Web Chat Application Using MERN Stack

Shivansh Sethi
Department of School of Computing
Graphic Era Hill University, Dehradun
India

Hemesh Mahra
Department of School of Computing
Graphic Era Hill University, Dehradun
India

Abstract:- The Internet has revolutionized the way we communicate and access information, making the world more interconnected. The primary objective of this project is to create a chat application using Java multi-threading and network concepts. The application enables users to engage in private conversations or participate in public chat rooms, facilitating the exchange of messages among users. Additionally, it facilitates the sharing of various resources such as files, images, and videos. This online chat system offers enhanced reliability and security compared to traditional systems. The implementation utilizes Java, multi-threading, and client-server architecture, with a scalable design for future enhancements.

Keywords:- *REST API, WebSocket, npm (Node Package Manager), MongoDB, Express.js, React.js.*

I. INTRODUCTION

Developers worldwide are continuously striving to improve user experience and streamline application development processes. Web development stacks, such as MEAN and MERN, have emerged as popular choices for building web applications rapidly. These stacks leverage pre-existing frameworks, including JavaScript, to simplify development tasks. This paper focuses on the MERN stack, which comprises MongoDB, Express.js, React, and Node.js. By using a single programming language, JavaScript, developers can avoid syntax errors and confusion, while benefiting from the flexibility offered by the MERN stack.

II. REALTION TO EXTERNAL ENVIRONMENT

The chat application allows users to easily connect and communicate with network-connected systems. By selecting a user or system from the list, a chat form opens, enabling seamless communication through a socket-based connection.

III. FLOWCHART

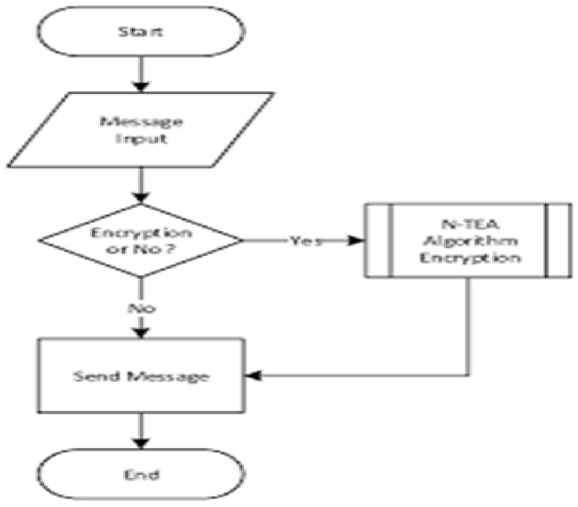


Fig. 1. Flowchart

The flowchart illustrates the step-by-step process of the chat application. It begins with the creation of a static server socket that binds to a specific host and port. The server listens for incoming requests and establishes connections with clients. Once a connection is established, the server enables simultaneous read and write operations, allowing clients to communicate and share resources. Finally, when the communication is complete, the socket is closed on both the client and server sides.

IV. METHODOLOGY

The proposed application aims to move away from a centralized system, commonly found in applications like Skype, towards a decentralized approach for improved robustness and security. The use of a distributed hash table for indexing systems allows users to create their own buddy lists without relying on a centralized database. When a user wants to communicate with another user, the latter acts as a server and authenticates the client. However, measures must be taken to prevent masquerading attacks. The proposed application incorporates the principles of decentralization and user authentication to address these issues.

V. ARCHITECTURE

A. SERVER

A server may be a computer dedicated to running a server application. Organizations have dedicated computer for server application which has to be maintained periodically and has to be monitored continuously for traffic loads would never let them go down which affects the company's revenue. Most organizations have a separate monitoring system to keep an eye over their server so that they can find their server downtime before its clients. These server computers accept clients over network connections that are requested.

The server responds back by sending responses being requested. There are many different server applications that vary based on their dedicated work. Some are involved for accepting requests and performing all dedicated works like business application servers while others are just to bypass the request like a proxy server. These server computers must have a faster Central processing unit, faster and more plentiful RAM, and bigger hard disc drive. More obvious distinctions include redundancy in power supplies, network connections, and RAID also as Modular design.

B. CLIENT

A client is a software application code or a system that requests another application that is running on dedicated machine called Server. These clients need not be connected to the server through wired communication. Wireless communication takes place in this process. Client with a network connection can send a request to the server.

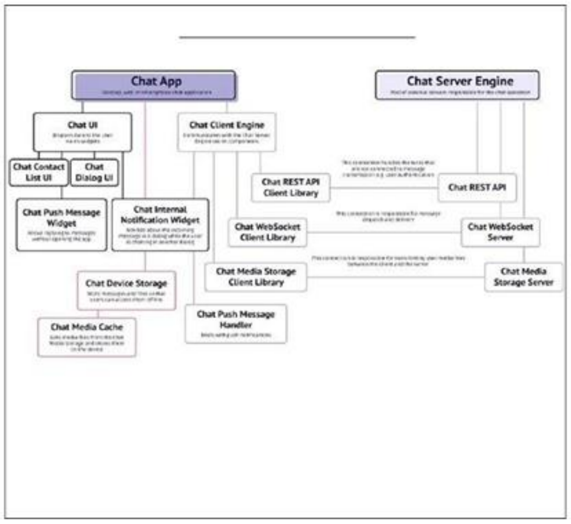


Fig. 2. Architecture

VI. CHAT APPLICATION OR CLIENT SIDE

The chat application is a crucial component of the overall chat architecture, providing a direct interface for users to interact with. It consists of two main parts:

1. **Chat Client Engine**: This component is responsible for managing communication with the Chat Server Engine. It utilizes internal modules such as the Chat REST API Client Library and the Chat Web Socket Client Library to facilitate seamless communication between the client and the server.
2. **Chat UI**: The Chat UI is responsible for presenting data to users in a user-friendly manner. It includes components such as the Chat Contact List UI, which displays the list of contacts or users available for communication, and the Chat Dialog UI, which enables users to engage in conversations and exchange messages.

By splitting the chat application into these distinct components, users can effectively communicate and interact with others while having a clear and intuitive user interface to facilitate their chat experience.

VII. DESCRIPTION

- Initially, a static server socket is created, which is then bound to a specific host and port.
- Once the server is instantiated and the socket is bound to the host and port, it starts listening for incoming connections on that port. The server is designed to accept client requests through this specific port.
- Upon starting the server, it becomes capable of accepting requests from clients.
- On the client side, a socket is instantiated to establish a connection with the server.
- To handle multiple client requests, a new server thread is created using the socket, enabling the server to accept requests from multiple clients concurrently.
- After accepting a request, the server facilitates simultaneous read and write operations, allowing the requesting clients to communicate with each other and share resources.
- Once the communication is complete, both the client and server close the socket to terminate the connection.

VIII. SCENARIOS AND OPERATIONAL CONCEPTS

The operational concepts of the application rely on user inputs. The List form presents a comprehensive list of connected systems within the network, allowing users to choose a specific name and initiate a connection through the Chat form. The Connect button establishes the connection, while the Refresh button updates the list of names. Error handling is implemented to display a message if no name is selected before clicking Connect.

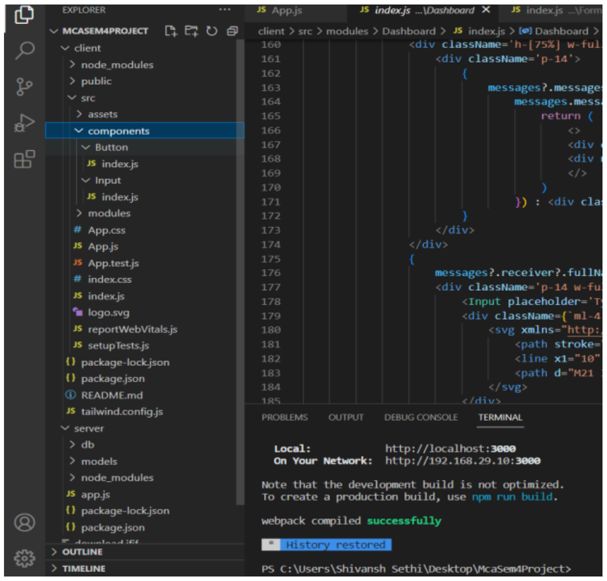
IX. DEPENDENCIES

The project relies on various third-party packages and modules installed using npm (Node Package Manager). The package.json file serves as the project's manifest, containing metadata and dependencies required for Node.js development.

X. COMPONENTS

Components are the fundamental building blocks of React applications. They can be JavaScript classes or functions that accept properties (props) and return a React element describing the UI section's appearance.

The chat application comprises several components, including the Chat Client Engine, Chat UI, and various UI elements for displaying the contact list and chat dialogs.



```
            <div className='h-[75%] w-ful
                <div className='p-14'>
                    {
                        messages?.message
                            messages.mess
                                return (
                                    <>
                                    <div
                                    <div
                                    </>
                                )
                            }) : <div cla
                    }
                </div>
            </div>
            {
                messages?.receiver?.fullN
                <div className='p-14 w-fu
                    <Input placeholder='T
                    <div className={`ml-4
                        <svg xmlns="http:
                            <path stroke=
                            <line x1="10"
                            <path d="M21
                        </svg>
                    </div>
```

```
  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.10:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully

 *  History restored

PS C:\Users\Shivansh Sethi\Desktop\McaSem4Project>
```

Fig. 3. React Components

XI. CHAT SERVER ENGINE

The Chat Server Engine is the core component responsible for message delivery and dispatch. It includes the Chat REST API, which handles tasks like user authentication and settings, and the Chat WebSocket Server, which transmits messages between users. The Chat App interacts with these components through the Chat REST API Client Library and Chat WebSocket Client Library.

XII. SYSTEM IMPLEMENTATION AND MAINTENANCE

A. *IMPLEMNTATION*

The implementation phase holds significant importance in ensuring the successful deployment of a new system. Regardless of how well-designed a system may be, its effectiveness can be compromised if the implementation process is not carried out properly. This phase involves a series of activities aimed at converting a newly developed information system into an operational system that can be used by end users.

B. *TESTING*

Testing plays a crucial role during the implementation phase of a system. It encompasses activities such as system testing, debugging computer programs, and evaluating the effectiveness of information processing procedures.

C. *TRAINING METHODS*

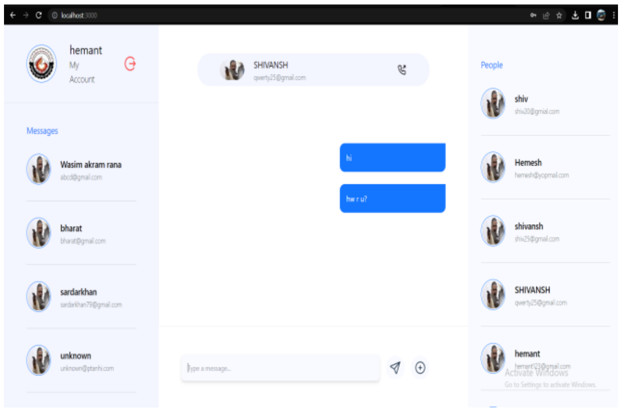
Training methods can include vendor-provided training and in-service training. Vendors often offer comprehensive educational programs as part of their services, providing courses by experienced trainers and sales personnel. Participants get hands-on experience using the system under the guidance of the trainer, who can quickly address any questions or issues.

D. *MAINTENANCE*

After the successful implementation of a system and its adoption by end users, the maintenance phase comes into play. System maintenance encompasses ongoing activities such as monitoring, evaluation, and modification of the operational information system to ensure necessary improvements are made. This phase also involves addressing user errors that may arise due to unfamiliarity with the new system and resolving any failures or issues that occur during its operation. Additionally, regular reviews or audits are conducted to verify that the system is fulfilling its intended objectives.

XIII. CONCLUSION

Continuous improvement is essential for any application. While the current focus is on text communication, there are opportunities for further enhancements. The goal is to develop a chat service web application with a high-quality user interface. Future extensions may include features such as file transfer, voice and video messages, audio and video calls, and group calling.



hi

hw r u?

Fig. 4. UI Design

REFERENCES

[1].https://www.ijeat.org/wp-content/uploads/papers/v9i5/E9578069520.pdf
[2].http://indusedu.org/pdfs/IJREISS/IJREISS_3661_55346.pdf
[3].https://thescipub.com/pdf/jcssp.2015.723.729.pdf
[4].https://www.ijrte.org/wp-content/uploads/papers/v7i5s2/ES2063017519.pdf
[5].https://core.ac.uk/download/pdf/187726106.pdf