**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

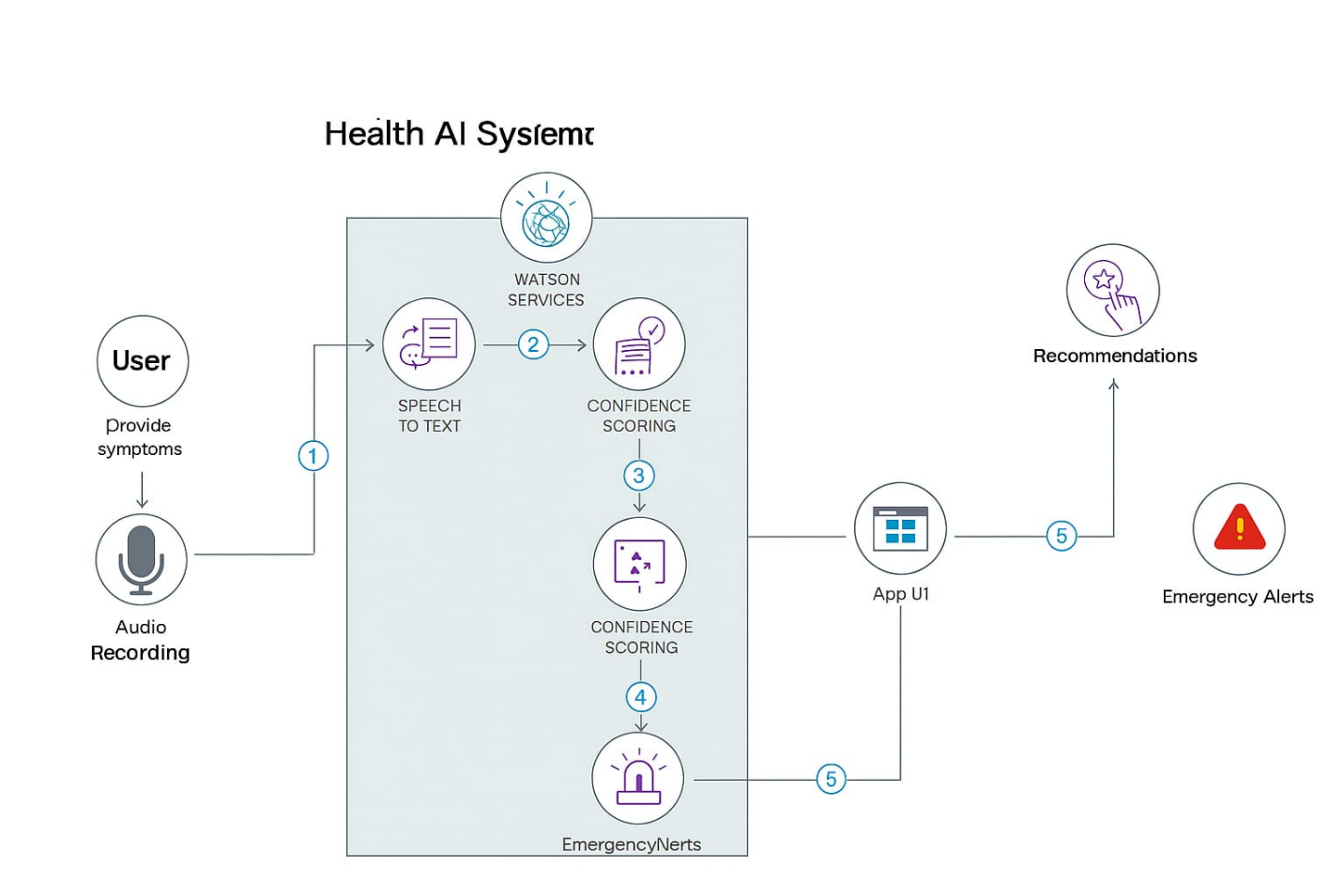
|  |  |
| --- | --- |
| Date | 26 June 2025 |
| Team ID | LTVIP2025TMID36168 |
| Project Name | HealthAI: Intelligent Healthcare Assistant Using IBM Granite |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Health AI application**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)



**Guidelines**

* Show all process blocks (e.g., input, prediction, alerts).
* Separate local (Gradio) and cloud (IBM Watson) components.
* Mark third-party APIs (e.g., Watson, Twilio).
* Indicate storage (e.g., IBM DB2, Cloud Object Storage).
* Highlight ML model interface if used.

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1 | **User Interface** | Gateway for symptom input, chat interactions, wellness tips, and reminders | **Gradio** (Blocks or ChatInterface) integrated with IBM Watson Assistant UI |
| 2 | **Application Logic – 1** | Core logic for processing user input, session/state management | **Python** (Flask or FastAPI) |
| 3 | **Application Logic – 2** | Manages AI-driven conversation, including intent recognition and responses | **IBM Watson Assistant** via Python SDK |
| 4 | **Database** | Stores user profiles, session logs, symptom entries, reminders | **SQLite** (dev) or **MySQL/PostgreSQL**, accessed via Python (e.g., SQLAlchemy) |
| 5 | **File Storage** | Holds uploaded images (e.g., rashes), transcripts, or voice recordings | Local filesystem or optional upload to **IBM Cloud Object Storage** |
| 6 | **Machine Learning Model** | Executes disease/risk prediction, symptom classification, and wellness advice | **Python** (scikit-learn, TensorFlow) with **IBM Watsonx.ai** |
| 7 | **Infrastructure** | Hosts all components, ensures scalability, integrates IBM services | **Dockerized Python containers** on **IBM Cloud** (Kubernetes or Cloud Foundry) |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristic** | **Description** | **Technology** |
| 1 | **Open‑Source Frameworks** | Core frameworks driving UI and backend | Gradio for UI; Flask or FastAPI for backend logic |
| 2 | **Security Implementations** | Methods to secure data and access | OAuth 2.0 (Flask‑OAuthlib), HTTPS/TLS, IAM roles, OWASP best practices |
| 3 | **Scalable Architecture** | Structured approach allowing independent scaling of components | Microservices with Docker; independent Python services |
| 4 | **Availability** | Ensures service remains online and responsive | Load‑balanced Docker containers on IBM Cloud/Kubernetes with auto‑scaling |
| 5 | **Performance** | Optimizations to support high traffic throughput | Redis cache, in‑memory caching, CDN for static assets; designed for 50+ requests/sec |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)