HEALTH AI: INTELLIGENT HEALTHCARE ASSISTANT

GENERATIVE AI WITH IBM

TEAM MEMBERS:

1. *JESSY D*
2. *SONASRI S*
3. *KALAISELVI M*
4. *DEENADAYALAN K*

INTRODUCTION :

In recent years, the integration of Artificial Intelligence (AI) into healthcare has gained significant attention due to its potential to improve patient care, streamline clinical workflows, and support medical decision-making. The Health AI Assistant project is designed as an intelligent healthcare support system that leverages the power of IBM Granite models from Hugging Face to provide smart, interactive, and user-friendly healthcare guidance. The objective of this project is not to replace doctors, but rather to assist patients and healthcare providers by offering reliable information, disease prediction, treatment suggestions, and a responsive patient chat system.

The proposed system uses Generative AI models to understand user queries and respond in simple, accessible language. It is developed using Google Colab as the primary environment for training and deployment, along with the Gradio framework for creating an interactive web-based interface. By combining these technologies, the project demonstrates how cloud-based AI solutions can be made available to a wider audience, offering scalability, accessibility, and real-time interaction.

Furthermore, the project workflow emphasizes hands-on learning and practical implementation. Students explore resources via the Naan Mudhalvan Smart Interz portal, select suitable Granite models from Hugging Face, and configure their applications in Google Colab with GPU support. Once the model is integrated and tested, the project is uploaded to GitHub for version control and collaboration. This structured workflow ensures that students gain exposure not only to AI concepts but also to essential development practices such as cloud deployment, version management, and collaborative coding.

The Health AI Assistant aligns with the growing need for digital healthcare tools, especially in rural and under-resourced regions where access to professional medical advice may be limited. By offering an AI-powered conversational assistant, the project highlights how technology can bridge gaps in healthcare delivery, support early disease detection, and provide patients with preliminary treatment information. Although it is not a substitute for clinical consultation, this project serves as a valuable demonstration of how AI, healthcare, and cloud technologies can come together to address real-world challenges in a safe and educational manner.

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 add more functionalities that you like . The project is 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 WORK OW:

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

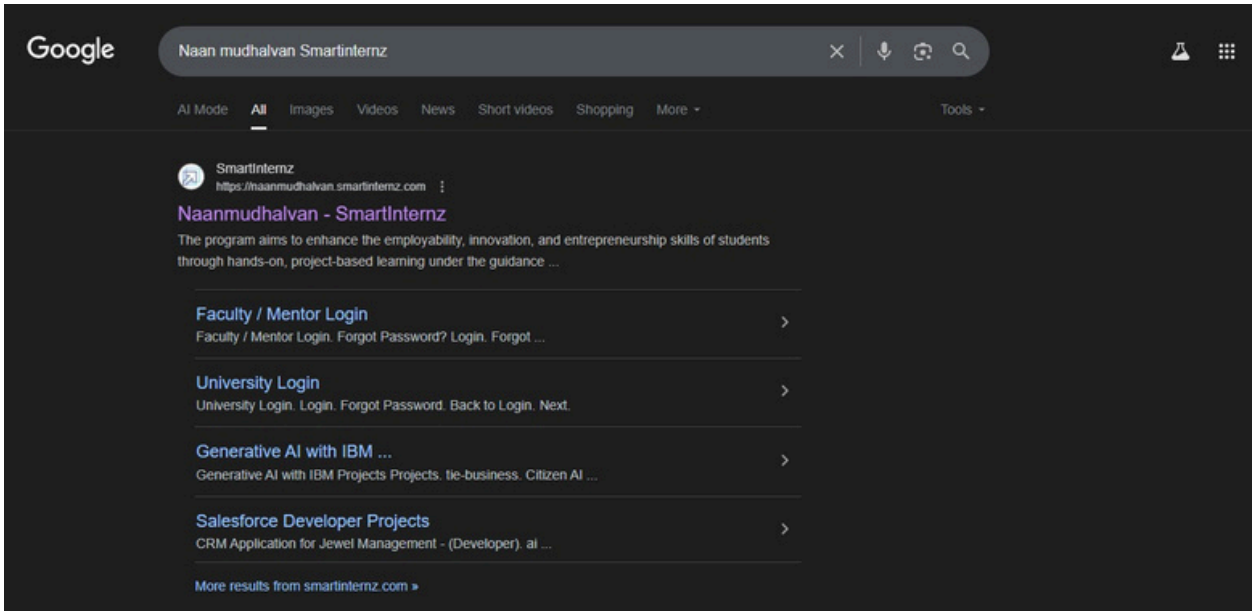
Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application In Google Colab. Activity-4:

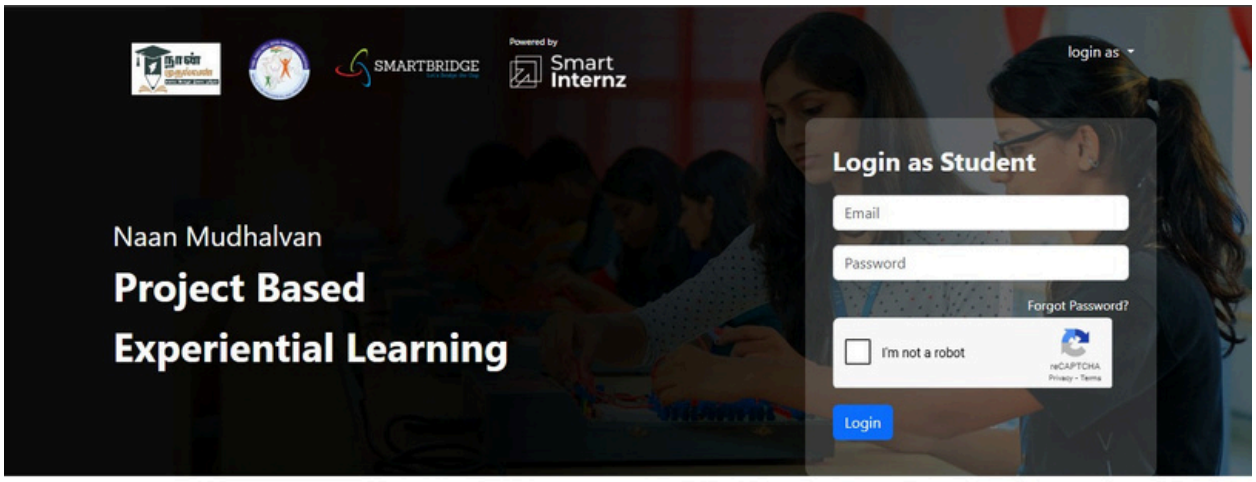
Upload your Project in Github.

ACTIVITY-1: EXPLORING NAAN MUDHALAVAN SMART INTERZ PORTAL.

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

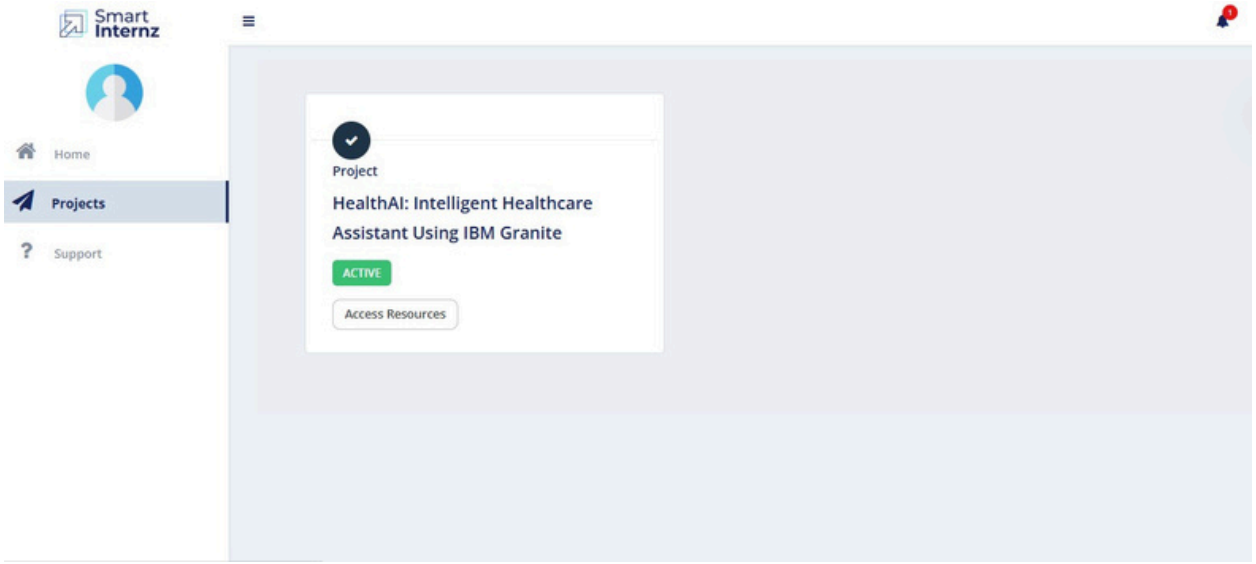


- Then Click on the first link. (Naanmudhalvan Smartinternz) Then login with your details.

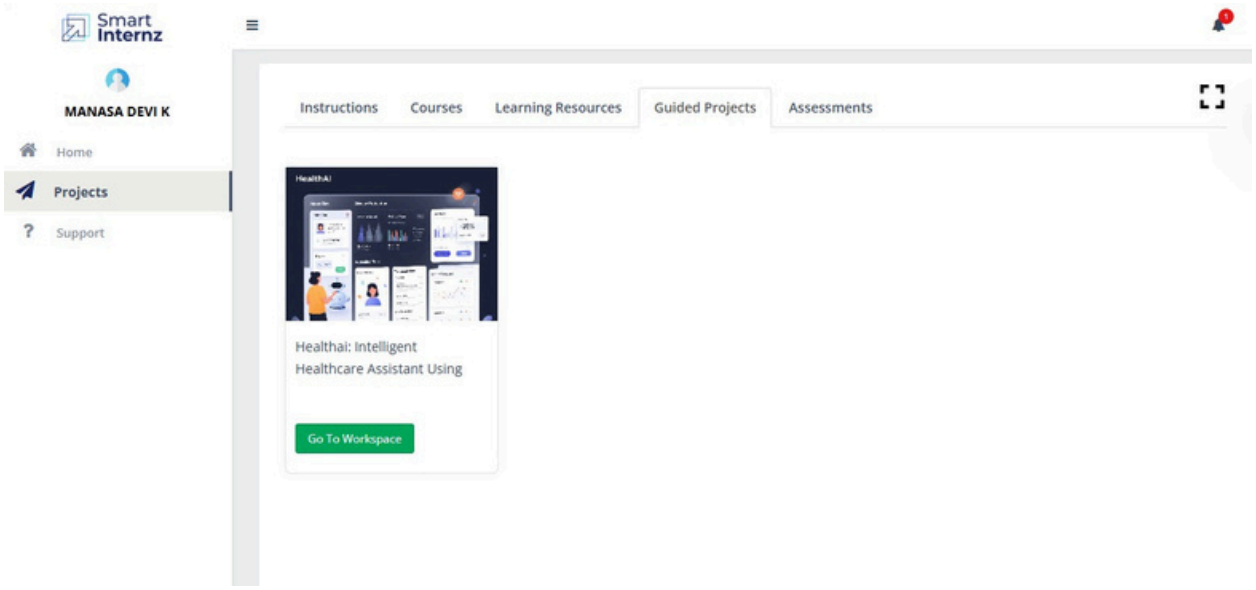


In Partnership with

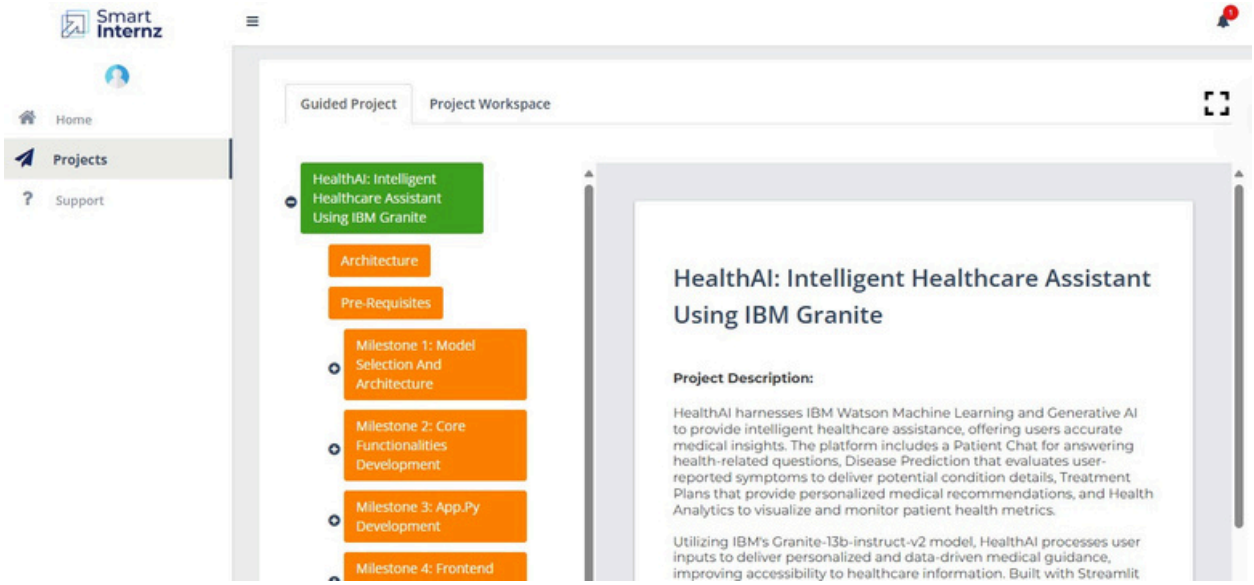
- Then you will be redirected to your account then click on “Projects” Section. There you can see which project you have enrolled in here it is “Health AI”.



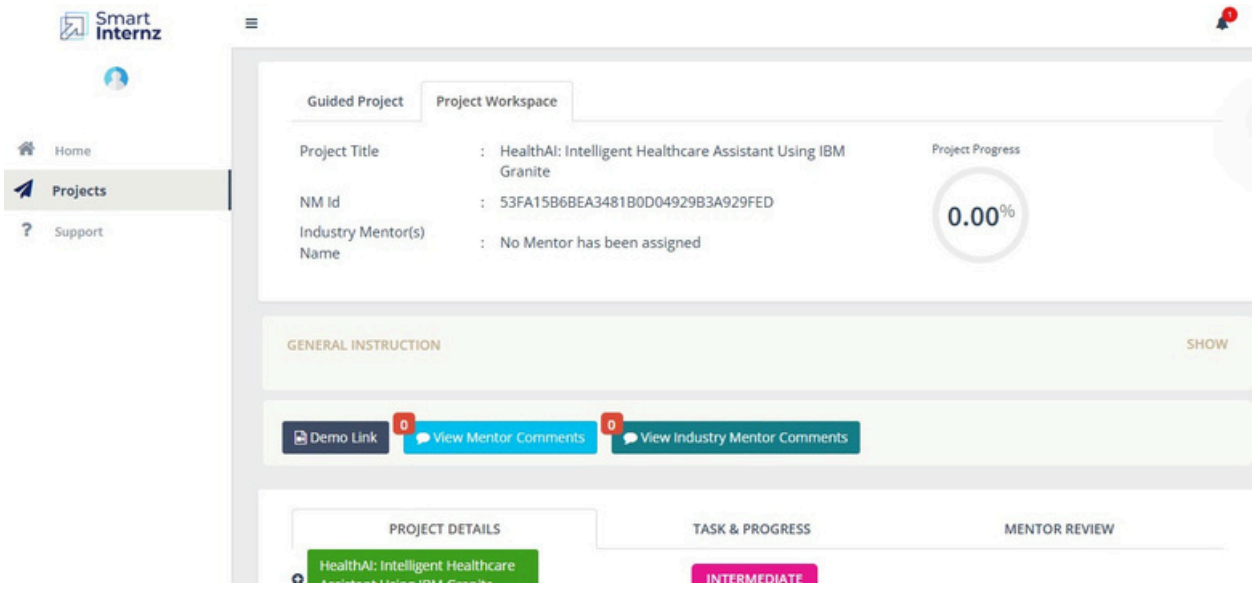
- Then click on “Access Resources” and go to the “Guided Project” Section.



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



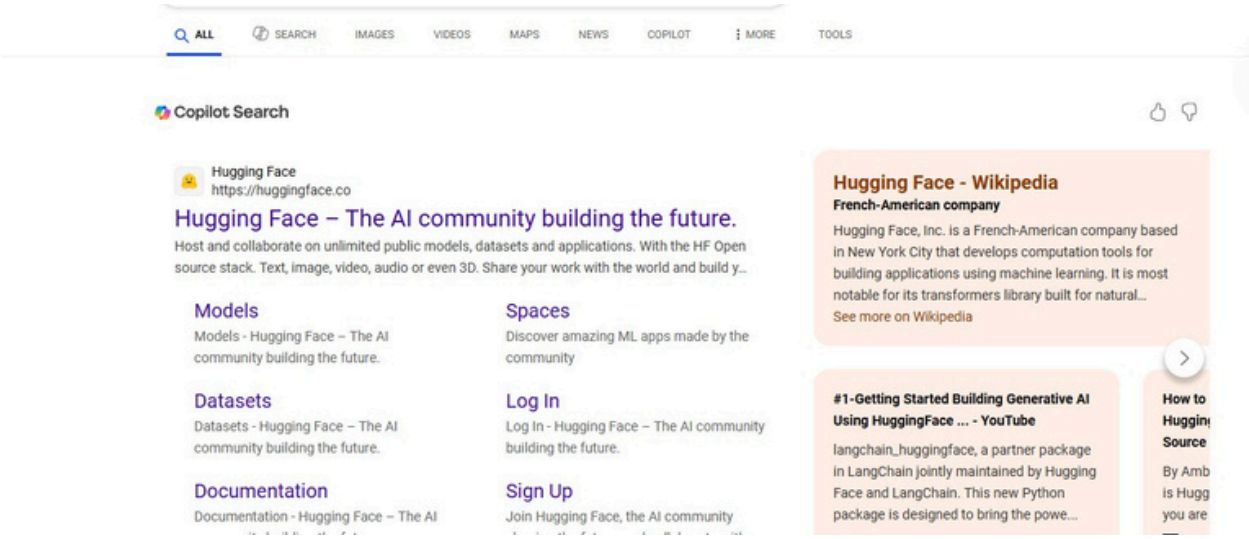
- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.



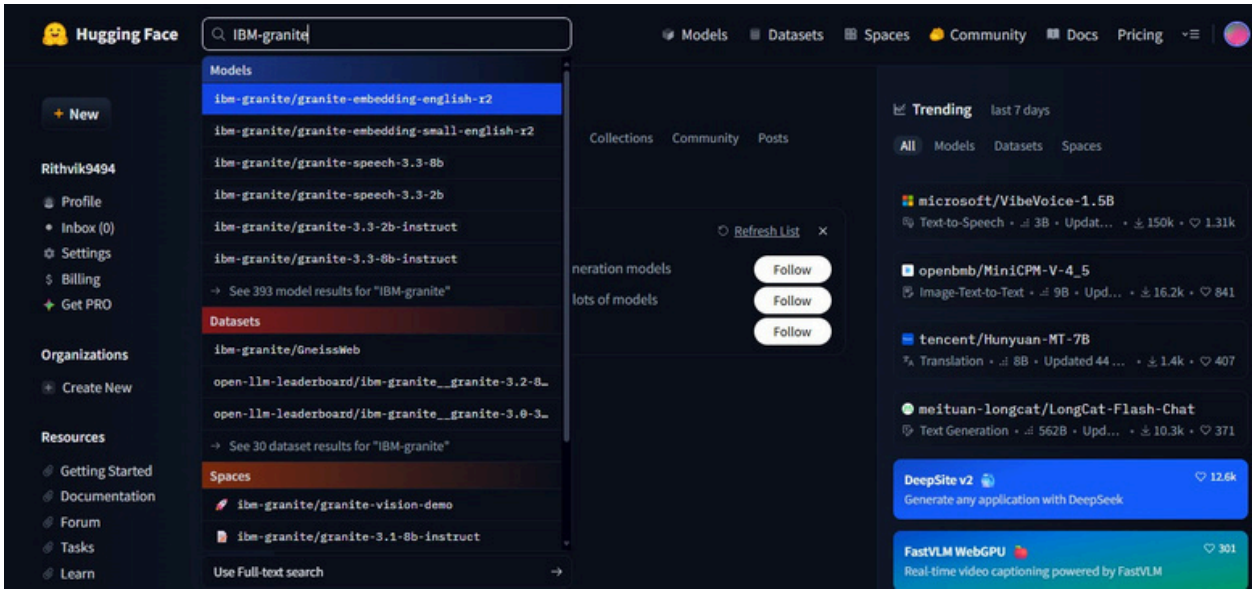
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

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

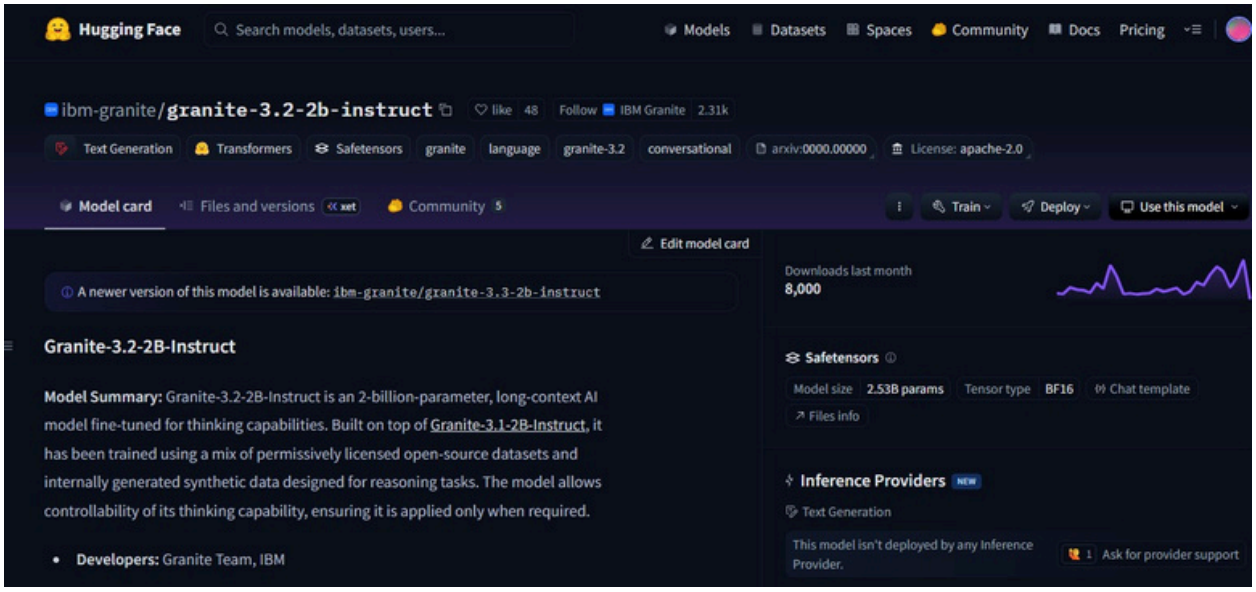
- Search for “Hugging face” in any browser.



- Then click on the first link (Hugging Face), then click on signup and create your own account in Hugging Face. Then search for “IBM-Granite models” and choose any model.



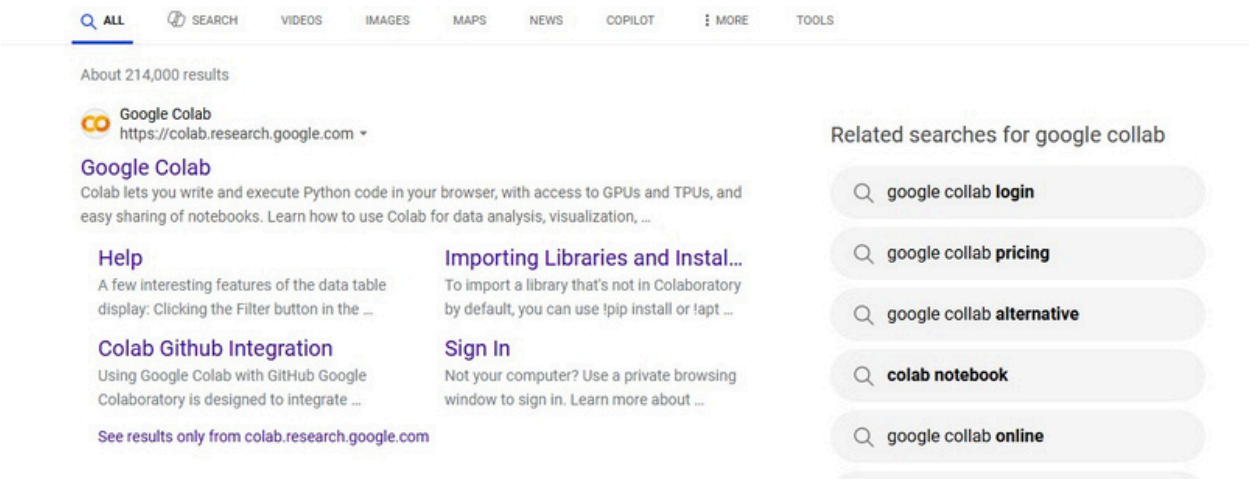
- Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.



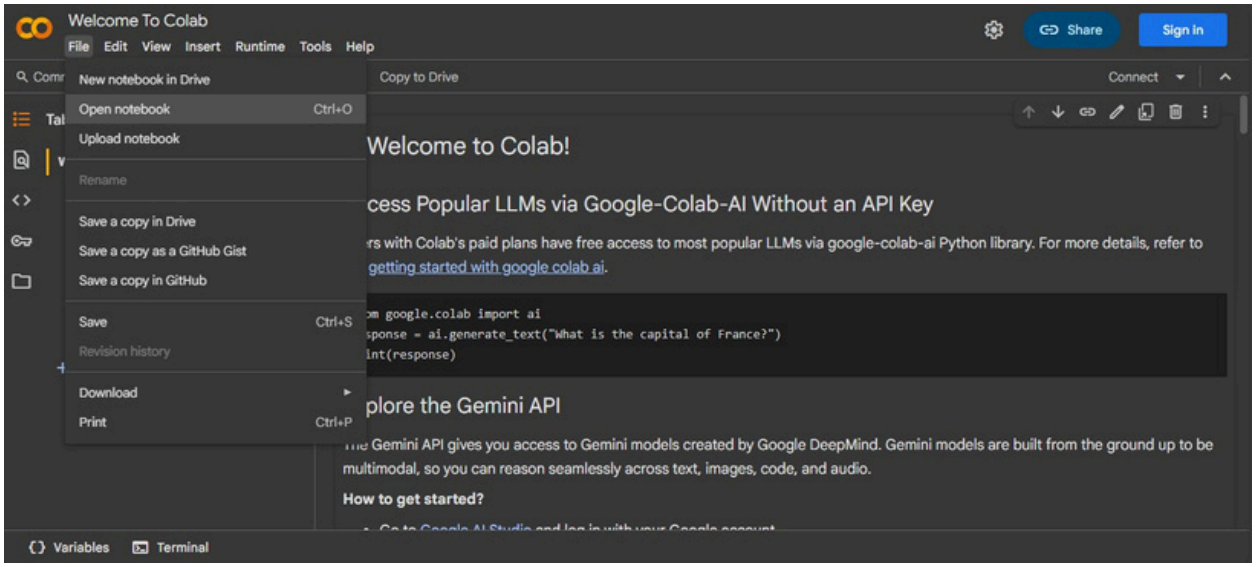
- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

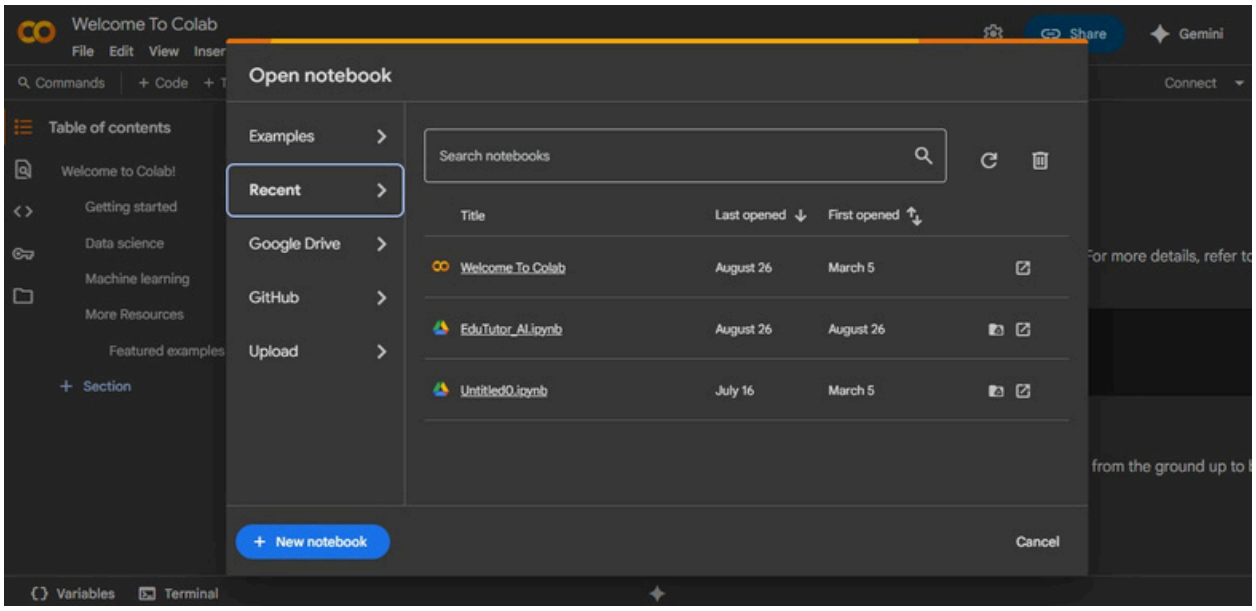
- Search for “Google collab” in any browser.



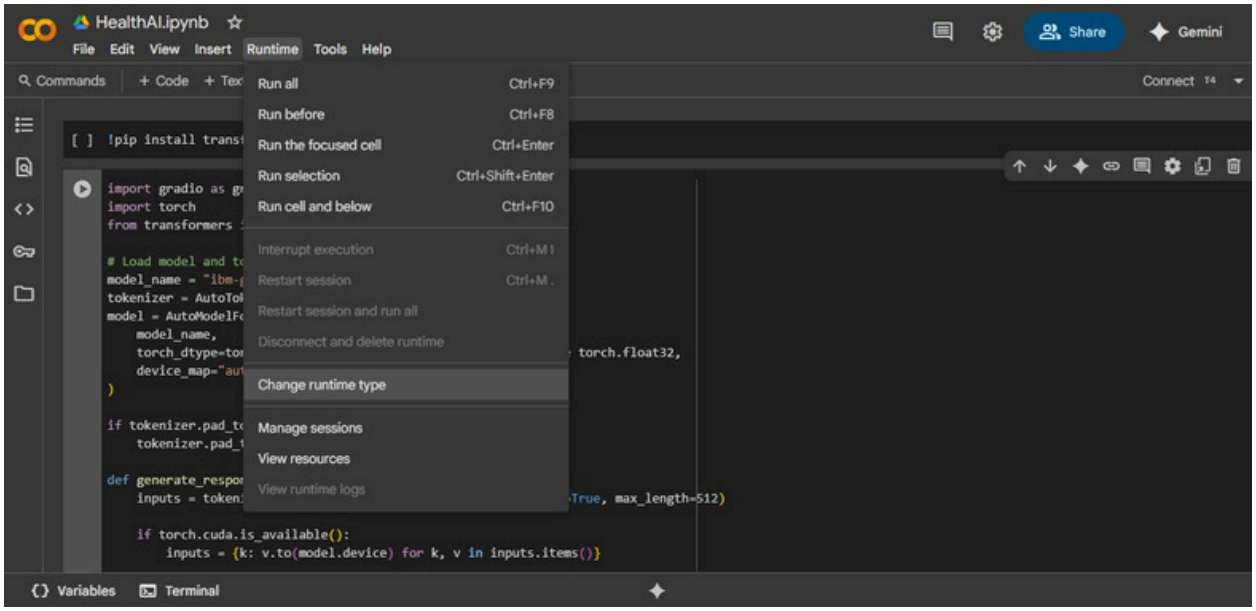
- Click on the first link (Google Colab), then click on “Files” and then “Open Notebook”.



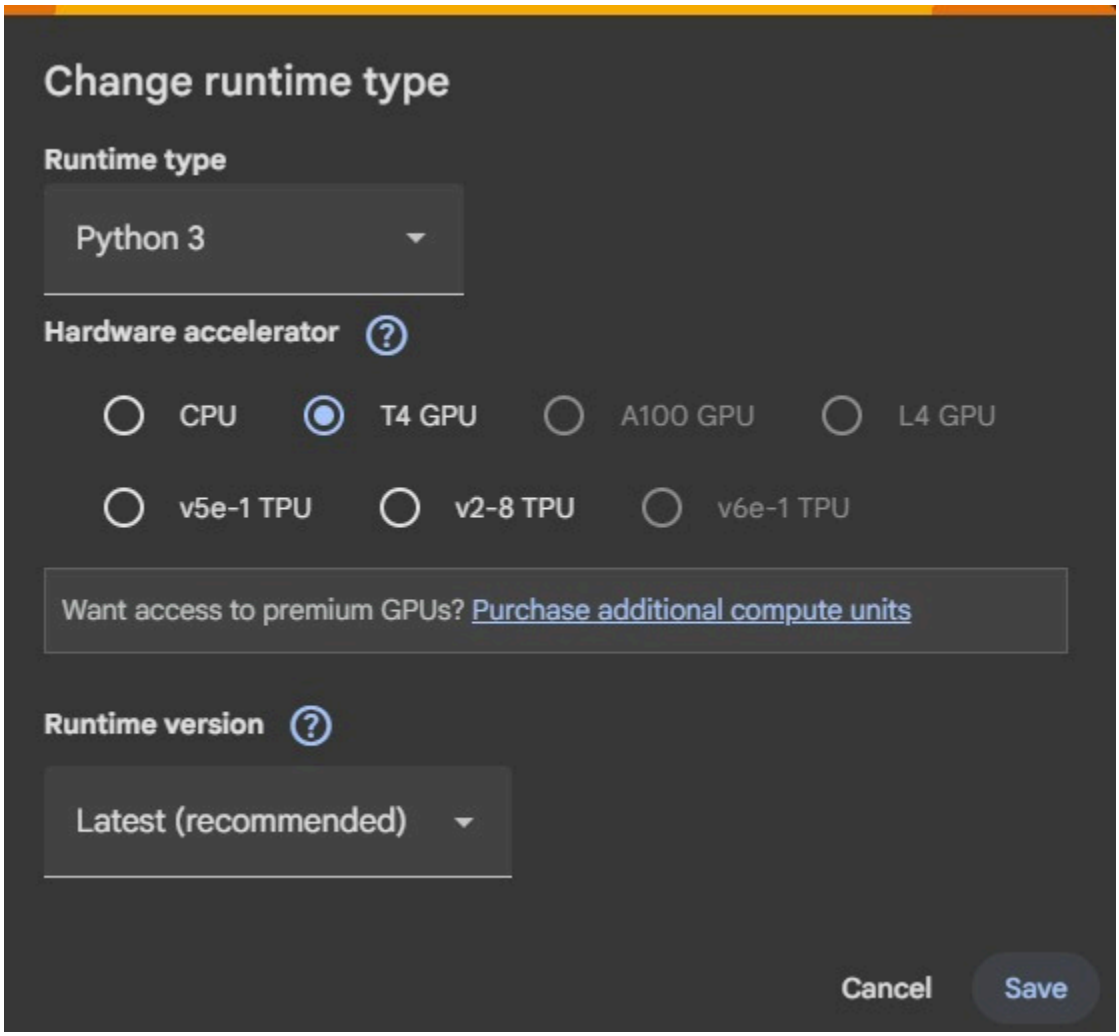
- Click on “New Notebook”



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



- Choose “T4 GPU” and click on “Save”



- Then run this command in the first cell “`!pipinstalltransformerstorch` gradio -q”. To install the required libraries to run our application.

- Then run the rest of the code in the next 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.**")
```

```python
# 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",
```

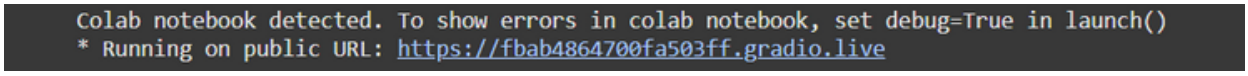
```python
        with gr.TabItem("Treatment Plans"):
            with gr.Row():
                with gr.Column():
                    condition_input = gr.Textbox(
                        label="Medical Condition",
                        placeholder="e.g., diabetes, hypertension, migraine...",
                        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]

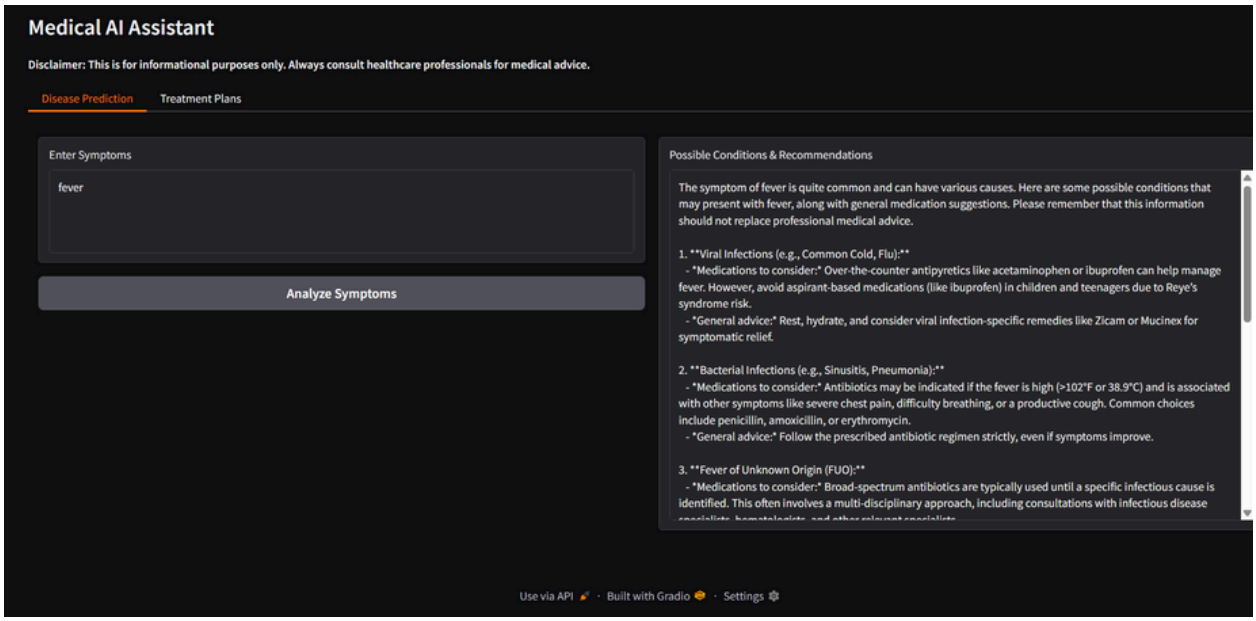
OUTPUT:

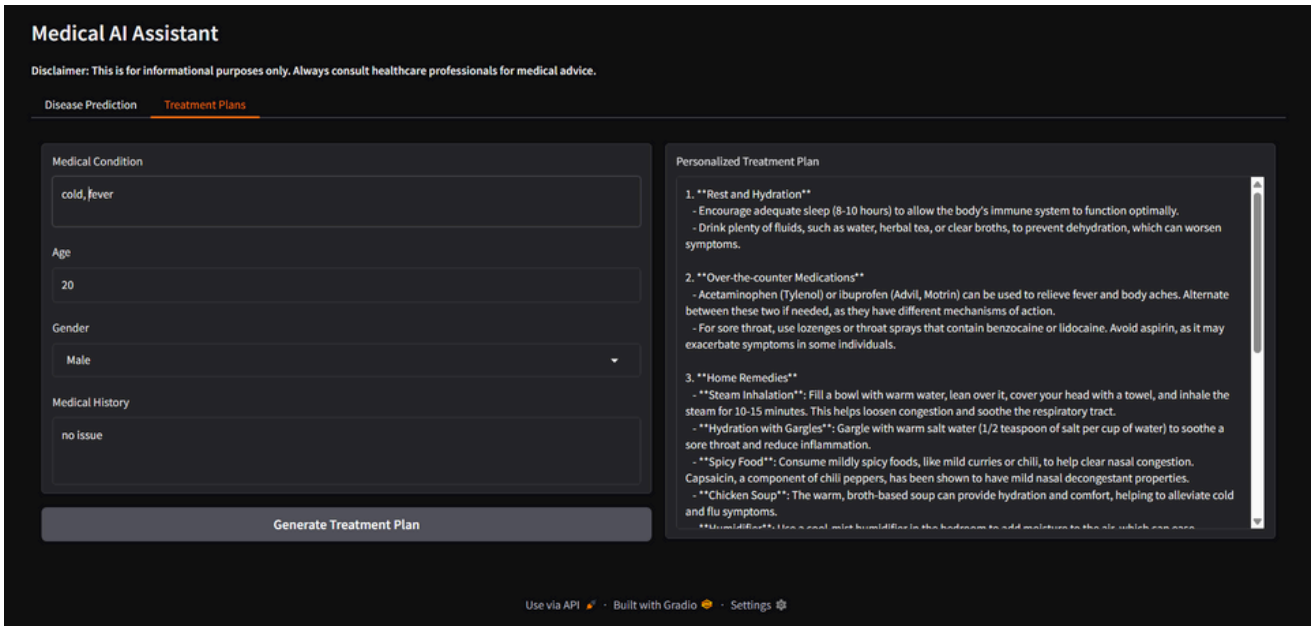
- Now you can see our model is being Downloaded and the application is running.

- Click on the URl to open the Gradio Application click on the link.



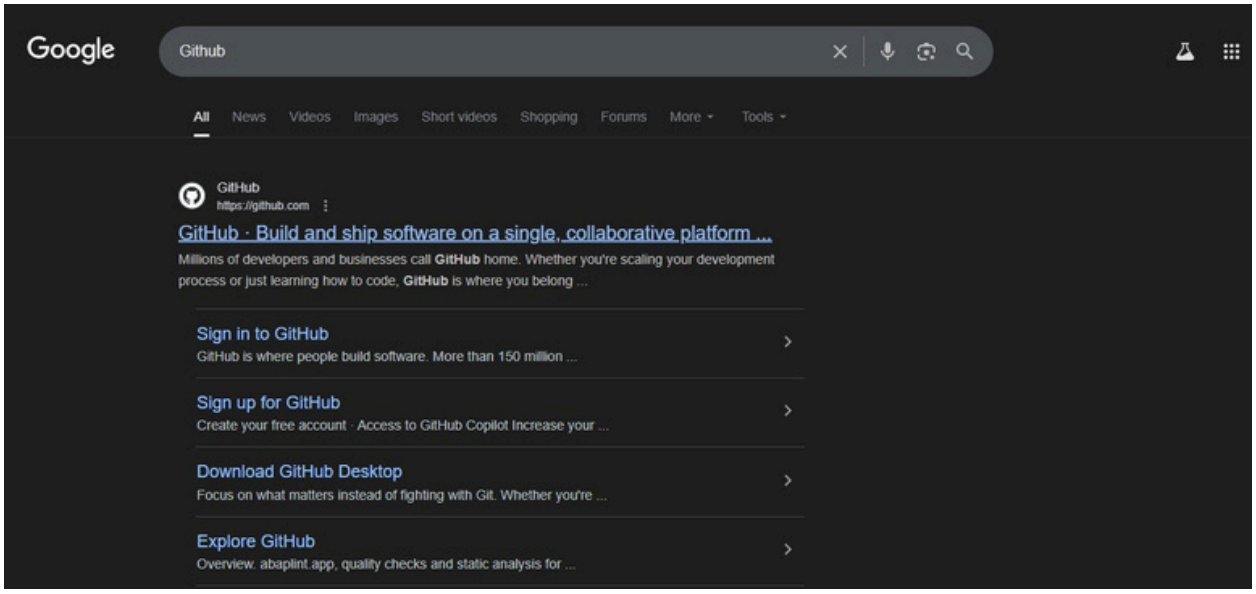
- You can View the Application is running in the other tab.



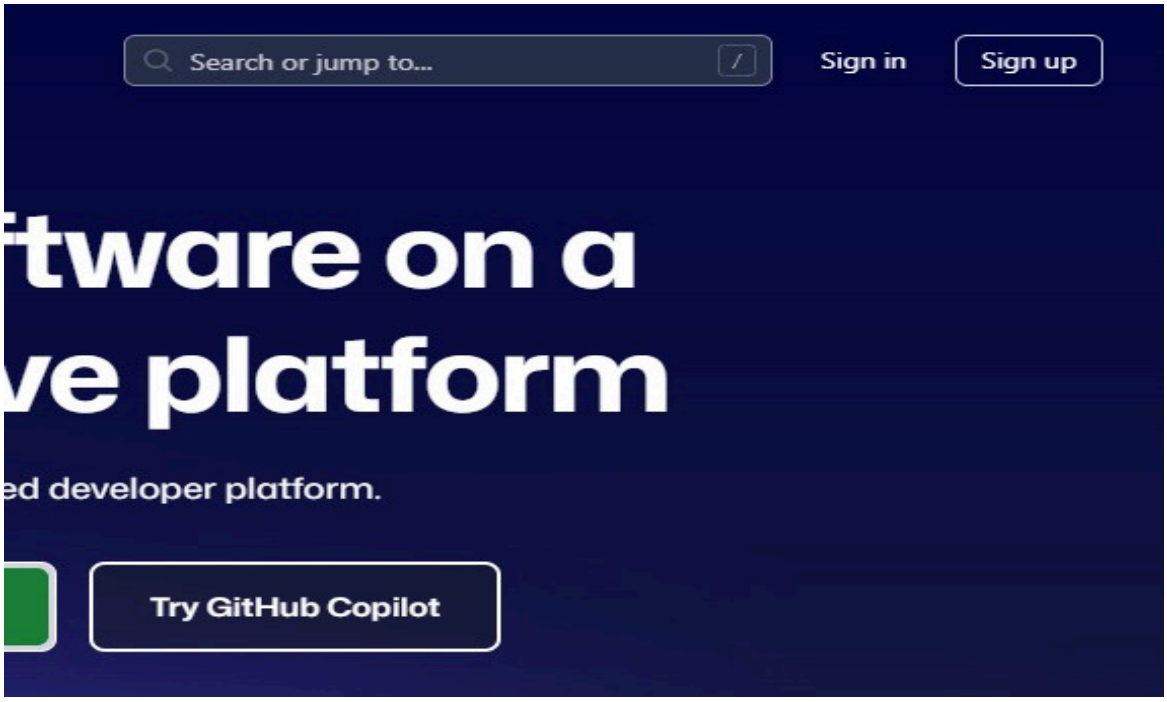

Medical AI Assistant
Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.
Disease Prediction
Treatment Plans
Medical Condition
cold, fever
Age
20
Gender
Male
Medical History
no issue
Generate Treatment Plan
Personalized Treatment Plan
1. **Rest and Hydration**
- Encourage adequate sleep (8-10 hours) to allow the body's immune system to function optimally.
- Drink plenty of fluids, such as water, herbal tea, or clear broths, to prevent dehydration, which can worsen symptoms.
2. **Over-the-counter Medications**
- Acetaminophen (Tylenol) or ibuprofen (Advil, Motrin) can be used to relieve fever and body aches. Alternate between these two if needed, as they have different mechanisms of action.
- For sore throat, use lozenges or throat sprays that contain benzocaine or lidocaine. Avoid aspirin, as it may exacerbate symptoms in some individuals.
3. **Home Remedies**
- **Steam Inhalation**: Fill a bowl with warm water, lean over it, cover your head with a towel, and inhale the steam for 10-15 minutes. This helps loosen congestion and soothe the respiratory tract.
- **Hydration with Gargles**: Gargle with warm salt water (1/2 teaspoon of salt per cup of water) to soothe a sore throat and reduce inflammation.
- **Spicy Food**: Consume mildly spicy foods, like mild curries or chili, to help clear nasal congestion. Capsaicin, a component of chili peppers, has been shown to have mild nasal decongestant properties.
- **Chicken Soup**: The warm, broth-based soup can provide hydration and comfort, helping to alleviate cold and flu symptoms.
Use via API · Built with Gradio · Settings

Activity-4: Upload Your Project in GitHub.

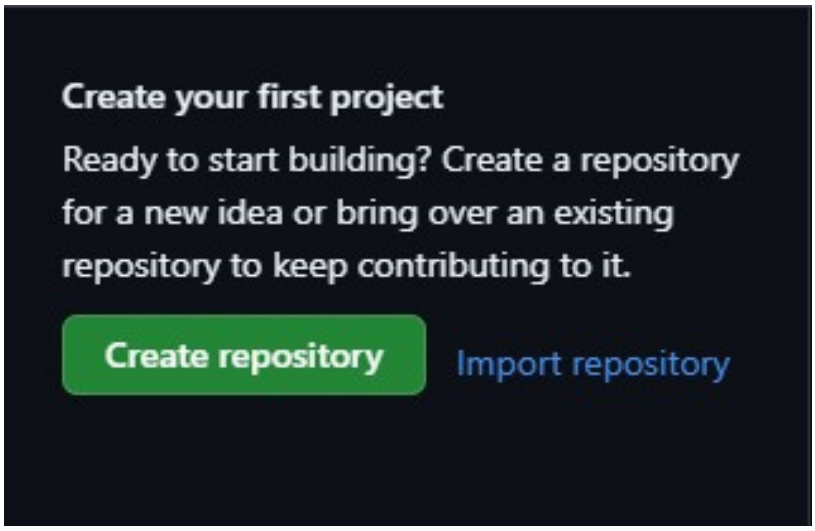
- Search for “GitHub” in any browser, then click on the first link (GitHub).



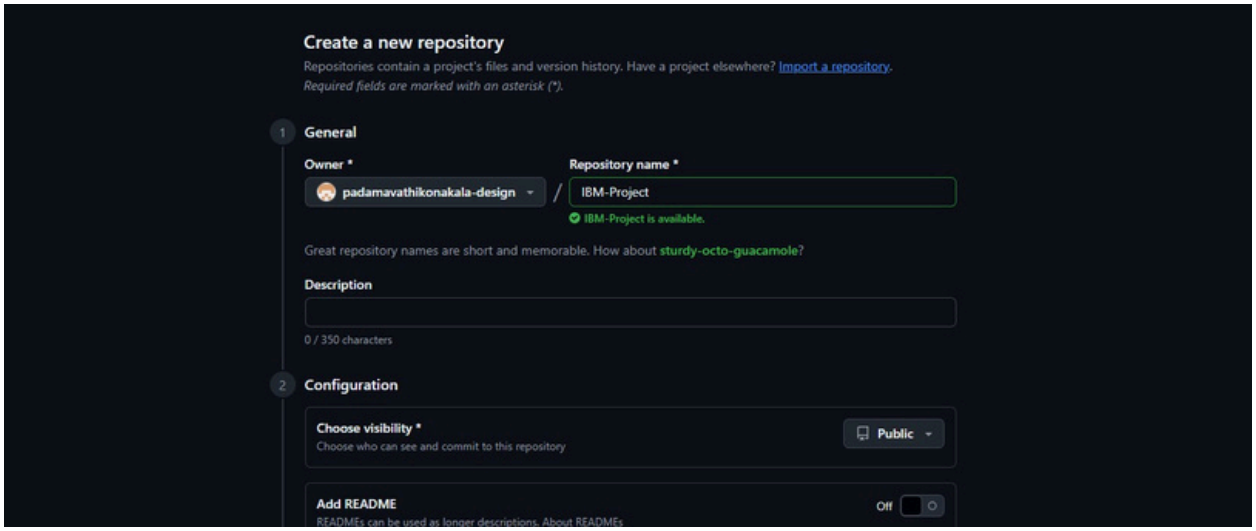
- Then click on “Signup” and create your own account in GitHub. If you already have an account click on “Sign in”



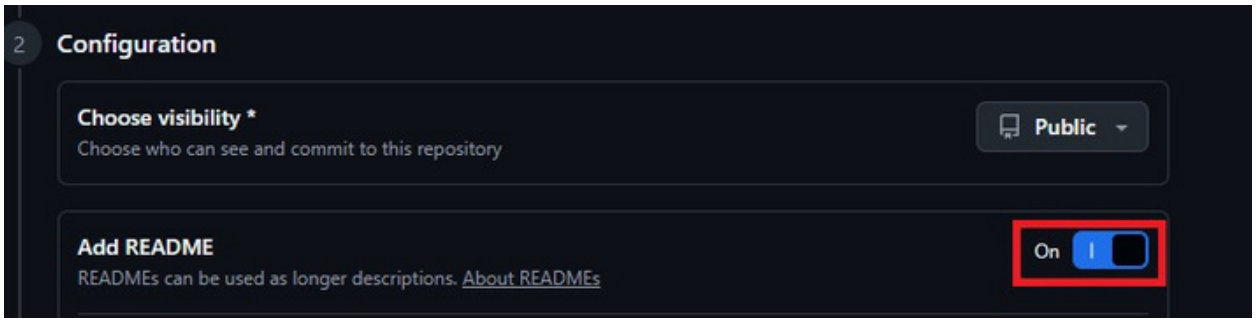
- Click on “Create repository”.



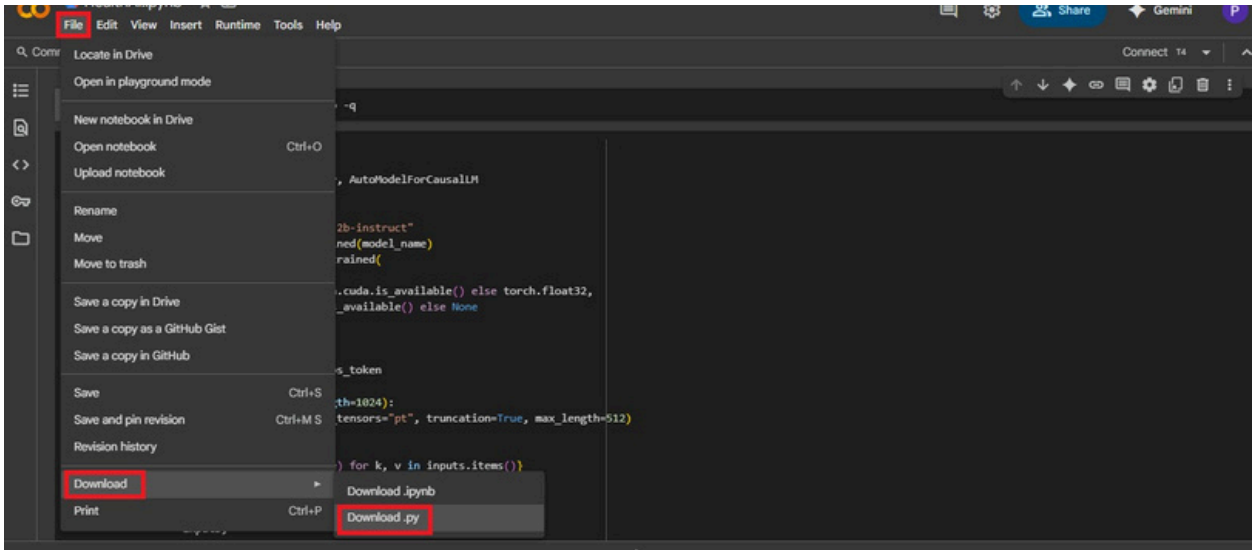
- In “General” Name your repo. (Here I have given “IBM-Project” as my repo name and it is available)



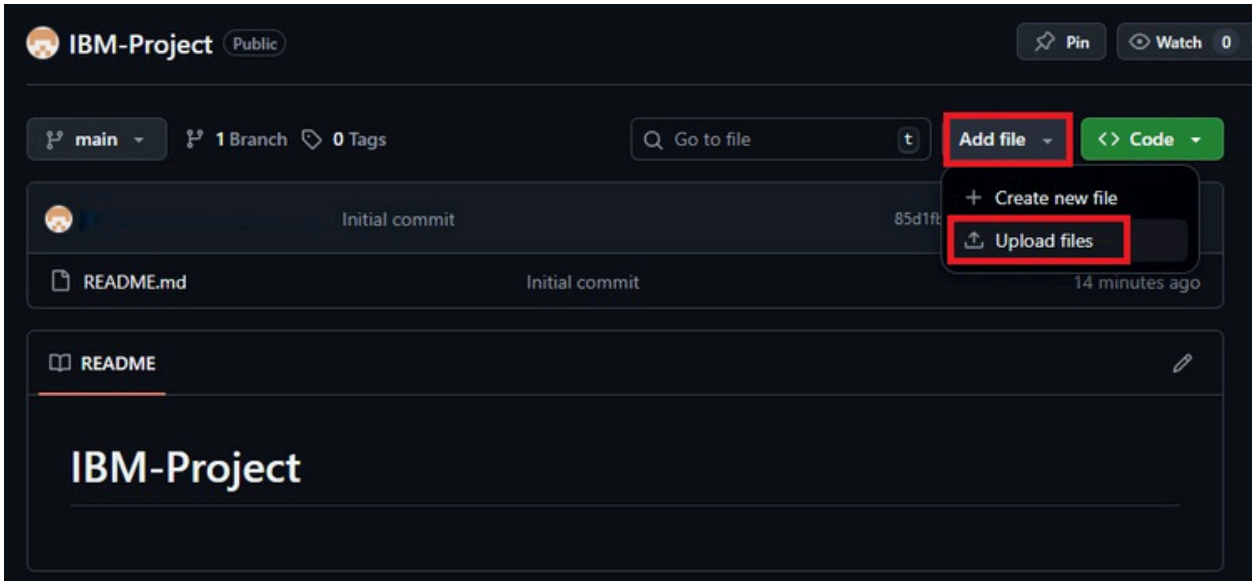
- In “Configurations” Turn On “Add readme” file Option.



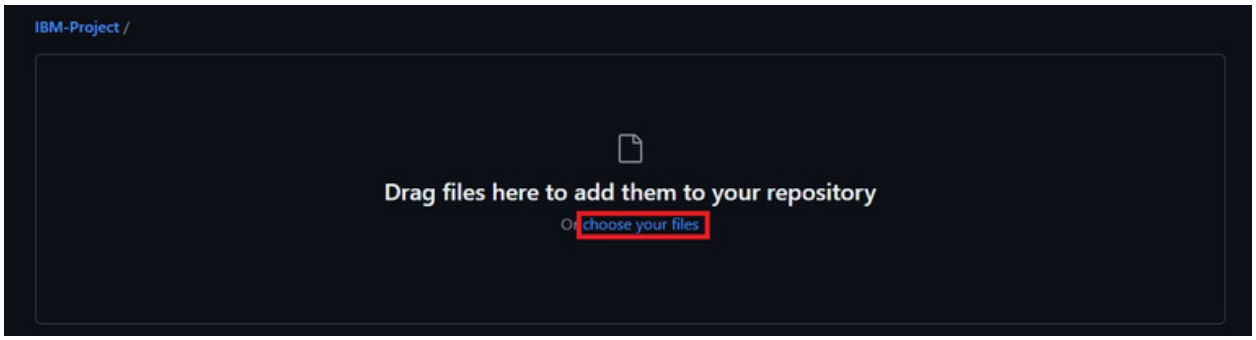
- Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.



- Then your repository is created, then Click on “Add file” Option. Then Click “Upload files” to upload your files.

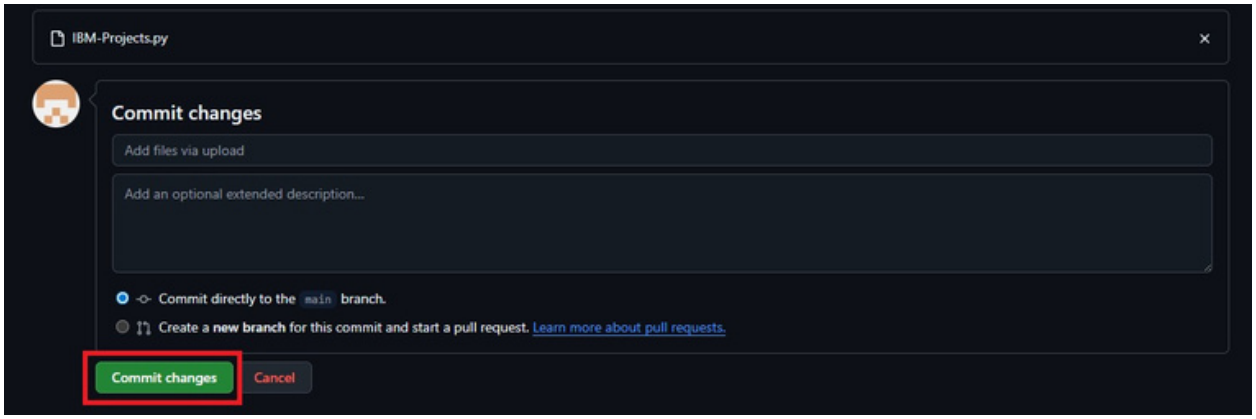


- Click on “choose your files”.



- Choose your project file and click on “Open”.

- After your file has Uploaded Click on “Commit changes”.



CONCLUSION :

The Health AI Assistant project successfully showcases the application of Generative AI in healthcare, demonstrating how advanced models like IBM Granite can be integrated into user-friendly platforms such as Gradio and Google Colab. Through its interactive features—patient chat, disease prediction, and treatment planning—the project highlights the potential of AI to assist in delivering accessible healthcare guidance.

From a learning perspective, this project has provided valuable experience in AI model integration, cloud-based deployment, and collaborative coding practices. It not only deepens technical knowledge but also emphasizes the ethical responsibility of designing healthcare applications with safety and accessibility in mind. Ultimately, the Health AI Assistant stands as a practical example of how students can leverage cutting-edge technologies to create impactful solutions that address societal needs while preparing themselves for future careers in AI and healthcare innovation.