**Project Title**

Health AI Chatbot

**Team Members**

* Team Member 1: Pradeep Kumar P
* Team Member 2: Ganesh S
* Team Member 3: Joshva A
* Team Member 4: Madhan A

**Project Overview**

The **Health AI Chatbot** is a Streamlit-based web application powered by IBM Watsonx Granite LLM, offering personalized healthcare management, predictive analytics, and chat-based guidance. Its design allows both patients and admins to interact, track health metrics, and leverage AI-driven symptom analysis. It aligns with modern, feature-rich architectures championed by IBM for scalable health or smart city assistants.

**Key Features**

* **Conversational Health Assistant:**  
  Natural language chat powered by IBM Watsonx Granite. Patients ask health queries and get empathetic, AI-generated responses.
* **Symptom-Based Predictions:**  
  Records symptoms and generates structured predictions, including likely conditions, confidence scores, recommended actions, and lifestyle advice.
* **Health Metrics Dashboard:**  
  Patients log, track, and visualize metrics (blood pressure, heart rate, weight, etc.) using a secure internal database with interactive Streamlit charts.
* **Role-Based Access:**  
  Separate login/signup for patients and admins, including secure hashed passwords. Admins can oversee multiple patients.
* **Data Management:**  
  Automatic health data generation and robust storage with SQLite. Support for chat histories and report logging.
* **Customizable Recommendations:**  
  Built-in health tips and personalized messages to encourage healthy behavior (e.g., reminders for drinking water).

**Architecture**

**Frontend (Streamlit)**

* **Multi-Page UI:**  
  The interface comprises login/signup, dashboards, chat, and metric pages, organized with columns, tabs, and navigation cards for easy access.
* **Real-Time Interaction:**  
  Conversations and metrics update instantly using Streamlit's rerun mechanism, forms, and session state variables.
* **Styling:**  
  Custom CSS and card layouts focus on readability and beginner user experience.

**Backend**

* **Database Layer (SQLite):**  
  Patient profiles, user accounts, chat history, health metrics, predictions, and admin-patient chat are all managed through a set of normalized database tables.  
  Encapsulated via the DatabaseManager class for easy CRUD operations.
* **AI Layer (Watsonx Granite):**  
  Uses langchain\_ibm and custom prompt templates for chat and prediction tasks.  
  Centralized through the HealthAI class, which wraps model calls and prompt engineering.
* **Role Management:**  
  Patients are mapped to admins for tailored interaction; role selection and access control are handled at signup.

**Extended Ecosystem (IBM Model)**

* **Document Analysis:**  
  Future extensions include PDF/CSV uploads, semantic vector search using Pinecone, and real-time policy summarization (see IBM sample document for smart city deployment analogues).
* **API Connectivity:**  
  IBM Watsonx and FastAPI for backend expansion, including endpoints for chat, metrics, tips, and feedback, tested via Swagger UI.

**Setup and Installation**

**Prerequisites**

* Python 3.9 or above
* pip, virtualenv
* IBM Watsonx API credentials
* Internet access

**Installation Steps**

1. Clone the project repository.
2. Install required packages via requirements.txt (includes Streamlit, langchain\_ibm, pandas, plotly, sqlite3, dotenv).
3. Configure API keys and environment variables in .env or Streamlit Cloud secrets:  
   WATSONX\_URL, WATSONX\_APIKEY, WATSONX\_SPACE\_ID, WATSONX\_MODEL\_ID.
4. Initialize the SQLite database (automatically via code).
5. Run on local machine using the command:

text

streamlit run healthAI.py

**Troubleshooting**

* Ensure .env is present and valid; handle missing API keys using application error messages.
* Windows users should check file paths and command prompt locations, as specified in previous documentation.

**Usage Flow**

1. **Account Creation**  
   Users choose patient or admin role during signup. Patient accounts are required to be linked to an existing admin. Passwords are hashed for security.
2. **Login**  
   On successful login, session state tracks current user, role, and profile.
3. **Main Dashboard**  
   Overview of available features: chat, health data, predictions, visualizations.
4. **Chat Interaction**  
   Patients and admins engage with the AI assistant for health queries. Previous chat context used to maintain session relevance.
5. **Health Metrics Visualization**  
   Patients can record and view metrics for blood pressure, heart rate, weight, temperature, and more.
6. **Symptom Prediction**  
   Enter symptoms and receive structured analysis with confidence scores, recommendations, and AI-generated reports.
7. **Admin Features**  
   Admins manage patients, check conversations, and send messages (optional future upgrade: see citizen feedback loop, vector document search).

**Advanced IBM Ecosystem (Inspired by Sample Doc)**

* **Smart City and Healthcare Extensions:**  
  Consider extending to handle multimodal inputs, policy summarization, anomaly detection, and eco-tip generation for broader use cases.
* **API Expansion:**  
  Add FastAPI endpoints for document handling, feedback, and KPI forecasts as in the reference architecture.
* **Security:**  
  Implement future upgrades like JWT or OAuth2, role/session management, and enhanced analytics (planned in IBM system design).

**Folder Organization**

| **Folder/File** | **Purpose** |
| --- | --- |
| healthAI.py | Main Streamlit app logic and UI |
| health\_ai.db | SQLite database (created at runtime) |
| .env / secrets | API keys and configuration |
| (future additions) | For extensions: FastAPI, document handler, vector search |

**Testing and Reliability**

* Unit tests for API and utility functions using sample data.
* Manual validation for login, chat, metric recording, and predictions.
* Edge case handling for user creation, API errors, and invalid inputs.

**Known Issues and Future Enhancements**

* Improve session isolation for multi-user deployments.
* Currently, the AI model is configured with a limit on the **maximum number of tokens or responses per interaction** (e.g., max\_new\_tokens = 500), which restricts very long or deeply detailed answers.
* This token limit can sometimes cause responses to be **cut off prematurely** or result in incomplete explanations.
* Future upgrades could include increasing token limits, enabling multi-turn dialogue continuity, or splitting responses for longer queries.
* Potential rate limits on IBM Watsonx API may affect extensive usage availability.