# INTEGRATED PROJECT – I (CS204)



# Project Report

# on

# BaatCheet.com

**BATCH 2022**

**Project Mentor**

Dr. Rani Kumari

**Student Names & ID**

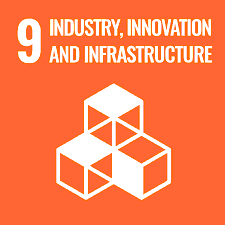
Shashank Rajput (2211981361)

Shreyansh (2211981378)

Shubham (2211981381)

Shreya (2211981377)

**SUSTAINABLE DEVELOPMENT GOALS**



Goal 9 seeks to build resilient infrastructure, promote sustainable industrialization and foster innovation.Economic growth, social development and climate action are heavily dependent on investments in infrastructure, sustainable industrial development and technological progress. In the face of a rapidly changing global economic landscape and increasing inequalities, sustained growth must include industrialization that first of all, makes opportunities accessible to all people, and second, is supported by innovation and

resilient infrastructure.



The tourism website can contribute to sustainable development by promoting decent work and economic growth. By partnering with local businesses, tour operators, and accommodations, the website can support job creation and economic opportunities within the toursim industry.

**Problem Statement:**

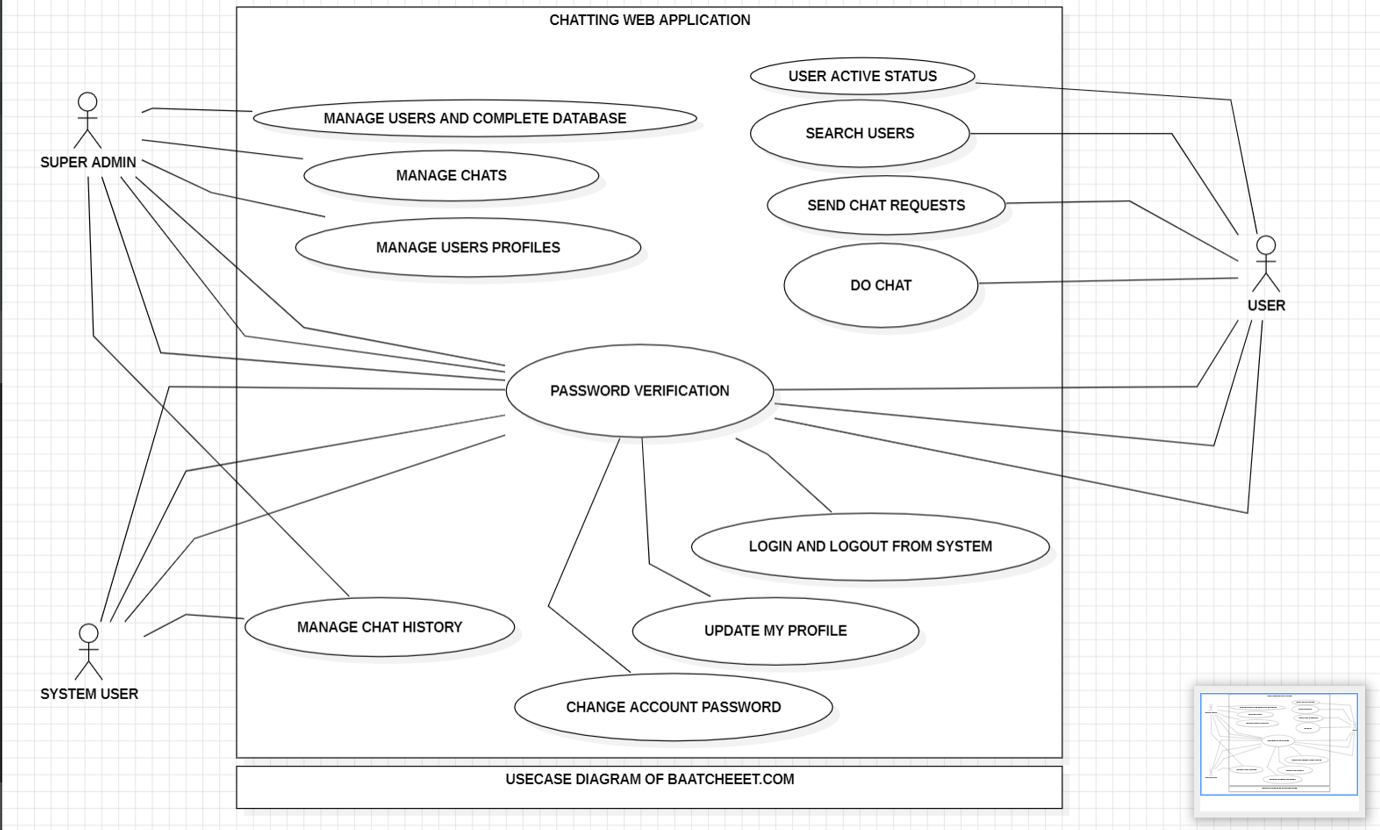
Despite advancements in technology, many communities still lack inclusive platforms for effective communication and collaboration, hindering their ability to achieve sustainable development goals. Addressing this gap, our project aims to develop a chat web application using the MERN stack, empowering connectivity and fostering community resilience. By leveraging technology to promote dialogue and knowledge-sharing, we seek to contribute to Sustainable Development Goals (SDGs), particularly Goal 9 (Industry, Innovation, and Infrastructure) .

**Solution Statement:**

Our solution is a user-friendly chat app built with the MERN stack, enabling seamless communication and collaboration. With features like real-time messaging and file sharing, it empowers users to connect effectively. We continuously improve through user feedback, ensuring adaptability. By enhancing connectivity and inclusivity, our solution contributes to Sustainable Development Goals (SDGs) like Goal 9 (Industry, Innovation, and Infrastructure).

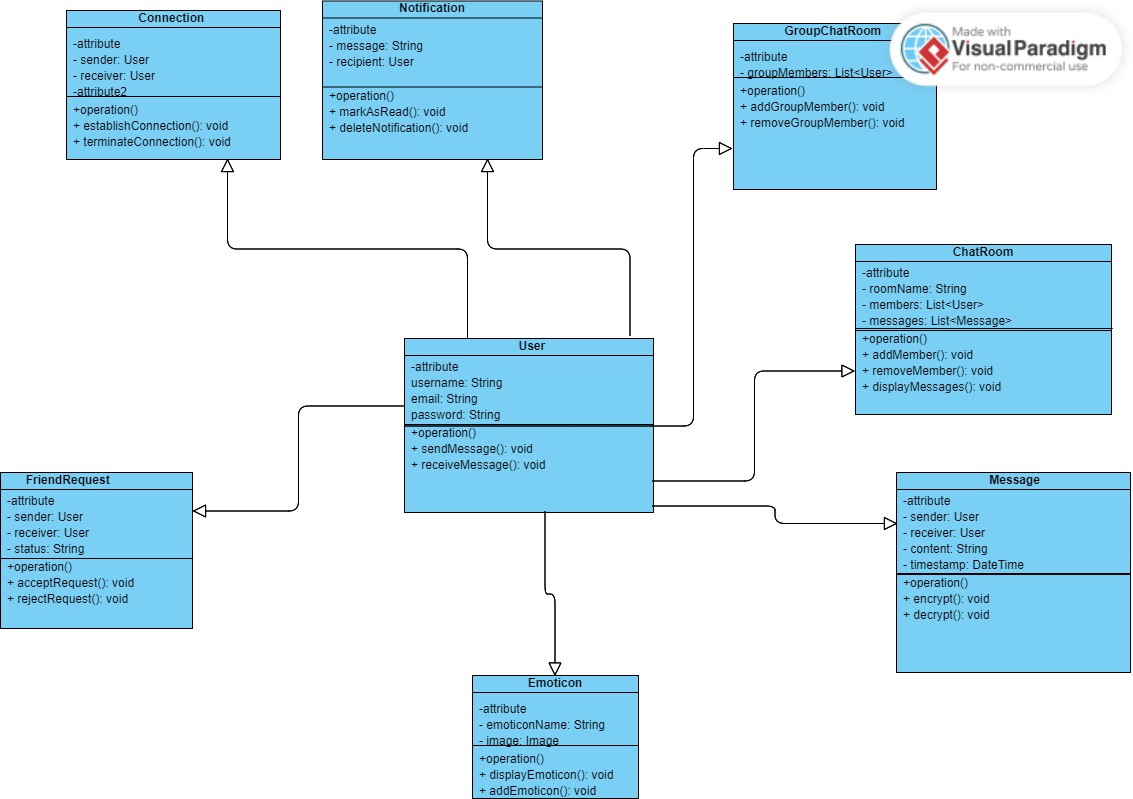
**UML Diagrams:**

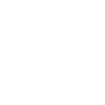
**Use case Diagram:**





**Class Diagram:**





**Introduction**

**Project Overview**

In the realm of modern web development, real-time communication applications have become integral in various domains such as social networking, customer support, and collaborative platforms. This project report outlines the development of a real-time chat application utilizing the MERN (MongoDB, Express.js, React.js, Node.js) stack and Socket.IO, undertaken as a part of the undergraduate curriculum in Computer Science and Engineering.

**Motivation**

The primary motivation behind this project is to explore and harness the capabilities of the MERN stack, a popular full-stack JavaScript solution, combined with Socket.IO for real-time, bidirectional communication. By developing a chat application, we aim to delve into practical aspects of real-time web technologies, enhancing our understanding and proficiency in full-stack development.

**Objectives**

The objectives of this project are multifaceted:

1. **Design and Implement a Real-Time Chat Application**: To create a functional, user-friendly chat platform that allows users to communicate instantaneously.
2. **Utilize MERN Stack Effectively**: To leverage MongoDB for data storage, Express.js and Node.js for server-side logic, and React.js for a dynamic front-end experience.
3. **Incorporate Socket.IO**: To enable real-time communication between the client and server, ensuring messages are delivered promptly and efficiently.
4. **Enhance User Experience**: To develop a responsive and intuitive user interface that caters to seamless interaction.

**Technologies Used**

* **MongoDB**: A NoSQL database used for storing user information and chat history, chosen for its scalability and flexibility in handling unstructured data.
* **Express.js**: A minimal and flexible Node.js web application framework providing a robust set of features for web and mobile applications, used for building the server-side application.
* **React.js**: A JavaScript library for building user interfaces, employed to create a responsive and interactive front-end.
* **Node.js**: A JavaScript runtime built on Chrome's V8 JavaScript engine, used for building the server-side of the application.
* **Socket.IO**: A library that enables real-time, bidirectional and event-based communication, used to facilitate live chat functionality.

**Key Features**

* **User Authentication**: Secure user login and registration system to manage user access.
* **Real-Time Messaging**: Instant message transmission and receipt, providing a seamless communication experience.
* **Chat Rooms**: Ability to create and join chat rooms, supporting multiple simultaneous conversations.
* **User Interface**: A clean, intuitive, and responsive UI designed to enhance user engagement.
* **Scalability and Performance**: Efficient handling of concurrent connections and data exchanges, ensuring a robust and scalable application.

**Scope of the Project**

This project encompasses the complete lifecycle of web application development, including requirement analysis, system design, implementation, testing, and deployment. The focus is on building a reliable, real-time communication system with an emphasis on practical implementation of theoretical concepts learned during the coursework.

**Detailed Planning**

**1. Requirement Analysis**

**Functional Requirements**

1. **User Registration and Authentication**
   * Users should be able to register with a username and password.
   * Users should be able to log in using their credentials.
   * Implement session management to keep users logged in.
2. **User Profile Management**
   * Users can view and edit their profile information.
   * Users can upload a profile picture.
3. **Real-Time Messaging**
   * Users can send and receive messages instantly.
   * Support for one-on-one and group chats.
   * Notifications for new messages.
4. **Chat Rooms**
   * Users can create and join chat rooms.
   * Chat rooms should support multiple participants.
   * Display list of active chat rooms.
5. **Message Storage and Retrieval**
   * Store chat history in MongoDB.
   * Allow users to retrieve past messages.

**Non-Functional Requirements**

1. **Performance**
   * The application should handle multiple simultaneous users without significant delay.
2. **Scalability**
   * The system should be able to scale horizontally to support an increasing number of users and messages.
3. **Security**
   * Protect user data with encryption.
   * Implement secure authentication mechanisms.
4. **Usability**
   * The user interface should be intuitive and easy to navigate.
   * Provide a responsive design that works on various devices.

**2. System Design**

**Architecture**

* **Frontend: React.js**
  + Handles user interactions and renders the user interface.
  + Communicates with the backend via RESTful APIs and Socket.IO.
* **Backend: Node.js with Express.js**
  + Manages user authentication, session management, and business logic.
  + Handles real-time communication using Socket.IO.
  + Provides RESTful APIs for CRUD operations.
* **Database: MongoDB**
  + Stores user information, chat history, and other necessary data.
  + Utilizes Mongoose for object data modeling (ODM).

**Data Flow**

1. User interacts with the frontend (React.js).
2. Frontend sends requests to the backend (Node.js/Express.js).
3. Backend processes requests and interacts with the database (MongoDB).
4. Real-time events are managed by Socket.IO for instant communication.

**3. UI/UX Design**

**Design Principles**

* Simplicity: Keep the interface clean and uncluttered.
* Consistency: Use consistent colors, fonts, and layout throughout the application.
* Feedback: Provide immediate feedback for user actions (e.g., message sent confirmation).
* Accessibility: Ensure the application is accessible to users with disabilities.

**Wireframes**

**Login and Registration Pages**

* **Login Page:** Form with fields for username and password, and a submit button.
* **Registration Page:** Form with fields for username, password, email, and a submit button.

**Main Chat Interface**

* **Header:** Application title and user profile access.
* **Sidebar**: List of chat rooms and friends.
* **Chat Window**: Display messages with timestamps and sender information.
* **Message Input**: Text area for typing messages and a send button.

**User Profile Page**

* **Profile Information**: Display and edit fields for username, email, and profile picture.
* **Save Button**: Save changes to the profile.

**Prototyping Tools**

* Use tools like Figma or Adobe XD for creating wireframes and interactive prototypes.

**4. Implementation Plan**

**Milestones**

1. **Setup and Initial Configuration: (Week 1-2)**
   * Set up the project repository.
   * Configure the development environment.
   * Initialize React.js and Node.js projects.
2. **User Authentication Module: (Week 3-4)**
   * Implement registration and login functionality.
   * Set up session management and security features.
3. **Real-Time Messaging Module: (Week 5-6)**
   * Integrate Socket.IO for real-time communication.
   * Develop the chat interface and message handling logic.
4. **Chat Room Functionality: (Week 7-8)**
   * Implement creation and management of chat rooms.
   * Enable joining and leaving chat rooms.
5. **User Profile Management: (Week 9-10)**
   * Develop profile view and edit features.
   * Integrate profile picture upload.
6. **Testing and Debugging: (Week 11-12)**
   * Conduct unit and integration testing.
   * Fix identified bugs and optimize performance.
7. **Final Deployment: (Week 13-14)**
   * Prepare for deployment on a cloud platform (e.g., Heroku, AWS).
   * Conduct final testing and user acceptance testing.

**5. Testing Strategy**

**Unit Testing**

* Test individual components and functions for correctness.
* Use testing libraries like Jest for JavaScript.

**Integration Testing**

* Test the interaction between different modules.
* Ensure seamless data flow and functionality across the system.

**User Acceptance Testing (UAT)**

* Conduct testing sessions with potential users.
* Gather feedback and make necessary adjustments.

**6. Deployment Plan**

**Preparation**

* Ensure all features are thoroughly tested and stable.
* Prepare documentation and user manuals.

**Deployment Steps**

* Deploy the backend on a cloud platform.
* Deploy the frontend on a CDN or web server.
* Configure the database on a managed database service.

**Monitoring and Maintenance**

* Set up monitoring tools to track application performance.
* Plan for regular maintenance and updates based on user feedback.

**Objectives and Key Learning**

**Objectives**

1. **Develop a Full-Stack Application:**
   * Build a complete web application using the MERN stack (MongoDB, Express.js, React.js, Node.js), showcasing the integration of front-end and back-end technologies.
2. **Implement Real-Time Communication:**
   * Use Socket.IO to enable real-time, bidirectional communication between the client and server, allowing for instant message exchange.
3. **Enhance User Experience (UX):**
   * Design a responsive and intuitive user interface that provides a seamless and engaging experience for users across different devices.
4. **Ensure Data Security:**
   * Implement secure authentication mechanisms and data encryption to protect user information and communication.
5. **Handle Scalability and Performance:**
   * Optimize the application to efficiently manage multiple simultaneous users and high volumes of data without performance degradation.
6. **Incorporate Modern Development Practices:**
   * Utilize contemporary software development methodologies and tools, such as version control, continuous integration, and automated testing.
7. **Facilitate Team Collaboration:**
   * Develop skills in project management, teamwork, and collaboration using tools like GitHub for version control and task tracking.

**Key Learning**

1. **Mastery of MERN Stack:**
   * Gain in-depth knowledge and practical experience with MongoDB, Express.js, React.js, and Node.js, understanding how to leverage each technology effectively in a full-stack application.
2. **Real-Time Web Development:**
   * Learn how to implement and manage real-time communication using Socket.IO, understanding its advantages and challenges in web applications.
3. **User Interface and Experience Design:**
   * Develop skills in UI/UX design principles, creating interfaces that are not only functional but also visually appealing and user-friendly.
4. **Database** **Management**:
   * Understand how to design, implement, and query a NoSQL database (MongoDB), including data modeling and schema design for optimal performance and scalability.
5. **Authentication and Security:**
   * Learn best practices for user authentication, session management, and data security, ensuring that user data and communication are protected against common vulnerabilities.
6. **Project Planning and Execution:**
   * Acquire project management skills, including requirement analysis, system design, milestone setting, and progress tracking, ensuring the successful and timely completion of the project.
7. **Problem Solving and Debugging:**
   * Enhance problem-solving skills by debugging issues that arise during development, from initial code errors to complex integration challenges.
8. **Testing and Quality Assurance:**
   * Gain experience in writing and executing unit tests, integration tests, and user acceptance tests, ensuring the application is robust and reliable.
9. **Deployment and Maintenance:**
   * Learn how to deploy a web application to a cloud platform, monitor its performance, and maintain it post-deployment, including handling updates and scaling.
10. **Collaboration and Communication:**
    * Develop teamwork and communication skills by collaborating with peers, documenting progress, and presenting the project to stakeholders.

**Methodology**

**1. Project Planning**

**Requirement Gathering**

* **Objective**: Identify and document the functional and non-functional requirements of the chat application.
* **Activities**:
  + Conduct brainstorming sessions and discussions.
  + Analyze existing chat applications for feature ideas and best practices.
  + Document detailed requirements for user authentication, real-time messaging, chat rooms, and user profile management.

**Project Timeline**

* **Objective**: Establish a realistic timeline with milestones and deadlines.
* **Activities**:
  + Break down the project into smaller tasks.
  + Allocate time frames for each task.
  + Use project management tools like Gantt charts or Trello boards for tracking progress.

**2. System Design**

**Architecture Design**

* **Objective**: Create a scalable and efficient architecture for the application.
* **Activities**:
  + Design the overall architecture using the MERN stack.
  + Define the interactions between the frontend, backend, and database.
  + Create data flow diagrams and system architecture diagrams.

**Database Design**

* **Objective**: Develop a robust database schema to store user information, chat history, and other data.
* **Activities**:
  + Define collections and documents for MongoDB.
  + Establish relationships and indexes for efficient data retrieval.
  + Create Entity-Relationship (ER) diagrams.

**UI/UX Design**

* **Objective**: Design a user-friendly and visually appealing interface.
* **Activities**:
  + Create wireframes and mockups using tools like Figma or Adobe XD.
  + Conduct usability testing and gather feedback.
  + Iterate on the design based on user feedback.

**3. Implementation**

**Frontend Development**

* **Objective**: Build the client-side application using React.js.
* **Activities**:
  + Set up the React.js project structure.
  + Implement components for login, registration, chat interface, and user profiles.
  + Use React Router for navigation between different views.
  + Integrate Socket.IO for real-time messaging.

**Backend Development**

* **Objective**: Develop the server-side application using Node.js and Express.js.
* **Activities**:
  + Set up the Node.js project with Express.js.
  + Implement RESTful APIs for user authentication, chat room management, and message storage.
  + Integrate Socket.IO for handling real-time communication.
  + Use Mongoose for MongoDB interactions.

**Integration**

* **Objective**: Ensure seamless communication between the frontend and backend.
* **Activities**:
  + Connect the frontend to the backend APIs.
  + Implement real-time data synchronization using Socket.IO.
  + Test the end-to-end functionality.

**4. Testing**

**Unit Testing**

* **Objective**: Verify the correctness of individual components and functions.
* **Activities**:
  + Write unit tests for React components using testing libraries like Jest and Enzyme..

**Integration Testing**

* **Objective**: Test the interaction between different modules.
* **Activities**:
  + Conduct integration tests to ensure seamless data flow.
  + Use tools like Postman to test API endpoints.

1. **Deployment**

**Preparation**

* **Objective**: Prepare the application for deployment.
* **Activities**:
  + Ensure all features are stable and thoroughly tested.
  + Prepare documentation and user manuals.

**Deployment**

* **Objective**: Deploy the application to a live environment.
* **Activities**:
  + Deploy the backend on a cloud platform onrender.com .
  + Deploy the frontend on a CDN or web server.
  + Set up a managed database service for MongoDB.

**Monitoring and Maintenance**

* **Objective**: Ensure the application remains functional and up-to-date.
* **Activities**:
  + Set up monitoring tools to track application performance.
  + Plan for regular maintenance and updates based on user feedback.

**6. Documentation**

**Project Documentation**

* **Objective**: Document all aspects of the project for future reference and user guidance.
* **Activities**:
  + Create technical documentation for the system design and implementation.
  + Develop user guides and manuals.
  + Maintain a project log with details of progress and issues encountered.

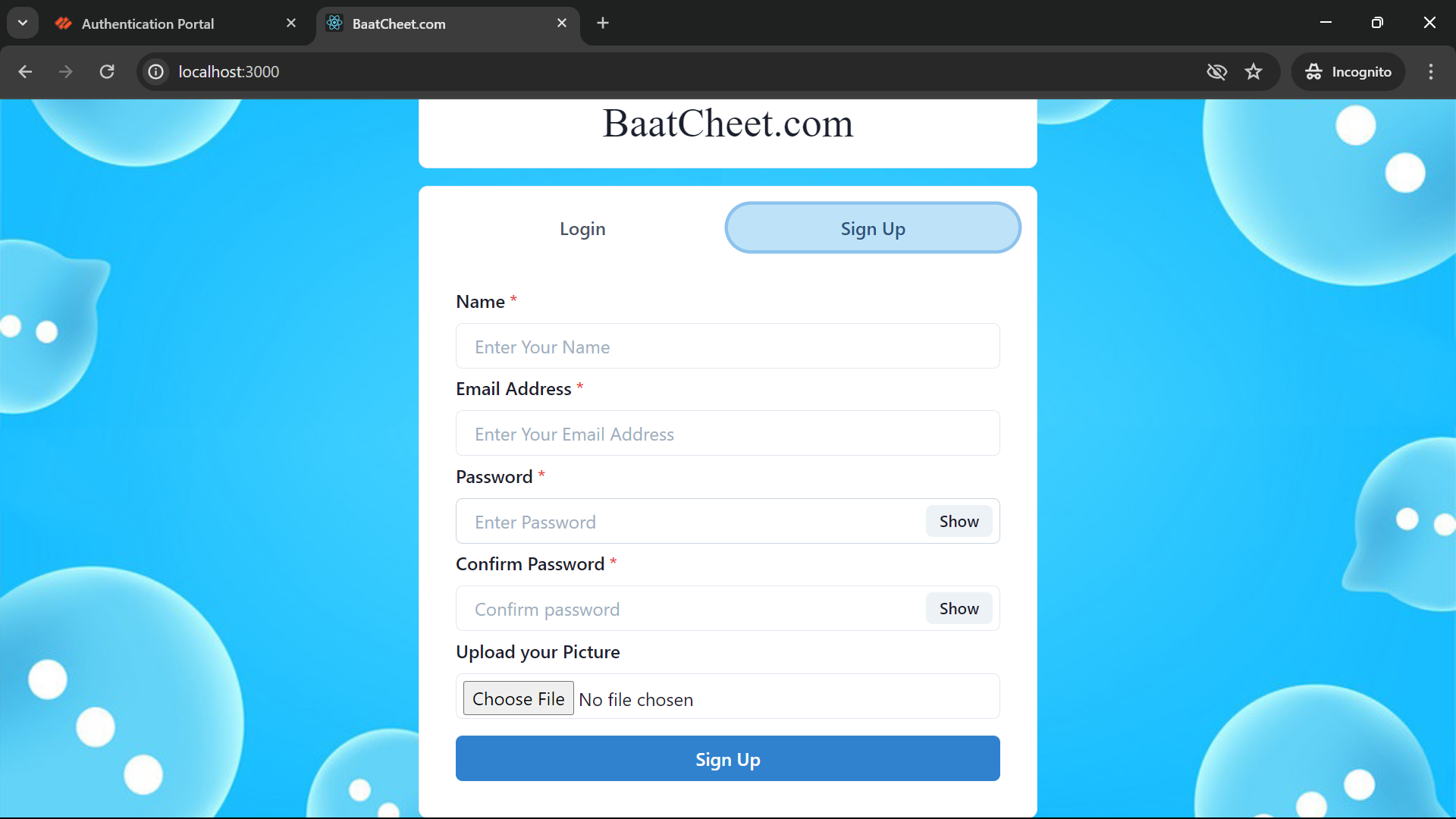
**7. Evaluation**

**Performance Evaluation**

* **Objective**: Assess the performance of the application.
* **Activities**:
  + Conduct load testing to evaluate how the application handles multiple users.
  + Optimize code and database queries based on performance results.

**User Feedback**

* **Objective**: Collect feedback from users to improve the application.
* **Activities**:
  + Distribute the application to a group of users.
  + Gather and analyze feedback.
  + Implement improvements based on user suggestions.



**SIGNUP PAGE**

**Signup Page Description**

The signup page of our chat application is designed to be easy to use and welcoming for new users. Here's what you'll find on this page:

**Input Fields**

1. **Name**:
   * A text field where you can enter your full name.
2. **Email**:
   * A text field where you can enter your email address.
3. **Password**:
   * A secure field where you can create a password for your account.
4. **Confirm Password**:
   * Another secure field where you need to re-enter the password to confirm it matches.
5. **Profile Photo**:
   * An upload field where you can choose and upload a profile photo from your device.

**Signup Button**

* After filling in all the required fields, click the "Sign Up" button to create your account.

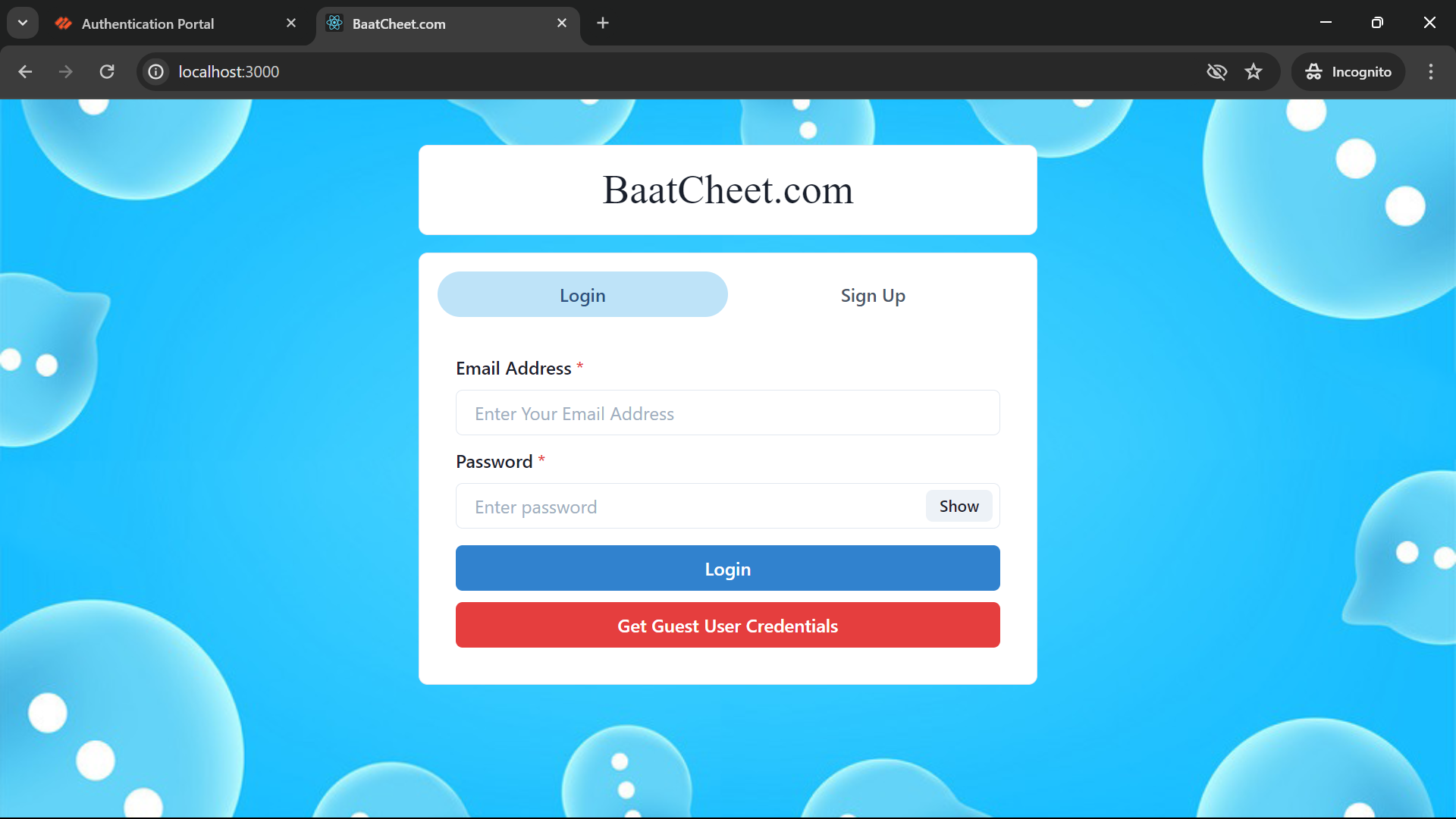
**Link to Login Page**

* Below the signup form, there’s a link that says "Already have an account? Log in". Clicking this link takes you to the login page where you can access your account if you already have one.

**Layout**

* The layout is simple and clear, with each field and button labeled for easy understanding.
* The form is centered on the page, ensuring it's the main focus for new users.

Our signup page aims to make the registration process quick and straightforward, so you can start using the chat application without any hassle.



**LOGIN PAGE**

**Login Page Description**

The login page of our chat application is designed to be straightforward and user-friendly. Here's what you'll find on this page:

**Input Fields**

1. **Email**:
   * A text field where you can enter your email address.
2. **Password**:
   * A secure field where you can enter your password.
   * There's a "Show Password" option next to this field. When you click it, you can see the password you’ve typed, making it easier to check for any mistakes.

**Buttons**

1. **Submit Button**:
   * Once you’ve entered your email and password, click the "Submit" button to log in to your account.
2. **Guest User Option**:
   * There is also a button that allows you to log in as a guest user. This option lets you explore the chat application without creating an account.

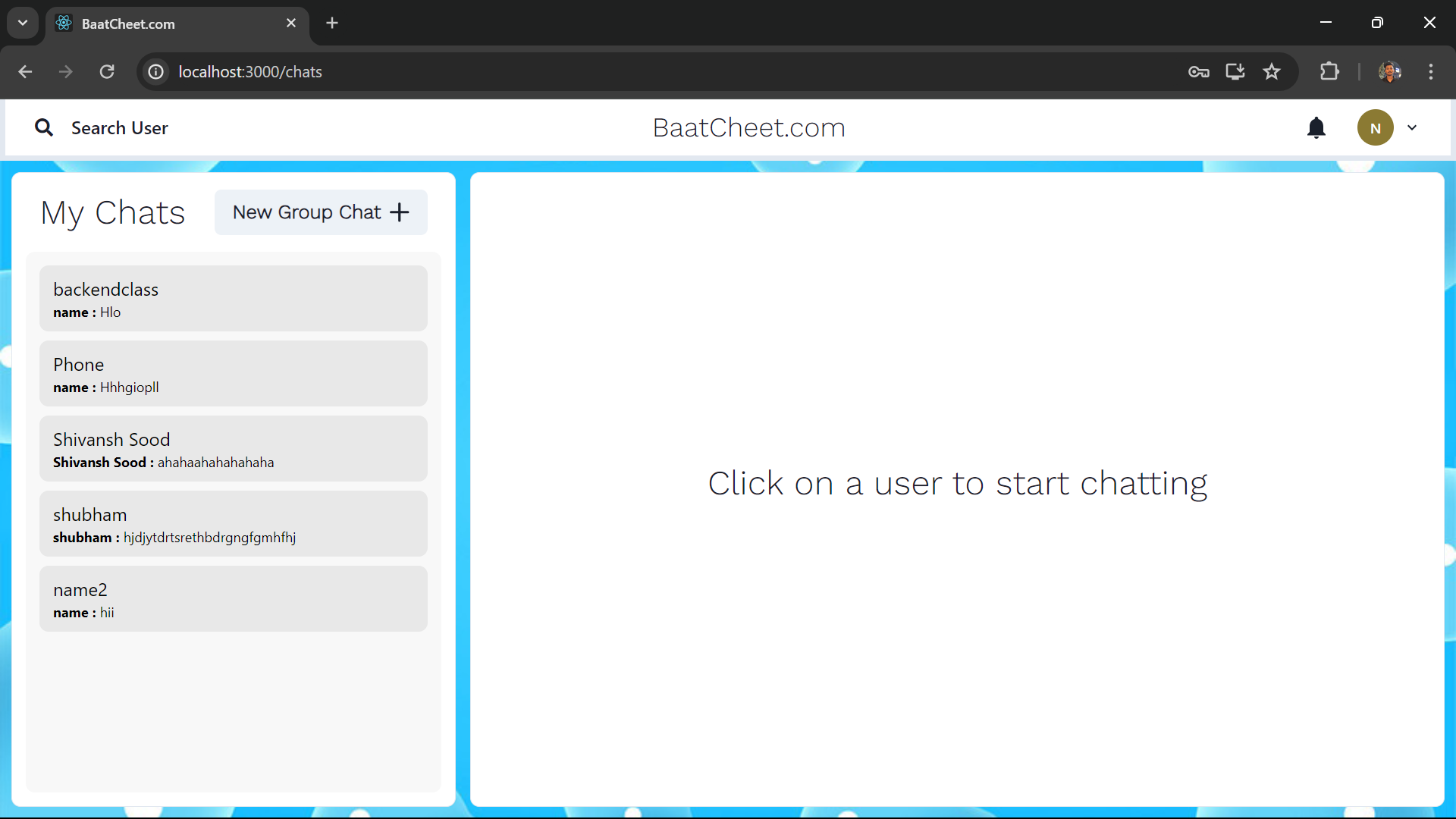
**Link to Signup Page**

* Below the login form, there’s a link that says "Don’t have an account? Sign up". Clicking this link takes you to the signup page where you can create a new account.

**Layout**

* The layout is clean and simple, with all elements clearly labeled and easy to find.
* The form is centered on the page, making it the focal point when you arrive.

Our goal is to make logging in quick and easy, so you can start chatting right away.



**HOME PAGE**

**Home Page Description**

The home page of our chat application is designed to provide users with a seamless and intuitive experience, combining functionality with a clean, modern interface. Here is an overview of the main sections and features available on the home page:

**Layout and Features**

**Left Sidebar**

1. **Search Users**:
   * A search bar at the top of the sidebar allows users to quickly find and connect with other users. As you type, the search bar provides real-time suggestions, making it easy to locate friends or new contacts.
2. **Notifications**:
   * Below the search bar, there is a notifications icon. Clicking on this icon opens a dropdown menu displaying recent notifications, such as new messages, friend requests, or mentions in group chats. Each notification includes a timestamp and a brief description.
3. **Profile Access**:
   * Users can see their own profile picture and username at the top of the sidebar. Clicking on the profile picture opens the user profile page, where users can view and edit their personal information, change their profile picture, and update their settings.
4. **All Chats and Groups**:
   * The main section of the left sidebar lists all ongoing chats and groups. Each entry includes the contact’s name or group name, a small profile picture or group icon, and a snippet of the most recent message. Chats and groups are sorted by the time of the last message, with the most recent conversations appearing at the top.
   * Users can click on any chat or group to open the conversation in the main chat window.

**Main Chat Window**

1. **Current Chat Display**:
   * The right side of the home page is dedicated to the currently selected chat or group conversation. This section displays all messages exchanged in the conversation, complete with timestamps and sender information.
   * Messages are displayed in a clear, threaded format, with different colors or styles distinguishing between sent and received messages.
2. **Message Input Area**:
   * At the bottom of the main chat window, users can find a text input area to type their messages. This area includes options to attach files, send emojis, and voice notes. The input area is designed for ease of use, allowing users to quickly send messages and attachments.
3. **Chat Header**:
   * The top of the main chat window features a header displaying the name and profile picture of the contact or group, along with online status or last seen information. The header also includes icons for accessing additional options such as viewing group members, searching within the conversation, and leaving or deleting the chat.

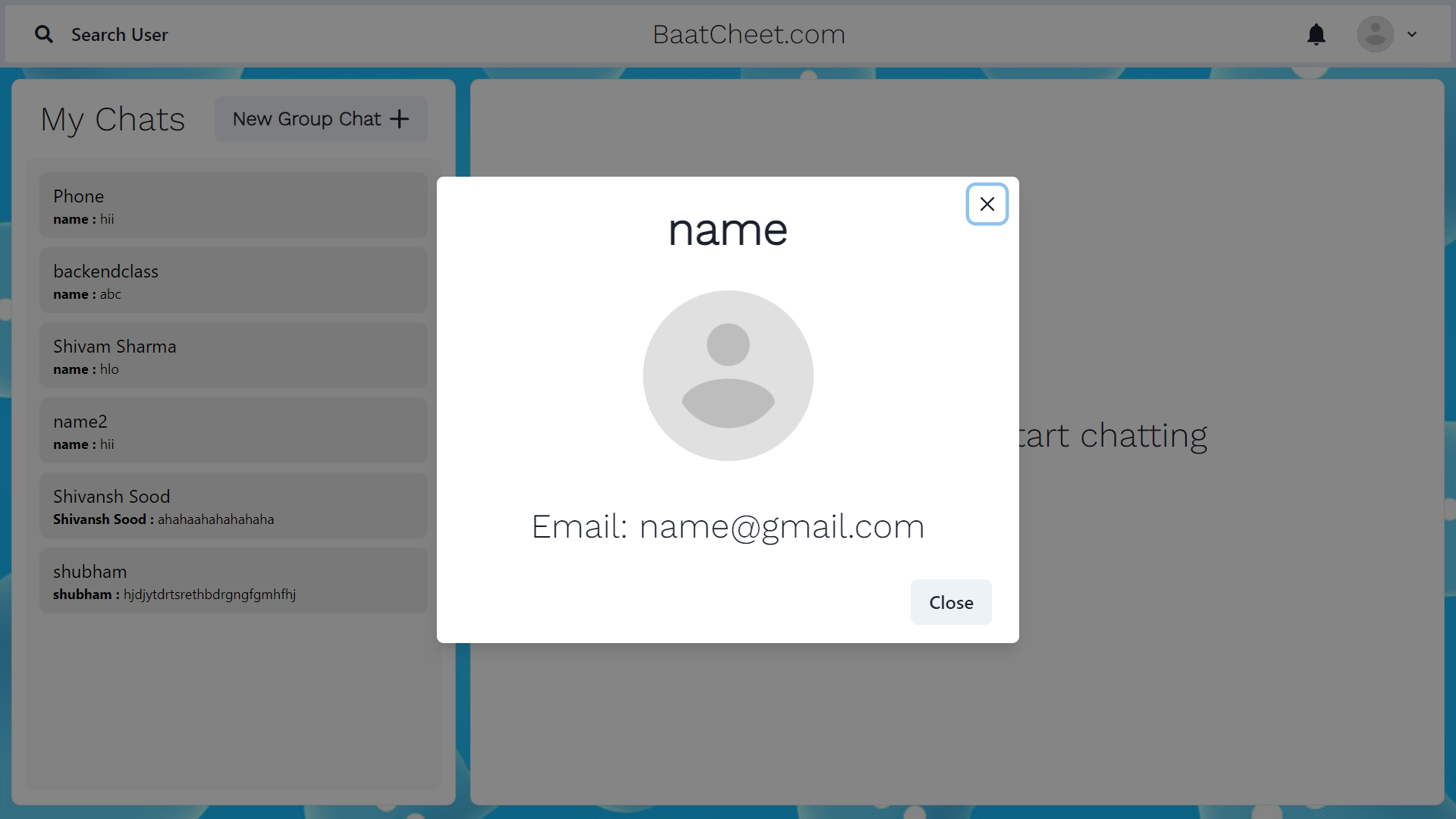
**User Experience**

The home page is designed to be highly responsive, ensuring that users have a smooth experience whether they are accessing the application from a desktop, tablet, or mobile device. The layout adjusts dynamically to fit different screen sizes, maintaining usability and readability.

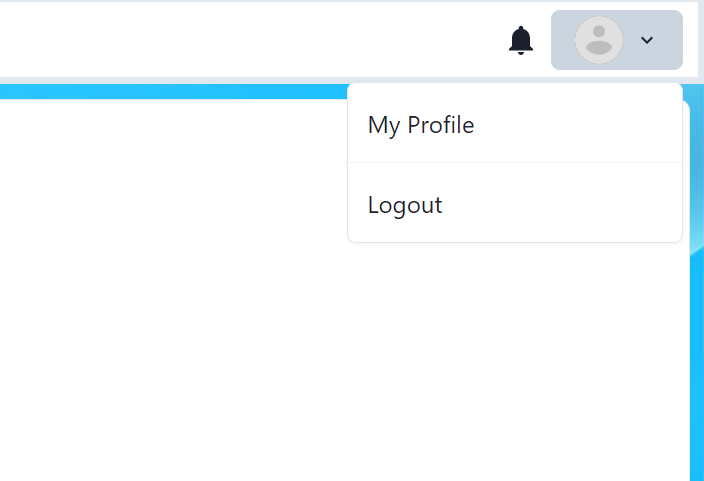
**Key Features**

* **Real-Time Updates**: Messages, notifications, and chat statuses are updated in real-time, ensuring that users always see the most current information.
* **Easy Navigation**: The intuitive layout and clearly labeled sections make it easy for users to navigate the application and access different features.
* **Personalization**: Users can customize their profile, manage their notification preferences, and organize their chat list to suit their preferences.

The home page of our chat application aims to provide a central hub for all communication activities, ensuring that users can easily stay connected, manage their conversations, and engage with their contacts in a streamlined and efficient manner.



**PROFILE PAGE**



**LOGOUT OPTION**

**Profile and Logout Options**

The home page of our chat application includes convenient options for accessing your profile and logging out. Here’s a detailed description of these features:

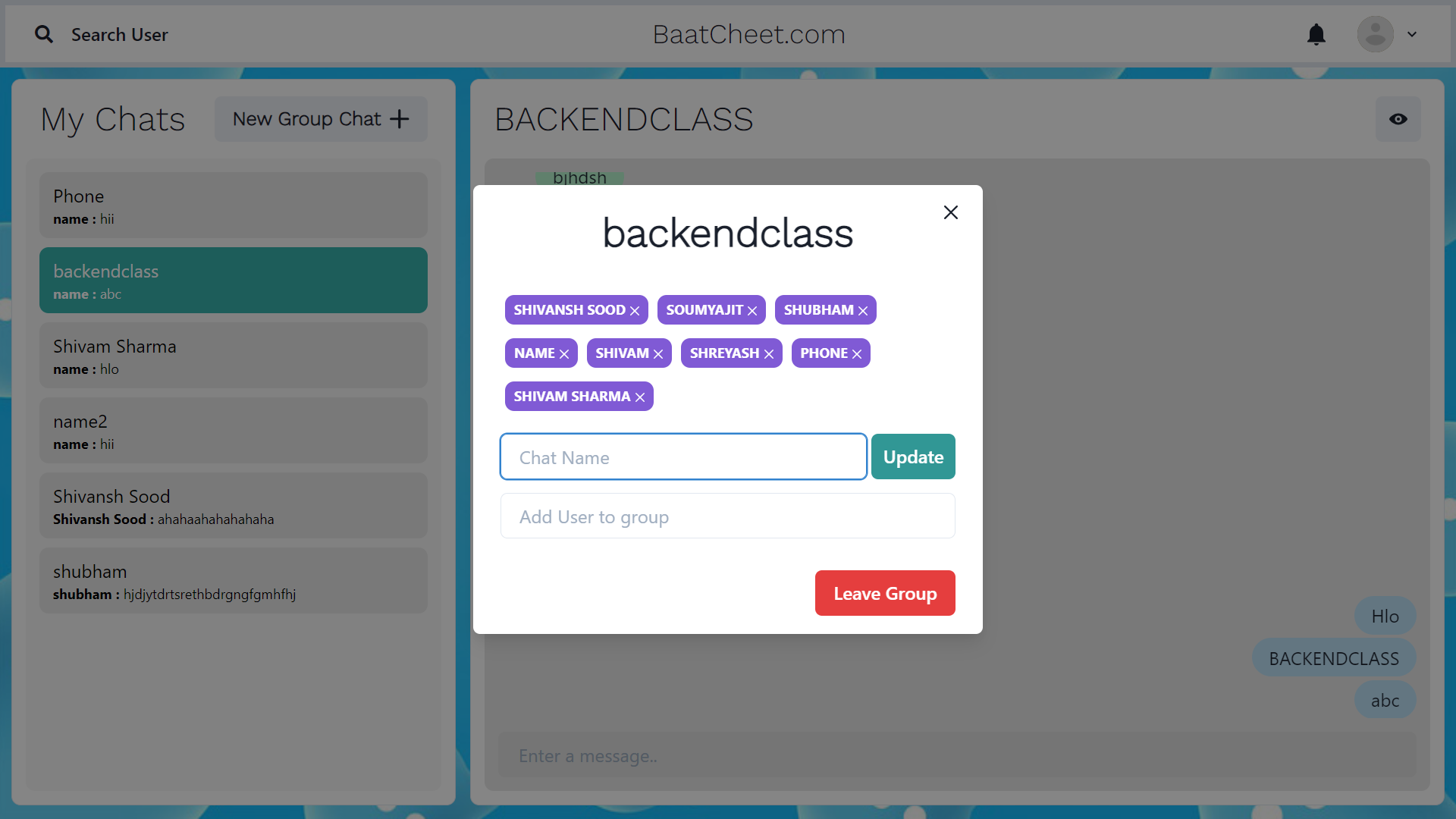
**Profile and Logout Options**

1. **Profile Button**:
   * Located in the top-right corner of the home page, there is a profile button represented by your profile picture or an avatar icon.
   * Clicking this button opens a dropdown menu with the following options:
     + **View Profile**: Takes you to your profile page where you can view and edit your personal information, including your name, email, and profile photo.
2. **Logout Option**:
   * Also found in the dropdown menu from the profile button, there is a "Logout" option.
   * Clicking "Logout" will sign you out of the application and redirect you to the login page, ensuring your session is securely ended.

**Layout and Accessibility**

* The profile button is easily accessible at the top of the page, making it quick and simple to find and use.
* The dropdown menu is designed to be intuitive, with clear options for viewing as well as logging out.
* These features ensure that managing your account and securing your session is straightforward and convenient.

Our goal is to provide a seamless experience for accessing your profile and logging out, enhancing the overall usability of the chat application.



**CREATE AND EDIT GROUP**

**Group Management Page Description**

The group management page of our chat application allows users to easily create new groups or modify existing ones. Here’s a detailed description of the features available on this page:

**Creating a New Group**

1. **Group Name Input:**
   * A text field where you can enter the name of the new group.
2. **Add Members:**
   * A search bar that allows you to find and select users to add to the group. You can type the names of users and select them from the suggestions list.
3. **Create Group Button:**
   * After entering the group name and adding members, click the "Create Group" button to finalize the creation of the new group. This will take you to the group chat page where you can start messaging.

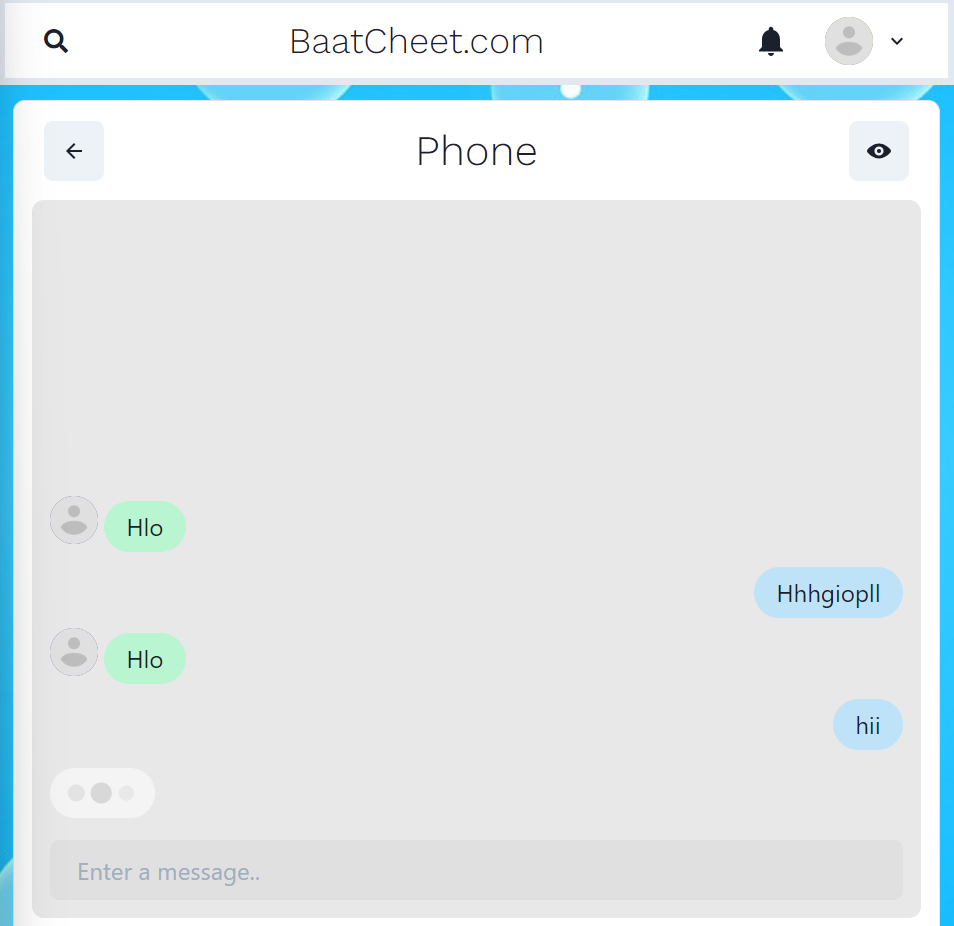
**Editing an Existing Group**

1. **Group Name:**
   * A text field displaying the current group name. You can edit this field to change the group name.
2. **Current Members List:**
   * A list showing all current members of the group. Each member’s name is displayed along with an option to remove them from the group.
3. **Add New Members:**
   * Similar to creating a new group, there’s a search bar to find and add new members to the group. Type the names of users and select them from the suggestions list.
4. **Save Changes Button:**
   * After making any changes to the group name or membership, click the "Save Changes" button to update the group information.

**Layout and Accessibility**

* The layout is designed to be intuitive, with clear labels and easy-to-find buttons for each action.
* The group name input and member management features are centrally located, ensuring that users can quickly make the necessary changes.
* The page provides real-time feedback on changes, such as updating the member list immediately after a user is added or removed.

Our group management page aims to make it simple and efficient to create new groups or modify existing ones, enhancing collaboration and communication within the chat application.

**CHAT PAGE**

**Chat Page Description**

The chat page of our chat application is designed to facilitate smooth and real-time communication between users. Here’s a detailed description of the features available on this page:

**Main Chat Area**

1. **Current Chat Display:**
   * The main section of the chat page shows the conversation with the selected contact or group. Messages are displayed in a threaded format with timestamps and sender information, making it easy to follow the conversation.
2. **Previous Chats:**
   * All previous messages in the conversation are displayed, allowing users to scroll up and view the chat history. This ensures that users can reference past discussions at any time.

**Typing Indicator**

1. **Typing Status Icon:**
   * An icon or message appears at the bottom of the chat window when the other user is typing. This real-time feedback lets you know that the other person is in the process of sending a message.

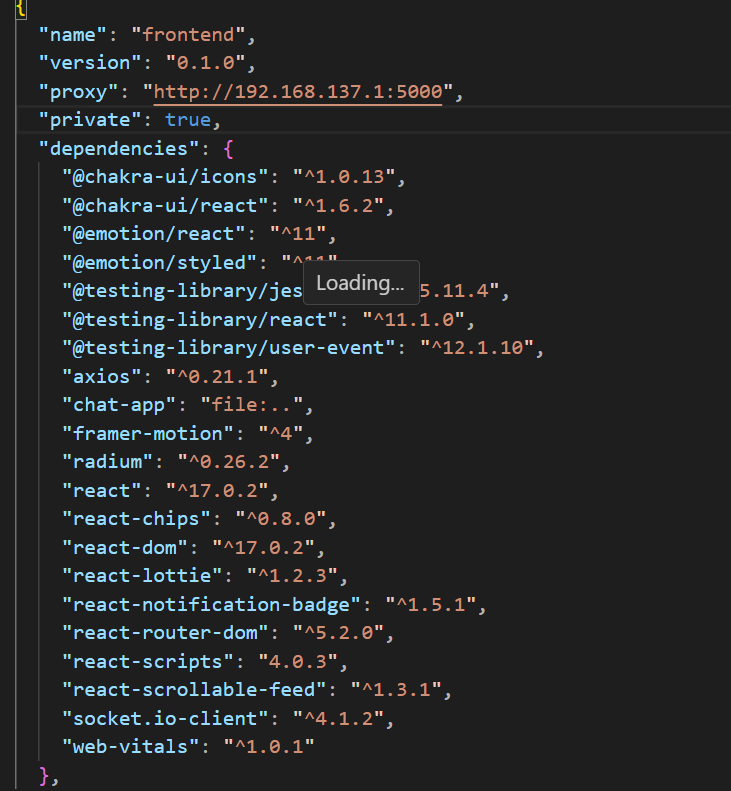
**Message Input Area**

1. **Text Input Field:**
   * At the bottom of the chat page, there is a text box where you can type your messages. This area is equipped with features to:
     + Attach files.
     + Send emojis.
     + Record and send voice notes.
   * The input field is designed to be user-friendly, allowing you to compose and send messages quickly.
2. **Send Button:**
   * Next to the text input field is a "Send" button that you can click to send your message. Alternatively, you can press "Enter" on your keyboard to send the message.

**Layout and Accessibility**

* The layout is clean and intuitive, with the chat window taking up most of the screen to focus on the conversation.
* Navigation to other parts of the application, such as the chat list or user profile, is easily accessible from the sidebar or top menu.
* The chat area automatically scrolls to the newest message, ensuring you are always up-to-date with the latest part of the conversation.

Our chat page is designed to provide a seamless and engaging messaging experience, allowing users to communicate in real time while having access to their chat history**.**

**CODE DETAILS –**

**CODE DEPENDENCIES OF FONTEND**

**Project Dependencies of Frontend**

**General Information**

This project is built using various libraries and frameworks to enhance its functionality and development process. The following are the key dependencies used in the frontend development of the project.

**List of Dependencies**

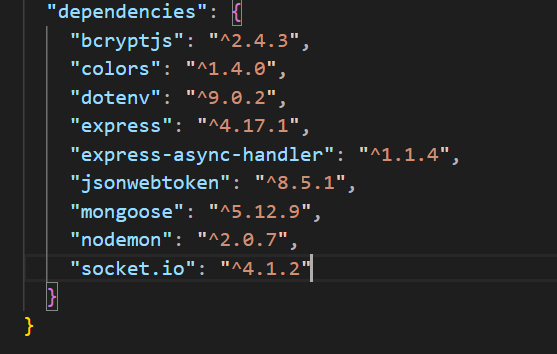
1. @chakra-ui/icons: ^1.0.13
   * Provides a set of commonly used icons for use in Chakra UI-based applications.
2. @chakra-ui/react: ^1.6.2
   * A simple, modular, and accessible component library for React applications.
3. @emotion/react: ^11
   * A library designed for writing CSS styles with JavaScript, supporting both styled components and css-in-js.
4. @emotion/styled: ^11
   * Part of the Emotion library, allows writing styled components using a syntax similar to styled-components.
5. @testing-library/jest-dom: ^5.11.4
   * Custom jest matchers for asserting on DOM nodes, making it easier to test the state of the DOM.
6. @testing-library/react: ^11.1.0
   * Simple and complete React DOM testing utilities that encourage good testing practices.
7. @testing-library/user-event: ^12.1.10
   * Provides a way to simulate user interactions with elements, improving the way user events are tested in React.
8. axios: ^0.21.1
   * A promise-based HTTP client for the browser and Node.js, used to make HTTP requests.
9. chat-app: file:...
   * A local file dependency, likely a custom module for the chat application part of the project.
10. framer-motion: ^4
    * A library for creating high-performance animations and gestures for React.
11. radium: ^0.26.2
    * A set of tools to manage inline styles on React elements, allowing for CSS styling in JavaScript.
12. react: ^17.0.2
    * A JavaScript library for building user interfaces, maintained by Facebook.
13. react-chips: ^0.8.0
    * A React component for creating and managing chips (tag-like elements).
14. react-dom: ^17.0.2
    * Provides DOM-specific methods that can be used at the top level of a web app to enable an efficient React tree.
15. react-lottie: ^1.2.3
    * A wrapper for lottie-web, which makes it easy to render animations in React applications.
16. react-notification-badge: ^1.5.1
    * A simple and customizable badge component to show notification counts in a React application.
17. react-router-dom: ^5.2.0
    * A collection of navigational components that compose declaratively with your application, enabling client-side routing.
18. react-scripts: 4.0.3
    * Scripts and configuration used by Create React App, providing a comfortable development environment.
19. react-scrollable-feed: ^1.3.1
    * A component to create an infinite scrollable feed for chat applications or comment sections.
20. socket.io-client: ^4.1.2
    * A client-side library for Socket.IO, which enables real-time, bidirectional and event-based communication.
21. web-vitals: ^1.0.1
    * A library for measuring essential web vitals, such as largest contentful paint, first input delay, and cumulative layout shift.

**Project Information**

* Name: frontend
* Version: 0.1.0
* Private: true
  + Indicates that this project is not intended to be published as a package to a registry like npm.

**Summary**

The above dependencies provide a robust setup for developing a modern React application. They cover a wide range of functionalities, including UI components, animations, HTTP requests, testing, routing, real-time communication, and more. Utilizing these libraries helps in maintaining code quality, enhancing user experience, and ensuring a smooth development workflow.



**BACKEND DEPENDENCIES**

**Project Dependencies of Backend**

**General Information**

This project is built using various libraries and frameworks to enhance its functionality and development process. The following are the key dependencies used in the backend development of the project.

**List of Dependencies**

1. bcryptjs: ^2.4.3
   * A library to help you hash passwords. It provides synchronous and asynchronous methods for hashing and comparing hashed passwords.
2. colors: ^1.4.0
   * A library for adding color and style to the console output, useful for enhancing the readability of log messages.
3. dotenv: ^9.0.2
   * A zero-dependency module that loads environment variables from a .env file into process.env. Useful for managing configuration settings.
4. express: ^4.17.1
   * A minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications.
5. express-async-handler: ^1.1.4
   * Simple middleware for handling exceptions inside async express routes and passing them to your express error handlers.
6. jsonwebtoken: ^8.5.1
   * An implementation of JSON Web Tokens (JWT) that allows for encoding and decoding tokens used for authentication and secure data transmission.
7. mongoose: ^5.12.9
   * An Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and translates between objects in code and the representation of those objects in MongoDB.
8. nodemon: ^2.0.7
   * A utility that will monitor for any changes in your source and automatically restart your server. Perfect for development environments.
9. socket.io: ^4.1.2
   * A library that enables real-time, bidirectional and event-based communication between web clients and servers. Useful for building real-time applications like chat apps.

**Summary**

The above dependencies provide a robust setup for developing a modern backend application using Node.js. They cover a wide range of functionalities, including password hashing, environment variable management, web server creation, asynchronous error handling, JSON Web Token creation, MongoDB interactions, automatic server restarts during development, and real-time communication. Utilizing these libraries helps in maintaining code quality, enhancing security, and ensuring a smooth development workflow.

**Reference used : -**

<https://react.dev/>

<https://socket.io/>

<https://www.w3schools.com/react/react_components.asp>

<https://www.geeksforgeeks.org/how-to-connect-mongodb-with-reactjs/>

<https://reactrouter.com/en/main>

https://chatgpt.com/

**DIVISION OF WORK IN TEAM:**

|  |  |  |
| --- | --- | --- |
| **Name of student** | **ID** | **Role** |
| Shubham | 2211981381 | Backend / Database / Deployment |
| Shashank Rajput | 2211981361 | Frontend / Database / Backend |
| Shreyansh | 2211981378 | Deployment / Backend |
| Shreya | 2211981477 | Frontend / Github |

**Result Page Description**

The result page of our chat application project provides an overview of the accomplishments, outcomes, and impact of the project. Here’s a detailed description of the content and insights presented on this page:

**Project Accomplishments**

1. **Implemented Features**:
   * A comprehensive list of all the features successfully implemented in the chat application, including user authentication, real-time messaging, chat rooms, user profiles, and group management.
2. **Technological Stack**:
   * Details about the technologies used in the project, such as the MERN stack (MongoDB, Express.js, React.js, Node.js) for full-stack development, and Socket.IO for real-time communication.

**Key Learnings**

1. **Technical Skills Gained**:
   * An overview of the technical skills acquired during the development process, including proficiency in JavaScript, React.js, Node.js, MongoDB, and Socket.IO. This section highlights the practical application of theoretical knowledge learned throughout the project.
2. **Project Management Experience**:
   * Insights into project management practices learned during the project, such as requirement analysis, system design, implementation planning, testing strategies, deployment, and documentation.

**Challenges and Solutions**

1. **Technical Challenges**:
   * Identification of technical challenges encountered during the project, such as integrating real-time communication with Socket.IO or optimizing database queries for performance.
2. **Solutions Implemented**:
   * Description of the solutions devised to overcome these challenges, including code optimizations, debugging techniques, and collaboration with team members or mentors.

**Project Impact**

1. **Future Enhancements**:
   * Ideas and plans for future enhancements and iterations of the chat application, based on user feedback and emerging technologies.

**Conclusion**

The result page serves as a comprehensive reflection of the achievements and learnings from the chat application project. It highlights the successful implementation of features, the acquisition of technical skills, the challenges faced, solutions implemented, and the overall impact of the project. This page provides valuable insights for stakeholders, team members, and future developers who may build upon the project.