HealthAI: Intelligent Healthcare Assistant Using IBM Granite
Category: Cloud Application Development
Skills Required: Python, IBM Cloud, Scikit-Learn

Project Description

HealthAI harnesses IBM Watson Machine Learning and Generative AI to provide intelligent healthcare assistance, offering users accurate medical insights. The platform includes:

- **Patient Chat:** Answer health-related questions with clear, empathetic responses.
- **Disease Prediction:** Evaluate user-reported symptoms to deliver potential condition predictions, likelihood assessments, and recommended next steps.
- **Treatment Plans:** Generate personalized, evidence-based treatment plans (medications, lifestyle modifications, follow-up testing).
- **Health Analytics:** Visualize and monitor patient health metrics (heart rate, blood pressure, blood glucose, etc.) with AI-driven insights.

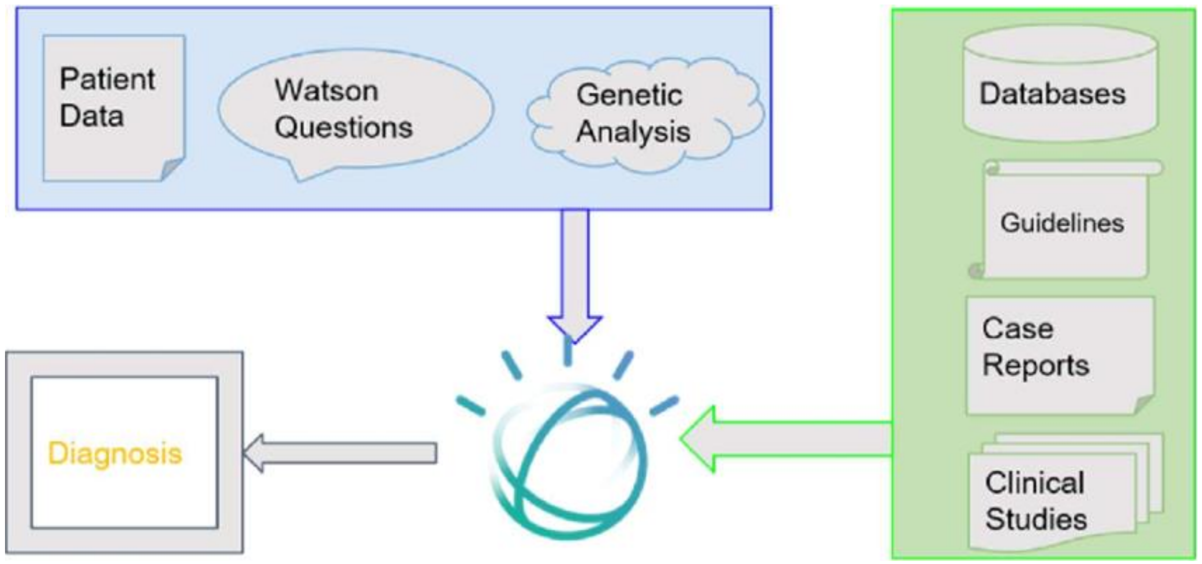
Built with Streamlit and powered by IBM Watson and the Granite-13b-instruct-v2 model, HealthAI ensures a seamless, user-friendly experience, secure API key management, and responsible data handling—empowering users to make informed health decisions with confidence.

Scenarios

1. **Symptom-Driven Disease Prediction**
   - *Action:* User inputs symptoms (e.g., headache, fatigue, mild fever).
   - *Outcome:* HealthAI analyzes symptoms plus patient profile, returns potential conditions with likelihoods and next-step recommendations.
2. **Personalized Treatment Planning**
   - *Action:* User enters a diagnosed condition.
   - *Outcome:* AI generates a comprehensive treatment plan including medications, lifestyle advice, and suggested tests.
3. **Health Trends Insight**
   - *Action:* User views the Health Analytics dashboard.
   - *Outcome:* Charts of vital signs over time appear alongside AI-generated insights highlighting concerns and improvement tips.
4. **On-Demand Patient Chat**
   - *Action:* User asks any health-related question via chat interface.

- *Outcome:* AI provides an empathetic, fact-based answer, acknowledges limitations, and advises when to consult a professional.
-

TECHNICAL ARCHITECTURE

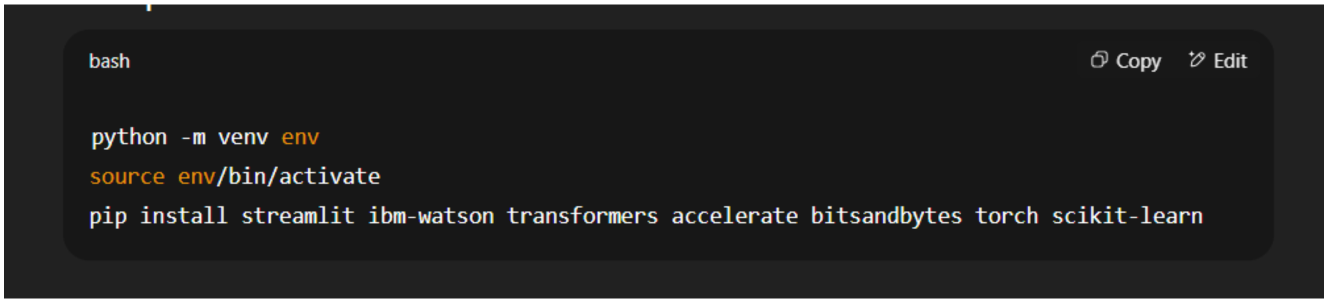


Prerequisites

1. **Python (3.7+)**
2. **Streamlit** for frontend UI
3. **IBM Watson SDK** and **Granite-13b-instruct-v2** model via Hugging Face (transformers, accelerate, bitsandbytes)
4. **Scikit-Learn** (for auxiliary analytics)
5. **IBM Cloud Account** with Watson Machine Learning service deployed
6. **Sufficient Hardware** (≥16 GB RAM; NVIDIA GPU with ≥8 GB VRAM recommended)
7. **Internet Connection** for initial model downloads
8. **Project Structure:**
    - app.py (Streamlit app entry point)
    - templates/ (if using Flask alternatively)
    - static/ (CSS, images)

Project Setup & Architecture

1. **Model & Libraries Selection**
   - Confirm Granite-13b-instruct-v2, transformers, accelerate, bitsandbytes, PyTorch, Streamlit.
2. **System Design**
   - Input → AI inference → Data processing → Visualization → UI.
   - Secure handling of API keys and patient data.
3. **Development Environment**



```bash
python -m venv env
source env/bin/activate
pip install streamlit ibm-watson transformers accelerate bitsandbytes torch scikit-learn
```

Core Functionalities

- **Activity 1:** Load Granite model and IBM Watson credentials.
- **Activity 2:** Implement Streamlit pages/components:
  - Chat input & response display
  - Symptom form → prediction
  - Condition form → treatment plan
  - Analytics charts
- **Activity 3:** Develop helper modules:
  - generate_response() (Granite inference)
  - predict_disease() (symptom analysis)
  - create_treatment_plan()
  - compute_health_metrics() (analytics)

- **Activity 4:** Secure API key/config management (e.g., using environment variables).

Data Handling & Logic

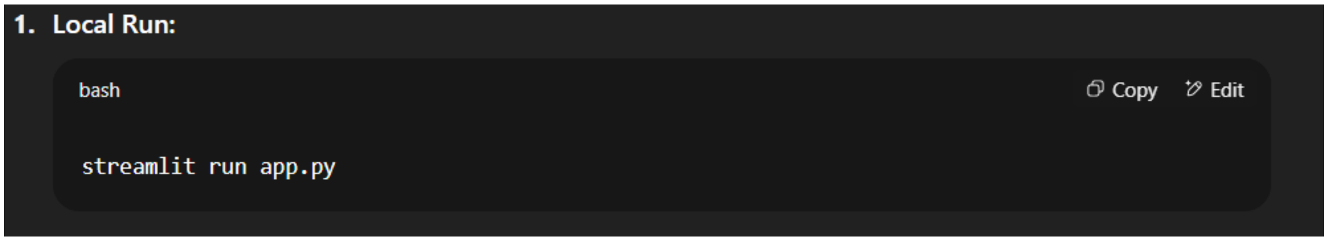
- Store session-based chat history and analytics in memory (or lightweight DB).
- Process inputs through AI functions, format outputs for UI.
- Aggregate time-series health data for insights.

Frontend Development (Streamlit)

- **Layout:** Sidebar navigation (Chat, Prediction, Treatment, Analytics).
- **Forms & Inputs:**
    - Text inputs for chat & symptoms
    - File uploader (optional) for health logs
- **Visualization:**
    - Line charts for vitals (Streamlit's st.line_chart)
    - Tables for predicted conditions & treatment steps

Integration & Testing

1. **Local Run:**


1. **Local Run:**

```bash
streamlit run app.py
```

1. **Test Flows:**
    - Chat Q&A
    - Symptom → prediction
    - Condition → treatment

- Data upload → analytics

2. **Debug & Refine UI/UX** based on feedback.

Deployment

1. **Containerize:** Dockerfile with streamlit image.
2. **Host:** IBM Cloud Run or similar.
3. **SSL & Security:** Ensure HTTPS, secure API key storage.
4. **Monitoring:** Track errors, usage metrics, model performance.

Documentation & Handover

- **README:** Setup, usage, API reference.
- **User Guide:** Screenshots, feature descriptions.
- **Demo Video:** https://drive.google.com/file/d/1BLbZqL1Wh79VcvT7xCTNid-EYhLvzKwz/view?usp=drive_link
- *HealthAI* delivers an end-to-end intelligent healthcare assistant—streamlining medical information access, personalized recommendations, and health analytics for better patient engagement and outcomes.