MediChat Documentation

Table of Contents

---

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 emergenacy contact system.

With the increasing demand for telemedicine and online healthcare services, MediChat serves as a comprehensive solution for connecting users with medical professionals while ensuring user privacy and security.

The rise of digital healthcare has revolutionized the way medical consultations take place. With advancements in AI and cloud computing, platforms like MediChat have the potential to transform patient-doctor interactions, making healthcare more accessible. The application not only provides real-time chat support but also integrates emergency response features, ensuring that patients receive assistance during critical situations.

The Need for Digital Healthcare Solutions

With healthcare access challenges in rural areas, long waiting times at hospitals, and increasing medical costs, there is a growing need for digital platforms that provide immediate consultation. MediChat aims to address these issues by providing an accessible and scalable solution that empowers users with information and emergency medical assistance.

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. It enables users to seek guidance, ask health-related questions, and access emergency medical assistance. The system ensures a seamless experience for both patients and healthcare providers.

With AI-driven assistance, MediChat can preemptively provide health recommendations, answer common questions, and redirect users to emergency contacts when needed. Additionally, the backend system ensures seamless communication between users and verified healthcare professionals.

Key Benefits:

- 24/7 availability of healthcare information.
- Emergency support with rapid response times.
- AI-powered chatbot for common health concerns.
- Secure storage of user health data.

3. Objectives

The main objectives of MediChat are as follows:

- Provide an intuitive and responsive platform for health-related discussions.
- Offer emergency assistance with quick response times.
- Ensure user privacy and data security by using encryption methods.
- Implement AI-driven chat support for common health queries.
- Improve accessibility by designing a user-friendly interface with a responsive layout.
- Enable seamless communication between doctors and patients.
- Develop a scalable and secure web-based system to handle multiple concurrent users.
- Introduce features like automated health assessments and reminders for check-ups.
- Support multilingual interfaces to cater to diverse user bases.

4. Scope

MediChat is designed to serve a broad audience, including patients, doctors, and healthcare institutions.

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

Limitations:

- MediChat does not replace physical examinations and clinical diagnostics.
- Availability of doctors and professionals may vary depending on time zones and workload.
- AI chat assistance may not provide conclusive medical advice and should be used as a preliminary guidance system.
- Internet dependency may limit access in rural or low-connectivity areas.

5. System Model

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

- **Client Side:** Users access the platform via a web browser, allowing them to send queries, initiate chats, and seek emergency assistance.
- **Server Side:** The backend processes user requests, authenticates users, and facilitates messaging.
- **Database:** All user data, messages, and emergency contacts are securely stored and managed in a structured database.

Data Flow in the System

1. A user logs into MediChat and requests a service.
2. The frontend sends the request to the backend API.
3. The backend processes the request, retrieves or updates information in the database, and returns a response.
4. The response is displayed on the user's screen, allowing further interaction.

6. System Architecture

MediChat is built using a **three-tier architecture**:

1. **Presentation Layer:** The frontend, developed using HTML, CSS, JavaScript, and Bootstrap, provides an interactive user interface.
2. **Business Logic Layer:** The backend, implemented with Node.js and Express, handles API requests, authentication, and communication logic.
3. **Data Layer:** MongoDB is used as the primary database for managing user information, messages, and emergency contacts.

The architecture ensures scalability, security, and efficiency by segregating different functionalities into modular components.

Key Components:

- **Frontend:** Handles user interactions and sends requests to the backend.
- **Backend APIs:** Processes requests and manages the application's logic.
- **Database:** Stores critical health data securely.
- **AI Module:** Provides intelligent chatbot responses and recommendations.
- **Authentication Module:** Ensures secure user authentication using JWT.
- **Monitoring & Logging System:** Tracks user interactions to enhance future improvements.

7. Use Case Diagrams

A use case diagram represents the interaction between users and the system. Key functionalities include:

- **User registration and authentication**
- **Messaging and consultation with healthcare professionals**
- **Emergency contact feature** for urgent medical assistance
- **Health monitoring and AI-assisted recommendations**
- **Admin functionalities for monitoring and managing users**

These diagrams illustrate how users navigate the system and interact with different components.

8. Entity-Relationship (ER) Diagram

The ER diagram provides a visual representation of the data model, showcasing the relationships between entities such as users, doctors, messages, and emergency contacts.

Key Entities:

- **Users:** Represents registered users who can seek consultations.
- **Doctors:** Represents verified medical professionals providing guidance.
- **Messages:** Stores conversations between users and healthcare providers.
- **Emergency Contacts:** Maintains emergency details for quick response.
- **AI Logs:** Stores data for improving automated responses over time.
- **Appointments:** Tracks scheduled consultations between users and doctors.

9. Database Schema

Tables:

- **Users:** Stores user credentials and profile data.
- **Messages:** Stores chat history between users and doctors.
- **Emergency Contacts:** Stores emergency contact details for quick assistance.
- **Health Logs:** Tracks interactions for data analytics and future improvements.
- **Appointments:** Stores data regarding scheduled doctor visits.
- **Medical History:** Keeps a log of past consultations for better patient tracking.

The database schema is optimized for fast retrieval, data integrity, and security.

Feasibility Analysis of MediChat

A feasibility analysis ensures that the **MediChat** project is viable in terms of **technical, economic, operational, legal, and scheduling aspects**. The following sections evaluate the feasibility of the system:

---

1. Technical Feasibility

MediChat is technically feasible due to the availability of modern web technologies and cloud computing infrastructure.

Technology Readiness: The project utilizes widely used and well-supported technologies such as **Node.js, Express.js, MongoDB, and Bootstrap**, ensuring scalability and ease of development.
Infrastructure: Cloud-based hosting services (AWS, DigitalOcean, Firebase) allow seamless deployment without requiring extensive physical resources.
Security & Compliance: Security measures like **JWT authentication, HTTPS, SSL encryption**, and adherence to **HIPAA/GDPR** make the platform safe for medical interactions.

Challenges:

- Requires a **stable internet connection** for real-time chat and emergency features.
- AI chatbot implementation needs **continuous improvements** for accuracy.

---

2. Economic Feasibility

The project is cost-effective, making it financially viable.

Estimated Costs:

- **Development Costs:** $5000 - $15,000 (team, resources, APIs)
- **Hosting & Maintenance:** $50 - $200 per month (AWS, DigitalOcean)
- **Security & Compliance:** ~$1000 per year (SSL, encryption, data protection)

Revenue Model:

- **Subscription plans for doctors/hospitals** (e.g., $10/month per doctor).
- **Freemium model** (basic services free, premium consultations paid).
- **Advertisements and sponsorships** (healthcare-related ads).

Challenges:

- Requires initial funding for development.

- User acquisition may take time, impacting revenue growth.

---

3. Operational Feasibility

The system can be smoothly integrated into real-world healthcare operations.

User Adoption: The intuitive UI makes it easy for users (patients, doctors, admins) to interact with the platform.
24/7 Availability: Cloud-based deployment ensures the system remains active and accessible at all times.
Scalability: The platform can handle a growing number of users with **load balancing and cloud infrastructure**.

Challenges:

- **Training healthcare professionals** to use the platform efficiently.
- Managing **real-time doctor availability** for consultations.

---

4. Legal Feasibility

MediChat must comply with medical and data privacy regulations to ensure legal operation.

Regulatory Compliance:

- **HIPAA (Health Insurance Portability and Accountability Act)** for US-based users.
- **GDPR (General Data Protection Regulation)** for European users.
- **Data Privacy Laws** of respective countries where MediChat operates.

- Regular updates are needed to stay compliant with healthcare data protection laws.
- Legal documentation for liability disclaimers and user consent is required.

---

5. Scheduling Feasibility

A realistic timeline ensures that MediChat is developed and launched on schedule.

🗓️ **Estimated Timeline:**

- **Phase 1 (1-2 months):** Requirement analysis, planning, UI/UX design.
- **Phase 2 (3-4 months):** Backend and frontend development, database integration.

- **Phase 3 (2-3 months):** AI chatbot integration, security implementation, and testing.
- **Phase 4 (1-2 months):** Deployment, marketing, and feedback-based improvements.

📌 **Challenges:**

- Possible delays in AI chatbot training and compliance verification.
- Requires a dedicated team for continuous updates and maintenance.

Conclusion

✅ **Overall Feasibility: HIGH**
MediChat is a technically, economically, and operationally feasible project. It addresses a significant market need, has a scalable technology stack, and follows a sustainable business model. Legal compliance and scheduling challenges exist, but they can be managed with proper planning and execution.

Use Case Diagram for MediChat

A **Use Case Diagram** represents the interactions between users and the system. It helps visualize different functionalities MediChat offers to various actors (users, doctors, and admins).

Actors in the System:

1. **User (Patient)** – Can register, log in, chat with doctors, and access emergency contacts.
2. **Doctor** – Can consult with users, provide health advice, and manage appointments.
3. **Admin** – Manages users, monitors the system, and ensures security.

Main Use Cases:

- **User Registration & Login** – Users can create accounts and log in securely.
- **Consultation with Doctors** – Real-time chat between users and medical professionals.
- **Emergency Assistance** – Access emergency contacts and request urgent help.
- **AI Chatbot Support** – Get automated responses for general health queries.
- **Admin Dashboard** – Manage user accounts, monitor chats, and ensure security.

Class Diagram for MediChat

A **Class Diagram** defines the system's structure by illustrating the system's **classes, attributes, methods, and relationships** between objects.

Key Classes and Their Attributes:

1. **User**
    - userID (int)
    - name (String)
    - email (String)
    - password (String)
    - role (String) *(User/Doctor/Admin)*
    - login()
    - logout()
2. **Doctor (inherits from User)**
    - specialization (String)
    - availability (Boolean)
    - chatWithUser(userID)
3. **Admin (inherits from User)**
    - manageUsers()
    - monitorChats()
4. **Message**
    - messageID (int)
    - senderID (int)
    - receiverID (int)
    - content (String)
    - timestamp (DateTime)
5. **EmergencyContact**
    - contactID (int)

- userID (int)
- phone (String)
- location (String)

6. **ChatBot**
    - query (String)
    - response (String)
    - generateResponse(query)
7. **Appointment**
    - appointmentID (int)
    - userID (int)
    - doctorID (int)
    - dateTime (DateTime)
    - status (String) *(Scheduled/Completed)*

Relationships in the Class Diagram:

- **A User can send multiple Messages** (One-to-Many).
- **A User can consult multiple Doctors** (Many-to-Many).
- **A Doctor can have multiple Appointments** (One-to-Many).
- **An Admin manages multiple Users** (One-to-Many).
- **A User can have multiple Emergency Contacts** (One-to-Many).
- **The AI ChatBot interacts with Users** (One-to-One)

Database Design for MediChat

The **MediChat** system requires a well-structured database to efficiently store and manage data related to users, doctors, messages, emergency contacts, and appointments. Below is the database schema, including tables and their relationships.

Database Schema

1. Users Table

Stores user details, including login credentials and roles (patient, doctor, or admin).

Column Name	Data Type	Constraints	Description
user_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each user.
name	VARCHAR(100)	NOT NULL	Full name of the user.
email	VARCHAR(255)	UNIQUE, NOT NULL	User's email for login.
password	VARCHAR(255)	NOT NULL	Encrypted password.
role	ENUM('User', 'Doctor', 'Admin')	NOT NULL	Defines user type.
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	User account creation time.

---

2. Doctors Table

Stores specific information about doctors in the system.

Column Name	Data Type	Constraints	Description
doctor_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each doctor.
user_id	INT	FOREIGN KEY (users.user_id)	Links to Users table.
specialization	VARCHAR(100)	NOT NULL	Doctor's medical specialty.
availability	BOOLEAN	DEFAULT TRUE	Indicates if the doctor is online.

---

3. Messages Table

Stores chat messages between users and doctors.

Column Name	Data Type	Constraints	Description
message_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each message.
sender_id	INT	FOREIGN KEY (users.user_id)	User who sends the message.
receiver_id	INT	FOREIGN KEY (users.user_id)	User who receives the message.
content	TEXT	NOT NULL	The message content.
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Time when the message was sent.

---

4. Emergency Contacts Table

Stores emergency contacts for users.

Column Name	Data Type	Constraints	Description
contact_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each contact.
user_id	INT	FOREIGN KEY (users.user_id)	Links to Users table.
phone	VARCHAR(15)	NOT NULL	Emergency contact number.
location	VARCHAR(255)	NOT NULL	Address of the emergency contact.

---

5. Appointments Table

Stores user-doctor appointments.

Column Name	Data Type	Constraints	Description
appointment_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each appointment.
user_id	INT	FOREIGN KEY (users.user_id)	User scheduling the appointment.
doctor_id	INT	FOREIGN KEY (doctors.doctor_id)	Doctor assigned to the appointment.
date_time	DATETIME	NOT NULL	Scheduled appointment date/time.
status	ENUM('Scheduled', 'Completed', 'Cancelled')	Tracks appointment progress.	

---

6. AI ChatBot Table

Stores user queries and AI-generated responses.

Column Name	Data Type	Constraints	Description
query_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each query.
user_id	INT	FOREIGN KEY (users.user_id)	Links to Users table.
query_text	TEXT	NOT NULL	User's health-related question.
response_text	TEXT	NOT NULL	AI-generated response.
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Time of interaction.

---

7. Health Records Table

Stores past consultations and health-related records.

Column Name	Data Type	Constraints	Description
record_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique ID for each record.
user_id	INT	FOREIGN KEY (users.user_id)	User linked to the record.
doctor_id	INT	FOREIGN KEY (doctors.doctor_id)	Doctor who provided the consultation.
diagnosis	TEXT	NOT NULL	Summary of the diagnosis.
prescription	TEXT	NOT NULL	Medicine and treatment advice.
timestamp	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Date of consultation.

Database Relationships

1. **Users and Messages** – One-to-Many (A user can send multiple messages).
2. **Users and Emergency Contacts** – One-to-Many (A user can have multiple emergency contacts).
3. **Users and Appointments** – Many-to-Many (Users can have appointments with multiple doctors).
4. **Doctors and Appointments** – One-to-Many (A doctor can have multiple appointments).
5. **Users and Health Records** – One-to-Many (A user can have multiple health records).
6. **AI ChatBot and Users** – One-to-Many (A user can ask multiple AI queries).

Database Management System (DBMS) Selection

- **MongoDB** (NoSQL) for flexibility in handling chat and health data.
- **MySQL/PostgreSQL** for structured relational data storage.

MediChat Home About Contact Emergency

medichat

Medicines can cure diseases, but only doctors can cure patients

Proceed

About Us
Learn more about our mission and the services we provide.
Know More

Our Services
Discover the range of services we offer to help you stay healthy.
Know More

Contact Us
Have questions? Get in touch with us today.
Know More

Contact Us

Our Services

24/7 Consultation
We provide round-the-clock consultation with healthcare experts to address your queries and concerns promptly.

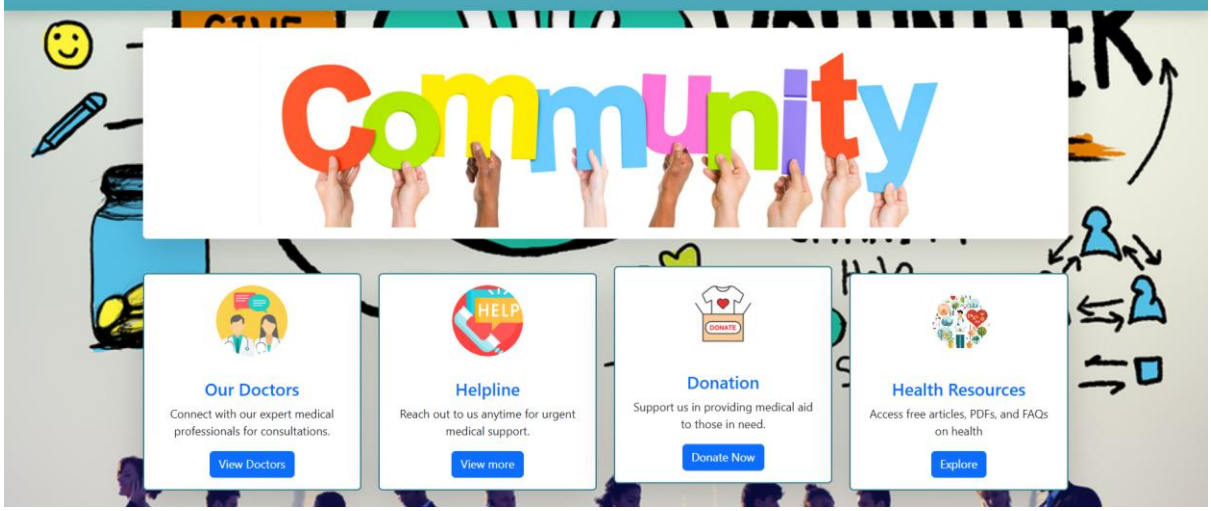
Digital Prescriptions
Get e-prescriptions from certified doctors directly on your device for easy access and use.

Medicine Reminders
Never miss a dose with our smart medicine reminder system, tailored to your schedule.

Health Records
Store and manage your health records securely in one place for quick and easy access.

Community Support
Connect with others and share experiences in our supportive community of patients and experts.

Community Support
We are here to help you with expert guidance and resources.



Our Doctors
Connect with our expert medical professionals for consultations.
View Doctors

Helpline
Reach out to us anytime for urgent medical support.
View more

Donation
Support us in providing medical aid to those in need.
Donate Now

Health Resources
Access free articles, PDFs, and FAQs on health
Explore

HealthBot

Chatbot Suggestions

Hello shubh! How are you feeling today? You can ask me about your health issues.

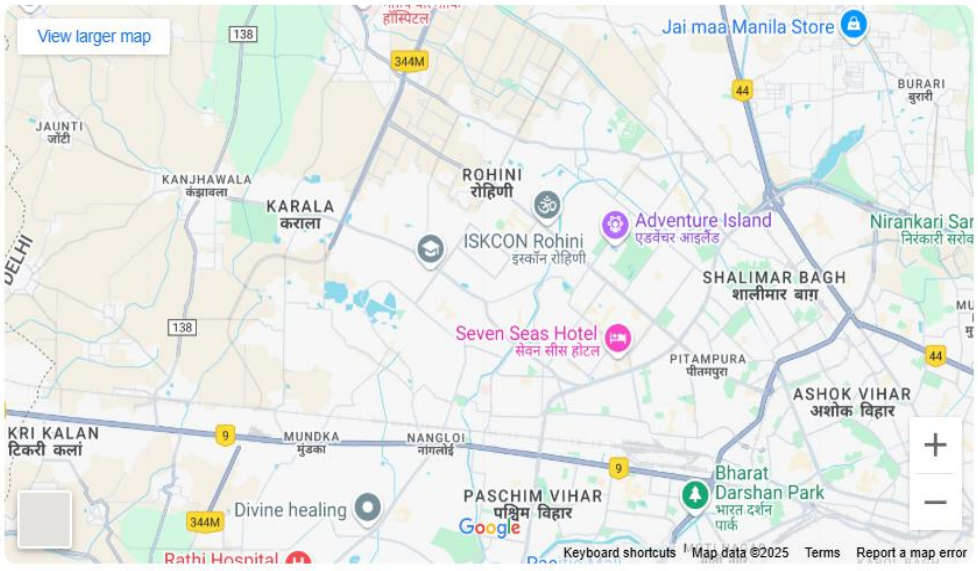
Ask about your health...

Send

Emergency Services

Ambulance Number: 112 (India) / 911 (USA)

Find Nearby Hospitals



Call Emergency