

**WEB BASED COMMUNICATION TOOL**

**A PROJECT REPORT**

**Submitted by**

**MOHAMED NADHEEM M**

**(Reg. No: 24MCR065)**

**SANJEEVI G**

**(Reg. No: 24MCR090)**

**YASIGA R**

**(Reg. No: 24MCR125)**

*in partial fulfillment of the requirements  
for the award of the degree  
of*

**MASTER OF COMPUTER APPLICATIONS**

**DEPARTMENT OF COMPUTER APPLICATIONS**



**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI, ERODE – 638 060**

**DECEMBER 2024**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**KONGU ENGINEERING COLLEGE**  
**(Autonomous)**

**PERUNDURAI, ERODE – 638 060**  
**DECEMBER 2024**

**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**WEB BASED COMMUNICATION TOOL**” is the bonafide record of project work done by **MOHAMED NADHEEM M (24MCR065)**, **SANJEEVI G (24MCR090)** and **YASIGA R (24MCR125)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

**SUPERVISOR**

**HEAD OF THE DEPARTMENT**

**(Signature with seal)**

**Date:**

Submitted for the end semester viva voce examination held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **DECLARATION**

We affirm that the project report entitled “**WEB BASED COMMUNICATION TOOL**” being submitted in partial fulfillment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**MOHAMED NADHEEM M**  
**(Reg. No: 24MCR065)**

**SANJEEVI G**  
**(Reg. No: 24MCR090)**

**YASIGA R**  
**(Reg. No: 24MCR125)**

I certify that the declaration made by the above candidates is true to the best of my knowledge.

**Date:**

Name and Signature of the supervisor

**(Dr.A.Tamilarasi)**

## **ABSTRACT**

The project titled as web-based communication tool aims to designed to independently of mobile connectivity. Users can access it directly from any web browser without relying on a mobile device for authentication or synchronization, ensuring flexibility and uninterrupted communication.

This system supports one-to-one chats, group conversations, user search, profile viewing, and real-time notifications, enabling dynamic and interactive communication. The platform supports JWT-based authentication for secure login and data protection. Socket.io powers the real-time messaging feature, enabling instant communication among users.

Its responsive design ensures compatibility across devices, offering a user-friendly interface and smooth user experience. It is built using React for the frontend, Node.js for the backend, and MongoDB as the database, ensuring a robust and scalable system.

## ACKNOWLEDGEMENT

We respect and thank our correspondent, **Thiru.A.K. Ilango BCom., MBA., LLB.,** and our principal, **Dr. V. Balusamy BE (Hons.), MTech., PhD,** Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of Department, Professor **Dr.A.TAMILARASI, MSc., MPhil., PhD., M.Tech.,** Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work successful.

We would also like to express our gratitude and sincere thanks to our project coordinator, **Mrs.S.HEMALATHA MCA.,** Associate Professor(Sr.G), Department of Computer Applications, Kongu Engineering College, who have motivated us in all aspects to complete the project within the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide, **Dr.A.TAMILARASI, M.Sc., MPhil., PhD., M.Tech.,** Head Of The Department, Department of Computer Applications, Kongu Engineering College, for giving her valuable guidance and suggestions that helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping us to overwhelm in all proceedings. We bow our hearts and heads with heartfelt thanks to all those who gave us their warm services to succeed and achieve our work.

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	<b>ABSTRACT</b>	Iv
	<b>ACKNOWLEDGEMENT</b>	V
	<b>LIST OF FIGURES</b>	Viii
	<b>LIST OF ABBREVIATIONS</b>	Ix
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 ABOUT THE PROJECT	1
	1.2 EXISTING SYSTEM	1
	1.3 DRAWBACKS OF EXISTING SYSTEM	2
	1.4 PROPOSED SYSTEM	2
	1.5 ADVANTAGES OF THE PROPOSED SYSTEM	3
<b>2</b>	<b>SYSTEM ANALYSIS</b>	4
	2.1 IDENTIFICATION OF NEED	4
	2.2 FEASIBILITY STUDY	5
	2.2.1 Technical Feasibility	5
	2.2.2 Operational Feasibility	5
	2.2.3 Economical Feasibility	6
	2.3 SOFTWARE REQUIREMENT SPECIFICATION	6
	2.3.1 Hardware Requirements	6
	2.3.2 Software Requirements	7
<b>3</b>	<b>SYSTEM DESIGN</b>	10
	3.1 MODULE DESCRIPTION	10
	3.2 DATA FLOW DIAGRAM	12
	3.3 DATABASE DESIGN	14

	3.4 INPUT DESIGN	16
	3.5 OUTPUT DESIGN	18
<b>4</b>	<b>IMPLEMENTATION</b>	19
	4.1 CODE DESCRIPTION	19
	4.2 STANDARDIZATION OF THE CODING	19
	4.3 ERROR HANDLING	19
	4.4 USER INTERFACE DESIGN	20
<b>5</b>	<b>TESTING AND RESULTS</b>	21
	5.1 TESTING	21
	5.1.1 Unit Testing	21
	5.1.2 Integration Testing	23
	5.1.3 Validation Testing	24
<b>6</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	25
	6.1 CONCLUSION	25
	6.2 FUTURE ENHANCEMENT	25
	<b>APPENDICES</b>	26
	A. SAMPLE CODING	26
	B. FIGURE EXPLANATION	36
	C. SCREENSHOTS	37
	<b>REFERENCES</b>	42

## LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Dataflow Diagram Level 0	13
3.2	Dataflow Diagram Level 1	14
3.3	Signup page	17
3.4	Login Page	18
3.5	Data Stored in Database	19
B.1	Signup page	37
B.2	Login page	37
B.3	Search users	38
B.4	Chat interface	38
B.5	One-to-one chat	39
B.6	Group chat	39
B.7	Notifications	40
B.8	Data storage of users	40
B.9	Data storage of chats	41
B.10	Data storage of messages	41



## LIST OF ABBREVIATIONS

HTML	Hyper Text Markup Language
JS	Java Script
DFD	Data Flow Diagram
DOM	Document Object Model
NPM	Node Package Manager
API	Application Programming Interface
UI	User Interface

## CHAPTER 1

### INTRODUCTION

#### 1.1 ABOUT THE PROJECT

The **Web-Based Communication Tool** stands out by operating independently of mobile connectivity, functioning directly through any web browser. This eliminates the need for mobile apps, ensuring flexible and uninterrupted communication, making it a reliable standalone solution for diverse user needs.

This offers secure user authentication using JWT for session management and encrypted passwords to protect data. A personalized dashboard provides an overview of recent activities, while real-time chat functionality supports one-to-one and group conversations powered by Socket.io for instant updates. Users can create and manage groups for organized discussions. The notification system delivers real-time updates on events like new messages, with stored notifications for later review. Advanced search functionality allows users to quickly find others using keywords, leveraging MongoDB's efficient text indexing. Designed with cross-platform compatibility, the tool ensures a seamless and responsive experience across web and mobile devices.

#### 1.2 EXISTING SYSTEM

Many popular chat platforms are designed primarily for mobile devices, such as WhatsApp and Telegram. These applications are optimized for mobile use, offering features like multimedia sharing, and push notifications. However, these platforms may lack robust web-based functionalities or require a mobile device for authentication and ongoing usage.

Some chat applications, such as WhatsApp Web, require users to maintain an active connection to their mobile device. These systems rely on the mobile app to synchronize data, authenticate sessions, and maintain connectivity. If the mobile device loses internet access or battery power, the web-based chat application becomes unusable.

### 1.3 DRAWBACKS OF EXISTING SYSTEM

- Web versions of chat applications often require active syncing with a mobile device, leading to disruptions when the mobile device is unavailable.
- Mobile-only applications restrict users who prefer or require web-based interfaces for professional or accessibility reasons.
- Users must rely on multiple platforms to balance mobile and web usage effectively, leading to inconsistencies in user experience.

### 1.4 PROPOSED SYSTEM

The **Web-Based Communication Tool** offers **Independent Web Functionality**, allowing users to access the platform directly from any web browser without relying on mobile devices for authentication or data synchronization. It features **User-Friendly Authentication**, utilizing secure JWT-based login and session management for multi-device access. The platform supports **Real-Time Communication**, including one-to-one and group chats, with instant message delivery powered by Socket.io.

For enhanced collaboration, **Group Chat and Collaboration** features enable users to create and manage chat groups efficiently, while the **Notifications System** keeps users updated on messages and group activities in real-time. The tool's **Search and Discovery** capabilities allow users to locate users quickly, backed by optimized indexing.

With a strong focus on **Security and Privacy**, the system ensures encrypted data storage and secure access using JWT tokens. Its **Cross-Platform Accessibility** guarantees seamless operation across devices with a fully responsive design.

## **1.5 ADVANTAGES OF THE PROPOSED SYSTEM**

- Users can access the chat platform anytime, anywhere, without relying on a mobile device.
- Features like notifications and search streamline user interactions.
- Real-time communication ensures instant delivery and response.
- A standalone web application eliminates the need for extensive mobile app dependency or syncing.
- A clean, intuitive interface ensures ease of use for users of all backgrounds.

## **CHAPTER 2**

### **SYSTEM ANALYSIS**

#### **2.1 IDENTIFICATION OF NEED**

##### **Efficiency Unleashed:**

Traditional communication methods often face challenges such as scattered information, delayed responses, and inefficiencies in manual processes. A web-based communication tool streamlines workflows by automating tasks such as message delivery, group management, user searches, and notifications. This automation reduces errors, accelerates communication, and ensures smooth collaboration among users.

##### **Seamless User Connections:**

Users today expect hassle-free, intuitive, and accessible communication platforms. A web-based tool provides a user-friendly experience, enabling effortless one-to-one and group chats, seamless user profile management, and efficient notifications. Such tools are vital for fostering collaboration in both personal and professional settings.

##### **Tech-Savvy Competition:**

The communication industry is highly competitive, with a constant demand for innovation. Adopting web-based solutions provides a competitive edge by offering unique features like cross-device compatibility, independence from mobile connectivity, and real-time messaging. Advanced functionalities such as search optimization, push notifications, and JWT-based secure logins attract a wider audience and enhance the platform's usability.

##### **Smart Management and Precision:**

Web-based communication platforms deliver tangible benefits through data accuracy and automation. Features like real-time synchronization, search efficiency, and secure user authentication ensure reliability and accuracy in communication. These advantages lead to improved user engagement, reduced operational costs, and streamlined workflows for organizations.

### **User Delight at Every Step:**

Implementing a web-based communication tool significantly enhances the user experience. Its real-time messaging, accessible design, and intuitive user interface create a satisfying user journey. By addressing user needs effectively, the platform fosters loyalty and long-term engagement.

## **2.2 FEASIBILITY STUDY**

A feasibility study for a web-based communication tool is critical for determining its viability and success. This involves evaluating technical, operational, and economic aspects to assess the practicality of implementation.

### **2.2.1 Technical Feasibility**

The technical feasibility examines whether the proposed system can be implemented using available technologies and resources. Key considerations include:

- **System Requirements:** The platform leverages **React.js** for a responsive and dynamic frontend, **Node.js** for scalable backend operations, and **MongoDB** for efficient data management.
- **Real-Time Features:** The integration of **Socket.io** ensures smooth real-time messaging and notifications.
- **Cross-Device Compatibility:** The responsive design guarantees seamless performance across desktops, laptops, and tablets.
- **Security:** Secure authentication using **JWT** tokens and encryption protocols addresses data privacy concerns.

### **2.2.2 Operational Feasibility**

Operational feasibility evaluates how effectively the proposed system aligns with user needs and existing business processes. Key considerations include:

- **Ease of Use:** A user-friendly interface ensures that users can quickly adapt to the platform.

- **Reliability:** Real-time messaging and stable connections enhance the user experience.
- **Integration:** The system is designed to integrate seamlessly with existing workflows and corporate communication processes.
- **User-Centric Design:** Features like notifications, search capabilities, and group management are tailored to meet diverse operational needs.

### 2.2.3 Economic Feasibility

Economic feasibility focuses on the financial viability of the system. Key considerations include:

- **Cost-Effectiveness:** Leveraging open-source technologies like React, Node.js, and MongoDB reduces development costs.
- **Low Maintenance Costs:** Cloud-based deployment minimizes hardware expenses and ensures scalability.
- **ROI Analysis:** The benefits, such as increased user engagement, efficiency, and competitive advantage, outweigh the initial investment.

## 2.3 SOFTWARE REQUIREMENT SPECIFICATION

A declaration that describes the capability that a system needs to satisfy the user's requirements is known as a system requirement. The system requirements for specific machines, software, or business operations are general. Taking it all the way down to the hardware and coding that operates the software. System requirements are the most efficient way to address user needs while lowering implementation costs.

### 2.3.1 Hardware Requirements

The hardware for the system is selected considering factors such as CPU processing speed, memory access, peripheral channel access speed, printed speed; seek time & relational data of hard disk and communication speed, etc.

Processor	:	Intel core i5
RAM	:	8.00 GB
Monitor	:	15''VGA MonitorHard
Disk	:	512 GB
Keyboard	:	ACCUTYPE keyboard
Mouse	:	Optical Mouse

### 2.3.2 Software Requirements

The software for the project is selected considering factors such as working front-end environment, flexibility in the coding language, database knowledge of enhanced backend technology, etc.

Operating System	: Windows 11
Front End	: Reactjs
Back End	: Nodejs
Database	: MongoDB
Tools	: Visual studio code

#### Front End

##### React JS

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application. The primary goal of ReactJS is to create User Interfaces (UI) that accelerate apps. It makes use of virtual DOM (JavaScript object), which enhances the app's performance. Faster than the standard DOM is the Javascript virtual DOM. ReactJS integrates with different frameworks and can be used on the client and server sides.

It makes use of components and data patterns to make larger apps more readable and maintainable. A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks.

ReactJS uses a virtual DOM-based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading the complete DOM every time. NodeJS and NPM are the platforms needed to develop any ReactJS application. React-Bootstrap replaces the Bootstrap JavaScript. Each component has been built from scratch as a true React component. The Bootstrap JavaScript is replaced by React-Bootstrap.



Each component was created from the ground up as a genuine React component, free of other dependencies like jQuery. React- Bootstrap is one of the first React libraries, and because it has developed and matured alongside React, it is a great option for your UI foundation. ReactJS primary goal is to create User Interfaces (UI) that increase the speed of apps. It makes use of virtual DOM (JavaScript object), which enhances the app's performance. Faster than the standard DOM is the JavaScript virtual DOM. ReactJS integrates with different frameworks and can be used on the client and server sides. It makes use of components and data patterns to make larger apps more readable and maintainable.

## **Server**

### **Node JS**

Node.js is a server-side JavaScript runtime built on Chrome's V8 JavaScript engine. It allows you to run JavaScript on the server, which can be used to build backend services, such as web servers. Node.js is commonly used for building server-side applications, as it provides an easy way to create a network application that can handle a large number of concurrent connections.

Node.js is an event-driven runtime, meaning that it allows you to build applications that can handle events and perform certain actions in response to those events. It is designed to be lightweight and efficient, making it well-suited for building scalable network applications. One of the main benefits of using Node.js is that it allows you to use JavaScript on both the frontend and back end of an application. This can make it easier to develop full-stack applications, as you can use the same language throughout the entire application.

### **Database**

MongoDB is a popular NoSQL database system that provides a flexible and scalable approach to storing and handling data. Unlike traditional relational databases, MongoDB is schema-less, which means developers can store and retrieve data in a more dynamic and agile manner. MongoDB is a document-oriented database in which data is stored in BSON (Binary JSON)-like documents. Each document is a JSON-like object with field-value pairs, which allows for nested structures and arrays. This flexibility makes it ideal for handling complex and dynamic data. Data is organized into collections, which are similar to tables in relational databases. Each collection consists of documents, and each document represents a record in the collection.

Collections and documents can be created and modified on-the-fly without requiring a predefined schema. MongoDB uses a rich query language that supports a wide range of queries, including filtering, sorting, and aggregation. It allows developers to perform complex queries and retrieve specific data from large datasets efficiently. Indexing is a crucial feature in MongoDB for optimizing query performance. Developers can create indexes on fields to speed up data retrieval, particularly for frequently queried fields. MongoDB uses a JSON-like syntax for queries, making it easy for developers to work with and understand. This aligns well with modern web development practices, where JSON is a common data interchange format. MongoDB includes a powerful aggregation framework that enables developers to perform data transformations, calculations, and aggregations directly within the database. This reduces the need for multiple round-trips between the application and the database. MongoDB has an active and vibrant community that provides ample resources, documentation, and support. Additionally, it offers official drivers for various programming languages.

## **CHAPTER 3**

### **SYSTEM DESIGN**

#### **3.1 MODULE DESCRIPTION**

A module description provides detailed information about the module and its supported components, which is accessible in different manners.

The project contains the following modules:

- Login/Signup
- Search Users
- One-to-One Chats
- Group Chats
- Notification
- View User Profile

##### **3.1.1 Login/Signup**

This module manages user registration and authentication. New users can create an account by providing their email, name, password, and profile details. Users can log in with their email and password. The system validates credentials and generates a JWT token for secure session management. JWT tokens ensure users stay logged in during active sessions, with an automatic logout upon token expiration.

##### **3.1.2 Search Users**

This module provides an advanced search functionality for finding users or groups. Users can search for others by username, email, or name. Uses MongoDB's text indexing for quick and efficient querying, even with large datasets. Displays relevant user profiles with direct links to start chats.

### **3.1.3 One-to-One Chats**

This module enables private, real-time messaging between two users. Users can start conversations with any registered user, allowing users to engage in private, real-time conversations. Powered by Socket.io, the system ensures instant message delivery and a smooth communication experience. All conversations are stored securely in the database for future reference.

### **3.1.4 Group Chats**

This module supports group communication for collaborative discussions. Users can assign names, add members and remove users while creating groups. Group members can send messages and announcements visible to all participants.

### **3.1.5 Notification**

This module ensures users are promptly notified of important events. Alerts users about new messages and group updates. Notifications are stored in the database and displayed until dismissed or marked as read. Includes chats messages and group messages.

### **3.1.6 View User Profile**

This module allows users to access basic information about other users. Profiles display details such as username, email and profile picture. Users can customize their profiles by uploading pictures while signup.

### 3.2 DATAFLOW DIAGRAM

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through.

#### LEVEL 0

The system initiates communication by allowing users to securely log in or sign up, enabling real-time messaging and group chats. Messages are processed, delivered instantly, and assigned unique IDs for easy tracking. The admin manages user activities, moderates groups, and ensures smooth platform operations, with notifications keeping users updated on important events. The following process is described in the below figure 3.1



Figure 3.1 Dataflow Diagram Level 0

## LEVEL 1

The system records each communication session with unique IDs, capturing user details, chat history, and activity status. Notifications or updates keep users informed of message delivery or system changes. Additionally, it allows users to provide feedback, storing reviews and ratings in the database to enhance user experience and platform reliability. The order details and issues are explained in the following diagram figure 3.2.

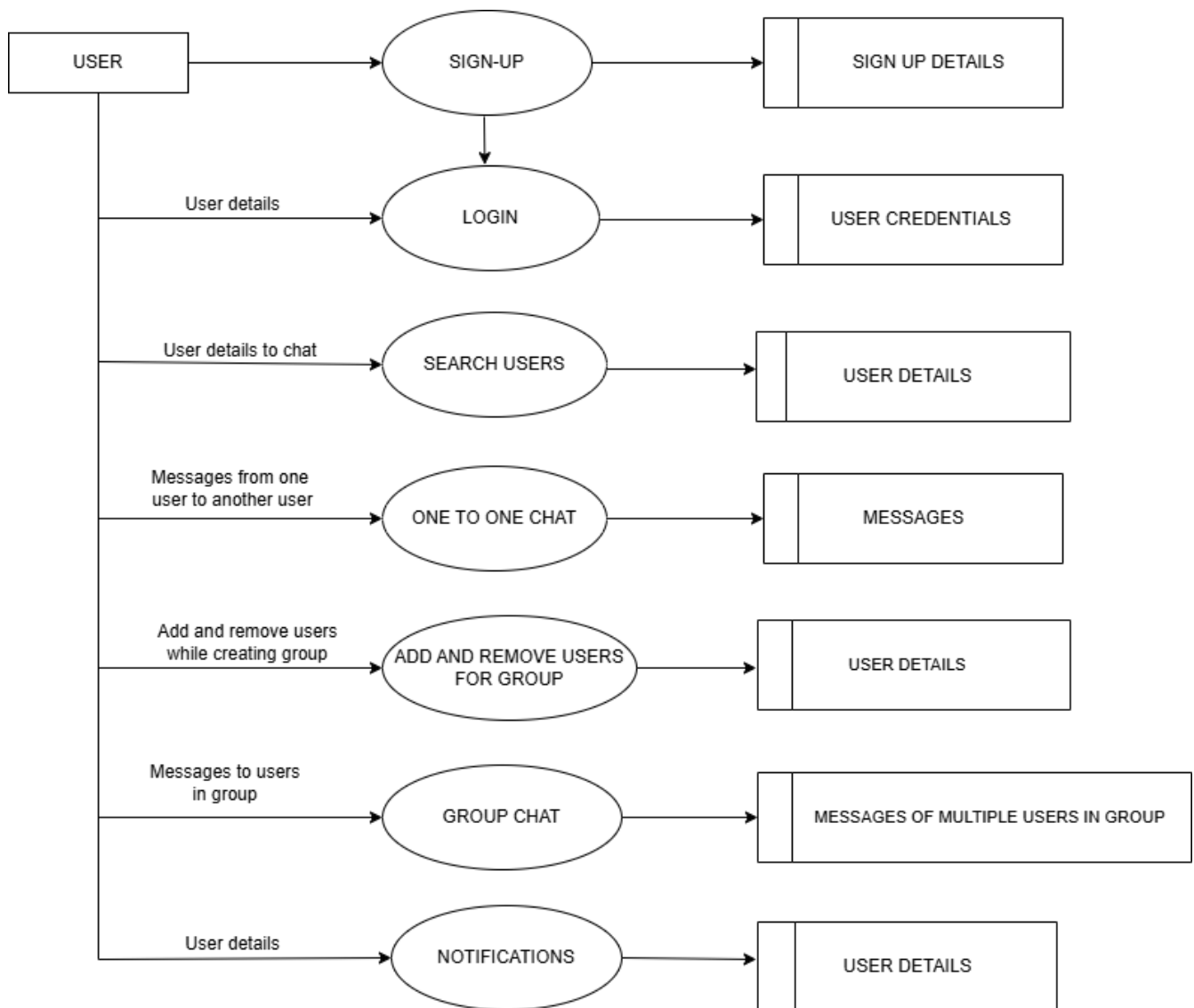


Figure 3.2 Dataflow Diagram Level 1

### **3.3 DATABASE DESIGN**

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data into the database model. Database management system manages the data accordingly. The term database design can be used to describe many different parts of the design of an overall database system.

Principally, and most correctly, it can be thought of as the logical design of the base data structure used to store the data. In an object database, the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structure, but also the forms and queries used as part of the overall database application within the database management system.

#### **Data Integration**

In a database, information from several files is coordinated, accessed, and operated upon as though it is in a single file. Logically, the information is centralized, physically, the data may be located on different devices, connected through data communication facilities. Data integrity means storing all data in one place only and determining how each application has to access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications., which use it. This leads to less data redundancy, data items need not be duplicated, and a reduction in the direct access storage requirement.

#### **Data Independence**

Data independence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming applications and to allow modifications to application programs without recognizing the physical data.

If the database changes and expands over time, the changes in one level must not affect the data at other levels of the database. The table needed for each module was designed and the specification of every column was given based on the records and details collected during the record specification of the system study, which is shown below.

Data independence is a critical aspect of the online web based communication tool's architecture, ensuring adaptability and flexibility. Logical data independence allows for modifications to the database's conceptual schema without necessitating changes to application programs. For example, introducing new features or altering user profiles can be seamlessly accommodated in the database structure without impacting existing application code. Similarly, physical data independence ensures that changes to the underlying storage structures or database management systems can be implemented without affecting the application's data retrieval and manipulation processes. This separation between data structure and application logic enhances the system's resilience to changes, simplifying maintenance and promoting long-term scalability.

### **Data Security**

Data security refers to the process of protecting data from unauthorized access and data corruption throughout its lifecycle. Data security includes data encryption, hashing, tokenization, and key management practices that protect data across all applications and platforms. Data security means protecting digital data, such as those in a database, from destructive forces and from unwanted actions of unauthorized users, such as cyberattacks or data breaches.

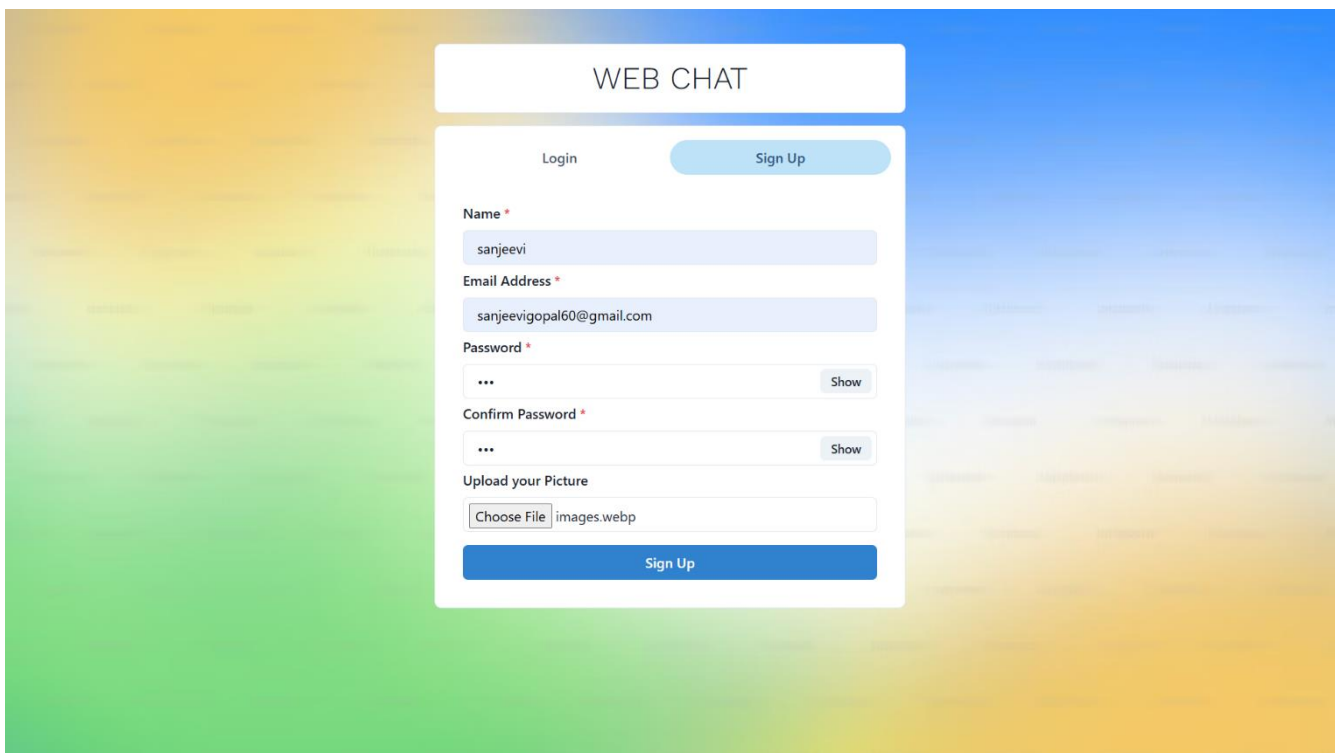


### 3.4 INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer- understandable format. Input design is one of the most expensive phases of the operation of a computerized system and is often the major problem of a system. A large number of problems with a system can usually be tracked back to fault input design and method.

Every moment of input design should be analyzed and designed with utmost care. The decisions made during the input design are the project gives the low time consumption to make sensitive application made simple. Thus, the developed system is well within the budget. This was achieved because most of the technologies used are freely available. Only the customized product had to be purchased. In the project, the forms are designed with easy-to-use options. The coding is being done such that proper validation is made to get the perfect input. No error inputs are accepted.

#### Sign up Page



WEB CHAT

Login Sign Up

Name \*  
sanjeevi

Email Address \*  
sanjeevigopal60@gmail.com

Password \*  
... Show

Confirm Password \*  
... Show

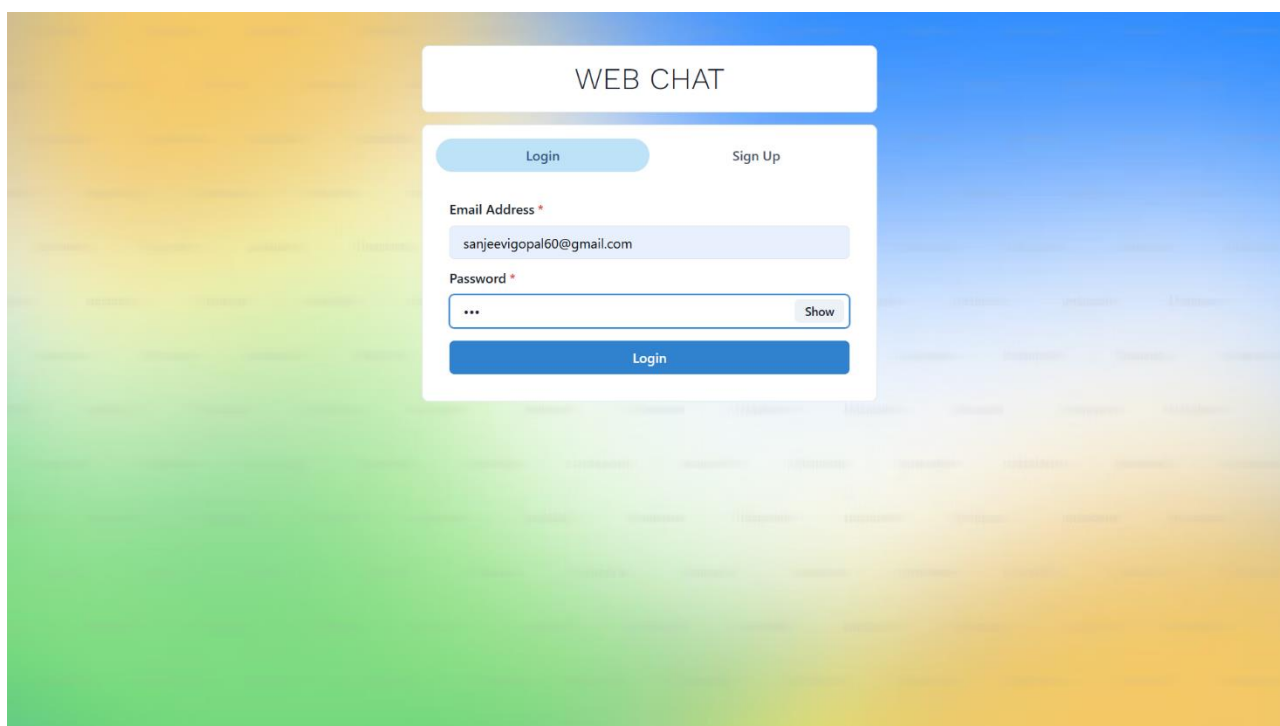
Upload your Picture  
Choose File images.webp

Sign Up

FIGURE 3.3 SIGNUP PAGE

## Login page

This is a admin login form, the admin can log in with their registered name and password to access the application. After login, the user can view the application. The figure 3.4 shows the admin logins with the particular password and directs to the specific page.



WEB CHAT

Login Sign Up

Email Address \*  
sanjeevigopal60@gmail.com

Password \*  
\*\*\* Show

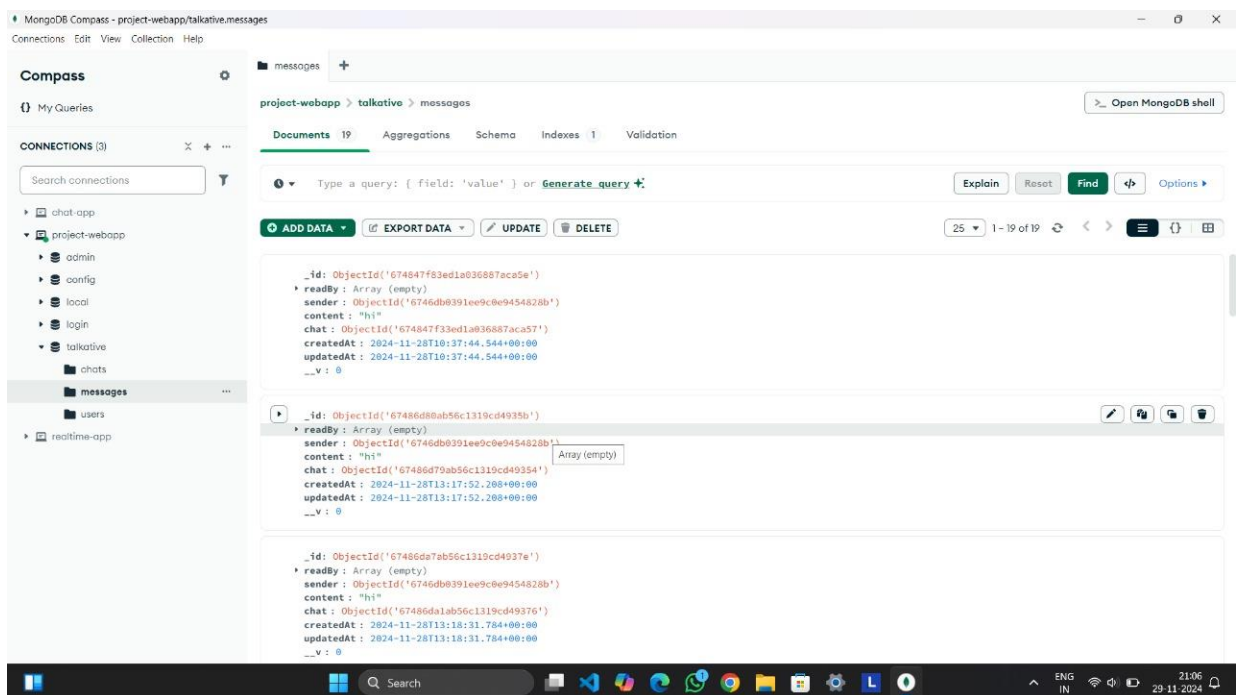
Login

**FIGURE 3.4 LOGIN PAGE**

### 3.5 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; it should be understandable with the enhanced format. Computer output is the most important direct source of information to the user. Output design deals with form design. Efficient output design should improve the interfacing with the user. The term output applies to any information produced by an information system in terms of data displayed. When analysts design system output, they identify the specific output that is needed to meet the requirements of the end user.

Previewing the output reports by the user is crucial because they are the ultimate judge of the quality of the output and the success of the system. System analysis helps to determine which applications, websites or documents are allowed or blocked. The output should be designed in an attractive, convenient, and informative manner. The following figure 3.5 shows the data storage of messages by the users.



**FIGURE 3.5 DATA STORED IN DATABASE**

## **CHAPTER 4**

### **IMPLEMENTATION**

#### **4.1 CODE DESCRIPTION**

Code description can be used to summarize code or to explain the programmer's intent. Good comments don't repeat the code or explain it. They clarify its intent. Comments are sometimes processed in various ways to generate documentation external to the source code itself by document generator or used for integration with systems and other kinds of external programming tools. I have chosen to react JS as the frontend, node JS as backend and MongoDB as the database.

#### **4.2 STANDARDIZATION OF THE CODING**

Coding standards define a programming style. A coding standard does not usually concern itself with wrong or right in a more abstract sense. It is simply a set of rules and guidelines for the formatting of source code. The other common type of coding standard is the one used in or between development teams. Professional code performs a job in such a way that is easy to maintain and debug. All the coding standards are followed while creating this project. Coding standards become easier, the earlier you start. It is better to do a neat job than cleaning up after all is done. Every coder will have a unique pattern that he adheres to. Such a style might include the conventions he uses to name variables and functions and how he comments his work. When the said pattern and style are standardized, it pays off the effort well in the long.

#### **4.3 ERROR HANDLING**

Exception handling is a process or method used for handling abnormal statements in the code and executing them. It also enables to handle the flow control of the code/program. For handling the code, various handlers are used that process the exception and execute the code. Mainly if-else block is used to handle errors using condition checking.

In many cases many corner cases must be checked during an execution but “if-else” can only handle the defined conditions. In if-else, conditions are manually generated based on the task. An error is a serious problem and an application does not usually get passed without incident. Errors cause an application to crash, and ideally send an error message offering some suggestion to resolve the problem and return to a normal operating state, There is no way to deal with errors “live” or in production the only solution is to detect them via error monitoring and bug tracking and dispatch a developer or two to sort out the code.

Exceptions, on the other hand, are exceptional conditions an application should reasonably be accepted to handle. A programming language allows developers to include try...catch statements to handle exceptions and apply a sequence of logic to deal with the situation instead of crashing. And when an application encounters an exception that there’s no workaround for, that’s called an unhandled exception, which is the same thing as an error.

#### **4.4 USER INTERFACE DESIGN**

Input design is the process of converting user-originated inputs to a computer-understandable format. Input design is one of the most expensive phases of the operation of a computerized system and is often the major problem of a system. A large number of problems with a system can usually be tracked back to fault input design and method. Every moment of input design should be analyzed and designed with utmost care.

- 369-9+· To provide a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is understood by the user.

System analysis decides the following input design details, what data to input, what medium to use, how the data should be arranged or coded, data items and transactions needing validation to detect errors, and at last, the dialogue to guide the user in providing input.

## **CHAPTER 5**

### **TESTING AND RESULTS**

#### **5.1 TESTING**

Software testing serves as the final assessment of specifications, designs, and coding and is a crucial component of software quality assurance. The system is tested throughout the testing phase utilizing diverse test data. The preparation of test data is essential to the system testing process. The system under study is tested after the test data preparation. Once the source code is complete, relevant data structures should be documented. The finished project must go through testing and validation when errors are explicitly targeted and attempted to be found.

#### **TYPES OF TESTING**

The various types of testing

- Unit testing
- Integration Testing
- Validation Testing

##### **5.1.1 Unit Testing**

A testing strategy known as unit and integration testing has been used to check that the system behaves as expected. The testing strategy was based on the functionality and the requirements of the system. In unit checking out, we have to check the applications making up the device. For this reason, Unit checking out once in a while is referred to as a program checking out. The software programs in a device are the modules and workouts that might be assembled and included to carry out a selected function, Unit checking out the first at the modules independently of 1 another, to find mistakes.

This enables, to stumble on mistakes in coding and common sense which might be contained with the module alone. The checking out became completed at some stage in programming level itself.

#### Test Case 1

Module	: User Login
Input	: Username and Password
Output	: Redirect to home page

#### Test 1

Module	: User Login
Username	: sanjeevigopal60@gmail.com
Password	: 123
Output	: Logged in successfully
Event	: Button click
Analysis	: Username and Password has been verified

### 5.1.2 Integration Testing

Integration testing is done to test if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus, the manual test data that is used to test the interfaces is replaced by that which is generated automatically from the various modules. It can be used for testing how the module would interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces.

#### Test Case 1

Module	: User Login
Input	: Email and Password
Output	: Redirect to home page

#### Test 1

Module	: User Login
Email	: sanjeevigopal60@gmail.com
Password	: 123
Output	: Redirected to approval page
Analysis	: Email Address and Password has been verified

#### Test 2

Module	: User Login
Email	: <a href="mailto:abi@gmail.com">abi@gmail.com</a>
Password	: 123456
Output	: Email Address or password are invalid
Analysis	: Email Address and Password has been checked and error shown



### 5.1.3 Validation Testing

Verification and validation checking out are critical tests, which might be achieved earlier than the product has been surpassed over to the user. This makes sure, that the software program checking out lifestyles cycle begins off evolved early. Each verification and validation intends to make certain that the product is made in step with the necessities of the user and does certainly fulfill the supposed purpose.

Test case 1

Module : User Registration

Input : Email and Password

Test 1

Module : User Registration

Email : sanjeevigopal60@gmail.com

Password : 123

Output : Registered successfully

## **CHAPTER 6**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **6.1 CONCLUSION**

The web-based communication tool has significantly enhanced how users connect and interact, offering robust features like one-to-one chat, group chat, user search, profile viewing, and real-time notifications. These features streamline communication, making it easier for users to stay connected, manage conversations, and find others with ease. The platform's scalability ensures it meets evolving communication needs, while its focus on security and seamless user experience positions it as an essential tool for personal and professional use. We thank our stakeholders for their support and look forward to continuous innovation and improvement, ensuring user satisfaction and efficient communication.

#### **6.2 FUTURE ENHANCEMENT**

Looking ahead, the future of the web-based communication tool presents exciting opportunities for enhancement. Continuous improvements will focus on refining the user interface to ensure an even more seamless and intuitive experience. One of the key features under consideration is the integration of audio and video calling, allowing users to engage in real-time conversations beyond text-based messaging. This will enhance the platform's versatility, meeting the growing demand for multimedia communication. Additionally, further optimization for mobile devices and integration with emerging technologies will ensure the system stays responsive to evolving user needs. As the communication landscape evolves, our commitment to innovation will drive updates, keeping the platform at the forefront of modern communication tools.

## APPENDICES

### A.SAMPLE CODING

```
import { Button } from "@chakra-ui/button";
import { FormControl, FormLabel } from "@chakra-ui/form-control";
import { Input, InputGroup, InputRightElement } from "@chakra-ui/input";
import { VStack } from "@chakra-ui/layout";
import { useState } from "react";
import axios from "axios";
import { useToast } from "@chakra-ui/react";
import { useHistory } from "react-router-dom";
import { ChatState } from "../../Context/ChatProvider";

const Login = () => {
  const [show, setShow] = useState(false);
  const handleClick = () => setShow(!show);
  const toast = useToast();
  const [email, setEmail] = useState();
  const [password, setPassword] = useState();
  const [loading, setLoading] = useState(false);

  const history = useHistory();
  const { setUser } = ChatState();

  const submitHandler = async () => {
    setLoading(true);
    if (!email || !password) {
      toast({
        title: "Please Fill all the Feilds",
        status: "warning",
        duration: 5000,
        isClosable: true,
```

```

        position: "bottom",
    });
    setLoading(false);
    return;
}

try {
    const config = {
        headers: {
            "Content-type": "application/json",
        },
    };

    const { data } = await axios.post(
        "/api/user/login",
        { email, password },
        config
    );

    toast({
        title: "Login Successful",
        status: "success",
        duration: 5000,
        isClosable: true,
        position: "bottom",
    });
    setUser(data);
    localStorage.setItem("userInfo", JSON.stringify(data));
    setLoading(false);
    history.push("/chats");
} catch (error) {
    toast({
        title: "Error Occured!",
        description: error.response.data.message,
    });
}

```

```

    status: "error",
    duration: 5000,
    isClosable: true,
    position: "bottom",
  });
  setLoading(false);
}
};

return (
  <VStack spacing="10px">
    <FormControl id="email" isRequired>
      <FormLabel>Email Address</FormLabel>
      <Input
        value={email}
        type="email"
        placeholder="Enter Your Email Address"
        onChange={(e) => setEmail(e.target.value)}
      />
    </FormControl>
    <FormControl id="password" isRequired>
      <FormLabel>Password</FormLabel>
      <InputGroup size="md">
        <Input
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          type={show ? "text" : "password"}
          placeholder="Enter password"
        />
        <InputRightElement width="4.5rem">
          <Button h="1.75rem" size="sm" onClick={handleClick}>
            {show ? "Hide" : "Show"}
          </Button>
        </InputRightElement>
      </InputGroup>
    </FormControl>
  </VStack>
);

```

```

        </InputRightElement>
      </InputGroup>
    </FormControl>
    <Button
      colorScheme="blue"
      width="100%"
      style={{ marginTop: 15 }}
      onClick={submitHandler}
      isLoading={loading}
    >
      Login
    </Button>

  </VStack>
);
};

export default Login;

```

## SIGNUP.JS

```

import { Button } from "@chakra-ui/button";
import { FormControl, FormLabel } from "@chakra-ui/form-control";
import { Input, InputGroup, InputRightElement } from "@chakra-ui/input";
import { VStack } from "@chakra-ui/layout";
import { useToast } from "@chakra-ui/toast";
import axios from "axios";
import { useState } from "react";
import { useHistory } from "react-router";

const Signup = () => {
  const [show, setShow] = useState(false);
  const handleClick = () => setShow(!show);

```

```

const toast = useToast();
const history = useHistory();

const [name, setName] = useState();
const [email, setEmail] = useState();
const [confirmpassword, setConfirmpassword] = useState();
const [password, setPassword] = useState();
const [pic, setPic] = useState();
const [picLoading, setPicLoading] = useState(false);

const submitHandler = async () => {
  setPicLoading(true);
  if (!name || !email || !password || !confirmpassword) {
    toast({
      title: "Please Fill all the Feilds",
      status: "warning",
      duration: 5000,
      isClosable: true,
      position: "bottom",
    });
    setPicLoading(false);
    return;
  }
  if (password !== confirmpassword) {
    toast({
      title: "Passwords Do Not Match",
      status: "warning",
      duration: 5000,
      isClosable: true,
      position: "bottom",
    });
    return;
  }
}

```

```
console.log(name, email, password, pic);
try {
  const config = {
    headers: {
      "Content-type": "application/json",
    },
  };
  const { data } = await axios.post(
    "/api/user",
    {
      name,
      email,
      password,
      pic,
    },
    config
  );
  console.log(data);
  toast({
    title: "Registration Successful",
    status: "success",
    duration: 5000,
    isClosable: true,
    position: "bottom",
  });
  localStorage.setItem("userInfo", JSON.stringify(data));
  setPicLoading(false);
  history.push("/chats");
} catch (error) {
  toast({
    title: "Error Occured!",
    description: error.response.data.message,
    status: "error",
```



```

    duration: 5000,
    isClosable: true,
    position: "bottom",
  });
  setPicLoading(false);
}
};

const postDetails = (pics) => {
  setPicLoading(true);
  if (pics === undefined) {
    toast({
      title: "Please Select an Image!",
      status: "warning",
      duration: 5000,
      isClosable: true,
      position: "bottom",
    });
    return;
  }
  console.log(pics);
  if (pics.type === "image/jpeg" || pics.type === "image/png") {
    const data = new FormData();
    data.append("file", pics);
    data.append("upload_preset", "chat-app");
    data.append("cloud_name", "piyushproj");
    fetch("https://api.cloudinary.com/v1_1/piyushproj/image/upload", {
      method: "post",
      body: data,
    })
      .then((res) => res.json())
      .then((data) => {
        setPic(data.url.toString());

```

```

        console.log(data.url.toString());
        setPicLoading(false);
    })
    .catch((err) => {
        console.log(err);
        setPicLoading(false);
    });
} else {
    toast({
        title: "Please Select an Image!",
        status: "warning",
        duration: 5000,
        isClosable: true,
        position: "bottom",
    });
    setPicLoading(false);
    return;
}
};

return (
    <VStack spacing="5px">
        <FormControl id="first-name" isRequired>
            <FormLabel>Name</FormLabel>
            <Input
                placeholder="Enter Your Name"
                onChange={ (e) => setName(e.target.value)}
            />
        </FormControl>
        <FormControl id="email" isRequired>
            <FormLabel>Email Address</FormLabel>
            <Input
                type="email"

```

```

placeholder="Enter Your Email Address"
onChange={ (e) => setEmail(e.target.value)}
/>
</FormControl>
<FormControl id="password" isRequired>
  <FormLabel>Password</FormLabel>
  <InputGroup size="md">
    <Input
      type={ show ? "text" : "password"}
      placeholder="Enter Password"
      onChange={ (e) => setPassword(e.target.value)}
    />
    <InputRightElement width="4.5rem">
      <Button h="1.75rem" size="sm" onClick={ handleClick }>
        { show ? "Hide" : "Show"}
      </Button>
    </InputRightElement>
  </InputGroup>
</FormControl>
<FormControl id="password" isRequired>
  <FormLabel>Confirm Password</FormLabel>
  <InputGroup size="md">
    <Input
      type={ show ? "text" : "password"}
      placeholder="Confirm password"
      onChange={ (e) => setConfirmpassword(e.target.value)}
    />
    <InputRightElement width="4.5rem">
      <Button h="1.75rem" size="sm" onClick={ handleClick }>
        { show ? "Hide" : "Show"}
      </Button>
    </InputRightElement>
  </InputGroup>
</FormControl>

```

```

<FormControl id="pic">
  <FormLabel>Upload your Picture</FormLabel>
  <Input
    type="file"
    p={ 1.5}
    accept="image/*"
    onChange={ (e) => postDetails(e.target.files[0]) }
  />
</FormControl>
<Button
  colorScheme="blue"
  width="100%"
  style={{ marginTop: 15 }}
  onClick={ submitHandler }
  isLoading={ picLoading }
>
  Sign Up
</Button>
</VStack>
);
};

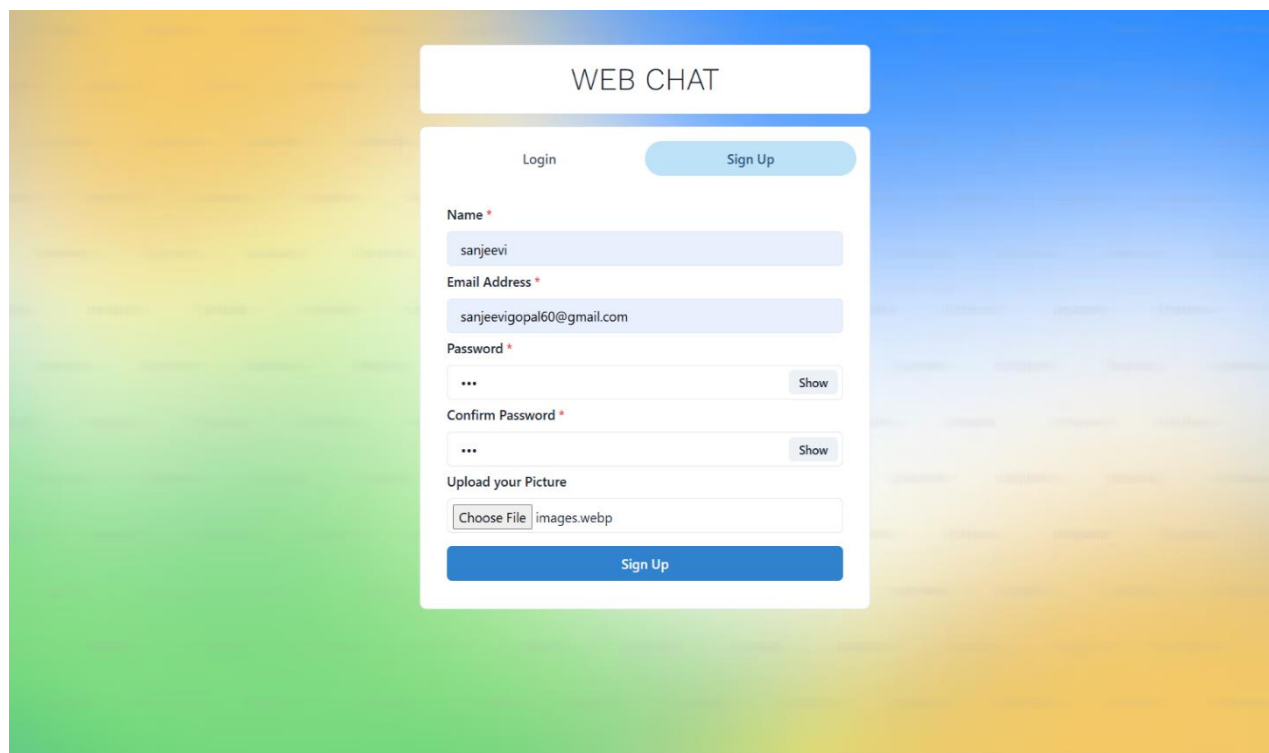
export default Signup;

```

## **B. FIGURES EXPLANATION**

The Web based communication tool website has a front page which is shown in figure B.1 depicts the login and register bars and the services. The signup page is to register the user details by giving their appropriate details such as name, email id and a password. Figure B.2 After registration, the user has to login the page with the registered email id and password it will redirect to application dashboard. Figure B.3 Shows search users of available users. Figure B.4. Shows chat interface to message each other. Figure B.5 Shows the one-to-one chat interface with typing indication. Figure B.6 Shows the group chat and admin of group can add members of the group. Figure B.7 Show how notifications look like. Figure B.8, B.9 and B.10 The data storage for the following tables like users, chats and messages are explained.

## C. SCREENSHOTS



The screenshot shows a web interface titled "WEB CHAT" with a "Sign Up" form. The form includes fields for Name, Email Address, Password, and Confirm Password, each with a "Show" button. There is also an "Upload your Picture" section with a "Choose File" button and a file name "images.webp". A large blue "Sign Up" button is at the bottom of the form. The background is a gradient of green and yellow.

WEB CHAT

Login Sign Up

Name \*  
sanjeevi

Email Address \*  
sanjeevigopal60@gmail.com

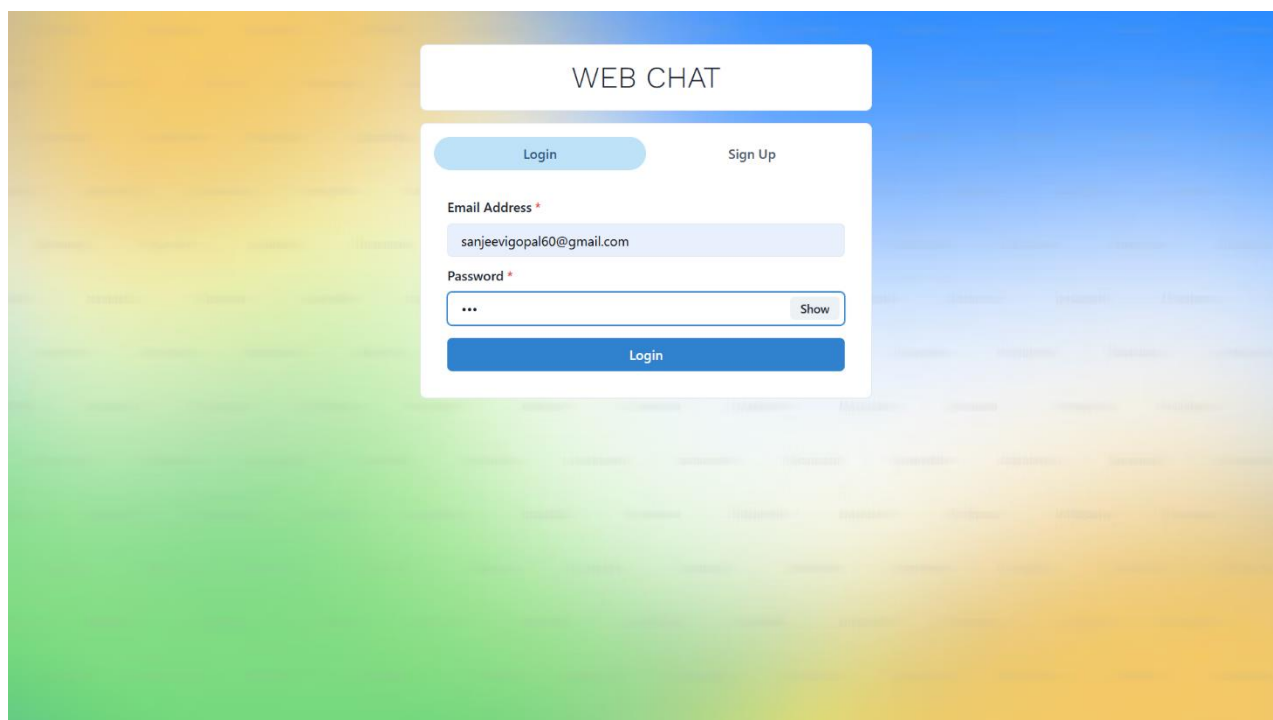
Password \*  
... Show

Confirm Password \*  
... Show

Upload your Picture  
Choose File images.webp

Sign Up

FIGURE B.1 SIGN UP PAGE



The screenshot shows a web interface titled "WEB CHAT" with a "Login" form. The form includes fields for Email Address and Password, each with a "Show" button. A large blue "Login" button is at the bottom of the form. The background is a gradient of green and yellow.

WEB CHAT

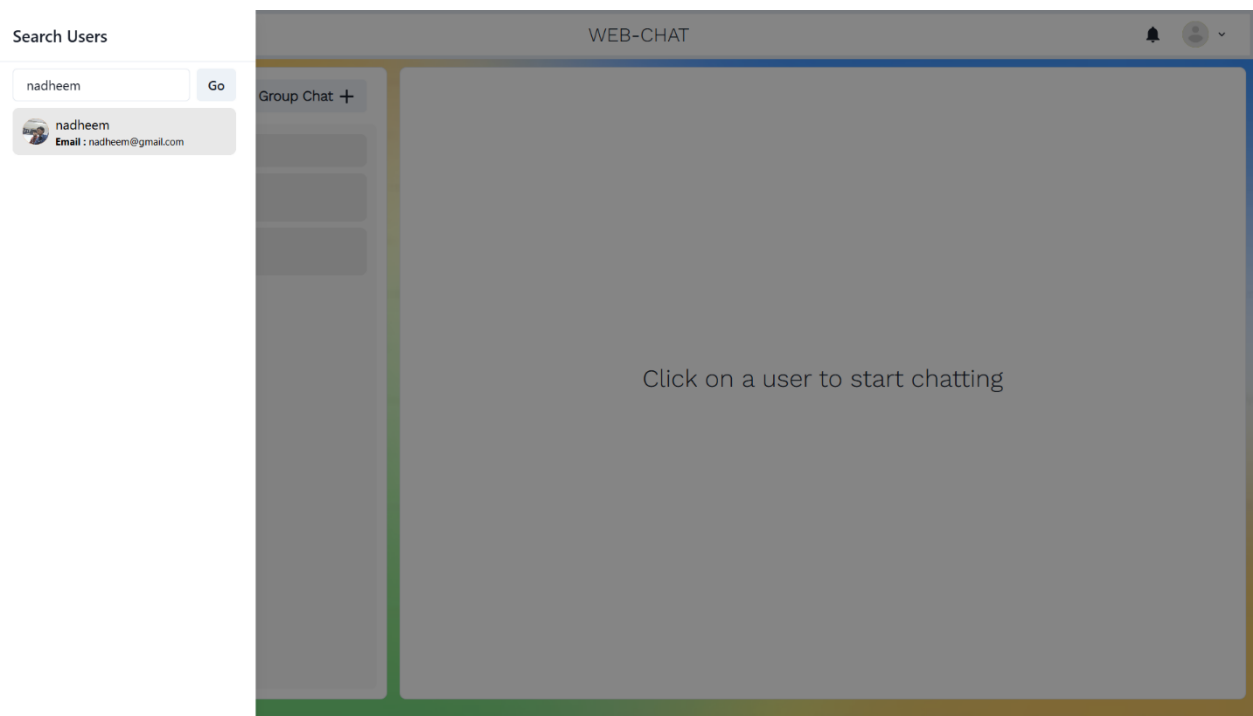
Login Sign Up

Email Address \*  
sanjeevigopal60@gmail.com

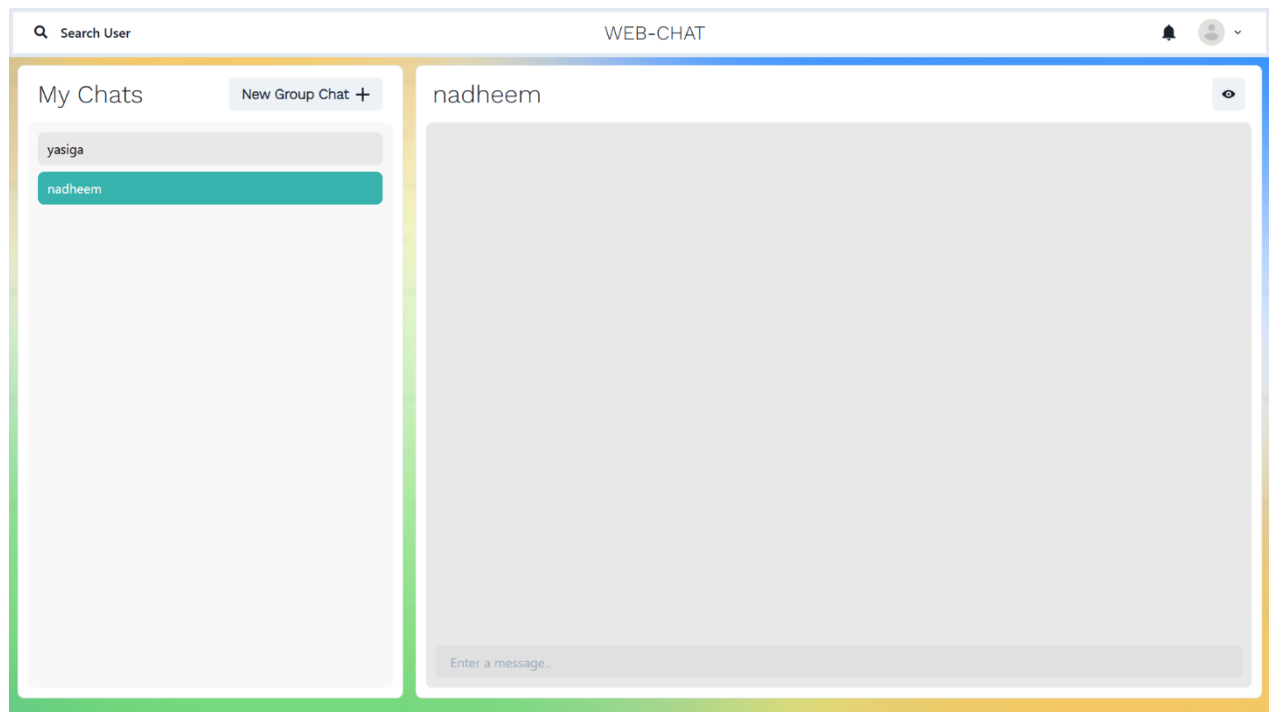
Password \*  
... Show

Login

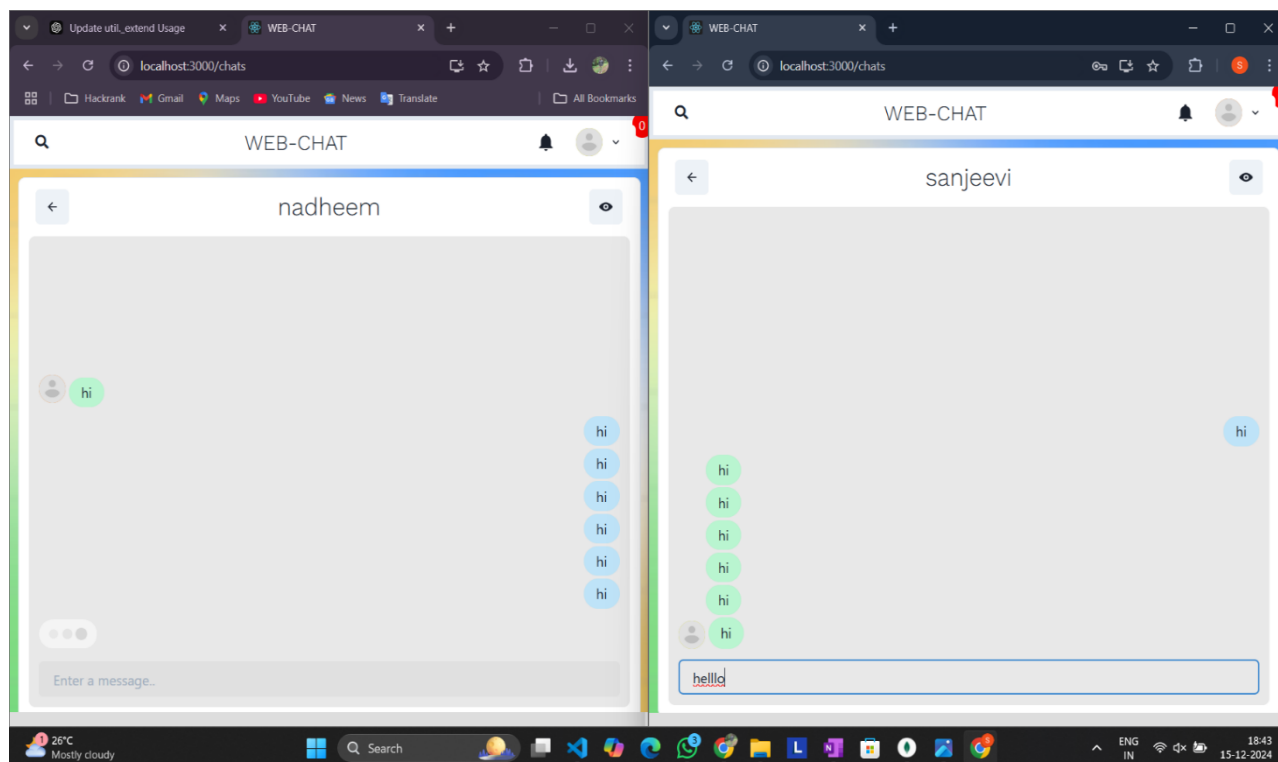
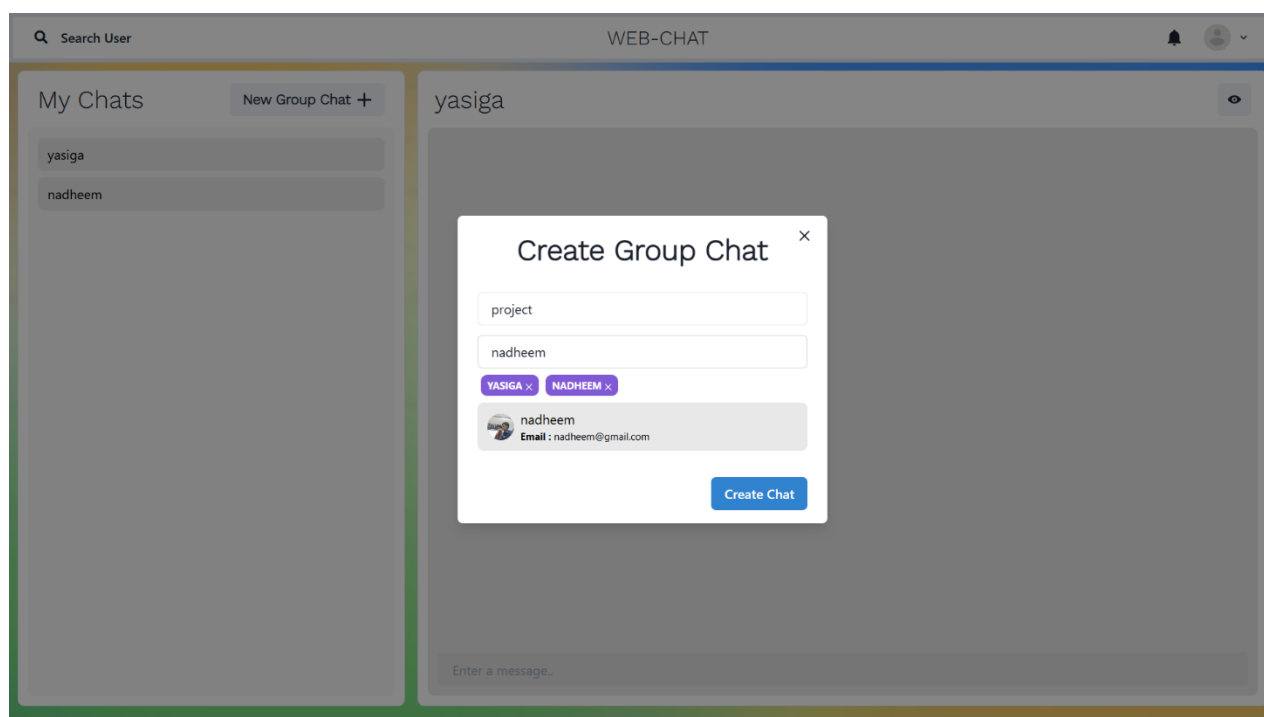
FIGURE B.2 LOGIN PAGE



**FIGURE B.3 SEARCH USERS**



**FIGURE B.4 CHAT INTERFACE**

**FIGURE B.5 ONE-TO-ONE CHAT****FIGURE B.6 GROUP CHAT**



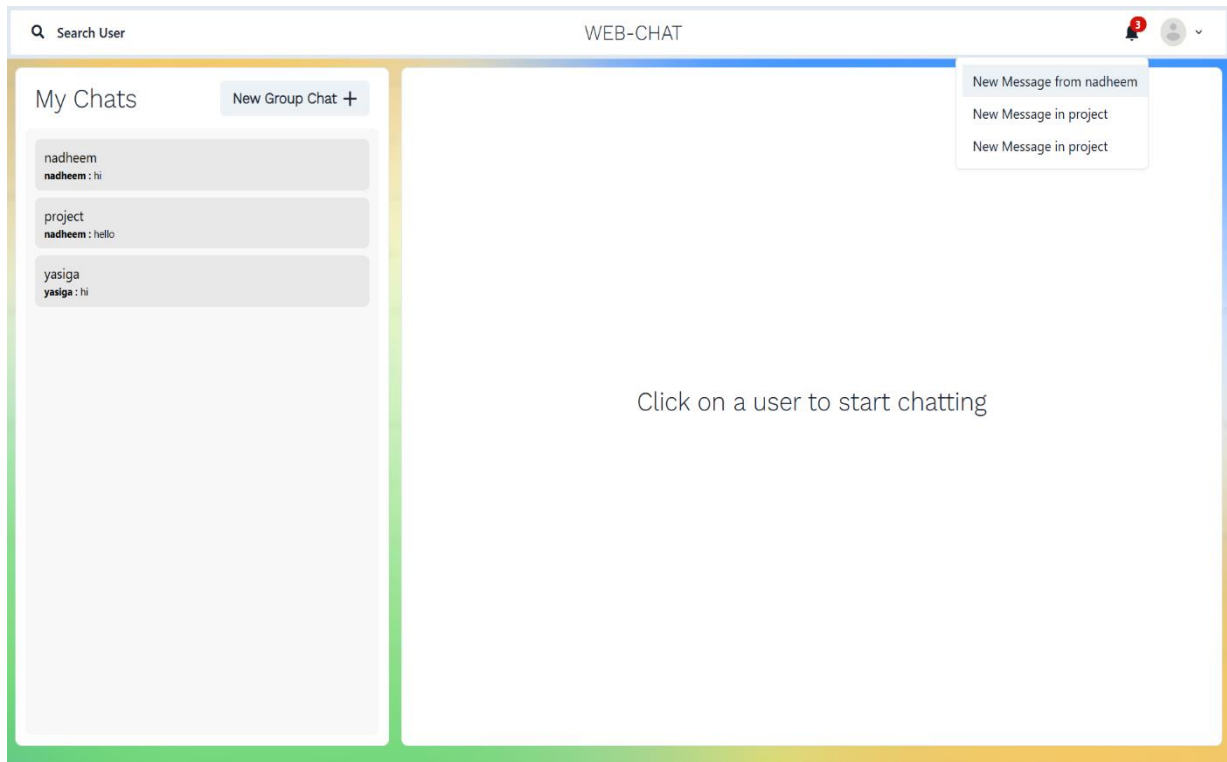


FIGURE B.7 NOTIFICATIONS

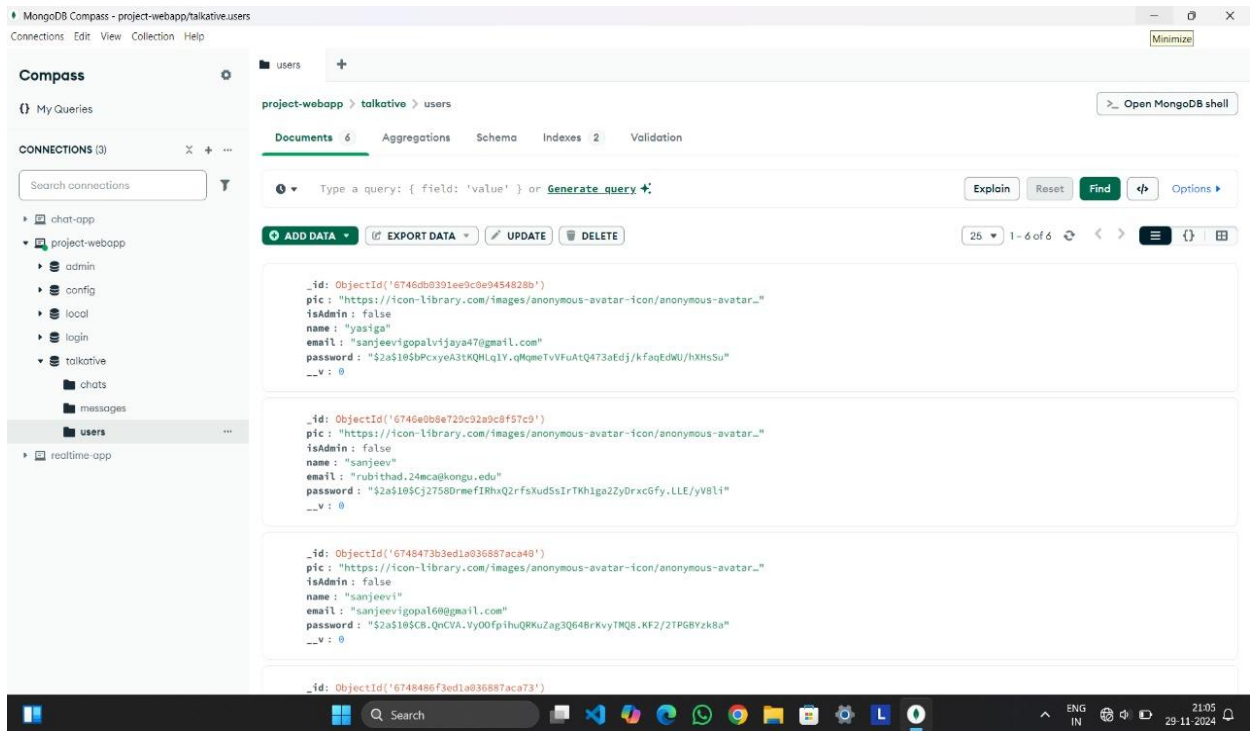


FIGURE B.8 DATA STORAGE OF USERS

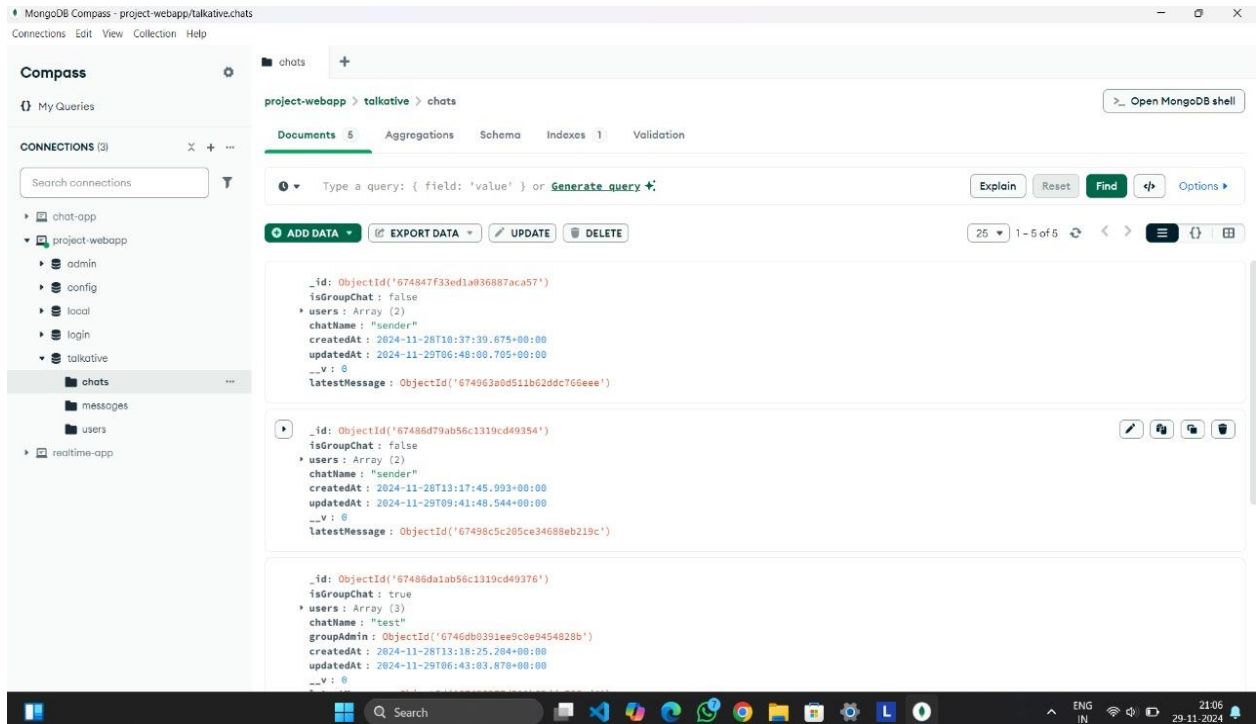


FIGURE B.9 DATA STORAGE OF CHATS

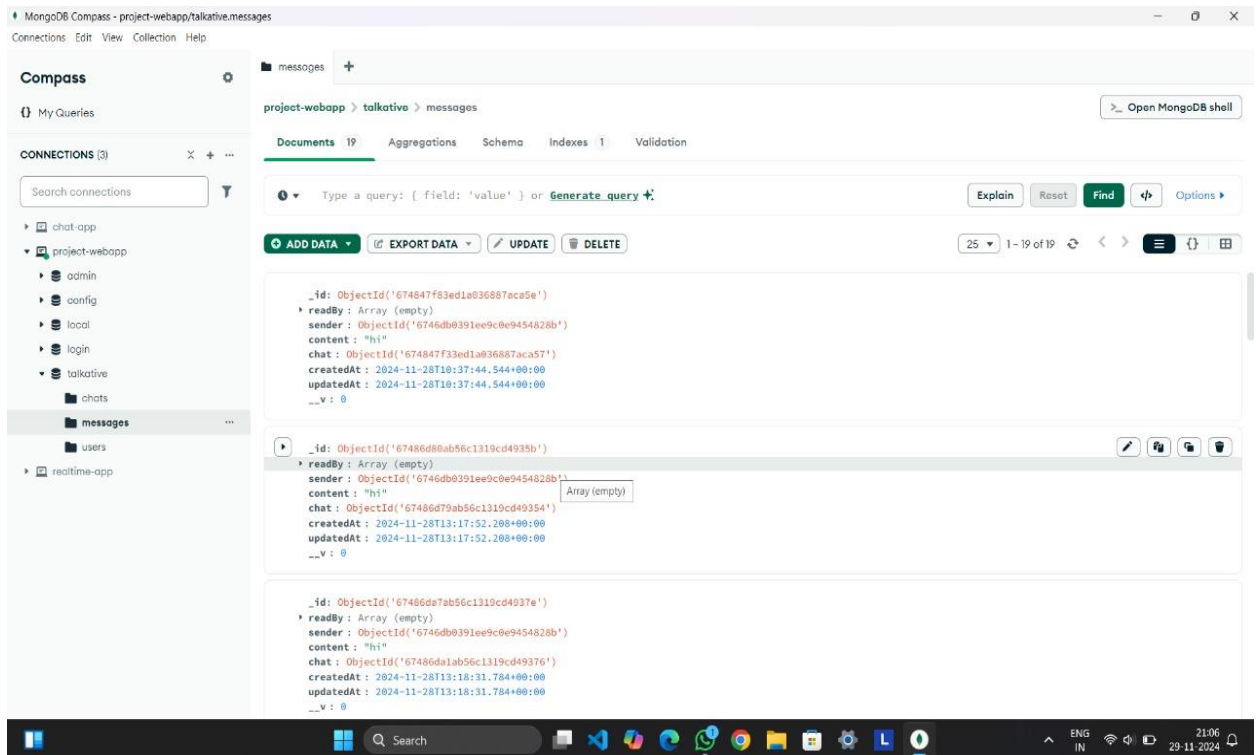


FIGURE B.10 DATA STORAGE OF MESSAGES

## REFERENCES

### WEB REFERENCES:

1. <https://www.toptal.com/front-end/how-to-build-a-real-time-chat-application-using-react-and-redux>
2. [https://developer.mozilla.org/en-US/docs/Web/API/Notifications\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API)
3. <https://socket.io/>
4. <https://reactjs.org/>

### BOOK REFERENCES:

1. “MongoDB: The Definitive Guide” by Kristina Chodorow  
Comprehensive guide covering MongoDB fundamentals, advanced queries, indexing, and more.
2. “Software Engineering: A Practitioner's Approach” by Roger S. Pressman and Bruce R. Maxim  
Focuses on practical software development techniques, from requirements to maintenance.