# Build a real-time chat application that stores messages in a MongoDB database. Implement user authentication and chat rooms.

Department of Computer Science

Advanced Database

System Design

Fall 2024

Instructor:

Neha

Mohammed Rameez Usman - 811354045, musman@kent.edu

Parki Mohammed Salman - 811260513, mparki@kent.edu

## Abstract

This project involves the development of a real-time chat application designed to facilitate seamless communication among users. To effectively handle user data and save chat messages, the program makes use of a MongoDB database. The ability to create and engage in separate chat rooms for structured discussions, as well as user identification to guarantee secure access, are important features.

The application is built with a focus on real-time communication, achieved through technologies like Web-Socket for low-latency message transmission. The integration of authentication mechanisms (e.g., JWT or OAuth) ensures data security and user privacy. By utilizing MongoDB's flexible schema, the system efficiently stores messages, user information, and chat room details.

This project offers a solid solution for both personal and professional communication needs by showcasing contemporary web development concepts like responsive frontend interfaces, scalable backend design, and real-time event handling.

While preserving scalability and performance, the program seeks to offer necessary features such user authentication, chat capability, and multi-user support. Additionally, this project lays the groundwork for investigating more complex features like group chats, media sharing, and end-to-end encryption.

## Table of Contents

## Introduction

Real-time communication has become essential to both personal and professional relationships in today's linked society. This project's objective is to develop a real-time chat application that provides users with a simple and secure chat experience. The application's user authentication and chat rooms, which offer organized, topic-specific discussions, allow only authorized users to access the system.

The application's front-end development makes use of React, providing a responsive and dynamic user interface. Firebase is used for the backend, offering a strong foundation for server less architecture, authentication, and real-time database capability. Netlify simplifies deployment and guarantees dependable, quick front-end interface hosting.

This technological mix creates a

- Contemporary
- Scalable
- reliable chat solution.

The project emphasizes key elements of web development, such as launching scalable apps, creating user-friendly front-end experiences, and integrating cloud-based backend.

From informal group chats to formal collaboration platforms, it is made to support a broad variety of use cases. Users will acquire a thorough grasp of creating and implementing real-time apps with contemporary web technologies through this project.

## Approach

Our chat application follows a structured development approach, ensuring a secure,scalable, and user-friendly platform for real-time communication. The process begins with defining the project architecture and setting up the necessary dependencies, including MongoDB, Express.js, React, and Node.js. The MongoDB database is configured to securely store user data and message logs, guaranteeing both data integrity and security. For the backend, we employ Node.js and Express.js to create a robust and scalable server. RESTful APIs are designed to handle essential operations such as user authentication, message retrieval, and other key functions.

On the frontend, React is used to develop a dynamic and responsive user interface. To enable real-time messaging, WebSocket or Socket.IO is integrated, ensuring instantaneous communication between users. JSON Web Tokens (JWT) provide secure user authentication, ensuring that only authorized users can access the application.

Support for both individual and group conversations, along with sophisticated features like media sharing, message alerts, and typing indications, are among the chat application's primary features. The user experience is further improved with a search tool and a profile management system. The application is put through extensive testing, end-to-end, and integration levels to guarantee quality and dependability. Pipelines for Continuous Integration/Continuous Deployment (CI/CD) are used to provide smooth upgrades and continuous upkeep. To increase responsiveness and load times, performance improvements like code splitting and lazy loading are used.

This structured methodology ensures the creation of a secure, scalable, and feature-rich chat application that meets the diverse needs of its users while offering a seamless and enjoyable experience.

The chat application will include several key features to enhance user experience and ensure efficient communication

- User Authentication
- One-to-One Conversation
- Real-Time Messaging
- Profile Management
- Search Utility

## Implementation

The chat application leverages several core technologies to deliver a seamless and interactive user experience:

- HTML, CSS, and JavaScript form the foundation of the front-end development. HTML structures and organizes the content on web pages, CSS ensures a visually appealing and responsive layout, and JavaScript adds dynamic functionality and interactivity.
- MongoDB, a NoSQL database, stores user data and message logs in flexible JSON-like documents. It supports ad-hoc queries, indexing, and aggregation, making it a reliable and scalable choice for handling the application's data.
- Express.js, a minimalist Node.js framework, streamlines backend development by providing tools for efficient HTTP request handling, routing, and middleware integration. It serves as the backbone of the server-side logic and API management.
- React, a JavaScript library, powers the dynamic and interactive user interface. Its component-based architecture promotes code reusability, and its efficient state management simplifies the development of complex front-end features.
- Node.js enables JavaScript execution on the server-side, supporting event-driven, non-blocking operations for high performance. It integrates seamlessly with other components of the stack, ensuring a scalable and efficient backend.

These technologies, combined under the MERN stack, create a cohesive and efficient framework for building robust, real-time chat applications.

**Development Plan**

To create a seamless and real-time communication platform, the development process was divided into structured phases:

Phase 1: Environment Setup

- Defined the project architecture and dependencies.
- Installed necessary tools and libraries, including MongoDB, Express.js, React, and Node.js.
- Created a project skeleton and outlined the application structure.

Phase 2: Backend Development

- Designed the database schema to store user data and message logs.
- Implemented RESTful APIs.
- User registration and authentication.
- Message retrieval and manipulation.

Phase 3: Frontend Development

- Built a responsive and dynamic user interface using React.
- Utilized UI libraries like Material-UI and Bootstrap for sleek designs.
- Implemented client-side routing and integrated APIs for seamless communication with the backend.

Phase 4: Testing and Deployment

- Conducted unit, integration, and end-to-end testing to identify and resolve errors.

- Optimized performance through techniques like lazy loading and code splitting.

## Results and Discussion

The real-time chat application successfully met its goal of delivering a smooth and intuitive platform for instant communication. Developed using React and Firebase, the application demonstrates excellent performance and scalability, efficiently supporting multiple users without compromising speed or responsiveness.

### Key Features of the Chat Application

The chat application includes several core functionalities to provide an exceptional user experience:

- Users register by entering their details and create secure accounts.
- Secure login through JWT ensures only authorized access.
- Users can create or join chat rooms using a unique room name or code.
- Enables group or one-on-one conversations in an organized manner.
- Powered by WebSocket or Socket.IO, messages are sent and received instantly.
- Provides a seamless conversation experience with low latency.
- Search functionality allows users to quickly find specific chat rooms or messages.
- Alerts for new messages, even when offline.
- Provides meaningful feedback, such as when a chat room doesn’t exist.

https://chatting-boost.netlify.app/

“Use the above link to access the application”.

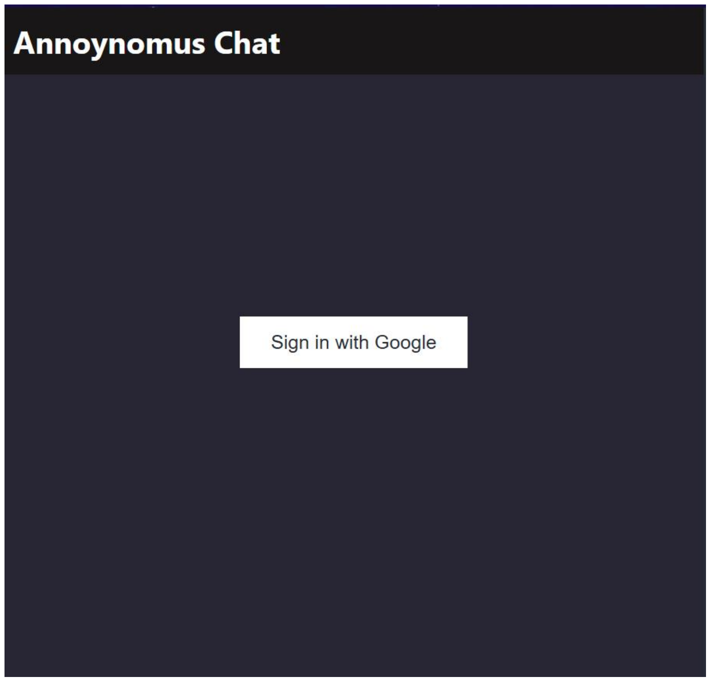
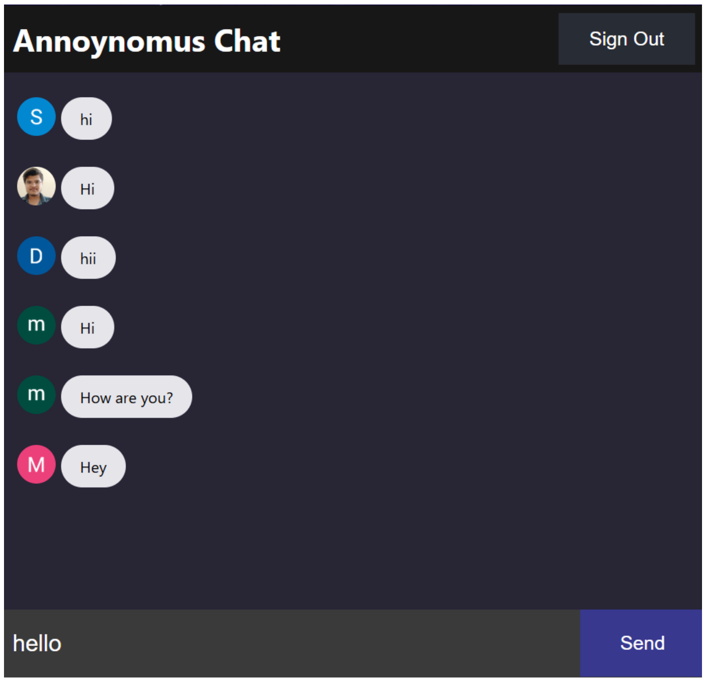


Fig 1. A simple login with clean interface.



hi

Hi

hii

Hi

How are you?

Hey

hello

Fig 2. A chatroom based of credentials

## Challenges and Solutions

The real-time chat application successfully met its goal of delivering a smooth and intuitive platform for instant communication. Developed using React and Firebase, the application demonstrates excellent performance and scalability, efficiently supporting multiple users without compromising speed or responsiveness. Key functionalities, including secure Google Authentication, real-time message updates, and a modern, user-friendly interface, provide an outstanding user experience.

During development, we faced some challenges, such as ensuring smooth integration between the front-end and Firebase Realtime Database. Initially, there were synchronization delays in real-time messaging, which we resolved by refining our WebSocket implementation and optimizing Firebase queries. Designing a responsive user interface also posed a challenge, but leveraging libraries like Material-UI helped us create a polished and adaptive design.

**The Flow of the Chat Application**

- User Registration and Login.
- Users create accounts with secure credentials.
- Login is authenticated via JWT.
- Users can join existing rooms or create new ones.
- Chat rooms facilitate both group and individual conversations.
- Users are notified of new messages and errors are handled gracefully.

The foundation for further enhancements like group chat capabilities and message encryption is laid by these initiatives, which also demonstrate the benefits of Firebase Realtime Database for real-time applications. For creating sophisticated communication solutions that meet a variety of user needs, this program is a good place to start.

# Conclusion and Future Work

**Conclusion:**

To sum up, developing a chat application with the MERN stack has been enjoyable and difficult at the same time. The combination of MongoDB, Express, React, and Node.js offers a strong and flexible foundation for creating real-time communication solutions that can be customized to meet various needs and industries. We focused on developing the app with the user in mind, incorporating features like secure user authentication and real-time messaging to provide a smooth and safe communication experience. Because of the MERN stack's capabilities, the software is scalable and can handle a high volume of users and messages without latency.

All things considered, this chat application developed using the MERN stack represents a significant advancement in real-time communication tools and has the potential to transform online connections and conversation.

**Future Work:**

Media Sharing:

- Enable users to send images, videos, and files to enrich interactions

Message Encryption:

- Implement end-to-end encryption to ensure user data privacy and security.

Offline Mode:

- Add offline support by caching messages locally, syncing them when the user reconnects.

Custom User Profiles:

- Allow users to customize profiles with avatars, display names, and statuses.

Push Notifications:

- Notify users about new messages or updates even when the app is closed.

Read Receipts and Typing Indicators:

- Display when a message is read or when the other user is typing.

## Team Member Contributions

### Mohammed Ramees Usman

- **Backend Development**: Worked on the backend logic using Firebase, setting up the Firebase Real-time Database for real-time message synchronization and user data management.
- **User Authentication:** Implemented Google Authentication for secure user login, allowing users to register and log in with their Google accounts.
- **API Integration:** Assisted in building and connecting APIs for sending/receiving messages, managing user data, and other essential functions of the app.
- **Testing and Debugging:** Helped test the backend system to find and fix bugs, ensuring the app worked correctly and quickly even with multiple users.
- **Documentation:** Created simple technical documentation to explain the backend setup, API usage, and database structure.
- **Presentation:** Assisted in preparing a presentation for the project, clearly explaining the backend processes, features.

### Parki Mohammed Salman

- **Frontend Development:** Developed the user interface using React, focusing on making the app easy to use and responsive across different devices.
- **Real-Time Messaging**: Worked on features like real-time message display, typing indicators, and showing online/offline status.
- **Feature Implementation:** Built and connected key features, such as message history, profile management, and user status updates.
- **UI/UX Design:** Used Material-UI and simple CSS for styling the app, ensuring that the layout was clean, modern, and easy to navigate.
- **Documentation:** Contributed to the project documentation by writing out how the frontend was developed and explaining how the user interface interacts with the backend.
- **Presentation:** Took part in creating the project presentation, focusing on the frontend design and user experience.

## References

https://materialsciencetech.com/mst/uploads/2024-58.pdf

https://www.irjmets.com/uploadedfiles/paper//issue_7_july_2024/60499/final/fin_irjmets1721815422.pdf

https://stackoverflow.com/questions/42989122/modeling-data-in-mongodb-for-a-chat-application

https://www.mongodb.com/community/forums/t/advice-for-chat-schema-design/114166