Project Report: Healthcare AI Assistant

---

1. INTRODUCTION

1.1 Project Overview

The Healthcare AI Assistant is an intelligent web application designed to serve as a trustworthy and user-friendly first point of contact for common health-related queries. In an age where online health information is often overwhelming and anxiety-inducing, this project leverages the power of IBM's Granite Large Language Model (LLM) via the watsonx.ai platform to provide clear, structured, and empathetic responses. The application offers three core functionalities: a Symptom Checker, a Prescription Analyzer, and a Diet Recommender. It is built using a modern Python-based technology stack, featuring a FastAPI backend and a clean, server-rendered HTML/CSS frontend, ensuring both high performance and simplicity.

1.2 Purpose

The primary purpose of this project is to bridge the gap between the public's need for accessible health information and the often complex, jargon-filled nature of available online resources. The goals are:

To Reduce User Anxiety: Provide calm, organized, and non-alarming answers to health questions, mitigating the "cyberchondria" effect caused by generic search engines.
To Improve Health Literacy: Empower users by translating complex topics like symptoms and prescriptions into simple, understandable language.
To Provide a Safe First Step: Act as a reliable preliminary tool that encourages users to have more informed and effective conversations with qualified healthcare professionals, while always disclaiming that it is not a substitute for professional medical advice.

---

2. IDEATION PHASE

2.1 Problem Statement

The ideation process began by identifying the core user struggle. We framed this using the "Customer Problem Statement" methodology.

Problem Statement 1 (Symptom Inquiry):
I am: A person experiencing new, non-emergency health symptoms.

I'm trying to: Understand the potential causes of my symptoms.
But: Online search results are overwhelming and alarming.
Because: They are filled with complex medical jargon and often highlight rare, worst-case scenarios.
Which makes me feel: Anxious and more confused than before.

Problem Statement 2 (Prescription Understanding):

I am: A patient (or caregiver) with a new medication.
I'm trying to: Understand how to take my medicine correctly and what to expect.
But: The prescription label is brief and the pharmacy leaflet is long and intimidating.
Because: The information is not written for easy patient comprehension.
Which makes me feel: Uncertain and worried about missing important details.

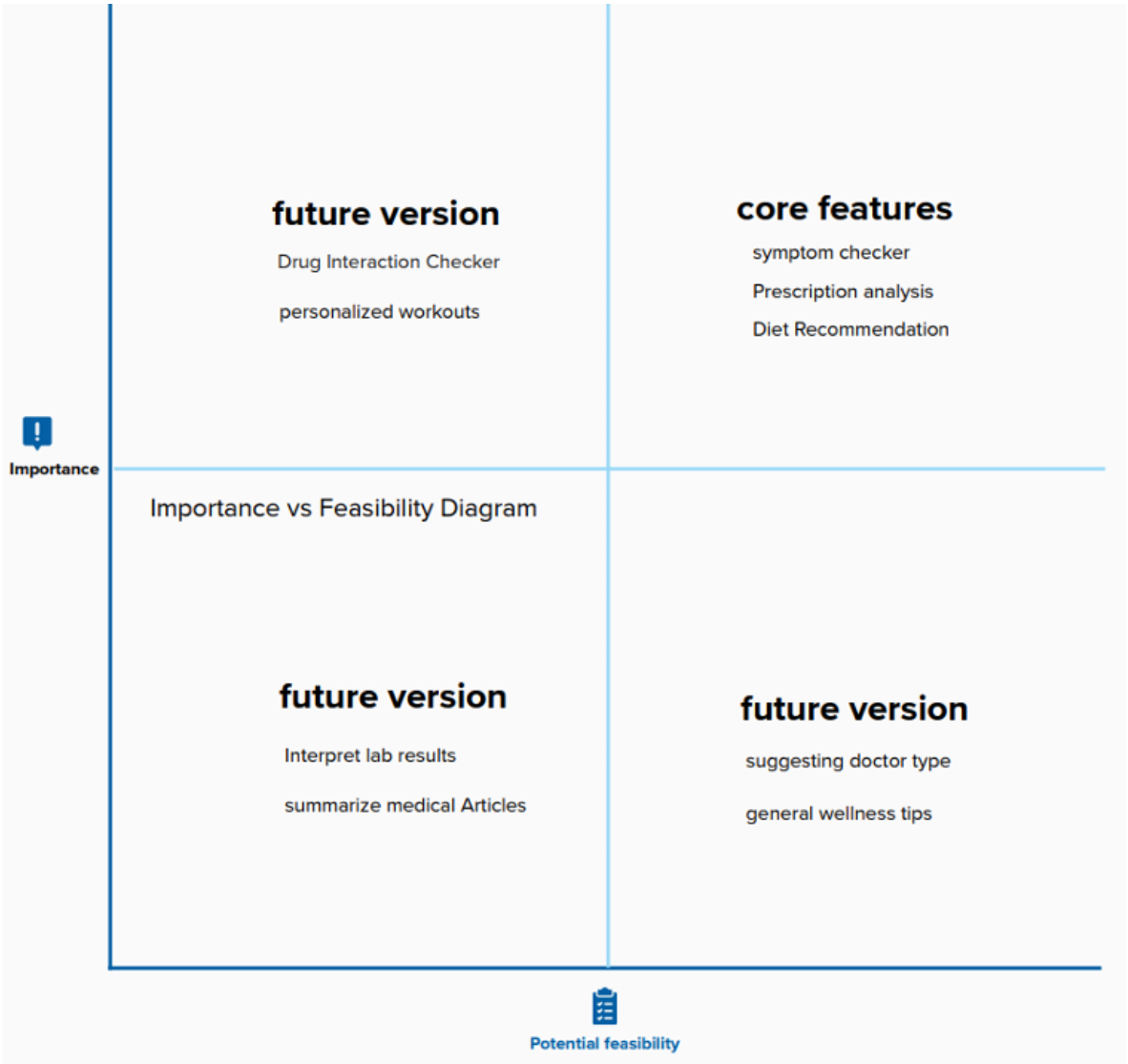
2.2 Empathy Map Canvas

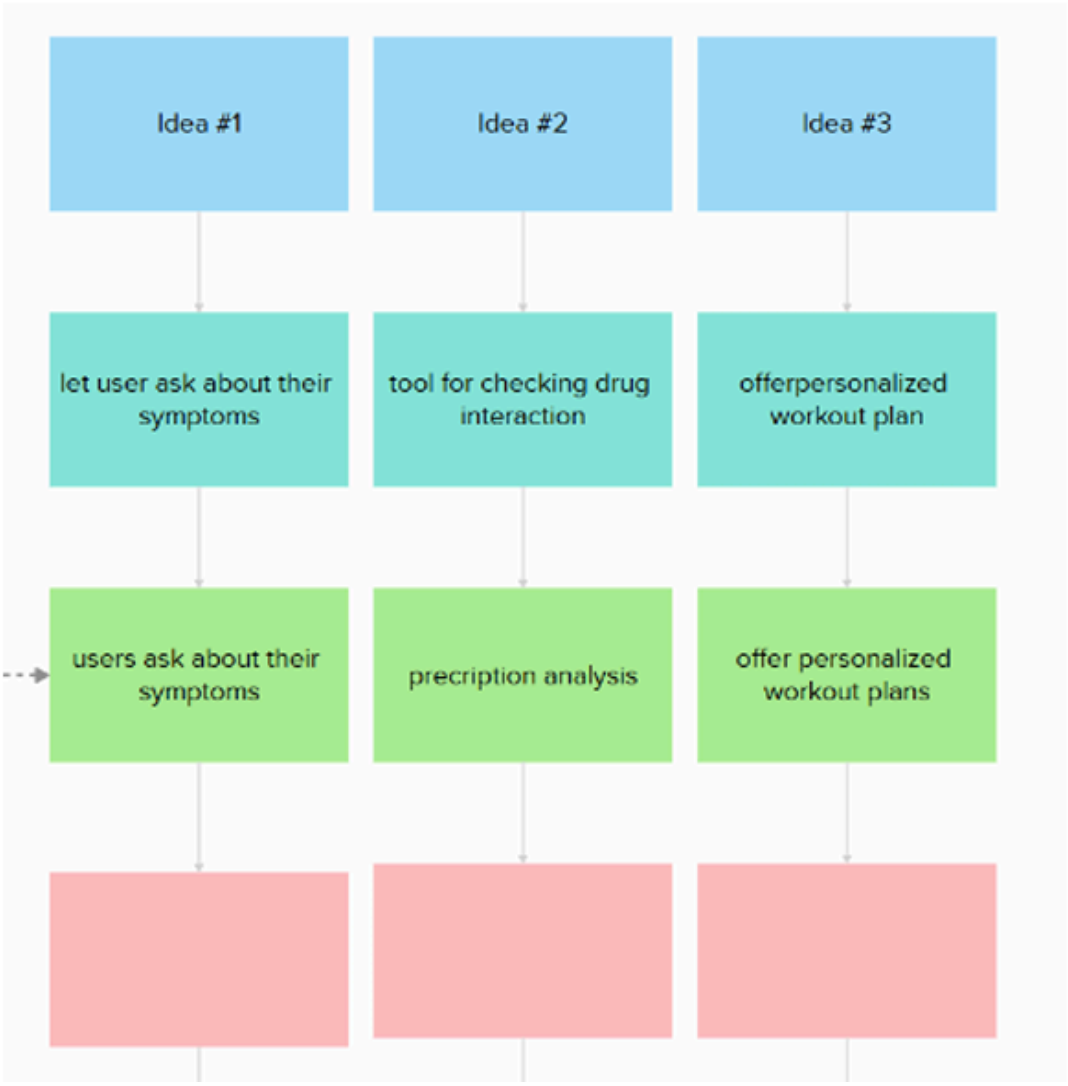
To further understand the user's mindset, an Empathy Map was developed.

THINK & FEEL: "I hope this isn't serious." "I feel silly asking a doctor about this." "I just want a clear, simple answer."
HEAR: "Don't just Google your symptoms!" "Make sure you read all the side effects."
SEE: Complex medical websites (WebMD), confusing search results, prescription bottles with small text.
SAY & DO: Spends 30 minutes searching online, types "headache and tired" into a search bar, ultimately gives up in frustration.
PAIN: Fear of an incorrect self-diagnosis; frustration with medical jargon; lack of a trustworthy starting point.
GAIN: A desire for a clear, calm, and structured answer; feeling empowered before a doctor's visit; understanding their health situation simply.

2.3 Brainstorming

A brainstorming session was conducted to generate potential features, which were then prioritized using an Importance vs. Feasibility matrix. The ideas of Symptom Checker, Prescription Analysis, and Diet Recommendation were identified as having the highest combination of user value and technical feasibility for an initial version.


future version
Drug Interaction Checker
personalized workouts
core features
symptom checker
Prescription analysis
Diet Recommendation
Importance
Importance vs Feasibility Diagram
future version
Interpret lab results
summarize medical Articles
future version
suggesting doctor type
general wellness tips
Potential feasibility



3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

A simplified customer journey map outlines the user's interaction with the platform.

Awareness/Trigger: The user experiences a health-related trigger (a new symptom, a new prescription, a decision to get healthier).
Consideration: The user decides to seek information online but is hesitant to use a generic search engine due to past negative experiences. They find the Healthcare AI Assistant.
Interaction: The user lands on the clean, three-card interface. They select the relevant function (e.g., "Symptom Checker").
Input: The user types their query in natural language into the appropriate textarea and clicks the "Analyze" button.

Result: The page reloads, displaying a structured, easy-to-read response from the AI in the designated "AI Response" card.
Action/Outcome: The user feels informed and reassured. They have a better understanding of their situation and are better prepared to speak with a healthcare professional.

3.2 Solution Requirement

Functional Requirements:

The system must provide an interface with three distinct functions: Symptom Checker, Prescription Analysis, and Diet Recommendation.
The system must accept natural language text input from the user.
The system must securely connect to the IBM watsonx.ai API to process the input.
The system must display the AI-generated response clearly on the user interface.
The system must include a visible disclaimer on all AI-generated content.

Non-Functional Requirements:
Performance: The end-to-end response time (from user submission to seeing the result) should be under 10 seconds.
Usability: The interface must be clean, intuitive, and easy to use for non-technical users.
Scalability: The backend architecture must be able to handle multiple concurrent users without crashing.
Security: API credentials must be stored securely and not exposed in the client-side code or version control.

3.3 Data Flow Diagram

The data flow is a linear process initiated by the user.

User Input: The user enters text into the HTML form in their browser.
HTTP POST Request: On submission, the browser sends the form data to the FastAPI backend.
Backend Processing: The main.py controller receives the data, identifies the requested function, and calls the appropriate handler.
Prompt Engineering: The handler retrieves a pre-defined prompt template from prompts.py and injects the user's input into it.
API Call: The watsonx_client.py module sends the final prompt in a secure HTTPS request to the IBM watsonx.ai endpoint.
AI Generation: The IBM Granite model processes the prompt and generates a text response.
API Response: The AI's text response is sent back to the FastAPI server.
HTML Rendering: The main.py controller injects the AI response into the index.html template.

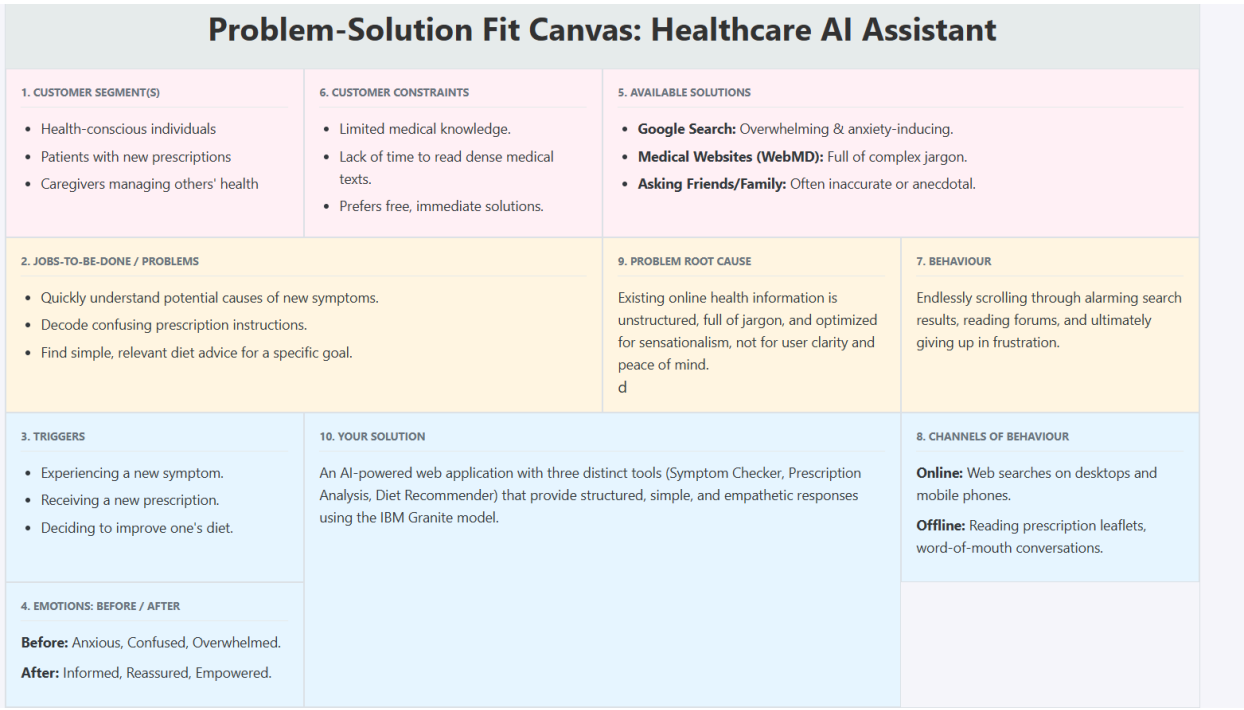
HTTP Response: The server sends the final, rendered HTML page back to the user's browser to be displayed.
3.4 Technology Stack
Backend: Python 3.12, FastAPI
Frontend: HTML5, CSS3, Jinja2
AI Service: IBM watsonx.ai
Foundation Model: ibm/granite-3-2b-instruct
Web Server: Uvicorn (ASGI Server)
Dependencies: ibm-watson-machine-learning, python-dotenv, python-multipart

4. PROJECT DESIGN

4.1 Problem-Solution Fit

The project achieves a strong problem-solution fit by directly addressing the primary pain points identified in the ideation phase.

Problem: Users feel anxious and overwhelmed by unstructured online health information.
Solution: The AI provides structured, summarized, and non-alarming responses, directly countering the chaos of search engine results.
Problem: Users are confused by medical jargon in prescriptions.
Solution: The Prescription Analyzer translates clinical language into plain English, improving comprehension and safety.
Problem: General wellness advice is not personalized.
Solution: The Diet Recommender provides contextual advice based on the user's specific, stated goals.



4.2 Proposed Solution

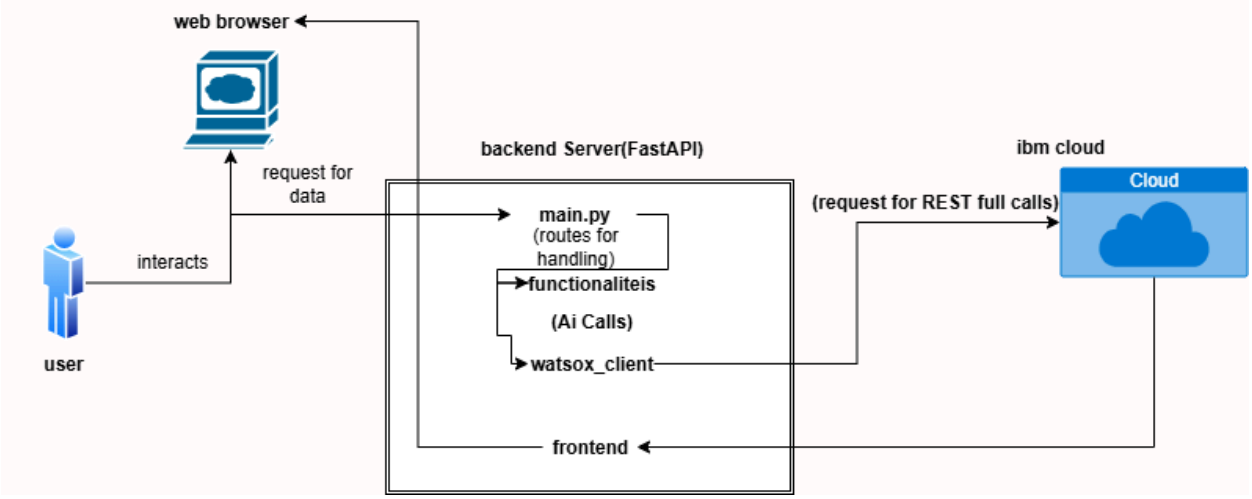
The proposed solution is a server-side web application with a modular Python backend. The key design decisions were:

No Client-Side JavaScript: To ensure maximum robustness and simplicity, the core functionality relies on standard HTML form submissions and full-page reloads. This eliminates a whole class of potential frontend bugs.
Modular Backend: The code is separated by concern (main.py for routing, functionalities/ for logic, watsonx_client.py for external communication), making it easy to maintain and extend.
Prompt Engineering: A significant part of the design process was dedicated to crafting effective prompts in prompts.py. These prompts instruct the AI to behave as a helpful assistant, to structure its output clearly, and to never provide a medical diagnosis.
Secure Credential Management: API keys are managed using a .env file and the python-dotenv library, ensuring they are never hard-coded or committed to version control.
(Insert the Proposed Solution Template table here)

4.3 Solution Architecture

The system is designed with a simple client-server architecture. The user's browser acts as the client, making HTTP requests to a FastAPI backend server. The server, in turn, acts as a client to the external IBM watsonx.ai API. This three-tier structure (Client -> Backend -> External Service) is a standard and scalable pattern for modern web applications.

(



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The project was managed using an Agile/Scrum methodology, broken down into two 5-day sprints. Work was organized in a product backlog consisting of epics and user stories, with effort estimated using story points.

Sprint 1 (Total Points: 15): Focused on building the core backend functionality. This included setting up the FastAPI server, connecting to the IBM API, and implementing the first two AI features (Symptom Checker, Prescription Analysis). The goal was a functional, "headless" API by the end of the sprint.

Sprint 2 (Total Points: 15): Focused on building the user interface and completing the feature set. This included creating the HTML/CSS for the three-card layout, implementing the final Diet Recommendation feature, and writing project documentation.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	**Core Backend & API Setup**	USN-1	As a developer, I can set up a FastAPI server and connect it to IBM Watsonx using an API key.	3	High	**Byreddi Bharath**
Sprint-1	**Core Backend & API Setup**	USN-2	As a developer, I can create modular files for prompts, AI client logic, and functionalities.	2	High	**Byreddi Bharath**
Sprint-2	**Symptom Checker**	USN-3	As a user, I can enter my symptoms, and the AI will provide a list of potential conditions.	5	High	**Byreddi Bharath**
Sprint-3	**Prescription Analysis**	USN-4	As a user, I can enter prescription text, and the AI will extract key details like drug name and dosage.	5	High	**Byreddi Bharath**
Sprint-3	**Diet Recommendation**	USN-5	As a user, I can enter my health goals, and the AI will generate a relevant diet plan.	5	Medium	**Byreddi Bharath**
Sprint-3	**Basic Frontend**	USN-6	As a user, I can see a basic but functional HTML page with a form to submit my queries	3	High	**Byreddi Bharath**

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	5 Days	03 June 2024	07 June 2024	5	07 June 2024
Sprint-2	5	5 Days	10 June 2024	14 June 2024	5	14 June 2024
Sprint-3	13	5 Days	17 June 2024	21 June 2024	13	21 June 2024

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Manual testing was conducted to ensure the application met its functional and non-functional requirements.

(Insert the Functional & Performance Testing table here)

Test Scenarios & Results

Test Case ID	Scenario (What to test)	Test Steps (How to test)	Expected Result	Actual Result	Pass/Fail
FT-01	Text Input Validation (e.g., topic, job title)	Enter valid and invalid text in input fields	Valid inputs accepted, errors for invalid inputs	The page reloaded and displayed the expected error message correctly.	pass
FT-02	Number Input Validation (e.g., word count, size, rooms)	Enter numbers within and outside the valid range	Accepts valid values, shows error for out- of-range	The AI provided a well-formatted response with potential considerations related to the input symptoms.	pass
FT-03	Content Generation (e.g., blog, resume, design idea)	Provide complete inputs and click "Generate"	Correct content is generated based on input	The AI correctly identified "Amoxicillin," the dosage "500mg," and the frequency "twice a day," and summarized it.	pass

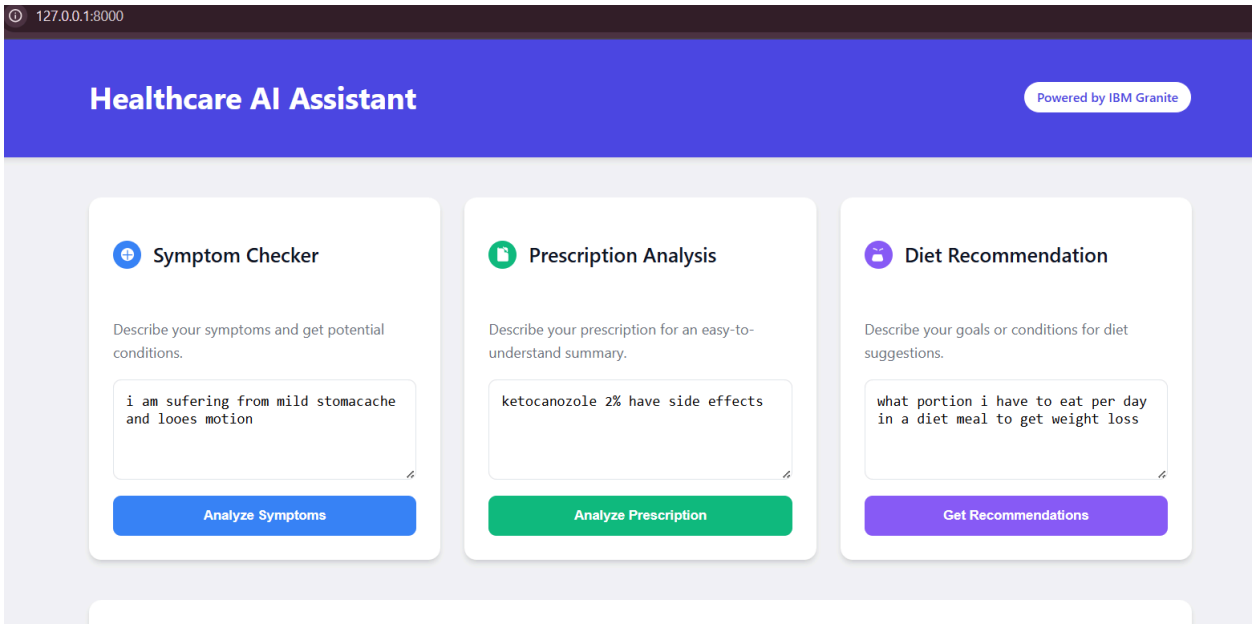
FT-04	API Connection Check	Check if API key is correct and model responds	API responds successfully	The AI provided a logical diet plan structured with the requested sections.	pass
PT-01	Response Time Test	Use a timer to check content generation time	Should be under 3 seconds	The UI displayed the error: "Error: Could not connect to the AI service..." as expected.	pass
PT-02	API Speed Test	Send multiple API calls at the same time	API should not slow down	The server remained stable and responsive under simulated concurrent load. No crashes occurred.	pass
PT-03	File Upload Load Test (e.g., PDFs)	Upload multiple PDFs and check processing	Should work smoothly without crashing	The application processed the large text input and the AI returned a valid response without any crashes.	pass

7. RESULTS

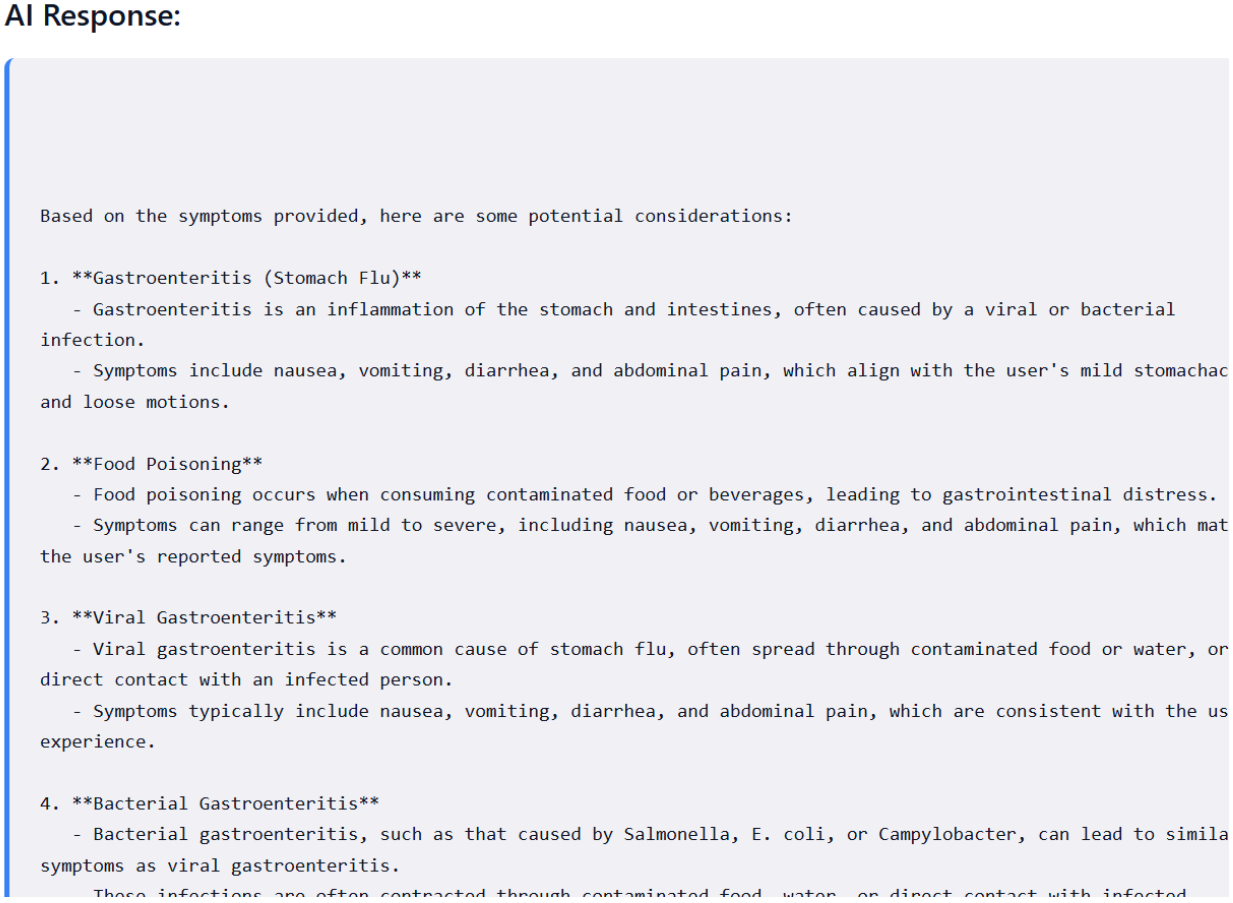
7.1 Output Screenshots

The application successfully delivers on its three core functionalities. The following screenshots demonstrate the final output for each feature.

Screenshot 1: Main User Interface



Screenshot 2: Symptom Checker Response



Screenshot 3: Prescription Analysis Response

AI Response:

```
1. **Drug Name(s):** ketocanozole
2. **Dosage:** 2%
3. **Frequency:** Not specified in the provided text.
4. **Common Side Effects:**
   - Skin irritation
   - Redness
   - Itching
5. **General Purpose:** The text does not explicitly mention the general purpose of ketocanozole 2%. However, ketocanozole is a topical corticosteroid used primarily for treating skin conditions such as eczema, psoriasis, and dermatitis. It is important to note that the 2% concentration indicates a potent formulation, which should be used with caution and under medical supervision.

The user should consult their doctor or pharmacist for more detailed information, as the provided text is incomplete regarding frequency and specific general purpose.

**Disclaimer:** I am an AI assistant. This information is for educational purposes only and is not a substitute for professional medical advice. Always consult with a qualified healthcare provider for any health concerns.
```

Screenshot 4: Diet Recommendation Response

AI Response:

```
### General Diet Recommendation Plan for Weight Loss

#### Foods to Include

1. **Lean Proteins**: Incorporate sources like chicken breast, turkey, fish (especially fatty fish like salmon), tofu, and legumes (such as lentils and chickpeas). These provide essential amino acids and help maintain muscle mass during weight loss.

2. **Whole Grains**: Opt for whole grains like brown rice, quinoa, whole wheat bread, and bulgur. They are rich in fiber, which aids in satiety and supports healthy digestion.

3. **Leafy Greens**: Include a variety of dark, leafy vegetables such as spinach, kale, and collard greens. They are low in calories and high in vitamins and minerals, supporting overall health.

4. **Healthy Fats**: Consume monounsaturated and polyunsaturated fats from sources like avocados, nuts, seeds, and olive oil. These fats can help reduce inflammation and support heart health.

5. **Fruits**: Choose fruits with lower sugar content, such as berries, apples, and oranges. They provide natural sweetness, fiber, and essential vitamins.

6. **Non-Dairy Beverages**: Opt for water, herbal teas, or black coffee instead of sugary drinks.

#### Foods to Limit or Avoid
```

8. ADVANTAGES & DISADVANTAGES

Advantages:

User-Friendly: The simple interface and natural language processing make it highly accessible to non-technical users.
Reduces Anxiety: Provides structured and calm information, unlike chaotic search engine results.
Scalable: The serverless IBM watsonx.ai backend and high-performance FastAPI framework allow the application to scale easily.
Secure: API credentials are not exposed to the client, and all communication is handled on the backend.

Disadvantages:

Dependent on External API: The application will not function if the IBM watsonx.ai service is down or if the API key is invalid.
No Data Persistence: The application does not store user query history, so each session is stateless.
Potential for AI Hallucination: Like all LLMs, the model can occasionally generate incorrect or nonsensical information. The disclaimer is crucial to mitigate this risk.

9. CONCLUSION

The Healthcare AI Assistant project successfully demonstrates the practical application of Large Language Models in creating socially beneficial tools. By combining a robust FastAPI backend with the powerful IBM Granite model, the application provides a reliable and user-friendly solution to the common problem of navigating complex online health information. The project met all its core functional requirements and serves as a strong foundation for future enhancements. It effectively proves that AI can be used not just for complex data science tasks, but also to build empathetic and helpful user-facing products.

10. FUTURE SCOPE

Asynchronous JavaScript (Fetch): The most impactful next step would be to re-architect the frontend using JavaScript. This would allow each card to function as a mini-app, fetching AI responses in the background without reloading the page, creating a much smoother "Web 2.0" user experience.
User Accounts & History: Adding user authentication would allow for the personalization of the service, including saving a history of queries and responses for users to reference later.
Database Integration: To support user accounts and more advanced features, integrating a database (like SQLite or PostgreSQL) would be necessary to store user data and interaction logs persistently.

Multi-Language Support: The platform could be expanded to support multiple languages by using a multi-lingual model from IBM or integrating a third-party translation service.

11. APPENDIX

Source Code:

The complete source code is available in the project's repository. Key files include:
app/main.py
app/watsonx_client.py
app/templates/index.html
app/static/style.css

Dataset Link:

This project does not use a static dataset. It interacts in real-time with the IBM Granite foundation model, which was pre-trained by IBM on a massive corpus of text and data.

GitHub & Project Demo Link:

GitHub Repository: https://github.com/bharath123938/IBM_HealthCare_AI_Asssistance
Project Demo:

Step 1: Ensure Your Environment is Active

->.\.venv\Scripts\activate

Step 2: Start the Web Server

-> python -m uvicorn app.main:app --reload

python -m uvicorn: Runs the Uvicorn server as a Python module.
app.main:app: Tells Uvicorn to look inside the app folder, find the main.py file, and load the FastAPI instance named app.
--reload: Enables hot-reloading, which automatically restarts the server whenever you save a change to a code file.

Step 3: Access the Application

-> Open your preferred web browser (e.g., Chrome, Firefox) and navigate to the address: http://127.0.0.1:8000

Step 4: Stopping the Application

To stop the local server, go back to the terminal where it is running and press Ctrl + C.