HEALTH AI

Project Documentation

1.INTRODUCTION

• Project Title: Health AI : Intelligent Healthcare Assistant

• Team Members:

Sridevi.V

Shanmuga Priya .B

Sharmila.L

Shree Nidhi.H

2.PROJECT OVERVIEW

• Purpose:

The purpose of the Health AI Assistant is to provide informational support for users seeking insights into potential medical conditions and treatment options. By leveraging AI and natural language processing, the assistant analyzes user-provided symptoms to suggest possible conditions and general recommendations, while also generating personalized treatment plans based on patient details. It emphasizes that all outputs are for informational purposes only and strongly advises consulting healthcare professionals for accurate

diagnosis and treatment. Ultimately, this assistant bridges AI technology with health awareness to promote informed decision-making in a user-friendly manner.

• Features:

Disease Prediction

Key Point: Symptom-based condition analysis
Functionality: Takes user-input symptoms and generates possible medical conditions along with general medication suggestions, including a disclaimer for professional consultation.

Treatment Plans

Key Point: Personalized health recommendations
Functionality: Accepts details like medical condition, age, gender, and history to provide tailored treatment suggestions, including home remedies and medication guidelines, with a strong emphasis on seeking medical advice.

Conversational Prompting

Key Point: AI-driven response generation
Functionality: Uses carefully crafted prompts to interact with the Granite LLM for natural, informative outputs.

Gradio UI

Key Point: Interactive web interface
Functionality: Provides a tabbed dashboard for easy access to disease prediction and treatment planning tools.

Disclaimer Integration

Key Point: Safety and ethical guidance
Functionality: Includes prominent disclaimers in the UI and responses to ensure users understand the limitations of AI advice.

3.ARCHITECTURE

Frontend (Gradio):
The frontend is built with Gradio, offering an interactive web UI with tabs for different functionalities, input fields, buttons, and output displays. It uses Gradio Blocks for layout, including markdown for disclaimers and rows/columns for organized user interaction. The interface is launched with a shareable link for easy access.

Backend:

The Health Al Assistant uses a script-based backend with no explicit REST API endpoints. The core logic is embedded in Python functions that interact directly with the Granite LLM model.
• Purpose: The backend handles natural language processing tasks, including generating responses for disease predictions and personalized treatment plans based on user inputs.

LLM Integration (IBM Watsonx Granite):

The Granite LLM model (ibm-granite/granite-3.2-2b-instruct) from Hugging Face is loaded using Transformers. It handles natural language generation for responses. Prompts are designed to incorporate user inputs, generate analyses, and append disclaimers. The model runs on CPU or GPU (if available) with torch for inference.

Core Functions:

Lightweight functions are used for response generation and specific tasks. Time-series or complex data handling is not applicable; focus is on text-based prompting and decoding.

SETUP INSTRUCTIONS

Prerequisites:
o Python 3.8 or later

o pip and virtual environment tools

o Internet access for downloading models from Hugging Face

o Optional: CUDA-enabled GPU for faster inference

Installation Process:

o Create a virtual environment

o Install dependencies: pip install gradio transformers torch

o Download the script (e.g., medical_ai_assistant.py)

o Run the script using python medical_ai_assistant.py

o Access the interface via the local or shared URL provided

5.FOLDER STRUCTURE

medical_ai_assistant.py – Main script containing all logic, including model loading, functions, and Gradio UI setup.
[No subdirectories or additional files specified; the project is contained in a single Python file for simplicity.]

6.RUNNING THE APPLICATION

To start the project:

➢ Run the Python script to load the model and launch the Gradio interface.

➢ Navigate to the provided URL (local or shared).

➢ Select tabs for Disease Prediction or Treatment Plans.

➢ Input data (e.g., symptoms or patient details) and click buttons to generate outputs.

➢ All interactions are real-time and handled within the script.

7.API DOCUMENTATION

No external APIs are used; internal functions serve as the core:

generate_response – Accepts a prompt and generates AI output using the Granite model. Parameters: prompt (string), max_length (integer).

disease_prediction – Processes symptoms to suggest conditions and recommendations. Parameter: symptoms (string).

treatment_plan – Generates personalized plans based on user details. Parameters: condition (string), age (number), gender (string), medical_history (string).

Functions are not exposed as REST endpoints but can be tested directly in the script or via the Gradio UI.

8.AUTHENTICATION

This version of the project runs in an open environment without authentication for demonstration purposes. However, secure deployments can integrate:

• API keys for model access if hosted

• User login via Gradio extensions

• Role-based access (e.g., user, admin)

• Planned enhancements include session tracking for personalized history.

9.USER INTERFACE

The interface is minimalist and functional, focusing on accessibility for non-technical users. It includes: Sidebar-free tabbed navigation Input fields for symptoms, condition, age, gender, and history Buttons for analysis and plan generation Real-time output textboxes Markdown disclaimers at the top The design prioritizes clarity, ease of use, and guidance with placeholders and labels.

10.TESTING

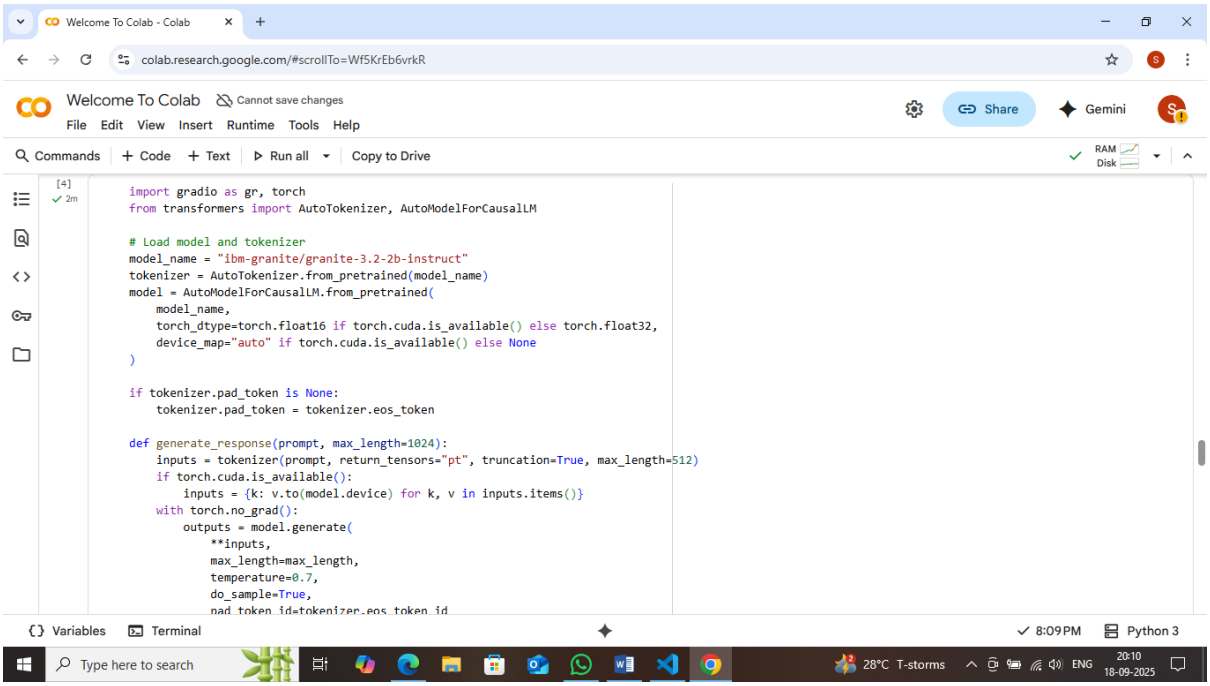
Testing was done in multiple phases:

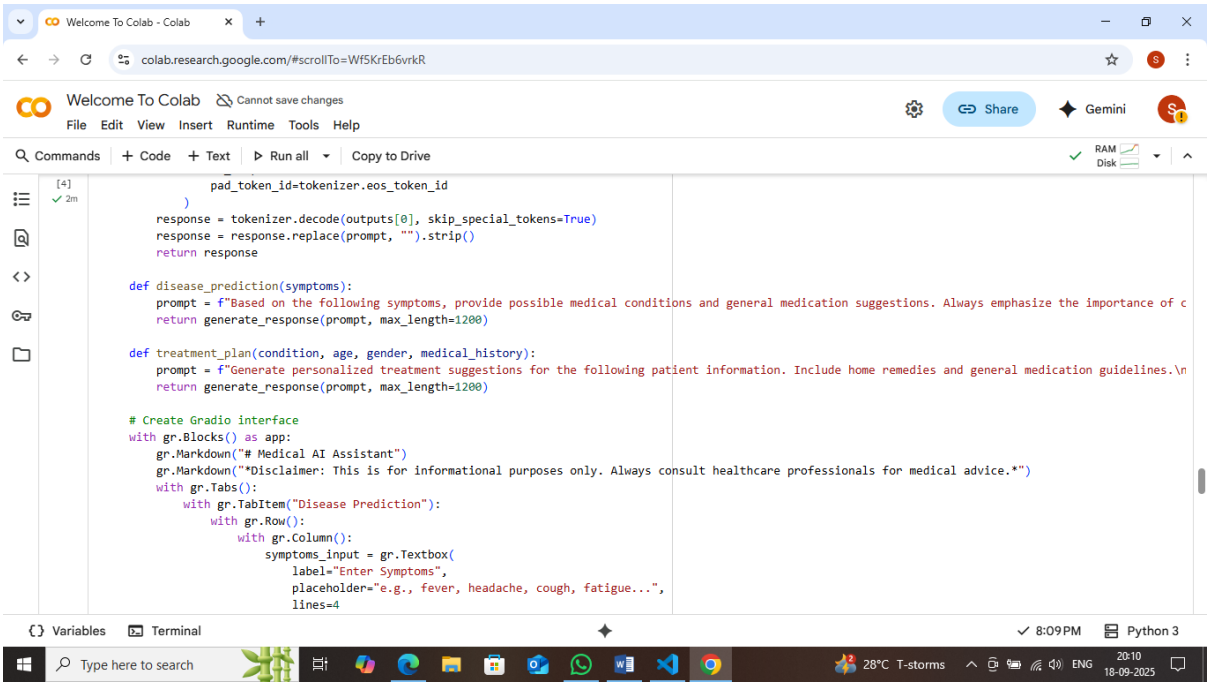
Unit Testing: For prompt functions and response generation

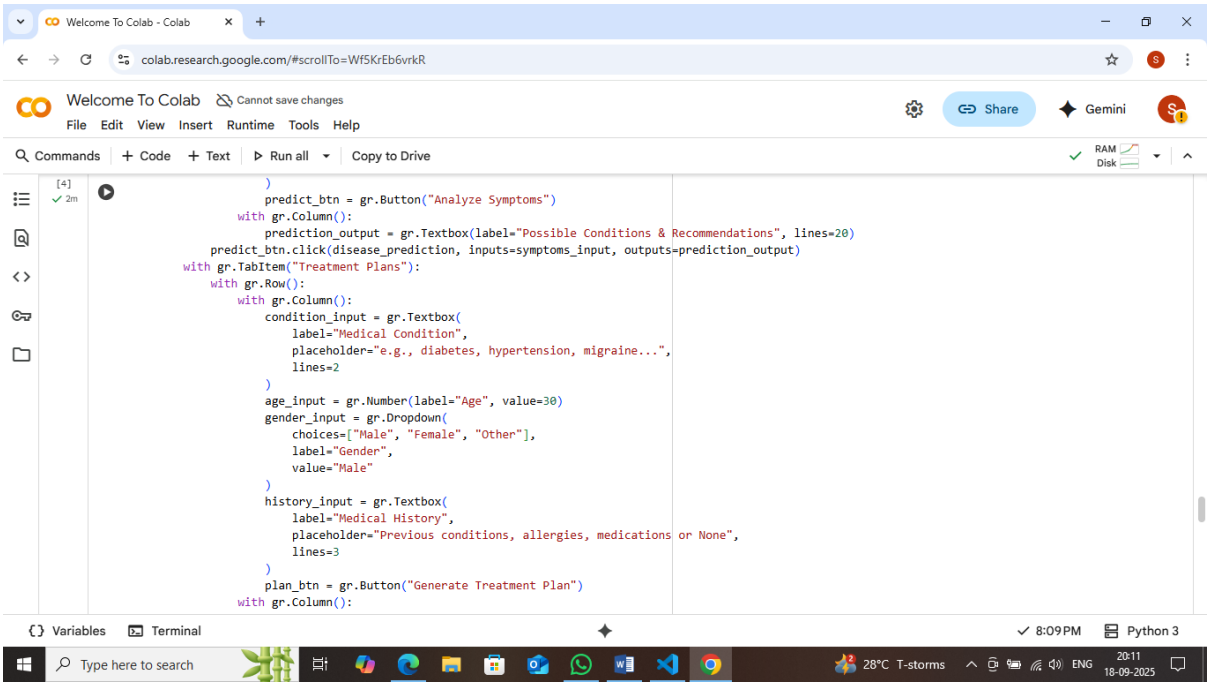
Manual Testing: For UI interactions, input handling, and output consistency

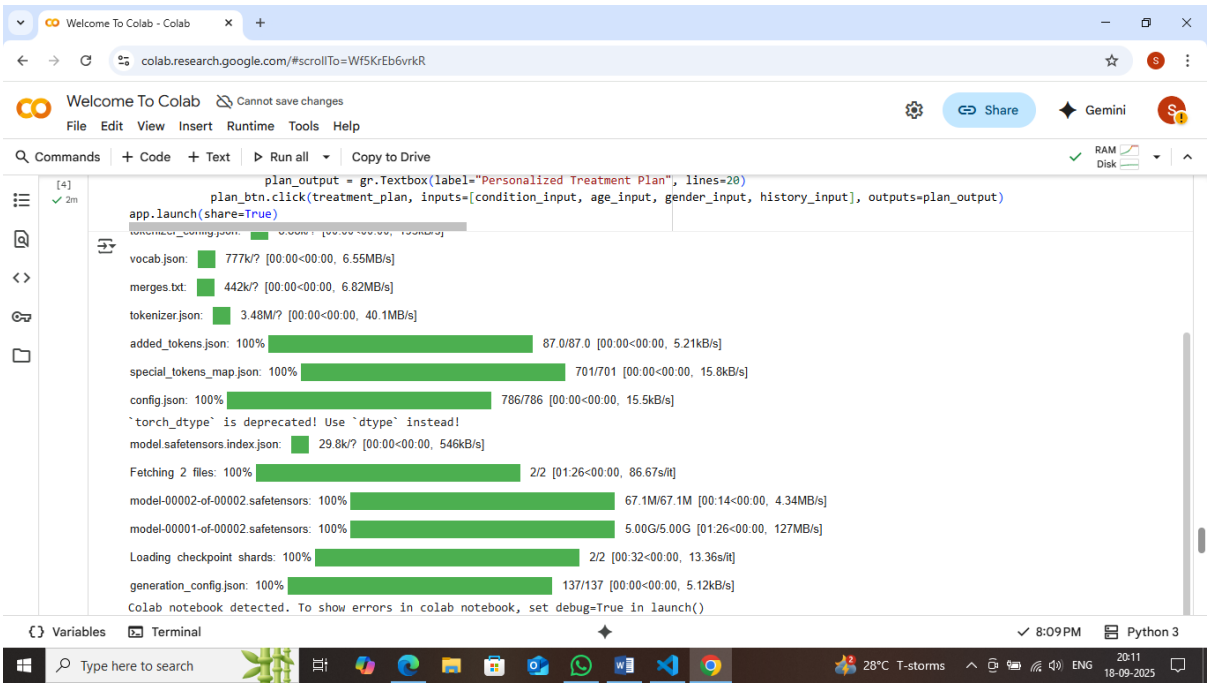
Edge Case Handling: Empty inputs, invalid ages, long symptoms Each function was validated to ensure reliable outputs and proper disclaimers.

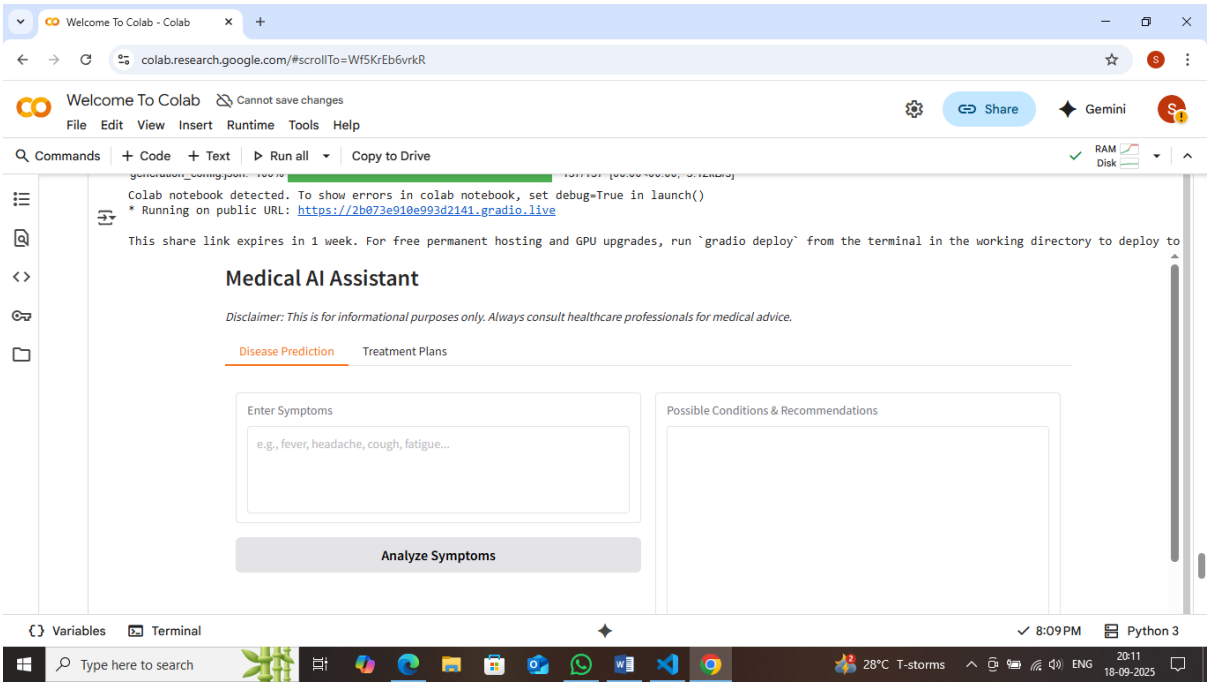
10.SCREEN SHOTS


Welcome To Colab - Colab
colab.research.google.com/#scrollTo=Wf5KrEb6vrkR
Welcome To Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all Copy to Drive
RAM
Disk
[4]
2m
import gradio as gr, torch
from transformers import AutoTokenizer, AutoModelForCausalLM
# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
model_name,
torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
device_map="auto" if torch.cuda.is_available() else None
)
if tokenizer.pad_token is None:
tokenizer.pad_token = tokenizer.eos_token
def generate_response(prompt, max_length=1024):
inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
if torch.cuda.is_available():
inputs = {k: v.to(model.device) for k, v in inputs.items()}
with torch.no_grad():
outputs = model.generate(
**inputs,
max_length=max_length,
temperature=0.7,
do_sample=True,
pad_token_id=tokenizer.eos_token_id
Variables
Terminal
8:09PM
Python 3
Type here to search
28°C T-storms
ENG
20:10
18-09-2025


Welcome To Colab - Colab
colab.research.google.com/#scrollTo=Wf5KrEb6vrkR
Welcome To Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all Copy to Drive
RAM
Disk
[4]
2m
pad_token_id=tokenizer.eos_token_id
)
response = tokenizer.decode(outputs[0], skip_special_tokens=True)
response = response.replace(prompt, "").strip()
return response
def disease_prediction(symptoms):
prompt = f"Based on the following symptoms, provide possible medical conditions and general medication suggestions. Always emphasize the importance of c
return generate_response(prompt, max_length=1200)
def treatment_plan(condition, age, gender, medical_history):
prompt = f"Generate personalized treatment suggestions for the following patient information. Include home remedies and general medication guidelines.\n
return generate_response(prompt, max_length=1200)
Create Gradio interface
with gr.Blocks() as app:
gr.Markdown("# Medical AI Assistant")
gr.Markdown("*Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.*")
with gr.Tabs():
with gr.TabItem("Disease Prediction"):
with gr.Row():
with gr.Column():
symptoms_input = gr.Textbox(
label="Enter Symptoms",
placeholder="e.g., fever, headache, cough, fatigue...",
lines=4
Variables
Terminal
8:09PM
Python 3
Type here to search
28°C T-storms
ENG
20:10
18-09-2025


Welcome To Colab - Colab
colab.research.google.com/#scrollTo=Wf5KrEb6vrkR
Welcome To Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all Copy to Drive
RAM
Disk
[4]
2m
predict_btn = gr.Button("Analyze Symptoms")
with gr.Column():
prediction_output = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)
predict_btn.click(disease_prediction, inputs=symptoms_input, outputs=prediction_output)
with gr.TabItem("Treatment Plans"):
with gr.Row():
with gr.Column():
condition_input = gr.Textbox(
label="Medical Condition",
placeholder="e.g., diabetes, hypertension, migraine...",
lines=2
)
age_input = gr.Number(label="Age", value=30)
gender_input = gr.Dropdown(
choices=["Male", "Female", "Other"],
label="Gender",
value="Male"
)
history_input = gr.Textbox(
label="Medical History",
placeholder="Previous conditions, allergies, medications or None",
lines=3
)
plan_btn = gr.Button("Generate Treatment Plan")
with gr.Column():
Variables
Terminal
8:09PM
Python 3
Type here to search
28°C T-storms
ENG
20:11
18-09-2025


Welcome To Colab - Colab
colab.research.google.com/#scrollTo=Wf5KrEb6vrkR
Welcome To Colab
Cannot save changes
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all Copy to Drive
RAM
Disk
[4]
2m
plan_output = gr.Textbox(label="Personalized Treatment Plan", lines=20)
plan_btn.click(treatment_plan, inputs=[condition_input, age_input, gender_input, history_input], outputs=plan_output)
app.launch(share=True)
vocab.json: 777k/? [00:00<00:00, 6.55MB/s]
merges.txt: 442k/? [00:00<00:00, 6.82MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 40.1MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 5.21kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 15.8kB/s]
config.json: 100% 786/786 [00:00<00:00, 15.5kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 546kB/s]
Fetching 2 files: 100% 2/2 [01:26<00:00, 86.67s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:14<00:00, 4.34MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:26<00:00, 127MB/s]
Loading checkpoint shards: 100% 2/2 [00:32<00:00, 13.36s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 5.12kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Variables
Terminal
8:09PM
Python 3
Type here to search
28°C T-storms
ENG
20:11
18-09-2025



11.KNOWN ISSUES

- Limited to text-based inputs; no support for images or files
- Requires stable internet for initial model download
- Outputs are AI-generated and may vary; not medically verified
- Potential for hallucinations in responses.

12.FUTURE ENHANCEMENTS

- Integration with medical databases for accuracy
- Multimodal support (e.g., image analysis for symptoms)
- User feedback collection

- Mobile optimization
- Advanced personalization with history tracking