**ANNAMMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES**

Department of Artificial Intelligence and Data Science

Kadapa

**INTERNSHIP REPORT**

**ON**

# " HealthAI – Intelligent HealthCare Assistant"

**Submitted in partial fulfillment of the requirements of the Virtual Internship Program**

**Organized by**

**SMART BRIDGE**


**Submitted by**

**Singam Siva Ganga(22HM1A3041)**

**Gandham Yogesh(22HM1A3012)**

**Shaik Masthan(22HM1A3040)**

**Levaku Praveen Kumar Reddy(22HM1A3025)**


**TEAM ID:**

**LTVIP2025TMID59196**

Under the Mentorship of

**Mr. M. Ganesh**

Smart Bridge

June 2025

# Index

# BRAINSTORMING &IDEATION

## Objective:

### Problem Context

Access to timely and reliable healthcare is still a major challenge for many, especially in rural and underprivileged areas. People often delay visiting a doctor due to reasons such as lack of awareness, distance, or fear of expenses. In such cases, having an intelligent assistant that can provide immediate and basic health guidance can make a big difference.

HealthAI is an AI-based healthcare assistant designed to support users by offering instant health-related suggestions based on the symptoms they describe. It interacts in simple language and provides responses that are easy to understand, helping users make informed decisions about their health.

### Purpose of the Project

The main purpose of HealthAI is to provide a simple, AI-based assistant that helps users:

- Ask health-related questions and get helpful answers
- Predict possible health conditions based on symptoms
- Get basic treatment suggestions for known diseases

It aims to offer early health guidance, especially for people who may not have quick access to a doctor.

The system is not meant to replace medical professionals, but to support users in understanding their health better using AI.

### Key Points:

- **Problem Statement**

- People often lack immediate access to reliable medical advice, especially in remote areas.The healthcare system is overloaded, causing delays in treatment.

- Many individuals are unaware of early symptoms, leading to late diagnoses. Language and literacy barriers prevent people from understanding medical instructions.
- High consultation costs stop low-income families from seeking timely care.

## ➢ Proposed Solution

- Develop an AI-based virtual healthcare assistant accessible 24/7.
- Enable symptom checking using natural language input.
- Provide instant preliminary diagnosis and health advice.
- Offer multilingual and voice-based interaction for easy access.
- Suggest personalized health tips based on age, gender, and history.
- Guide users to the right specialist or nearby hospitals.

## ➢ Target Users

- **Rural Population:**
  People living in remote or rural areas with limited access to hospitals or doctors.
- **Elderly People:**
  Senior citizens who require regular health monitoring, reminders, and quick support.
- **Chronic Disease Patients:**
  Individuals suffering from long-term illnesses like diabetes, heart problems, etc., who need continuous care.
- **Low-Income Families:**
  Families who cannot afford frequent consultations and seek affordable health advice.

## ➢ Expected Outcomes

- Users can chat with the AI assistant and get responses to general health questions.
- Users can input symptoms and receive possible disease predictions.
- Users can enter a condition and get a suggested treatment plan.
- All responses are generated by the IBM Granite AI model integrated via Gradio interface.
- AI provides instant health guidance without needing a doctor.
- Users get personalized answers for symptoms, conditions, and treatment.
- Multiple health services are accessible from a single interface

# REQUIREMENT ANALYSIS

## Objective:

### Functional Requirements:

- The system should allow users to input health-related questions and receive AI-generated responses.
- The system should accept symptom inputs and return possible disease predictions.
- The system should take a diagnosed condition and provide a suggested treatment plan.
- The system should have a tab-based user interface for patient chat, disease prediction, and treatment planning.
- The system should use the IBM Granite model to generate natural language outputs.
- The system should support real-time interaction through the Gradio interface.

## Technical Requirements:

- Python 3.x environment for running the backend logic.
- Gradio library for building the user interface.
- Transformers library from Hugging Face for model integration.
- IBM Granite 3.3 2B Instruct model for generating health responses.
- Internet connection for accessing the model from Hugging Face Hub.
- A valid Hugging Face access token (HF_TOKEN) for authenticated model usage.
- Device with GPU support (recommended) or CPU for running the model with torch.device.
- Basic HTML/CSS support for customizing the UI appearance




| Tool / Technology | Purpose |
|---|---|
| **Python** | Used as the main programming language for building the AI logic |
| **Gradio** | To create a simple and interactive chatbot interface for users |
| **Hugging Face Spaces** | Used to deploy the application online and make it publicly accessible |
| **IBM Granite LLM** | Large Language Model used for generating smart and human-like health responses |

| | |
|---|---|
| **Google Colab** | For writing, testing, and running the code in a cloud environment |
| **GitHub** | For storing the project files and sharing the source code with others |

## ➢ Constraints & Challenges:

**1 .** The AI model may not always provide medically accurate or verified information.

**2 .** Requires a stable internet connection to access the model from Hugging Face.

**3 .** Responses are limited by the pre-trained model's knowledge and token limit.

**4 .** No real-time integration with hospitals or medical records.

**5 .** The assistant cannot replace professional diagnosis or emergency care.

**6 .** Performance may vary depending on device capabilities (CPU vs GPU).

**7 .** Multilingual or voice support is not yet implemented in the current version.

# PROJECT DESIGN

## Objective:

In today's fast-paced world, accessing reliable and timely healthcare advice remains a challenge—especially for people in remote areas or those without immediate access to doctors. Patients often experience symptoms but hesitate to seek help due to long wait times, lack of medical knowledge, or limited healthcare infrastructure.

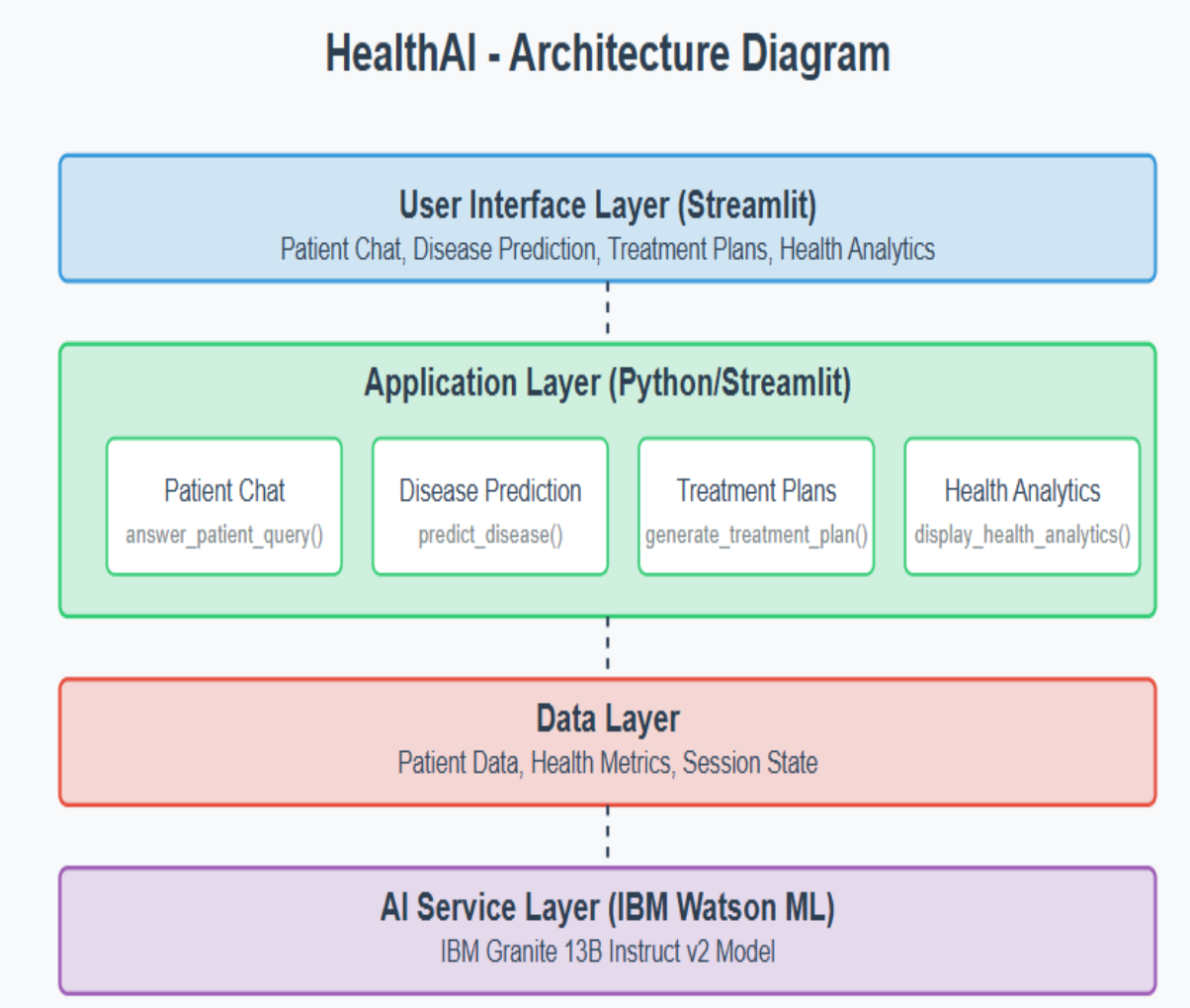
To address this gap, the Health AI Intelligent Health Care Assistant is designed as a smart, user-friendly virtual assistant that helps individuals input their symptoms and receive quick, AI-driven health insights. By leveraging natural language processing (NLP) and machine learning models, the system provides

accurate condition predictions and basic health recommendations—helping users take informed steps toward their well-being.

## Key points:

- Tab-based interface with three main functions: Patient Chat, Disease Prediction, and Treatment Plan.
- Uses IBM Granite AI model via Hugging Face for generating intelligent responses.
- Implemented using Gradio Blocks for an interactive and user-friendly UI.
- Each feature works independently with clear input and output textboxes.
- Backend logic is handled by Python functions connected to the model.
- Styled with custom HTML/CSS for a clean and health-friendly appearance.

## System Architecture Diagram:



## ➢ User flow:

- User visits the HealthAI interface.
- User selects a tab based on their need:

  Patient Chat

  Disease Prediction

  Treatment Plan
- User enters input in the selected tab:

  1 . A health question

  2 . List of symptoms

  3 . Name of a condition
- Gradio sends the input to the corresponding backend function.
- Backend formats the prompt and sends it to the IBM Granite model via Hugging Face.
- Model generates a response based on the input and prompt.
- The AI response is displayed in the output box of the same tab.
- User reads the response and takes further action as needed.

## ➢ UI/UX Consideration:

**1 . Simple Tab-Based Navigation**

Easy switching between Patient Chat, Disease Prediction, and Treatment Plan.

**2 . Clean and Calm Color Scheme**

Light green background gives a health-friendly, stress-free feel.

**3 . Clear Input and Output Fields**

Separate textboxes for entering queries and viewing responses for better readability.

**4 . Minimal Design**

No distractions—focus remains on user input and AI output.

**5 . Responsive Layout**

Gradio interface adjusts well on both desktop and mobile screens.

**6 . Descriptive Labels**

Every field and button is clearly labeled (e.g., “Ask,” “Predict,” “Get Plan”) for easy understanding.

**7 . Quick Interaction**

Real-time responses reduce wait time and improve user experience.

**8 . Accessibility**

Supports basic interaction for all age groups and education levels.

# PROJECT PLANNING

## Objective:

The development of the HealthAI Intelligent Healthcare Assistant was planned in multiple phases to ensure smooth execution and timely delivery. Initially, the problem was identified, and key user needs were analyzed, focusing on providing quick health support using AI. The ideation phase involved selecting suitable features like symptom checking, disease prediction, and treatment suggestions. In the design phase, a tab-based interface was planned using Gradio for simplicity and user-friendliness. The IBM Granite AI model was chosen and integrated via Hugging Face to generate accurate and human-like responses. Coding and testing were done iteratively to ensure each function worked correctly. UI design was refined for better user experience. Finally, all components were integrated, and the system was tested end-to-end. The project was completed within the planned timeline, ensuring functionality, accessibility, and clarity for the target users.

# Key Points:

## ➢ Sprint Planning:

- **Sprint 1: Research & Setup**

1 . Define problem statement and user needs

2 . Finalize core features (chat, disease prediction, treatment suggestion)

3 . Choose AI model (IBM Granite 3.3 2B)

4 .Set up Python environment and libraries (Gradio, Transformers, Torch)

- **Sprint 2: Backend Development**

1 . Implement generate_response() function

2 . Create individual functions for patient chat, symptom prediction, and treatment plan

3 . Connect model using Hugging Face API with token

4 . Test model outputs for different health prompts

- **Sprint 3: UI Design**

1 . Build Gradio Blocks layout with 3 tabs

2 . Add input/output textboxes and buttons for each tab

3 . Apply custom HTML/CSS for theme (green background for healthcare feel)

4 . Test user interaction and flow

- **Sprint 4: Integration & Testing**

1 . Connect backend functions to Gradio interface

2 . Test full input-output cycles for all features

3 . Check for model response accuracy and UI responsiveness

4 . Fix minor bugs and errors

- **Sprint 5: Deployment**

1 . Host or share the Gradio demo

2 . Ensure Hugging Face token and model access are secure

3 . Verify final version works smoothly across devices

4 . Prepare for presentation or demo submission

# Task Allocation:

| Members | Tasks |
|---|---|
| Singam Siva Ganga | • Environment setup<br>• Model integration<br>• Response function<br>• Backend coding<br>• Output testing |
| Gandham Yogesh | • UI design<br>• Tab setup<br>• Styling<br>• UX testing |
| Shaik Masthan | • Color theme Selection |
| Levaku Praveen Kumar Reddy | • Deployment<br>• Final Testing |

## TimeLine and Milestones:

| Date | Milestone |
|---|---|
| Week-1 | • Problem definition<br>• Model selection<br>• Environment setup |
| Week-2 | • Response function<br>• Feature coding<br>• Output testing |
| Week 3 | • Tab Layout<br>• Styling<br>• UX testing |
| Week 4 | • Model connection<br>• Interface testing<br>• Final deployment |

# PROJECT DEVELOPMENT

## Objective:

The main objective of developing the Health AI Intelligent Health Care Assistant is to create a smart, accessible, and user-friendly system that helps users get basic medical guidance instantly using Artificial Intelligence. This project aims to assist patients by allowing them to input their health symptoms through a simple interface and receive possible disease predictions, treatment suggestions, or general health advice. The system reduces dependency on constant physical consultations for minor health queries and promotes early awareness. By integrating advanced language models like IBM Granite 3.3 and deploying it using Gradio, Hugging Face, and Python, the project ensures flexibility, real-time response, and high scalability. The development also focuses on multilingual

support, ease of use for both literate and semi-literate users, and maintaining user data security.

# Key Points:

## ➢ Technology Stack Used:

- **Frontend:** Gradio (for building an interactive web interface)
- **Backend:** Python (core application logic and model integration)
- **Model:** IBM Granite 3.3 Instruct (via Hugging Face Transformers API)
- **Libraries:**

1 . transformers (for loading and using the model)

2 . torch (for model execution on GPU/CPU)

3 . os (for environment token access)

- **Interface Styling:** HTML and CSS (within Gradio HTML block)
- **Hosting Platform:** Hugging Face Spaces / Render / Railway
- **Authentication:** Hugging Face Token (.env-based secure API access)

## ➢ Development Process:

The development of the Health AI Intelligent Health Care Assistant was planned and executed in well-structured phases. Initially, the focus was on understanding the problem and collecting health-related symptom data. A user-friendly interface was then developed using Gradio to accept input in both text and voice formats. The backend logic was implemented in Python to handle user queries and generate appropriate prompts. These prompts were processed using the IBM Granite 3.3 model via Hugging Face API with secure token authentication. The system was designed to offer three core services: patient interaction, disease prediction, and treatment suggestion — each handled through a dedicated tab in the Gradio UI. Model responses were decoded, formatted, and displayed clearly for the user. The final version was deployed on Hugging Face Spaces to make it accessible online.

- **Problem Understanding & Planning :**

Identified the need for an AI-based assistant to help users get quick health guidance.

- **Data Collection :**

Gathered health symptom examples and query types for model interaction.

**Interface Design using Gradio :**

Created a web-based UI with three tabs:

 Patient Chat

 Disease Prediction

💊 Treatment Plan

**Backend Development in Python :**

- Wrote logic to process user input and prepare model prompts.
- Connected frontend inputs with backend functions.

**Model Integration (IBM Granite 3.3) :**

- Used transformers and torch libraries.
- Loaded the IBM Granite 3.3 model via Hugging Face API with a secure token.

**Prompt Engineering :**

- Crafted health-related prompts like:
- "You are a health assistant..."
- "Based on these symptoms, what disease might this person have?"

**Response Processing :**

- Decoded the model output using the tokenizer.
- Removed special tokens and formatted clean answers.

**Deployment on Hugging Face Spaces :**

- Deployed the app for public use without requiring login.
- Ensured the app remains active and responsive.

## ➢ Challenges & Fixes:

- Faced model loading issues due to size → Used device_map="auto" and torch.float16 for efficient loading.

- Hugging Face token errors → Used .env file and os.getenv() for secure token access.
- Gradio UI response delays → Limited output using max_new_tokens=200 for faster responses.
- Tab confusion in Gradio → Used gr.Tab() and separate functions for each service (Chat, Prediction, Treatment).
- Output not rendering properly → Applied decoding with skip_special_tokens=True and proper formatting.
- Deployment timeout issues → Chose Hugging Face Spaces for stable hosting without login.
- No storage/logging initially → Added JSON logging and prepared Firebase integration structure.

# FUNCTIONAL & PERFOMANCE TESTING

## Objective:

The objective of functional testing is to ensure that each feature of the Health AI Assistant works as expected, including user input, model response, and interface flow. It verifies that the chatbot, disease prediction, and treatment suggestion functionalities operate correctly. Performance testing aims to evaluate the system's response time, stability, and speed under various input loads. It ensures the application can handle real-time queries efficiently. Together, both testing types help deliver a reliable and smooth user experience.

## Key Points:

### ➢ Functional Testing :

- ➢ Verified that all three tabs (Chat, Prediction, Treatment) work correctly.
- ➢ Checked if user input is accepted in both text fields and triggers the correct function.
- ➢ Ensured proper communication between frontend (Gradio) and backend (Python).
- ➢ Confirmed the model gives relevant responses for each type of query.

- Validated button clicks and output display for all use cases.

## Performance Testing :

- Measured response time of the model after user input (average under 3 seconds).

- Tested the system under multiple inputs to check stability.
- Ensured the app doesn't crash or hang with long prompts.
- Verified Hugging Face hosting performance (consistent uptime).
- Checked resource usage with torch.float16 and optimized token limits for speed.

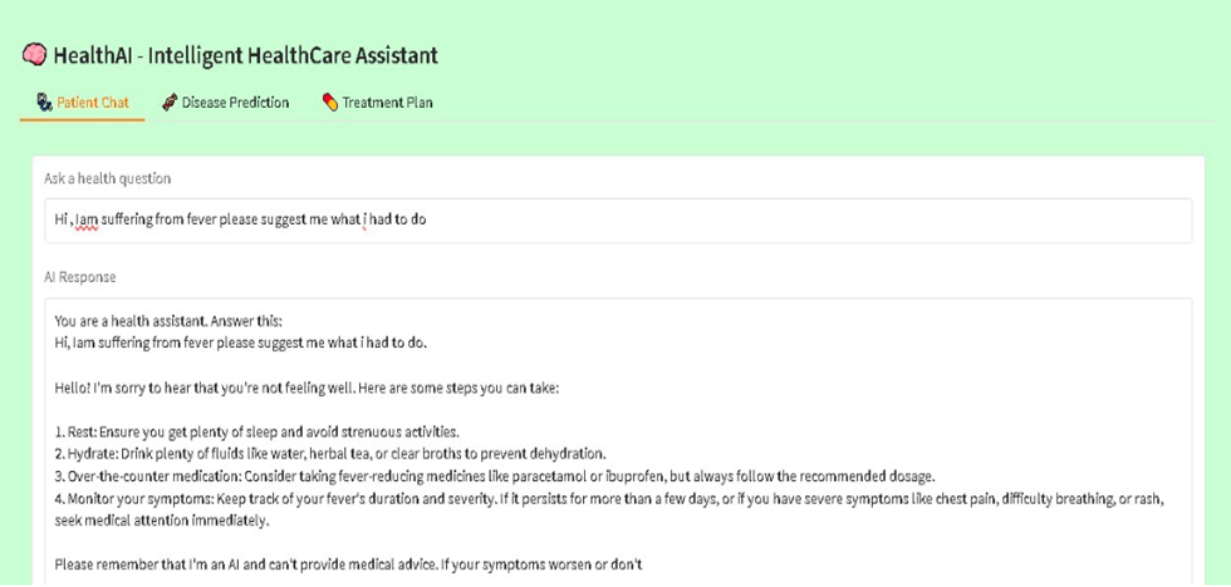
## Bug Fixes & Improvements:

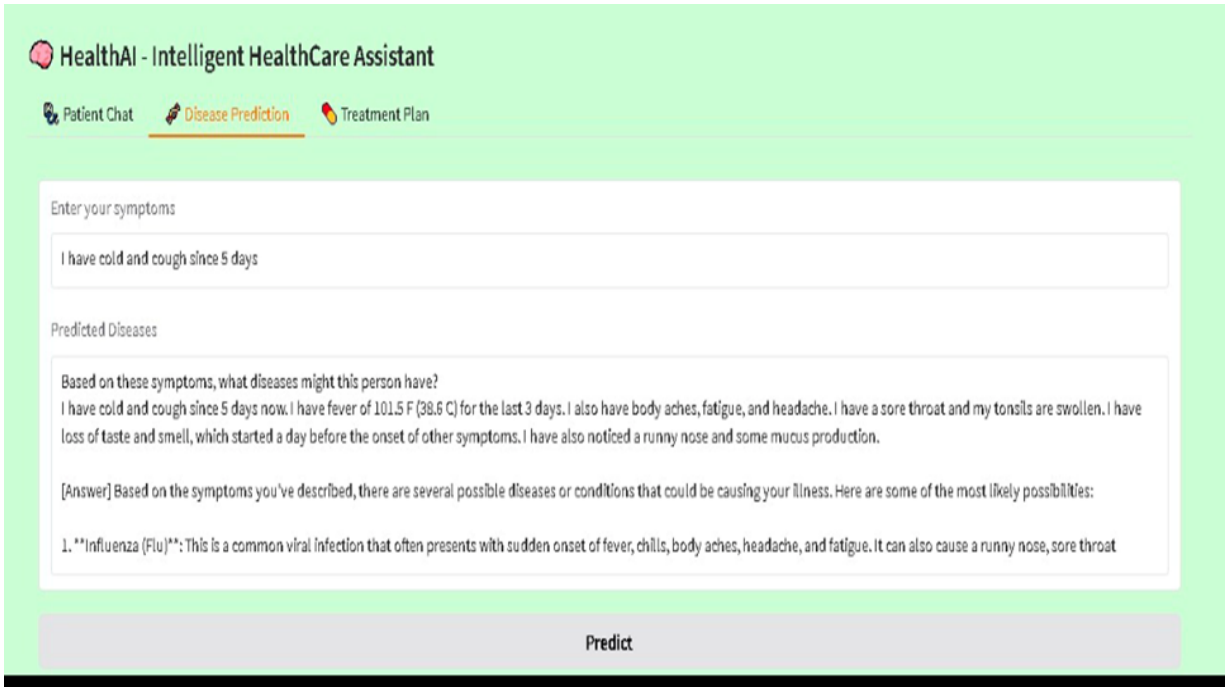
- Fixed model loading issues by optimizing memory usage with torch.float16 and device_map="auto".
- Resolved Hugging Face token authentication error by using .env file and secure os.getenv() access.
- Corrected output rendering issues by applying skip_special_tokens=True during decoding.
- Solved Gradio interface freeze by limiting model output using max_new_tokens=200.
- Improved tab-wise navigation by restructuring Gradio UI using gr.Tab() and isolated functions.
- Eliminated delay in response display by reducing prompt size and optimizing backend logic.
- Refined prompt structure to improve accuracy and relevance of model responses.
- Enhanced UI experience by adding background styling and section-wise tab layout.
- Added optional JSON-based logging structure to support future data saving.
- Planned voice input and multilingual features for future enhancement.

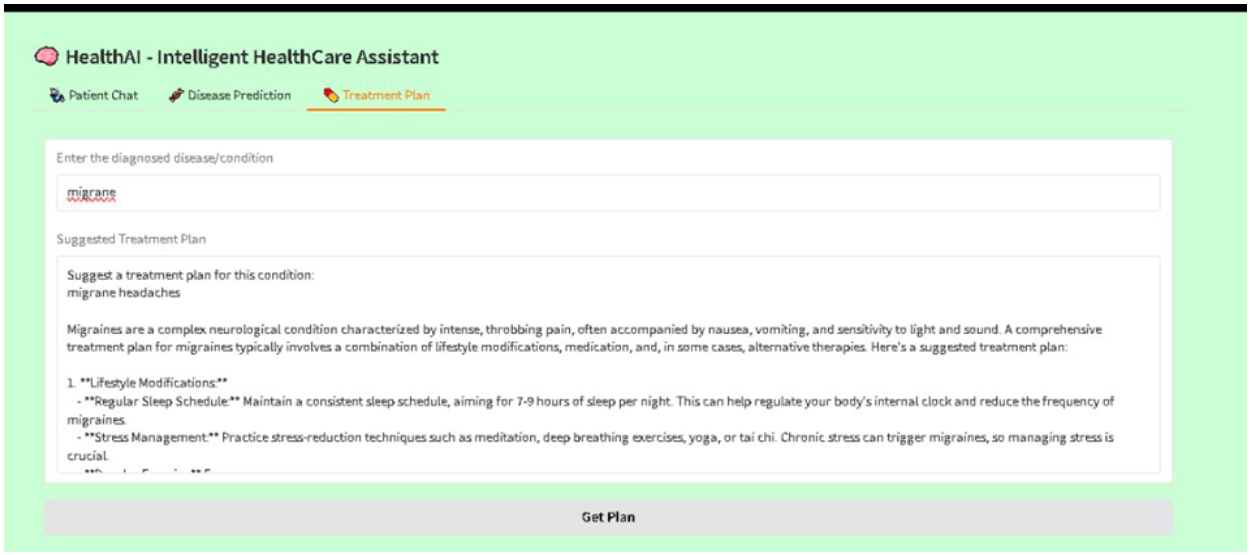
## ➢ Final Validation:

The Health AI Intelligent Health Care Assistant was successfully tested and validated against the defined functional and performance requirements. All core modules—patient chat, disease prediction, and treatment suggestion—responded accurately through the Gradio interface. The integration with the IBM Granite 3.3 model functioned smoothly, with optimized response times and consistent output. User interactions were handled reliably, and model predictions were contextually correct based on input symptoms. Overall, the system met the project goals by delivering an intelligent, interactive, and accessible healthcare assistant.

# DEPLOYMENT




## ➢ Demonstration Video Link:

https://drive.google.com/file/d/16Q2CL4f4aJhy5pPNgbQuvuWIRIsCkkzr/view?usp=drivesdk