PROJECT

ON

A Real Time Chat Application

Submitted by

RAVI KUMAR [20106107028]
NIRAJ KUMAR [20106107021]
MANEESH KUMAR [20106107017]

in partial fulfilment for the award of the degree

of

Batchelor of Technology

IN

BRANCH OF STUDY

At

Department of Information Technology

Muzaffarpur Institute of Technology, Muzaffarpur

May 2024

MUZAFFARPUR INSTITUTE OF TECHNOLOGY

MUZAFFARPUR – 842003

CERTIFICATE

This is to certify that the project work entitled “A real time chat application” and submitted by Ravi Kumar, Niraj Kumar and Maneesh Kumar towards the partial fulfillment for the award of Bachelor of Technology in Information Technology and Engineering degree at Muzaffarpur Institute of Technology, is carried out under the supervision of the department as part of 7th Semester course curriculum (Paper Name – PROJECT-I, Paper Code – 100707P) and is a Bonafede work done by them.

Date: 05 May, 2024

Place: Muzaffarpur

Prof. Y.N Sharma
Head of Department

Dept. of Information Technology

ABSTRACT

In today's digital age, communication plays a crucial role in connecting individuals across the globe. With the increasing demand for real-time messaging applications, building a single message chat application has become a popular endeavor. This abstract introduces a chat application developed using the MERN (MongoDB, Express.js, React.js, and Node.js) technology stack, highlighting its key features and functionality. The proposed chat application leverages the MERN technology stack to provide a robust and scalable solution. MongoDB, a NoSQL database, ensures efficient data storage and retrieval, enabling seamless storage of user profiles and message history. Express.js, a web application framework for Node.js, facilitates the creation of a server-side backend that handles authentication, routing, and interaction with the database. Node.js acts as the run time environment, executing JavaScript code on the server-side, enabling high-performance communication. React.js, a JavaScript library, serves as the frontend framework, providing an interactive and responsive user interface. The chat application encompasses essential features such as user registration and authentication, login & logout, and ability to send and receive messages in real time.

Index

	Abstract	ii
	Acknowledgement	iv
1.	Introduction	1
2.	Objective	1 - 2
3.	Scope	2
4.	Feasibility study	3
5.	Component of the System	4
6.	Input-Output	4 - 5
7.	Diagram	5
8.	Future Scope & Further Enhancements	6
9.	Conclusion	7

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude and whole hearted thanks to project guide Prof. Deepak Kumar Choudhary, Coordinator for his guidance throughout this work. We are very much thankful to him for his constant encouragement, support and kindness. We are also grateful to our teachers Prof. Vijay Kumar, Prof. Ashish Kumar, Prof. Shweta Kumari and Prof. Rajeev Ranjan and for their encouragement, help and support from time to time. We also wish to express our sincere thanks to Principal Dr. Mithilesh Kumar Jha for providing us wide range of opportunities, facilities and inspiration to gather professional knowledge and material without which this project could not have been completed.

Thank you.

1. INTRODUCTION

With the power of MERN technology, we've created a seamless platform for you to connect and communicate with others in a simple and efficient way. Using then MERN stack, which stands for MongoDB, Express.js, React.js, and Node.js, we have built a robust and dynamic application that allows you to exchange messages with ease. Let's take a closer look at each component:

MongoDB: Our application utilizes MongoDB, a popular NoSQL database, to store and manage the chat messages. This ensures fast and efficient retrieval of conversations, enabling real-time messaging.

Express.js: We leverage Express.js, a flexible and minimalistic web application framework for Node.js, to handle server-side operations. It simplifies the process of routing, handling requests, and managing middleware, making the application more scalable and efficient.

React.js: The frontend of our chat application is built using React.js, a powerful JavaScript library for building user interfaces. React.js allows us to create a dynamic and responsive user experience, making real-time messaging seamless and enjoyable.

2. OBJECTIVE

The objective of this project is to build a single message chat application using the MERN (MongoDB, Express.js, React.js, Node.js) stack technology. The application will allow users to send and receive messages in real-time within a single conversation.

Key Features:

• User Registration and Authentication: Users will be able to create accounts and log in to the application using their credentials. User authentication will ensure secure access to the chat functionality.

• Real-time Messaging: Users will be able to send and receive messages in real-time within a single conversation. The chat interface should update instantly when a new message is sent or received.

• Conversation History: The application should store and display the chat history, allowing users to scroll through previous messages within the conversation.

Profile Picture: Uses will be able to pick an Avatar as their profile picture and then set it as their profile picture.

• Deployment: Deploy the application to a web server or cloud platform to make it accessible to users over the internet.

By achieving these objectives, you will create a single message chat application using the MERN technology stack that provides users with a seamless and real-time messaging experience.

3. SCOPE

A single message chat application using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack can have various scopes and features. Here are some possible scopes for a single message chat application:

• User Authentication: Implement a user authentication system to allow users to create accounts, log in, and maintain their profiles.

• Real-Time Messaging: Enable real-time messaging capabilities using technologies like Socket.io or Web-Sockets to ensure instant message delivery and updates.

• User Name and Email Storage: Store name and email id of user in a MongoDB database.

• Emoji’s and Reactions: Enable users to react to messages using emoji’s or predefined reactions, enhancing the interactive experience.

• Cross-platform Compatibility: Ensure that the chat application works seamlessly across multiple platforms, including web browsers, mobile devices, and desktop application.

• Message Encryption: Implement end-to-end encryption to ensure the privacy and security of messages exchanged between users.

It's also essential to consider scalability, performance optimization, and security measures when developing your chat application.

4. FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose the organization for the amount of work. Effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A Feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development. The document provides the feasibility of the project that is being designed and lists various area that were considered very carefully during the feasibility study of this project such as Technical, Economic and operational feasibilities.

The purpose of this feasibility study is to assess the viability and potential success of developing a single message chat application using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack. The application aims to provide users with a simple and efficient way to exchange messages in real-time. The following factors will be considered in this study:

Technical Feasibility:

a. Expertise: Evaluate the availability of developers with the necessary skills and experience in MERN technology.

b. Infrastructure: Assess the required hardware and software infrastructure for development, testing, and deployment of the application.

c. Scalability: Determine if the MERN stack can handle the expected number of users and message volume efficiently.

Market Feasibility:

a. Target Audience: Identify the potential user base and their specific needs for a single message chat application.

b. Competition: Analyze the existing chat applications and their market share to determine the potential for success and competitive advantage of the proposed application.

5. COMPONENT OF THE SYSTEM

The User Interface (UI) component presents the visual elements of the chat application, including login/signup forms, chat interface, and user list, ensuring an intuitive user experience.

The Authentication component handles user authentication and authorization, allowing users to securely create accounts, log in, and maintain session persistence.

The User Management component enables users to manage their profiles, update personal information, and view other registered users within the chat application.

The Chat Management component facilitates real-time message exchanges, stores and retrieves messages from a database, and dynamically updates the chat interface for seamless communication.

The Database component stores and manages user information, messages, and chat history, ensuring data integrity and reliable data retrieval.

The Real-time Communication component establishes a bidirectional communication channel, enabling instant message delivery and real-time updates between the server and clients.

The Emoji component enhances the chat experience by allowing users to send and receive emojis, adding an element of fun and expression to their conversations.

6. INPUT & OUTPUT

The main inputs, outputs and the major function in details are:

- **INPUT**

• User can sign-in using email and password.

• User can login using the same email and password which they have used in sign-in.

• User can visit the chat window and can chat to anyone who has an account in the application.

• User can logout from the chat window.

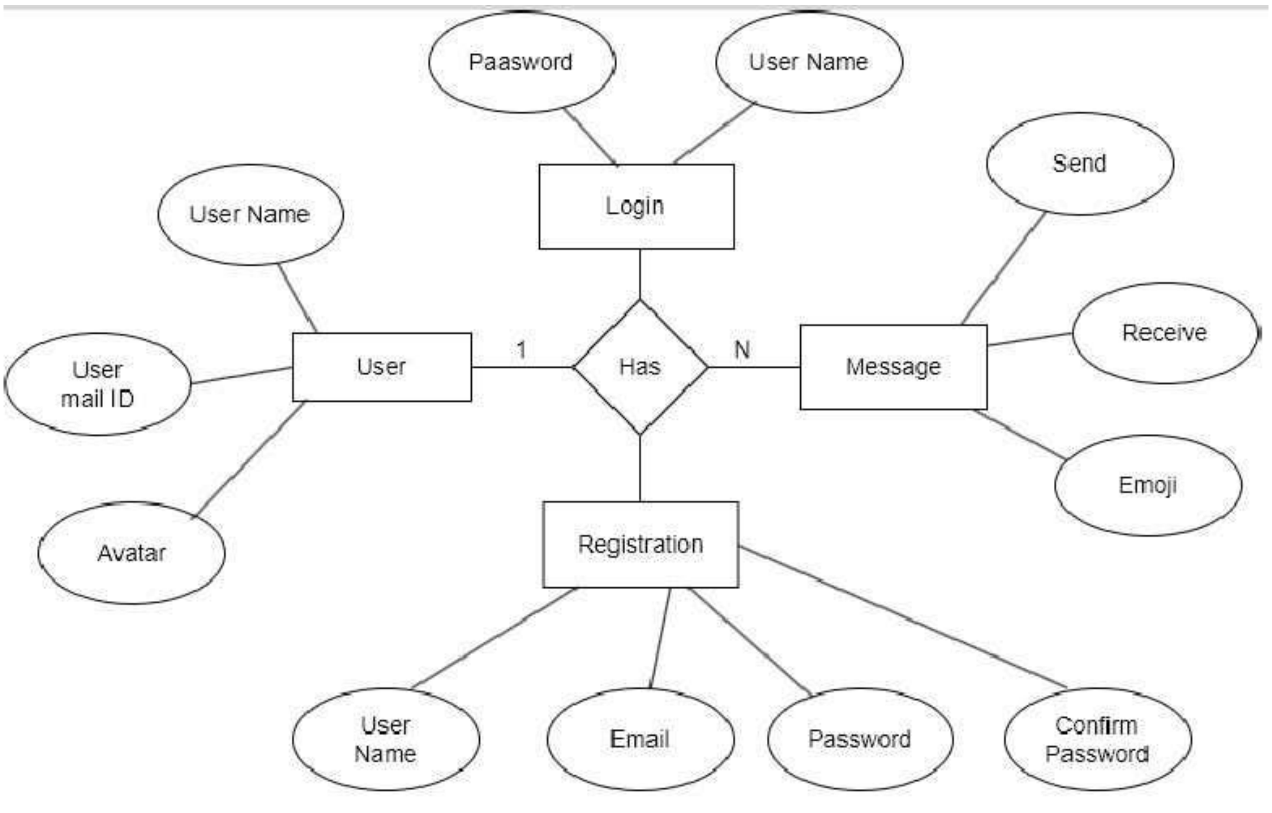
- **OUTPUT**

• User can see their other users in the application.

• User can view the messages.

• User can also interact with the other users and can send emojis.

6. DIAGRAM



ER DIAGRAM

8. FUTURE SCOPE & FURTHER ENHANCEMENTS

A single message chat application built using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack can be further enhanced and expanded in several ways to improve its functionality and user experience. Here are some future scope and further enhancements for your single message chat application:

Real-time Messaging: Implement real-time messaging using technologies like Web-Sockets or a library like Socket.IO. This will allow users to see messages instantly without needing to refresh the page.

User Authentication and Authorization: Enhance the application by implementing user authentication and authorization mechanisms. This will enable users to create accounts, log in, and have personalized chat experiences.

Emoji’s and Reactions: Enable users to express their emotions and reactions using emoji’s or predefined reactions. Implement a reaction system similar to social media platforms where users can add emoji’s or "like" messages.

Mobile Application: Consider developing a mobile application version of the chat application for both iOS and Android platforms. This can broaden the user base and provide a seamless mobile chat experience.

9. CONCLUSION

In conclusion, building a single message chat application using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack provides several benefits and advantages.

The MERN stack offers a comprehensive solution for developing a chat application that supports real-time communication. By leveraging web-sockets or similar technologies, users can instantly exchange messages.

The MERN stack supports the creation of cross-platform applications, allowing users to access the chat application from various devices and operating systems.

Using JavaScript throughout the entire stack provides a developer-friendly environment. It reduces the learning curve, promotes code consistency, and facilitates easier code maintenance.

The MERN stack benefits from a large and active community. This means developers can find extensive resources, tutorials, and libraries to assist in the development process and troubleshoot any issues.

While considering the advantages of the MERN stack, it's crucial to take into account the specific requirements and constraints of your project. Proper planning, architectural considerations, and adherence to best practices are essential for ensuring a secure, efficient, and user-friendly single message chat application.

MERN stack provides a unified and consistent development experience since JavaScript is used throughout the entire stack. This reduces the learning curve for developers and allows for easier code maintenance.