Research on Real Time Messaging Web Application

Akhil Tripathi, Raghubir, Vishwajeet Gond
Bachelor of Computer Application, Students at
Digvijay Nath P.G. College, Gorakhpur
Guided by. Kaushal Pratap Singh

1. ABSTRACT

The swift rise of the digital communication has escalated the need to have efficient and scalable real-time messaging systems. The old models of web-based communications based on request-response paradigms are usually characterized by latency and are not really real-time interactive. The current paper will describe the design and implementation of a web based real time messaging application which is used to facilitate real time communication between users. The suggested system will employ a current full-stack architecture, which combines a responsive frontend and a powerful backend to ensure a smooth exchange of messages. Persistent communication protocols are used to provide real time data transmission, which is a low latency, high- reliability protocol. The system can support multiple users at the same time without compromising the performance and responsiveness. Experimental analysis indicates that there is better time of delivery of message and resource utilization as opposed to the traditional methods.

2. INTRODUCTION

The growing use of digital communication mediums has had a great impact on how people and organizations communicate with each other. The use of real time messaging systems has been an essential part of the modern web application as it has provided the ability to communicate real time information between users who are geographically distributed. As the use of web technologies becomes more popular, there is an increasing need to have effective, scalable, and responsive communication services that can be used in the web browsers without having to use special native applications. The classic web communication is mostly based on the request-response pattern, in which the client has to request the server to keep him/her updated, again and again.

This method adds latency and inefficiencies, especially those applications which need to continuously exchange data, like chat systems. This has caused users to complain about slow delivery of messages, tremendous network overhead, and general slowness. These shortcomings underscore the necessity to enhance better mechanisms that can facilitate real-time, two-way communication between clients and servers.

The main aim of this project is to create and build a web based real-time messaging application that transcends the shortcomings of traditional models of communication. The target system will provide instant message delivery based on persistent communication methods and provide low latency and high reliability. The system will

also be scalable to support several parallel users without compromising the system. The other goal is to introduce a modular and efficient design based on the recent web technologies, which will enable easy extensions and subsequent upgrades. The system is aimed at offering a viable and efficient solution to real time communication within web-based systems through this method.

3. RELATED WORKING

A. Existing Chat Systems-

Modern communication systems have extensively applied real-time messaging, including both social media apps and enterprise collaboration systems. The popular systems like web-based chat application and instant messaging services use all sorts of tricks to facilitate communication almost in real-time. Initial applications were based on HTTP-based methods like short polling and long polling, in which the client occasionally asked the server to send updates. Although these techniques enhanced responsiveness over the conventional, static web pages, they nonetheless added latency and network overhead.
Developing web technologies, the new chat systems have turned to persistent communication protocols like WebSockets, where full-duplex communication between the client and the server is possible. This will facilitate uninterrupted data communication without the need to start a request more than once, which will significantly enhance the speed of message delivery and the efficiency of the system.
Also, most modern systems utilise scalable backend systems and distributed databases to support high numbers of simultaneous users and messages.

B. Limitations-

In spite of the considerable improvement, there are a number of setbacks to the current chat systems. In the use of polling-based methods, the method is easy to put in place, however, it is not efficient in terms of bandwidth, and server load, particularly at high user concurrency. Scalability is also an issue, even in WebSocket-based systems, when there is a large number of persistent connections, and a careful resource management and load balancing approach is needed.
Other constraints are system complexity, which can require more infrastructure and synchronization, to implement real-time communication. Moreover, data consistency, fault tolerance, message reliability issues may occur in distributed environments. There are also issues of security and privacy, including the unauthorized access and unavailability of end-to-end encryption in certain implementations. These constraints underscore the necessity of developing high-performance, highly scalable real-time messaging systems that trade-off between performance, reliability and security.

4. METHODOLOGY / SYSTEM DESIGN

A. System Architecture:

The proposed real time messaging system is based on the client-server architecture which involves front end client that tries to communicate with a central server known as the backend server so that messages can be exchanged. It is configured to provide two- way communication via WebSocket-based protocols, and enable real-time data exchange between the connected users.
The user interface, message creation and presentation are handled by the client, whereas the server handles authentication, routing messages, and persistence of data. WebSocket communication allows maintaining connections between the client and the server, which is why there is no need to repeat HTTP requests, and latency is minimized. This will provide the benefit of delivering messages immediately to the recipients.

B. Technology Stack:

The system is developed on a modern full-stack JavaScript framework, namely, the MERN (MongoDB, Express.js, React, Node.js) stack, which offers a scalable and efficient development environment.
The application is built with React on the frontend to build a dynamic and responsive user interface. Other libraries include the Axios libraries that are used to handle the HTTP requests and the Socket.IO enhances real-time communication. React Router DOM can support client-side routing, and additional libraries like React Hot Toast and icon libraries are useful in improving the user experience and interface design.
At the server-side, the system is developed in Node.js with the Express.js framework to process server-side logic and API endpoints. The database is MongoDB and the object data modelling (ODM) library is Mongoose. Authentication and security are applied through JSON Web Tokens (JWT) and password hashing with the help of the bcrypt, Request handling, configuration and environment variable control are handled by middleware like CORS, cookie-parser and dotenv, Socket. The server also has IO to handle real-time connections, and Nodemon is utilized during development to restart the server automatically.

C. Data flow:

The data flow within the system is geared to facilitate efficient and real time delivery of messages. When an user makes a message, the frontend client forwards the information to the backend server via a WebSocket connection. The message is transmitted to the server where the required authentication procedures are performed and it is then stored in the database.
After storing, the server sends the message event to the target recipient via an active WebSocket connection. The message is received immediately by the recipient of the server and real-time updates the user interface. When the receiver is offline, the message is reclaimed in the database when he/she goes online.

Also, non-real-time operations, like user authentication, chat history retrieval, and profile maintenance are made via HTTP requests. Such a hybrid solution makes sure both request-response and persistent communication models are optimally used to provide a responsive and efficient messaging system.

5. IMPLEMENTATION

A. Key features -

To make the real-time messaging application developed usable, efficient, and scalable, a number of key features are integrated in the application. The system offers secure user authentication through token-based authentication allowing users to register, log in and sustain authenticated sessions. One-to-one messaging feature enables users to communicate privately and in an organized way.
The user interface will be user-friendly and interactive to guarantee smooth inter-communication between devices. Other characteristics are live updates of messages, history of chats and notification systems that are user friendly. The system is designed to be modular and there is easy integration of new features in the future like group chats and sharing of media.

B. Message Handling:

The system is configured to be reliable and consistent in message handling. When a user sends a message, it is validated at the client side first and then sent to the server. The message is processed at the backend by authenticating the sender and determining who the recipient is. Once the message is validated, the message is inserted into the database along a structured schema that has the sender, receiver, timing and the contents of the message. This makes sure that all the communications are constantly available and can be accessed whenever needed. Moreover, the system supports the situation whereby messages are sent to offline users by saving the messages and sending them back when an offline user goes online.
Mechanisms that handle errors are also in place to deal with failed message transmissions and guarantee data integrity. This is a systematic method that will ensure that messages are not lost and will always be in sync among users.

C. Real-Time Communication Logic:

Real time communication based on WebSocket-based protocols allows the basic functionality of the system. Socket. The use of IO is to create and maintain enduring connections between clients and the server. Upon the user connecting to the application, the unique socket session is created and linked to the identity of the user.

When a message is sent the client sends an event to the server via the socket connection that has been established. The server waits to receive incoming events, takes the message and sends it immediately to the active socket session of the recipient. This communication model is event-driven, which guarantees low latency and immediate delivery of messages.

Connection events like user join, disconnect and reconnect are also handled by the system to ensure that all the clients are synchronized. Mechanisms of broadcasting and targeted delivery of messages are adopted to make sure that messages are delivered to the right recipients in an efficient manner. This application is based on this real time communication logic which provides smooth and interactive user experiences

6. RESULTS and EVALUATION

A. Performance Analysis

The proposed real-time messaging system is tested based on the criteria of latency, responsiveness, and scalability. This is made possible by the implementation of communication based on WebSockets which allows the clients to maintain persistent connections with the server, which in turn dramatically minimizes the delay in message transmission as opposed to the conventional request response paradigm. It has been observed that messages are conveyed with low latency, and this gives almost instantaneous communication among the users.

Scalability is considered by means of the effective management of simultaneous connections on the server side. The system has proven to handle a number of active users at a time without any major change in performance. Through the use of an event- driven architecture, the server is good at receiving and sending messages to and from the server making sure that the resources are optimally used. Moreover, a NoSQL database enables flexible and scalable storage of data which can accommodate more and more message data as time goes by.

B. Testing Results

Both performance and functional testing methods are used to test the system. Functional testing: This ensures that all the core features such as user authentication, sending messages, and real-time updates are working as per requirement. The application manages to keep messages in sync among various clients such that all users get synchronized updates.

Performance testing is the process of simulating a number of users to test the behaviour of systems under load. The findings indicate a high level of stability in performance where there were a steady message delivery and no considerable loss of data. Other edge cases like disconnection and reconnection of users are also effectively handled in the system to facilitate delivery of messages without missing them.

On the whole, the results of the evaluation show that the proposed system is able to provide reliable real-time communication and perform efficiently. The low latency, scalability, and robust message handling combination confirms the suitability of the adopted architecture in the contemporary web-based messaging application.

7. CONCLUSION

This paper presented the design and implementation of a web-based real-time messaging application aimed at overcoming the limitations of traditional communication models. By leveraging a modern full-stack architecture and integrating persistent communication protocols, the system enables efficient and instantaneous message exchange between users. The proposed approach combines a responsive frontend with a robust backend, ensuring seamless interaction and reliable data handling.

The key outcomes of this work demonstrate that real-time communication can be effectively achieved within web applications using event-driven architectures. The system successfully delivers low-latency messaging, maintains data consistency, and supports multiple concurrent users without significant performance degradation.

Additionally, the modular design and use of scalable technologies make the application adaptable for future enhancements.

Overall, the developed system provides a practical and efficient solution for real-time messaging in web environments, highlighting the importance of modern communication protocols in building responsive and scalable applications.

8. REFERENCES

[1] RFC 6455 WebSocket Protocol, "The WebSocket Protocol," Internet Engineering Task Force, 2011.

[2] React Documentation, Meta. [Online]. Available: https://react.dev/

[3] Node.js Documentation, OpenJS Foundation. [Online]. Available: https://nodejs.org/

[4] Express.js Documentation. [Online]. Available: https://expressjs.com/

[5] MongoDB Documentation, MongoDB Inc. [Online]. Available: https://www.mongodb.com/

[6] Socket.IO Documentation. [Online]. Available: https://socket.io/

[7] JSON Web Token, "JSON Web Token (JWT)," IETF, RFC 7519, 2015.

[8] bcrypt Documentation. [Online].

[9] A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS, 2014.

[10] D. Li and Y. Zhang, "Scalability and Performance Optimization in Real-Time Chat Applications," International Journal of Computer Networks and Communications, vol. 7, no. 6, pp. 98-107, 2020.

[11] S. P. Gupta, "Security Concerns in Real-Time Communication Systems," Journal of Information Security, vol. 18, no. 1, pp. 22-30, 2019.

[12] M. P. Papazoglou and W.-J. Van Den Heuvel, "Service-Oriented Architectures: Approaches, Technologies and Research Issues," The VLDB Journal, vol. 16, no. 3, pp. 389–415, 2007.