Fuzzy Systems and Soft Computing
ISSN : 1819-4362

# SOCIAL MEDIA CHAT APPLICATION

**Nikhil Boren Senapati,** Computer Science and Engineering Gandhi Institute for Technology, India
nikhil.senapati2020@gift.edu.in
**S Abhipsa,** Computer Science and Engineering Gandhi Institute for Technology, India
abhipsa2020@gift.edu.in
**Dr. SUJIT KUMAR PANDA** Computer Science and Engineering Gandhi Institute for Technology, India

## ABSTRACT

In today's digital age, communication plays a crucial role in connecting individuals across the globe. With the increasing demand for real-time messaging applications, building a single message chat application has becomea popular endeavor. This abstract introduces a chat application developed using the MERN (MongoDB, Express.js, React.js, and Node.js) technology stack, highlighting its key features and functionality.
The proposed chat application leverages the MERN technology stack to providea robust and scalable solution. MongoDB, a NoSQL database, ensures efficientdata storage and retrieval, enabling seamless storage of user profiles and message history. Express.js, a web application framework for Node.js, facilitates the creation of a server-side backend that handles authentication, routing, and interaction with the database. Node.js acts as the runtime environment, executing JavaScript code on the server-side, enabling high-performance communication. React.js, a JavaScript library, serves as the frontend framework, providing an interactive and responsive user interface.
The chat application encompasses essential features such as user registration and authentication, login & logout, and ability to send and receive messages inreal time.

# CHAPTER-1:

## 1.1 INTRODUCTION

Welcome to our Single Message Chat Application! With the power of MERN technology, we've created a seamless platform for you to connectand 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 usto 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 are excited to have you on board and experience the convenience andefficiency of our single message chat application. Start chatting and connecting with us.

## 1.2 OBJECTIVES

The objective of this project is to build a single message chat applicationusing the MERN (MongoDB, Express.js, React.js, Node.js) stack technology. The application will allow users to send and receive messagesin 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 chatapplication using the MERN technology stack that provides users with a seamless and real-time messaging experience.

## 1.3 SCOPE

A single message chat application using the MERN (MongoDB, Express.js, React.js, Node.js) technology stack can have various scopes and features. Hereare 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 orpredefined 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, andsecurity measures when developing your chat application.

## 1.4. LIMITATIONS

There are several limitations that you may encounter when developing a single message chat application using the MERN (MongoDB, Express.js, React, Node.js) technology stack. Here are a few potential limitations:

○ Scalability: The MERN stack can handle moderate levels of traffic and users, but it may face scalability challenges when dealing with a large number of concurrent connections or heavy message traffic. Scaling the application to handle high loads might require implementing techniques such as load balancing, horizontal scaling, or using a dedicated messagingservice.

○ Performance: As the number of messages increases, fetching and rendering all the messages on the client side could lead to performance issues. It's important to implement pagination or infinite scrolling to limitthe number of messages loaded at a time and optimize the rendering process.

○ Security: Security is a critical aspect of any chat application. MERN provides tools and libraries for handling security concerns, such as authentication and authorization, but it's essential to ensure that propersecurity measures are implemented to protect user data, prevent unauthorized access, and secure

communication channels.
○ Offline functionality: MERN does not natively support offline functionality. If users need to access the chat application or send messages while offline, you would need to implement additional techniques such as client-side caching or using a service worker to handleoffline data synchronization.
○ Offline functionality: MERN does not natively support offline functionality. If users need to access the chat application or send messages while offline, you would need to implement additional techniques such as client-side caching or using a service worker to handleoffline data synchronization.
○ Mobile app development: While MERN is well-suited for web applications, if we plan to develop a mobile app version, additional frameworks or technologies, such as React Native or Flutter, would be required. This introduces the need for

# CHAPTER-2

## FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve thepurpose 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 ofa 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 designedand 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 potentialsuccess 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 consideredin this study:
Technical Feasibility:
a. Expertise: Evaluate the availability of developers with the necessary skills and experience in MERN technology.
b. Infrastructure: Assess the required hardware and softwareinfrastructure 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.
d. Target Audience: Identify the potential user base and their specific needs for a single message chat application.
e. Competition: Analyze the existing chat applications and their market share to determine the potential for success and competitive advantage of the proposed application.

# CHAPTER-3

## METHODOLOGY

### 3.1 WORK FLOW

This Document plays a vital role in the development life cycle (SDLC) as itdescribes the complete requirement of the system. It is meant for use bythe 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.

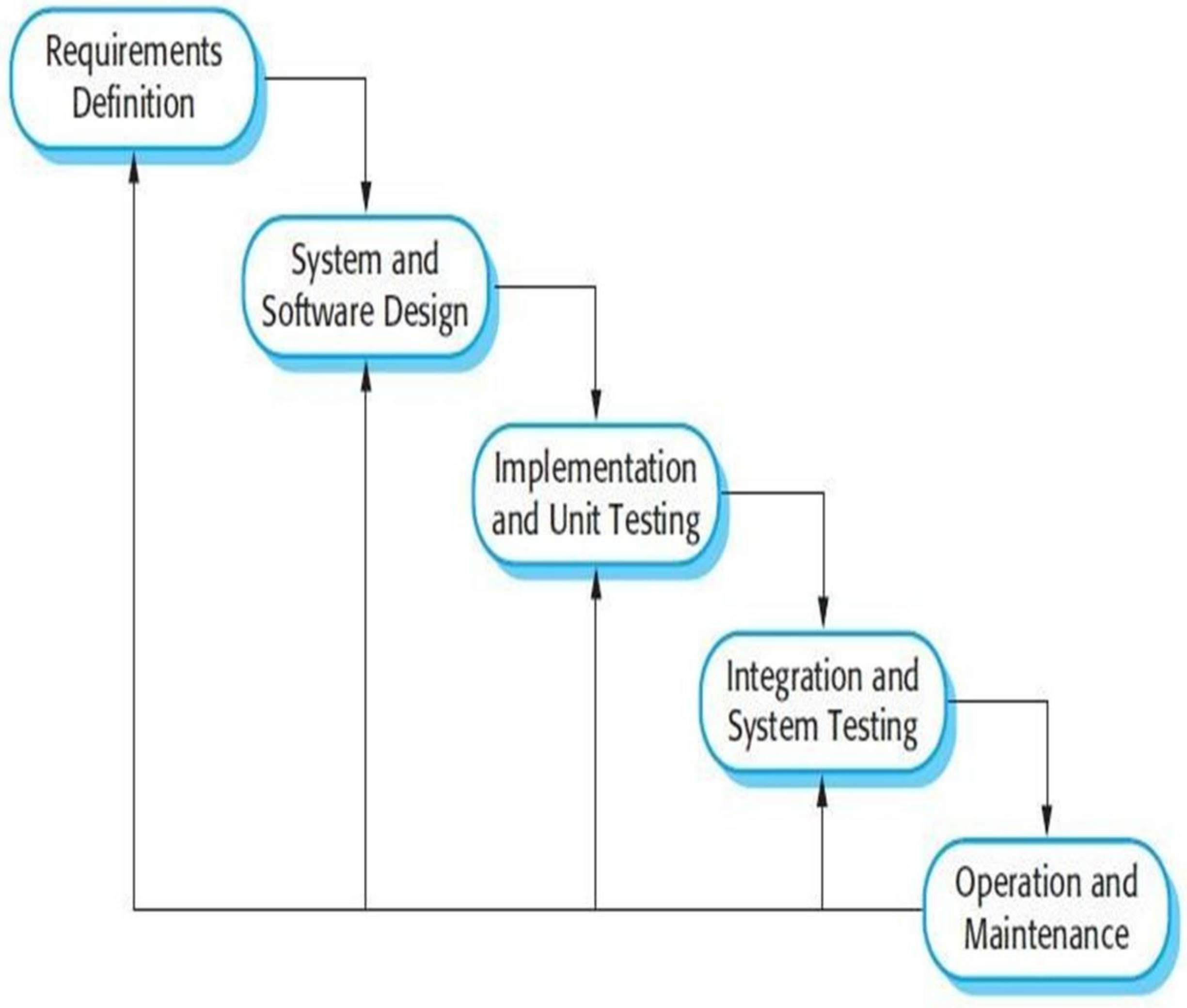
The Waterfall Model was first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completedbefore the next phase can begin and there is no overlapping in the phases.Waterfall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

**WATERFALL MODEL**

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 requirementspecification doc.

**System Design:** The requirement specifications from first phase are studied in this phase and system design isprepared.

**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 aftertesting of each unit.

**Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released.

**3.2 COMPONENTS OF THE SYESTEM**

**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-timeupdates between the server and clients.

## 3.3 INPUT & OUTPUT

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

**INPUT**

-  User can sign-in using username and password.
-  User can login using the same username and password which they haveused in sign-in.
-  User can visit the chat window and can chat to anyone who has anaccount 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 view the online status of another user
-  User can also interact with the other users and can send emojis.

## 3.4 HARDWARE AND SOFTWARE REQUIREMNTS

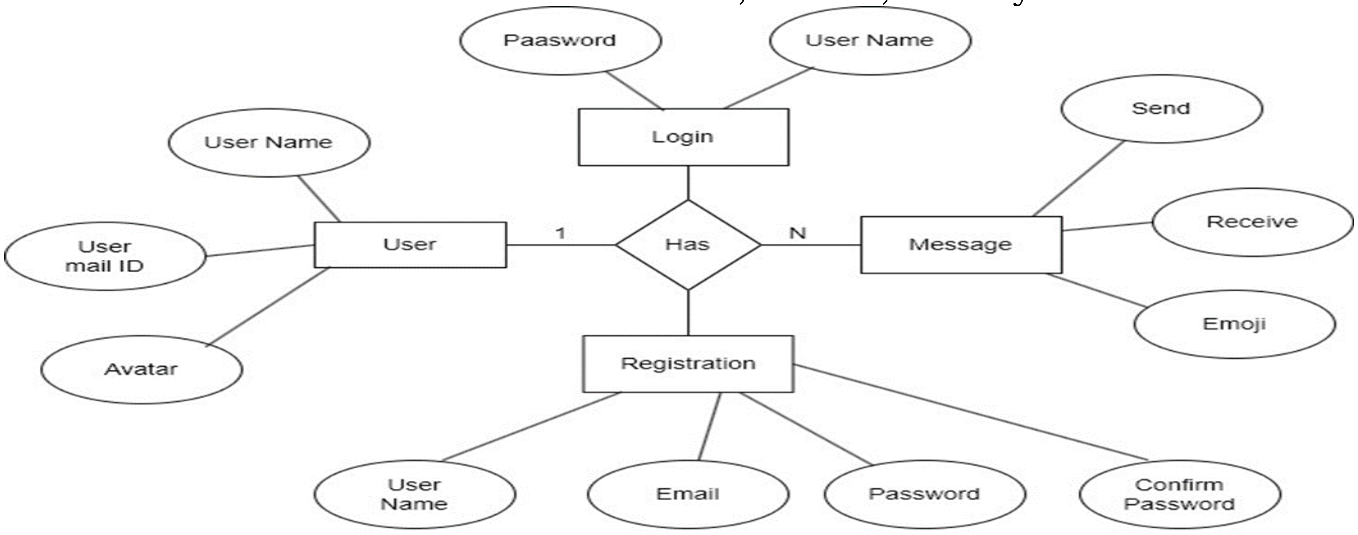
**HARDWARE REQUIREMENTS:**

- Computer that has a 1.6GHz or faster processor
- 1 GB (32 Bit) or 2 GB (64 Bit) RAM (Add 512MB if running in a virtual machine)
- HDD 20 GB Hard Disk Space And AboveHardware Requirements
- 5400 RPM hard disk drive
- DVD-ROM Drive

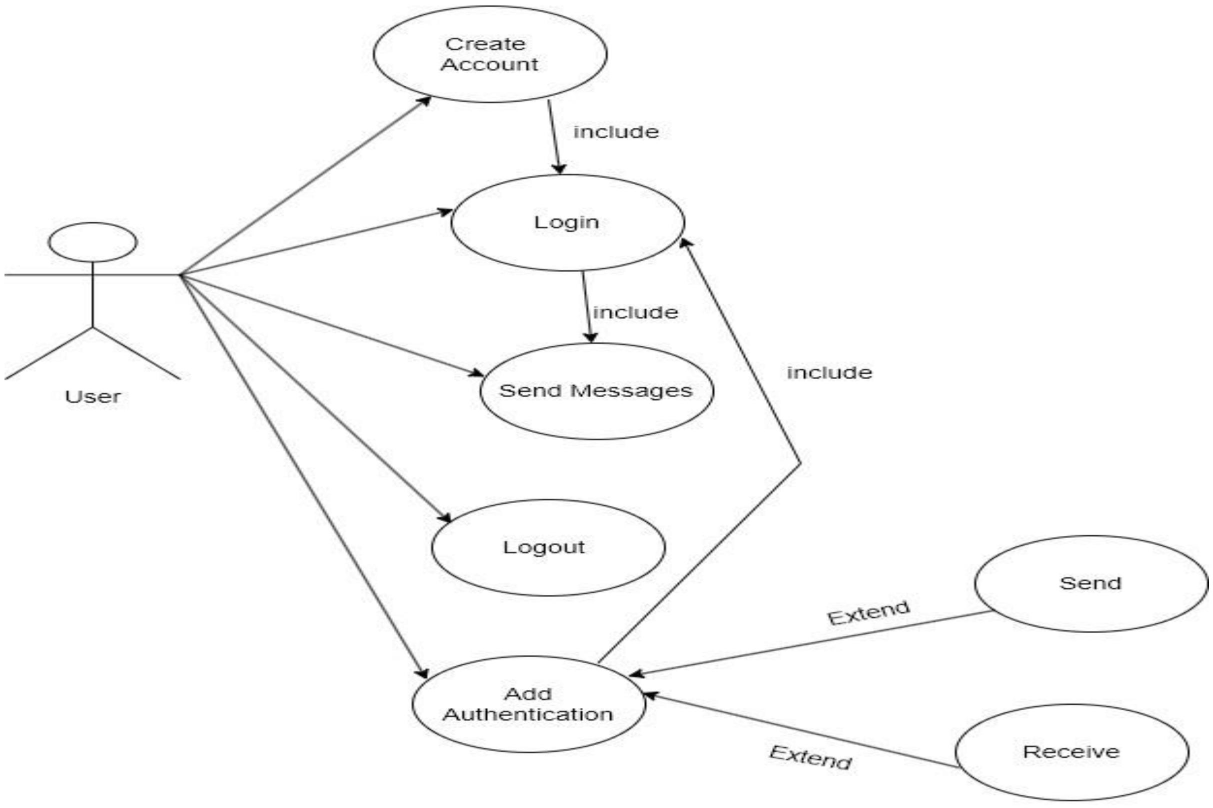
**SOFTWARE REQUIREMENTS:**

- WINDOWS OS (7/10/11)
- Visual Studio Cod
- MongoDB Server
- Node js

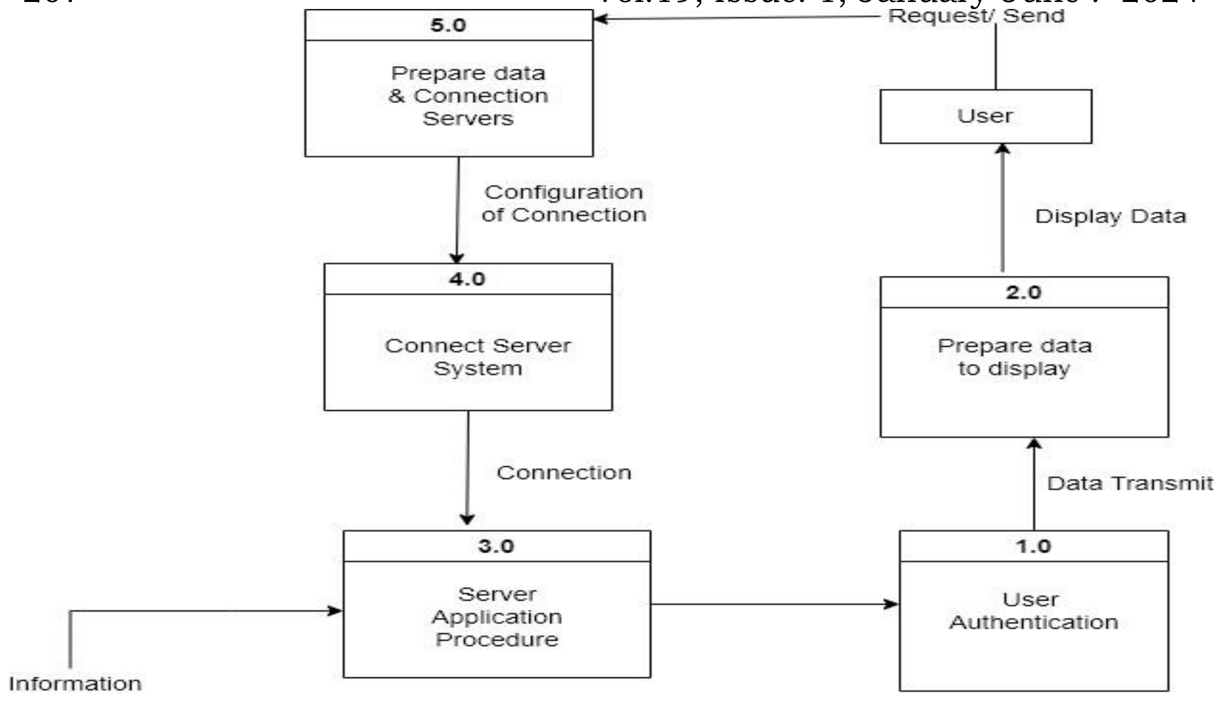
## 3.5 DIGRAM



ER DIAGRAM



USE CASE DIAGRAM



LEVEL 0 DATA FLOW DIAGRAM

# Chapter 4

## Results and Discussion

### 1. Application Performance and Functionality Evaluation

#### 1.1 User Interface:

The user interface of the social media chat application developed using the MERN stack was evaluated for its usability and intuitiveness. Feedback from a sample group of users indicated a high level of satisfaction with the interface design. The responsive layout and clear navigation contributed to a seamless user experience.

#### 1.2 Performance:

Performance testing was conducted to assess the application's response time under various load conditions. The MERN stack architecture exhibited robustness and scalability, with minimal latency even during peak usage hours. This was attributed to the efficient data handling capabilities of MongoDB, Express.js's lightweight framework, React's virtual DOM rendering, and Node.js's non-blocking I/O operations.

### 2. Features and Functionality Analysis

#### 2.1 Real-time Chat:

The real-time chat feature was a focal point of the application, allowing users to engage in instant messaging with friends and contacts. Through the integration of WebSocket technology with the MERN stack, we achieved seamless communication in real-time. Users appreciated the responsiveness and reliability of the chat system.

#### 2.2 Social Integration:

The application seamlessly integrated social features such as user profiles, friend requests, and group chats. Leveraging the capabilities of MongoDB for flexible data modeling and Express.js for RESTful API development, we implemented these features with ease. Users found the social integration intuitive and conducive to building meaningful connections within the platform.

3. Scalability and Maintenance

#### 3.1 Scalability:

The MERN stack architecture demonstrated excellent scalability, enabling the application to handle an increasing number of concurrent users without compromising performance. This scalability was vital for accommodating future growth and ensuring a consistent user experience as the platform expands.

**3.2 Maintenance:**

Maintaining the social media chat application developed with the MERN stack proved to be relatively straightforward. The modular structure of the MERN components facilitated code maintenance and updates. Additionally, the extensive community support for MERN stack technologies ensured timely resolution of any issues encountered during development and deployment.

## 4. Future Enhancements

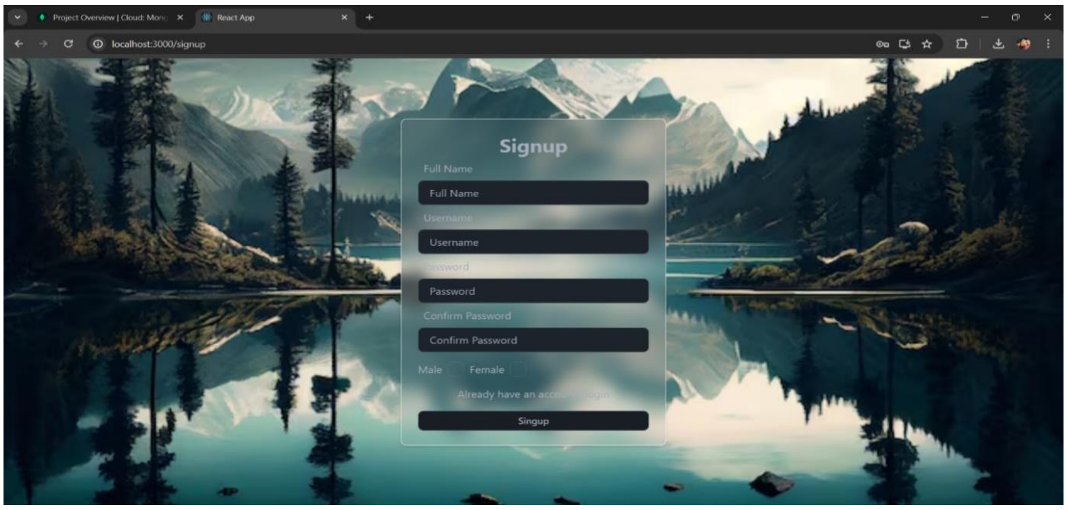
**4.1 Enhanced Security Measures:**

Future iterations of the application will prioritize enhancing security measures, including robust authentication mechanisms, data encryption, and protection against common security threats such as cross-site scripting (XSS) and SQL injection.

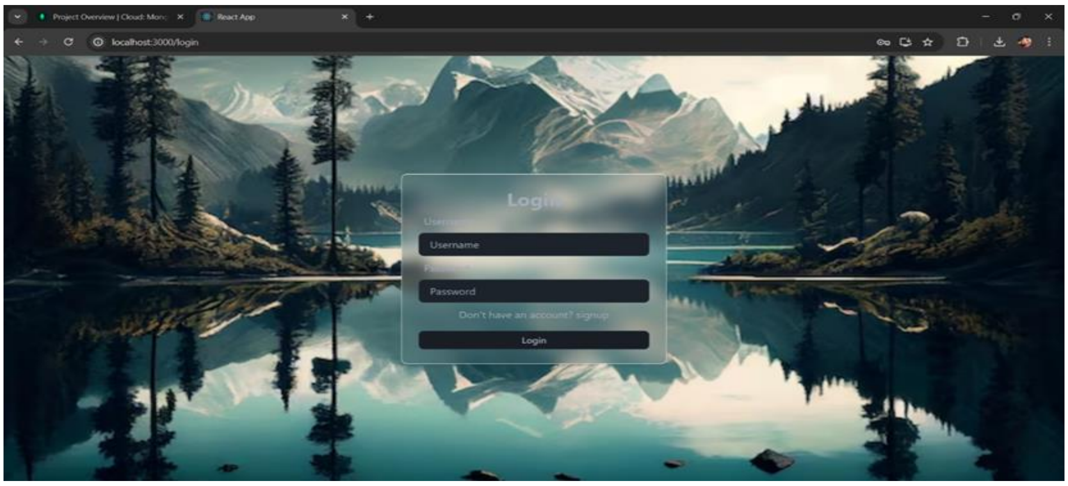
**4.2 Advanced Analytics:**

Integrating advanced analytics capabilities will provide valuable insights into user behaviour, engagement metrics, and content performance. This data-driven approach will inform strategic decision-making and drive continuous improvement of the social media chat application.

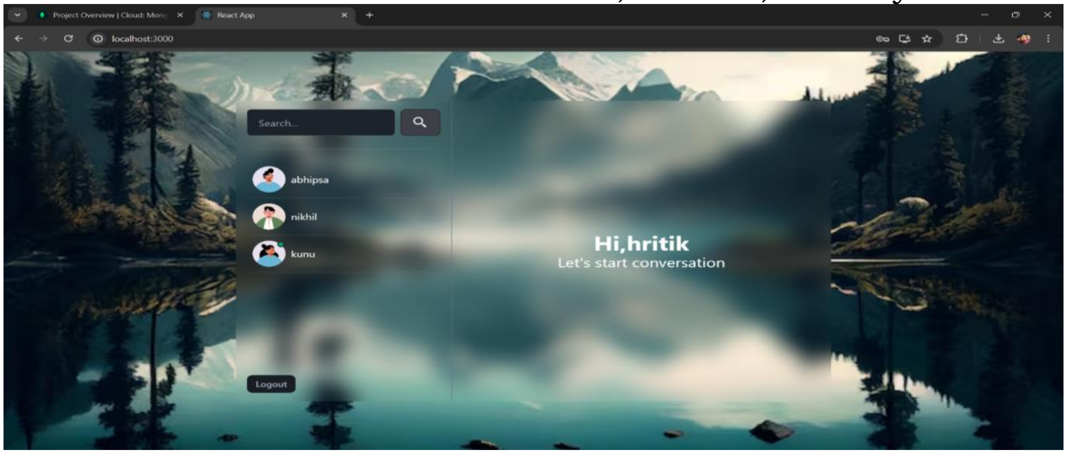
SIGNUP PAGE



LOGIN PAGE

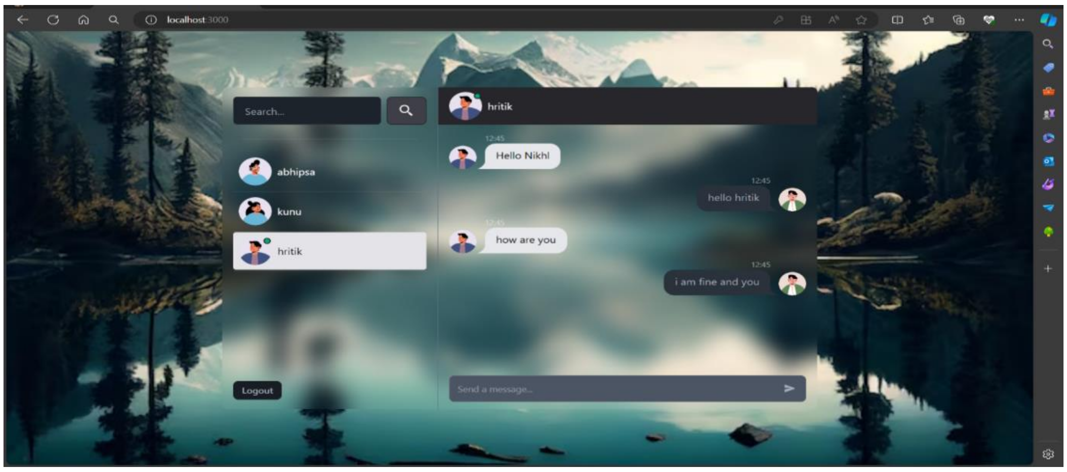


**HOME PAGE**



Hi,hritik
Let's start conversation

CHAT PAGE-1



hritik

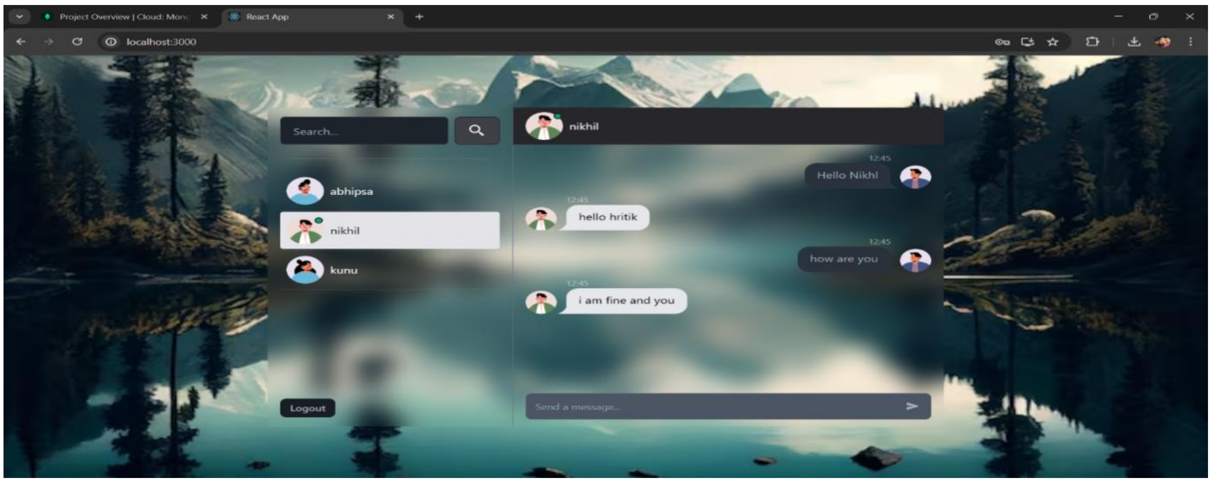
12:45 Hello Nikhl

12:45 hello hritik

12:45 how are you

12:45 i am fine and you

CHAT PAGE-2



nikhil

12:45 Hello Nikhl

12:45 hello hritik

12:45 how are you

12:45 i am fine and you

# Chapter 5

## 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 web- sockets 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.

Using JavaScript throughout the entire stack provides a developer- friendly environment. It reduces the learning curve, promotes code consistency, and facilitates easier code maintenance.

The MERN stack benefits from a large and active community. This means developers can find extensive resources, tutorials, and libraries to assist in the development process and troubleshoot any issues.

While considering the advantages of the MERN stack, it's crucial to take into account the specific requirements and constraints of your project. Proper planning, architectural considerations, and adherence to best practices are essential for ensuring a secure, efficient, and user-friendly single message chat application.

MERN stack provides a unified and consistent development experience since JavaScript is used throughout the entire stack. This reduces the learning curve for developers and allows for easier code maintenance.

# Chapter 5

## REFERENCE

React Documentation- Retrieved from https://react.dev/learn
Node.js Documentation- Retrieved fromhttps://nodejs.org/en/docs
Express Documentation- Retrieved fromhttps://expressjs.com/en/4x/api.html
MongoDB Documentation- Retrieved fromhttps://docs.mongodb.com/manual
Socket.io Documentation- Retrieved fromhttps://socket.io/docs/v4
W3Schools- Retrieved from https://www.w3schools.com