Health AI: Intelligent Healthcare Assistant

Generative AI with IBM

Project Description:

HealthAI uses the Granite model from Hugging Face to deliver smart, easy-to-understand healthcare help. It includes Patient Chat, Disease Prediction, Treatment Plans, and adds more functionalities that you like . The project will be deployed in Google Colab using Granite for fast, accessible, and secure medical guidance.

Pre-requisites:

1. Gradio Framework Knowledge: Gradio Documentation
2. IBM Granite Models (Hugging Face): IBM Granite models
3. Python Programming Proficiency: Python Documentation
4. Version Control with Git: Git Documentation
5. Google Collab's T4 GPU Knowledge: Google collab

Project Workflow:

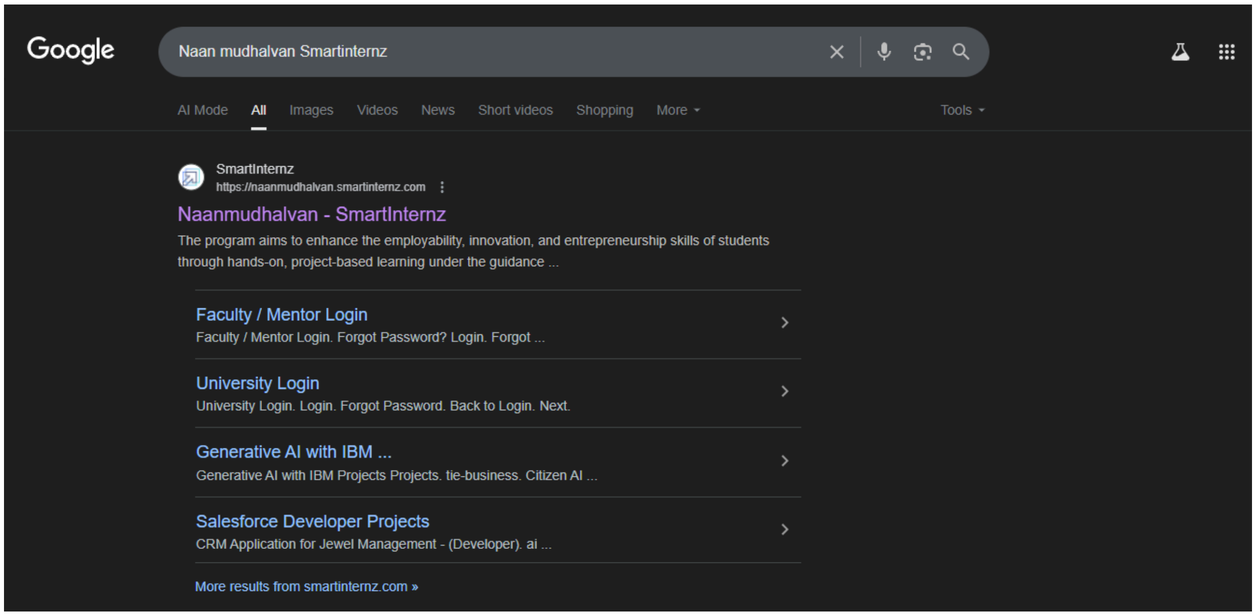
Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application In Google Colab.

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

- Search for “Naan Mudhalavan Smart Interz” Portal in any Browser.



SmartInternz
https://naanmudhalvan.smartinternz.com

Naanmudhalvan - SmartInternz

The program aims to enhance the employability, innovation, and entrepreneurship skills of students through hands-on, project-based learning under the guidance ...

Faculty / Mentor Login
Faculty / Mentor Login. Forgot Password? Login. Forgot ...

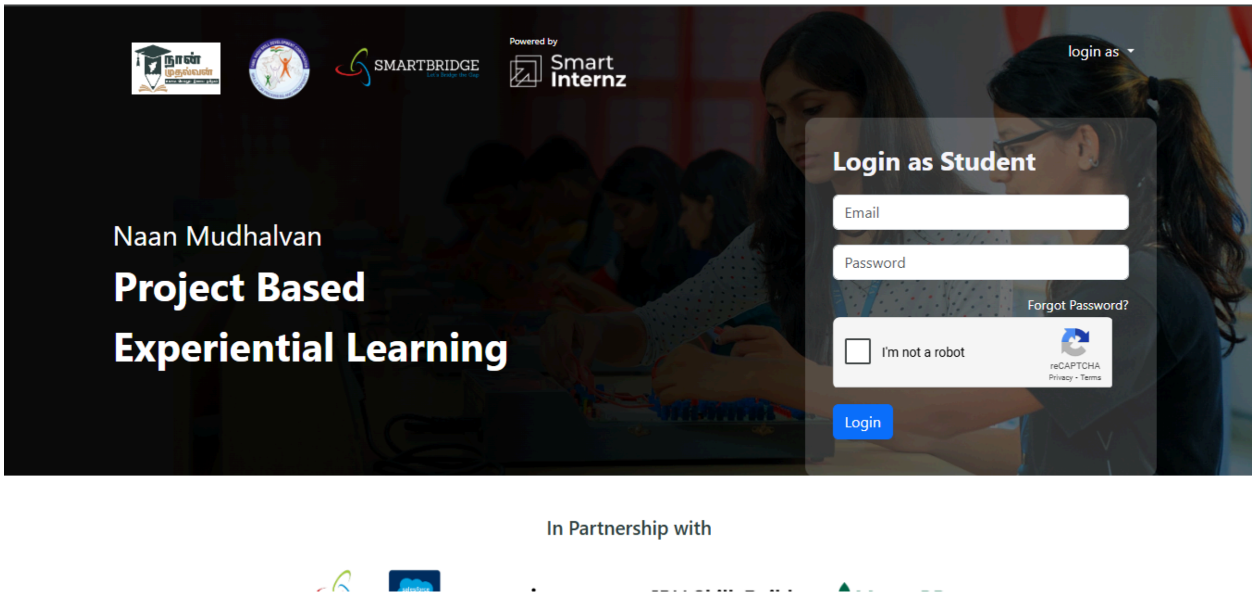
University Login
University Login. Login. Forgot Password. Back to Login. Next.

Generative AI with IBM ...
Generative AI with IBM Projects Projects. tie-business. Citizen AI ...

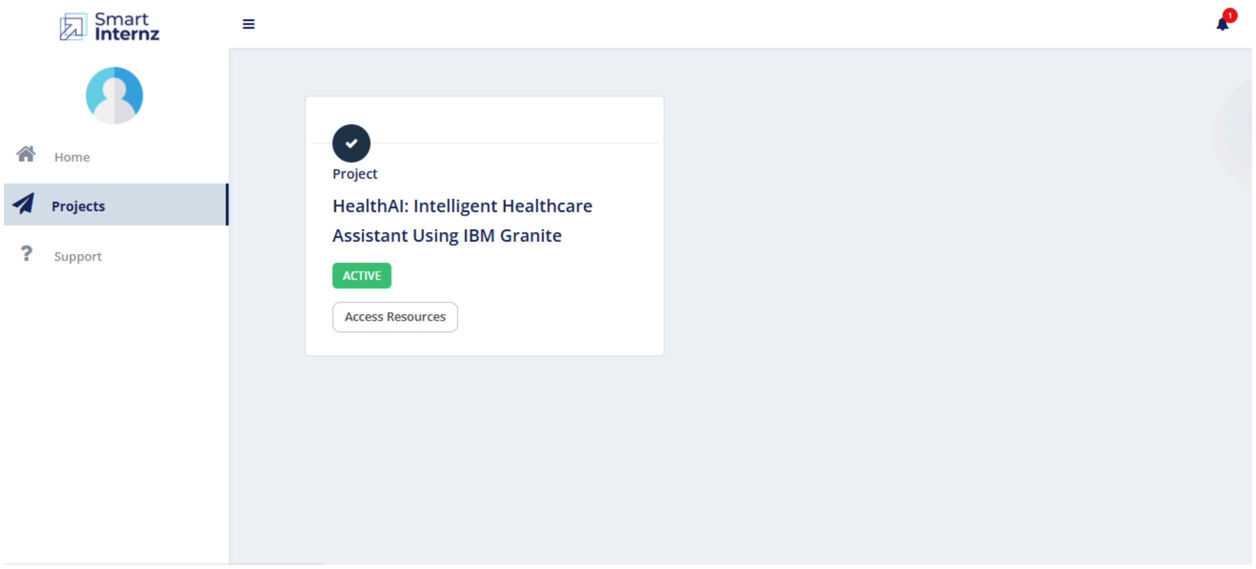
Salesforce Developer Projects
CRM Application for Jewel Management - (Developer). ai ...

More results from smartinternz.com »

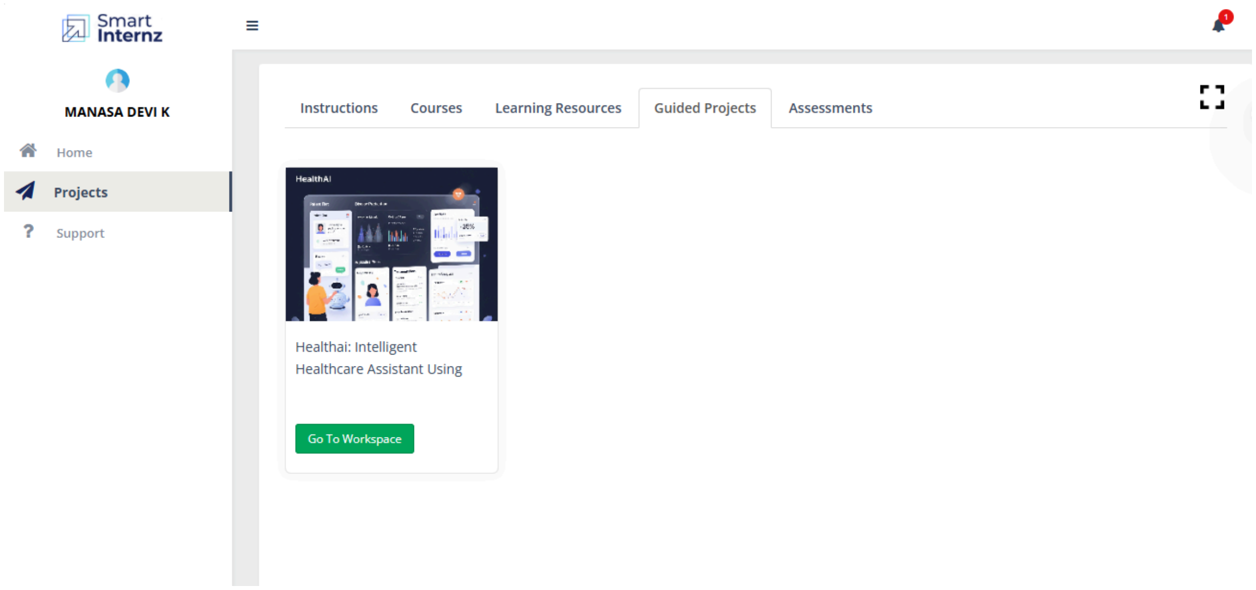
- Then Click on the first link. ([Naanmudhalvan Smartinternz]) Then login with your details.



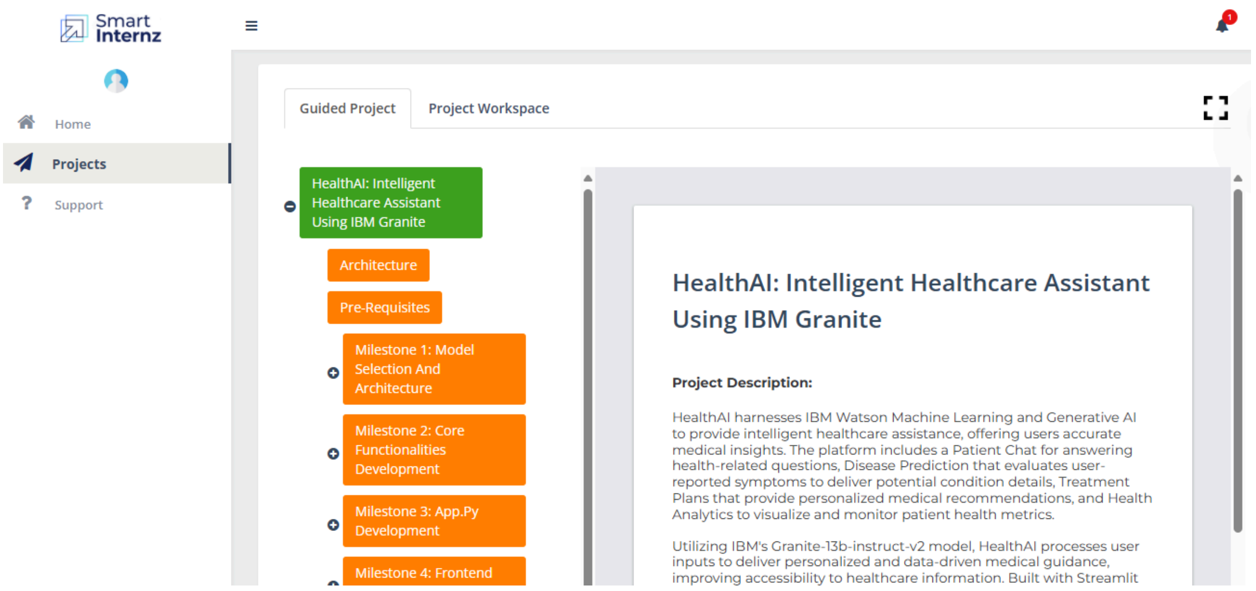
- Then you will be redirected to your account then click on “Projects” Section. There you can see which project you have enrolled in here it is “Health AI”.



- Then click on “Access Resources” and go to the “Guided Project” Section.



- Click on the “Go to workspace” section. Then you can find the detailed explanation of Generative AI Project using IBM Watson API key.


Smart Internz
Home
Projects
Support
Guided Project
Project Workspace
HealthAI: Intelligent Healthcare Assistant Using IBM Granite
Architecture
Pre-Requisites
Milestone 1: Model Selection And Architecture
Milestone 2: Core Functionalities Development
Milestone 3: App.Py Development
Milestone 4: Frontend

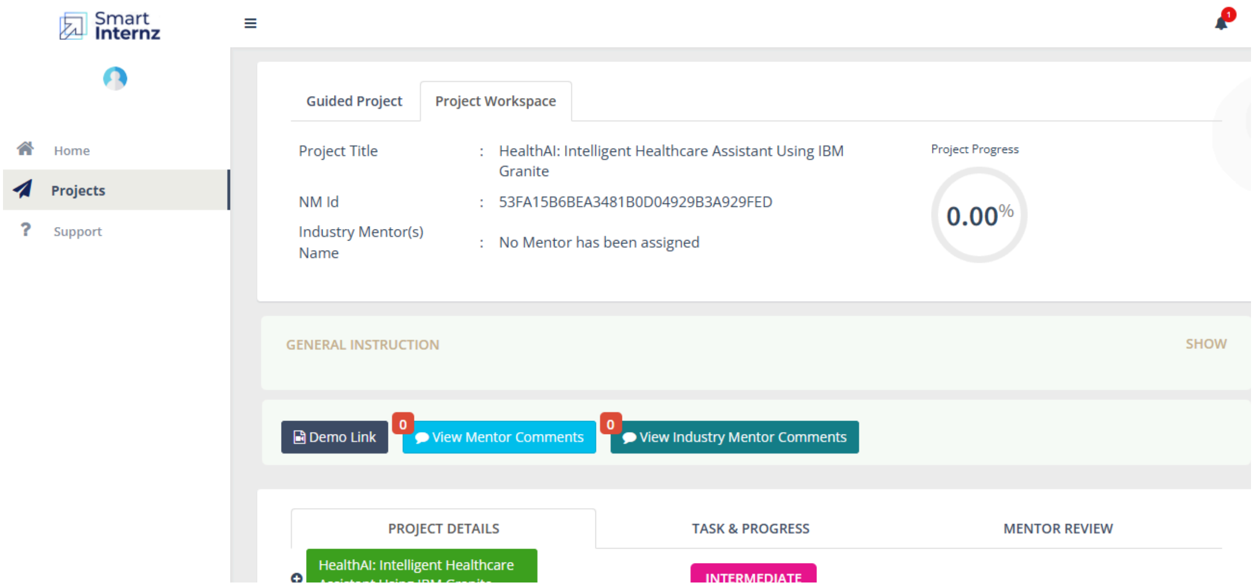
HealthAI: Intelligent Healthcare Assistant Using IBM Granite

Project Description:

HealthAI harnesses IBM Watson Machine Learning and Generative AI to provide intelligent healthcare assistance, offering users accurate medical insights. The platform includes a Patient Chat for answering health-related questions, Disease Prediction that evaluates user-reported symptoms to deliver potential condition details, Treatment Plans that provide personalized medical recommendations, and Health Analytics to visualize and monitor patient health metrics.

Utilizing IBM's Granite-13b-instruct-v2 model, HealthAI processes user inputs to deliver personalized and data-driven medical guidance, improving accessibility to healthcare information. Built with Streamlit

- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.

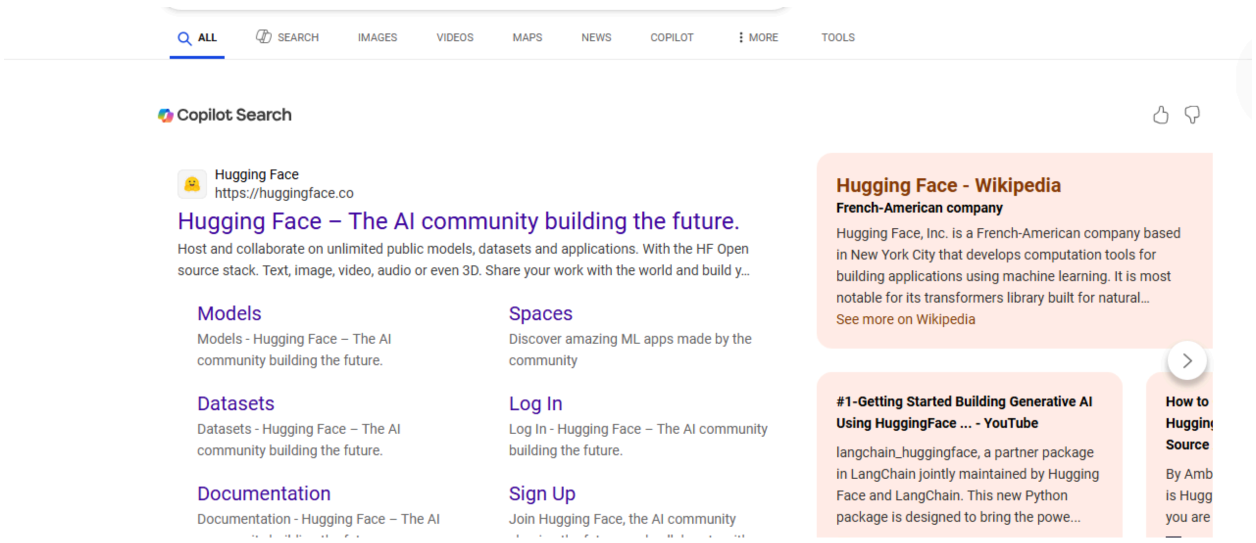


Project Title	:	HealthAI: Intelligent Healthcare Assistant Using IBM Granite
NM Id	:	53FA15B6BEA3481B0D04929B3A929FED
Industry Mentor(s) Name	:	No Mentor has been assigned

- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

Activity-2: Choose a IBM Granite model From Hugging Face.

- Search for "Hugging face" in any browser.



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 community building the future.

Spaces
Discover amazing ML apps made by the community

Datasets
Datasets - Hugging Face – The AI community building the future.

Log In
Log In - Hugging Face – The AI community building the future.

Documentation
Documentation - Hugging Face – The AI

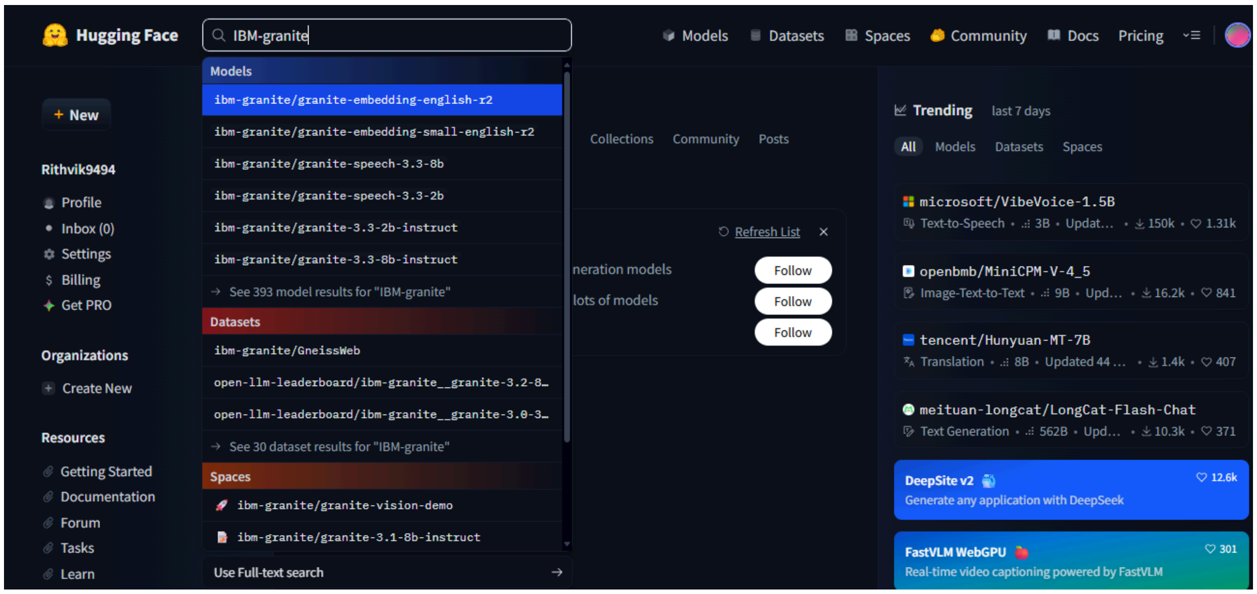
Sign Up
Join Hugging Face, the AI community

Hugging Face - Wikipedia
French-American company
Hugging Face, Inc. is a French-American company based in New York City that develops computation tools for building applications using machine learning. It is most notable for its transformers library built for natural...
See more on Wikipedia

#1-Getting Started Building Generative AI Using HuggingFace ... - YouTube
langchain_huggingface, a partner package in LangChain jointly maintained by Hugging Face and LangChain. This new Python package is designed to bring the powe...

How to Huggin Source
By Amb is Hugg you are

- Then click on the first link ([Hugging Face](https://huggingface.co)), then click on signup and create your own account in Hugging Face. Then search for "IBM-Granite models" and choose any model.



IBM-granite

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-3.2-8...
open-llm-leaderboard/ibm-granite__granite-3.0-3...
→ See 30 dataset results for "IBM-granite"

Spaces
ibm-granite/granite-vision-demo
ibm-granite/granite-3.1-8b-instruct

neration models
lots of models

microsoft/VibeVoice-1.5B
Text-to-Speech • 3B • Updat... • 150k • 1.31k

openbmb/MiniCPM-V-4_5
Image-Text-to-Text • 9B • Upd... • 16.2k • 841

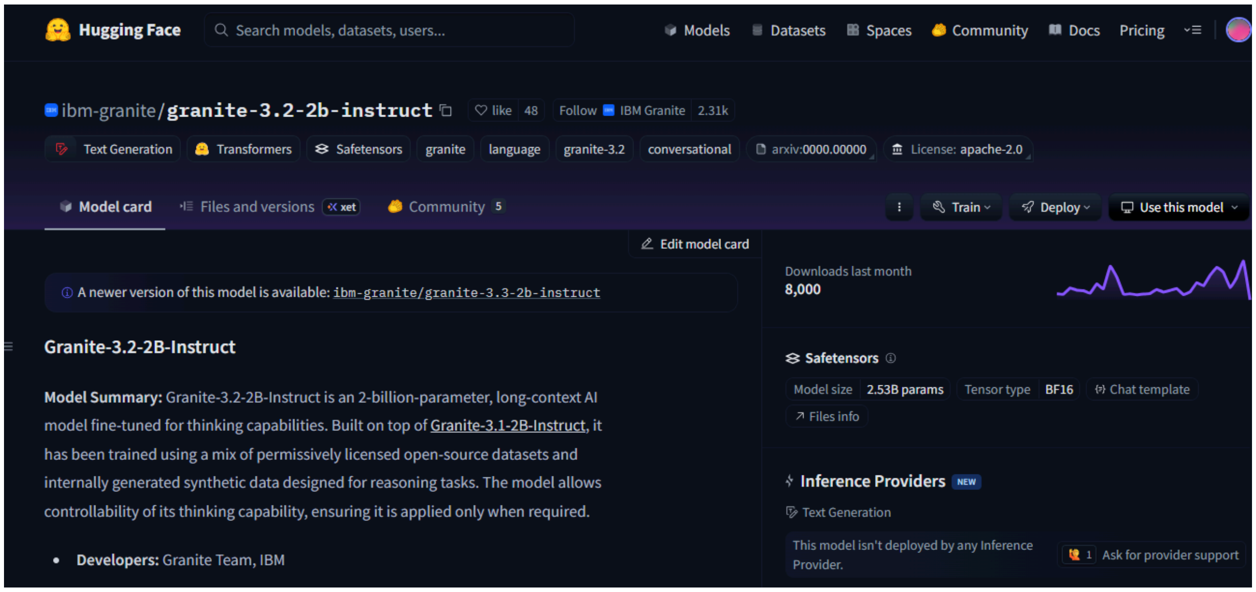
tencent/Hunyuan-MT-7B
Translation • 8B • Updated 44 ... • 1.4k • 407

meituan-longcat/LongCat-Flash-Chat
Text Generation • 562B • Upd... • 10.3k • 371

DeepSite v2 — 12.6k
Generate any application with DeepSeek

FastVLM WebGPU — 301
Real-time video captioning powered by FastVLM

- Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.



A newer version of this model is available: ibm-granite/granite-3.3-2b-instruct

Granite-3.2-2B-Instruct

Model Summary: Granite-3.2-2B-Instruct is an 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. Built on top of Granite-3.1-2B-Instruct, it has been trained using a mix of permissively licensed open-source datasets and internally generated synthetic data designed for reasoning tasks. The model allows controllability of its thinking capability, ensuring it is applied only when required.

- **Developers:** Granite Team, IBM

Downloads last month 8,000

Safetensors

Model size 2.53B params Tensor type BF16 Chat template

Files info

Inference Providers NEW

Text Generation

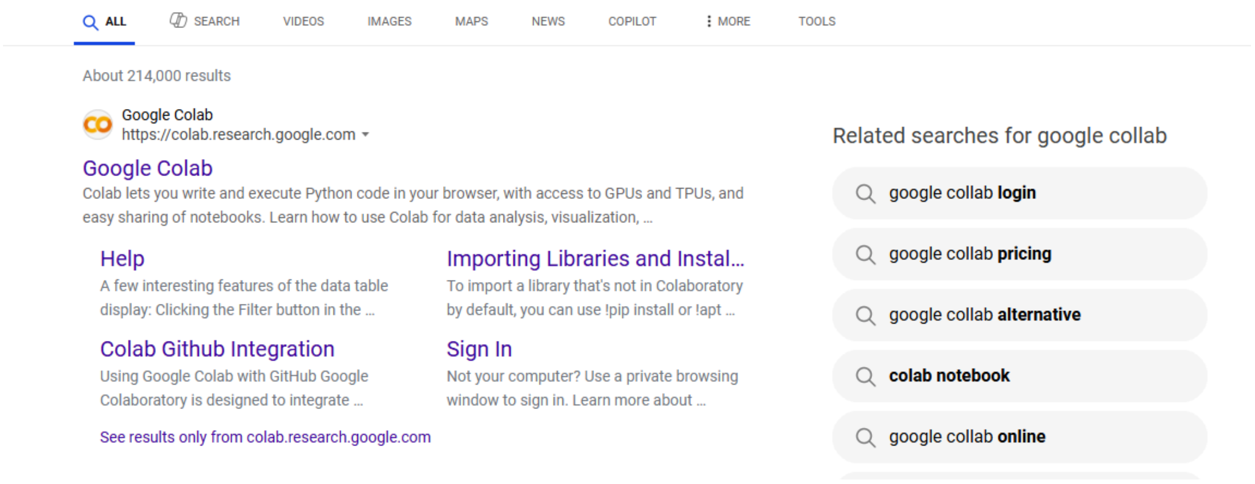
This model isn't deployed by any Inference Provider.

1 Ask for provider support

- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

- Search for “Google collab” in any browser.



About 214,000 results

Google Colab
https://colab.research.google.com

Google Colab
Colab lets you write and execute Python code in your browser, with access to GPUs and TPUs, and easy sharing of notebooks. Learn how to use Colab for data analysis, visualization, ...

Help
A few interesting features of the data table display: Clicking the Filter button in the ...

Importing Libraries and Instal...
To import a library that's not in Colaboratory by default, you can use !pip install or !apt ...

Colab Github Integration
Using Google Colab with GitHub Google Colaboratory is designed to integrate ...

Sign In
Not your computer? Use a private browsing window to sign in. Learn more about ...

See results only from colab.research.google.com

Related searches for google collab

google collab **login**

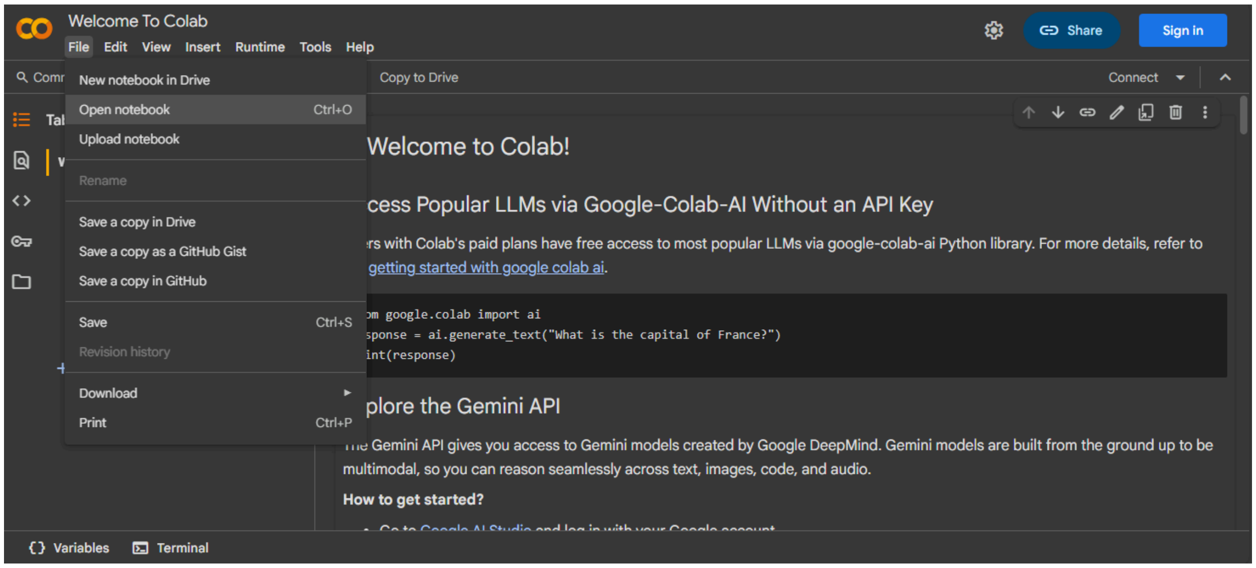
google collab **pricing**

google collab **alternative**

colab notebook

google collab **online**

- Click on the first link (Google Colab), then click on “Files” and then “Open Notebook”.



Welcome to Colab!

cess Popular LLMs via Google-Colab-AI Without an API Key

rs with Colab's paid plans have free access to most popular LLMs via google-colab-ai Python library. For more details, refer to getting started with google colab ai.

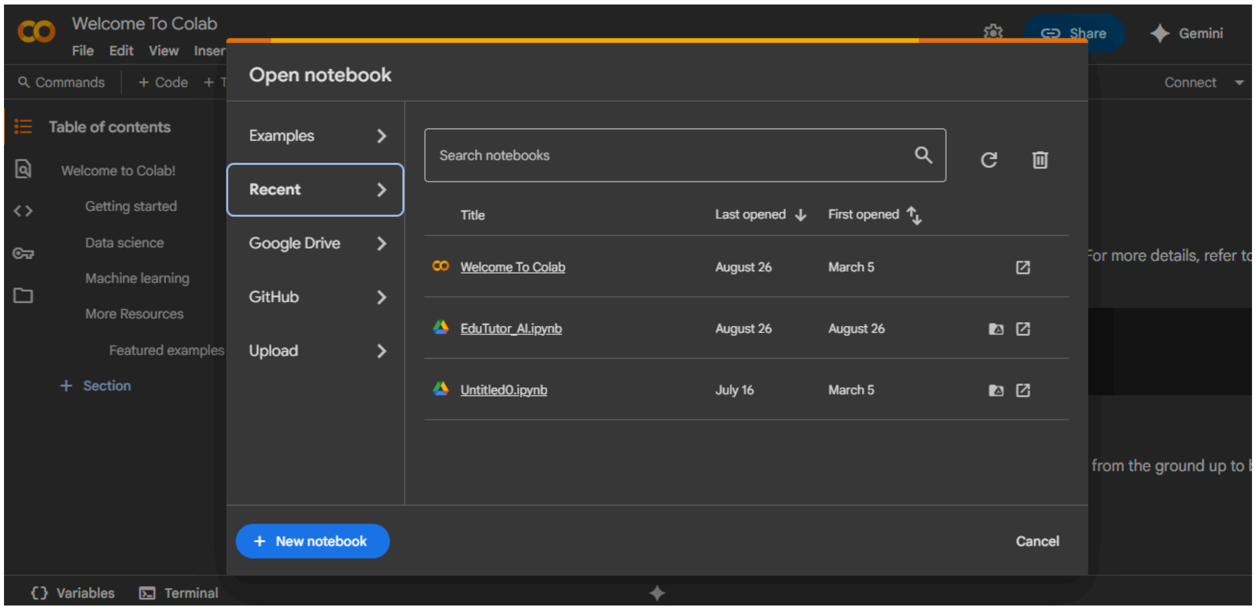
```
om google.colab import ai
sponse = ai.generate_text("What is the capital of France?")
int(response)
```

plore the Gemini API

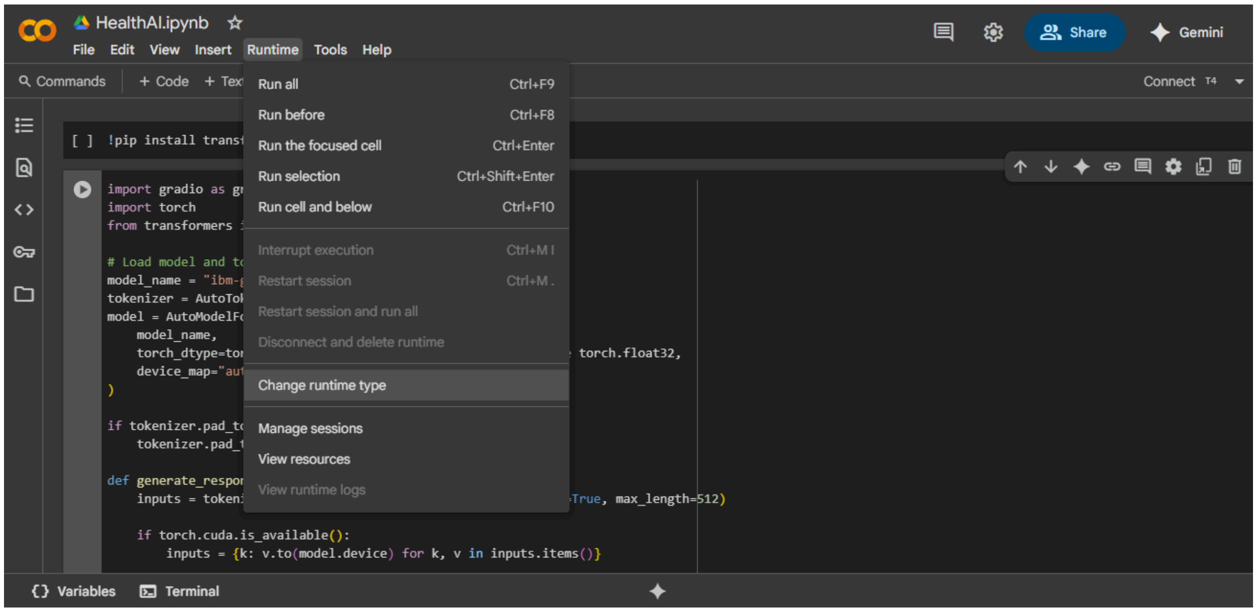
The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started?

- Click on “New Notebook”

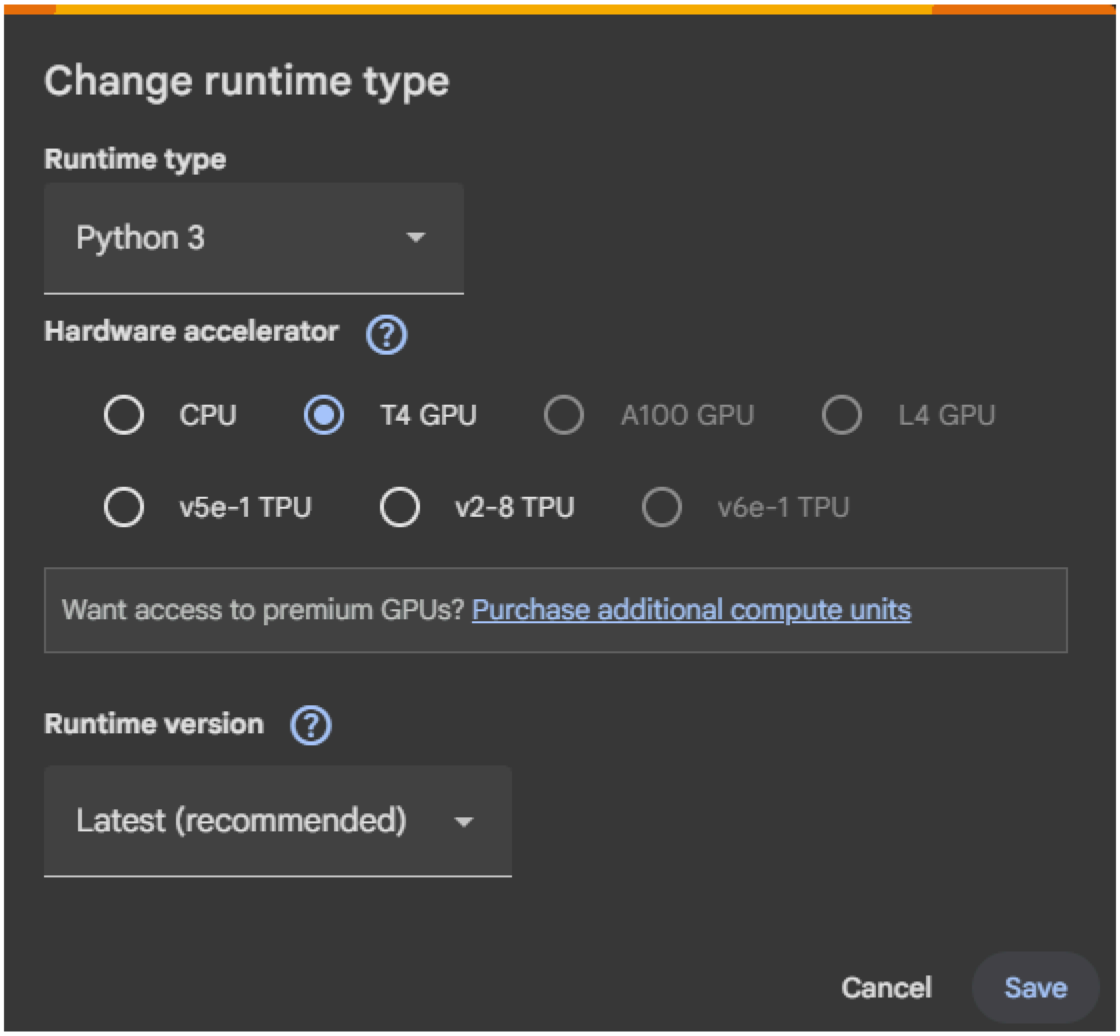


- Change the title of the notebook “Untitled” to “Health AI”. Then click on “Runtime”, then go to “Change Runtime Type”.


HealthAI.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Tex
Run all Ctrl+F9
Run before Ctrl+F8
Run the focused cell Ctrl+Enter
Run selection Ctrl+Shift+Enter
Run cell and below Ctrl+F10
Interrupt execution Ctrl+M I
Restart session Ctrl+M .
Restart session and run all
Disconnect and delete runtime
Change runtime type
Manage sessions
View resources
View runtime logs
Share
Gemini
Connect T4
Variables
Terminal

```
[ ] !pip install trans

import gradio as gr
import torch
from transformers i

# Load model and to
model_name = "ibm-g
tokenizer = AutoTok
model = AutoModelFo
    model_name,
    torch_dtype=tor                                  torch.float32,
    device_map="aut
)

if tokenizer.pad_to
    tokenizer.pad_t

def generate_respon
    inputs = tokeni                                  True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}
```

- Choose "T4 GPU" and click on "Save"



Change runtime type

Runtime type: Python 3

Hardware accelerator

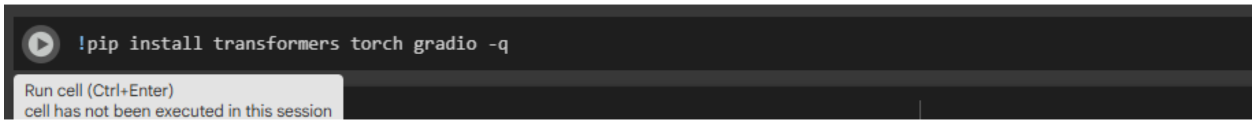
( ) CPU [✓ T4 GPU] ( ) A100 GPU ( ) L4 GPU

( ) v5e-1 TPU ( ) v2-8 TPU ( ) v6e-1 TPU

Want access to premium GPUs? Purchase additional compute units

Runtime version: Latest (recommended)

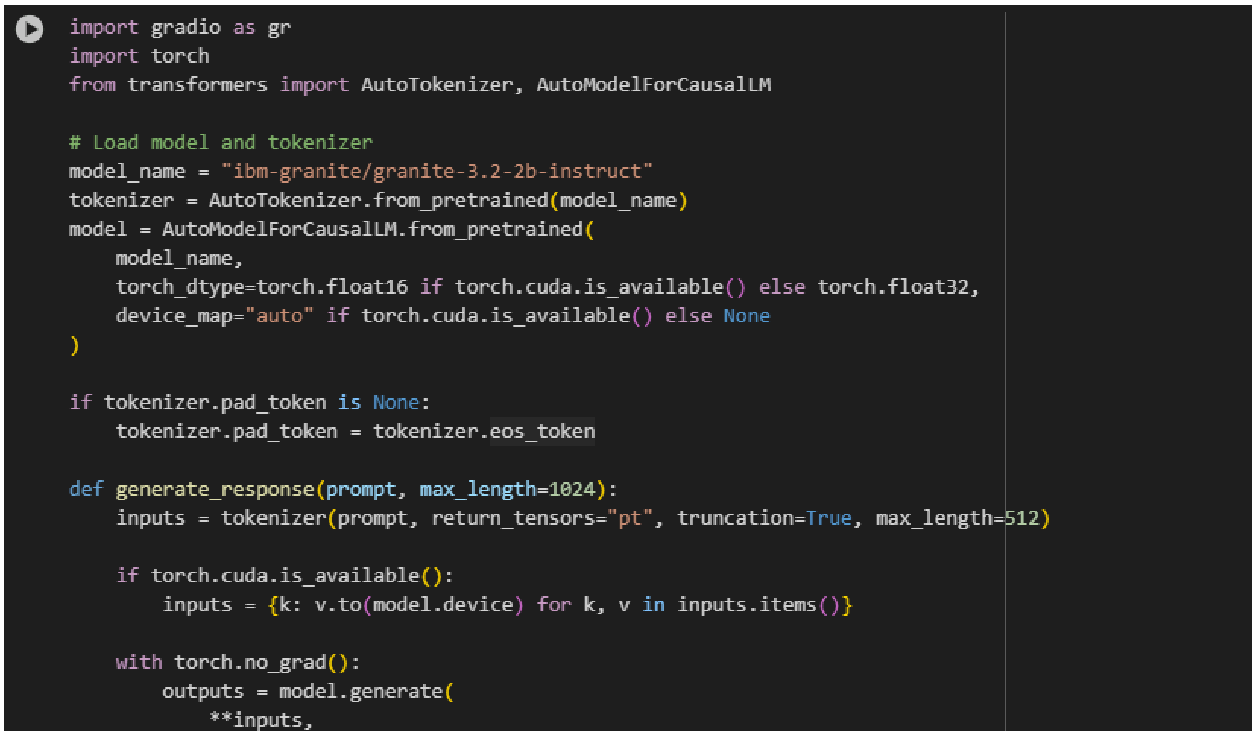
- Then run this command on the first cell "`!pip install transformers torch gradio -q`". To install the required libraries to run our application.



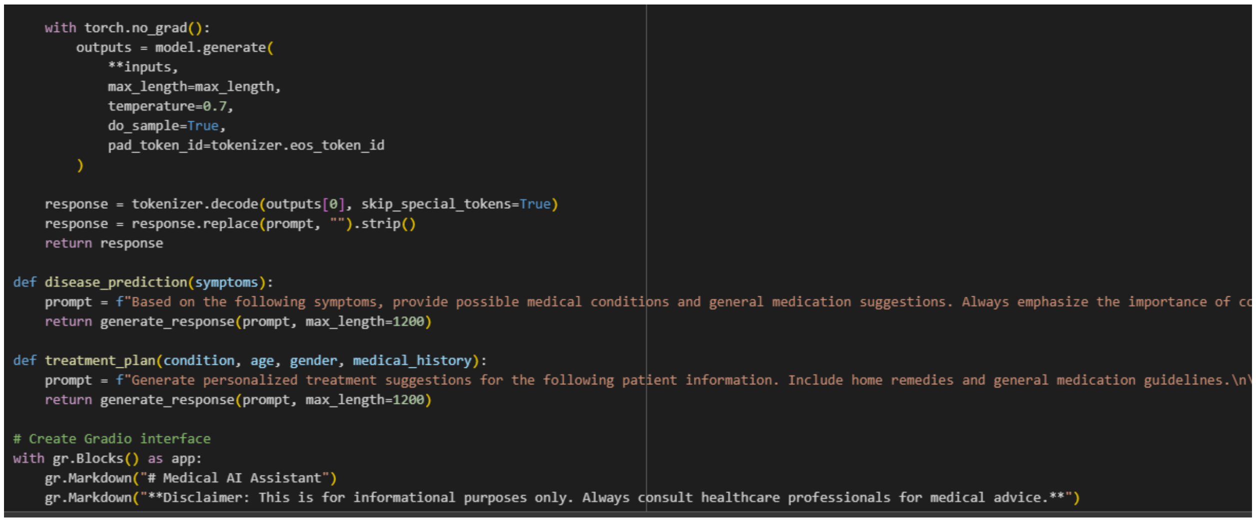
```
!pip install transformers torch gradio -q
```

Run cell (Ctrl+Enter)
cell has not been executed in this session

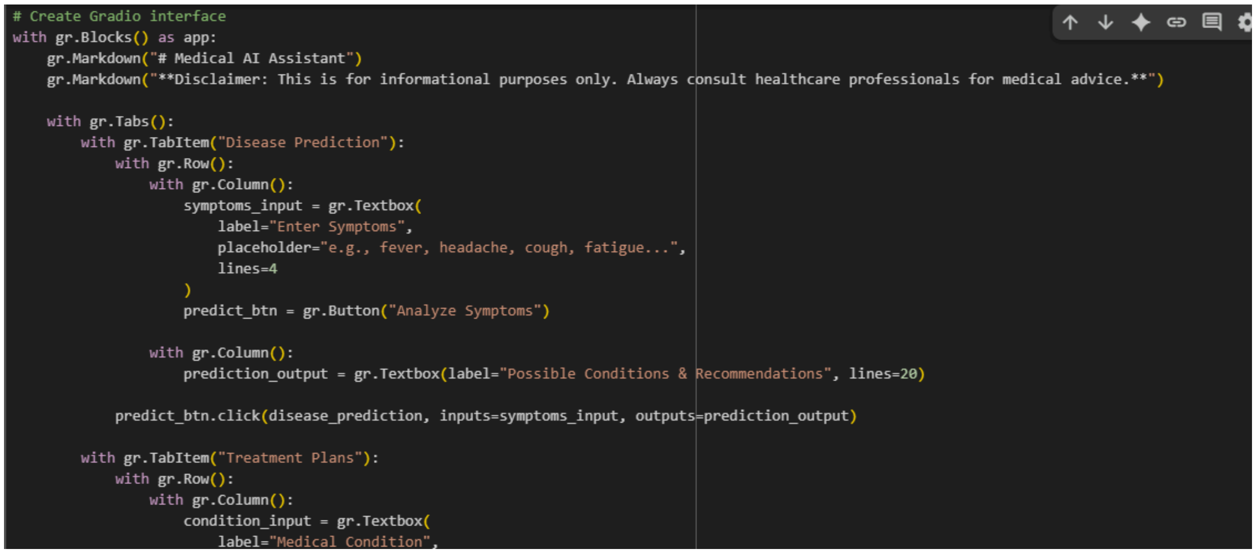
- Then run the rest of the code in the single cell.



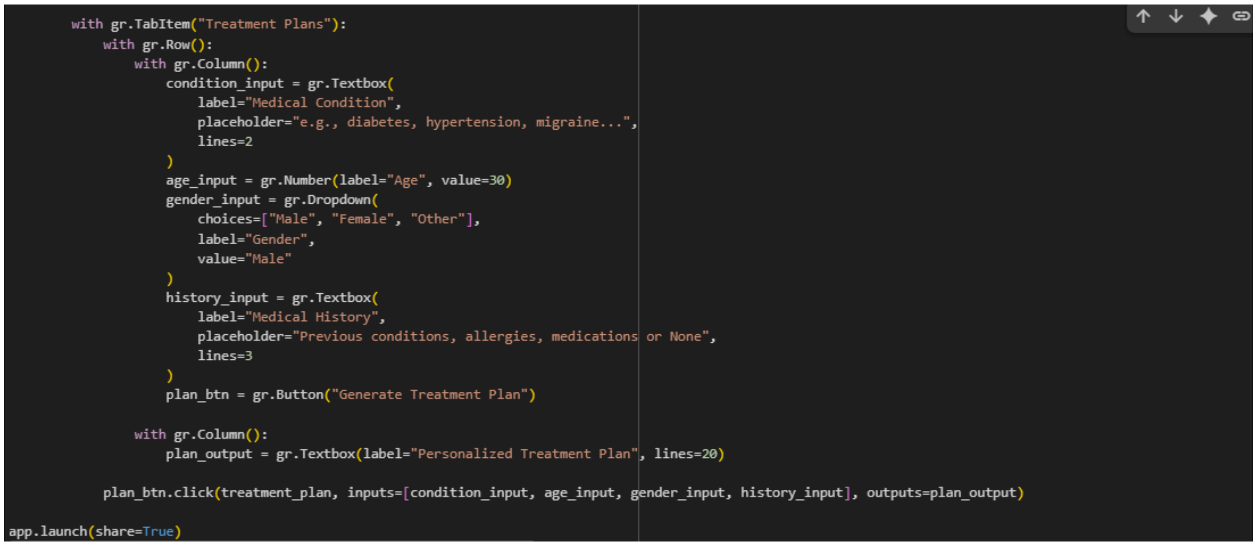
```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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,
```



```python
    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
        )

    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 co
    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.**")
```



```
# 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)

        with gr.TabItem("Treatment Plans"):
            with gr.Row():
                with gr.Column():
                    condition_input = gr.Textbox(
                        label="Medical Condition",
```



```
        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)
```

- You can find the code here in this link: [HealthAI Code]