HealthAI — Project Documentation

# 1. Overview

HealthAI is a Streamlit-based demo application that simulates an AI-driven health assistant. It provides:

- Patient Chat for Q&A with keyword-based mocked responses.  
- Disease Prediction suggesting possible conditions from entered symptoms.  
- Treatment Plan Generator for common conditions (e.g., migraine, diabetes, asthma).  
- Health Analytics Dashboard with synthetic patient health data and insights.

Disclaimer: This project is for educational/demo purposes only. It is not medical advice and must not be used for clinical decision-making.

# 2. Features

- Interactive Chat — Patients can ask health-related questions.  
- Disease Prediction — Keyword-based likelihoods and recommendations.  
- Treatment Plans — Predefined management plans for 10+ conditions.  
- Health Analytics — Synthetic vitals visualization, trends, and symptom frequencies.

# 3. System Architecture

## High-Level Flow

[ User Input ]  
↓  
[ Streamlit UI ] ───► [ Core Functions ] ───► [ Mock AI Model Adapter]

## Components

- UI (Streamlit): Renders pages, forms, charts, and navigation.  
- Core Functions: predict\_disease, generate\_treatment\_plan, answer\_patient\_query, generate\_sample\_health\_data.  
- Mock Model: call\_granite\_model — keyword-based, not a real AI/LLM.

# 4. Installation & Setup

## Requirements

Python 3.9–3.11  
Recommended packages:  
streamlit>=1.33  
pandas>=2.0  
numpy>=1.24  
plotly>=5.20

## Steps

# Create virtual environment  
python -m venv .venv  
  
# Activate environment  
# Windows:  
.venv\Scripts\activate  
# macOS/Linux:  
source .venv/bin/activate  
  
# Install dependencies  
pip install -r requirements.txt  
  
# Run app  
streamlit run app.py

# 5. Application Pages

Sidebar:  
- Patient profile form (name, age, gender, existing conditions).  
- Navigation to features: Patient Chat, Disease Prediction, Treatment Plans, Analytics.

Patient Chat:  
- Users type health questions.  
- Responses returned from mock AI.  
- Chat history preserved in session state.

Disease Prediction:  
- Input: Symptom text, age, gender, conditions.  
- Output: Possible conditions with likelihoods.

Treatment Plans:  
- Input: Condition + patient profile.  
- Output: Structured treatment plan (medication, lifestyle, follow-up).

Health Analytics:  
- Synthetic 30-day data: heart rate, blood pressure, glucose.  
- Visualizations: line charts, pie chart, summary metrics.  
- “AI-generated” insights (mocked).

# 6. API Reference

call\_granite\_model(prompt: str) -> str  
- Mocked model returning canned responses.  
  
answer\_patient\_query(query: str, chat\_history: list) -> tuple  
- Handles patient Q&A, updates chat history.  
  
predict\_disease(symptoms: str, patient\_profile: dict) -> str  
- Returns possible conditions and recommendations.  
  
generate\_treatment\_plan(condition: str, patient\_profile: dict) -> str  
- Returns structured treatment plans.  
  
generate\_sample\_health\_data(days: int = 30) -> (pd.DataFrame, dict)  
- Generates synthetic health metrics and symptom frequencies.  
  
generate\_ai\_health\_insights(df: pd.DataFrame) -> str  
- Analyzes last 7 days’ vitals and returns mock recommendations.

# 7. Usage Example

Start the App:  
streamlit run app.py  
  
Patient Chat Example:  
Input: I have a headache  
Output: 'A headache may be caused by stress, dehydration... Seek medical help if persistent.'  
  
Disease Prediction Example:  
Input symptoms: fever, cough  
Output:  
- Viral infection (50%)  
- Bacterial infection (30%)  
- COVID-19 (20%)  
Recommendations: hydrate, monitor, seek care if worsens

# 8. Limitations

- Mocked AI: Responses are static and rule-based.  
- Not medically validated: Must not be used in healthcare practice.  
- Session-only state: No persistence or real patient data handling.  
- No typo handling in symptom input (requires exact keywords).

# 9. Future Improvements

- Replace mock model with a real LLM API (e.g., OpenAI, local model).  
- Add input normalization (symptom spell-check, synonyms).  
- Support CSV/real health data uploads for analytics.  
- Add testing (unit + integration).  
- Improve safety guardrails (standard disclaimers, restricted outputs).  
- Modularize code into separate files (core.py, ui.py, etc.).

# 10. Safety & Legal

- This software is not a medical device.  
- Outputs are for demonstration only.  
- Always consult licensed professionals for medical concerns.