International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET2310267

Real-Time Chat Application

Ms. Archana Nikose, Sakshi Dosani, Shreya Pardhi, Deep Nikode, Anurag Jais
Department of Computer Science and Engineering, Priyadarshini Bhagwati College of Engineering, Nagpur, Maharashtra, India

ARTICLEINFO

Article History:
Accepted: 01 April 2023
Published: 15 April 2023

Publication Issue
Volume 10, Issue 2
March-April-2023

Page Number
496-501

ABSTRACT

The emergence of new technologies has brought about significant changes in the way people communicate with each other. One of the most popular ways of communication in today's digital age is through messaging applications. To facilitate this need, several chat applications have been developed. In this thesis, we introduce a chat application built using the MERN stack, which is a popular technology stack used for building web applications. This application provides users with the ability to create accounts, join chat rooms, and send messages to other users in real-time. With the use of web sockets and Angular's two-way binding, the application allows users to see messages as soon as they are sent. Moreover, the application also includes features such as user authentication and authorization to ensure secure access to the chat rooms. Through this project, we aim to demonstrate the feasibility and effectiveness of building real-time chat applications using the Mern stack.

Keywords: Real time chat app, Chat app using mern stack, Chat app.

I. INTRODUCTION

Chat applications have become an integral part of our day-to-day life and have had a significant impact on how we communicate with each other. With numerous chat applications available in the market, each offering unique features and capabilities, users are spoilt for choice. Companies that develop these applications compete with each other to add new features and improve the user experience with each release. This competition has led to the development of some of the world's top companies, generating high revenue and employing a large number of people.

However, with the growing concern of data theft, companies must ensure the security of their users' data and protect them from third-party data breaches. To address this, the basic chatting system should involve both sending and receiving processes simultaneously, which can be achieved through the MERN concept.

Developers worldwide are constantly striving to enhance the user experience of chat applications and improve their workflow to deliver projects and changes quickly. This is where stacks come into play, which allow developers to build web applications quickly and efficiently. Mern and MERN are two popular stacks built on JavaScript that offer an end-to-end framework for building comprehensive web apps that enable browsers to connect with databases.

Our team recognized the need for a reliable and user-friendly chat app that could be used across different platforms and devices, and we decided to build it using the Mern stack. This allowed us to take advantage of the strengths of each technology and create a seamless user experience.

II. METHODS AND MATERIAL

To develop a chat application, the first step was to design a database schema that could store user information, chat rooms, and messages. MongoDB, a document-oriented database, was chosen for this purpose due to its flexibility and ease of use. The schema was designed to have multiple collections, each responsible for storing different types of data.

Once the database schema was in place, the next step was to build the server-side API that would handle requests from the client-side application. Express.js, a fast and flexible Node.js web application framework, was used to build the server-side API. It provides features such as routing, middleware, and templating, making it an ideal choice for building web applications and APIs.

On the client-side, AngularJS was chosen to create a responsive and intuitive user interface. AngularJS is a JavaScript-based framework that simplifies the development of web applications by providing a structured framework for creating dynamic views. It allows developers to create complex user interfaces with ease, using reusable code and components.

The chat application required real-time message updates, which were achieved using socket.io, a JavaScript library for real-time web applications. Socket.io enables bidirectional communication between the server and the client, allowing for real-time updates and notifications.

User authentication was also an important feature of the chat application, and this was implemented using JSON Web Tokens (JWT). JWT is a secure and easy-to-use authentication mechanism that allows users to securely transmit information between parties.

Finally, Node.js was used to deploy the application on a cloud hosting service. Node.js is a platform built on the Chrome V8 JavaScript engine that allows developers to run JavaScript code outside of a web browser on the server side, providing a powerful and efficient way to build server-side applications using JavaScript. It allows Users to run JavaScript code on the server. It is fast and scalable, making it ideal for building web applications that can handle high traffic.

In conclusion, developing a chat application requires a combination of technologies and frameworks, each serving a specific purpose. The choice of technologies depends on the specific requirements of the application, such as real-time message updates, user authentication, and scalability. MongoDB, Express.js, AngularJS, socket.io, JWT, and Node.js were the technologies chosen for this particular chat application, resulting in a robust and feature-rich application that met the requirements of the project.

Here is a more detailed explanation of the technologies used in the chat application:

1. HTML, CSS, and JavaScript form the backbone of web development and are essential tools for creating dynamic and interactive websites. HTML is used to structure and define the content of web pages, CSS is used for styling and layout, and JavaScript is used for creating dynamic interactions and functionality.
2. MongoDB: It is a cross-platform document-oriented NoSQL database. MongoDB stores data in JSON-like documents, making it easy to work with for developers.

It provides support for ad-hoc queries, indexing, and aggregation, making it a popular choice for building web and mobile applications that require a flexible and scalable database.

3. Express: Express is a web application framework for Node.js that provides developers with a powerful set of tools to build back-end web applications and APIs. It includes features such as routing and middleware, which make it easier to handle HTTP requests and responses. Express is known for being flexible and easy to use, making it a popular choice among developers for building web applications and APIs.

4. React: React is based on a component-based architecture, which Merns developers can create reusable UI components that can be used across an application. React also manages the state of an application more efficiently, making it easier to build complex applications.

5. Node.js: Node.js is a JavaScript runtime environment that enables developers to execute JavaScript code on the server-side. It is cross-platform and open-source, which allows for the creation of scalable and high-performance web applications and APIs. Node.js is known for its speedy and efficient event-driven, non-blocking I/O model.

The MERN stack combines these technologies to create a comprehensive web application framework. MongoDB provides a flexible and scalable database, Express provides a simple and minimalist back-end framework, React provides a powerful front-end library for building user interfaces, and Node.js used to run JavaScript code on the server-side. Together, these technologies provide a full-stack solution for building robust and scalable web applications.

III. PROPOSED SCHEME

This section explains the development plan for this web application :

The development plan for our chat application involved creating a platform for users to communicate with each other in real-time using the MERN stack, which is a widely used and modern technology stack in the industry. The application's frontend was developed using React, while the backend was developed using Node.js and Express.js, and the data was stored in MongoDB.

In the first phase, we focused on setting up the development environment and installing the necessary tools and libraries. We created a project skeleton and defined the basic structure of our application. This phase was critical in establishing a solid foundation for our development process.

In the second phase, we designed the database schema and created the necessary models and controllers. We also implemented user authentication and authorization, as well as RESTful APIs to handle data retrieval and manipulation. This phase was essential for ensuring that our application was secure and that users' data was protected.

In the third phase, we built the frontend user interface using React. We used various UI libraries and frameworks, such as Material-UI and Bootstrap, to create an attractive and responsive design. We also implemented client-side routing and integrated with the backend APIs.

In the final phase, we conducted extensive testing and debugging to ensure that our application was free of errors and performed optimally. We also deployed the application to a cloud platform, such as AWS or Heroku, to make it accessible to users.

Overall, we believe that our chat application using the MERN stack provides users with a seamless and efficient way to communicate with each other in real-time.

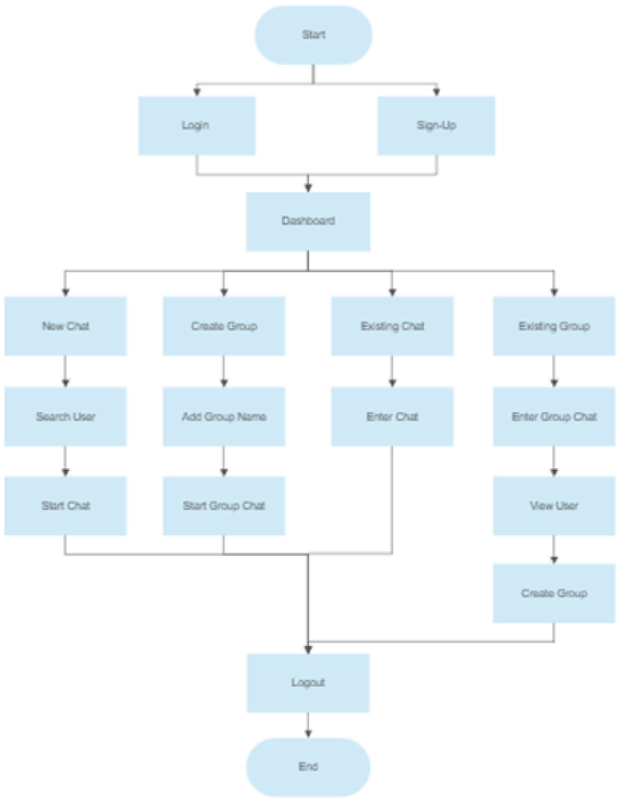


Fig 4.1:Data flow diagram of the system

The flow of working of a chat application built using the Mern stack can be explained as follows:

User registration and authentication: The first step in using the chat application is to create an account. The user is required to enter their details and register to access the application. Once registered, the user can log in to the application using their email and password.
Joining chat rooms: The next step is to join a chat room. The user can either create a new chat room or join an existing one by entering the room name or code. Once the user has joined a chat room, they can start chatting with other users who are also part of that chat room.
Sending and receiving messages: The core feature of the chat application is the ability to send and receive messages in real-time. Using web sockets, the application enables users to send and receive messages instantly, allowing for a seamless conversation experience.

User authentication and authorization: To ensure data security and user privacy, the application incorporates user authentication and authorization features. Users must log in to access the application and only authorized users can join specific chat rooms.
Error handling and notifications: The application also includes error handling and notifications to ensure a smooth user experience. For example, if a user tries to join a chat room that does not exist, the application will display an error message. Similarly, if a user receives a new message while they are offline, the application will send them a notification.

IV. IV. RESULTS AND DISCUSSION

Following are some of the results from our application:

Fig 3.1 :Sign-up Page

This is our signup page where users have to enter their details like name ,email address, password and also can upload your pictures to signup our applications.

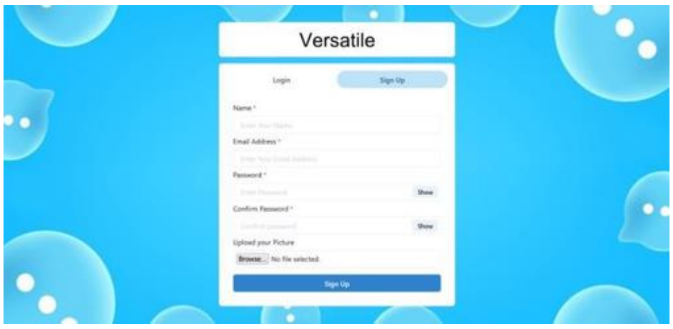


Fig 3.2 : Main Interface

This is the main interface of our application where different options are available like user can create groups , can message any person in personal , can see how many users are online ,etc.

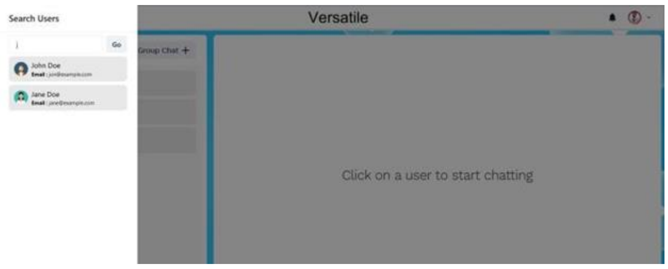
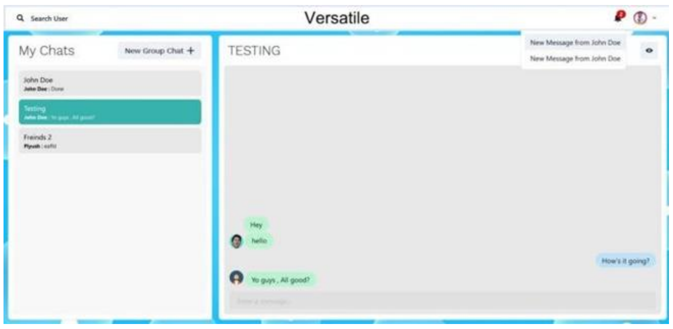


Fig 3.3 : SEARCH USERS

The application interface allows users to communicate with any registered user directly, enabling seamless communication and fostering community engagement.



New Message from John Doe

New Message from John Doe

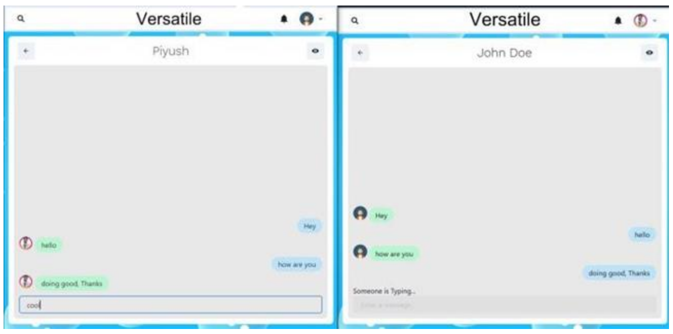
Hey

hello

How's it going?

Yo guys , All good?

Fig 3.4: NOTIFICATIONS



Hey

hello

how are you

doing good, Thanks

cool

Someone is Typing..

Fig 3.5: CHATTING INTERFACE

V. V. CONCLUSION

In conclusion, developing a chat application using the MERN stack has been a challenging but rewarding experience. The use of MongoDB, Express, React, and Node.js provides a powerful and flexible framework for creating real-time communication and collaboration solutions that can be tailored to meet a wide range of use cases and industries. The app has been designed with user experience in mind, and features such as real-time message updates and user a

uthentication have been implemented to provide a seamless and secure communication experience. The scalability and robustness of the MERN stack ensure that the app can handle a high volume of users and messages without compromising on performance. Going forward, there are many opportunities for further development and improvement of the app. This includes adding new features such as video and voice chat, integrating with other applications and platforms, and enhancing the user interface to make it more intuitive and user-friendly. Overall, the chat app developed using the MERN stack represents a significant achievement in the field of real-time communication and collaboration, and has the potential to revolutionize the way people connect and communicate online.

VI. REFERENCES

[1]. Masiello Eric. Mastering React Native. January 11; 2017. This book is a comprehensive guide to building mobile applications using React Native.

[2]. Naimul Islam Naim. ReactJS: An Open-Source JavaScript library for front-end development. Metropolia University of Applied Sciences. This article provides an overview of ReactJS and its key features for front-end web development.

[3]. Stefanov Stoyan, editor. React: Up and Running: Building web Applications. First Edition; 2016. This book is a beginner-friendly introduction to React, covering its core concepts and providing practical examples for building web applications.

[4]. Horton Adam, Vice Ryan. Mastering React; February 23; 2016. This book provides a comprehensive guide to React, covering its core concepts, practical examples, and advanced techniques for building complex applications.

[5]. Alex Kondov. Express Architecture Review. This article provides a review of the architecture of Express.js, a popular web framework for building Node.js applications.

[6]. Express.js documentation. This documentation provides a comprehensive guide to building web applications using Express.js.

[7]. Adam Horton. Node.js vs Python: What to Choose. This article provides a comparison of Node.js and Python for web development, highlighting their strengths and weaknesses.

[8]. Node.js documentation. This documentation provides a comprehensive guide to building server- side applications using Node.js.

[9]. VSChart. This website provides a comparison of various programming languages and frameworks based on popularity, community support, and other factors.

[10]. MongoDB documentation. This documentation provides a comprehensive guide to using MongoDB, a popular NoSQL database for building web applications.

[11]. The paper by Lakshmi Prasanna Chitra and Ravikanth Satapathy aims to compare the performance of Node.js and traditional web servers, specifically Internet Information Services (IIS), in optimizing web application development.The authors conducted various tests to evaluate the performance of both platforms and determine which one is better for developing high-performance web applications.

[12]. Guru99. React vs Angular: Key Differences. This article provides a comparison of React and Angular, two popular front-end frameworks for building web applications.

Cite this article as :

Ms. Archana Nikose, Sakshi Dosani, Shreya Pardhi, Deep Nikode, Anurag Jais, "Real-Time Chat Application", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 10 Issue 2, pp. 496-501, March-April 2023. Available at doi : https://doi.org/10.32628/IJSRSET2310267
Journal URL : https://ijsrset.com/IJSRSET2310267