HEALTHAI : INTELLIGENT HEALTHCARE ASSISTANT
USING IBM GRANITE

📌 Project Title:

Health AI: Intelligent Healthcare Assistant Using IBM Granite

---

👥 Team Members:

1. G Aswini (TEAM LEADER)

2. Gangula Manjunadha (TEAM MEMBER)

3. I Mani Sankar Reddy (TEAM MEMBER)

4. Yarrasu Harini (TEAM MEMBER)

🔎 Phase 1 – Brainstorming & Ideation

Objectives:

Identify a real-world healthcare issue: slow or inaccessible medical consultation.

Use AI to provide fast, intelligent preliminary symptom analysis.

Key Points:

1. Project Statement:
Many people delay medical help due to inaccessibility, cost, or uncertainty. This assistant provides a basic AI-driven health check based on symptoms.

2. Project Solution:

A command-line (or web-based) AI assistant that uses IBM Granite LLM to analyze user-input symptoms and suggest possible conditions.

3. Target User:

General public

Rural population with limited access to doctors

Students and researchers studying health-tech

4. Expected Output:

A list of possible health conditions based on symptoms

Timestamped report for review

Disclaimer advising professional consultation

📋 Phase 2 – Requirement Analysis

Objectives:

Gather the necessary tools to build the assistant.

Define core functionalities.

Key Points:

1. Technical Requirements:

Programming Language: Python

LLM: IBM Granite granite-3.3-2b-instruct

Libraries: transformers, torch

(Optional) IBM Cloud or Streamlit for deployment

2. Functional Requirements:

Input: Name and symptoms

Process: Format input as prompt, send to IBM Granite, get response

Output: Display possible conditions, disclaimer, and timestamp

---

🔗 3. Constraints & Challenges

Limited medical accuracy – the AI does not replace doctors.

Dependency on model prompt quality – output changes if prompt isn't well-structured.

Internet access needed – for model inference (especially if using IBM Cloud).

No real-time diagnosis – only suggestion-based support.

 Phase 3 – Project Design

Objectives:

Design a user-friendly and technically robust system structure.

Key Points:

1. System Architecture Diagram:

User Input → Prompt Builder → IBM Granite Model → Output Parser → Display Result

2. User Flow:

User opens app
Enters name and symptoms

System formats and sends prompt to model

Model responds → output shown to user

3. UI/UX Consideration:

Simple CLI (can be extended to Streamlit UI)

Minimal input steps for ease of use

Clear display of results with warnings/disclaimers

📅 Phase 4 – Project Planning (Agile Methodologies)

Objectives:

Implement iterative development with flexibility and team collaboration.

Key Points:

1. Sprint Planning:

Sprint 1: Input/output + symptom prompt generation

Sprint 2: Model integration & testing

Sprint 3: Output formatting & basic UI

2. Task Allocation:

Member 1: Input & prompt logic

Member 2: Model integration

Member 3: UI/UX design
Member 4: Testing & documentation

3. Time & Milestones:

Week 1: Base functionality

Week 2: Model connection + UI

Week 3: Testing + fixes

Week 4: Final report and presentation

 Phase 5 – Project Development

Objectives:

Build the application and integrate all components.

Key Points:

1. Technology Stack Used:

Python

Hugging Face Transformers

IBM Granite 3.3-2B Instruct

(Optional) Streamlit / IBM Cloud

2. Development Process:

Code structure set up

Tokenizer + model loaded

User input handling
Response generation and output cleanup

3. Challenges and Fixes:

Challenge: Long or irrelevant model outputs
Fix: Limit tokens and improve prompt structure

Challenge: Model loading time
Fix: Use smaller test prompts during development

 Phase 6 – Functional and Performance Testing

Objectives:

Ensure the app works as intended and is efficient for users.

Key Points:

1. Functional Testing:

Test valid/invalid symptom inputs

Test multiple users

Ensure proper output generation

2. Performance Testing:

Time taken for model response

Memory usage while generating outputs

3. Output Accuracy Checks:

Compare AI suggestions with verified sources
Validate disclaimer visibility

✅ Phase 6 – Functional and Performance Testing

Objectives:

Ensure the assistant performs as expected under various input conditions and is stable for end users.

Key Points:

1. Test Cases Executed:

Input with common symptoms (e.g., fever, cough)

Input with rare symptoms

Input with no symptoms or empty string

Long symptom strings to test output truncation

2. Bug Fixes & Improvements:

Fixed prompt formatting issues for better model output

Limited token length (max_new_tokens=200) to avoid overload

Added clearer instructions and timestamp formatting

3. Final Validation:

Verified output relevance using known medical databases

Ensured disclaimer is always shown

Confirmed input → prompt → output cycle functions without crash

---

📝 Final Submission

1. Project Report Based on the Templates:

Includes filled sections from Phases 1 to 6

Clear documentation of objectives, designs, code, and results

2. Demo Video:

A walkthrough showing:

Starting the app

Entering symptoms

AI response display

Conclusion with timestamp and disclaimer
(Use screen recorder + voiceover or subtitles)

3. GitHub/Code Repository Link:

> 📎 https://github.com/YourUsername/HealthAI-Granite (Replace with your real repo)

4. Presentation:

📌 Slide 1: Project Overview

Title: HealthAI – AI-Powered Healthcare Assistant

Goal: Use IBM Granite LLM to assist users by analyzing symptoms and suggesting possible conditions

Type: AI + Healthcare + NLP

Platform: Python CLI (can be extended to Streamlit Web UI)

🔍 Slide 2: Problem & Solution

Problem:

Delayed medical help in remote areas

Limited early screening options

Solution:

A fast, AI-driven tool that provides possible condition suggestions based on symptoms

Helps users decide if they should consult a doctor

⚙️ Slide 3: Architecture

Diagram:

User Input → Prompt Generator → IBM Granite Model → Output Parser → Display Result

Components:

transformers, torch, AutoTokenizer, AutoModelForCausalLM

Model: ibm-granite/granite-3.3-2b-instruct

 Slide 4: Features & Tech Stack

Key Features:

Symptom analysis

Real-time response using LLM

Timestamps and disclaimers

Tech Stack:

Python, Hugging Face Transformers

IBM Granite LLM

(Optional) Streamlit / IBM Cloud

 Slide 5: Testing & Results

Test Cases:

Common/rare/empty symptom inputs

Bug Fixes:

Prompt optimization

Token length control

Results:

Reliable AI suggestions (non-diagnostic)

Fast inference time (~1–2s)

 Slide 6: Demo Snapshot / Conclusion

Screenshots of:

Input prompt

AI-generated suggestions

Timestamp and warning

Conclusion:

HealthAI improves early health awareness

Can be scaled for multi-language, doctor-assisted apps

Future: connect with APIs for real medical advice
