# Solution Architecture – HealthAI

Date: 25 JUNE 2025

Team ID: LTVIP2025TMID31761

Project Name: HealthAI

Maximum Marks: 4

## Overview

The solution architecture defines the structure and behavior of the HealthAI system, bridging the gap between healthcare needs and scalable AI solutions. It outlines how different components interact to deliver intelligent healthcare assistance.

## Goals of the Architecture

* Provide a robust AI-driven assistant for medical queries using natural language processing.
* Enable symptom-based disease prediction with clear likelihood and actionable insights.
* Generate custom, personalized treatment plans using generative AI logic.
* Visualize patient health data trends through an interactive dashboard.
* Ensure secure API integration, efficient session management, and a foundation for multi-format data export and persistence.

## Architectural Components

Frontend (UI): Streamlit-based interface with a tabbed layout for core functionalities (Patient Chat, Disease Prediction, Treatment Plans, Health Analytics) and a persistent sidebar for Patient Profile management.

Application Logic (Python Backend): Python code utilizing Streamlit's framework for managing UI state, orchestrating data flow, and calling AI models. Incorporates Pandas for data manipulation and Plotly for visualization generation.

AI Models:  
- Current Simulation: Google Gemini-2.0-Flash for conversational AI, disease prediction, treatment plan generation, and health insights.  
- Target Integration: IBM Granite-13B-instruct-v2 via IBM Watson Machine Learning for the primary AI functionalities.

Data Flow & Management:  
- Patient Profile Data: Stored in Streamlit's session state for in-session persistence.  
- Health Metrics Data: Synthetic data generated for demonstration purposes; future integration with actual data sources and a database (e.g., Firestore) is planned for persistent storage.  
- Chat History: Managed within Streamlit's session state.  
- Logging Layer: (Future/Advanced) Placeholder for storing detailed interaction logs and performance metrics.

Analytics Engine: Plotly for generating interactive line and pie charts from health data.

Security Layer: Environment variable management (python-dotenv locally) for API token access. (Future: Robust authentication and authorization for persistent data).

Output Services: Display of AI-generated text, structured JSON outputs, and interactive visualizations within the Streamlit UI.

## Data Flow Summary

User Interaction: Text input (queries, symptoms, conditions) is entered into the Streamlit UI via text boxes or areas. Patient profile data is managed via sidebar inputs.

Model Processing:  
- User queries for Chat, Prediction, or Treatment are sent to the Python backend.  
- The backend constructs specific prompts, incorporating relevant patient profile data and current context.  
- These prompts are then sent to the Google Gemini API (currently) for AI response generation.  
- (Future: Prompts will be sent to IBM Granite-13B-instruct-v2 via IBM Watson Machine Learning API).

Visualization: Processed health metrics data is used by Plotly to render and update interactive charts on the Health Analytics dashboard.

Session Management: All user inputs, chat history, and generated data (e.g., predictions, plans) are maintained within Streamlit's session state for the duration of the user's active session.

Logging: (Future) Operational data and AI interaction details can be logged for performance monitoring and debugging.

## Development Phases

1. Phase 1: UI Setup & Core Layout (Current)  
   - Basic Streamlit layout with tabbed navigation for Patient Chat, Disease Prediction, Treatment Plans, and Health Analytics.  
   - Implementation of the persistent Patient Profile sidebar.  
   - Integration of custom CSS for enhanced aesthetics.
2. Phase 2: AI Functionality Simulation (Current)  
   - Integration of Google Gemini API to simulate AI responses for Patient Chat, Disease Prediction, Treatment Plan Generation, and Health Insights.  
   - Development of robust prompting strategies for each AI feature.
3. Phase 3: Data Management & Visualization (Current)  
   - Implementation of session state management for user profile and chat history.  
   - Development of synthetic health data generation for the dashboard.  
   - Integration of Plotly for dynamic health trend charts and metrics display.
4. Phase 4: Testing & Performance Validation (Ongoing)  
   - Conducting Unit and User Acceptance Testing (UAT) for all implemented features.  
   - Evaluating Streamlit application responsiveness and AI response times.
5. Phase 5: IBM Watson ML Integration (Future)  
   - Transition from Gemini API simulation to direct integration with IBM Granite-13B-instruct-v2 via IBM Watson Machine Learning SDK.  
   - Refactoring API calls and authentication mechanisms.
6. Phase 6: Data Persistence & Scalability (Future)  
   - Integration with a database (e.g., Firestore) for long-term storage of patient profiles, chat history, and health data.  
   - Implementation of user authentication and data privacy measures for multi-user support.

## Architecture Diagram

(To be included or rendered separately in final submission - A visual representation of the components and data flow as described above.)

## High-level Flow

Current State:  
User → Streamlit UI → Python Backend (App Logic) → Google Gemini API (Simulated AI) → Output to Streamlit UI

Future State:  
User → Streamlit UI → Python Backend (App Logic) → IBM Watson ML (IBM Granite) → Output to Streamlit UI

## Conclusion

The solution architecture for HealthAI ensures alignment between real healthcare problems and scalable AI solutions. It provides a robust foundation for an intelligent healthcare assistant, with a clear roadmap from its current simulated state to full integration with enterprise-grade AI models and comprehensive data management.