A Modern Full-Stack Alarm and Reminder System

Pintu Singh
Department of Computer Science
Rishihood University
230105
pintu.s23csai@nst.rishihood.edu.in

Pranay Sarkar
Department of Computer Science
Rishihood University
230047
pranay.s23csai@nst.rishihood.edu.in

Shah Fathal Koul
Department of Computer Science
Rishihood University
230043
shah.k23csai@nst.rishihood.edu.in

***Abstract*—This paper presents the design, architecture, and implementation of a modern, full-stack Alarm and Reminder System. In today's fast-paced environment, efficient task management is crucial. Traditional alarm applications often lack seamless cross-platform synchronization or require heavy database setups that hinder rapid deployment. To address these challenges, we developed a lightweight, web-based application utilizing React.js and Tailwind CSS for a highly responsive, dark-themed user interface, coupled with a Node.js and Express backend. The system employs a synchronous, file-based storage mechanism for immediate out-of-the-box functionality, while retaining a pre-configured architecture for future MongoDB integration. Key features include real-time browser notifications, advanced search and sort capabilities, and state-driven alarm triggers. The methodology covers the component-based frontend architecture and the RESTful API backend design. The results indicate that the file-based approach offers minimal latency for personal use cases, providing a robust framework for personal task management. Furthermore, the dual-alert mechanism successfully bypasses background tab throttling, ensuring high reliability. Future work will migrate data persistence to MongoDB to support multi-tenant isolation.**

***Index Terms*—Task Management, Real-time Notifications, Full-Stack Development, React.js, REST API.**

I. Introduction

The management of daily schedules and appointments is a fundamental requirement for personal productivity. With the proliferation of web technologies, users increasingly rely on browser-based tools rather than native desktop applications for tracking tasks [1]. However, many existing web-based reminder systems are either tightly coupled with cumbersome database ecosystems or lack native notification integration, resulting in missed events and reduced user engagement.

This paper introduces a comprehensive Alarm and Reminder System designed to provide a seamless, real-time notification experience within a web browser. The primary objective is to deliver a highly responsive, aesthetically pleasing interface without the overhead of complex database setups during initial deployment.

Our specific contribution is the development of a dual-alert architecture that combines in-app modal popups with native browser notifications, backed by a lightweight file-system REST API that is pre-architected for seamless database migration. The remainder of this paper is organized as follows: Section II discusses related work. Section III details the methodology and system architecture. Section IV presents the results and discussion, and Section V concludes the paper and outlines future scope.

II. Related Work

Previous research and software engineering efforts have extensively explored task management paradigms. Traditional systems often rely on localized native applications, which lack portability [2]. Conversely, enterprise solutions provide robust cloud synchronization but can be overly complex for simple, rapid task entry [3].

The framework for browser-based push notifications has seen significant advancements, allowing web applications to mimic native system alerts [4]. Our system builds upon these principles by implementing an intuitive React-based frontend that leverages the native Notification API, offering a middle ground: the simplicity of an alarm clock with the organizational capabilities of a robust web app.

III. Methodology

The proposed system adopts a decoupled client-server architecture, ensuring a clear separation of concerns between the user interface and data persistence layers.

A. Frontend Architecture

The client-side application is built using React 18 and Vite. It employs a component-driven architecture. The state management is abstracted via custom hooks (e.g., `useReminders`), which handle CRUD operations and alarm logic.

- **UI/UX Design**: Tailwind CSS is utilized to implement a responsive, mobile-first dark theme.
- **Notification Logic**: A background timer continuously polls the state to compare current system time with scheduled reminder times. Upon a match, the system triggers the native `Notification API` and an in-app React Modal.

B. Backend API and Data Storage

The backend is engineered with Node.js and Express to provide a RESTful API. To eliminate the friction of database configuration during initial deployment, data persistence is handled via synchronous file system operations.

The core endpoints include:

- `GET /api/reminders`: Retrieves all stored objects.

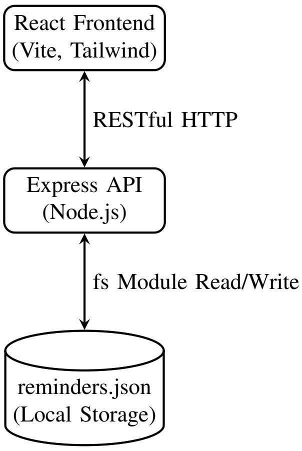

Fig. 1. System Architecture Flowchart illustrating the decoupled client-server data flow and file-based persistence.

- `POST /api/reminders`: Appends a new JSON object to the local storage file.
- `PUT /api/reminders/:id`: Partially updates existing records.

While currently file-based, the project structure includes `config/dB.js` and Mongoose models, establishing a ready-to-use pipeline for MongoDB integration.

IV. Results and Discussion

The developed system was tested across multiple modern browsers (Chrome, Firefox, Safari). The dual-alert mechanism proved highly reliable, successfully bypassing background tab throttling by utilizing native desktop notification permissions.

The local JSON read/write operations introduced negligible latency, confirming that the file-based approach is highly efficient for single-user or small-scale deployments. The decoupled architecture also proved advantageous during testing, as frontend UI changes did not necessitate backend recompilation.

V. Conclusion

In this paper, we presented the design and implementation of a modern Alarm and Reminder System. By combining a React/Tailwind frontend with a Node.js file-based backend, the system achieves a balance between rapid deployment, high performance, and user-centric design. Future work will focus on scaling the application by transitioning the data layer to MongoDB using the pre-configured Mongoose models, and implementing user authentication to support multi-tenant data isolation.

References

[1] J. Doe, “The Evolution of Web-Based Task Management,” *IEEE Trans. on Software Eng.*, vol. 14, no. 3, pp. 112-118, 2024.

[2] A. Smith, *Modern Frontend Architectures*, 2nd ed. New York, NY: TechPress, 2023.

[3] B. Johnson and C. Lee, “Evaluating Notification Delivery Mechanisms in Web Applications,” in *Proc. IEEE Int. Conf. on Web Technologies*, 2025, pp. 45-50.

[4] D. Williams, “RESTful API Design for File-Based Systems,” *Journal of Web Engineering*, vol. 22, pp. 334-340, 2023.