

---

## ■ Generative AI Project using IBM Cloud – HEALTHAI

### ◇ Project Documentation Format

---

#### 1. Introduction

- **Project Title: HEALTHAI: Intelligent Healthcare Assistant using IBM Granite (Generative AI with IBM Cloud)** □ **Team Members:**
  - **Koti Venkata Srinivasarao (Team Leader – Requirements gathering ,Interaction& Testing):**  
Contributed by assisting in prompt design, testing the AI model outputs across modules like Disease Prediction and Health Chat, and refining interactions with IBM Granite.
  - **Kesani Santhosh Kumar (Team Member - Development , Integration , Deployment & documentation):**  
Led the complete development of the HEALTHAI application, including IBMGranite integration, Streamlit-based UI design, module creation, and model API handling,project deployment
  - **Kambhampati Dharaneeswar (Team Member - UI Structuring , Feature Enhancement & Testing):**  
Supported in designing user flow, organizing the Streamlit interface across all modules, and suggesting improvements in user interaction and feature behavior.
  - **Kaja Bala Nithin Reddy (Team Member - Feature Enhancement ,Testing & documentation):**  
Supported in- Feature Enhancement of project, testing the project and helped in the documentation.

---

#### 2. Project Overview

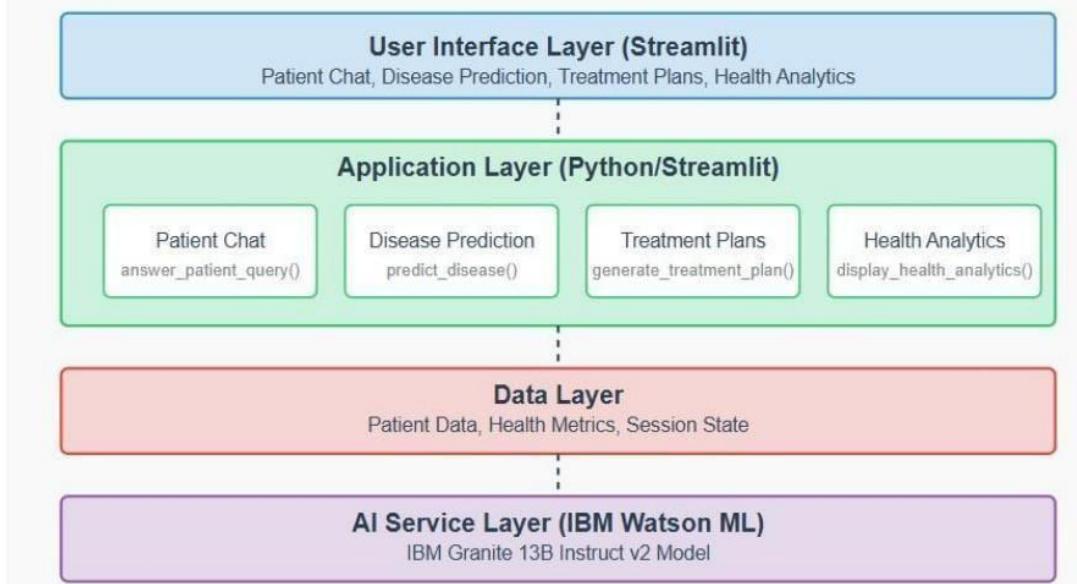
- **Purpose:**  
To build a Generative AI-based healthcare assistant using IBM Granite, capable of answering health queries, predicting diseases, suggesting treatments, and displaying analytics.
- **Features:**
  -  AI Health Chat using IBM Granite
  -  Disease Prediction from user symptoms

-  Treatment Plan Suggestions
-  Health Analytics Dashboard
-  Centralized shared model for performance optimization

### 3. Architecture

- **Frontend:**  
Built using **Streamlit** for a clean and responsive web interface. Each feature is modularized for easy navigation via sidebar.
- **Backend & Model:**
  - No traditional backend. All logic handled in Streamlit using Python.
  - Uses **IBM Granite 3-2b Instruct model** from IBM Watsonx: [ibm-granite/granite-3.2b-instruct](#)
  - Supports both API and **local model loading** (grainite/folder).
- **Shared Model Loader:**  
The shared\_model.py file centrally loads and shares the AI model across modules to prevent memory crashes and redundancy.

**HealthAI - Architecture Diagram**



### 4. Setup Instructions



## Prerequisites

- Python 3.10+
- pip
- IBM cloud account and Streamlit Community Cloud account
- Installed model files if using local (granite/ folder) **Installation**

git clone: [https://github.com/KESANI-SANTHOSH-KUMAR/HealthCare\\_AI.git](https://github.com/KESANI-SANTHOSH-KUMAR/HealthCare_AI.git)

cd Healthai

pip install -r requirements.txt

## Environment Variables

Create a .env file in the root folder:

IBM watsonx\_API\_Key= ESdPlIw78JY8LM32Vp6ujw\_EVP1dySvqb5ZQ5lc7K4wi

.env file must be excluded in .gitignore.

---

## 5. Folder Structure

Health-ai/

```
|—— app.py           # Main entry point  
|—— shared_model.py    # Shared AI model instance  
|—— patient_chat.py     # AI Health Chat module  
|—— disease_prediction.py  # Disease Prediction logic  
|—— treatment_plans.py    # Treatment Plan suggestions  
|—— health_analytics.py   # Analytics module  
|—— requirements.txt      # Python dependencies  
|—— .env                # API token (not pushed to GitHub)  
|—— granite/            # [Optional] Local model folder  
└—— assets/             # Logos and screenshots
```

---

## 6. Running the Application



### For IBM watsonx API:

streamlit run app.py

### For Local Model:

Ensure granite/ folder contains the downloaded model and tokenizer files.

In shared\_model.py, update: model\_path = "./granite"

---

## 7. API Documentation

**Endpoint:** <https://eu-de.ml.cloud.ibm.com/ml/v1/text/chat?version=2023-05-29>

### Headers:

```
{  
    "Authorization": "Bearer <IBM _API_Key>",  
    "Content-Type": "application/json"  
}
```

### Example Request:

```
{  
    "inputs": "What are the symptoms of diabetes?"  
}
```

### Example Response:

```
{  
    "generated_text": "Common symptoms of diabetes include frequent urination..."  
}
```

---

## 8. Authentication

- Streamlit cloud Secrets is securely stored the .env credentials.
- .env is excluded via .gitignore
- App is currently public and stateless (no user login)
- HuggingFace or Firebase Auth can be added in future

---

## 9. User Interface



**SMARTBRIDGE**  
Let's Bridge the Gap

- Built entirely with **Streamlit**
  - Sidebar for navigation
  - Text/chat inputs for interaction
  - Visual graphs and health tips in Analytics
  - Centralized theme and branding

## 10. Testing

- Manual testing across all modules
  - Model tested with varied prompts and edge cases
  - Handled errors for invalid inputs and model timeouts

## 11. Screenshots or Demo

```
File Edit Selection View Go Run Terminal Help < > Healthcare_AI

Healthcare_AI
  +-- .vscode
    +-- tasks.json
    +-- settings.json
    +-- launch.json
  +-- health
    +-- __init__.py
    +-- util.py
    +-- __pycache__
      +-- __init__.cpython-310.pyc
    +-- predict.py
    +-- __pycache__
      +-- __init__.cpython-310.pyc
  +-- __pycache__
    +-- __init__.cpython-310.pyc
  +-- requirements.txt

Healthcare_AI> .vscode> tasks.json > get_response
  1 import os
  2 import json
  3 import requests
  4 from dotenv import load_dotenv
  5
  6 load_dotenv()
  7
  8 API_KEY = os.getenv("AI_API_KEY")
  9 URL_UNI = os.getenv("AI_URL")
10 PROJECT_ID = os.getenv("PROJECT_ID")
11
12 def gen_access_token():
13     token_url = "https://iam.cloud.ibm.com/identity/token"
14     print(f" Getting access token... ")
15     headers = {"Content-Type": "application/x-www-form-urlencoded"}
16     data = {
17         "apikey": API_KEY,
18         "grant_type": "client_credentials"
19     }
20
21     token_response = requests.post(token_url, headers=headers, data=data)
22     if token_response.status_code != 200:
23         raise Exception(f"Error fetching access token")
24
25     access_token = token_response.json().get("access_token")
26     print(f" Access token received!")
27     print(f" Sending prompt to generate model...")
28
29     model_id = "lmi/granite-2b-instruct"
30     inference_url = f"{URL_UNI}/ml/v1/test/generation/version={PROJECT_ID}-01"
31
32     payload = {
33         "model_id": model_id,
34         "input": prompt,
35         "parameters": [
36             {"decoding_method": "greedy",
37              "max_new_tokens": 100},
38         ],
39         "parameters": [
40             {"decoding_method": "greedy",
41              "max_new_tokens": 100},           # longer output
42         ],
43         "project_id": PROJECT_ID
44     }
45
46     headers = {
47         "Authorization": f"Bearer {access_token}",
48         "Content-Type": "application/json"
49     }
50
51     response = requests.post(inference_url, json=payload, headers=headers)
52     if response.status_code == 200:
53         return response.json()[0].get("generated_text")
54     else:
55         return f"API Error: {response.status_code} - {response.text}"
56
57 if __name__ == "__main__":
58     print("Healthcare AI: Predict medical intelligence in simple words.")
59     response = get_ai_response(prompt)
60     print(f"Response: {response}")
61
62 def predict_disease(symptoms):
63     A patient is experiencing the following symptoms: {symptoms}.
64
65     Please consider the most common and likely causes first, based on global clinical guidelines. Avoid listing rare or severe diseases unless strongly indicated.
66
67     List top 5 possible diagnoses with estimated likelihoods in percentage format, example:
```



The image displays two side-by-side Jupyter Notebook interfaces, both titled "HealthCare\_AI".

**Top Notebook (Left):**

```
import streamlit as st
import requests
from util import get_st_response, predict_disease, generate_treatment_plan
from util.visualizer import display_health_analytics
```

This notebook contains Streamlit code for a user interface. It includes a sidebar with navigation options: "Patient Chat", "Disease Prediction", "Treatment Plan", and "Health Analytics". The main area handles user input for symptoms, disease prediction, treatment plans, and health analytics.

**Bottom Notebook (Right):**

```
import streamlit as st
import pandas as pd
import plotly.express as px
from util import predict_disease, generate_treatment_plan
```

This notebook contains Streamlit code for displaying health analytics. It reads CSV or Excel files, processes the data, and generates various charts (Line Series Charts, Histograms, Bar Charts) to summarize the health data.

## • OUTPUT :



**SMARTBRIDGE**  
Let's Bridge the Gap

The screenshot shows a web browser with multiple tabs open. The active tab is titled "healthcareai-7777.streamlit.app". The page content is as follows:

# HealthAI - Intelligent Healthcare Assistant

## Patient Profile

Name: [Input field]

Age: 30 - +

Gender: Male

Medical History: [Input field]

Current Medications: [Input field]

## AI-Powered Healthcare Assistant

Enter your health concern:

i am feeling ill

Know what is the problem

1. Assess the severity of symptoms:
  - Have you experienced fever, chills, or body aches?
  - Are you experiencing shortness of breath or chest pain?
  - Have you noticed changes in your appetite or sleep patterns?
2. Home remedies for mild symptoms:
  - If you're experiencing a common cold or flu:
    - Stay hydrated by drinking plenty of fluids (water, herbal tea, etc.)

The screenshot shows a dark-themed Streamlit application titled "HealthAI - Intelligent Healthcare Assistant". The interface is divided into two main sections: "Patient Profile" on the left and "AI-Powered Healthcare Assistant" on the right.

**Patient Profile Section:**

- Name:** Input field (disabled, showing placeholder text).
- Age:** Input field set to "30" with increment/decrement buttons.
- Gender:** Select dropdown menu set to "Male".
- Medical History:** Input field (disabled, showing placeholder text).
- Current Medications:** Input field (disabled, showing placeholder text).

**AI-Powered Healthcare Assistant Section:**

- Symptom Input:** Text input field containing "headache, body pains, nose bleeding".
- Predict Disease Button:** A large blue button labeled "Predict Disease".
- Diagnosis Output:** A list titled "Top 3 likely diagnoses with estimated likelihoods:"
  1. Migraine: 70%
  2. Tension headache: 20%
  3. Sinusitis: 10%
- Explanation:** A detailed paragraph explaining the likelihood of each diagnosis based on common symptoms like headache and body pain.



SMARTBRIDGE  
Let's Bridge the Gap

## Patient Profile

AI-Powered Healthcare Assistant

Name: [Input Field]

Age: 30

Gender: Male

Medical History: [Input Field]

Current Medications: [Input Field]

Allergies: [Input Field]

Get Treatment Plan

disease: migraine

Treatment Plan:

- First-line medication: Acute treatment with over-the-counter pain relievers (e.g., ibuprofen, naproxen) or prescription-strength triptans (e.g., sumatriptan, rizatriptan).
  - For mild to moderate migraines, start with an over-the-counter pain reliever. If symptoms persist or worsen, proceed to triptans.
  - Triptans should be taken as soon as possible after the onset of migraine symptoms.
- Supportive care:
  - Ensure adequate hydration and rest.
  - Consider using non-pharmacological interventions such as:
    - Cold or warm compress on the head or neck.
    - Ginger or other natural remedies (consult a healthcare provider before use).
    - Relaxation techniques, deep breathing, or biofeedback.
- When to escalate to a healthcare provider:
  - If acute medication fails to provide relief within 2 hours.

Save Profile

Manage app

## Health Analytics Dashboard

Raw Data

Date	HeartRate	SystolicBP	DiastolicBP	BloodGlucose	Symptom
0 2024-01-01	72	120	80	95	Headache
1 2024-01-02	78	122	82	105	Fatigue
2 2024-01-03	85	130	88	110	Dizziness
3 2024-01-04	90	135	85	98	Headache
4 2024-01-05	76	118	76	92	Nausea
5 2024-01-06	70	115	75	89	Fatigue
6 2024-01-07	88	140	90	120	Dizziness

Data Summary

Upload your health data (CSV or Excel)

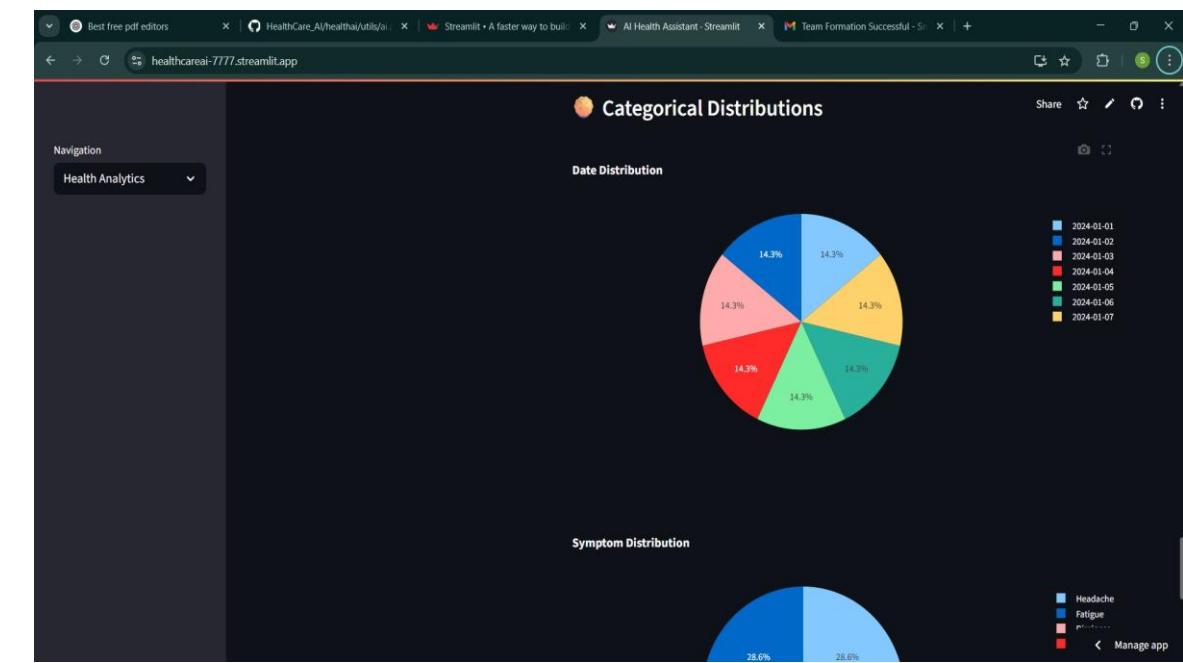
Drag and drop file here  
Limit 200MB per file • CSV, XLSX

Browse files

health\_data\_with\_symptoms.csv 291.0B

Save Profile

Manage app



## 12. Known Issues

- ⌚ Generic model outputs due to lack of medical domain fine-tuning
- ⌚ Internet dependency when using IBM Watson.
- ⌚ No data persistence (currently stateless app)

---

## 13. Future Enhancements

- Add user authentication and patient record storage
  - Multilingual prompt support
  - Mobile version of the app
  - Integrate with real-time health APIs or EHRs
-