A PROJECT REPORT

ON

TALKATIVE CHAT WEB APPLICATION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING
INFORMATION TECHNOLOGY

BY

Name: Maitreya Awad
Roll No: 33205
Class: TE-10

Under the guidance of
Mr. Sandeep Warhade

DEPARTMENT OF INFORMATION TECHNOLOGY
PUNE INSTITUTE OF COMPUTER TECHNOLOGY
SR. NO 27, PUNE-SATARA ROAD, DHANKAWADI
PUNE - 411 043.
AY: 2022-2023

SCTR's PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

<u>C E R T I F I C A T E</u>

This is to certify that the SPPU Curriculum-based Project report entitled
Talkative Chat Web Application

Submitted by

Name : Maitreya Awad
Roll No : 33205 .

is a bonafide work carried out under the supervision of Mr. Sandeep Warhade and it is submitted towards the partial fulfillment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Information Technology).

Mr. Sandeep Warhade	Dr. A. S. Ghotkar
Project guide	HOD IT

Date:
Place:

Acknowledgement

I would like to thank my project guide Mr. Sandeep Warhade, Department of Information Technology, PICT, for guiding me appropriately at each stage of this project. He has been extremely helpful in the completion of this project.
I would also like to thank Dr. Archana S. Ghotkar, Head of Department, Department of Information Technology, PICT, for providing me the opportunity to complete my project and present this report by providing me with all the resources required.
I would also like to extend my sincere gratitude to my team members and all those who have contributed either directly or indirectly towards the completion of this project.

Name : Maitreya Awad
Roll No: 33205

Contents

1. Introduction

1.1 Introduction

In today's interconnected world, real-time communication has become an integral part of our lives. From instant messaging to collaborative work environments, the need for seamless, interactive, and responsive chat applications has grown significantly. In response to this demand, developers have turned to modern technologies such as the MERN stack and Socket.IO to build powerful chat web applications.

The MERN stack, which consists of MongoDB, Express.js, React.js, and Node.js, has gained immense popularity in recent years due to its ability to create robust and scalable web applications. By combining these technologies, developers can take advantage of JavaScript's full-stack capabilities, simplifying the development process and creating highly efficient applications.

Socket.IO, on the other hand, is a real-time communication library that enables bidirectional and event-based communication between the web server and client. It provides a reliable and efficient means of transmitting data in real-time, making it the perfect choice for building chat applications where instant message delivery is paramount.

This report focuses on the development of a real-time chat web application using the MERN stack and Socket.IO. The application allows users to engage in instant messaging, facilitating seamless communication and collaboration.

By the end of this report, readers will gain a comprehensive understanding of the MERN stack, Socket.IO, and how they can be leveraged to build a real-time chat web application. Additionally, they will gain insights into the best practices, design patterns, and considerations involved in developing similar applications.

1.2 Objectives

1. To understand the basic foundations for building a real time chat application.
2. To implement best javascript practices and industry standards.
3. To learn about MERN stack and how to build projects using it.
4. To understand how frontend and backend are integrated with each other.

1.3 Scope

Through this exploration of the MERN stack and Socket.IO, we aim to demonstrate the power and versatility of these technologies in creating immersive and dynamic chat applications that cater to the growing demands of today's interconnected world.

2. Architecture and Technologies

2.1 ARCHITECTURE

The MERN stack chat web app using Socket.IO follows a client-server architecture. The client-side of the application is built using React.js, a popular JavaScript library for building user interfaces. React.js provides an efficient and easy-to-use interface for the end-user to interact with the application. The client-side also utilizes Socket.IO client, which enables the client to receive real-time data updates from the server.

On the server-side, the application is built using Node.js, an open-source, server-side JavaScript runtime environment, and Express.js, a web application framework for Node.js. The server-side also uses Socket.IO server, which facilitates bidirectional, event-based communication between the server and the client.

The application's data is stored in MongoDB, a NoSQL database that stores data in JSON-like documents. MongoDB provides a scalable and flexible data storage solution that can be easily integrated with Node.js and Express.js.

The MERN stack chat web app using Socket.IO architecture is based on the Model-View-Controller (MVC) pattern. The model layer contains the MongoDB database schema and the logic for accessing and manipulating data. The view layer is responsible for rendering the user interface using React.js. The controller layer is implemented using Express.js, which handles incoming HTTP requests and sends appropriate responses to the client. The Socket.IO library is used to manage real-time communication between the server and the client.

To ensure a smooth and responsive user experience, the application uses asynchronous programming techniques, such as callback functions, promises, and async/await, to handle real-time data updates and user interactions. The application also uses JSON Web Tokens (JWT) to authenticate users and protect against unauthorized access.

Overall, the MERN stack chat web app using Socket.IO architecture is a scalable and efficient solution for building real-time chat applications. By leveraging the power of React.js, Node.js, Express.js, Socket.IO, and MongoDB, developers can build highly interactive, real-time chat applications that cater to the needs of today's digital world.

2.2 TECHNOLOGIES

1. **MongoDB:** MongoDB is a NoSQL database that stores data in a flexible, JSON-like format. It provides scalability and high-performance data storage, making it suitable for handling large amounts of data in web applications.

2. **Express.js:** Express.js is a minimalistic web application framework for Node.js. It simplifies the process of building server-side applications by providing a set of robust features and middleware for handling routes, requests, and responses. Express.js is known for its simplicity, flexibility, and ease of use.

3. **React.js:** React.js is a JavaScript library developed by Facebook for building user interfaces. It follows a component-based architecture, allowing developers to build reusable UI components. React.js provides efficient rendering and updates by using a virtual DOM, which results in fast and responsive user interfaces.

4. **Node.js:** Node.js is a JavaScript runtime environment that allows developers to execute JavaScript code on the server-side. It provides an event-driven, non-blocking I/O model, making it highly efficient and scalable for building server-side applications. Node.js enables server-side JavaScript execution, allowing developers to use a unified language and share code between the client and server.

5. **Socket.IO:** It is a JavaScript library that enables real-time, bidirectional communication between the web server and client. It allows for instant and continuous data exchange, making it a powerful tool for building real-time applications such as chat systems, collaborative editing tools, and live dashboards.

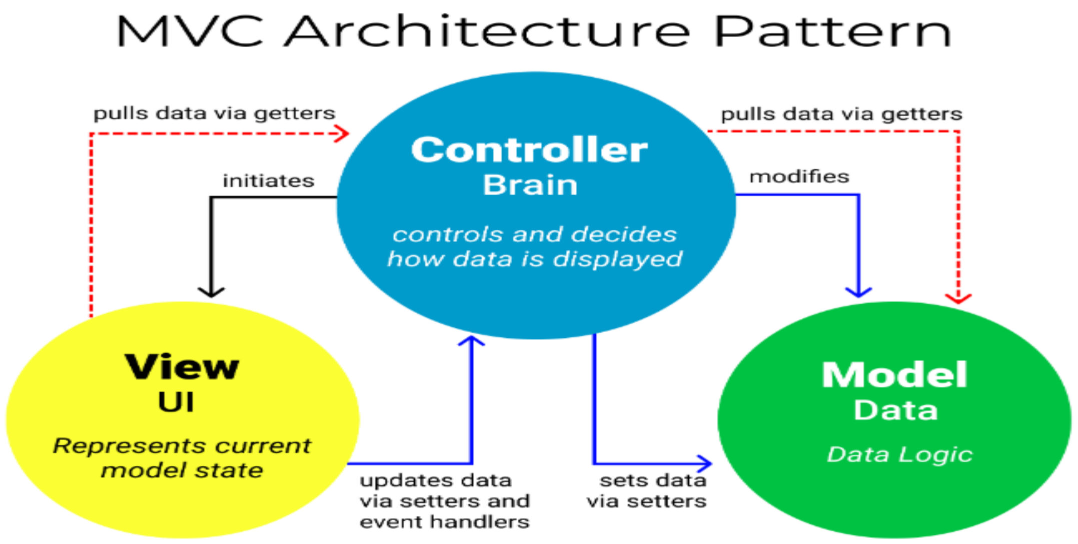
3. Design and Implementation

The chat web app is designed based on the MVC(Model View Controller) framework. MVC (Model-View-Controller) is a software architectural pattern commonly used for developing web applications. It separates the application into three interconnected components: the Model, the View, and the Controller. Each component has its own distinct role and responsibilities.

1. **Model:** The Model represents the application's data and business logic. It encapsulates the data and provides methods to manipulate and access that data. The Model is responsible for managing the application's data, handling data validation, and implementing business rules.

2. **View:** The View is responsible for presenting the data to the user and handling the user interface. It provides a visual representation of the application's data and allows users to interact with the application. The View communicates with the Model to retrieve data and updates the user interface accordingly.

3. **Controller:** The Controller acts as an intermediary between the Model and the View. It receives user input from the View and determines the appropriate actions to take. In an online chat web application, the Controller would handle user actions such as sending messages, retrieving messages, and managing user authentication. It updates the Model based on user actions and notifies the View of any changes that need to be reflected in the user interface.



4. Features and Functionality

1. **User Registration and Authentication:** The chat web app provides the user registration and authentication functionality, allowing users to create accounts and securely log in. This ensures that only authorized users can access the chat features and maintain privacy and this is done using bcrypt and JWT.

   User registration and authentication using bcrypt and JWT involves several steps to ensure secure handling of passwords and user verification. During user registration, the user provides their details, and a salt is generated using bcrypt's genSaltSync() function. The password is then hashed using bcrypt's hashSync() function with the generated salt. The hashed password, along with other user information, is stored in the database. During authentication, the user provides their login credentials, and the hashed password is retrieved from the database based on the provided username/email. The bcrypt compareSync() function is used to compare the provided password with the hashed password. If they match, authentication is successful. A JWT token is generated containing relevant user information and a secret key, which is returned to the client. To protect routes, server-side middleware is implemented to verify the JWT token's validity, expiration, and integrity.

2. **Real-Time Messaging:** The core functionality of the chat web app is real-time messaging. Socket.IO facilitates this real-time, bidirectional communication between the server and client, ensuring seamless message delivery.

   To implement real-time messaging, both the server and the client need to establish a Socket.IO connection. The server sets up a Socket.IO server and listens for incoming events from clients, while the client connects to the server using Socket.IO and emits events to the server. When a client sends a message, it emits a custom event to the server. The server receives the event and can perform actions based on the event's data. The server can then emit events back to the appropriate client or to all connected clients, allowing for real-time updates.

3. **Multiple Chat Rooms:** The app supports the creation of multiple chat rooms or channels, allowing users to be added to specific rooms based on their interests or groups. This feature enables organized conversations and ensures users can engage with others based on shared topics.

   Multiple chat rooms can be created and users can be added to or deleted from those chat rooms according to our choice. The messages in the chat room are visible to all the participants of the room. This is implemented using Socket.io.

4. **User Profile and Avatar:** Users have the ability to customize their profiles and set avatars or profile pictures. This feature personalizes the chat experience and helps users identify each other visually.

   Users have the choice to set up a profile picture which can also be an avatar during the registration process. If they don't set a picture, a default user photo is provided to them which is visible to other users too.

5. **Search Users:** The chat web app includes a search functionality that enables users to find and connect with other users. This feature allows users to search for specific individuals by their usernames or display names.

   The search results are filtered and sorted based on relevance, alphabetical order, or other criteria. This helps users quickly find the desired individuals among a potentially large user base.

   The search results provide a preview of each user's profile, including their profile picture, username, and additional information. This helps users evaluate the relevance of the search results and make informed decisions when initiating conversations.

5. DEPLOYMENT

Render is a cloud platform that provides a simple and scalable infrastructure for deploying and managing web applications. It offers a streamlined experience for developers, allowing them to easily deploy code to highly available and secure environments. The web application is deployed on Render (both the frontend as well as the backend). Key features of Render include:

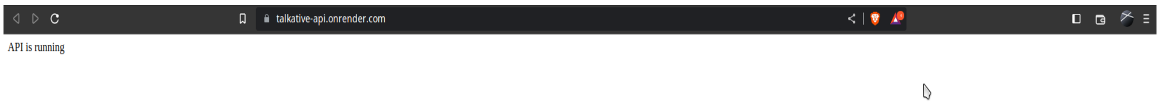
1. **Ease of Deployment:** Render simplifies the deployment process by connecting directly to your Git repository. It automatically builds and deploys your application whenever changes are pushed to the repository, eliminating the need for manual setup and configuration.

2. **Scalability and High Availability:** Render's infrastructure is designed to handle high traffic and provide reliable service. It automatically scales your application based on demand, ensuring optimal performance and availability even during periods of increased traffic.

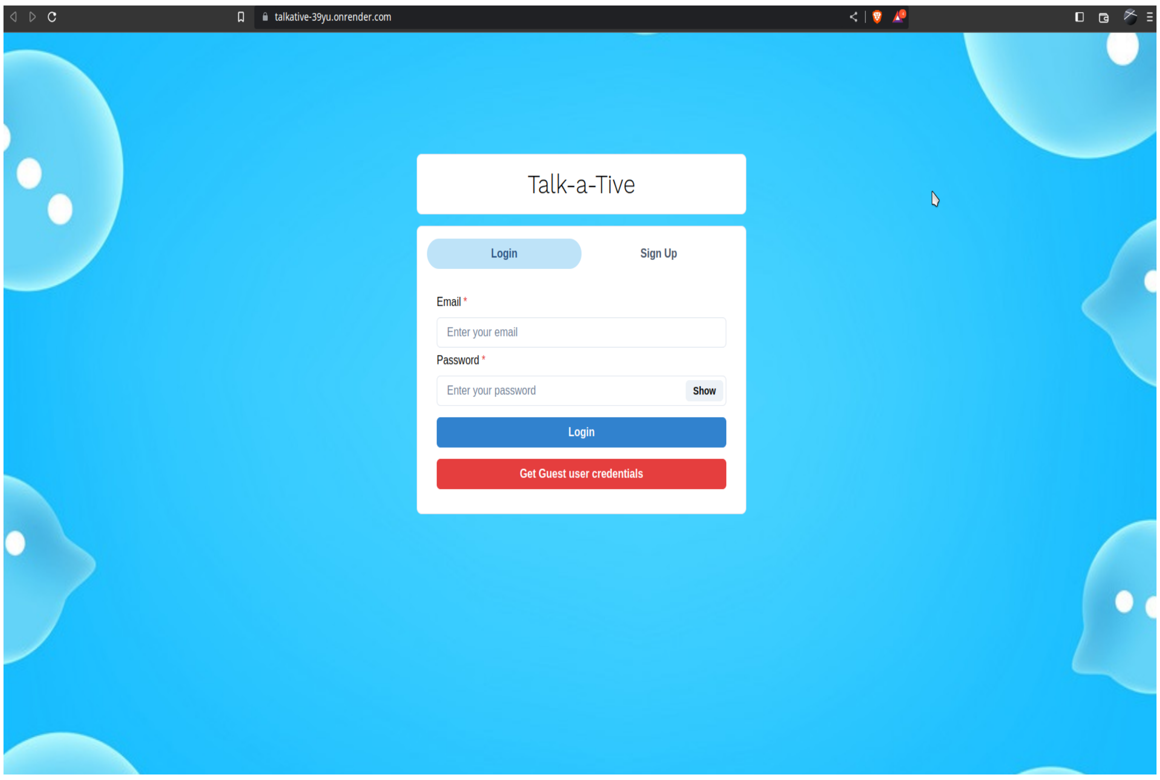
3. **Database Integration:** Render provides seamless integration with various databases, including managed services like PostgreSQL and MySQL. This simplifies the process of setting up and managing databases for your application, allowing you to focus on the development aspect.

4. **Continuous Deployment and Rollbacks:** Render supports continuous deployment, automatically deploying new changes as they are pushed to your Git repository. In case of issues or bugs, you can easily roll back to a previous version of your application with just a few clicks.

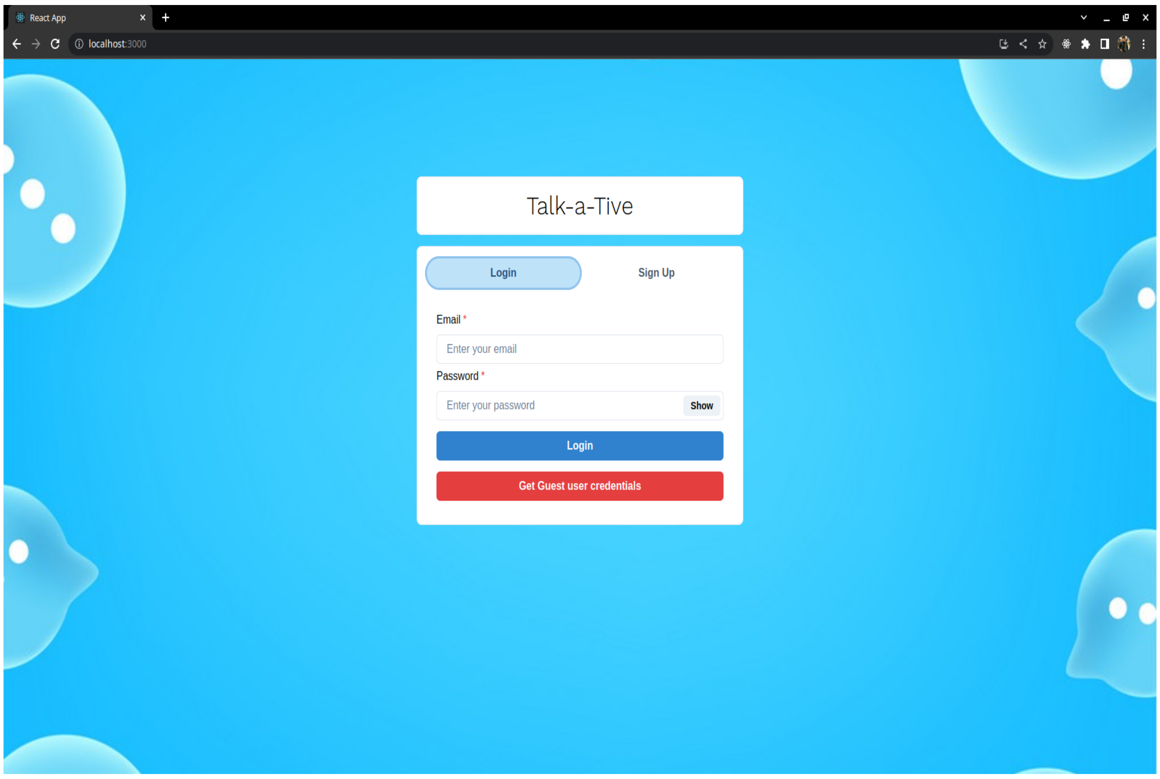
5. **Monitoring and Logs:** Render provides built-in monitoring and logging functionality, allowing you to track the performance and health of your application. You can easily access logs and metrics to diagnose issues and make data-driven decisions for optimization.

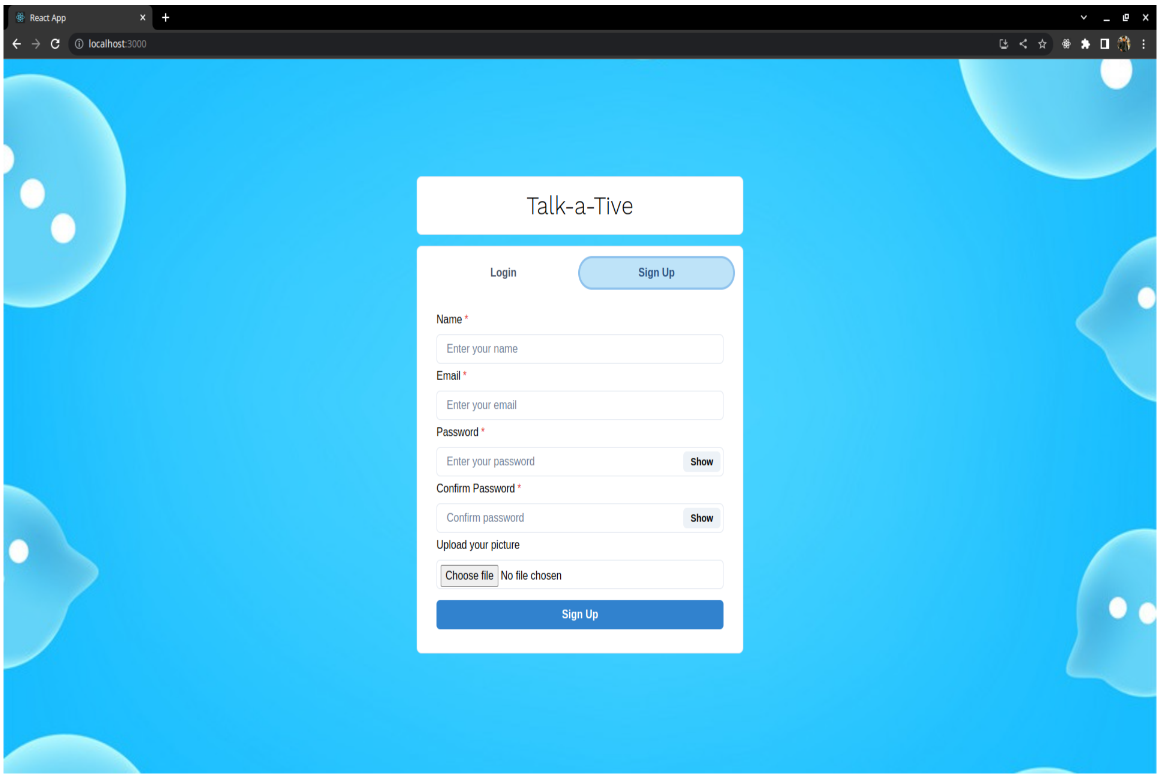


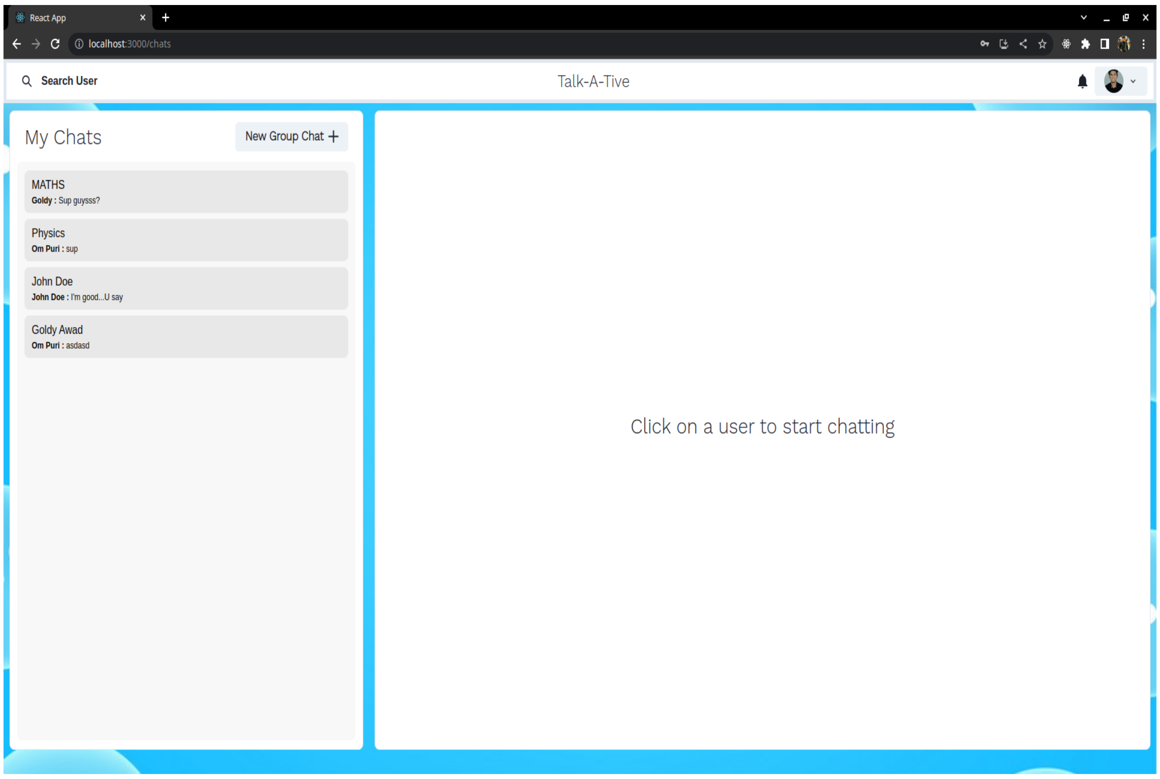
API is running

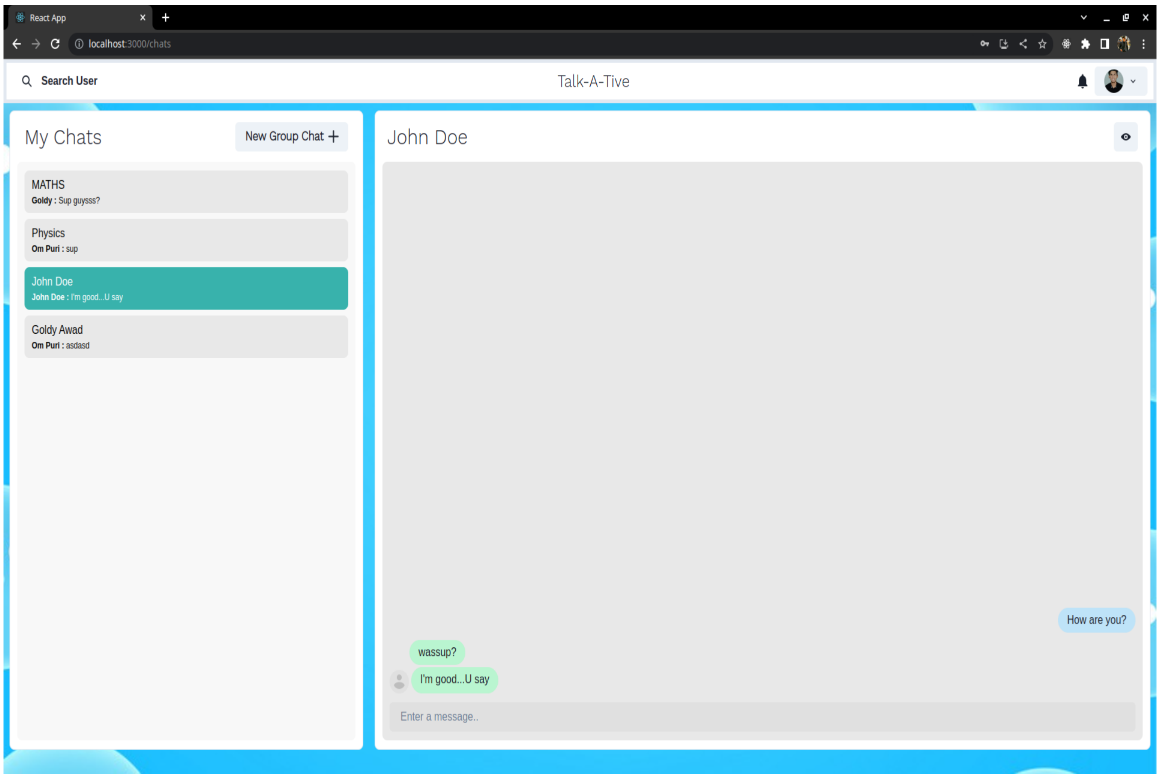


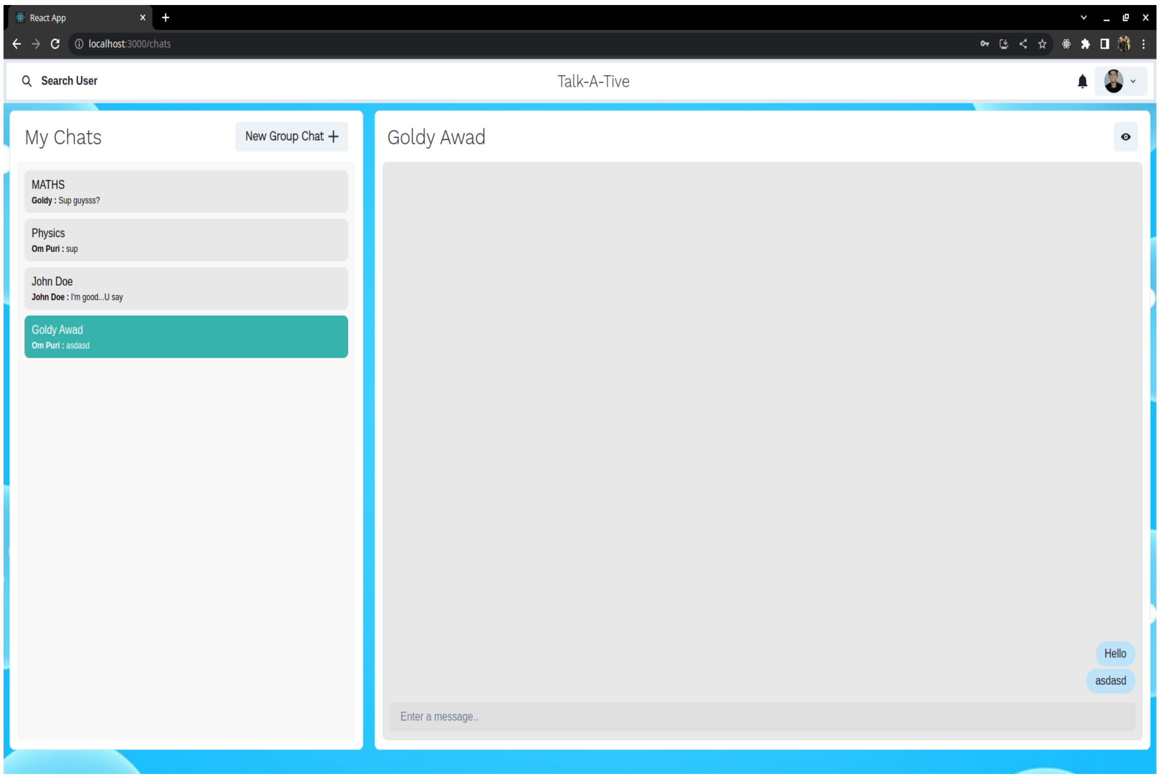
6. Results


React App
localhost:3000
Talk-a-Tive
Login
Sign Up
Email *
Enter your email
Password *
Enter your password
Show
Login
Get Guest user credentials


React App
localhost:3000
Talk-a-Tive
Login
Sign Up
Name *
Enter your name
Email *
Enter your email
Password *
Enter your password
Show
Confirm Password *
Confirm password
Show
Upload your picture
Choose file
No file chosen
Sign Up


React App
localhost:3000/chats
Search User
Talk-A-Tive
My Chats
New Group Chat +
MATHS
Goldy : Sup guysss?
Physics
Om Puri : sup
John Doe
John Doe : I'm good...U say
Goldy Awad
Om Puri : asdasd
Click on a user to start chatting


React App
localhost:3000/chats
Search User
Talk-A-Tive
My Chats
New Group Chat +
MATHS
Goldy : Sup guysss?
Physics
Om Puri : sup
John Doe
John Doe : I'm good...U say
Goldy Awad
Om Puri : asdasd
John Doe
How are you?
wassup?
I'm good...U say
Enter a message..



MATHS
Goldy : Sup guysss?

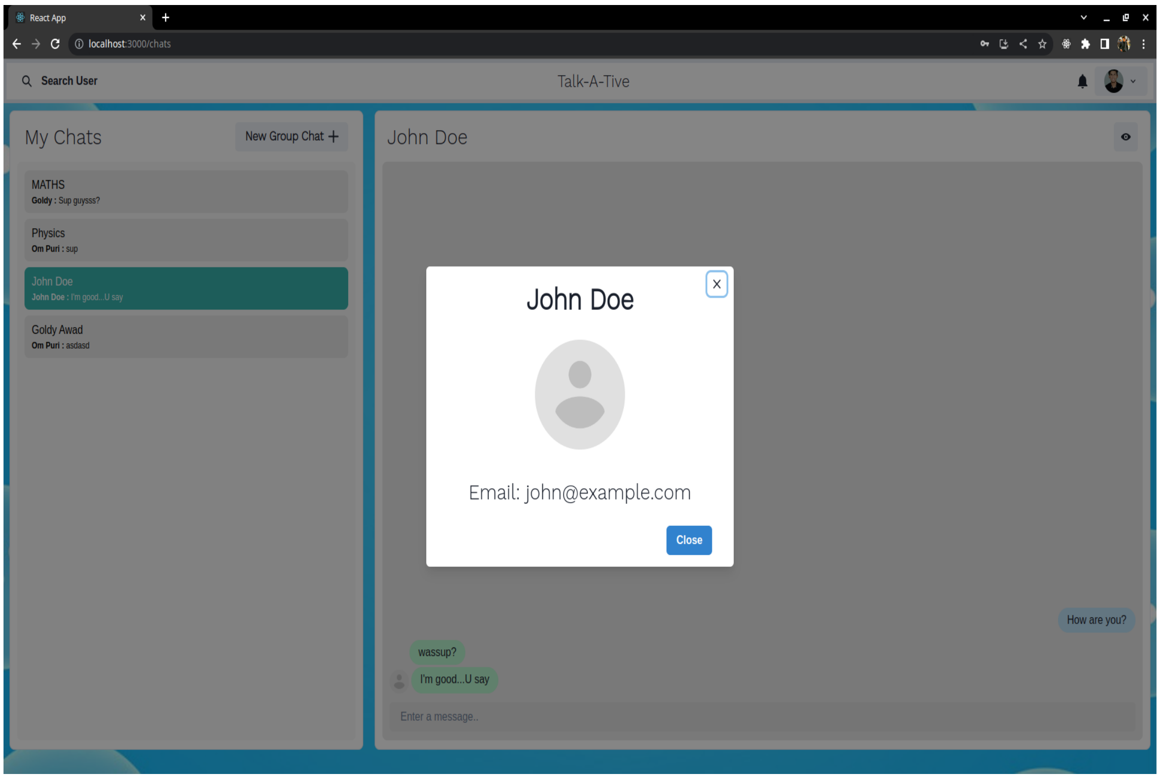
Physics
Om Puri : sup

John Doe
John Doe : I'm good...U say

Goldy Awad
Om Puri : asdasd

Hello

asdasd



MATHS
Goldy : Sup guysss?

Physics
Om Puri : sup

John Doe
John Doe : I'm good...U say

Goldy Awad
Om Puri : asdasd

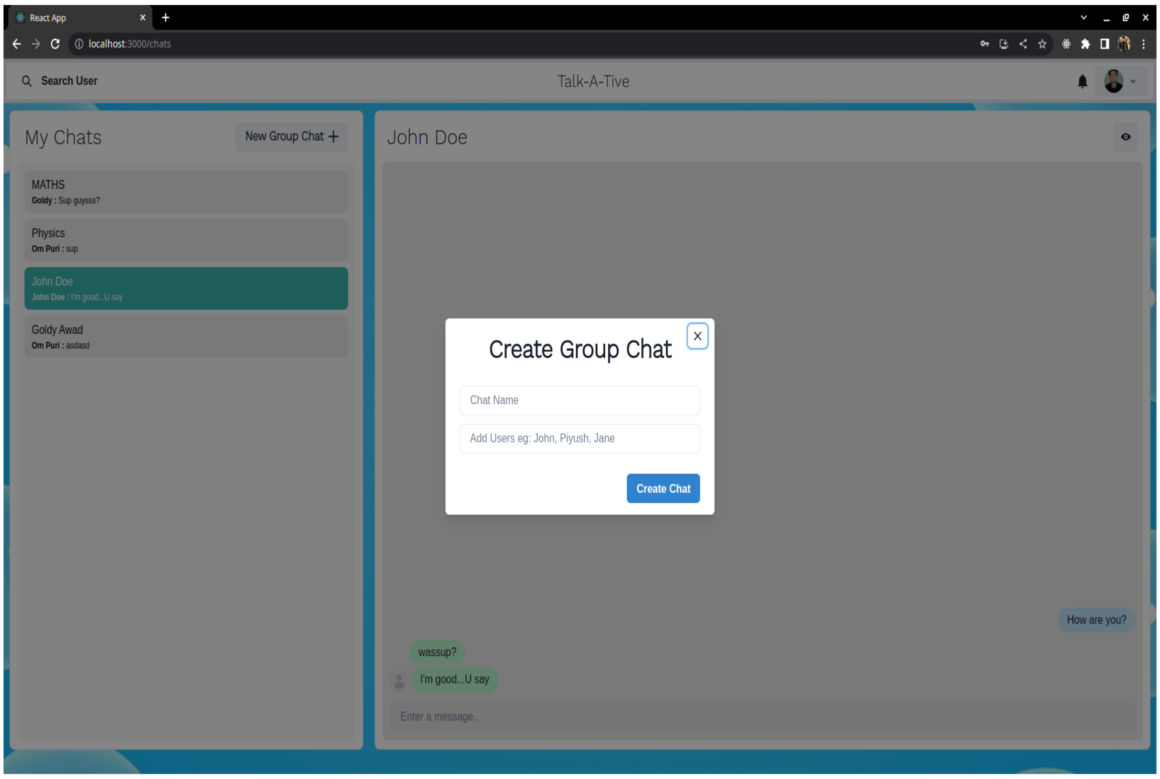
John Doe

Email: john@example.com

How are you?

wassup?

I'm good...U say



MATHS
Goldy : Sup guysss?

Physics
Om Puri : sup

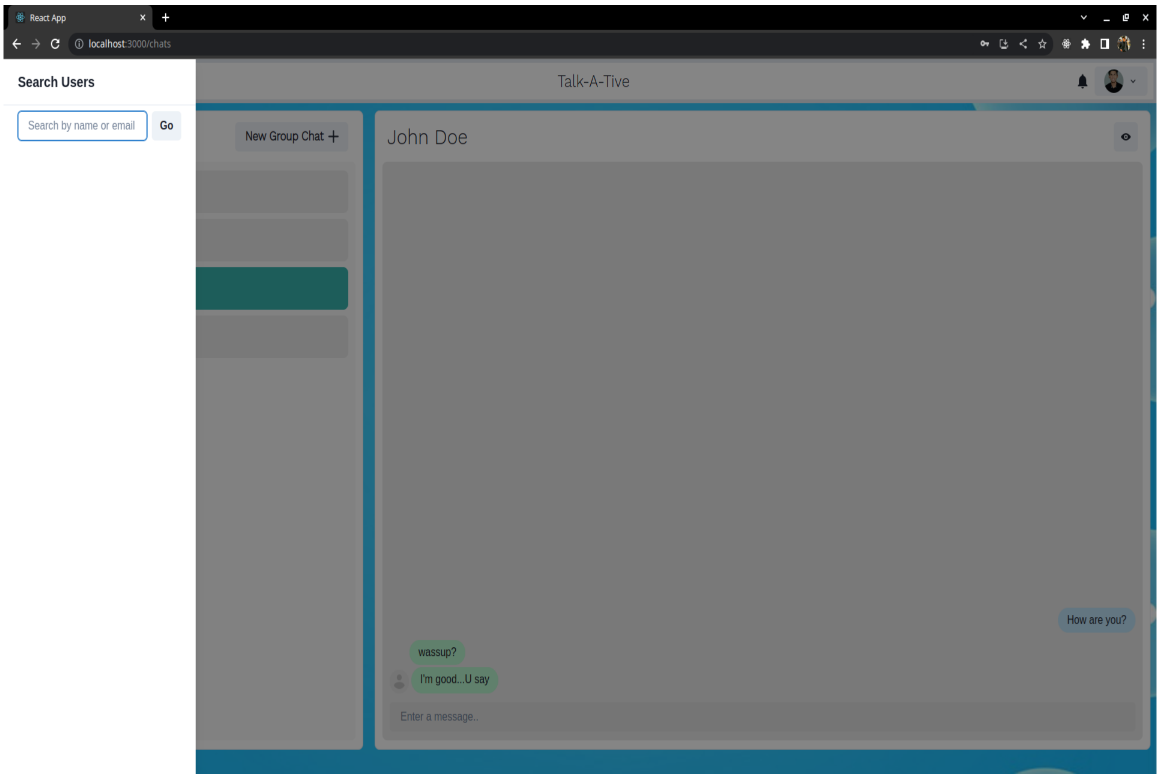
John Doe
John Doe : I'm good...U say

Goldy Awad
Om Puri : asdasd

How are you?

wassup?

I'm good...U say



How are you?

wassup?

I'm good...U say

7. Conclusion

In conclusion, the MERN Stack Chat Web App using Socket.IO deployed on the Render platform provides a robust and user-friendly solution for real-time messaging and communication. The project leverages the MERN (MongoDB, Express.js, React.js, Node.js) stack, combined with the power of Socket.IO for real-time bidirectional communication.

The web app offers various features and functionalities, including user registration and authentication, real-time messaging, multiple chat rooms, user profiles, and enhanced security and data privacy measures.

The design and implementation of the web app prioritize a seamless user experience, with a modern and intuitive user interface. The Render platform simplifies the deployment process, enabling easy integration with Git repositories and providing scalable infrastructure for optimal performance and high availability.

Overall, the MERN Stack Chat Web App deployed on Render demonstrates the ability to build a feature-rich and real-time communication platform. It showcases the effective utilization of the MERN stack and Socket.IO to create a responsive and interactive user experience. The deployment on Render ensures a reliable and scalable infrastructure, making it a suitable choice for hosting and managing web applications.

By successfully completing this project, valuable insights have been gained into the development of real-time applications, web-based communication, and the deployment process on the Render platform. The project demonstrates the ability to leverage cutting-edge technologies to create modern and engaging applications that meet the requirements of a dynamic and interconnected world.