**HealthAI: Intelligent Healthcare Assistant Using IBM Granite**

**Project Documentation**

**1.1 Project Overview**

HealthAI is an intelligent healthcare assistant designed to answer medical and health-related queries using natural language understanding. It leverages IBM’s Granite language model hosted on Hugging Face, with a user interface built using Streamlit.

**TEAM MEMBERS** : (Team ID: LTVIP2025TMID30148)

MEMBER 1(Leader): Bandi Naveen

MEMBER 2: Kalluri Nikhila

MEMBER 3: Reddyvari Prameela

**1.2 Purpose**

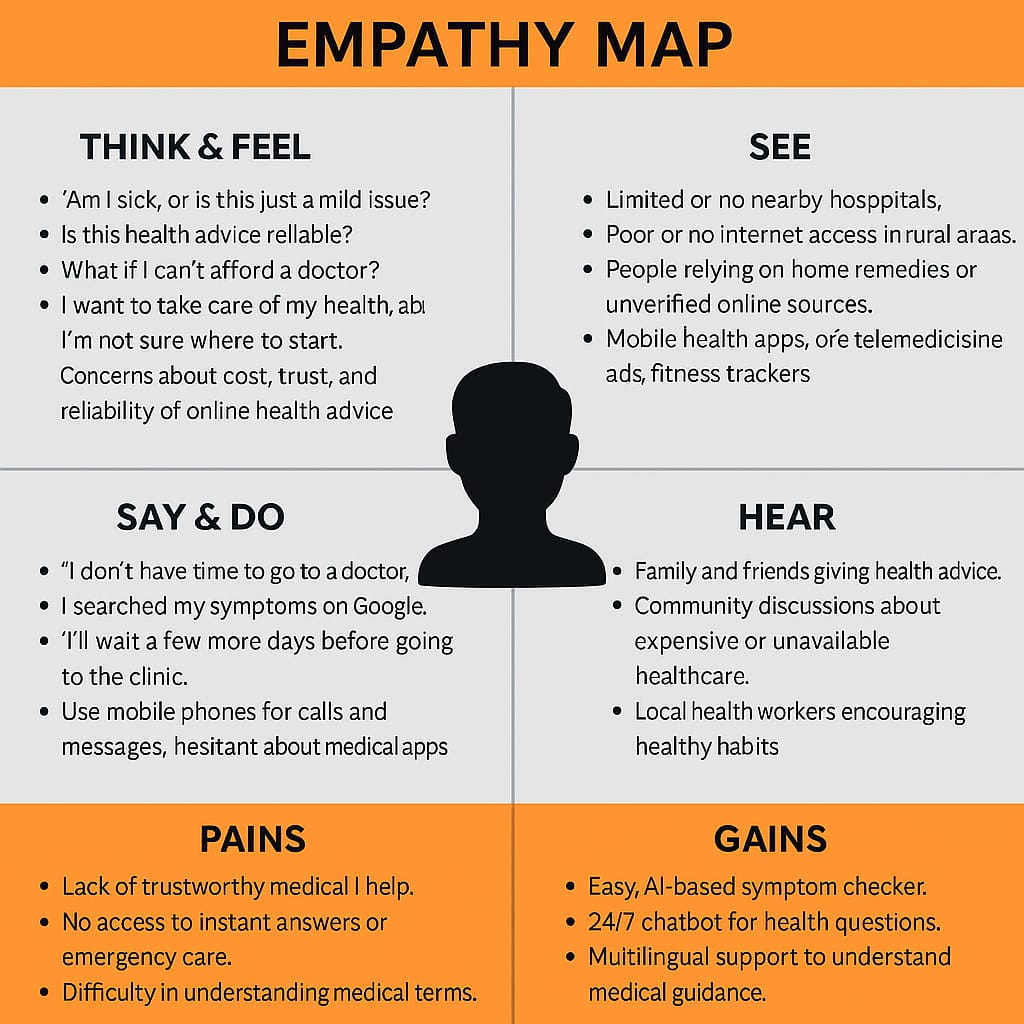
The purpose of this project is to provide users with accurate, AI-powered responses to their health concerns while demonstrating the integration of a powerful large language model in a real-world healthcare support tool.

**2. IDEATION PHASE**

**2.1 Problem Statement**

People often rely on unverified sources for health information. There’s a need for an intelligent system that can provide trustworthy, AI-generated answers to basic health-related questions.

**2.2 Empathy Map Canvas**

  
The empathy map considered users who need instant health advice, prefer a simple UI, and value data privacy.

**2.3 Brainstorming**

Key ideas:

* Integrate LLM for accurate health Q&A
* Use a no-code interface (Streamlit)
* Keep API keys secure with .env
* Handle edge cases like blank inputs or long queries

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**  
From visiting the app → asking a question → receiving a response → closing the app.

**3.2 Solution Requirement**

* Functional API Integration
* Responsive UI
* Reliable model outputs
* Minimal latency
* No need for registration/login

**3.3 Data Flow Diagram**

User input → Streamlit frontend → Hugging Face API → IBM Granite Model → Response → Streamlit output box.

**3.4 Technology Stack**

* **Frontend:** Streamlit
* **Backend:** Python
* **Model:** IBM Granite (granite-3b-instruct)
* **API Provider:** Hugging Face
* **Deployment:** Streamlit Cloud
* **Token Management:** Streamlit token manager

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit**

The project addresses the gap in reliable, instant health info using cutting-edge AI with zero setup effort for users.

**4.2 Proposed Solution**

An easy-to-use app where users input queries and get AI-generated answers from a healthcare-tuned language model.

**4.3 Solution Architecture**  
Components: UI (Streamlit), Query Processor (Python), AI Model (IBM Granite), Deployment Layer (Streamlit Cloud).

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

* Week 1: Problem analysis & brainstorming
* Week 2: Model integration using Hugging Face
* Week 3: UI development with Streamlit
* Week 4: Testing, Debugging, Deployment

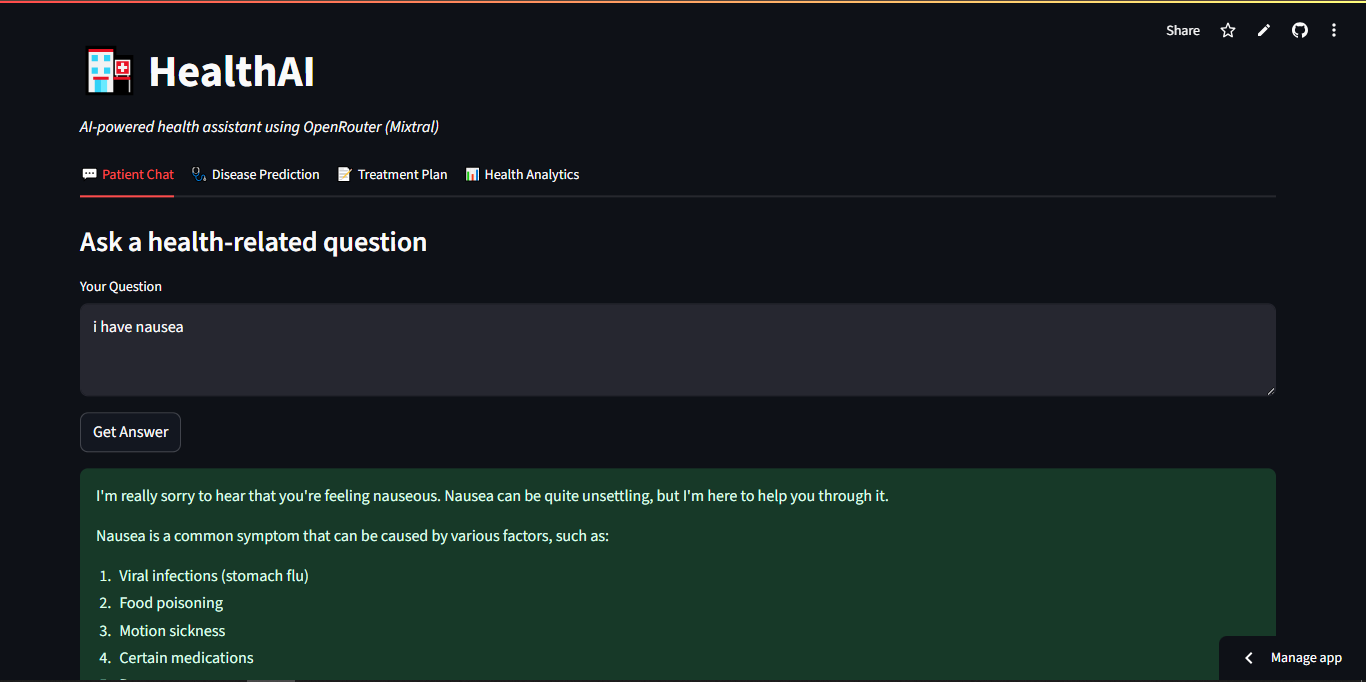
**6. FUNCTIONAL AND PERFORMANCE TESTING**

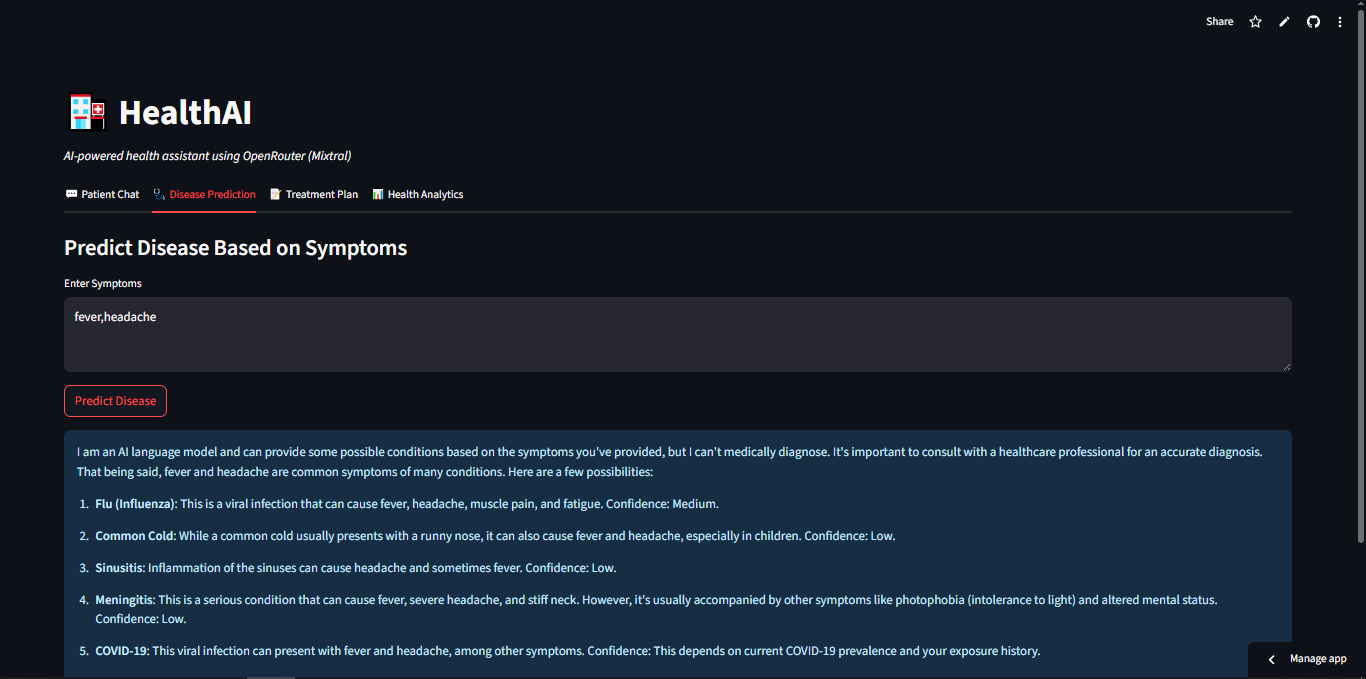
**6.1 Performance Testing**

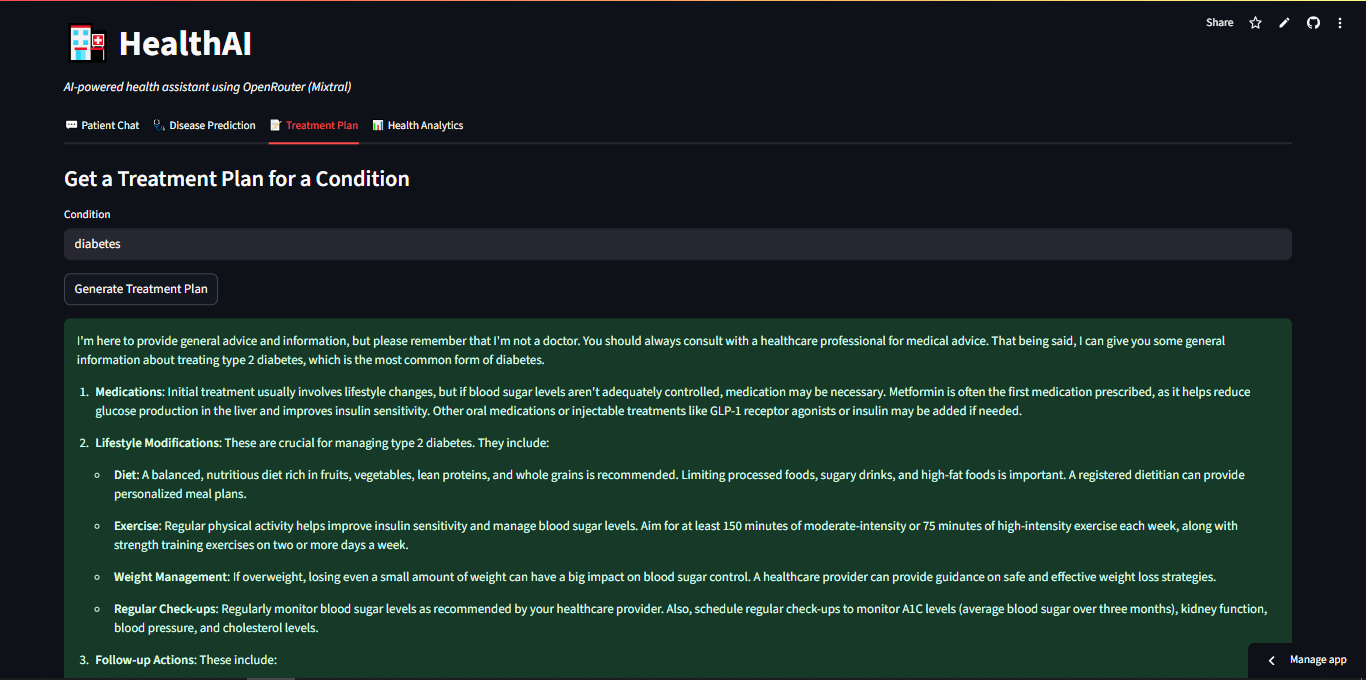
* API response time under 5 seconds on average
* Handled 10+ queries in one session with no crash
* All expected UI elements rendered correctly on different devices
* Error handling worked for invalid inputs

**7. RESULTS**

**7.1 Output Screenshots**







**8. ADVANTAGES & DISADVANTAGES**

**Advantages:**

* Simple, clean interface
* No login required
* Quick and accurate answers
* Deployed online and publicly accessible

**Disadvantages:**

* Depends on API availability
* No medical validation of AI answers
* Not intended for emergency or diagnostic use

**9. CONCLUSION**

HealthAI showcases how modern LLMs(IBM Granite) can be effectively used in healthcare assistance with minimal code. It simplifies access to medical information and shows the power of integrating open-source tools with cloud deployment.

**10. FUTURE SCOPE**

* Add voice input and output
* Enable multilingual support
* Include visual charts for health metrics
* Integrate with real-time health databases or APIs (e.g., WHO, CDC)
* Add chatbot memory to track context

**11. APPENDIX**

**Source Code:**  
import streamlit as st

import pandas as pd

import plotly.express as px

import requests

# Load API Key from Streamlit Secrets

API\_KEY = st.secrets["OPENROUTER\_API\_KEY"]

# Function to query OpenRouter

def query\_openrouter(prompt):

headers = {

"Authorization": f"Bearer {API\_KEY}",

"Content-Type": "application/json"

}

data = {

"model": "mistralai/mixtral-8x7b-instruct",

"messages": [

{"role": "system", "content": "You are a helpful, medically accurate AI health assistant."},

{"role": "user", "content": prompt}

]

}

response = requests.post("https://openrouter.ai/api/v1/chat/completions", headers=headers, json=data)

if response.status\_code == 200:

return response.json()['choices'][0]['message']['content']

else:

return f"❌ Error {response.status\_code}: {response.text}"

# Streamlit UI

st.set\_page\_config(page\_title="HealthAI", layout="wide")

st.title("🏥 HealthAI")

st.markdown("\*AI-powered health assistant using OpenRouter (Mixtral)\*")

tab1, tab2, tab3, tab4 = st.tabs(["💬 Patient Chat", "🩺 Disease Prediction", "📝 Treatment Plan", "📊 Health Analytics"])

# 💬 Tab 1: Patient Chat

with tab1:

st.subheader("Ask a health-related question")

question = st.text\_area("Your Question", placeholder="e.g., What should I do if I have chest pain?")

if st.button("Get Answer"):

if question.strip() == "":

st.warning("Please enter a valid question.")

else:

prompt = f"A patient asked: '{question}'. Provide a medically accurate, clear, and empathetic answer."

st.success(query\_openrouter(prompt))

# 🩺 Tab 2: Disease Prediction

with tab2:

st.subheader("Predict Disease Based on Symptoms")

symptoms = st.text\_area("Enter Symptoms", placeholder="e.g., fever, headache, nausea")

if st.button("Predict Disease"):

if symptoms.strip() == "":

st.warning("Please enter valid symptoms.")

else:

prompt = f"Symptoms reported: {symptoms}. Predict possible diseases or conditions with confidence levels."

st.info(query\_openrouter(prompt))

# 📝 Tab 3: Treatment Plan

with tab3:

st.subheader("Get a Treatment Plan for a Condition")

condition = st.text\_input("Condition", placeholder="e.g., Diabetes")

if st.button("Generate Treatment Plan"):

if condition.strip() == "":

st.warning("Please enter a valid condition.")

else:

prompt = f"Condition: {condition}. Suggest a detailed treatment plan including medications, lifestyle modifications, and follow-up actions."

st.success(query\_openrouter(prompt))

# 📊 Tab 4: Health Analytics

with tab4:

st.subheader("Upload Health Report (CSV)")

uploaded\_file = st.file\_uploader("Upload a CSV file", type=["csv"])

if uploaded\_file is not None:

try:

df = pd.read\_csv(uploaded\_file)

st.write("✅ File uploaded successfully!")

st.write(df.head())

# Get numeric columns for plotting

numeric\_cols = df.select\_dtypes(include=["float64", "int64"]).columns.tolist()

if numeric\_cols:

metric = st.selectbox("Select metric to plot", numeric\_cols)

fig = px.line(df, y=metric, title=f"{metric} Trend")

st.plotly\_chart(fig)

else:

st.warning("No numeric columns found for plotting.")

except Exception as e:

st.error(f"Error reading file: {e}")

else:

st.info("Upload a CSV file to view your health analytics.")

**Dataset Link:**  
*Not applicable (LLM-based, not trained on local dataset)*