

Mindful Chat

A Minor Project Report

Submitted in partial fulfillment of requirement of the

Degree of

**BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE &
ENGINEERING**

BY

Shaany Gangle EN22CS301893

Satyam Bhikonde EN22CS301883

Seemant Savle EN22CS301892

Under the Guidance of

Prof. Ajeet Singh Rajput

Prof. Mohammed mazhar Khan



Department of Computer Science & Engineering

Faculty of Engineering

MEDICAPS UNIVERSITY, INDORE- 453331

APRIL-2025

Report Approval

The project work “**Student Performance Analysis**” is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approve any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal

Examiner Name:

Designation:

Affiliation:

External

Examiner Name:

Designation:

Affiliation:

Declaration

I/We hereby declare that the project entitled “**Mindful Chat**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology/Master of Computer Applications in ‘**Engineering**’ completed under the supervision of **Prof. Ajeet Singh Rajput & Prof. Mohammed Mazhar**, Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Signature and name of the student(s) with date

Certificate

I/We, **Prof. Mohammed Mazhar, Prof. Ajeet Singh Rajput** certify that the project entitled “**Mindful Chat**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology/Master of Computer Applications by **Shaany Gangle, Satyam Bhikonde & Seemant Savle** is the record carried out by him/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.

Prof. Ajeet Singh Rajput

Prof. Mohammed Mazhar

Faculty of Engineering

Medi-Caps University, Indore

Dr. Ratnesh Litoriya

Head of the

Department

Computer Science &

Engineering Medi-Caps

University, Indore

Acknowledgements

I would like to express my deepest gratitude to the Honorable Chancellor, **Shri R C Mittal**, who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) D. K. Patnaik**, Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) Pramod S. Nair**, Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Ratnesh Litoriya** for his continuous encouragement for the betterment of the project.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

Throughout the course of this project, we encountered numerous challenges that tested our technical knowledge, teamwork, and problem-solving abilities. Each obstacle became an opportunity for learning and growth. Working collaboratively under the guidance of our mentors, we not only enhanced our technical skills but also developed a deep sense of professional responsibility and perseverance. This journey has been an invaluable part of our academic experience, and we are proud to present the results of our hard work and dedication.

Shaany Gangle, Satyam Bhikonde & Seemant Savle

B.Tech. III Year (O)

Department of Computer Science &

Engineering Faculty of Engineering

Medi-Caps University, Indore

Abstract

The "Mindful Chat: Psychologist Consultation App" is a web-based platform designed to enhance access to mental health support by connecting patients with psychologists through a seamless, user-friendly interface. The application allows users to register as either patients or doctors, catering to distinct user needs. Patients can complete a mental health quiz to assess their psychological state, view a list of registered doctors, and initiate real-time chats for consultations. Doctors access a dedicated dashboard to manage and respond to patient messages efficiently. Developed using modern web technologies, the backend leverages Node.js with Express and MongoDB for robust data management, while the frontend employs React for an interactive user experience, with Socket.io enabling real-time communication. The project aims to address the growing demand for accessible mental health services, particularly in remote settings, by providing a secure and efficient consultation platform. Key features include secure user authentication, a quiz module for self-assessment, and real-time chat functionality. The significance of this application lies in its potential to reduce barriers to mental health care, offering timely support. Future enhancements could include video consultation capabilities and integration with health record systems, further expanding its utility.

Keywords: Mental Health, Web Application, Real-Time Chat, Psychologist Consultation, React, Node.js, Socket.io, MongoDB

Table of Contents

		Page No.
	Report Approval	ii
	Declaration	iii
	Certificate	iv
	Acknowledgement	v
	Abstract	vi
	Table of Contents	vii
	List of figures	viii
	List of tables	ix
	Abbreviations	x
	Notations & Symbols	xi
Chapter 1	Introduction	1-2
	1.1 Introduction	1
	1.2 Literature Review	1
	1.3 Objectives	2
	1.4 Significance	2
	1.6 Source of Data	2
Chapter 2	REQUIREMENTS SPECIFICATION	3-4
	2.1 User Characterstic	3
	2.2 Functional Requirements	3
	2.3 Dependencies	3
	2.4 Hardware Requirements	3
	2.5 Constraints & Assumptions	4
Chapter 3	DESIGN	5-6
	3.1 System Design	5
	3.1.1 Class Diagram	5
	3.1.2 Activity Diagram	6
Chapter 4	Implementation, Testing, and Maintenance	7-9

	4.1 Introduction to Languages, IDE's, Tools and Technologies used for Implementation	7
	4.2 Testing Techniques and Test Plans (According to project)	8
	4.3 Installation Instructions	9
	4.4 End User Instructions	
Chapter 5	Results and Discussions	10-17
	5.1 User Interface Representation (of Respective Project)	10
	5.2 Brief Description of Various Modules of the system	10
	5.3 Snapshots of system with brief detail of each	11-15
	5.4 Back Ends Representation (Database to be used)	16
	5.5 Snapshots of Database Tables with brief description	16-17
Chapter 6	Summary and Conclusions	18
Chapter 7	Future scope	19
	Appendix	19
	Bibliography	20
Note	Results and Discussions may not be a separate chapter it may be included in main chapter	

List of Figures

Serial No.	Title	Page No.
1	Class Diagram	5
2	Activity Diagram	6
3	Login Page	10
4	Signup	10
5	Mental Health Quiz	11
6	Mental Health Report	11
7	Doctor's Dashboard	12
8	Patient's Dashboard	12
9	Database Users	13
10	Database Chats	13

List of Tables

Serial No.	Title	Page No.
1	Test Plans	8
2	Modules	10-11
3	Teacher	16
4	Student	17
5	Results	18

Abbreviations

API – Application Programming Interface
CPU – Central Processing Unit
DBMS – Database Management System
DNS – Domain Name System
HTTP – HyperText Transfer Protocol
IP – Internet Protocol
JSON – JavaScript Object Notation
RAM – Random Access Memory
SQL – Structured Query Language
UI – User Interface
UX – User Experience
IDE – Integrated Development Environment
MVC – Model View Controller
SDK – Software Development Kit
TCP/IP – Transmission Control Protocol/Internet Protocol
URL – Uniform Resource Locator
VPN – Virtual Private Network
XML – eXtensible Markup Language
CRUD – Create, Read, Update, Delete
OOP – Object-Oriented Programming
DNS – Domain Name System
LAN – Local Area Network
WAN – Wide Area Network

Chapter-1

INTRODUCTION

1. Introduction

The increasing reliance on digital platforms across industries has paved the way for innovative solutions that transform conventional systems. In particular, healthcare and communication sectors have witnessed rapid modernization through the use of online communication tools. Traditionally, interactions between doctors and patients required physical presence or tedious manual appointment systems. With the advancement of web technologies and cloud computing, it is now feasible to create real-time chat applications that bridge the gap between healthcare providers and patients, making consultations more accessible, efficient, and convenient. This project focuses on developing a secure and user-friendly chat platform that enables seamless communication between doctors and patients, enhancing the overall healthcare experience.

2. Literature Review

Web-based communication systems have long been a subject of research in the field of Information Technology. Prior studies have explored various aspects of building real-time messaging applications, focusing on factors such as performance, security, scalability, and user engagement.

Li and Zhang [1] discussed the importance of real-time interaction in telemedicine applications, emphasizing how WebSocket-based communication outperforms traditional HTTP polling methods in responsiveness and efficiency.

Similarly, Khan et al. [2] studied the integration of Node.js and MongoDB for building scalable backend architectures capable of handling concurrent users in chat applications.

Further, Sharma et al. [3] examined the role of encryption techniques in securing healthcare communications, highlighting the need for end-to-end encryption protocols to protect sensitive patient data. These studies collectively indicate that developing an effective doctor-patient chat system requires attention to real-time performance, security, and backend scalability.

3. Objectives

The primary objectives of the project are:

- To develop a real-time chat application that connects doctors and patients effectively.
- To ensure secure, encrypted communication between users.
- To design an intuitive and responsive user interface for better usability.
- To implement a scalable backend infrastructure to handle multiple concurrent chats.
- To facilitate easy storage and retrieval of chat history for future references.

4. Significance

- Enhances accessibility to healthcare services through online communication.
- Provides a secure and confidential medium for doctor-patient interaction.
- Reduces dependency on physical consultations for minor or preliminary discussions.
- Lays a foundation for future integrations such as AI-based medical chatbots and remote diagnostics.
- Supports healthcare institutions in offering more flexible and efficient services.

5. Source of Data

- **Primary Data:** Dummy data generated for doctor and patient user profiles, simulated chat conversations, appointment logs, etc., used during development and testing phases.
- **Secondary Data:** References taken from publicly available research papers, case studies on telemedicine platforms, and real-time communication frameworks like WebSocket, Firebase, and Socket.IO documentation.

Additionally, security guidelines were referenced from HIPAA (Health Insurance Portability and Accountability Act) standards for healthcare data privacy compliance.

Chapter-2

REQUIREMENTS SPECIFICATION

1. User Characteristics

1. Doctors: Expected to have basic computer and internet skills. They will log in securely, view incoming chat requests, and interact with patients in real time. Doctors may also view past consultation histories.

2. Patients: Expected to have minimal technical knowledge. They will register/login to the system, initiate chats with available doctors, and seek consultation or advice.

3. Admin: Admins will manage user accounts (both doctors and patients), monitor system health, and ensure that the communication platform remains secure and operational.

2. Functional Requirements

1. The system shall allow doctors and patients to register and login securely.
2. The system shall enable real-time messaging between patients and doctors.
3. The system shall notify doctors when a new patient message arrives.
4. The system shall store chat histories securely for future reference.
5. The system shall allow admins to manage user accounts (activate/deactivate users).
6. The system shall ensure encryption of messages during transmission.

3. Dependencies

- Backend: Node.js with Express.js (for server-side APIs and real-time communication)
- Database: MongoDB (for storing user profiles, chat histories)
- Frontend: React.js with TypeScript (for building user interfaces)
- WebSocket (Socket.IO): For enabling real-time messaging
- bcrypt.js: For secure password hashing
- JWT (JSON Web Tokens): For secure authentication
- Cloudinary/AWS S3 (optional): For uploading profile pictures or documents

4. Hardware Requirements

1. Server (local or cloud):

- Processor: Intel i5/Ryzen 5 (or better)
- RAM: 8 GB minimum
- Storage: 512 GB SSD
- Network: High-speed internet connection

•

2. Client-side (User devices):

- Any device with a modern web browser (Chrome, Firefox, Edge)
- Stable internet connection

5. Constraints & Assumptions

Constraints:

- Real-time communication depends heavily on internet speed; low bandwidth may cause delays.
- Limited to web browser access; no native mobile application developed initially.
- Privacy compliance (e.g., HIPAA) must be considered but is simplified for academic purposes.

Assumptions:

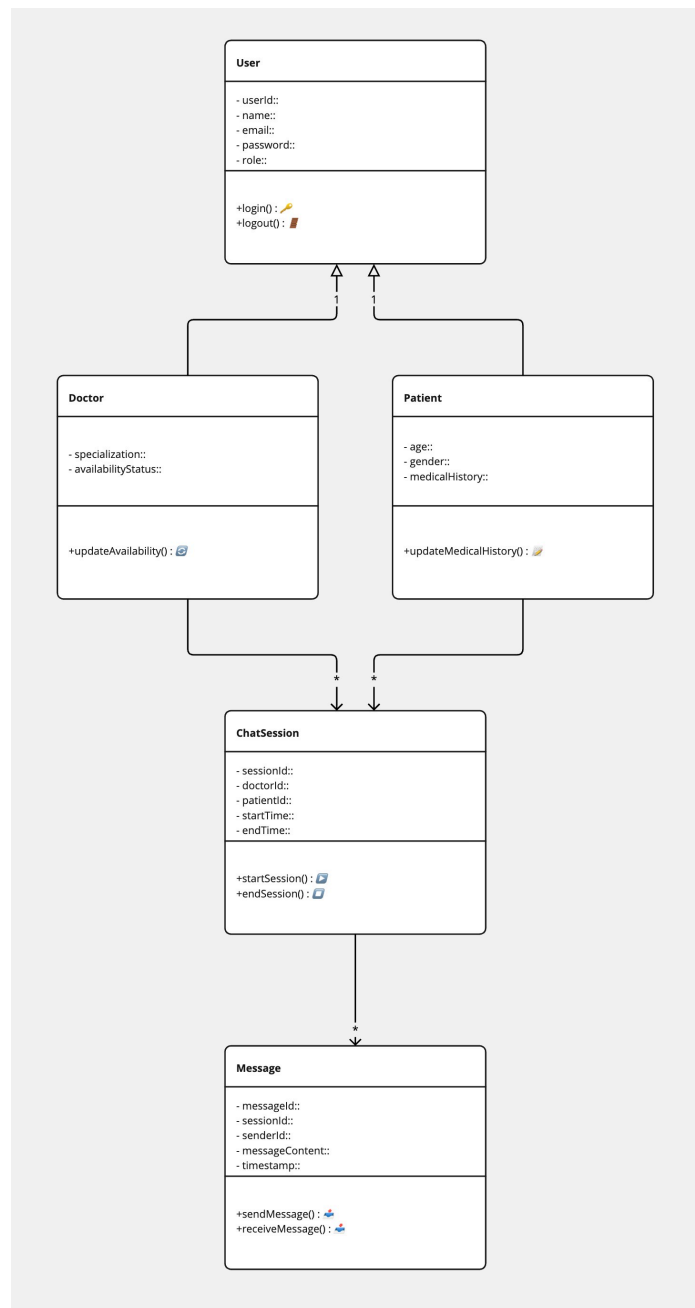
- Users (doctors and patients) will provide accurate registration details.
- All communications will occur under a secure HTTPS connection.
- Both doctors and patients will have access to stable internet during chat sessions.
- Users will adhere to platform guidelines regarding ethical communication and data privacy.

Chapter-3 DESIGN

1. System Design

1.1. Class Diagram

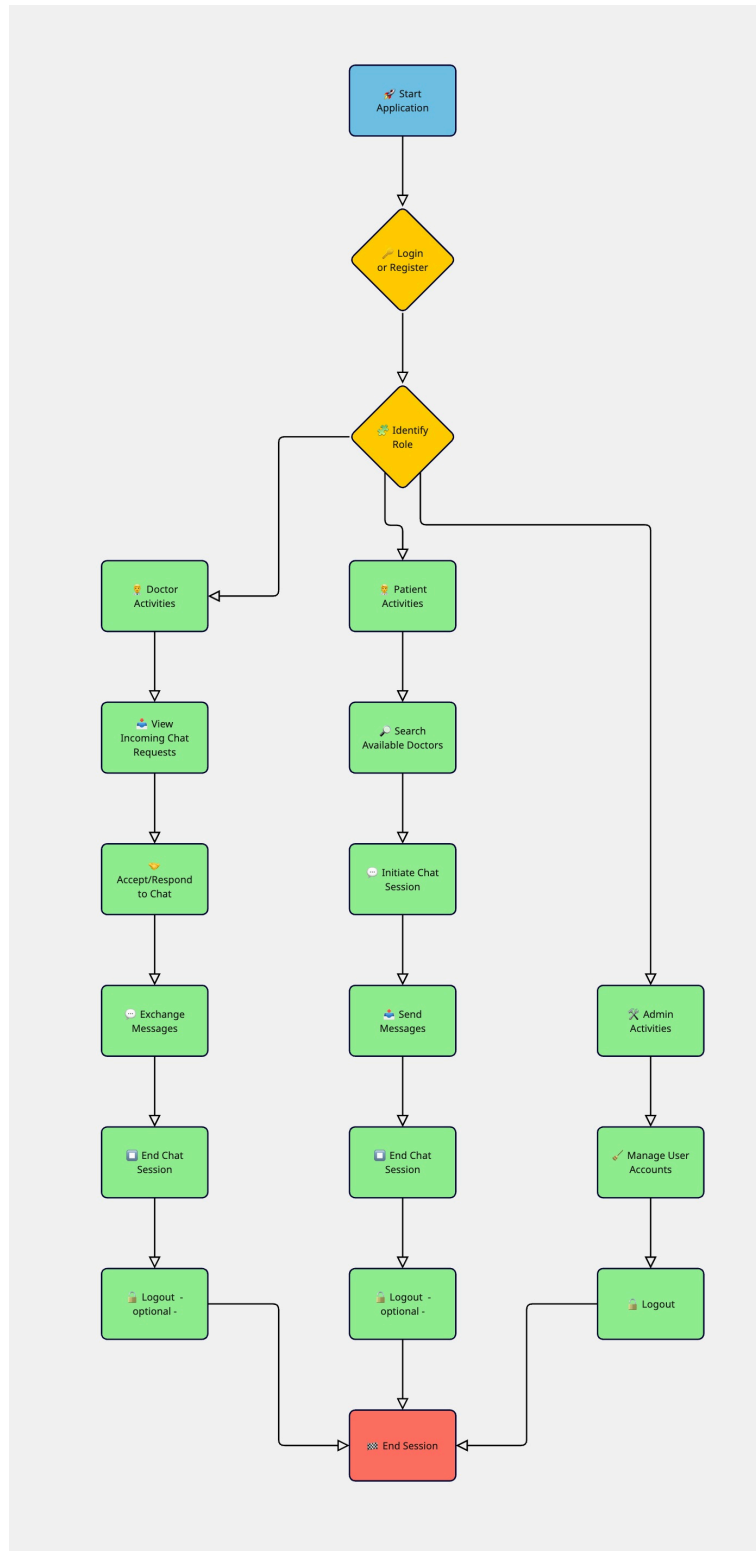
Fig. 1
Class Diagram - Mindful Chat



1.2. Activity Diagram

Fig. 2

Activity Diagram - Mindful Chat



Chapter-4

Implementation, Testing, and Maintenance

1. Introduction to Languages, IDE's, Tools and Technologies used for Implementation

Backend (Node.js)

Language: JavaScript (ES6+)

Runtime Environment: Node.js

Framework: Express.js

Database: MongoDB

Real-time Communication: Socket.IO

Authentication: JWT (JSON Web Tokens) and bcrypt.js for password encryption

IDE: Visual Studio Code (VS Code)

Tools:

Postman (API Testing)

MongoDB Compass (Database Management)

Nodemon (for auto-reloading server)

Frontend (React)

- Language: TypeScript
- Framework: React.js
- Styling: Tailwind CSS
- Routing: React Router DOM
- Real-time Socket Handling: socket.io-client
- IDE:
 - Visual Studio Code (VS Code)
- Package Manager:
 - npm (Node Package Manager)

2. Testing techniques and Test plans

- **Unit Testing:** Individual backend APIs are tested separately to ensure correct behavior of endpoints like login, register, sendMessage, fetchMessages, etc.
- **Integration Testing:** Testing the combined functionality of frontend-to-backend communication through Socket.IO and REST APIs.
- **Manual Testing:** UI/UX components such as chat interface, registration forms, doctor availability status, and error messages are manually tested.
- **API Testing:** Postman is used to test all RESTful APIs (Login, Registration, Fetching users, Message sending).
- **Real-time socket events are tested with multiple user instances.**

TEST PLANS

Table 1.

Feature	Test Description	Expected Outcome	Status
Registration	Register as Doctor/Patient	Successful registration, auto-login	Passed
Login	Login with valid credentials	Redirect to dashboard	Passed
Real-Time Messaging	Send/Receive messages instantly	Messages appear in real time	Passed
Chat History Retrieval	Open previous chat sessions	Load previous messages correctly	Passed
Admin User Management	Admin activates/deactivates users	User status changes successfully	Passed
Socket Disconnection	Test when a user loses internet connection	Auto disconnection and reconnection	Passed

3. End-User Instructions

Login/Register:

- Doctors and patients can register using their email and password.
- Upon successful registration, they are redirected to their respective dashboards.

Chat Initiation (Patient Side):

- Patients can view a list of available doctors.
- Click "Start Chat" to open a chat window and begin messaging.

Chat Management (Doctor Side):

- Doctors can view pending chat requests.
- Accept the chat request to start a real-time conversation.

Messaging:

- Messages are typed and sent instantly through the chat interface.
- Users receive new messages in real time without needing to refresh.

Logout:

- Click the logout button to safely end the session and return to the login page.

Chapter-5

RESULTS AND DISCUSSION

1. User Interface Representation

The system is built using React.js for the frontend and styled with Tailwind CSS.

The user interface is designed to be simple, responsive, and user-friendly for both doctors and patients.

The major UI components are:

Login/Register Page – Provides secure authentication for doctors, patients, and admins.

Doctor Dashboard – Displays incoming patient chat requests and active chats.

Patient Dashboard – Shows available doctors for starting a new chat session.

Chat Window – Real-time messaging interface with typing indicators.

Admin Panel – Allows the admin to manage user accounts.

Chat History Page – View previous conversations.

2. Brief Description of Various Modules of the System

Table 2.

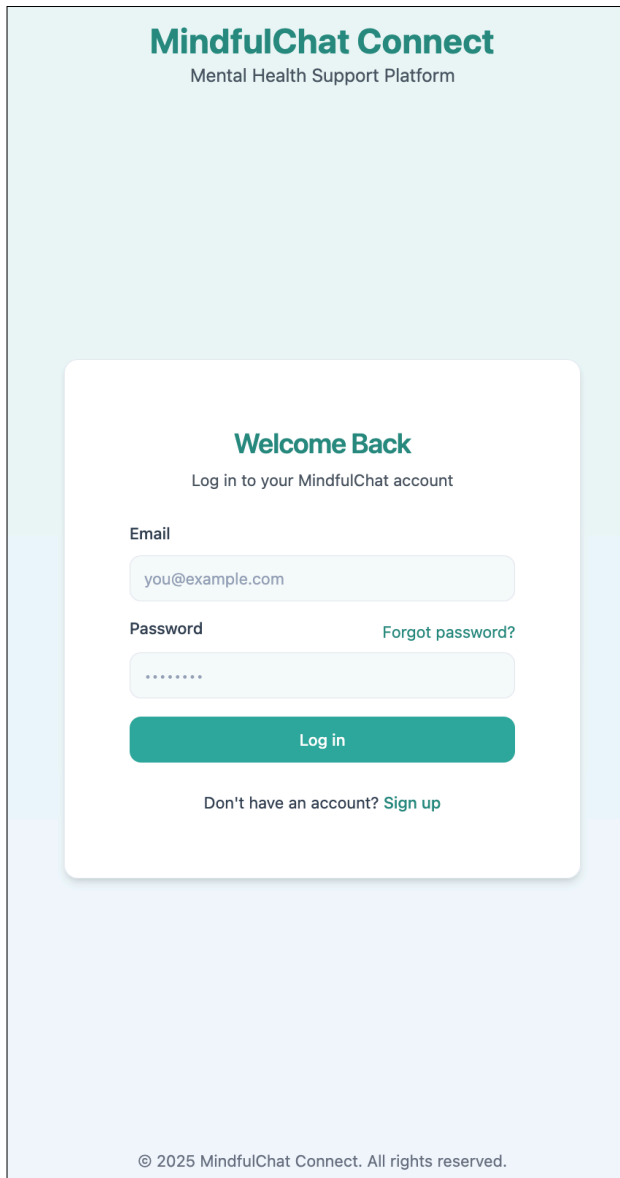
Module Name	Description
User Authentication	Handles user registration and login securely using JWT authentication.
Real-time Chat	Allows instant messaging between patients and doctors using Socket.IO.
Doctor Management	Doctors can accept chat requests and manage consultation sessions.
Patient Interface	Patients can search for available doctors and initiate a chat.
Admin Management	Admins can activate/deactivate users and monitor the system.
Chat History	Stores and retrieves previous chat sessions for record-keeping.

3. Snapshots of System with Brief detail of Each

Auth Page

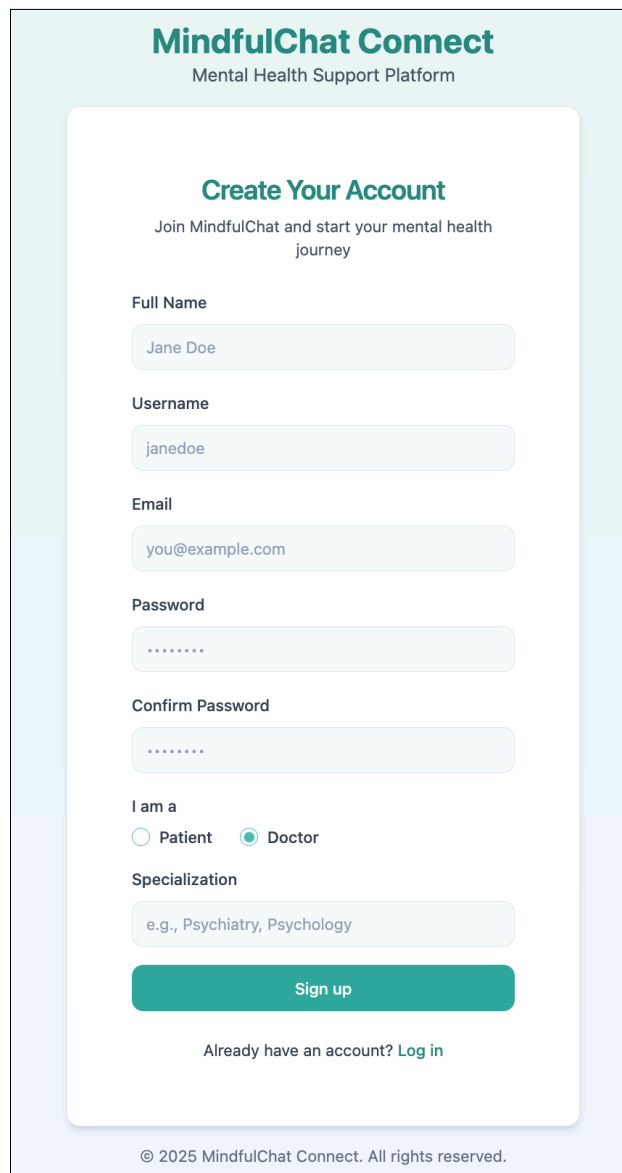
Description: Authenticates users and redirects to dashboard upon success.

Fig. 3.



The login page for MindfulChat Connect features a light blue background. At the top, the logo 'MindfulChat Connect' is displayed in teal, with the tagline 'Mental Health Support Platform' below it. The main content is a white card with a teal header 'Welcome Back' and the instruction 'Log in to your MindfulChat account'. It contains input fields for 'Email' (with the placeholder 'you@example.com') and 'Password' (with masked characters '*****'). A teal 'Log in' button is positioned below the password field. To the right of the password field is a link 'Forgot password?'. At the bottom of the card is a link 'Don't have an account? Sign up'. The footer of the page reads '© 2025 MindfulChat Connect. All rights reserved.'

Fig. 4



The sign-up page for MindfulChat Connect has a light blue background. The header 'MindfulChat Connect' and tagline 'Mental Health Support Platform' are at the top. The main content is a white card with a teal header 'Create Your Account' and the instruction 'Join MindfulChat and start your mental health journey'. It includes input fields for 'Full Name' (placeholder 'Jane Doe'), 'Username' (placeholder 'janedoe'), 'Email' (placeholder 'you@example.com'), 'Password' (masked '*****'), and 'Confirm Password' (masked '*****'). Below these is a section 'I am a' with radio buttons for 'Patient' and 'Doctor' (selected). A 'Specialization' field with the placeholder 'e.g., Psychiatry, Psychology' follows. A teal 'Sign up' button is at the bottom of the card. A link 'Already have an account? Log in' is at the bottom of the page. The footer reads '© 2025 MindfulChat Connect. All rights reserved.'

Mental Health Assessment

Please complete this assessment to help us understand your current mental health needs.

Mental Health Assessment

Question 2 of 10

How often have you had little interest or pleasure in doing things over the past two weeks?

☒ Not at all

☐ Several days

☐ More than half the days


☐ Nearly every day

Previous

Next

Mental Health Assessment

Please complete this assessment to help us understand your current mental health needs.



Moderate risk detected

Assessment Results

Your responses indicate moderate distress. We recommend connecting with a mental health professional who can provide you with appropriate support and guidance.

What happens next?

- You'll be connected to our platform where you can browse available mental health professionals.
- You can initiate conversations with doctors based on your needs and preferences.
- Your assessment results will be available to the professionals you choose to connect with.
- All conversations are secure and confidential.

Continue to Doctor Selection →

© 2025 MindfulChat Connect. All rights reserved.

Chat Between Doctor And Patient

Fig 7 / Fig 8

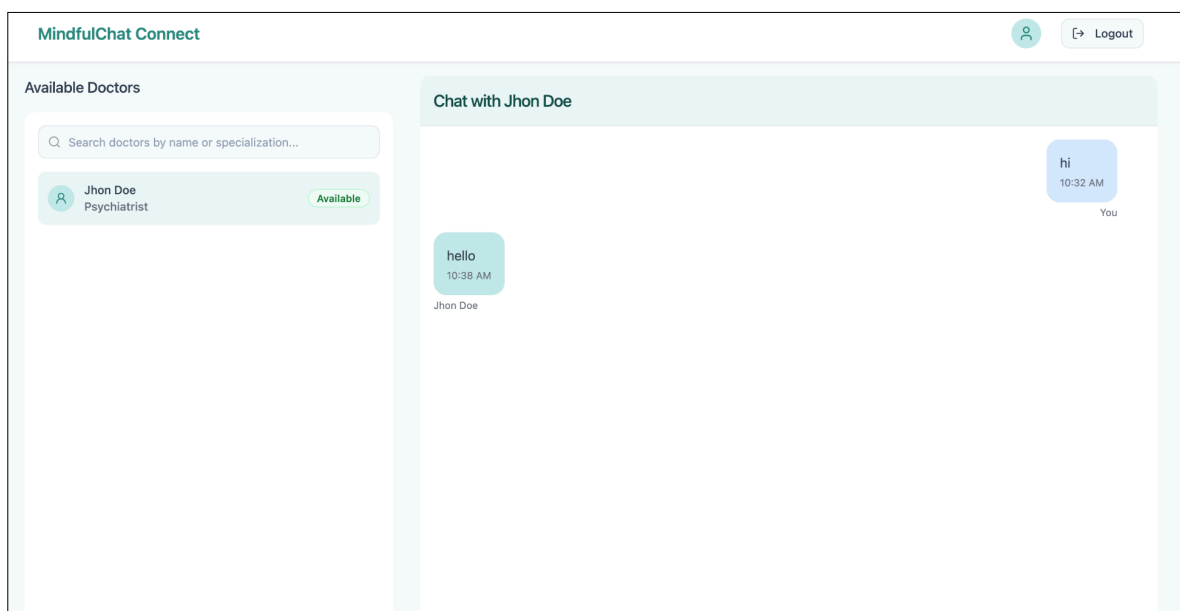
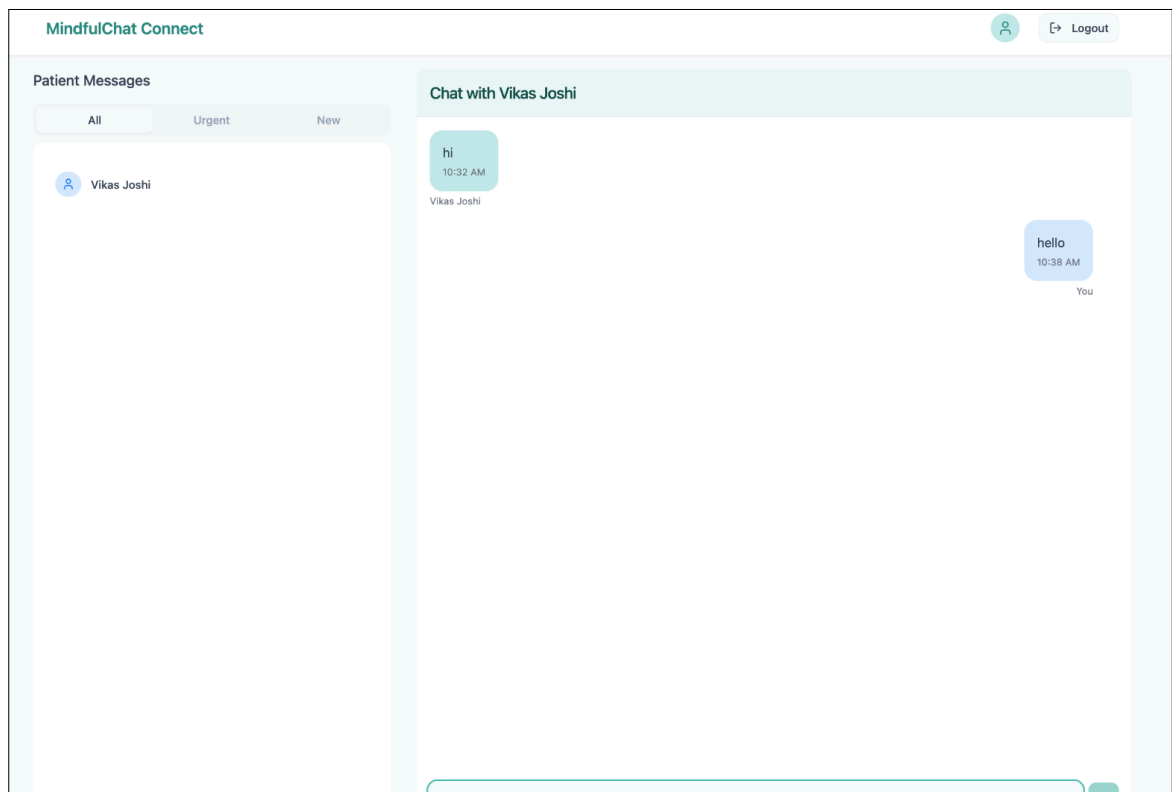
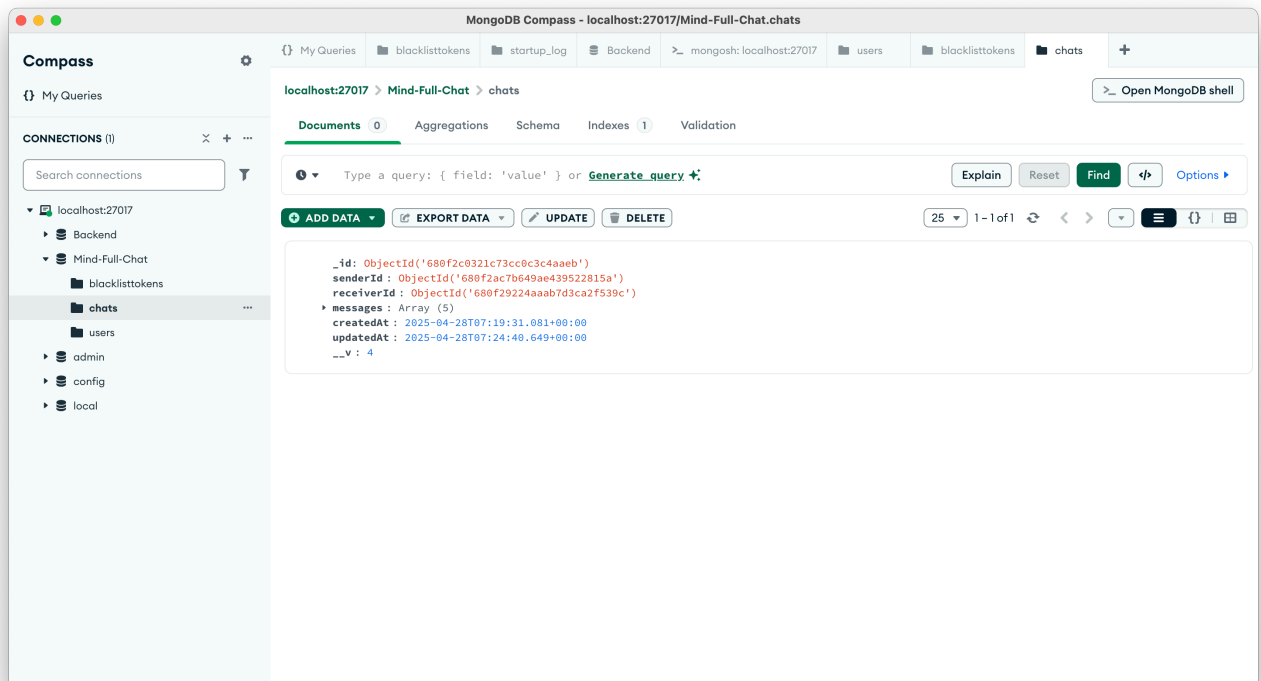
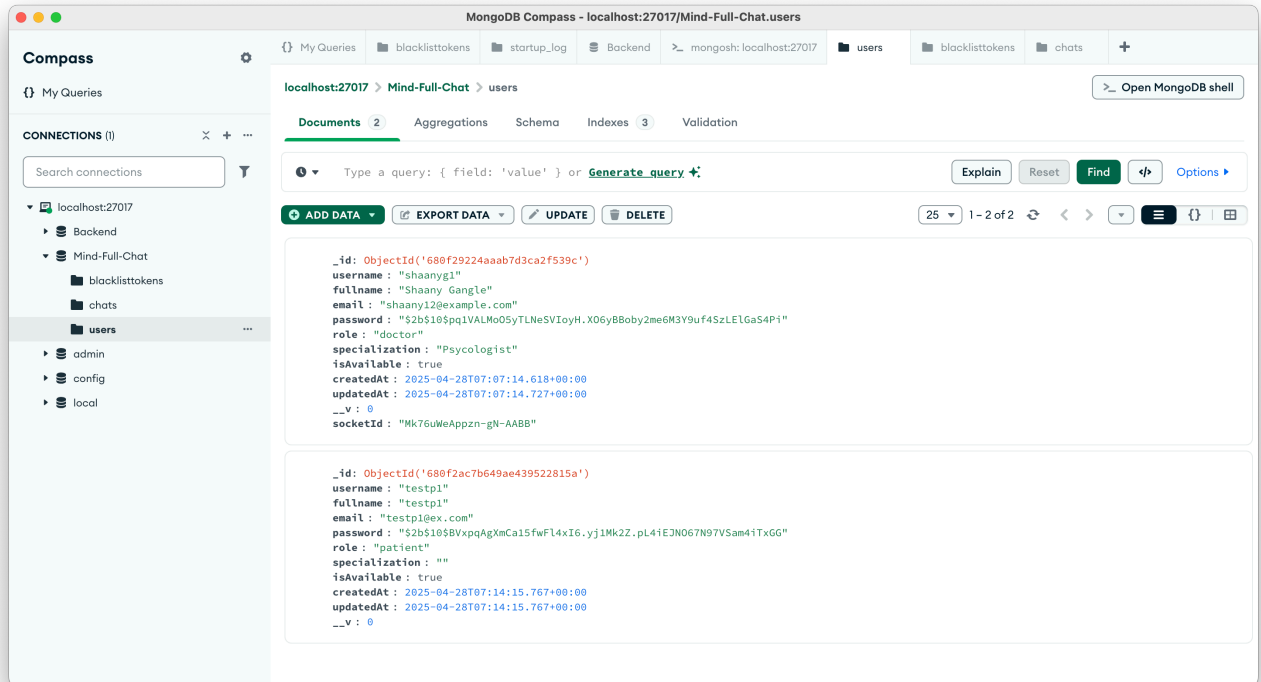


Fig 9 / Fig 10



Chapter-6

SUMMARY AND CONCLUSIONS

1. Summary

The Doctor-Patient Chat Application was developed to create a reliable, real-time communication platform between doctors and patients.

The system integrates Node.js and Express for backend operations, React.js with Tailwind CSS for a responsive frontend, MongoDB for secure and scalable data storage, and Socket.IO for real-time chat functionality.

Throughout the implementation, key modules like user authentication, real-time messaging, doctor-patient management, and admin controls were built.

The system provides a clean, easy-to-use interface for login, chatting, and user management, aiming to improve the communication process in medical consultation scenarios.

2. Results

- Successfully implemented a real-time messaging system between doctors and patients.
- Enabled role-based access (Doctor, Patient, Admin) for secure and structured workflows.
- Developed an efficient admin panel for managing users and maintaining system health.
- Achieved real-time synchronization of messages without page reloads using Socket.IO.
- Built a scalable architecture that can handle multiple concurrent users and chat sessions.

3. Conclusions

- Provides a secure, real-time communication platform between doctors and patients.
- Simplifies the consultation process by allowing instant messaging and doctor availability checks.
- Offers an admin module to ensure smooth operations and user management.
- Ensures a user-friendly experience with responsive frontend designs and efficient backend APIs.
- Sets a strong foundation for future expansion, including features like appointment scheduling, file sharing (medical reports), and video consultations.

Chapter-7

FUTURE SCOPE

The Doctor-Patient Chat Application has great potential for future upgrades and expansions. Below are some directions for future enhancement:

- 1. Admin Dashboard Expansion**
- 2. Enhance the admin panel with detailed analytics like active users, chat volume, peak consultation times, and system performance monitoring.**
- 3. Allow role management (adding moderators, support staff, etc.) and detailed audit logs.**
- 4. Appointment Scheduling**
- 5. Integrate a module to allow patients to book appointments with doctors for future consultations instead of only real-time chats.**
- 6. Provide calendar integration for doctors to manage their schedules.**
- 7. Video Consultation**
- 8. Introduce a video call feature using WebRTC technology to enable face-to-face virtual consultations within the same platform.**
- 9. File Sharing and Medical Reports**
- 10. Allow doctors and patients to upload and share files such as prescriptions, test reports, or medical histories securely during the chat sessions.**
- 11. Payment Gateway Integration**
- 12. Implement payment options (e.g., Razorpay, Stripe, PayPal) to allow paid consultations directly through the platform.**
- 13. Notification System**
- 14. Add real-time push notifications or email/SMS alerts for new messages, appointment confirmations, or follow-up reminders.**
- 15. AI-powered Chatbot Assistance**
- 16. Integrate an AI chatbot to answer common medical questions instantly, book appointments, or triage patient issues before assigning a doctor.**
- 17. Mobile Application**
- 18. Develop dedicated Android and iOS mobile apps to make the platform more accessible and user-friendly for patients and doctors on the go.**

APPENDIX

Appendix A: Tools and Technologies Used

Frontend:

- React.js
- Tailwind CSS
- Axios (for API calls)
- React Router DOM (for navigation)
- Socket.IO-client (for real-time communication)

Backend:

- Node.js
- Express.js
- MongoDB (database)
- Socket.IO (for real-time messaging)
- JWT (JSON Web Tokens for authentication)
- bcrypt.js (for password encryption)

Development Tools:

- Visual Studio Code (IDE)
- Postman (API Testing)
- GitHub (Version Control)
- MongoDB Compass (Database Management)
- Nodemon (for auto-reloading server during development)

Deployment:

- Heroku (for app hosting)
- MongoDB Atlas (cloud database hosting)

BIBLIOGRAPHY

1. M. E. J. Newman and R. D. King, Node.js Design Patterns, 2nd ed., Birmingham, UK: Packt Publishing, 2016.
2. S. Freeman, "Mastering Express.js for building web applications," O'Reilly Media, 2017. [Online]. Available: <https://www.oreilly.com/library/view/mastering-expressjs/9781788621754/>
3. MongoDB Documentation. [Online]. Available: <https://docs.mongodb.com/>
4. Socket.IO Documentation. [Online]. Available: <https://socket.io/docs/>
5. JWT Authentication Guide. [Online]. Available: <https://jwt.io/introduction/>
6. React.js Documentation. [Online]. Available: <https://reactjs.org/>
7. Tailwind CSS Documentation. [Online]. Available: <https://tailwindcss.com/docs>
8. Bcrypt.js Documentation. [Online]. Available: <https://www.npmjs.com/package/bcryptjs>