HEALTH AI-ASSITANT

Project Documentation

1.Introduction

- Project title : HEALTH AI ASSISTANT
- Team member : KRISHNAPRIYA M
- Team member : MOHANA SUNDARI N
- Team member : NIVETHA A
- Team member :NAVEENA L

2. Project Overview

Purpose:
The Health AI Assistant is designed to provide users with quick, AI-driven medical insights for symptoms, conditions, and general wellness recommendations. It is not a replacement for doctors, but an **informational tool** that promotes awareness, encourages early consultation, and provides users with personalized guidance.

By leveraging AI and natural language models, the assistant can:

- Predict possible medical conditions based on symptoms
- Suggest general treatment and home remedies
- Provide personalized treatment plan drafts based on patient details

Important: All outputs emphasize the need for professional medical consultation.

Features:

- **Conversational Symptom Analysis**
  - Accepts symptoms in plain text and returns possible conditions.
- **Treatment Plan Generator**

    - Creates a customized treatment plan with home remedies and lifestyle tips based on user info (age, gender, history).
- **Disclaimer Integration**
    - Ensures every response reminds users to consult a healthcare professional.
- **Gradio-based UI**
    - Clean, accessible web interface for disease prediction and treatment planning.
- **LLM Integration**
    - Uses IBM Watsonx Granite (granite-3.2-2b-instruct) for high-quality medical text generation.

3. Architecture

Frontend (Gradio):

- Built with **Gradio Blocks & Tabs**.
- Provides two main modules:
    - *Disease Prediction* (symptom input → conditions & recommendations)
    - *Treatment Plans* (condition + patient details → personalized plan).
- Simple, user-friendly interface with disclaimers.

Backend (Python + Transformers):

- Uses Hugging Face Transformers with IBM Watsonx Granite model.
- Handles:
    - Symptom → prompt → AI response
    - Condition + patient details → prompt → AI-generated treatment plan

LLM Integration (IBM Watsonx Granite):

- Provides natural language understanding and generation.
- Prompts carefully structured to:
    - Return medical insights
    - Enforce safety disclaimers

4. Setup Instruction

Prerequisites:

- Python 3.9+
- pip and venv
- GPU (optional, for faster inference)
- Internet access (to fetch models)

5.Folder Structure:

app Core application code (Gradio interface, model handling, prompt functions).

- main.py Entry point – The script you shared (loads model, defines functions, launches Gradio UI).
- model_utils.py Helper functions for model loading&text generation (to keep main.py clean).
- prompts.py Stores reusable prompt templates (e.g., disease prediction, treatment plan).
- model Local model cache if you want to save/download IBM Granite weights instead of pulling every run.

README.md Info on downloading or linking Hugging Face model.

- Requirements Dependency management.

requirements.txt Python libraries: transformers, torch, gradio, etc.

environment.yml Conda environment file if using Conda.

- static CSS, images, or custom frontend assets if you style Gradio.
- Notebooks Jupyter Colab notebooks for experimentation or data analysis
- tests Unit tests for generate_response, disease_prediction, etc.
- docs Documentation for setup, usage, and API details.

README.md Main project documentation (installation, running instructions).

- .env Environment variables (API keys, secrets if needed).

README.md Top-level project description and quick start guide.

LICENSE Project license (MIT, Apache, etc.).

- .gitignore Ignore cache files, model weights, and virtual environments.

6. Running the Application

python health_ai.py

1. Launch the Gradio interface (python health_ai.py).
2. Open the given localhost/Share link.
3. Navigate between **Disease Prediction** and **Treatment Plan** tabs.
4. Enter symptoms or patient details.
5. Receive AI-generated suggestions with medical disclaimer.

7. API Documentation

If integrated with FastAPI, the following APIs could be exposed:

- **POST /predict-symptoms** – Returns conditions & recommendations
- **POST /generate-plan** – Returns personalized treatment plan.

8. Authentication

Currently runs in **open mode** for demo. Future secure deployments can add:

- API key-based access
- Role-based usage (doctor vs. patient mode)

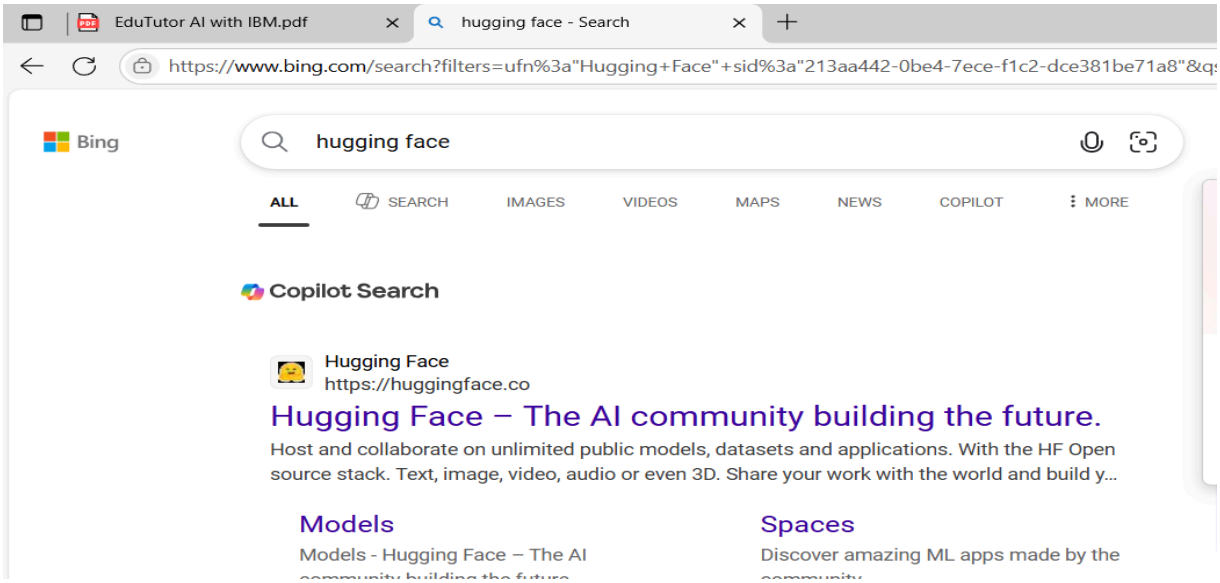
9. User Interface

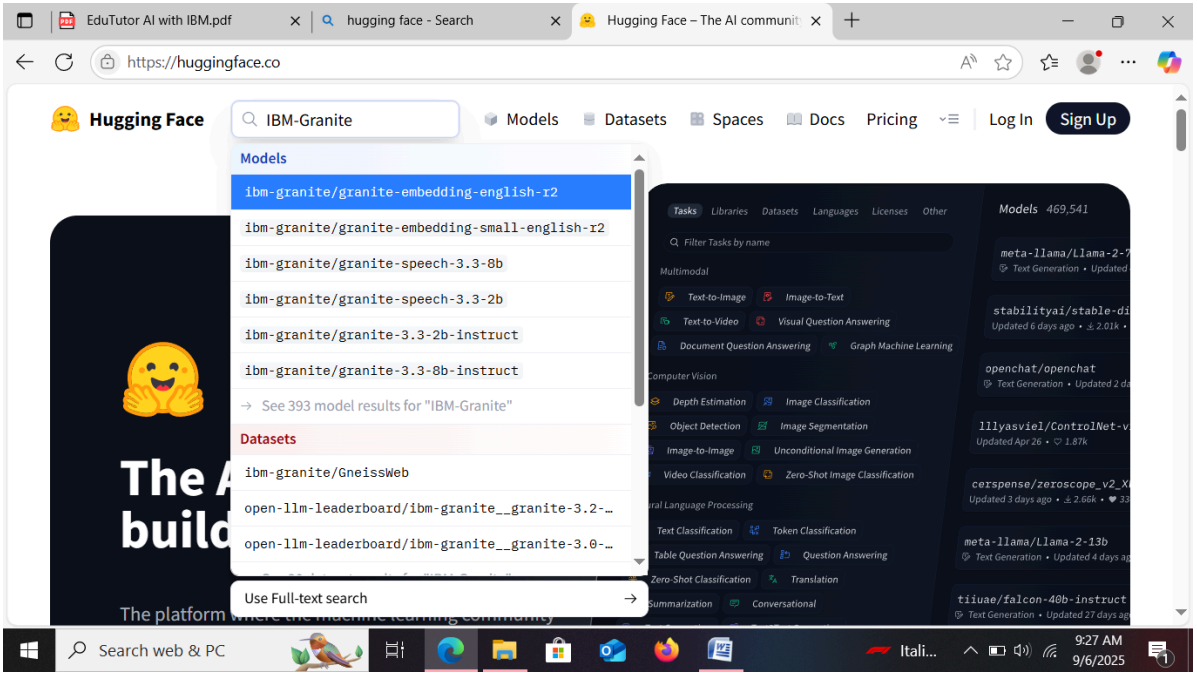
- **Tabs:** Disease Prediction | Treatment Plans
- **Inputs:** Textboxes, dropdowns, number inputs
- **Outputs:** Multi-line text with AI response
- **Design Priority:** Minimalist, clarity-first, with disclaimers at every step

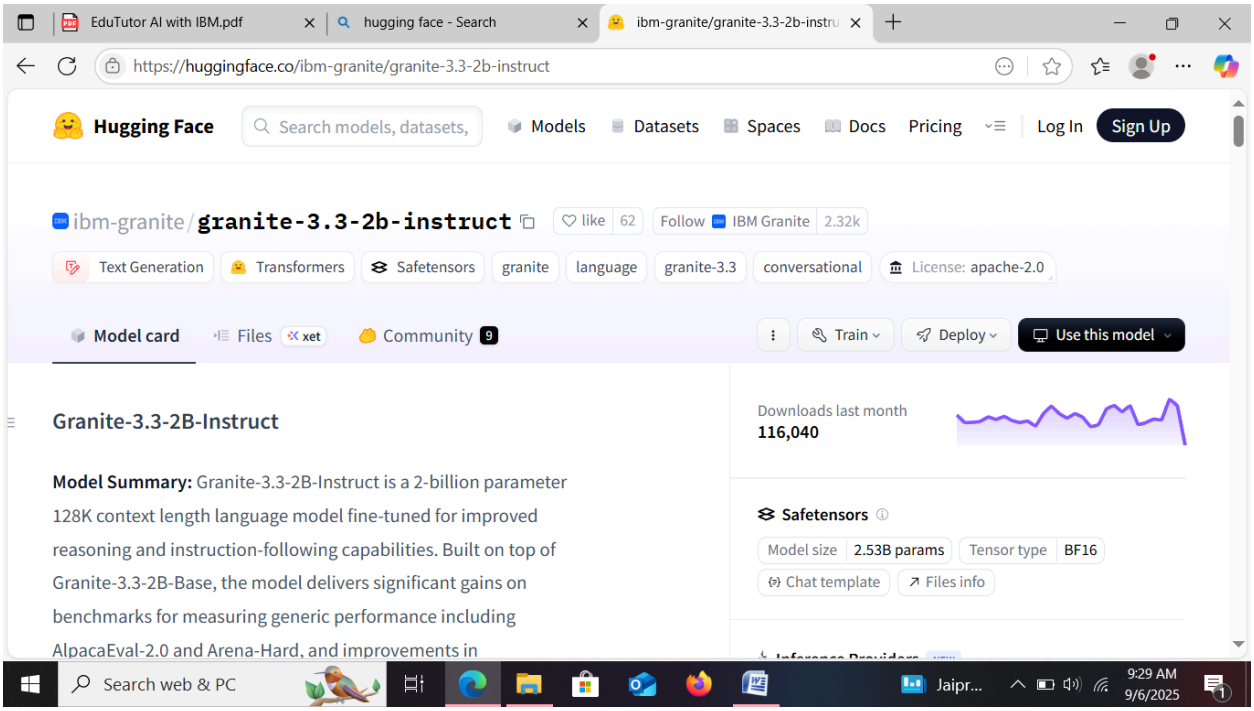
10. Testing

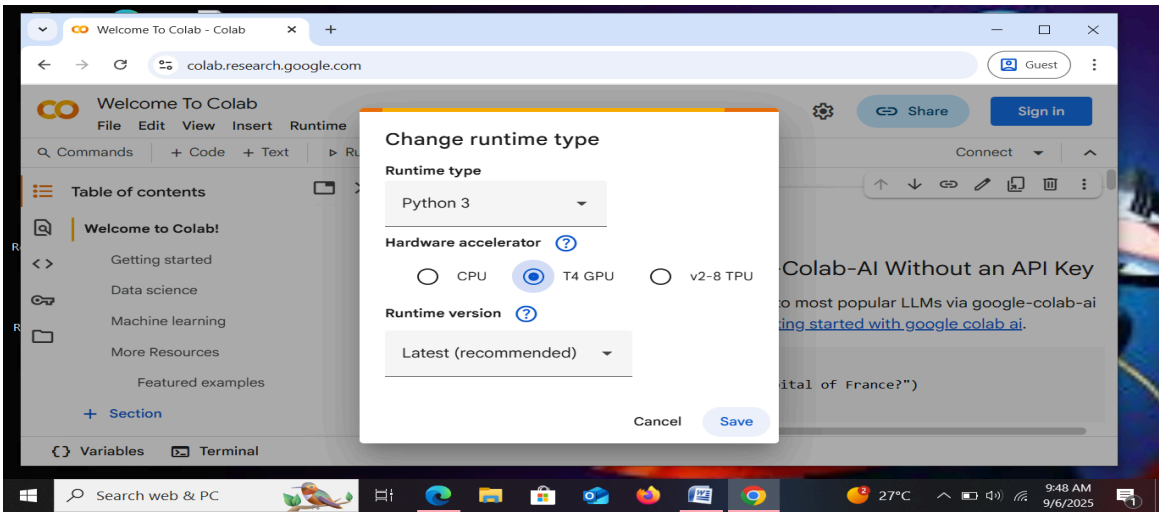
- **Unit Testing:** Prompt functions tested for valid AI responses
- **Manual Testing:** UI tested for different symptoms and conditions
- **Edge Cases:** Empty input, long symptom lists, invalid formats
- **Safety Validation:** Ensured disclaimer appears in every output

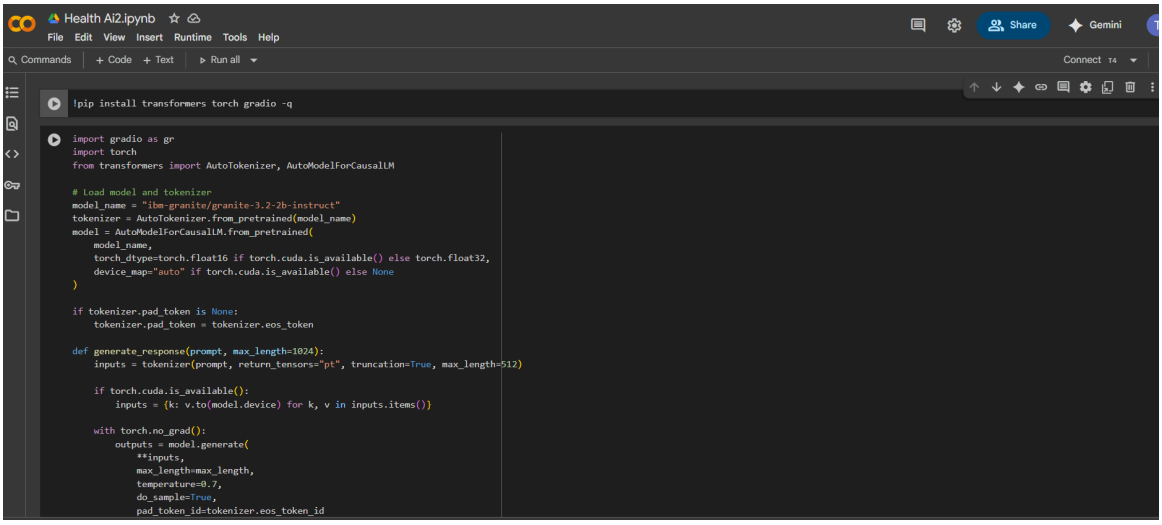
11. Screenshots

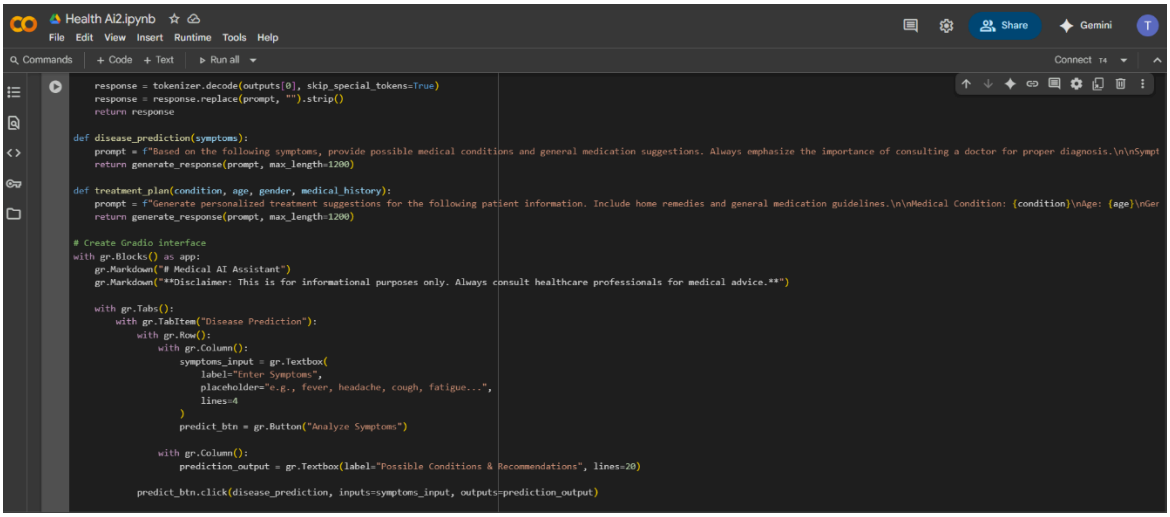

EduTutor AI with IBM.pdf
hugging face - Search
https://www.bing.com/search?filters=ufn%3a"Hugging+Face"+sid%3a"213aa442-0be4-7ece-f1c2-dce381be71a8"&q
Bing
hugging face
ALL
SEARCH
IMAGES
VIDEOS
MAPS
NEWS
COPILOT
MORE
Copilot Search
Hugging Face
https://huggingface.co
Hugging Face – The AI community building the future.
Host and collaborate on unlimited public models, datasets and applications. With the HF Open source stack. Text, image, video, audio or even 3D. Share your work with the world and build y...
Models
Models - Hugging Face – The AI
Spaces
Discover amazing ML apps made by the

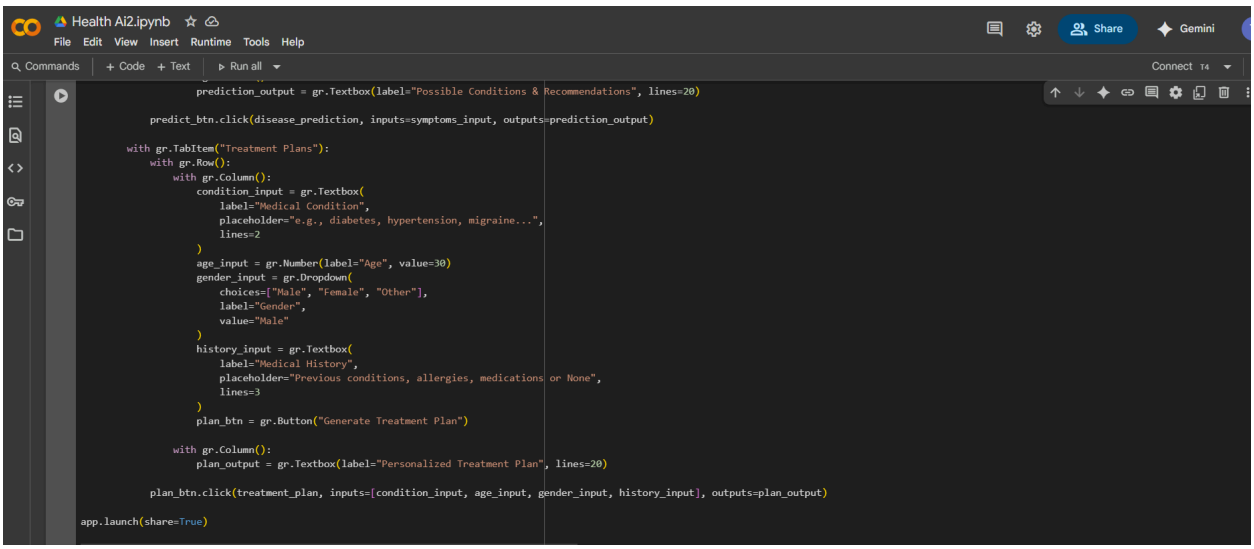

EduTutor AI with IBM.pdf
hugging face - Search
Hugging Face – The AI communit
https://huggingface.co
Hugging Face
IBM-Granite
Models
Datasets
Spaces
Docs
Pricing
Log In
Sign Up
Models
ibm-granite/granite-embedding-english-r2
ibm-granite/granite-embedding-small-english-r2
ibm-granite/granite-speech-3.3-8b
ibm-granite/granite-speech-3.3-2b
ibm-granite/granite-3.3-2b-instruct
ibm-granite/granite-3.3-8b-instruct
See 393 model results for "IBM-Granite"
Datasets
ibm-granite/GneissWeb
open-llm-leaderboard/ibm-granite__granite-3.2-...
open-llm-leaderboard/ibm-granite__granite-3.0-...
Use Full-text search
Models 469,541
meta-llama/Llama-2-13b
tiiuae/falcon-40b-instruct
Search web & PC
9:27 AM
9/6/2025

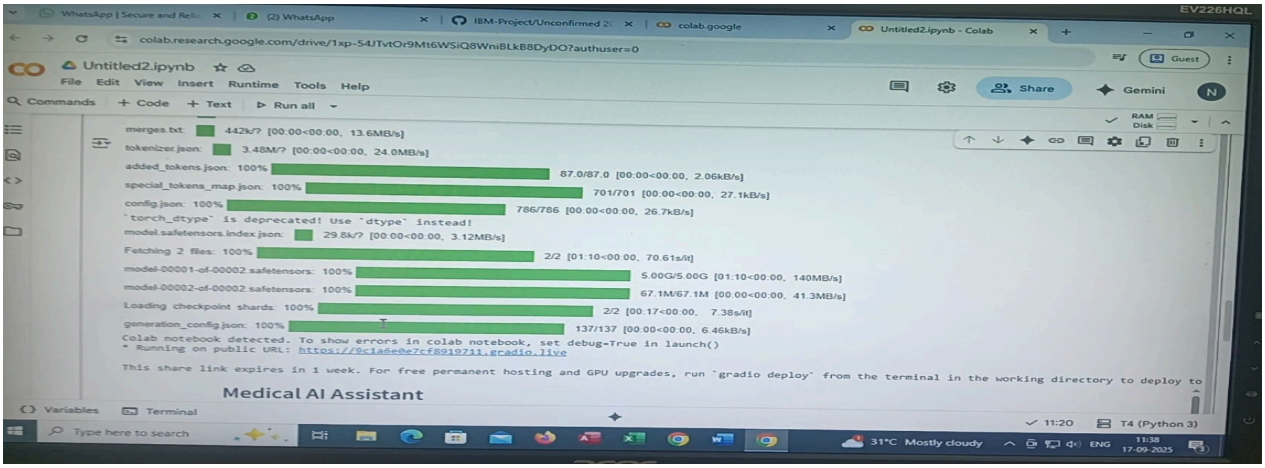

EduTutor AI with IBM.pdf
hugging face - Search
ibm-granite/granite-3.3-2b-instru
https://huggingface.co/ibm-granite/granite-3.3-2b-instruct
Hugging Face
Search models, datasets,
Models
Datasets
Spaces
Docs
Pricing
Log In
Sign Up
ibm-granite / granite-3.3-2b-instruct
like 62
Follow IBM Granite 2.32k
Text Generation
Transformers
Safetensors
granite
language
granite-3.3
conversational
License: apache-2.0
Model card
Files xet
Community 9
Train
Deploy
Use this model
Granite-3.3-2B-Instruct
Model Summary: Granite-3.3-2B-Instruct is a 2-billion parameter 128K context length language model fine-tuned for improved reasoning and instruction-following capabilities. Built on top of Granite-3.3-2B-Base, the model delivers significant gains on benchmarks for measuring generic performance including AlpacaEval-2.0 and Arena-Hard, and improvements in
Downloads last month
116,040
Safetensors
Model size 2.53B params
Tensor type BF16
Chat template
Files info
Search web & PC
Jaipr...
9:29 AM
9/6/2025

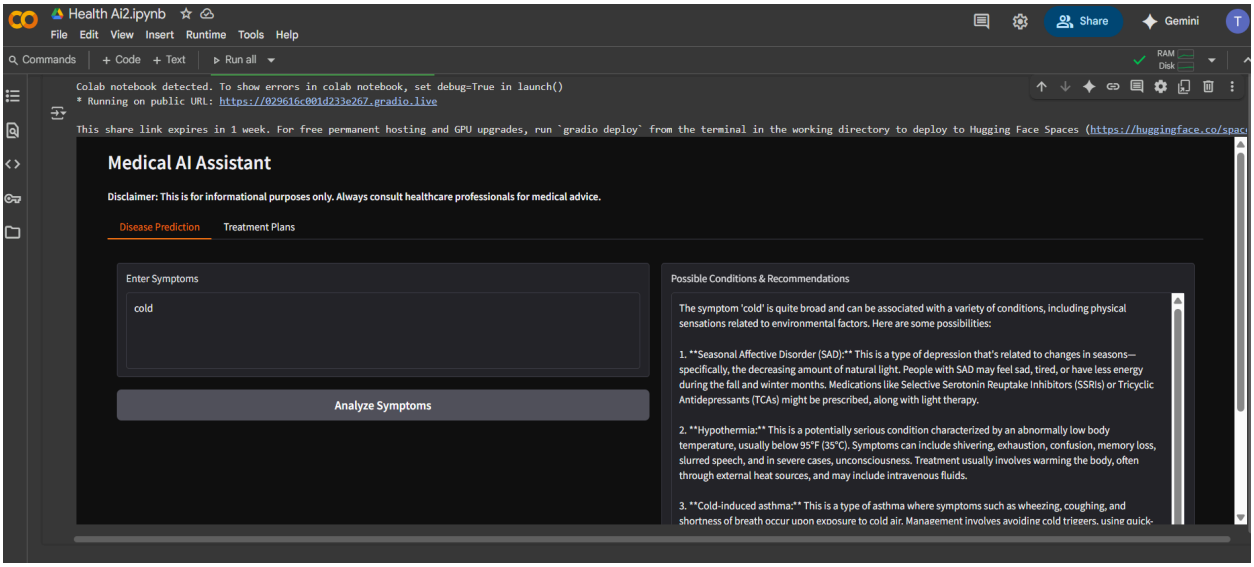

Welcome To Colab - Colab
colab.research.google.com
Guest
Welcome To Colab
File Edit View Insert Runtime
Share
Sign in
Commands
+ Code
+ Text
Connect
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More Resources
Featured examples
+ Section
Variables
Terminal
Change runtime type
Runtime type
Python 3
Hardware accelerator
CPU
T4 GPU
v2-8 TPU
Runtime version
Latest (recommended)
Cancel
Save
Colab-AI Without an API Key
ital of France?")
Search web & PC
27°C
9:48 AM
9/6/2025

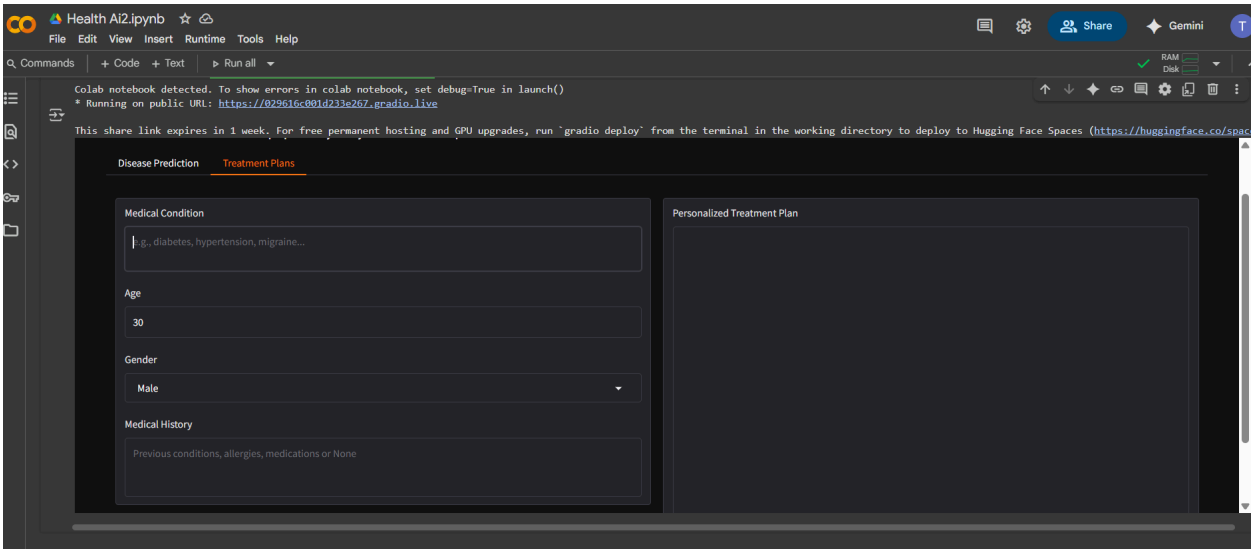

Health Ai2.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect T4
!pip install transformers torch gradio -q
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
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


Health Ai2.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect T4
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 consulting a doctor for proper diagnosis.\n\nSympt
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\nMedical Condition: {condition}\nAge: {age}\nGer
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
)
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)


Health Ai2.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect T4
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():
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)







12. Known Issues

- Responses may vary in accuracy since model is not a certified medical system.
- No real-time medical data integration yet.
- Limited medical terminology handling for rare conditions.

13. Future Enhancements

- Integration with **FastAPI** for API-based deployment
- **Medical knowledge base integration** (PubMed, WHO, etc.)
- **Speech-to-text support** for accessibility
- **Secure authentication** for patient data privacy
- **Multilingual support** for non-English speakers