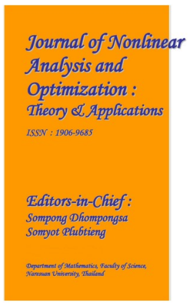
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 16, Issue. 1: 2025
ISSN : **1906-9685**


Journal of Nonlinear Analysis and Optimization : Theory & Applications
ISSN : 1906-9685
Editors-in-Chief :
Sompong Dhompongsa
Somyot Plubtieng
Department of Mathematics, Faculty of Science, Naresuan University, Thailand

CHAT APP

Santosh Kumar Sahoo
Sai Sankar Patra
Dept. of Master of Computer Science
Email ID: kumarsahoo2023@gift.edu.in
Email ID: sspatra2023@gift.edu.in

Dr. Pratyush Ranjan Mohapatra
Assistant Professor, Department of MCA, GIFT Autonomous, Bhubaneswar, BPUT, India
Email ID: pratyush@gift.edu.in

ABSTRACT :

Conversation has become an essential tool in bridging distances and fostering global connectivity, and technology plays a central role in making this possible. The advancement of web technologies and real-time communication protocols has paved the way for chat applications that are faster, more interactive, and more accessible. Our project is a practical implementation of such a system, using the MERN stack—MongoDB, Express.js, React.js, and Node.js—to build a modern, scalable, and user-friendly chat platform. The system consists of a server-side application responsible for handling real-time data exchange and a client-side interface for users to interact seamlessly. It supports features like one-on-one messaging, group chats, and media sharing, including files, images, and videos. The integration of Socket.io ensures instant communication, while MongoDB handles the storage of chat history and user data securely. With the rise of remote work, online learning, and digital communities, this application demonstrates how efficient and secure communication tools are crucial in today's interconnected world. Designed with scalability and performance in mind, this app holds potential for deployment across various sectors, including educational institutions, corporate environments, and community networks, making it a valuable contribution to the evolution of digital communication.

Keywords- Chat-App, React.js, Node.js, Express.js and MongoDB

INTRODUCTION:

In today's fast-paced digital world, developers and engineers are under increasing pressure to deliver high-quality web applications within strict deadlines. To meet these demands, many have turned to modern development stacks that streamline the process of building, deploying, and scaling applications. Among these, the MERN stack comprising MongoDB, Express.js, React.js, and Node.js has emerged as a powerful and popular choice. This stack allows developers to use JavaScript across the entire development process, eliminating the need to switch between languages for front-end and back-end tasks. This unified approach not only enhances efficiency but also minimizes errors, improves consistency, and accelerates the overall development timeline.

MERN, like its counterpart MEAN, is made up entirely of open-source components, making it accessible and widely supported by the developer community. One of the key advantages of the

MERN stack is its flexibility and adaptability, which allows developers to build everything from small-scale applications to complex, data-driven systems. React.js enhances the user
experience with dynamic and responsive interfaces, while Node.js and Express.js provide a robust backend capable of handling real-time data and high-performance networking.
MongoDB, being a NoSQL database, offers scalable and flexible data storage. Together, these components provide a comprehensive solution for developing full-stack web applications.
Using the MERN stack also aligns with current trends in web development, where single-page applications (SPAs), real-time features, and cross-platform compatibility are in high demand. With the growing reliance on cloud computing and remote access, the ability to quickly develop and maintain responsive, interactive applications has never been more important. This paper will further analyze each component of the MERN stack, explore its practical applications, and compare its performance with other development technologies to showcase its effectiveness in modern software engineering.

Explanation about MERN:
The MERN stack is a full-stack JavaScript framework used for developing modern web applications. It is composed of four main technologies: MongoDB, which functions as the database; Express.js, which serves as the backend web framework; React.js, which manages the front-end user interface; and Node.js, which provides the server-side runtime environment. Together, these components enable developers to build dynamic, scalable, and efficient web applications entirely using JavaScript, from front-end to back-end.

MongoDB:
MongoDB is a NoSQL database that stores data in a flexible, JSON-like format, making it well-suited for modern web applications. Its schema-less structure allows for easy scalability and rapid development, enabling developers to adapt to changing data requirements without complex migrations. This flexibility makes MongoDB ideal for applications that need to handle large volumes of data efficiently and require high performance and adaptability.

Express.js:
Express.js is a lightweight and flexible web application framework for Node.js that streamlines server-side development. It offers a robust set of features for building APIs and handling HTTP requests and responses efficiently. With built-in support for middleware, Express.js enables developers to manage the request-response cycle more effectively, making it easier to implement functionalities such as authentication, logging, and error handling in a modular and organized manner.
React:
React.js is a powerful JavaScript library used for building user interfaces, especially well-suited for single-page applications. It follows a component-based architecture, which encourages reusability, modular design, and easier maintenance of code. One of React's key features is its virtual DOM, which optimizes rendering by updating only the parts of the UI that have changed. This results in improved performance and a smoother user experience during dynamic content updates.

Node.js:
Node.js is a JavaScript runtime environment that enables developers to execute JavaScript code on

the server side. Built on Chrome's high-performance V8 engine, it features a non-blocking, event-driven architecture that is ideal for building scalable and efficient network applications. Node.js excels at handling multiple simultaneous connections, making it well-suited for real-time applications such as chat servers, APIs, and streaming services.

Work Of the Components Data Flow:
In a MERN stack application, the data flow begins with the client-side (React), which interacts with the server through API calls. These requests are handled by Express.js, the backend framework, which processes the input, performs necessary logic, and communicates with the MongoDB database to retrieve or store data. Once the required operations are completed, Express.js sends the appropriate responses back to the React client. Throughout this process, Node.js acts as the runtime environment, allowing JavaScript code to run on the server efficiently, enabling seamless communication between the front end and back end. This structure ensures a smooth and consistent flow of data across the application.

Development Efficiency:
Using JavaScript across the entire MERN stack streamlines the development process by creating a unified and cohesive workflow. This consistency allows developers to work more efficiently, as they can use the same language—JavaScript—for both client-side and server-side development.
As a result, developers can seamlessly switch between front-end and back-end tasks without the need to learn or adapt to different programming languages. This not only reduces context-switching but also simplifies debugging, enhances collaboration within teams, and accelerates the overall development cycle.

PROPOSED MODEL

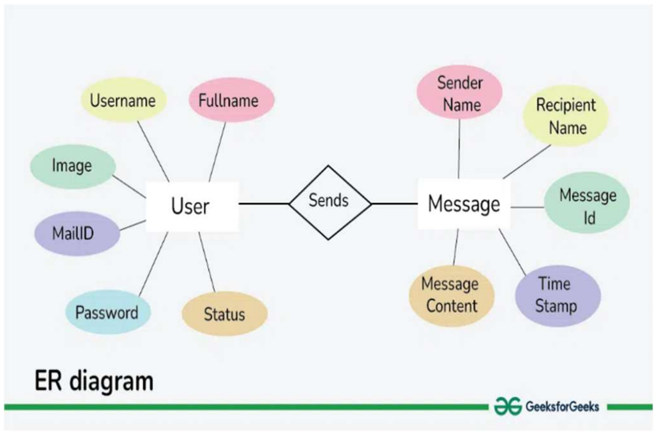


Figure 1: Proposed work

The image is a clear and structured Entity-Relationship (ER) diagram that represents the database schema of a chat application. It highlights four primary entities: **USER**, **CHAT**, **MESSAGE**, and **MSER**, each represented by yellow rectangular boxes. These entities are connected through diamond-shaped relationship indicators labeled **HAS**, illustrating how they relate to each other.
Each entity includes relevant attributes displayed in oval shapes, with different colors used for visual distinction. The **USER** entity contains attributes such as **name**, **email**, **password**, **group**, and **groupAd**, each shown in unique colors like orange, blue, and green to enhance clarity. The **CHAT**

entity includes attributes like **isGroupChat** and **content**, while the **MESSAGE** entity features **sender** and **timestamp**. The **MSER** entity is also linked to USER and MESSAGE, indicating a specific role or relationship in the system.
All attribute ovals are outlined with a thin black border, which separates them neatly from the background and improves overall readability. This ER diagram effectively presents the structural design of the chat application's database in a clean, organized, and visually appealing format.

METHODOLOGY

This image illustrates a structured development approach for building a chat application, represented as a flowchart. The process begins by formulating a Problem Statement, which defines the core issue the application aims to address. Following this, Requirements are clearly outlined to set the scope and objectives of the project. The next step involves Designing the User Interface, ensuring the application's front-end is intuitive and user-friendly. Subsequently, the Backend is Set Up to manage the server-side functionalities. With the backend in place, developers Create REST APIs to facilitate communication between the frontend and backend. Simultaneously, the Frontend is Implemented, focusing on the client-side experience. To enable instant communication, Real-Time Messaging features are established. Afterwards, the Frontend is Integrated with the Backend, connecting all system components. Notably, the flowchart contains a minor redundancy where integrate front end with backend appears twice. Finally, the entire system undergoes a Testing Phase to ensure the chat application performs reliably and meets user expectations. This methodical, step-by-step development approach ensures organized progress from concept to a fully functioning chat solution.

Problem Statement

Examine the Cleveland dataset to determine if a person is at risk for heart disease. A person's heart disease is represented by the number 1, and their absence of heart disease is represented by the number 0.

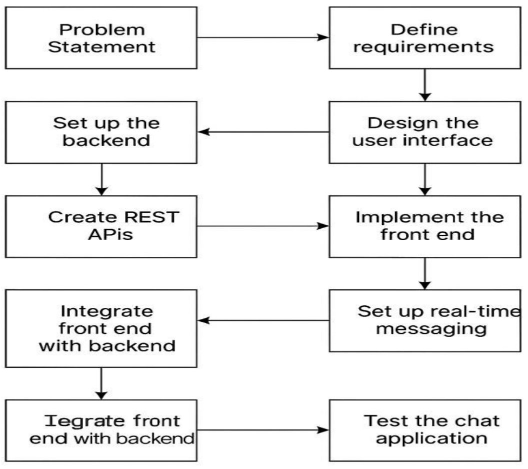

Problem Statement
Define requirements
Set up the backend
Design the user interface
Create REST APIs
Implement the front end
Integrate front end with backend
Set up real-time messaging
Iegrate front end with backend
Test the chat application

Figure 2: Development Approach

General Explanation:

A MERN stack chat application combines MongoDB, Express.js, React.js, and Node.js for seamless, full-stack development. MongoDB manages data storage, Express and Node handle server logic, and React delivers a responsive user interface. WebSockets enable real-time messaging, ensuring smooth, interactive chat experiences.

Layer-wise Role Description:
In a MERN chat app, MongoDB stores messages and users. Node.js executes backend code, Express.js builds APIs and manages routes, and React.js powers dynamic interfaces. Socket.io with Node enables instant message delivery, keeping conversations real-time without requiring manual page refreshes.

Technical Workflow Perspective:
User actions on React's interface send HTTP or WebSocket requests to Express.js, processed by Node.js. Data operations occur in MongoDB. React's dynamic rendering ensures smooth, responsive message flow, delivering uninterrupted, real-time conversations.

Development Convenience Angle:
MERN's JavaScript-only environment simplifies full-stack development, enabling teams to work efficiently without switching languages. MongoDB's flexible document model suits dynamic chat data, while Express streamlines API creation. Node manages concurrent users, and React ensures a responsive, engaging front-end chat experience.

Business Value Perspective:
The MERN stack offers businesses scalable, affordable, and maintainable solutions for real-time communication. JavaScript unification reduces operational overhead, accelerates development, and simplifies hiring. MongoDB's scalability and React's modern UI support growth, making it ideal for startups and enterprises targeting real-time markets.

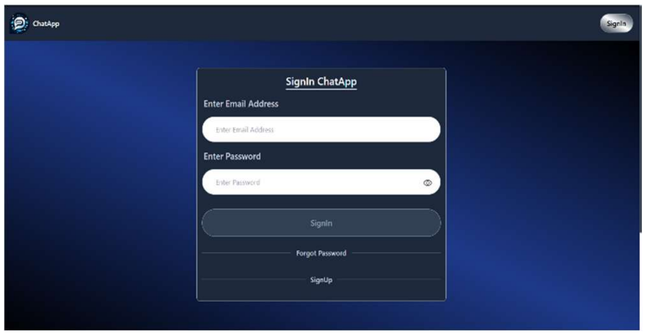
User Experience (UX) Perspective:
MERN-based chat apps offer instant message delivery, responsive interfaces, and dynamic experiences. React ensures real-time updates without reloads. MongoDB quickly retrieves data, and WebSocket maintain seamless communication. Features like typing indicators, notifications, and file sharing create a smooth, reliable chat experience.

RESULTS:
In conclusion, the development and implementation of Web- Based Real-Time Chat Application using the MERN Stack represents a groundbreaking achievement in the realm of modern communication platforms. This project successfully harnesses the power and versatility of Mongo DB, Express.js, React.js, Node.js to deliver a feature-rich, secure, and scalable solution that caters

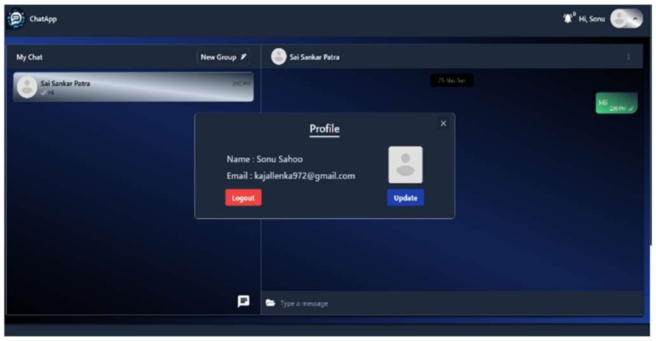
to the dynamic communication needs of today's interconnected world.
The MERN Stack's seamless integration and cohesive architecture have empowered the creation of a real-time chat experience that goes beyond conventional boundaries. The use of WebSocket technology ensures instant messaging, enabling users to engage in fluid con-versations with minimal latency. The platform's responsiveness and adaptability across various devices underscore its commitment to user convenience and accessibility. New features that we look to add:
(1) Mailing service for verifying new users and a forgotten password option.
(2) Read receipts for messages.
(3) Delete and edit messages.
(4) Attach messages or files with messages.

(5) Develop a mobile app to increase the user base.

Login-image: -

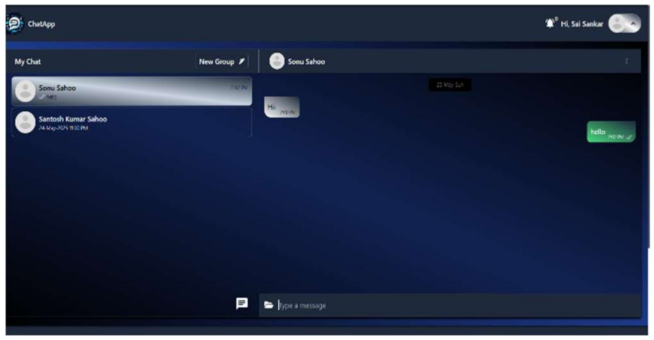


Profile: -



Profile

Name : Sonu Sahoo
Email : kajallenka972@gmail.com



CONCLUSION

The chat application delivers a modern, efficient, and highly flexible communication platform, designed with the latest web technologies to meet the evolving demands of users and organizations alike. Its primary strength lies in offering **instant messaging, real-time communication, enhanced security measures, group chat functionalities, and an intuitive user interface**. By integrating cutting-edge tools and frameworks, the system ensures reliable performance, scalability, and seamless connectivity for users across various devices and networks.

Furthermore, this application holds significant market potential, particularly for **businesses, educational institutions, and social platforms** seeking independent, customizable chat solutions. The real-time messaging system not only enhances user engagement but also improves operational efficiency in collaborative environments. With its scalable architecture and robust features, this chat application stands poised to become a preferred choice for organizations aiming to deliver **secure, private, and high-performance communication services** tailored to their unique operational needs.

FUTURE SCOPE :

Other enhancements will be involved

1. Profile Picture Updating
2. Video call
3. large size
4. Conference call
5. Voice recording will be added
6. Improving different text style and font size

REFERENCES :

[1] https://www.ijraset.com/research-paper/development-of- chat-application
[2] https://www.researchgate.net/publication/385964379_Web-based_real_time_chat_application_using_MERN_stack
[3] https://www.scribd.com/document/647840397/final-report- on-chat-application-using-mern
[4] https://www.suprsend.com/post/how-to-build-a-real-time- chat-app-with-mern-stack-and-suprsend-javascript-sdk
[5] https://github.com/RishiBakshii/mern-chat- app/blob/main/readme.md
[6] https://github.com/tsengm6h6/chat-app-client-v2