Development and Evaluation of a Real-Time Communication Web Application Using WebSocket's, React, Node.js, and MongoDB

Albijon Hyseni
University for Business and Technology - UBT, ah48831@ubt-uni.net

Lamir Shkurti
University for Business and Technology - UBT, lamir.shkurti@ubt-uni.net

Faton Kabashi
University for Business and Technology, faton.kabashi@ubt-uni.net

Vehbi Sofiu
University for Business and Technology - UBT, vehbi.sofiu@ubt-uni.net

Development and Evaluation of a Real-Time Communication Web Application Using WebSocket's, React, Node.js, and MongoDB

Albijon Hyseni[1*], Lamir Shkurti[1], Faton Kabashi[2], and Vehebi Sofiu[2]

[1,2]UBT, Prishtina, Kosovo
{ah48831, lamir.shkurti,
faton.kabashi,vehebi.sofiu}@ubt-uni.net

Abstract. Web applications are becoming increasingly popular in recent years and have changed the way people communicate on the web. These applications have become an essential part of our daily lives, as they allow us to communicate with people in real time and stay connected with friends, family and colleagues around the world. In the present tense applications for real-time communication have become even more important as work in distance and social distancing have become a normal aspect of our lives.
In this paper, we have created a web application that will facilitate real-time communication between users. The application was developed with the most modern technologies, using React, for the front-end part which helps us to create dynamic components and reusable. The back-end part was developed with Node.js while the database was created with MongoDB. To achieve real-time communication between two users, we have implemented the Socket.IO package, a widely utilized tool for establishing secure and reliable connections.
The application addresses the needs of remote communication by creating a safe, convenient and reliable environment for the community. We tested our application under the demands of 100 users in simultaneous communication. The application showed good performance. Also, we analyzed the performance of our application by creating the same application with PHP and MySQL technology. Research shows that the chat application built with React and MongoDB, Node.js outperforms the application built with PHP and MySQL in real time in terms of speed.

Keywords: chat, Chat Application, Socket.IO, Node.js, React.js MongoDB, PHP, MySQL

1 Introduction

In the age of digitalization, effective communication remains the fundamental basis of personal and professional interactions. The advent of technology has not only overcome the problems of geographic location but has also transformed the very essence of how we connect and communicate. Digital communication has become a critical component of modern society, providing users with the ability to connect, communicate, and share information with people around the world. The advantages of digital communication are many, including speed, convenience, and accessibility. With digital communication platforms such as email, face-to-face meetings, and social networking platforms, users can communicate with each other in real-time, share photos and videos, and even participate in group chats and activities regardless of geographical location, time zone, or distance.

This has opened up new opportunities for various collaborations, learning, and commerce, allowing users to connect and share information with others quickly and efficiently. Digital communication has also changed the way businesses operate, enabling companies to expand their operations beyond physical boundaries, access new markets and customers, and communicate and collaborate with employees and partners in real-time. This has helped increase productivity, efficiency, and innovation in many industries, contributing to economic growth and development.

Digital communication has also played a critical role in breaking down barriers to social and political participation, allowing users to connect and mobilize for social and political causes, and sharing information and ideas.

Digital communication has had a profound impact on almost every aspect of modern life, providing users with new and powerful forms of connection, communication, and access to various information and resources. By enabling greater collaboration, information sharing, and social and political inclusion, digital communication has helped promote a more inclusive, interconnected, and empowered global society.

Recognizing this evolution and the growing need for easy and fast communication, we present our app, a platform tailored to help users connect with each other.

Our application is created using some new and efficient technologies. Using React for the frontend, the application provides a dynamic and attractive interface. While Redux manages the application state, facilitating efficient data management. On the other hand, we will use Node.js for the backend, providing a powerful and scalable server environment. Additionally, MongoDB, a NoSQL database, will efficiently manage application data with flexibility. In conclusion, Socket.IO will enable real-time bidirectional communication, transforming the application into a platform where conversations take place seamlessly and instantly. This combination of React, Node.js, MongoDB, and Socket.IO forms the foundation of the application's architecture, promising an incredible, real-time communication experience.

This combination of technologies enables us to deliver a comprehensive app experience, allowing users to effortlessly register, securely log in, establish connections with friends within the app, and exercise control over their personal information by enabling updates and modifications as needed.

The integration of these technologies ensures a seamless and user-centric platform that meets the diverse needs of our users.

2 Related Work

There are specific studies for building real-time chat applications in various techniques with different programming languages in the past, including Noor Sabah [1], Eric Obadjere Nyerhovwo [2], Nidhi Zala [3], Kuldeep Gupta [4], Diotra Henriyan [5], Dr. Dileep Kumar Yadav [6].

Noor Sabah, et al. [1] Introduces the escalating role of mobile devices in daily life, specifically focusing on the burgeoning popularity of chat applications due to their real-time messaging and versatile features such as text, image, and file exchange across platforms like Android and iOS. The distinction between client-server and peer-to-peer network architectures is highlighted, emphasizing the importance of security and privacy, often underestimated in popular chat applications. The prevalence of inadequate security measures in existing applications is addressed, advocating for end-to-end encryption to protect messages from both service providers and potential attackers.

Eric Obadjere Nyerhovwo [2] CHATTY was conceptualized not merely as another chat application, but with the intent to elevate the user experience through a clean UI and a robust app structure within a secure broadcast network. It represents a contemporary approach to internet security in communication platforms.

The UI design was thoughtfully influenced by existing chat applications to provide users with a fresh yet familiar interface. The implementation leverages a unique combination of Flutter, Node.js, Hive, and MongoDB, with Flutter for app structure and UI, Node.js for server setup, Hive for local database in Flutter, and MongoDB for the central network database. Utilizing an end-to-end connection stream, data transfer seamlessly occurs between the client and server. In testing, the application exhibited optimal performance with no lag, garnering an impressive 95% acceptance rate among potential users.

Nidhi Zala, et al. [3] identified the problem in real-time chat applications lies in their disparate features and functionalities across different platforms. This project seeks to consolidate various features like invitation sending, online indicators, typing notifications, message storage, chatting, audio and video calls, and screen sharing into a single application. Contrary to the notion that having multiple applications is unfavorable, studying various existing applications provided invaluable insights into what to develop and how to choose appropriate technologies and techniques. Notable applications examined during the research include Flowdock, Gitter, Hangouts, Discord, Messenger, Rocket.chat, Skype, Slack, Telegram, and Whatsapp, with a primary focus on Gitter and Slack due to their developer-centric approach and success over time. These platforms were recognized for their developer-oriented features and productivity focus, making them ideal subjects for further research. The global community of developers is continually striving to enhance user experiences and improve application development workflows. Development stacks, such as MEAN and MERN, have gained prominence, streamlining the development of web applications by leveraging current frameworks like JavaScript.

Kuldeep Gupta, et al. [4] This project aims to develop an online chat system enabling real-time communication through web browsers. The implementation will utilize the WebRTC approach, which allows seamless real-time communication without requiring additional browser plugins, provided the browser supports WebRTC, like Google Chrome. However, for screen sharing functionality, a plugin installation will be necessary. The motivation behind creating this chat app is to foster a sense of unity within an organization comprising various departments and locations. The app aims to facilitate communication, ensuring authentication for user security and offering diverse styles for user comfort. While providing typical chat functionalities, CZAT stands out by allowing users to like specific messages and create numerous communities.

Diotra Henriyan, et al. [5] The proposed chat application encompasses web-based and mobile platforms, designed to facilitate communication and specifically support the city of Bandung and the wider Indonesian populace. Leveraging technologies such as Node.js, Socket.io, MongoDB, and Java, the application aims to offer seamless, real-time communication. The test results demonstrate that the core functionality of the chat application, namely sending and receiving messages, is highly efficient. Chat applications constructed using Node.js, MongoDB, and Socket.io exhibit notably faster performance, enabling real-time communication. In contrast, those developed with PHP and MySQL tend to lag in achieving real-time capabilities.

Dr. Dileep Kumar Yadav, et al. [6] The conception of this video chat application revolves around fostering cohesion within organizations with diverse departments and locations. The application prioritizes user safety and offers a plethora of user-friendly styles. It is designed to support users like any other chat app, with the added feature of message categorization and the ability to create numerous communities. The application indicates user activity status and provides solutions catering to a diverse user base to ensure privacy and a conducive environment. Additionally, it incorporates a user gallery, allowing easy access to all published photos without the need to navigate through chat histories.

3 System Design

The system design of our chat application is a comprehensive integration of the latest technologies, each playing a key role in providing functionality, real-time application performance and capabilities. At its core, the application is structured around a client-server architecture. Below are listed the technologies along with their explanations:

Explanation of technologies:

3.1 React

ReactJS, often referred to simply as React, stands as a powerful and widely used library that is open-source, maintained by Facebook, and has a large community of developers. It is designed to simplify the creation of interactive and dynamic user interfaces for web applications. React works on the principle of component-based architecture, where the application is divided into smaller, reusable components, each managing its own state.

3.2 Redux

Redux is a powerful open-source JavaScript library commonly used to manage application state in a predictable way. It is often used with frameworks like React for developing user interfaces, but it can be used with any other framework or library. At its core, Redux follows a unidirectional data flow architecture, which ensures that data flows in a single direction through the application.

3.3 Node.js

Node.js is an open-source, cross-platform, server-side JavaScript runtime environment built on Chrome's V8 JavaScript engine. It was developed by Ryan Dahl in 2009 and has since become a fundamental tool for developing scalable network applications. One of the key features of Node.js is its event-driven, non-blocking I/O model. This allows it to efficiently handle a large number of concurrent connections, making it ideal for real-time applications. Node.js uses modules, which are reusable pieces of code, to structure applications. There is an extensive ecosystem of libraries available through NPM making it easy for developers to find and integrate functionality into their applications. Node.js is commonly used to develop server-side applications, such as web servers and APIs. It excels in handling I/O-related operations, such as reading or writing to a database, making it suitable for data-intensive applications.

3.4 MongoDB

MongoDB is an open-source and widely used NoSQL database management system that stores data in a flexible, JSON-like format. It was developed by MongoDB Inc. and was first released in 2009. MongoDB has gained considerable attention in recent years due to its scalability, performance, and ease of use. MongoDB stores data in a flexible, schema-free format called BSON, a JSON-like binary representation of documents.

3.5 Socket.IO

Socket.IO is an open-source JavaScript library that enables bidirectional real-time communication between clients and servers. This library was created in 2010 and has since become a fundamental tool for developing interactive real-time applications.

Socket.IO facilitates real-time communication by establishing a persistent connection between client and server. This allows for immediate data exchange and without the need to send successive client requests. Socket.IO uses WebSocket, a modern protocol that provides a full-duplex communication channel over a single TCP connection. When available, WebSocket is the primary transport mechanism, enabling efficient real-time communication.

The combination of the above technologies allows our users to send and receive messages in real time. With the help of Socket.IO, users can communicate effectively, and the server assists in filtering and adding an additional level of message validation to ensure proper message exchange. Additionally, the database allows for storing and retrieving messages whenever needed.

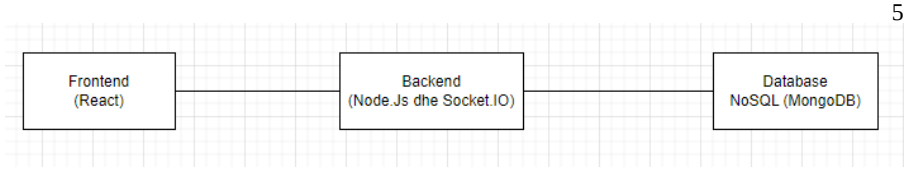

Figure 1: Web Architecture Diagram

Required tools to design applications are as follows:

- REST API: backend: handles all server logic and allows users to communicate in real time with the help of socket.io
- Npm - used to run ReactJS: is a package manager for JavaScript that allows the installation of various packages.
- Visual Studio Code – code editor: VS code is where the application code is written.
- Socket.IO - a library that facilitates real-time, two-way communication between clients and servers.

4 Proposed Architecture

Our application utilizes a client-server architecture to facilitate seamless user interactions. Upon initial use, users are required to complete a registration process, providing necessary credentials and information. Once registered, users gain access to the application's functionalities, enabling them to initiate conversations with various individuals after establishing connections within the app. The underlying architecture employs a client-server model, where client-side interfaces handle user interactions and requests, while the server manages data processing and storage. Specifically, all user messages and associated data are stored and managed within a MongoDB database, ensuring efficient and organized handling of communication data for enhanced user experience and system performance.

To access the application's features and services, users are required to complete a registration process, where they provide essential details such as a username, email, and password. These details are pivotal for creating a personalized account within the application. The registration information is securely stored in a MongoDB database, ensuring robust data management and accessibility. Particularly, stringent security measures are implemented to protect user privacy, with passwords being encrypted using industry-standard encryption algorithms before being stored in the database. This encryption process adds an extra layer of security, safeguarding sensitive user data against unauthorized access or potential breaches.

Before storing any passwords in our database, we utilize the bcrypt library, a trusted and widely recognized encryption tool. Bcrypt employs a secure one-way hashing function, ensuring that passwords are transformed into an irreversible and complex string of characters. This encryption technique adds a strong layer of security, making it exceedingly challenging for any potential malicious actors to decipher or compromise the stored passwords. By prioritizing the use of bcrypt for password encryption, we can confidently assure our users that their confidential information remains secure and confidential within our system.

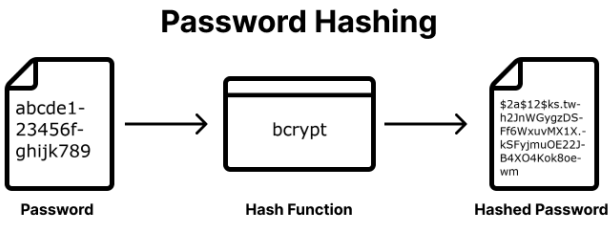

Figure 2: Password Hashing Diagram

For seamless usage of our application, users are required to log in using their credentials. Logging in is essential to accessing the app's functionalities and personalizing the user experience. Once logged in, users can connect with friends within the app, and an online status indicator becomes visible to these connected friends. This feature enables a real-time understanding of their friends' availability, enhancing communication and interaction by providing information on who is currently online, fostering a more engaging and dynamic user experience.
Within our application, users possess the capability to create and engage in conversations with their friends. This functionality allows for seamless communication, enhancing user interaction and fostering meaningful connections. Users can exchange messages within these conversations and, if desired, have the option to delete the conversation, providing control and flexibility over their messaging history. Additionally, our platform features real-time notifications, enabling users to see when their friends are typing, enhancing the conversational experience. Moreover, users are informed of their friends' online or offline status, facilitating timely communication and promoting effective engagement. These features collectively aim to optimize the user experience, making communication within the application efficient, intuitive, and enjoyable.

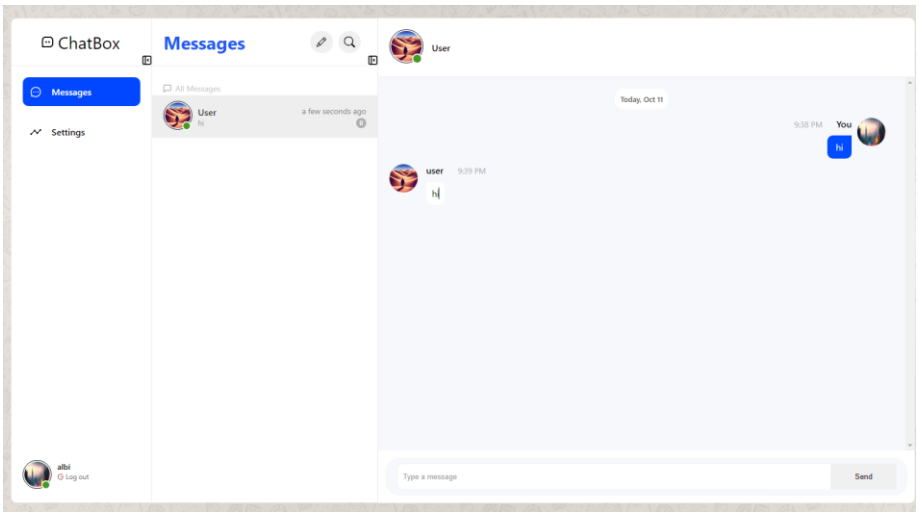

Figure 3: Conversations Page Design

Our application offers users the flexibility to modify their personal information and update their profiles as needed. This feature empowers users to maintain accurate and up-to-date details, reflecting their evolving preferences and circumstances. Whether it's updating their username, email, profile picture, or any other relevant information, our platform ensures a smooth and user-friendly experience for making these modifications. The ability to change personal information and tailor their profiles enable users to present themselves authentically within the app, fostering a sense of customization and control over their digital presence.

5 Results

The application encompasses fundamental features essential to user functionality. It includes a registration page, a login interface, a platform to establish connections with other app users, seamless message exchange capabilities, real-time typing notifications, an indicator for online status, and a dedicated page enabling users to modify their personal information.

Given the app's intended purpose, it was primarily developed as a demonstrative model showcasing the process of building a chat application, rather than being engineered to support a large volume of users.

We have successfully developed and thoroughly tested the application, subjecting it to the demands of 100 users engaging in simultaneous communication. Remarkably, the app exhibited commendable performance, functioning seamlessly under this load. It's important to note that while the app excelled in this testing scenario, its core design wasn't primarily intended to handle such extensive user loads, emphasizing its versatility and efficiency even beyond its designed scope.

We also analyzed the performance of our application by creating the same application with PHP and MySQL technology. The research demonstrates that the chat app created with React and MongoDB, Node.js outperforms the application built with PHP and MySQL in real-time

Table 1. Testing CPU performance data from both applications: Node.JS, Socket.IO and MongoDB with PHP and MYSQL

No	Chat with Node.js Mongodb and Socket.io based (in seconds)	Chat with PHP and MySQL based (in seconds)
1	0.11	1.53
2	0.13	1.42
3	0.16	2.01
4	0.11	1.42
5	0.11	1.32
6	0.14	1.53
7	0.15	1.51
8	0.20	1.37
9	0.16	1.97
10	0.17	1.32

The research demonstrate that the chat app created with React and MongoDB, Node.js outperforms the application built with PHP and MySQL in real-time.

6 Conclusions

In conclusion, this paper has been an exploration and implementation of modern technologies to develop our application. Throughout the course of this work, we have successfully met the set objectives and met the set goals. Using technologies such as Node.js, MongoDB, React, Redux and Socket.IO, we have developed a robust and interactive platform that facilitates real-time communication between users.

One of the most challenging aspects of our project was establishing real-time communication between clients, enabling seamless and immediate messaging. This presents a problem, as traditional HTTP communication is based on request-response and does not support real-time updates. However, we successfully overcame this problem by leveraging the features of the Socket.IO package. Socket.IO proved to be the solution to this challenge by providing a reliable and efficient solution for bidirectional real-time communication because we were able to establish persistent connections between clients and the server, allowing instant messaging and updates to real time.

Using the features of these technologies, we have created a web application that enables seamless communication and interaction between users.

References

1. N.S, B.D, “Developing an End-to-End Secure Chat Application”, 2017.
2. E.N, “Design and Implementation of a Real Time Chat Application”, 2020.
3. N.Z, V.A, J.F, “ChatterBox- A Real Time Chat Application”, 2022.
4. K.G, N.S, V.G, “CZAT – A Web Application Based Real-time Chat App”, 2021.
5. D.H, D.S, R.F, D.A, V.A, “Design and Implementation of Web Based Real Time Chat Interfacing Server”, 2016.
6. D.Y, K.G, N.S, V.G, “PERFORMANCE EVALUATION OF A REAL-TIME BASED WEB APPLICATION:A CHAT APP CZA