**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 19/05/2025 – 30/06/2025 |
| Team ID | LTVIP2025TMID31711 |
| Project Name | HealthAI: Intelligent Healthcare Assistant Using IBM Granite |
| Maximum Marks | 4 Marks |

**Solution Architecture:**

What is Solution Architecture?

Solution Architecture defines how a software solution addresses a specific business problem using the right technologies. In the case of HealthAI, the goal is to provide AI-powered medical assistance (symptom understanding, disease identification, and treatment suggestions) using IBM Watsonx, with Streamlit as the front end and cloud infrastructure for deployment.

🎯 Goals of the Solution Architecture

1. Solve the business problem:

Help users get intelligent health suggestions without needing medical expertise.

1. Describe software behavior:

Show how users, AI, APIs, and databases interact.

1. Define features and development stages:

From profile input to AI response and analytics.

1. Specify technologies and flow:

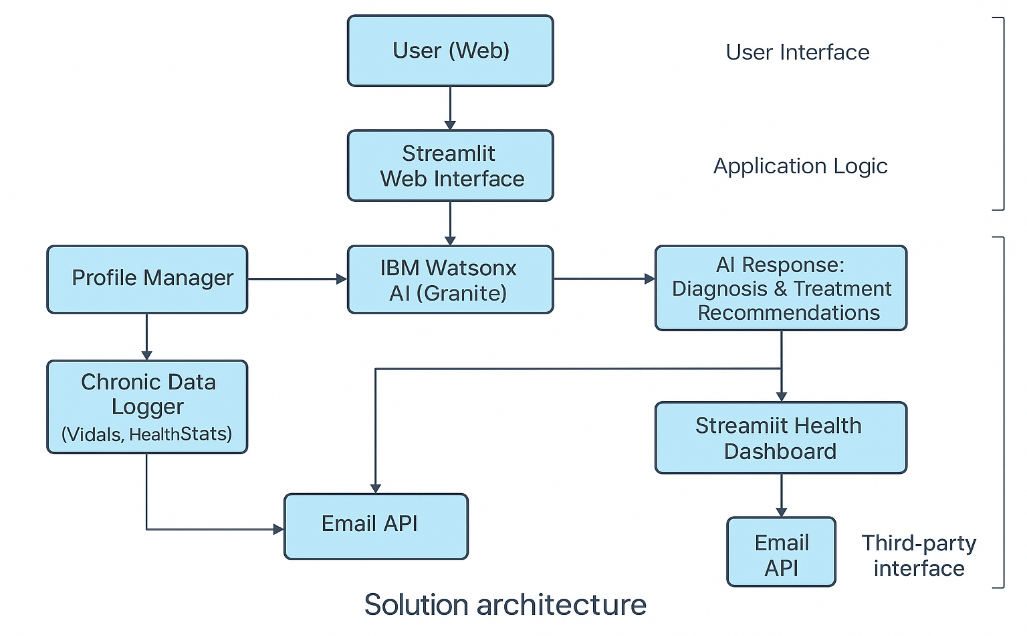
Clear backend-frontend interaction with IBM Watsonx on IBM Cloud.

**Solution Architecture Diagram – HealthAI**

Below is a conceptual layout of how your app functions internally. It includes:

Key Components:

* User Interface (Streamlit)
* Application Logic (Python)
* AI Backend (IBM Watsonx)
* Local Storage / IBM Cloudant (optional)
* Email API (optional)

****

**1. Find the Best Tech Solution to Solve Existing Business Problems**

**Business Problem:**

* Citizens often lack quick access to preliminary medical advice.
* They struggle with symptom understanding, treatment options, or when to see a doctor.
* There is no single multilingual AI-based health assistant available on-demand.

**Tech Solution:**

* HealthAI provides a Generative AI-powered assistant using IBM Watsonx, deployed via Streamlit.
* It enables users to:
  + Input symptoms in their preferred language
  + Receive AI-generated diagnosis and treatment advice
  + Log and track health data (e.g., glucose, heart rate)
  + Access their analytics via an interactive dashboard
* IBM Cloud ensures scalability, speed, and security.

**2. Describe the Structure, Characteristics, Behavior, and Other Aspects of the Software to Project Stakeholders**

|  |  |
| --- | --- |
| **Aspect** | **Details** |
| **Structure** | Modular components (UI, AI logic, storage, analytics) integrated using Python and APIs |
| **Front-end** | Built using **Streamlit** – responsive, interactive |
| **Back-end** | Python-based logic that communicates with **IBM Watsonx** for AI responses |
| **AI Behavior** | IBM Watsonx (Granite model) interprets symptoms and generates natural-language diagnosis and treatment |
| **Storage** | User profile and logs stored locally (SQLite) or on **IBM Cloudant** if deployed on cloud |
| **Security** | Email confirmation (optional), app hosted on trusted platforms (IBM Cloud, Streamlit Cloud) |
| **Scalability** | Cloud-compatible architecture for future API integrations, ML-based upgrades, or voice interfaces |

**3. Define Features, Development Phases, and Solution Requirements**

**Key Features**

* Health Profile Setup (mandatory)
* Symptom-to-Disease Diagnosis
* Treatment Advice Generation
* Chronic Disease Logger (Glucose, Heart Rate)
* Health Analytics Visualization
* Email Delivery (optional)
* Multilingual Support (via prompt formatting)

**Development Phases**

|  |  |
| --- | --- |
| **Phase** | **Feature** |
| Phase 1 | Streamlit Interface & Profile Setup |
| Phase 2 | IBM Watsonx AI Integration |
| Phase 3 | Logging Module + Health Analytics |
| Phase 4 | Output + Deployment Testing |

**Solution Requirements**

* **Functional**: Symptom checker, treatment generator, health logger, dashboard
* **Non-functional**: Fast response (<2s), user-friendly UI, secure storage, always available

**4. Provide Specifications for Definition, Management, and Delivery**

|  |  |
| --- | --- |
| **Spec Type** | **Details** |
| **Technology Stack** | Streamlit (UI), Python (Backend), IBM Watsonx (AI), SQLite/IBM Cloudant (DB), Gmail API (email) |
| **Architecture** | Component-based, loosely coupled modules |
| **Deployment** | Streamlit Cloud or IBM Cloud |
| **Model Integration** | IBM Watsonx API using secure keys |
| **Versioning & Management** | Code managed via GitHub, modular scripts, and updated prompts |
| **Delivery** | Fully functional web app delivered as Streamlit link or cloud-hosted app (with optional ZIP for offline use) |