Intelligent healthcare assistant

Project Documentation

Project Title

Intelligent Healthcare Assistant

Team Members

- Rithika
- Rithika
- Reshma Mary
- Ronicka Mariya

1. Introduction

The **Intelligent Healthcare Assistant** project is designed to enhance healthcare services by integrating Artificial Intelligence (AI) to assist patients, doctors, and healthcare providers. It provides medical information, symptom analysis, appointment scheduling, and health monitoring, offering a reliable, user-friendly digital assistant for healthcare needs.

2. Project Overview

Purpose:
The purpose of the Intelligent Healthcare Assistant is to empower patients with instant access to healthcare information while supporting medical professionals with predictive analytics and patient insights. It will act as a 24/7 virtual health companion, reducing dependency on physical consultations for routine guidance and increasing healthcare accessibility.

Features:

- **Conversational Interface** – Natural language interaction to answer health queries.
- **Symptom Checker** – AI-powered symptom analysis with guidance on next steps.

- **Medical Record Management** – Secure storage and retrieval of patient health history.
- **Appointment Scheduling** – Integration with hospital/clinic systems for booking.
- **Health Monitoring** – Real-time monitoring for vitals like heart rate, sugar level, etc.
- **Medication Reminders** – Personalized notifications for timely medication.
- **Predictive Analytics** – AI-driven health risk prediction based on patient data.
- **Multimodal Input Support** – Accepts text, speech, and uploaded reports for analysis.

3. Architecture

Frontend (Streamlit or Mobile App):

- Interactive UI with chat, appointment booking, medical history, and reminders.
- User-friendly dashboards for patients and doctors.

Backend (FastAPI / Flask):

- Handles medical queries, scheduling, and health data management.
- Connects AI models for symptom analysis and recommendations.

AI Integration:

- NLP models for health query understanding.
- ML models for symptom prediction and patient risk assessment.

Database:

- Stores patient health data securely (using encryption).
- Tracks appointments and medication schedules.

Optional Integration:

- Wearable device data (fitness trackers, smartwatches).

4. Setup Instructions

Prerequisites:

- Python 3.9+
- Virtual environment setup
- API keys for AI/NLP services (IBM Watsonx / OpenAI / HuggingFace)
- Secure database credentials

Installation Process:

1. Clone the repository.
2. Install dependencies via `requirements.txt`.
3. Configure credentials in `.env`.
4. Start the backend (FastAPI/Flask server).
5. Launch frontend dashboard or app.
6. Interact with modules (chatbot, appointments, reports).

5. Folder Structure

/app – Backend logic (APIs, models, integrations)
/ui – Frontend components (Streamlit/Mobile UI)
/models – AI models for NLP, symptom analysis
/database – Scripts for database setup and security
assistant.py – Main chatbot logic
scheduler.py – Appointment and reminders
health_monitor.py – Vitals monitoring and prediction
report_generator.py – Generates health reports

6. Running the Application

1. Run backend server.
2. Launch frontend dashboard/app.
3. Use sidebar navigation for chat, health monitoring, appointments, and reports.
4. Upload health documents, interact with AI assistant, and access predictions.

7. API Documentation

- **POST /chat/ask** – Accepts medical queries and returns AI-generated response.
- **POST /upload-record** – Uploads medical reports for analysis and storage.
- **GET /appointments** – Fetch available slots and book appointments.
- **GET /reminders** – Sends personalized medication reminders.
- **GET /predict-health-risk** – Returns predictions for possible health risks.

8. Authentication

- Secure login for patients and doctors.
- Token-based authentication (JWT).
- Role-based access (patient, doctor, admin).

- Data encryption for medical records.

9. User Interface

- Clean, minimal design with intuitive navigation.
- Dashboard with health overview.
- Chat interface for symptom checker.
- Appointment booking and reminders section.
- Downloadable health reports.

10. Testing

- **Unit Testing**: For chatbot responses and scheduling functions.
- **API Testing**: Using Postman/Swagger UI.
- **Manual Testing**: For UI workflows and user interactions.
- **Edge Cases**: Handling invalid symptoms, wrong input formats, or missing

11. Future Enhancements

- Integration with wearable health devices.
- Support for telemedicine (video consultations).
- Multi-language support for accessibility.
- AI-powered personalized health coaching