CHATTER BOX-INSTANT CHATS INSTANT BONDS

USING MERN

An Industrial/Practical Training Report

Submitted to the Faculty of Engineering of

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA.

Inpartial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

G. Snehalatha	**(21481A0571)**
J. Keerthi Lahari	**(21481A0584)**
K. Devi Durga Bhavani	**(21481A0593)**
G. Hemanth	**(21481A0573)**

Under the Enviable and Esteemed Guidance of

DR. M. BABU RAO, M.Tech, Ph.D.

Professor, Department of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

SESHADRIRAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521356

ANDHRA PRADESH

2024-25

SESHADRI RAO
GUDLAVALLERU ENGINEERING COLLEGE

(AnAutonomous Institute with Permanent Affiliationto JNTUK, Kakinada)
SESHADRI RAO KNOWLEDGE VILLAGE, GUDLAVALLERU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"Chatter Box - Instant Chats Instant Bonds using MERN"** is a bonafide record of work carried out by **G. Snehalatha (21481A0571), J. Keerthi Lahari (21481A0584), K. Devi Durga Bhavani (21481A0593), G. Hemanth (21481A0573)** under the guidance and supervision of **Dr.M.BABU RAO** in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University Kakinada, Kakinada during the academic year 2024-25.

Project Guide **Headof the Department**

(DR.M.BABU RAO) **(Dr. M. Babu Rao)**

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and sincere thanks to **Dr. M. Babu Rao**, Department of Computer Science and Engineering for his constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. M. Babu Rao, Head of the Department**, Computer Science and Engineering for his encouragements all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like to take this opportunity to thank our beloved principal **Dr. B. Karuna Kumar** for providing a great support for us in completing our project and giving us the opportunity for doing project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff and our friends, who had directly or indirectly helped and supported us in completing our project in time.

Team members

G. Snehalatha (21481A0571)
J. Keerthi Lahari (21481A0584)
K. Devi Durga Bhavani (21481A0593)
G. Hemanth (21481A0573)

INDEX

ABSTRACT

This project focuses on the development of a real-time chat application leveraging the MERN stack—MongoDB, Express.js, React.js, and Node.js—alongside Socket.IO for instant communication. The goal is to deliver a responsive, secure, and dynamic messaging platform suitable for both individuals and groups, integrating modern technologies for seamless cross-device interaction.

The application features secure user authentication, enabling registration, login, and session management with encrypted passwords to ensure data privacy. Users can customize their profiles, including avatar selection, and exchange messages in real time with features like delivery status indicators (sent, delivered, and seen). Socket.IO facilitates instant and bidirectional communication, with plans for future enhancements such as group chat support.

The backend employs RESTful APIs for efficient communication between the client and server. MongoDB ensures scalable and high-performance data storage, while Express.js handles routing and server-side logic. Node.js supports asynchronous operations, enabling the system to manage multiple connections simultaneously. Socket.IO powers real-time data exchange, ensuring a responsive user experience.

On the frontend, a React.js-based mobile-first design ensures compatibility across devices and provides an intuitive user interface. The responsive layout prioritizes accessibility, making navigation seamless for both desktop and mobile users. Reusable components and services simplify updates and maintenance.

Security measures include encryption, input validation, and protection against vulnerabilities such as XSS, CSRF, and SQL injection. Performance optimization strategies, such as database indexing and streamlined WebSocket connections, ensure the application can handle high traffic and concurrent users efficiently.

The application is designed for extensibility, with potential upgrades including multimedia sharing, voice and video calling, AI-driven chatbots, and advanced analytics. These enhancements will ensure the platform evolves to meet user needs and technological advancements.

In conclusion, this real-time chat application highlights the power of modern web development frameworks. By integrating the MERN stack with Socket.IO, it provides a scalable and secure communication solution, showcasing the principles of full-stack development while remaining adaptable for future improvements.

CHAPTER-1

INTRODUCTION

1.1 Overview

The real-time chat application is designed to provide seamless and instantaneous communication between users. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), the project leverages modern web technologies to deliver a responsive, scalable, and user-friendly platform. Real- time messaging is achieved through Socket.IO, enabling users to exchange messages instantly without page refreshes.

Key features of the application include user authentication, profile management with avatar selection, and a responsive chat interface that ensures usability across devices. The backend is responsible for handling RESTful API requests, securely managing user data, and facilitating real-time communication through WebSocket connections. The frontend, developed with React.js, focuses on delivering a dynamic and engaging user experience.

This project serves as a foundation for advanced features such as group chats, multimedia sharing, and video call integration. By emphasizing modularity, scalability, and responsiveness, the application demonstrates the practical application of full-stack development in real-world scenarios. It highlights how modern frameworks and technologies can be combined to create efficient and reliable communication tools for diverse use cases.

1.1 PURPOSE

The purpose of this project is to develop a scalable and efficient real-time chat application that leverages the MERN stack (MongoDB, Express.js, React.js, Node.js) and Socket.IO to provide seamless instant communication. The application aims to address the growing need for responsive and reliable communication platforms by enabling core features such as user registration, secure login, profile management, and real-time messaging. By focusing on modularity and scalability, the project sets a strong foundation for future enhancements like group chats, video calls, and multimedia sharing, making it adaptable for both personal and professional communication needs. Through this implementation, the project seeks to demonstrate the integration of modern web technologies to create user-friendly, high-performing, and secure communication systems.

CHAPTER-2
LITERATURE SURVEY

2.1 Existing Problem:

The development of real-time communication platforms comes with several challenges that need to be addressed to ensure functionality, scalability, and user satisfaction. The existing problems in this domain can be categorized as follows:

1. Scalability Issues

Traditional chat systems often face challenges in scaling to accommodate a growing number of users. Managing concurrent connections, especially in real-time applications, can overwhelm the server infrastructure if not designed for scalability.

2. Latency in Communication

Ensuring low-latency message delivery is critical for real-time applications. Delays in message transmission can disrupt the user experience, especially in scenarios requiring immediate feedback.

3. Data Management and Security

Handling user data, chat histories, and profile information securely is a significant concern. Improper data encryption or storage can lead to data breaches, compromising user privacy.

4. Cross-Platform Compatibility

Providing a consistent user experience across various devices and screen sizes is a challenge. Designing responsive user interfaces and managing compatibility with different browsers is essential for user satisfaction.

5. Real-Time Event Handling

Managing real-time events like message delivery, read receipts, and typing indicators requires efficient and reliable event handling mechanisms. Inadequate implementations can lead to synchronization issues.

6. High Development and Maintenance Costs

Traditional approaches often involve complex setups and integrations, increasing development time and costs. Maintaining such systems becomes challenging as new features or updates are added.

7. Integration of Advanced Features

Real-time chat systems often lack advanced functionalities like multimedia sharing, group chats, and video/audio calls. Adding these features can introduce complexities in design and resource management.

8. Server Downtime and Reliability

Ensuring the server's uptime is crucial for uninterrupted communication. However, server downtime due to high loads or improper infrastructure can affect application reliability.

By addressing these challenges, the real-time chat application developed in this project aims to provide a scalable, secure, and user-friendly solution for seamless communication.

2.2 Proposed Solution:

The project proposes a real-time chat application using the MERN stack (MongoDB, Express.js, React.js, Node.js) integrated with Socket.IO for efficient, low- latency communication. The solution ensures scalability to handle numerous concurrent users and implements secure storage of user credentials with encrypted data. Real-time messaging is facilitated through WebSocket connections, while the responsive React- based frontend provides a consistent user experience across devices. The modular architecture allows for easy integration of advanced features like group chats and multimedia sharing. By addressing key challenges, the solution offers a reliable and user- friendly communication platform.

CHAPTER-3

THEORETICAL ANALYSIS

The Chatterbox application is a real-time chat solution that leverages the MERN stack—MongoDB, Express.js, React.js, and Node.js—along with Socket.IO for instant communication. The theoretical foundation of this application revolves around ensuring efficient data flow, scalability, and seamless user interaction. This analysis explores the underlying principles and design considerations that make Chatterbox a robust and dynamic messaging platform.

MongoDB: Data Flexibility and Scalability

MongoDB, as a NoSQL database, provides a document-oriented structure that supports dynamic schema designs. This flexibility is critical for managing diverse user data, message histories, and real-time updates. MongoDB's scalability enables the system to handle large datasets and high-frequency operations typical in chat applications, ensuring uninterrupted service even under heavy load.

Express.js: Middleware and API Layer

Express.js serves as the middleware, bridging the frontend and backend by facilitating secure and efficient data exchange. The use of RESTful APIs ensures modularity and adherence to standard communication protocols. Express.js is also responsible for routing, handling user authentication, and managing session data, forming the backbone of the application's server-side logic.

React.js: Dynamic and Interactive Frontend

React.js powers the user interface with a component-based architecture, ensuring reusability and ease of maintenance. Its virtual DOM enhances performance by minimizing unnecessary re-rendering, creating a responsive and fluid user experience. React's declarative syntax simplifies UI development, making features like real-time notifications and message threading intuitive and user-friendly.

Node.js: Non-Blocking Architecture

Node.js underpins the backend with its event-driven, non-blocking I/O model. This architecture enables Chatterbox to handle thousands of concurrent connections efficiently. Node.js's asynchronous nature ensures that server performance remains high, even when managing multiple simultaneous data streams.

Socket.IO: Real-Time Communication

Socket.IO is the cornerstone of the real-time messaging feature in Chatterbox. It establishes persistent WebSocket connections between clients and the server, enabling instantaneous message delivery. Socket.IO's event-driven communication model ensures low latency and supports bidirectional data flow, making real-time interactions smooth and reliable.

Security and Performance Considerations

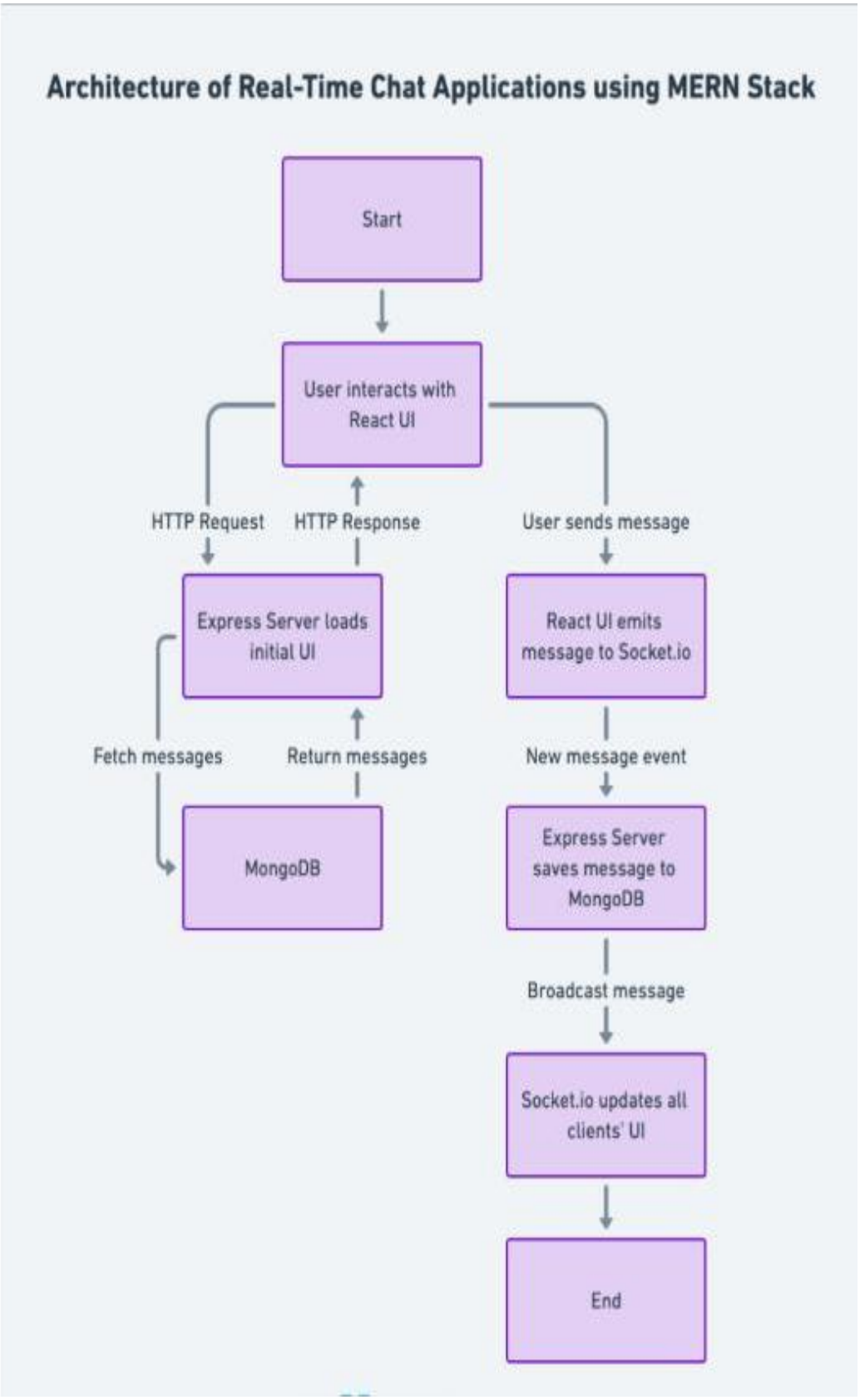
Chatterbox employs robust security measures, including encrypted data transmission, secure authentication protocols, and protection against common vulnerabilities like XSS and CSRF attacks. Performance optimizations, such as load balancing and caching mechanisms, ensure that the application remains fast and responsive, even during peak usage.

Scalability and Future Enhancements

The modular design of Chatterbox allows for easy scalability. Features like multimedia messaging, group chats, and AI-driven chatbots can be seamlessly integrated into the existing framework. The system's adaptability ensures that it can evolve to meet emerging user needs and technological advancements.

In conclusion, Chatterbox represents a practical implementation of modern web development principles. By combining the strengths of the MERN stack with the real-time capabilities of Socket.IO, it delivers a robust, scalable, and user-centric chat application that exemplifies the potential of full-stack development

3.1 Technical Architecture:



3.1.1 Technical Architecture

The architecture of a real-time chat application built with the MERN stack (MongoDB, Express.js, React.js, Node.js) integrated with Socket.IO for instant communication. The workflow begins with users interacting with the React-based UI, which serves as the frontend for initiating and receiving messages.

For initial UI rendering, React sends an HTTP request to the Express server. The server retrieves the required data, such as previous chat messages, from MongoDB, a NoSQL database used for efficient and scalable data storage. This data is returned to the React UI, ensuring that users can view historical messages seamlessly.

When a user sends a message, the React UI emits the message to Socket.IO, which handles real-time communication. The Express server listens for these new message events and saves the messages to MongoDB for persistence. Once saved, the server broadcasts the new message to all connected clients using Socket.IO. This ensures that all users receive real-time updates in their chat interface.

The architecture leverages the asynchronous, event-driven nature of Node.js to manage multiple concurrent connections efficiently. By integrating these technologies, the system ensures a fast, dynamic, and reliable chat experience with scalability, modularity, and real-time interaction as core principles.

CHAPTER-4

PROJECT STRUCTURE AND CODE

4.1 Project Structure:

```
chat-app-react-nodejs/  # Root directory of the project
├── public/  # Contains static assets like images, fonts, and index.html for React
│   └── (public assets, e.g., favicon.ico, logo.png, etc.)
├── src/  # Source code for the React frontend
│   ├── pages/  # Contains React components for different routes/pages
│   │   ├── Chat.jsx  # Chat page component for user messaging
│   │   ├── Login.jsx  # Login page component for user authentication
│   │   ├── Register.jsx  # Registration page component for new users
│   │   └── SetAvatar.jsx  # Page for users to set their avatar/profile picture
│   ├── utils/  # Utility files (e.g., helper functions, API calls)
│   │   └── (utility files, if any)
│   ├── App.js  # Main React component defining application routes
│   ├── index.css  # Global CSS styles for the application
│   └── index.js  # Entry point for the React application
├── server/  # Backend code using Node.js and Express.js
│   ├── controllers/  # Contains business logic for handling API requests
│   │   └── (e.g., userController.js, chatController.js)
│   ├── models/  # Defines database models/schemas (e.g., MongoDB, SQLite)
│   │   └── (e.g., userModel.js, messageModel.js)
│   ├── routes/  # Defines API endpoints and maps them to controllers
│   │   └── (e.g., userRoutes.js, chatRoutes.js)
├── .env  # Environment variables (e.g., API keys, database credentials)
├── .env.example  # Example environment variables file for documentation
├── .gitignore  # Specifies files and directories to ignore in Git
├── Dockerfile  # Instructions to build a Docker container for the app
├── docker-compose.yml  # Configures multi-container Docker applications
├── package.json  # Project metadata and dependencies for Node.js
├── package-lock.json  # Lockfile for npm dependencies
├── README.md  # Documentation for the project
├── yarn.lock  # Lockfile for Yarn dependencies
```

The project follows a modular architecture, separating concerns between customer-facing and admin-facing components. The **src/app/components** folder contains reusable components for customer features like registration, login, home, accounts, balance, transaction history, feedback, and more. These components are user-focused and designed to provide a seamless customer experience for various functionalities, such as checking balances or viewing transaction details.

The **src/app/modules** directory houses feature-specific modules, each encapsulating a set of components for particular sections of the app. For example, the admin module includes components for tasks like managing balances, updating user accounts, monitoring feedback, and handling payments. This modular approach ensures the scalability and maintainability of the application by isolating functionalities into distinct modules.

Routing for the application is defined in **app-routing.module.ts**, where navigation paths are mapped to their respective components or modules. The root component (app.component.ts and app.component.html) serves as the main container, orchestrating the high-level structure and layout of the app. This modular design, along with lazy loading of modules, optimizes performance and simplifies development.

4.2 Code:

App.js

```
import React from "react";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import SetAvatar from "./components/SetAvatar";
import Chat from "./pages/Chat";
import Login from "./pages/Login";
import Register from "./pages/Register";
export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/register" element={<Register />} />
        <Route path="/login" element={<Login />} />
        <Route path="/setAvatar" element={<SetAvatar />} />
        <Route path="/" element={<Chat />} />
      </Routes>
    </BrowserRouter>
  );
}
```

CHAPTER 5

PROPOSED METHOD

3.1 Proposed Method

The development of the real-time chat application follows a systematic approach to ensure a scalable, secure, and efficient communication platform. The proposed method is divided into the following stages:

1. System Architecture

The application is built using a client-server architecture where:

- **Client (Frontend)**: React.js is used to create a responsive user interface that handles user interactions and communicates with the backend via REST APIs and WebSocket connections.
- **Server (Backend)**: Node.js and Express.js manage business logic, user authentication, and routing. Socket.IO handles real-time communication.
- **Database**: MongoDB stores user data (e.g., login credentials, profile details) and chat messages.

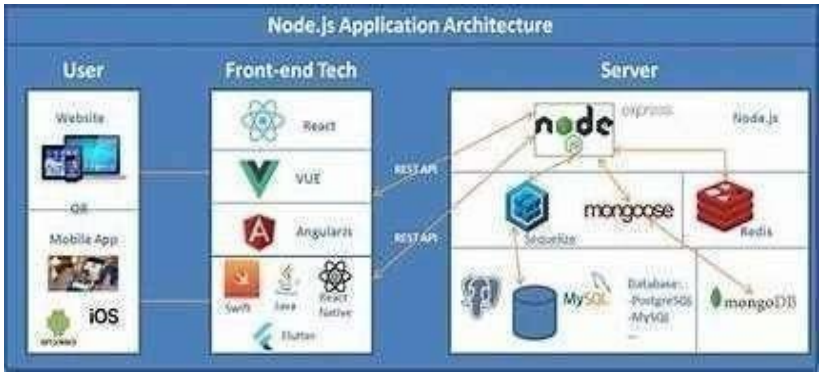


Fig: Node.js Architecture

2. Module Design

a. User Authentication

- Allows users to register and log in securely using encrypted credentials.
- Implements JSON Web Tokens (JWT) for session management.

b. Real-Time Messaging

- Uses Socket.IO to establish WebSocket connections for instantaneous message delivery between users.
- Features include live typing indicators, message delivery statuses, and real-time notifications.

c. Profile Management

- Users can set profile avatars to personalize their chat experience.
- Avatar data is stored and retrieved from the database.

d. Responsive Design

- The frontend is designed with responsive layouts to ensure compatibility across various devices and screen sizes.

3. Workflow Process

Step 1: User Registration/Login

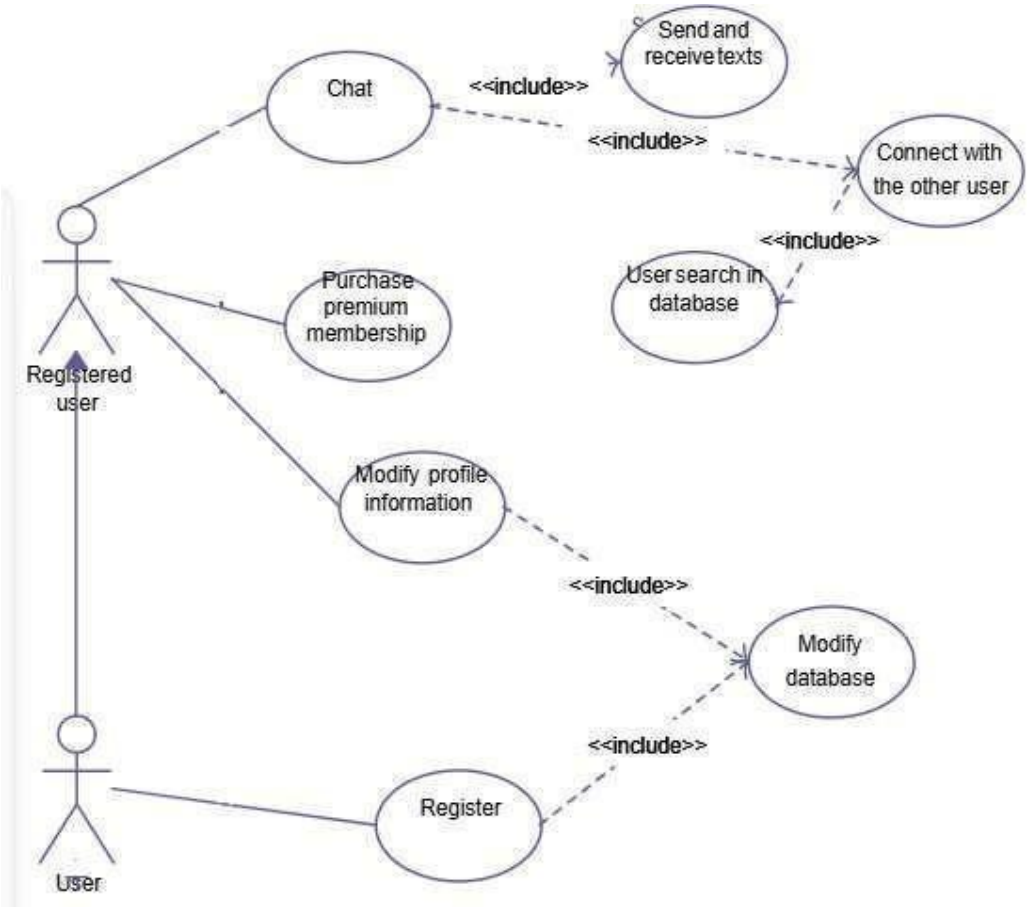
- User inputs data via the frontend.
- Data is sent to the backend for validation and stored in MongoDB upon successful registration.

Step 2: Real-Time Messaging

- Upon login, the user establishes a WebSocket connection to the server using Socket.IO.
- Messages are sent via WebSocket, saved in the database, and broadcasted to the recipient in real time.

Step 3: Data Handling

- MongoDB collections are used for:
    - **Users Collection**: Storing user credentials and profile details.
    - **Messages Collection**: Storing chat messages with timestamps.



3.2 Implementation

1. Frontend

Technologies: HTML, CSS, JavaScript, and a frontend framework/library like React, Vue, or Angular.

Key Features:

- **User Interface**: Design a clean and user-friendly interface for users to send and receive messages.
- **Real-Time Updates**: Implement real-time updates so that messages appear instantaneously.

2. Backend

Technologies: Node.jswith Express.js, Python with Django or Flask, or any other server- side language/framework.

Key Features:

- **APIs**: Develop APIs for sending, receiving, and storing messages.
- **WebSocket Implementation**: Use WebSocket for real-time communication.

3. Database

Technologies: MongoDB, PostgreSQL, MySQL, etc.

Key Features:

- **User Data**: Store user information like usernames, passwords, profile pictures, etc.
- **Message Data**: Store messages with metadata like timestamp, sender, receiver, etc.

4. Authentication and Authorization

Technologies: JWT (JSON Web Tokens), OAuth, etc.

Key Features:

- **User Registration and Login**: Allow users to create accounts and log in securely.
- **Session Management**: Keep track of user sessions.

Commands to Run

Set Up the Backend:

Go to the backend folder:

"cd D:\chat-clone\chat-app-react-nodejs\server"

Install backend dependencies:

"npm install"

The server will run at: "http://localhost:5000"

Set Up the Frontend:

Go to the frontend folder:

"cd D:\chat-clone\ chat-app-react-nodejs\public"

Install frontend dependencies:

"npm install"

Start the frontend server:

"npm start"

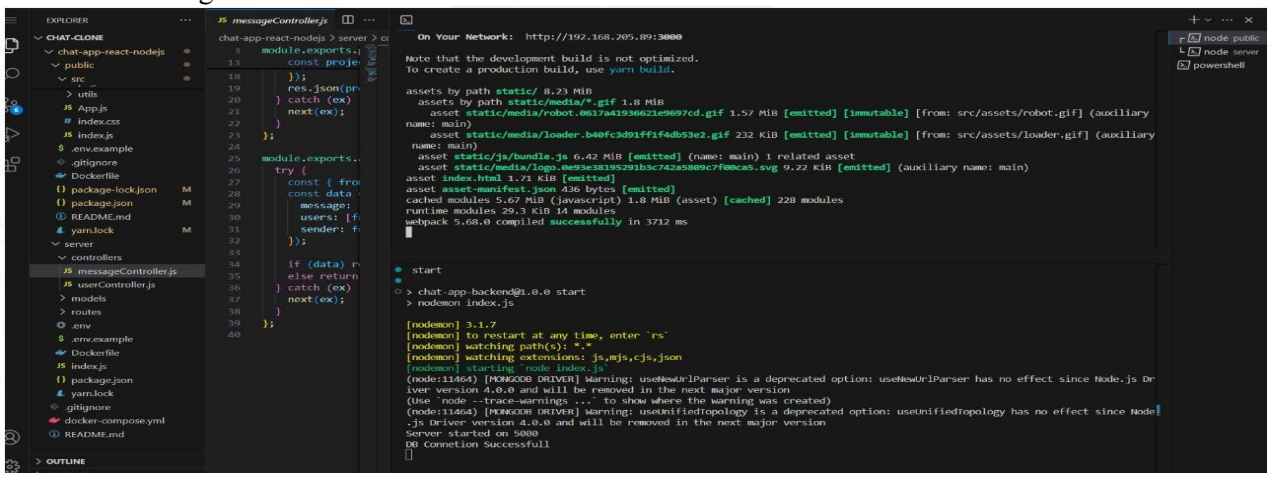
Open your browser and visit "http://localhost:3000" to see the app.

CHAPTER-6

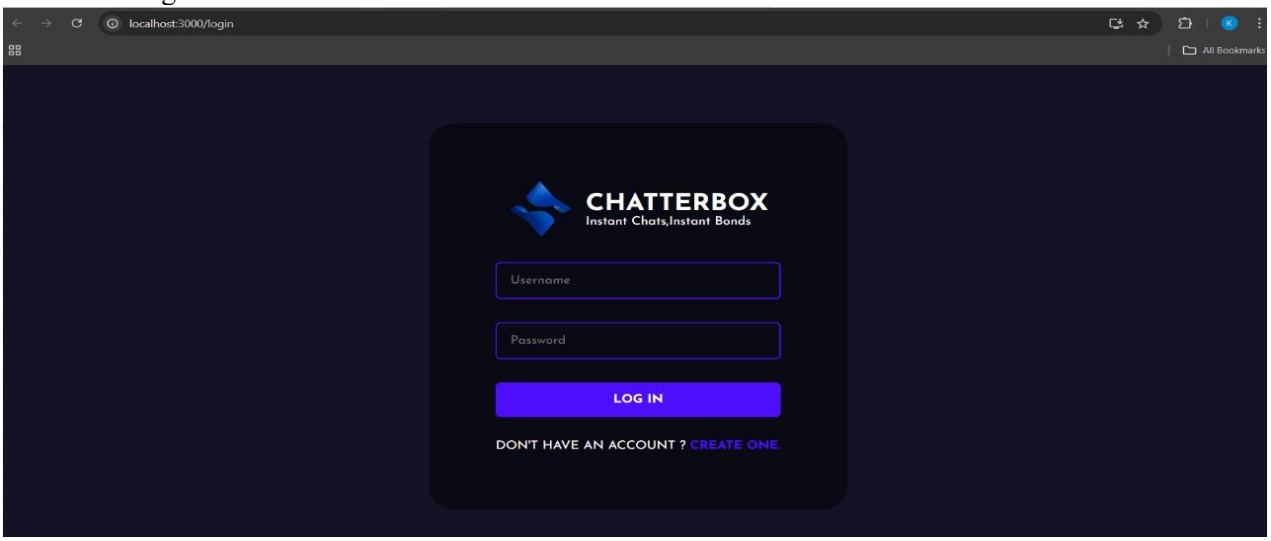
RESULTS

Final Output of the Project

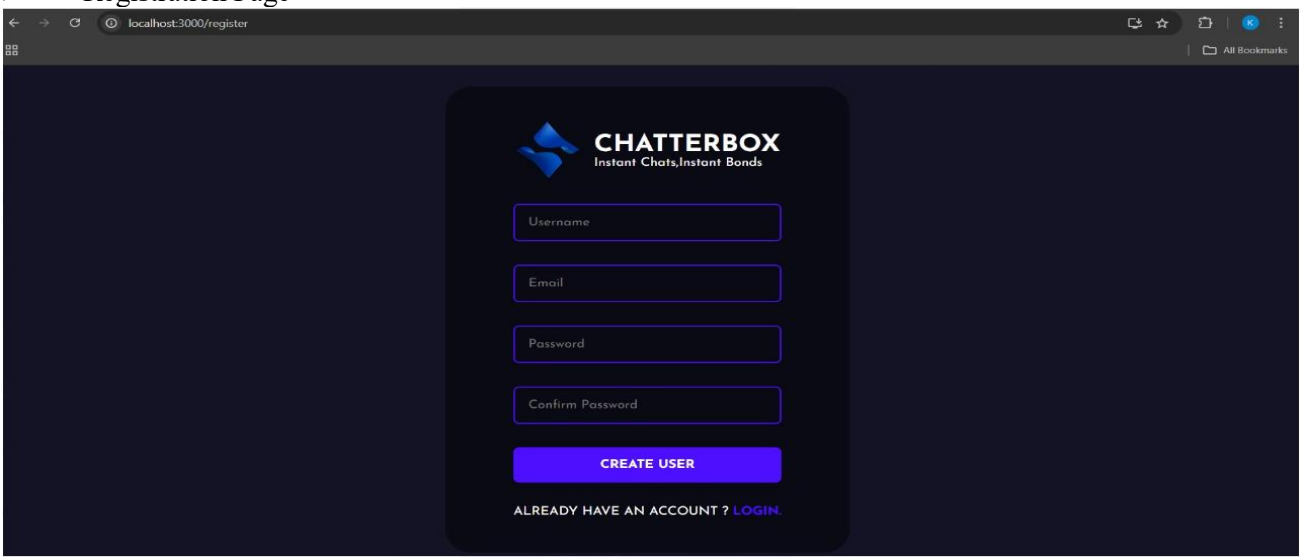
- After running both frontend and backendl



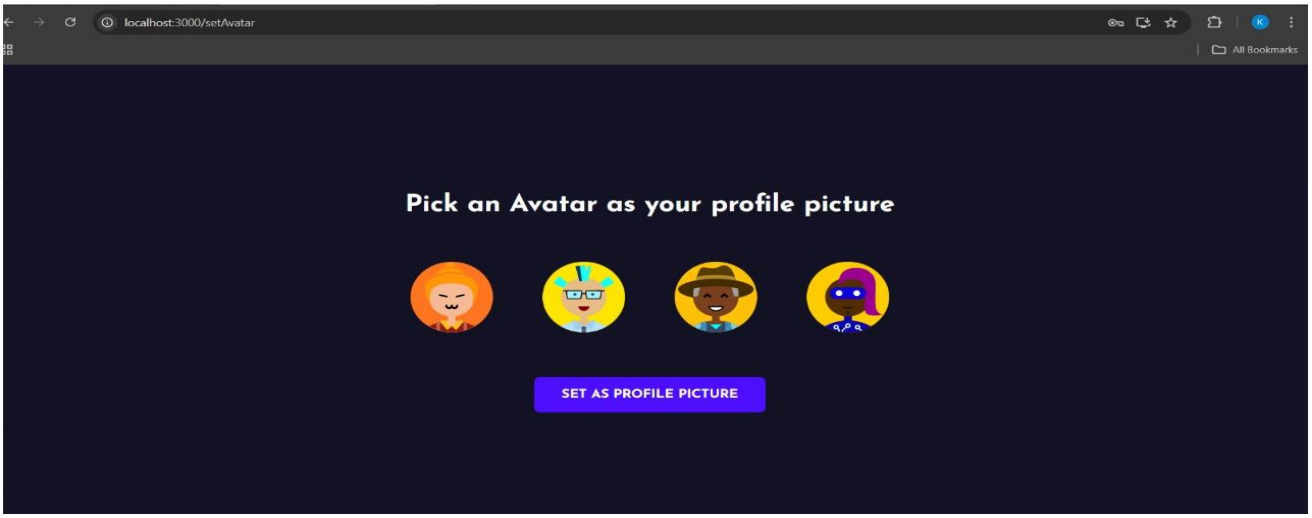
- Home Page



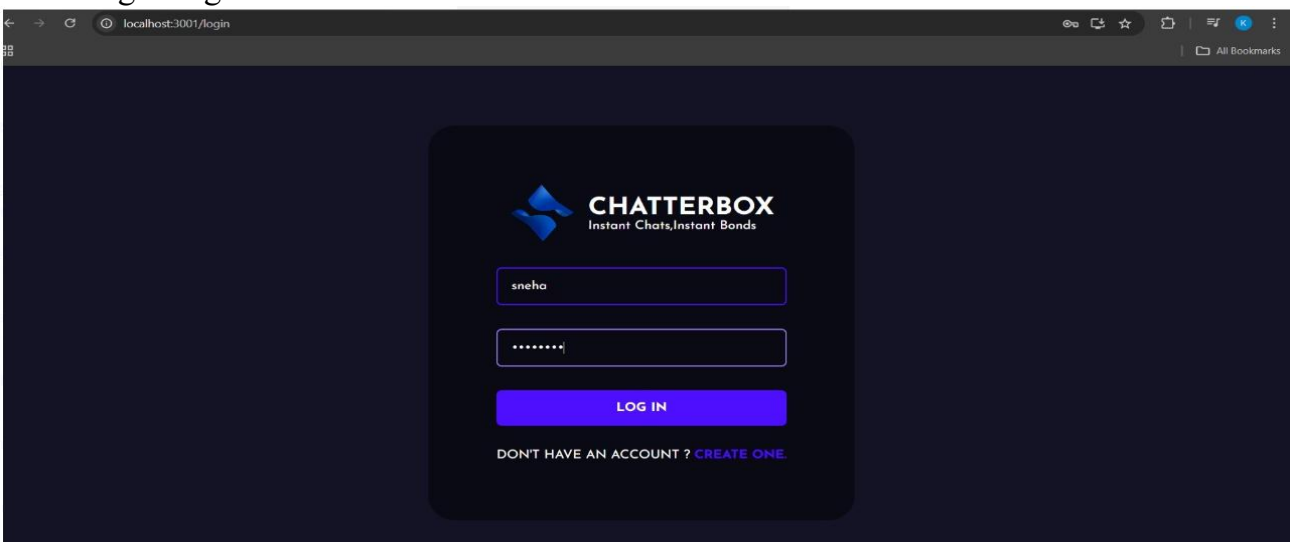
- Registration Page



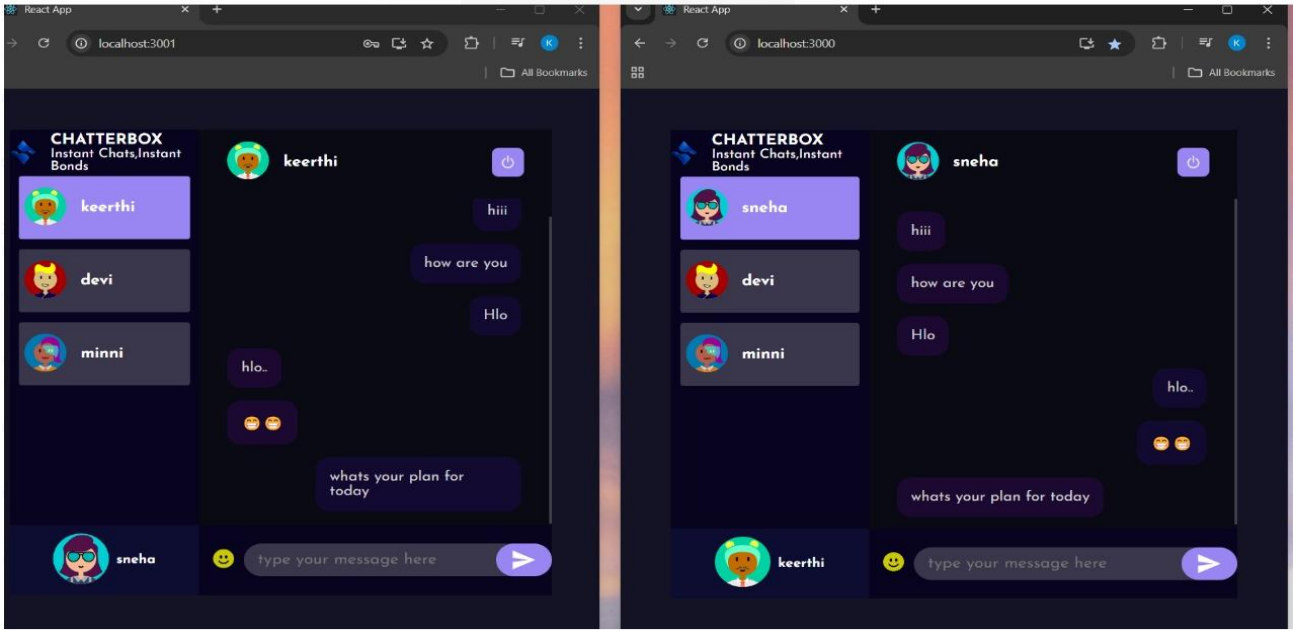
- Avatar selection



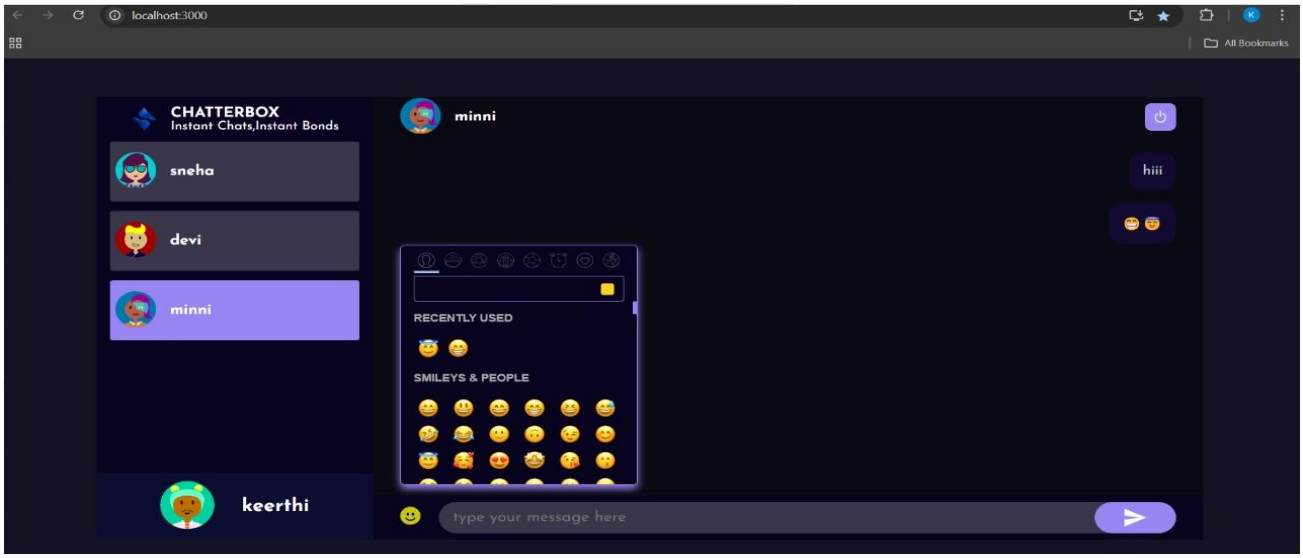
- Login Page:



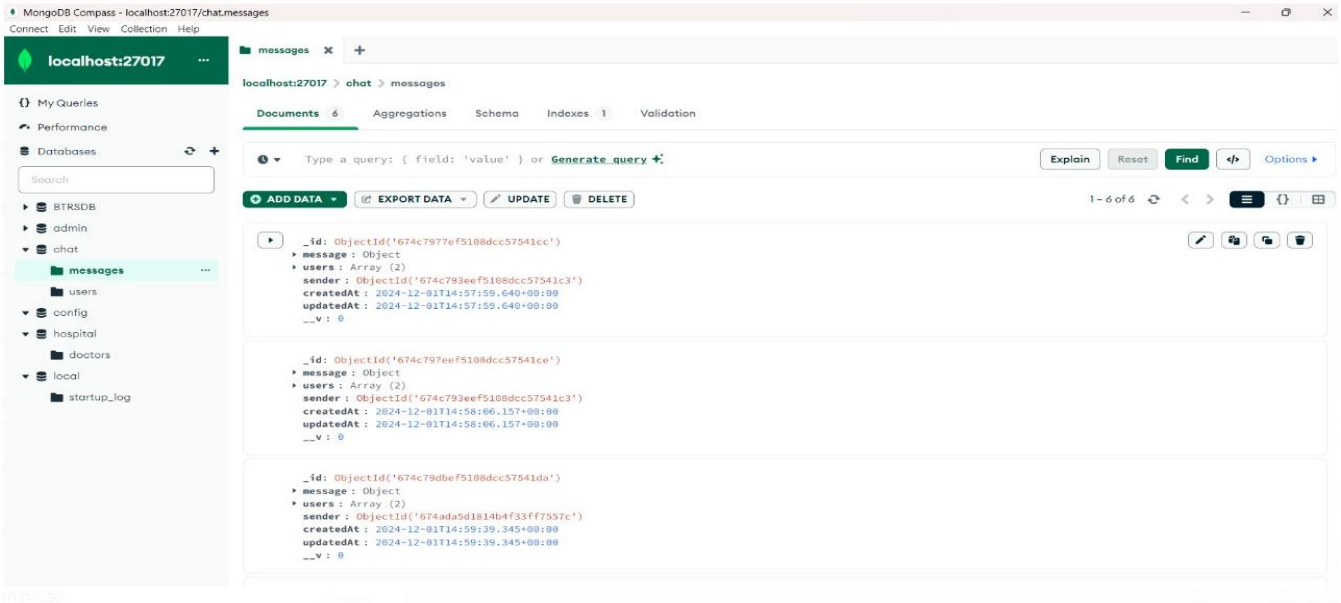
- Messages sent and received



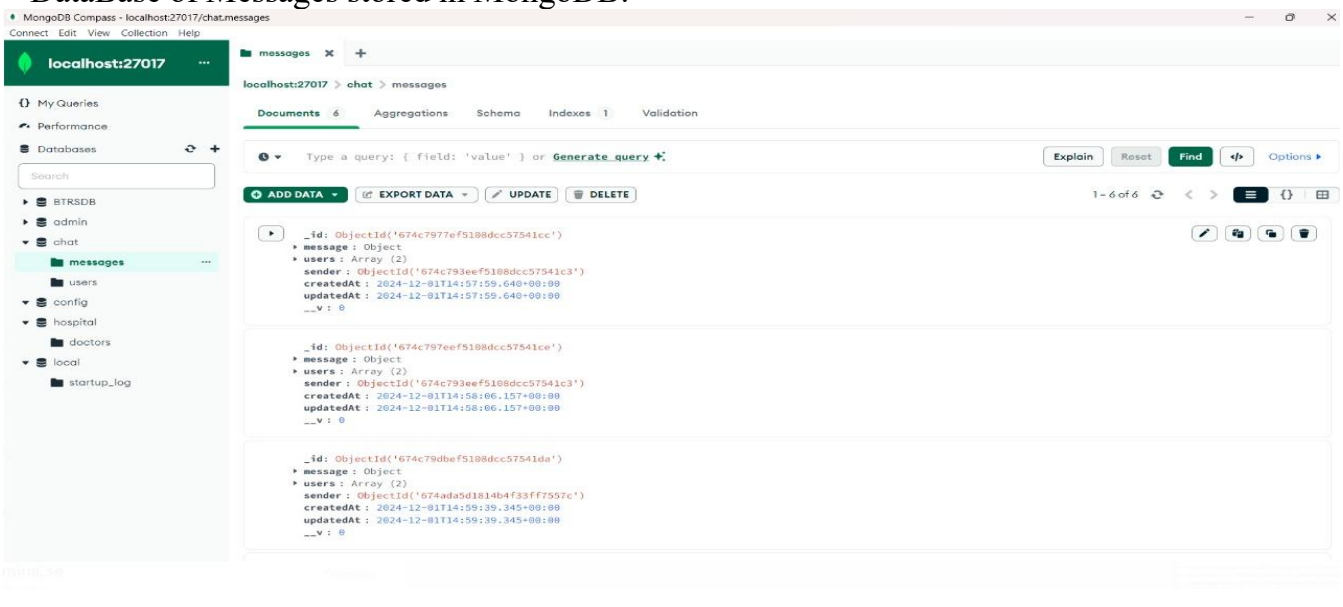
- Chat with Emoji's:



- DataBase of Users stored in MongoDB:



- DataBase of Messages stored in MongoDB:



CHAPTER-7
APPLICATIONS

Real-Time Chat Platforms: This can be used for personal messaging and group chats. Users can communicate instantly in private messages or public rooms, enabling efficient and seamless interactions in real-time.

Customer Support System: Businesses can integrate live chat for customer service. Chatbots handle initial inquiries, while human agents provide more detailed assistance, ensuring quick responses and better customer satisfaction.

Social Networking: Provides a real-time chat feature for social networking platforms. Users can interact with friends, join conversations, and share updates or content instantly, enhancing the social experience.

E-commerce Customer Communication: Customers can instantly reach out to sellers or support teams to inquire about products, track orders, or resolve issues. It streamlines communication and enhances the customer shopping experience.

Legal and Consultation Services: Clients can securely message legal advisors for consultations, document reviews, or legal advice in real time. It allows quick responses and ongoing communication for a smooth client experience.

Real-Time Customer Feedback: Allows businesses to collect instant feedback from customers. After a service or product purchase, customers can chat with the business to provide reviews or suggest improvements, helping to improve products and services continuously.

CHAPTER 8
CONCLUSION & FUTURE SCOPE CONCLUSION

CONCLUSION

In conclusion, a MERN-based real-time chat application offers immense versatility, providing robust solutions across various sectors such as customer support, healthcare, e-commerce, education, and more. Its ability to facilitate seamless communication, whether for personal, professional, or service-based interactions, enhances user engagement and streamlines processes. By integrating technologies like WebSocket and MongoDB, it ensures fast, secure, and scalable communication, making it an invaluable tool for businesses and organizations seeking to improve real-time connectivity, customer experience, and operational efficiency.

FUTURE SCOPE

The future scope of a MERN-based real-time chat application is vast, with opportunities to integrate advanced technologies such as artificial intelligence (AI) for automated support, natural language processing (NLP) for intelligent chatbots, and machine learning for predictive analytics in conversations. As virtual and augmented reality (VR/AR) technologies continue to evolve, there is potential for immersive chat experiences that blend real-time communication with interactive virtual environments. Additionally, further advancements in security, such as end-to-end encryption and blockchain for privacy, will make these platforms more reliable and trustworthy, fostering widespread adoption across industries like healthcare, finance, and education. The continued evolution of cloud computing will also ensure scalability and flexibility, enabling these applications to handle increasing user demands and grow globally.

CHAPTER-9

PROJECT WORK MAPPING WITH PROGRAMME OUTCOMES

Pos	1	2	3	4	5	6	7	8	9	10	11	12
Project	2	3	3	2	3	3	2	2	2	2	3	3

PROGRAMMEOUTCOMES(POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

SESHADRI RAO
GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru

Department of Computer Science and Engineering

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions., component, or software to meet the desired needs.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

PSO1 : Design, develop, test and maintain reliable software systems and intelligent systems.

PSO2 : Design and develop web sites, web apps and mobile apps.

PROJECT PROFORMA

Classification of Project	Application	Product	Research	Review
	√			

Note: Tick Appropriate category

Project Outcomes	
CourseOutcome (CO1)	Acquiretechnical competence in the specific domain during the training.
CourseOutcome (CO2)	Identify the problem statement based on the requirementsof the industry
CourseOutcome (CO3)	Adaptproject management skills on par with industrial standards.
CourseOutcome (CO4)	Develop a system model to obtain a solution and generateareport.

Mapping Table

CS3523: INTERNSHIP/ INDUSTRIAL TRAINING/ PRACTICAL TRAINING														
Course outcomes	**Program Outcomes and Program Specific Outcome**													
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**		**PSO1**	**PSO2**
CO1	3	2	2	2	2			2	2	2	1	2		2	
CO2	3	3	2	2	1			2	2	2	1	2		2	2
CO3	1		1		1	1	1	2	2	2	3	2		2	
CO4	3	2	3	3	3	2	1	2	2	2	3	2		2	2
INTERNSHIP/ INDUSTRIAL TRAINING/ PRACTICAL TRAINING	3	2	2	2	2	1	1	2	2	2	2	2		2	1

Note: Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows:

1-Slightly (Low) mapped2-Moderately (Medium) mapped 3-Substantially (High) mapped

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P)

CERTIFICATE OF COMPLETION

This is to certify that Ms./Mr. **Ganji Snehalatha** of **Computer Science Engineering** with Registered Hall ticket no. **21481A0571** under **Seshadri Rao Gudlavalleru Engineering College** of **JNTUK** has successfully completed Short-Term Internship of 2 months on **Full Stack Development (MERN)** Organized by **SmartBridge Educational Services Pvt. Ltd.** in collaboration with **Andhra Pradesh State Council of Higher Education.**

Certificate ID: **EXT-APSCHE_FSD-23838**

Date: 24-Jul-2024

Place: Virtual

Amarendar Katkam

Founder & CEO

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P)

CERTIFICATE OF COMPLETION

This is to certify that Ms./Mr. **Keerthi Lahari Jakkula** of **Computer Science Engineering** with Registered Hall ticket no. **21481A0584** under **Seshadri Rao Gudlavalleru Engineering College** of **JNTUK** has successfully completed Short-Term Internship of 2 months on **Full Stack Development (MERN)** Organized by **SmartBridge Educational Services Pvt. Ltd.** in collaboration with **Andhra Pradesh State Council of Higher Education.**

Certificate ID: **EXT-APSCHE_FSD-26162**

Date: 25-Jul-2024

Place: Virtual

Amarendar Katkam

Founder & CEO

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P)

CERTIFICATE OF PARTICIPATION

This is to certify that Ms./Mr. **Kalidindi Devi Durga Bhavani** of **Computer Science Engineering** with Registered Hall ticket no. **21481A0593** under **Seshadri Rao Gudlavalleru Engineering College** of **JNTUK** has successfully completed Short-Term Internship of 2 months on **Full Stack Development (MERN)** Organized by **SmartBridge Educational Services Pvt. Ltd.** in collaboration with **Andhra Pradesh State Council of Higher Education.**

Certificate ID: **EXT-APSCHE_FSD-23857**

Date: 25-Jul-2024

Place: Virtual

Amarendar Katkam

Founder & CEO