IBM SkillsBuild Internship on AI and Cloud

Health Symptom Checker Project Documentation

Kirti vardhan singh

Kirtivardhan7549@gmail.com

August 5, 2025

Contents

1 Project Overview **2**

2 Features **2**

3 Architecture **3**

4 Technology Stack **3**

5 System Setup **4**

5.1 Watson Assistant 4

5.2 Backend (Flask API) 4

5.3 Twilio WhatsApp + Flex Integration (Optional) 5

6 Sample Queries and Responses **6**

7 Limitations **8**

8 Conclusion **8**

9 License **8**

Abstract

This document presents the Health Symptom Checker project, a conversational AI solution developed using IBM Cloud's Watson Assistant and a custom Flask-based backend. The system is designed to provide users with relevant health advice by allowing them to describe their symptoms through a natural language interface. It also features an optional escalation path to a live agent, facilitated by Twilio, for cases requiring human intervention.

1. Project Overview

The Health Symptom Checker is an interactive AI tool created to guide users through their health concerns by interpreting common symptoms and offering appropriate first-level advice. The core of the project is the integration between IBM Watson Assistant, which handles the conversational flow, and a dedicated backend search service that delivers dynamic, customized responses based on a predefined dataset. To ensure users can receive further help when needed, a seamless handoff to a live agent is enabled through a Twilio-powered escalation path.

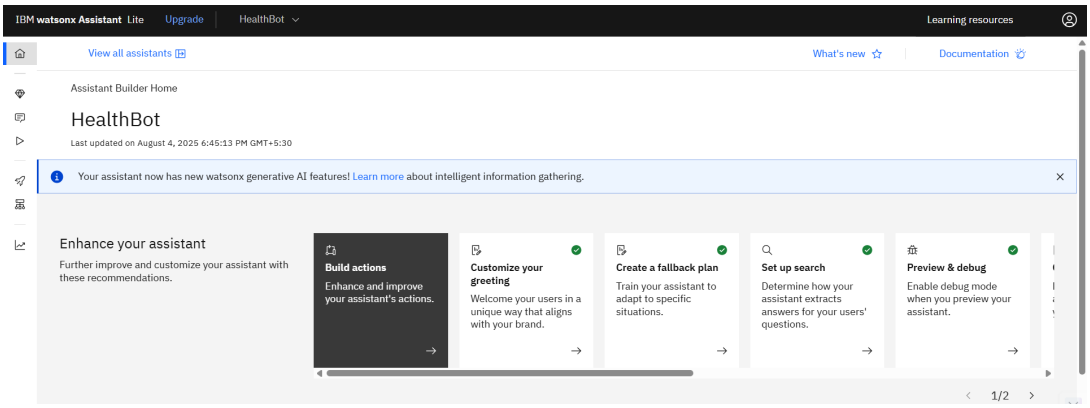

Figure 1: The IBM Watson Assistant dashboard for the HealthBot project.

2. Features

The Health Symptom Checker includes the following key features:

- **Natural Language Understanding:** The ability to interpret symptom descriptions provided by users in plain, natural language.
- **Dynamic Responses:** Delivers relevant advice based on a structured dataset, which is retrieved via a custom backend service.
- **Custom Backend:** A dedicated Flask API is responsible for handling all symptom search queries from the assistant.
- **Live Agent Escalation:** Supports seamless handoff to live agents for more complex queries using Twilio Flex.

- **Free-Tier Deployment:** The entire system is designed to be fully deployable using the free tiers of cloud services, including IBM Cloud, Render, and the Twilio Sandbox.

3. Architecture

The system architecture is composed of three primary components that work together to deliver the conversational experience:

1. **IBM Watson Assistant:** Serves as the frontend of the system, responsible for interpreting user messages, managing the conversation state, and triggering the appropriate actions based on user intent.
2. **Flask Backend API:** A custom search API that acts as the brain of the operation. When called by Watson Assistant, it searches a knowledge base to find the most relevant advice for the user's described symptoms and returns it in a structured format.
3. **Twilio Flex Integration (Optional):** Provides the connection to a live agent. When a user requests to speak to a person, Watson Assistant triggers a signal to Twilio, which then manages the handoff to an available agent.

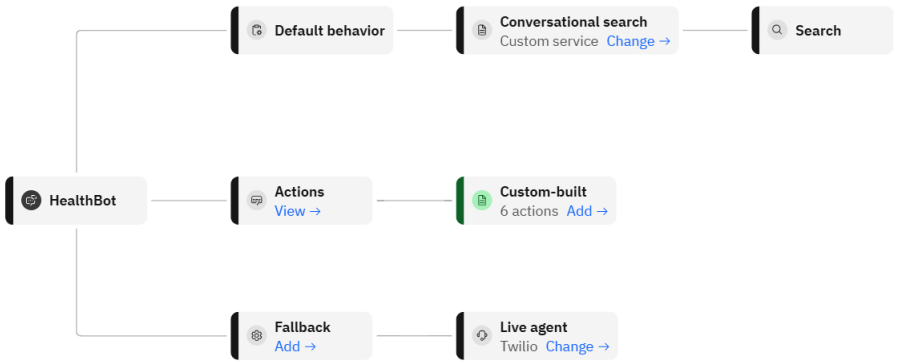

Figure 2: System Architecture Diagram showing the flow from HealthBot to Actions, Fallback, and Conversational Search.

4. Technology Stack

The project leverages a modern, cloud-native technology stack:

- **Natural Language Processing:** IBM Watson Assistant
- **Custom Backend:** Python with the Flask framework
- **Hosting:** Render.com (free tier) for the backend API

- **Agent Handoff:** Twilio for WhatsApp messaging and Flex for the agent interface
- **Version Control:** GitHub for code management and configuration

5. System Setup

5.1. Watson Assistant

The assistant is built within IBM Watson Assistant using the Actions skill, which provides a straightforward way to define conversational flows. Each action is mapped to a specific user intent, such as checking a particular symptom (e.g., headache, fever). A dedicated search action is configured to make external API calls to the Flask backend, allowing the assistant to fetch and display dynamic advice to the user.

5.2. Backend (Flask API)

A lightweight Flask web application serves as the project's backend. It is deployed on Render and configured to listen for POST requests from Watson Assistant. When a request containing a symptom query is received, the API searches through a predefined dictionary of symptoms and corresponding advice. The relevant information is then returned to Watson Assistant in a structured JSON format to be displayed to the user.

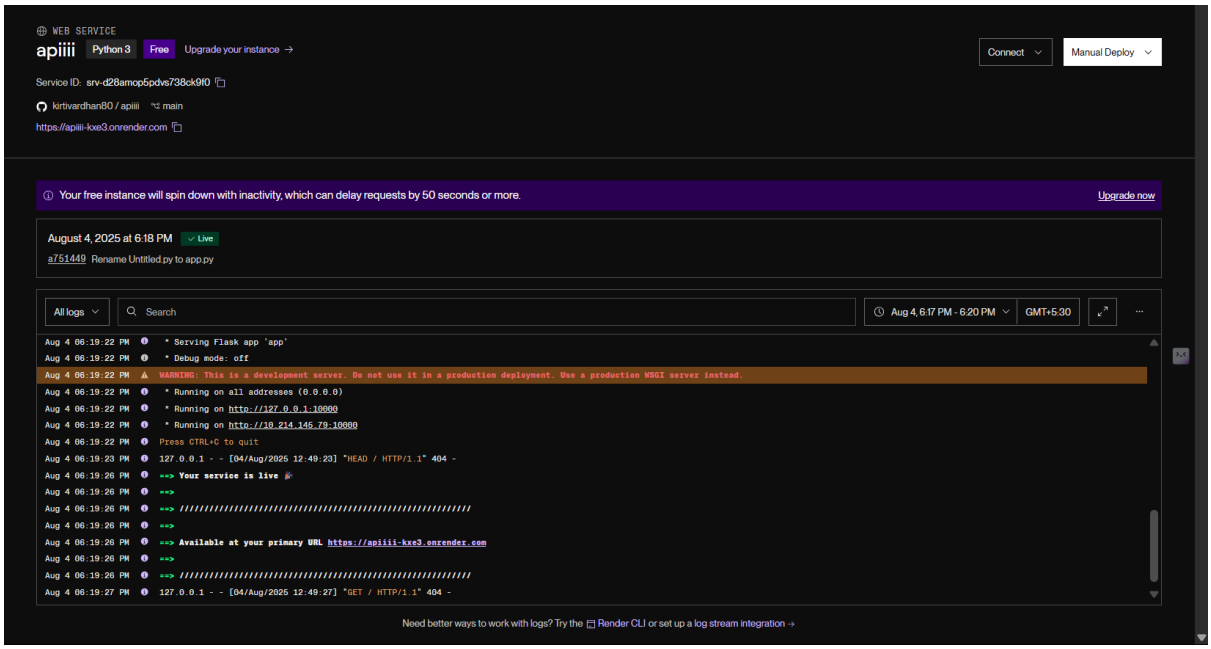

Figure 3: The Render.com dashboard showing the hosting status and activity logs for the backend Flask API.

5.3. Twilio WhatsApp + Flex Integration (Optional)

To enable live agent escalation, the Twilio Sandbox for WhatsApp is used for testing without requiring official WhatsApp approval. A flow built in Twilio Studio or Flex handles routing user messages to a live agent when triggered. The handoff is initiated when Watson Assistant sends a specific 'connect$_{t}o_{a}gent$'$signal, which tells Twilio to begin the escalation process.$

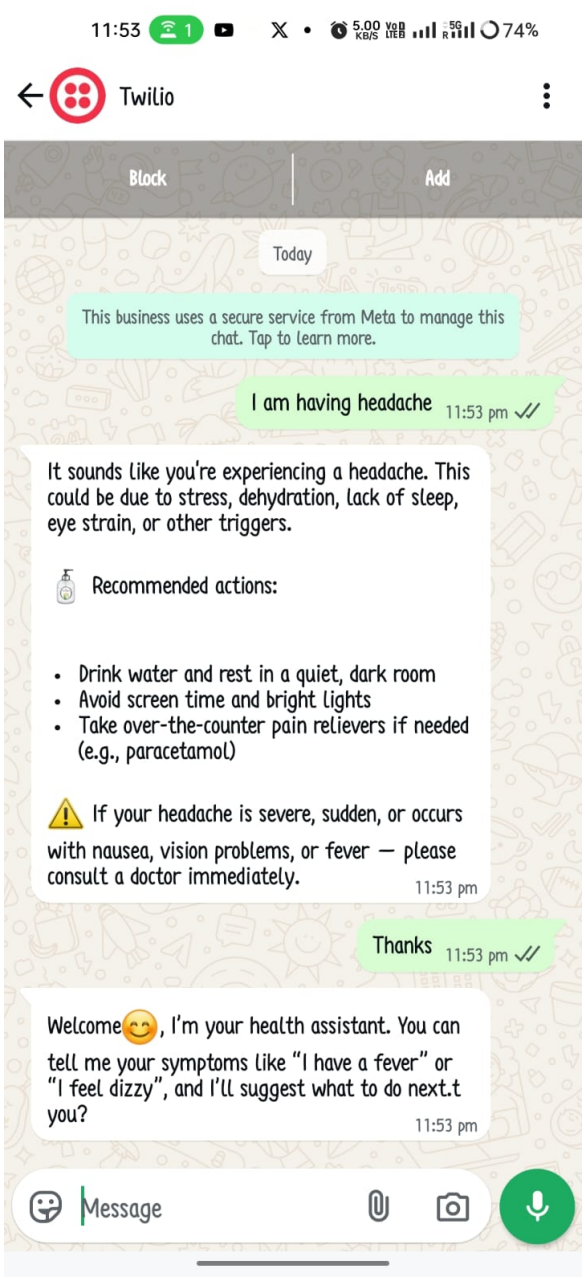

Figure 4: Configuration of the Twilio Sandbox for WhatsApp.

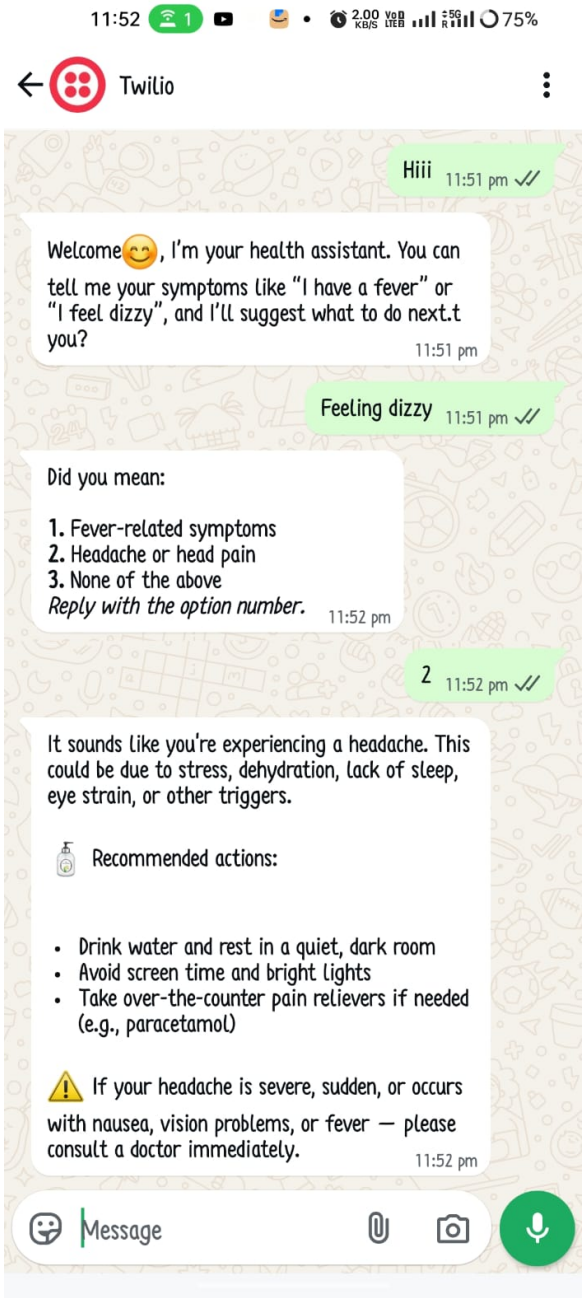

Figure 5: Sandbox number and QR code for connection.

6. Sample Queries and Responses

Users can interact with the HealthBot using simple, natural language. The system is designed to recognize symptoms and provide immediate, actionable advice.

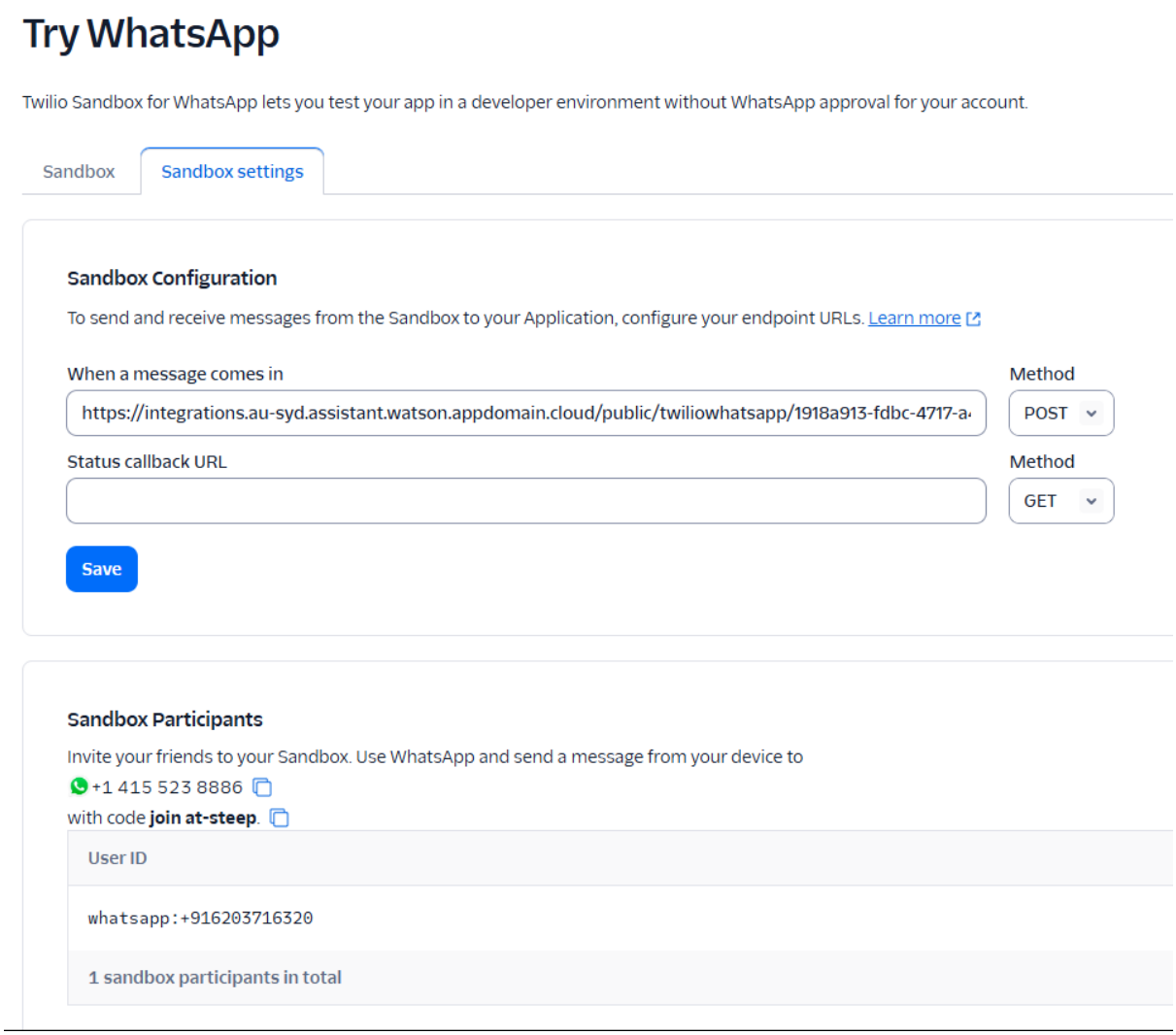

Figure 6: Chatbot Sample Conversation.

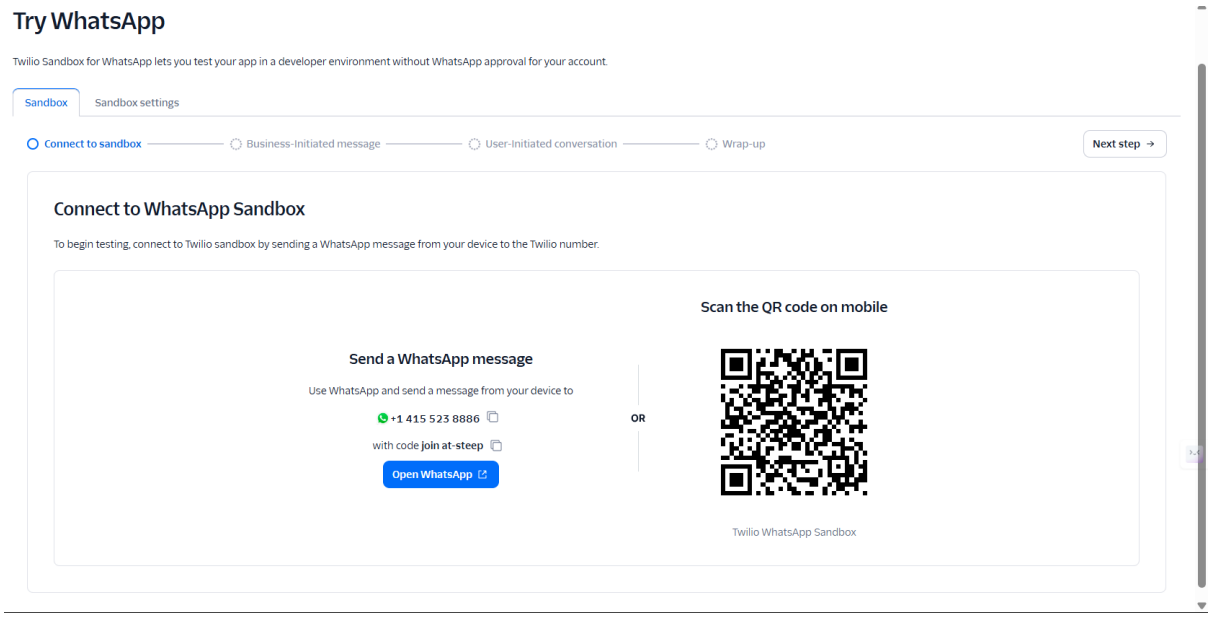

Figure 7: Chatbot Sample Conversation.

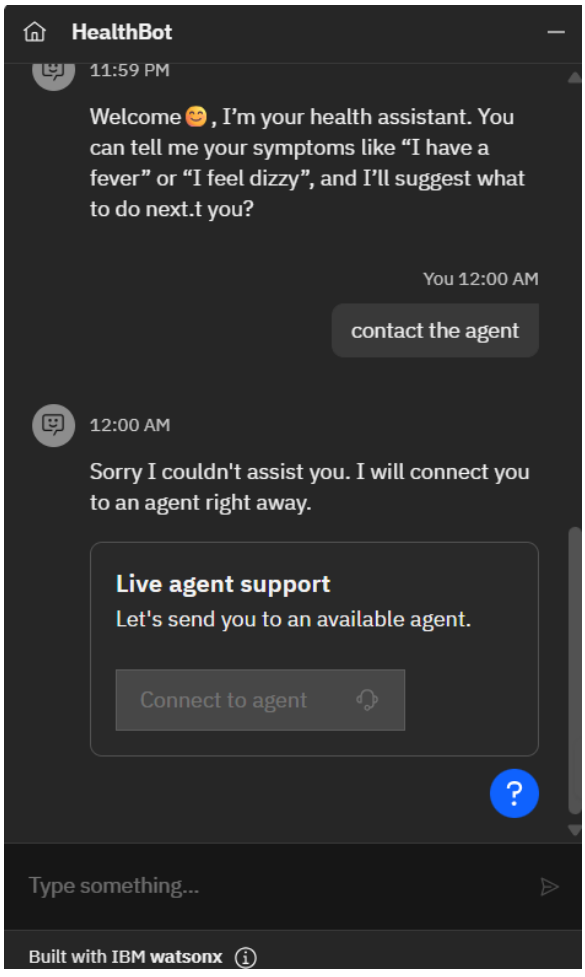

Figure 8: Agent view of a user conversation.

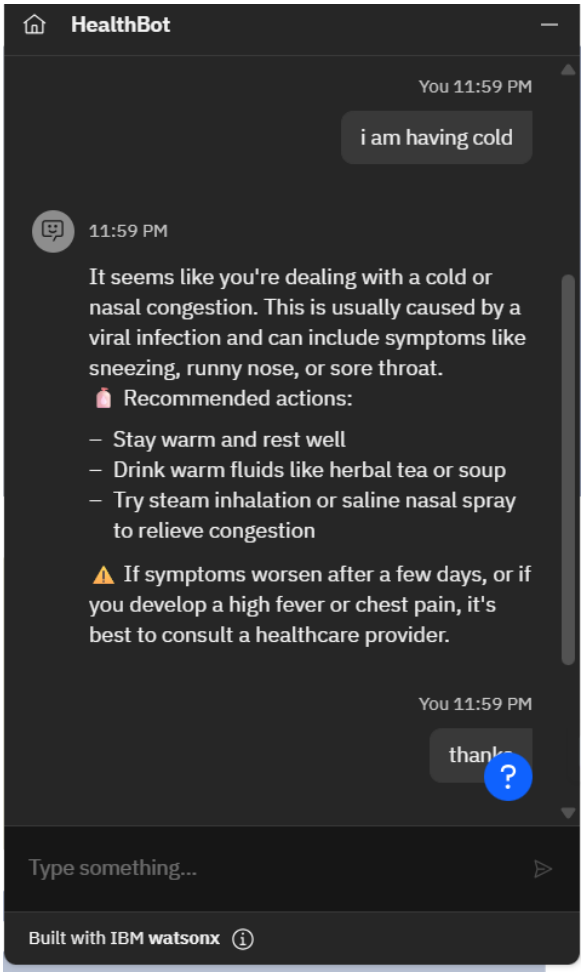

Figure 9: HealthBot detecting symptoms.

7. Limitations

While functional, the project has several limitations inherent to its design as a first-level support tool:

- The advice provided is based on a static, pre-written dataset and does not constitute a real-time medical diagnosis.
- The system does not track user sessions or maintain a medical history for personalized advice.
- Continuous internet connectivity and the availability of all integrated cloud services are required for the system to function.
- The live agent handoff feature is entirely dependent on the proper configuration and availability of the external Twilio Flex service.

8. Conclusion

This project successfully demonstrates the power of combining modern conversational AI platforms like IBM Watson Assistant with custom backend logic and third-party APIs like Twilio. The resulting system is a modular, scalable, and cost-effective solution for providing first-level health advice. Its architecture, built entirely on free-tier cloud services, makes it an excellent template for student projects, rapid prototypes, or pilot programs for health-focused help desks.

9. License