Fuzzy Systems and Soft Computing
ISSN : 1819-4362

ONLINE CHATTING APPLICATION USING MERN STACK

Rajdeep Swain Computer Science And Engineering Gandhi Institute For Technology, India
rajdeep2020@gift.edu.in ,
Ipsita Prusty Computer Science And Engineering Gandhi Institute For Technology, India
ipsita2020@gift.edu.in

ABSTRACT—
Muse-Matrix is a groundbreaking web application that redefines the blogging experience by leveraging the power of the MERN (MongoDB, Express.js, React.js, Node.js) stack. Designed with innovation, functionality, and usercentric design principles in mind, Muse-Matrix revolutionizes content creation, management, and engagement for bloggers, content creators, and readers alike.
At its core, Muse-Matrix prioritizes user experience, productivity, and security. With its sleek and intuitive interface, customizable dark mode, and responsive design, Muse- Matrix provides users with a seamless and immersive content creation platform. Advanced features such as real-time notifications, advanced search functionality, and user-friendly profile management enhance productivity and engagement, while robust authentication mechanisms and encryption techniques ensure the privacy and confidentiality of user data.
Muse-Matrix empowers users to connect, collaborate, and share ideas in a vibrant community atmosphere. Through features such as community forums, social sharing capabilities, and collaborative content creation tools, Muse-Matrix fosters a sense of belonging and encourages active participation among users.
Furthermore, Muse-Matrix serves as a catalyst for innovation in the blogging landscape, inspiring developers, entrepreneurs, and content creators to push the boundaries of what's possible. By showcasing the potential of the MERN stack and demonstrating best practices in web development, Muse-Matrix sets a new standard for excellence in content creation platforms.
In summary, Muse-Matrix represents a paradigm shift in the blogging landscape—a fusion of technology, creativity, and community that empowers users to unleash their potential and make their voices heard in the digital age. With its transformative impact and unwavering commitment to excellence, Muse-Matrix is poised to shape the future of content creation and engagement for years to come.

1. INTRODUCTION

Welcome to our Single Message Chat Application! 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 the 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.

Node.js: We rely on Node.js, a server-side JavaScript runtime environment, to power the backend of our chat application. Node.js enables event-driven, non-blocking I/O operations, resulting in a highly scalable and efficient server architecture.
We're excited to have you on board and experience the convenience and efficiency of our single message chat application. Start chatting and connecting with us.

Objectives :
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.

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 applications.
- 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.

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. xpertise: 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.

WORK FLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

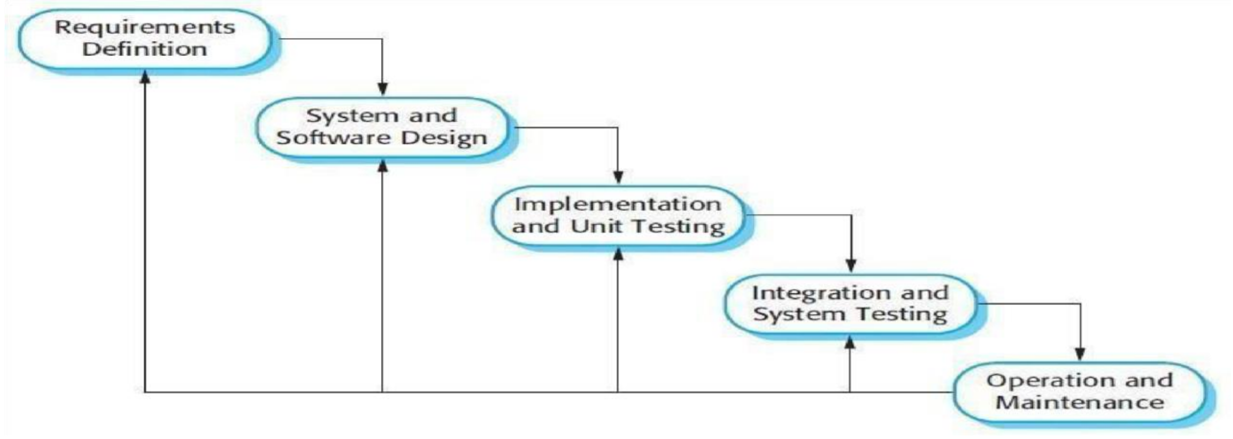
The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In waterfall model phases do not overlap.

Waterfall Model design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

Following is a diagrammatic representation of different phases of waterfall model.


Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.
- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared.
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase.
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released.

COMPONENTS

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.

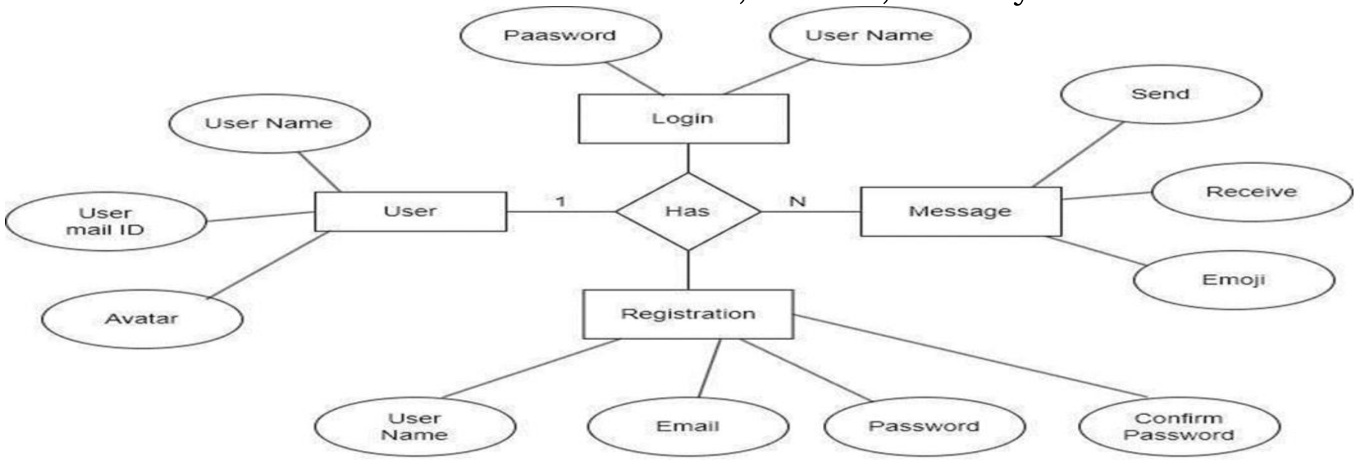
INPUT & OUTPUT

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

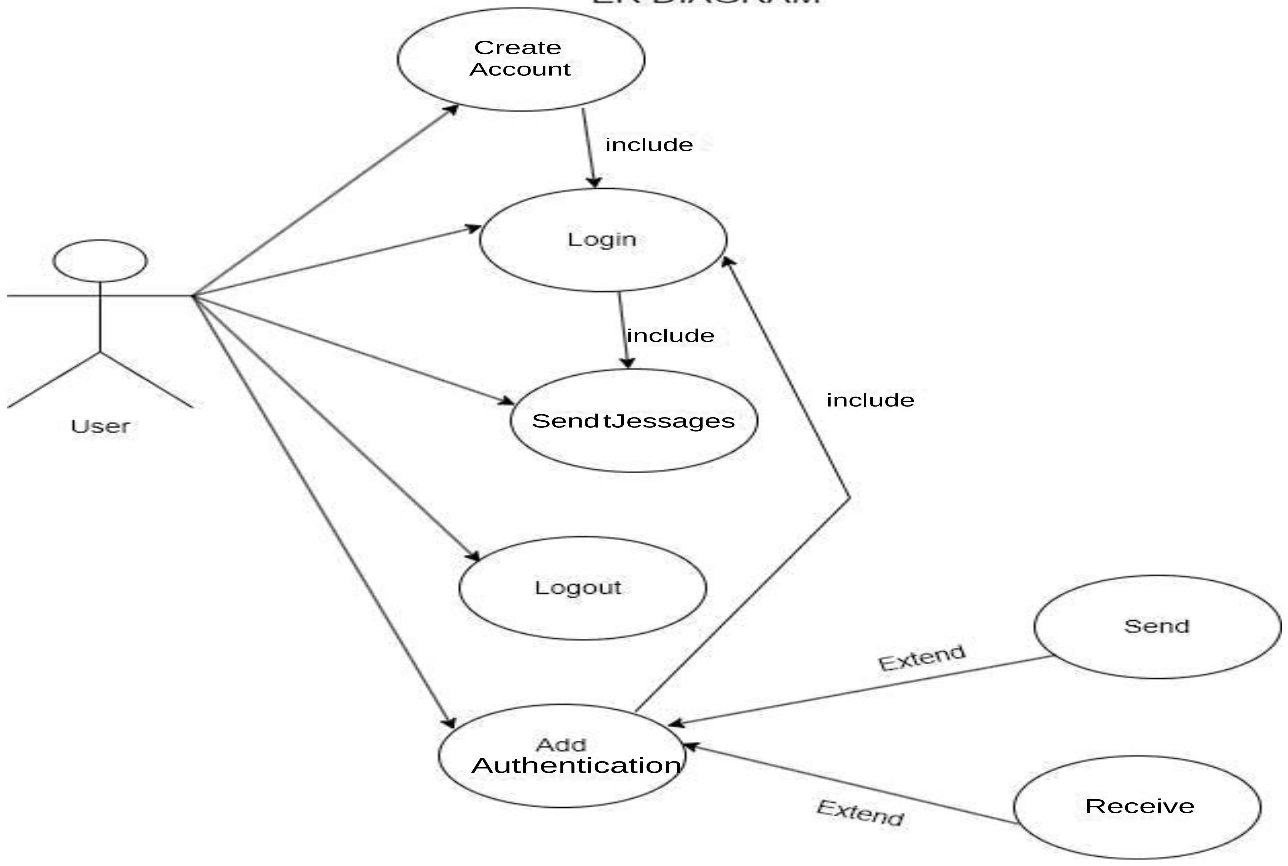
- User can sign-in using email and password.
- User can login using the same email and password which they have used in sign-in.
- User can visit the chat window and can chat to anyone who has an account in the application.
- User can logout from the chat window.

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.



ER DIAGRAM



USE CASE DIAGRAM

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 websockets 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.

REFERENCES

• React Documentation- Retrieved from https://react.dev/learn
lode.js Documentation- https://nodejs.org/en/docs *Retrieved from*
• *Express Documentation- https://expressjs.com/en/4x/api.html* Retrieved from

- *MongoDB Documentation-*[*https://docs.mongodb.com/manual*](https://docs.mongodb.com/manual) Retrieved from
- *Socket.io Documentation-* [*https://socket.io/docs/v4*](https://socket.io/docs/v4) Retrieved from
- *W3Schools- Retrieved from* [*https://www.w3schools.com*](https://www.w3schools.com)
- *Emoji Mart- Retrieved* from https://www.npmjs.com/package/emoji-mart

-Thank You

X