimize the search filter functionality in the Wiki-Inf platform using ReactJS technology. The research process consisted of several stages, namely problem identification, system requirement analysis, search filter design, implementation, testing, and evaluation. The object of this study was the search feature in Wiki-Inf, particularly the filtering mechanism based on keywords, categories, and other search criteria. Data collection techniques included observation of the existing system, literature review of previous studies related to ReactJS and information retrieval systems, and performance testing before and after optimization implementation (Pressman & Maxim; 2020).

The optimization process utilized several ReactJS features, including state management through Hooks, component reusability, and Virtual DOM rendering to improve application responsiveness and reduce unnecessary rendering processes. The effectiveness of the implementation was evaluated based on response time, rendering efficiency, and user interaction quality during search activities. Data analysis was conducted using descriptive comparative analysis by comparing system performance before and after the implementation of the optimized search filter to determine the impact of ReactJS on improving search performance in Wiki-Inf (Banks & Porcello; 2020).

D. Results and Discussion

The implementation of search filter optimization in Wiki-Inf using ReactJS resulted in significant improvements in the efficiency of information retrieval and overall application responsiveness. Before the optimization process was carried out, the search mechanism relied heavily on repeated rendering and complete page updates whenever users entered new keywords or modified filter criteria. This condition caused noticeable delays, particularly when the system processed large datasets or multiple filtering parameters simultaneously. Users experienced slower search responses and less fluid interactions, which negatively affected usability and productivity. The optimization effort therefore focused on reducing rendering overhead and

improving the speed of search result generation through a more efficient frontend architecture (Pressman & Maxim; 2020).

One of the primary improvements involved the implementation of ReactJS state management using Hooks such as useState and useEffect. The use of these features enabled the application to manage filter values dynamically and synchronize interface updates with user input in real time. Whenever users entered search terms or selected categories, only the affected components were updated instead of refreshing the entire application interface. This mechanism substantially reduced processing time and improved the responsiveness of the search functionality. The ability to manage application state efficiently has become one of the key advantages of ReactJS in developing modern interactive web applications (Banks & Porcello; 2020).

The optimization process also benefited significantly from the implementation of the Virtual DOM mechanism provided by ReactJS. Unlike conventional DOM manipulation methods that update the entire page structure whenever data changes occur, ReactJS compares changes between the current and previous virtual representations of the interface and updates only the necessary elements. This reconciliation process minimizes computational overhead and reduces browser workload during filtering operations. As a result, users can obtain search results much faster even when interacting with extensive information collections stored in Wiki-Inf (Facebook; 2023).

Another important aspect of the optimization involved the development of reusable search filter components. Previously, similar filtering logic had to be repeated across different sections of the application, increasing maintenance complexity and code redundancy. By adopting a component-based architecture, the filtering functionality could be encapsulated into independent and reusable modules that can be deployed throughout the system whenever required. This approach not only simplified application maintenance but also improved scalability and consistency across the user interface. Component reusability is widely recognized as one of the fundamental strengths of ReactJS in large-scale web application development (Wieruch; 2022).

The implementation of memoization techniques such as React.memo, useMemo, and useCallback further contributed to performance enhancement. These techniques prevented unnecessary component rendering and recalculation of filter results when the underlying data remained unchanged. During testing, it was observed that repeated searches using similar

parameters consumed fewer system resources and generated faster responses compared to the previous implementation. The reduction in redundant processing improved browser performance and contributed to a smoother user experience during prolonged usage sessions (Banks & Porcello; 2020).

Performance testing was conducted by comparing the behavior of the search feature before and after optimization. Several indicators were used, including response time, rendering frequency, memory consumption, and user interaction smoothness. The results demonstrated measurable improvements in all evaluation categories. Search response times decreased considerably, while rendering operations became more efficient due to the selective update strategy implemented by ReactJS. Furthermore, browser memory utilization became more stable, indicating improved resource management during search activities involving multiple filtering criteria (Nielsen; 2012).

From the perspective of information retrieval, the optimized search filter contributed to higher levels of precision and relevance in search results. Users were able to combine keywords with categories and other filtering parameters to narrow down information more effectively. This capability reduced the amount of irrelevant content displayed and enabled users to locate desired information more quickly. The findings support the argument that advanced filtering mechanisms play an essential role in improving the effectiveness of information retrieval systems and enhancing user productivity in digital information environments (Hearst; 2009).

Overall, the results of this study indicate that ReactJS provides an effective and scalable solution for optimizing search filter functionality in web-based information systems such as Wiki-Inf. The integration of state management, Virtual DOM technology, reusable components, and memoization techniques successfully improved application responsiveness, reduced unnecessary rendering processes, and enhanced the overall quality of user interactions. These findings are consistent with previous research emphasizing the importance of frontend optimization in modern web development and demonstrate that ReactJS can serve as a reliable framework for developing high-performance information retrieval applications in increasingly data-intensive environments (Pressman & Maxim; 2020).

E. Conclusion and Suggestions

The results of this study indicate that the implementation of search filter optimization using ReactJS significantly improves the performance and responsiveness of the Wiki-Inf platform. By utilizing ReactJS features such as state management through Hooks, Virtual DOM technology, reusable components, and memoization techniques, the application was able to reduce unnecessary rendering processes and provide faster search responses. The optimized filtering mechanism enabled users to retrieve relevant information more efficiently and interact with the system in a smoother and more intuitive manner. These findings demonstrate that ReactJS is an effective solution for developing high-performance search functionalities in modern web-based information systems.

The optimization process contributed positively to information retrieval effectiveness by improving search precision and reducing the display of irrelevant results. The ability to combine multiple filtering criteria allowed users to narrow down search results according to their needs, thereby enhancing usability and overall user satisfaction. The study also confirmed that frontend optimization plays an important role in supporting application scalability and maintaining stable performance when handling increasing volumes of data and user interactions.

Based on the findings of this research, future studies are recommended to explore the integration of more advanced technologies such as server-side filtering, Elasticsearch implementation, artificial intelligence-based recommendation systems, and machine learning algorithms for personalized search experiences. In addition, further performance evaluations involving larger datasets and real user testing environments are necessary to provide a more comprehensive understanding of the effectiveness of search optimization strategies in large-scale information management platforms.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern Information Retrieval: The Concepts and Technology Behind Search* (2nd ed.). Harlow: Addison-Wesley.

Banks, A., & Porcello, E. (2020). *Learning React: Modern Patterns for Developing React Apps* (2nd ed.). Sebastopol, CA: O'Reilly Media.

Facebook. (2023). *React Documentation*. React Team.

Hearst, M. A. (2009). *Search User Interfaces*. Cambridge: Cambridge University Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

Nielsen, J. (2012). *Usability 101: Introduction to Usability*. Nielsen Norman Group.

Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). New York: McGraw-Hill Education.

Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th ed.). Boston: Pearson Education.

Sommerville, I. (2016). *Software Engineering* (10th ed.). Boston: Pearson Education Limited.

Wieruch, R. (2022). *The Road to React: Your Journey to Master Plain Yet Pragmatic React.js*. Self-Published.

Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-Centered Design. In W. Bainbridge (Ed.), *Encyclopedia of Human-Computer Interaction*. Thousand Oaks, CA: Sage Publications.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston: Addison-Wesley.

Krug, S. (2014). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability* (3rd ed.). Berkeley, CA: New Riders.

Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide*. Boston: Addison-Wesley Professional.

Newman, S. (2021). *Building Microservices* (2nd ed.). Sebastopol, CA: O'Reilly Media.

Richards, M., & Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. Sebastopol, CA: O'Reilly Media.

Rubin, J., & Chisnell, D. (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests* (2nd ed.). Indianapolis: Wiley Publishing.

Tidwell, J., Brewer, C., & Valencia, A. (2020). *Designing Interfaces: Patterns for Effective Interaction Design* (3rd ed.). Sebastopol, CA: O'Reilly Media.