Project Report

on

Real Time Chat Application

Submitted in partial fulfilment of the requirement for the Course

Exploratory Project (22CS401)

of B.E. (CSE)

Batch 2022-2026

Under the Guidance of:

Dr. Swati Goel

Submitted By:

Archi – 2210990150

Rittika - 2210990732

Ravneet Kaur - 2210990720

Department of Computer Science and Engineering

Chitkara University Institute of Engineering & Technology,

Chitkara University, Punjab

CERTIFICATE

This is to be certified that the project entitled "CHAT APP" has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Punjab during the academic semester July 2025- Dec 2025 is a bonafide piece of project work carried out by "Archi (2210990150), Ravneet Kaur (2210990720), Rittika (2210990732)" towards the partial fulfilment for the award of the course Integrated Project (22CS026) under the guidance of "**Dr. Swati Goel**" and supervision.

Sign of Project Guide:

Dr. Swati Goel

(Assistant Professor)

CANDIDATE'S DECLARATION

We,

Archi (2210990150)

Ravneet Kaur (2210990720)

Rittika (2210990732)

members of GROUP - 6, B.E.-2022 of the Chitkara University, Punjab hereby declare that the Integrated Project Report entitled "CHAT APP" is an original work and data provided in the study is authentic to the best of our knowledge. This report has not been submitted to any other Institute for the award of any other course.

Archi	Rittika	Ravneet Kaur
2210990150	2210990732	2210990720

Place: Chitkara University

Date: 18, December 2025

ACKNOWLEDGEMENT

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behaviour and acts during the course of study. We express our sincere gratitude to all for providing me an opportunity to undergo Integrated Project as the part of the curriculum. We are thankful to "**Dr. Swati Goel**" for his support, cooperation, and motivation provided to us during the training for constant inspiration, presence and blessings.

Thanking you

Archi	Rittika	Ravneet Kaur
2210990150	2210990732	2210990720

Table of Contents

1. ABSTRACT

A full-stack web application, the chat application created for this project offers users a safe, instantaneous communication platform. The program provides a stable and engaging user experience with a frontend built using React.js and a backend built with a combination of Node.js, Express.js, MongoDB, and socket.io. User credentials are securely saved thanks to the system's secure authentication mechanism, which uses the bcrypt library for password hashing.

After completing the authentication process, users can log in and take part in live chats where they can send real-time text messages and photographs. Because socket.io enables real-time communication, messages may be sent quickly without requiring page refreshes. User information and conversation history are stored in MongoDB, which guarantees persistence between sessions.

Users can customize their experience by adding and changing profile images, which are updated constantly within the program, in addition to communicating. The backend guarantees scalability and security, while the frontend, constructed with React.js, offers an intuitive interface that permits smooth interaction.

This project's main goal is to create a contemporary chat application that takes into account crucial elements of data security and real-time communication. Future iterations of this platform might include more sophisticated services like file sharing, video calling, and improved encryption.

2. INTRODUCTION

2.1 Background

In today's digital age, communication is increasingly shifting towards instant messaging platforms that allow real-time, online interaction. These platforms are essential for personal communication, business collaboration, and social networking. The demand for secure and efficient chat systems has grown rapidly, especially with the rise of remote work and virtual teams. Traditional messaging systems have evolved significantly, with real-time communication becoming the standard expectation for users. This evolution has been accompanied by a need for more robust security features, such as user authentication, message encryption, and protection of personal data, given the growing concerns about data privacy. However, building such a system involves addressing several technical challenges, such as ensuring seamless real-time communication, managing user authentication securely, and maintaining a scalable architecture to accommodate multiple users simultaneously.

2.2 Problem Statement

The increasing reliance on digital communication tools calls for a chat application that not only enables real-time messaging but also ensures secure user authentication, media sharing, and an intuitive user interface. Despite the availability of various messaging platforms, there is still a need for customizable, lightweight solutions that offer secure and real-time interactions while maintaining user privacy.

Real-time Communication: Ensuring that messages and media are transmitted instantly between users without delays or disruptions.

Secure Authentication: Implementing a robust user authentication system that safeguards passwords and user credentials using secure hashing techniques like bcrypt.

3. S/W & H/W REQUIREMENTS

3.1 S/W Requirements

1. Operating System:

- Windows 10 or higher, macOS, or Linux.

2. Frontend:

- React.js: A JavaScript library for building user interfaces.
- HTML/CSS: For structuring and styling the user interface.
- JavaScript (ES6+): The programming language for handling client-side logic.

3. Backend:

- Node.js: A JavaScript runtime environment that enables server-side scripting.
- Express.js: A web framework for building REST APIs and handling server logic.
- MongoDB: A NoSQL database for storing user data, chat history, and media.
- Socket.io: A library for enabling real-time, bi-directional communication between web clients and servers.

4. Authentication:

- JWT (JSON Web Tokens): For token-based user authentication.
- bcrypt: For password hashing and validation.

5. Development Tools:

- Visual Studio Code (VS Code): A source code editor with support for various extensions for JavaScript, React, Node.js, etc.
- Thunder Client: For testing API endpoints.

6. Version Control & Collaboration:

- Git: For version control and managing source code.
- GitHub: For collaboration and repository hosting.

7. Package Managers:

- npm (Node Package Manager): To manage project dependencies and install required libraries.
- MongoDB Compass: For database management and query visualization.

3.2 H/W Requirements

- Processor: Intel Core i3 or higher, or equivalent AMD processor.
- RAM: 4 GB minimum (8 GB recommended for smooth performance).
- Hard Disk: At least 20 GB of free storage space for project files, Node.js, MongoDB, and development tools.
- Network: A high-speed internet connection for downloading libraries, real-time communication, and collaboration.

4. Code and GUI Snippets

1 LOGIN PAGE –

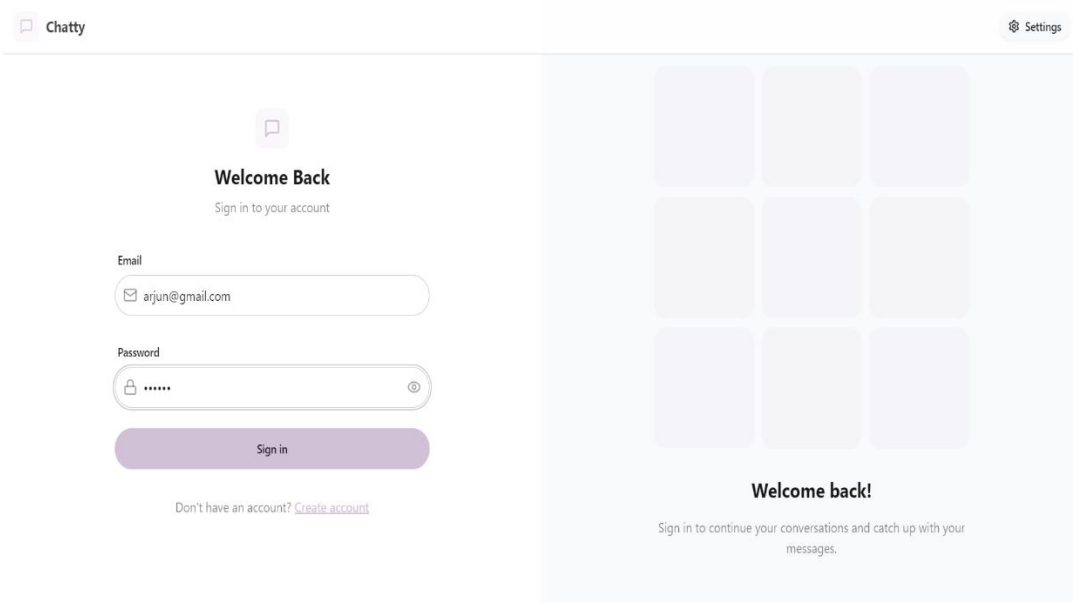

Fig 4.1

2 SIGNUP PAGE –

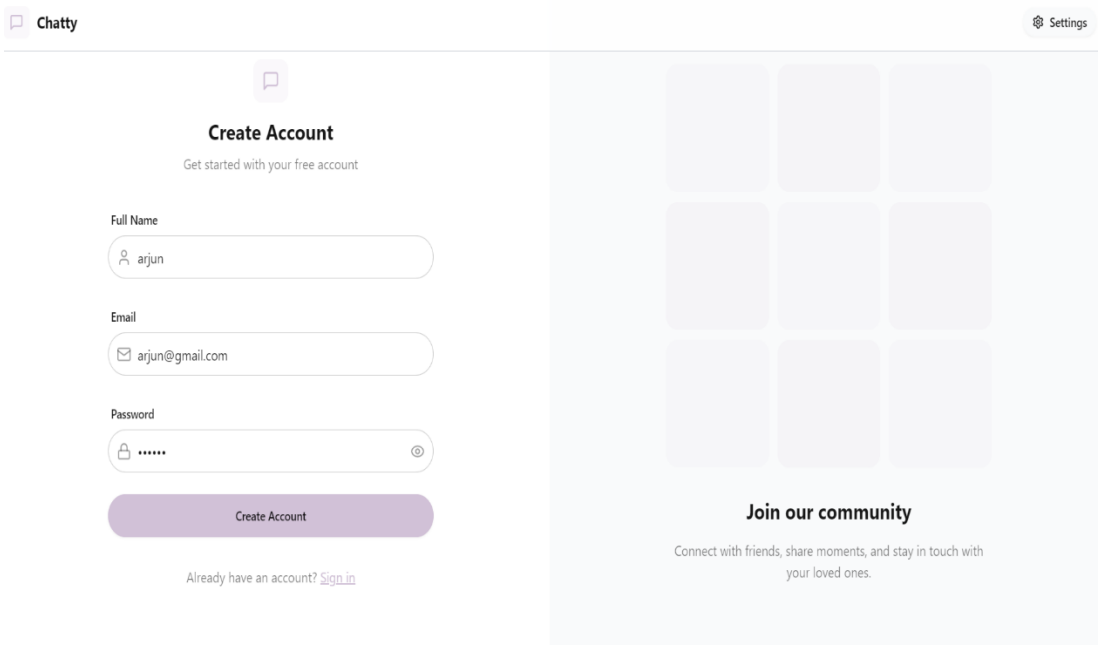

Fig 4.2

3 SETTINGS PAGE –

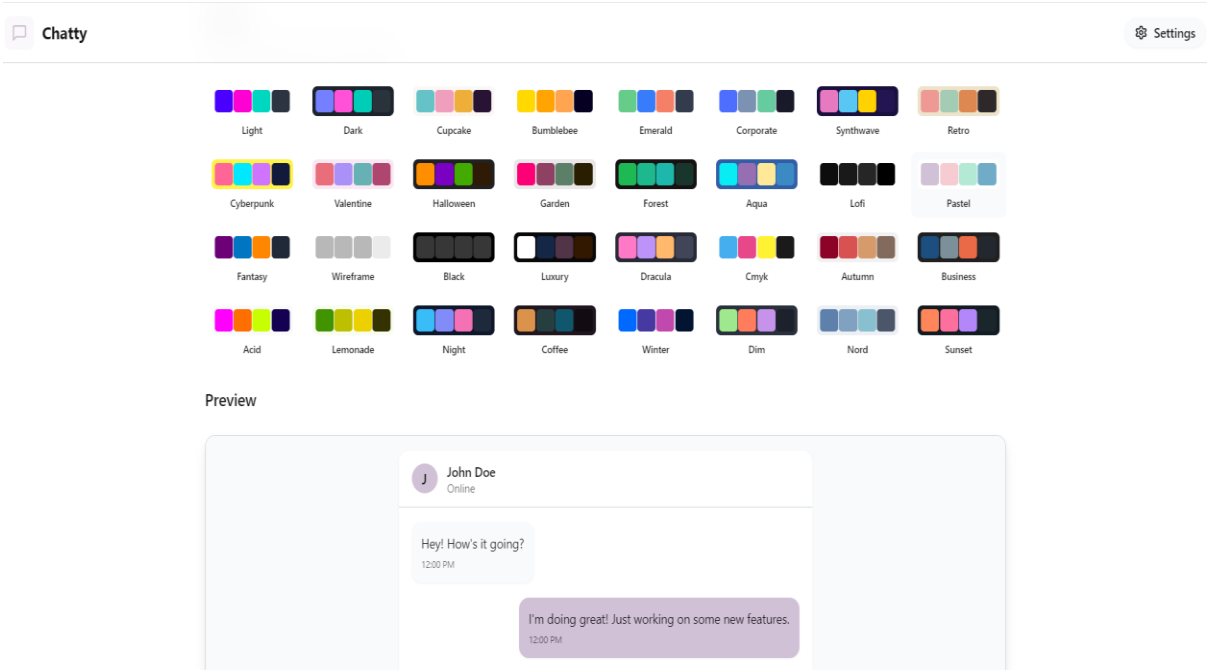

Fig 4.3

4 PROFILE PAGE –

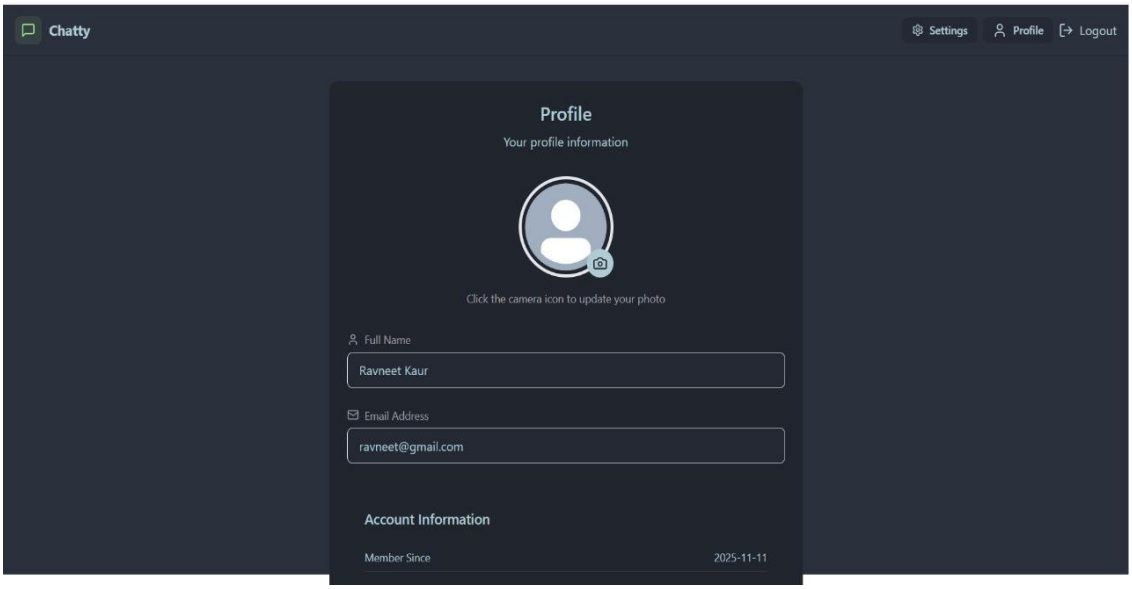

Fig 4.4

5 Home Page (Showing only online users) -

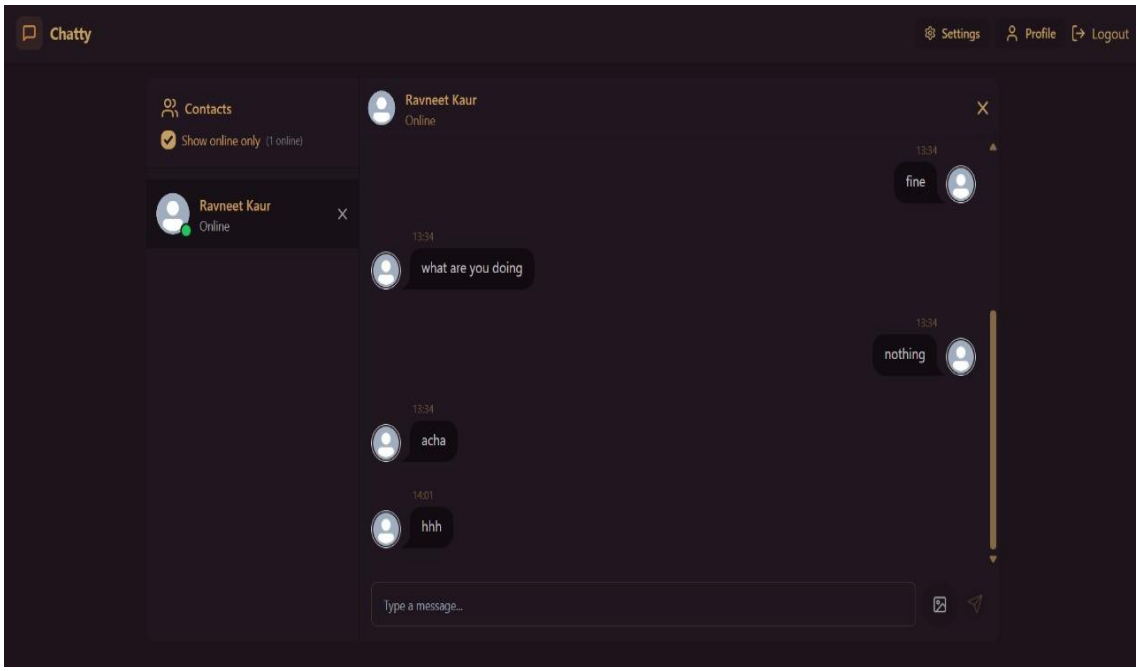

Fig 4.5

6 LIMITATIONS

- Offline Functionality: The application requires a stable internet connection, with no support for offline messaging.
- File Size Limit: There may be limitations on the size of images or files that can be shared within the chat.
- Scalability: While the system handles a small number of users efficiently, additional optimizations are needed for large-scale usage.
- Scalability: The current application is designed to support a limited number of users and real-time connections. Without significant optimization and infrastructure improvements (e.g., load balancing, distributed databases), the system may struggle to support a large number of simultaneous users or handle a large volume of messages effectively.
- No Group Chat Functionality: The application currently supports one-to-one messaging only. There is no functionality to create group chats, which is a common feature in most modern chat applications.

7 CONCLUSION

This chat application successfully demonstrates the use of modern web technologies to enable real-time communication. With features like secure login/logout, text and image messaging, and profile management, the application serves as a functional prototype of a scalable messaging platform. The implementation of bcrypt ensures secure user authentication, while socket.io provides smooth real-time communication.

Despite its current functionality, there are areas for future enhancement, such as improving scalability, adding advanced security measures like end-to-end encryption, and extending the feature set to support file sharing and group chats. The project provides a solid foundation for a scalable chat platform, with room to expand into areas such as mobile application development, multimedia communication (voice and video), and push notifications.

8 FUTURE SCOPE

- Video and Voice Calls: Integrating video and voice call functionality to enhance user engagement.
- Message Encryption: Implementing end-to-end encryption to further secure user data.
- Push Notifications: Adding push notifications to alert users of new messages even when the app is in the background.
- File Sharing: Enhancing file-sharing capabilities to support more formats and larger files.
- Mobile Application: Expanding the project into a mobile-friendly version using React Native.

9 REFERENCES

- Node.js Documentation: https://nodejs.org/en/docs/
- MongoDB Documentation: https://docs.mongodb.com/
- React.js Documentation: https://reactjs.org/docs/getting-started.html
- bcrypt Documentation: https://www.npmjs.com/package/bcrypt
- socket.io Documentation: https://socket.io/docs/
- Brad Traversy, "Modern JavaScript and Node.js Mastery: A Full-stack Developer Guide", Traversy Media, 2021.
- Steve Jobs, "Programming in the Era of Real-time Web Applications", WWDC Conference, 2010.