# MediChat Documentation

## Table of Contents

1. Introduction

2. Project Overview

3. Objectives

4. Scope

5. System Model

6. System Architecture

7. Use Case Diagrams

8. Entity-Relationship (ER) Diagram

9. Database Schema

10. Functional Requirements

11. Non-Functional Requirements

12. Features & Modules

13. Technology Stack

14. User Interface Design

15. Security Measures

16. API Documentation

17. Deployment Strategy

18. Testing Plan

19. Future Enhancements

20. Conclusion

## 1. Introduction

MediChat is a web-based application aimed at providing users with an online platform for health-related queries, emergency assistance, and medical support. The platform integrates various functionalities such as real-time communication, health-related information, and an emergency contact system.

## 2. Project Overview

MediChat is designed to bridge the gap between medical professionals and patients by offering an easy-to-use web application. The project incorporates interactive features to ensure accessibility and efficiency in healthcare communication.

## 3. Objectives

- Provide an intuitive and responsive platform for health-related discussions.  
- Offer emergency assistance with quick response times.  
- Ensure user privacy and data security.  
- Implement AI-driven chat support for common health queries.

## 4. Scope

- Users can access general health information and consult professionals.  
- Emergency contact feature for quick medical assistance.  
- Secure communication channels for users.  
- Future enhancements include AI-driven diagnosis suggestions and integration with healthcare providers.

## 5. System Model

The MediChat project follows the Client-Server Model where users interact with the application through a web interface, and all data processing is handled on the server.

## 6. System Architecture

MediChat is built using a three-tier architecture:  
1. Presentation Layer: Frontend developed using HTML, CSS, JavaScript, and Bootstrap.  
2. Business Logic Layer: Backend implemented with Node.js and Express.  
3. Data Layer: Uses MongoDB for database management.

## 7. Use Case Diagrams

A use case diagram represents the interaction between users and the system. It includes:  
- User registration and authentication  
- Messaging and consultation with healthcare professionals  
- Emergency contact feature

## 8. Entity-Relationship (ER) Diagram

The ER diagram showcases the relationship between different entities such as users, doctors, messages, and emergency contacts.

## 9. Database Schema

Tables:  
- Users: (user\_id, name, email, password, role)  
- Messages: (message\_id, sender\_id, receiver\_id, content, timestamp)  
- Emergency Contacts: (contact\_id, user\_id, phone, location)

## 10. Functional Requirements

- User authentication (registration & login)  
- Real-time messaging  
- Emergency contact feature  
- Admin dashboard for managing users

## 11. Non-Functional Requirements

- Scalability to support multiple users  
- High security with encrypted data storage  
- User-friendly interface

## 12. Features & Modules

- User Module: Registration, login, profile management  
- Chat Module: Secure messaging, AI bot support  
- Emergency Module: Quick access to emergency contacts  
- Admin Module: User management, system monitoring

## 13. Technology Stack

- Frontend: HTML, CSS, Bootstrap, JavaScript  
- Backend: Node.js, Express.js  
- Database: MongoDB  
- Authentication: JWT-based authentication

## 14. User Interface Design

- Simple and responsive UI with Bootstrap.  
- Light-themed design for a healthcare-friendly look.  
- Interactive chat interface for messaging.

## 15. Security Measures

- Data encryption using HTTPS & SSL.  
- User authentication using JWT.  
- Database security with encrypted storage.

## 16. API Documentation

Example API Endpoints:  
- User Authentication: /api/auth/login  
- Send Message: /api/messages/send  
- Fetch Emergency Contacts: /api/emergency/list

## 17. Deployment Strategy

- The project will be deployed on AWS / DigitalOcean.  
- Continuous integration with GitHub Actions.  
- Scalable cloud database (MongoDB Atlas).

## 18. Testing Plan

- Unit testing using Jest.  
- Integration testing for API endpoints.  
- UI testing with Selenium.

## 19. Future Enhancements

- AI-based diagnosis suggestions.  
- Integration with telemedicine services.  
- Mobile app development for better accessibility.

## 20. Conclusion

MediChat aims to revolutionize online healthcare communication by offering an intuitive and secure platform for users to seek medical assistance, engage in discussions, and access emergency support. Future updates will further enhance the capabilities of the platform to provide a seamless experience.