**Health AI : Intelligent Healthcare Assistant Using IBM Granite Project Documentation**

**1. Introduction**

Project Title: Health AI : Intelligent Healthcare Assistant

Using IBM Granite.

Team Member: Deepalakshmi V

Team Member: Ayesha Nilofer K

Team Member: Bhagyashri M

Team Member: Bhavana Sakthi G

**2. Project Overview**

The Medical AI Assistant is an artificial intelligence–powered application designed to assist users in predicting possible diseases based on symptoms and generating treatment plan suggestions. It uses IBM Granite language models, fine-tuned for instruction following, along with the Hugging Face Transformers library for natural language processing.

This project was implemented in Python and made interactive using the Gradio interface, allowing end-users to input symptoms or patient details through a simple web-based interface. The solution is primarily for educational and informational purposes, and it is not a substitute for professional medical consultation.

The purpose of this document is to provide comprehensive technical documentation covering the system overview, architecture, setup instructions, folder structure, user interface, testing methods, and potential future enhancements.The project is focused on building a lightweight, deployable AI assistant capable of generating health-related insights.

**Key Features:**

* Symptom-based disease prediction.
* Patient detail–based treatment plan generation.
* Deployment-ready Gradio web interface.
* Uses IBM Granite Instruct Model hosted on Hugging Face.
* Portable and can be executed on Google Colab for cloud-based experiments.

**Objectives:**

1. Provide quick medical-related insights for educational use.

2. Demonstrate how transformer models can be integrated into real-world healthcare-inspired applications.

3. Build an interface that is easy for non-technical users to operate.

**3. Architecture**

The architecture of the Medical AI Assistant is modular and consists of the following components:

**1. Frontend Layer** – Gradio Interface

Provides a user-friendly UI for input and output.

Implements two functional tabs: Disease Prediction and Treatment Plans.

**2. Backend Layer** – Python with Transformers Library

Handles model loading and response generation.

Manages tokenization, prompt formatting, and response decoding.

**3. Model Layer** – IBM Granite Model on Hugging Face

Pre-trained causal language model.

Responsible for generating medically relevant responses.

**4. Deployment Layer** – Google Colab / Local Execution

Supports running on GPU-enabled Colab notebooks.

Easy for researchers and developers to test without heavy local setup.

**4. Google Colab Integration**

The project is optimized to run in Google Colab for seamless execution.

**Advantages of Colab:**

Free GPU access for faster inference.

Pre-installed packages like PyTorch and Transformers.

No local installation overhead.

**Steps:**

1. Open Google Colab and upload the project notebook/script.

2. Ensure GPU is enabled under Runtime → Change runtime type → GPU.

3. Install missing dependencies (pip install transformers gradio).

4. Run the application code and use the public Gradio link for access.

**5. Hugging Face Integration**

The application relies on Hugging Face Transformers for model loading.

Tokenizer: Handles input text tokenization.

Model: Loads ibm-granite/granite-3.2-2b-instruct for inference.

Pipeline: Generates responses based on user input prompts.

The Hugging Face hub allows seamless integration of IBM Granite and ensures reproducibility.

**6. IBM Granite Model**

IBM Granite 3.2-2B Instruct is a large language model optimized for instruction-following tasks.

**Why Granite?**

1. Lightweight (2B parameters) → deployable on standard GPUs.
2. Optimized for natural language understanding and reasoning.
3. Supports instructional prompting, making it suitable for medical Q&A.

**7. Gradio Interface**

Gradio provides a web-based frontend for the model.

Tabs Implemented:

**1. Disease Prediction Tab**

Input: Symptom description.

Output: Possible conditions & recommendations.

**2. Treatment Plan Tab**

Input: Condition, Age, Gender, Medical History.

Output: Personalized treatment suggestions.

This ensures non-technical users can interact with the AI model effortlessly.

**8. Python under Transformers Library**

* The application uses Python 3.8+ along with the Transformers library for handling the model.
* Core Python Features Used:
* Conditional GPU/CPU selection with PyTorch.
* Tokenizer for input encoding.
* Model.generate() for sequence output.
* Text decoding and formatting for clean results.

**9. GitHub Setup Instructions**

Developers can set up the project from GitHub using the following steps:

1. Clone the repository:

git clone <repository\_link>

cd medical-ai-assistant

2. Install dependencies:

pip install -r requirements.txt

3. Run the application:

python app.py

**10. Installation Process**

**Required Libraries:**

1. torch
2. transformers
3. gradio
4. Command:
5. pip install torch transformers gradio

**11. Folder Structure**

A typical folder structure:

medical-ai-assistant/

│

├── app.py # Main application script

├── requirements.txt # Dependencies

├── docs/ # Documentation

├── screenshots/ # UI Screenshots

├── models/ # Pre-trained models (if cached)

└── README.md # Project description

**12. Running the Application**

**Local Execution:**

python app.py

Google Colab Execution:

Run script cells sequentially.

Access Gradio public link.

**13. API Documentation**

The project includes two core functions:

1. disease\_prediction(symptoms)

Input: Symptom description string.

Output: Text response with conditions & recommendations.

2. treatment\_plan(condition, age, gender, history)

Input: Patient details.

Output: Personalized treatment plan.

**14. Authentication**

Currently, the system does not require authentication for local/Colab runs.

For production deployment, API keys or OAuth can be integrated for access control.

**15. User Interface**

Textbox Inputs: For symptoms and patient details.

Buttons: To trigger model prediction.

Textbox Outputs: Display conditions or treatment plans.

The UI emphasizes simplicity and accessibility.

**16. Testing**

Testing Strategy:

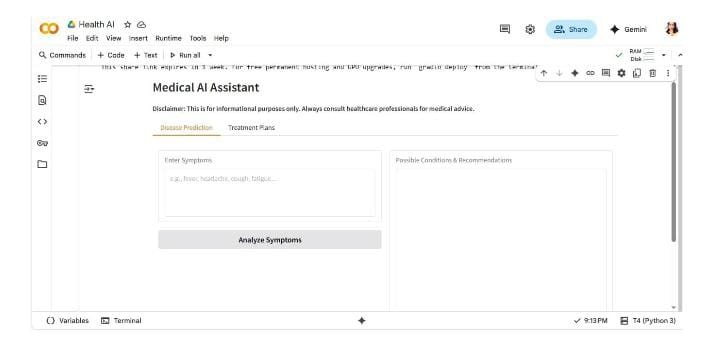
1. Unit testing of functions (generate\_response, disease\_prediction, etc.).

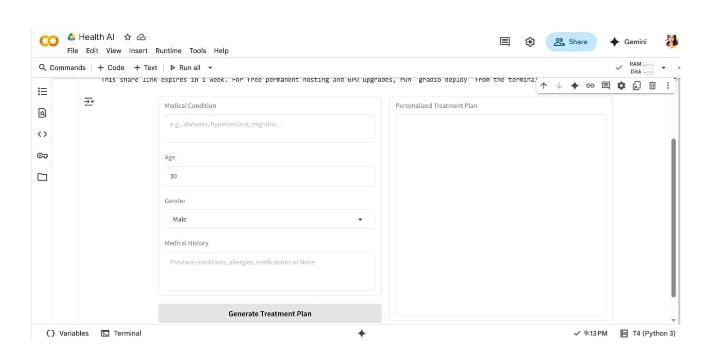
2. Functional testing of Gradio interface.

3. GPU vs CPU performance comparison.

4. Edge case testing (empty inputs, invalid data).

**17. Screenshots**

****



**18. Future Enhancements**

* Planned improvements include:
* Integration with medical knowledge bases.
* Support for voice input/output.
* Deployment as a mobile-friendly web app.
* Addition of user authentication and session tracking.
* Expansion with multi-language support.

**Conclusion**

The Medical AI Assistant is a robust AI-driven healthcare-inspired project that integrates transformers, Hugging Face, IBM Granite, and Gradio into one system. While it is a powerful educational tool, it is essential to note that it is not a replacement for medical professionals.