

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 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 Workflow:

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

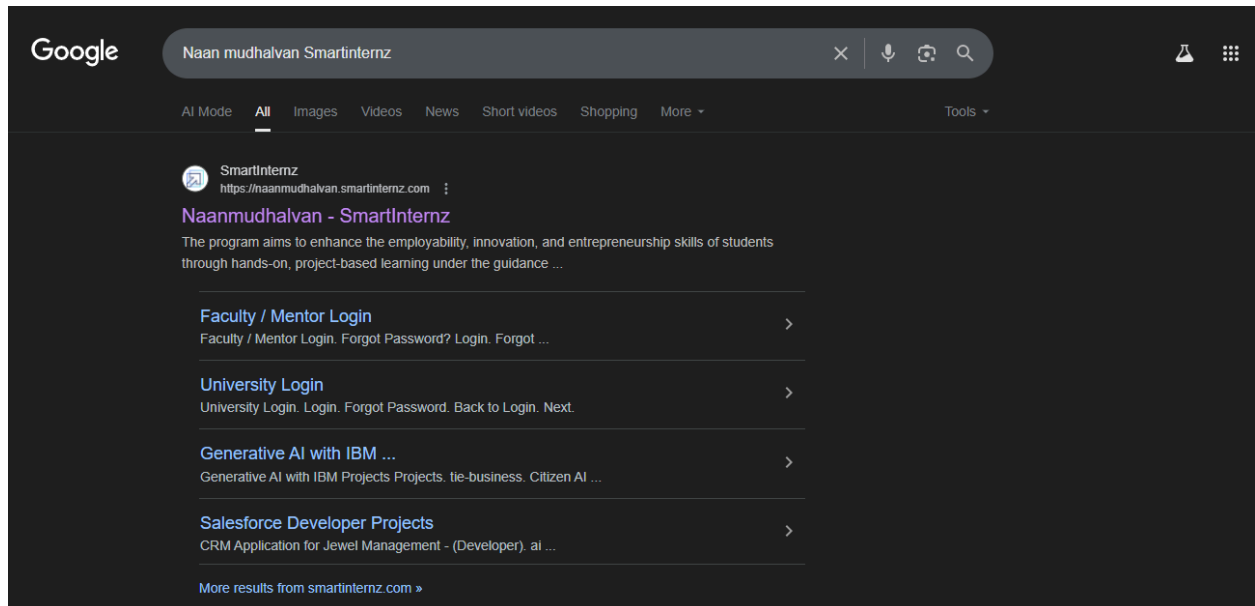
Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application In Google Colab.

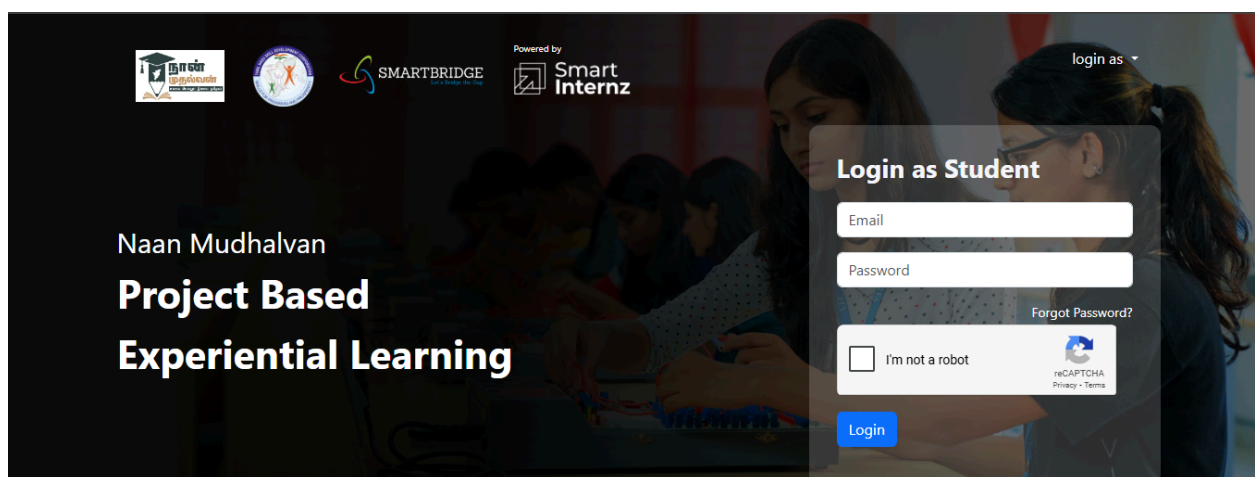
Activity-4: Upload your Project in Github.

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

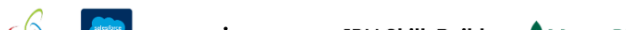
- Search for “Naan Mudhalavan Smart Interz” Portal in any Browser.



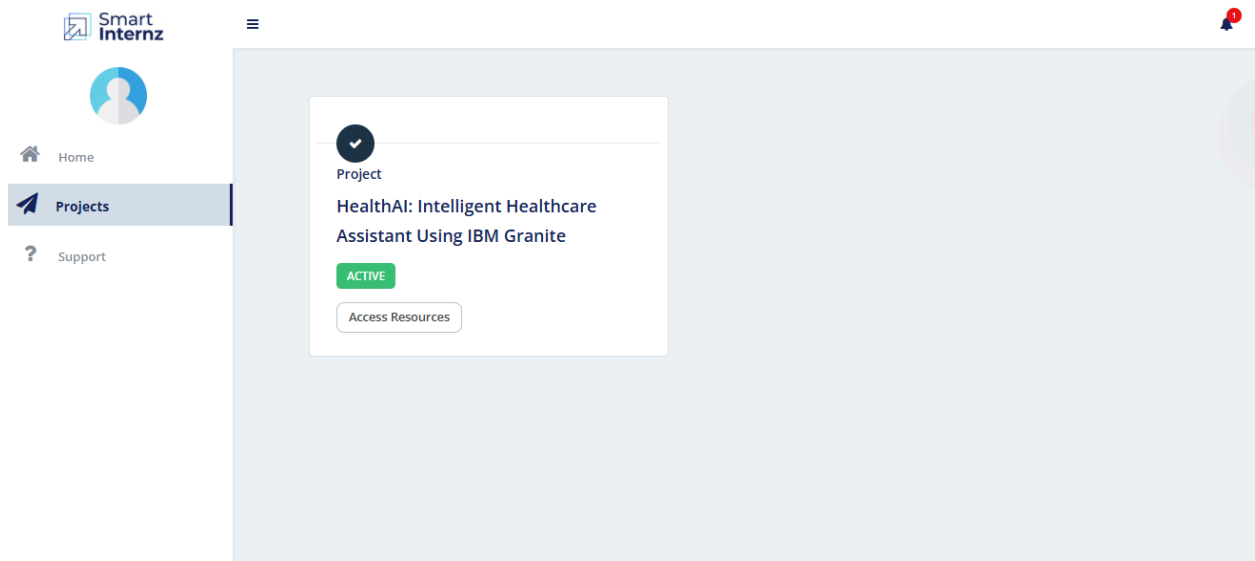
- Then Click on the first link. ([Naanmudhalvan Smartinternz](https://naanmudhalvan-smartinternz.com)) Then login with your details.



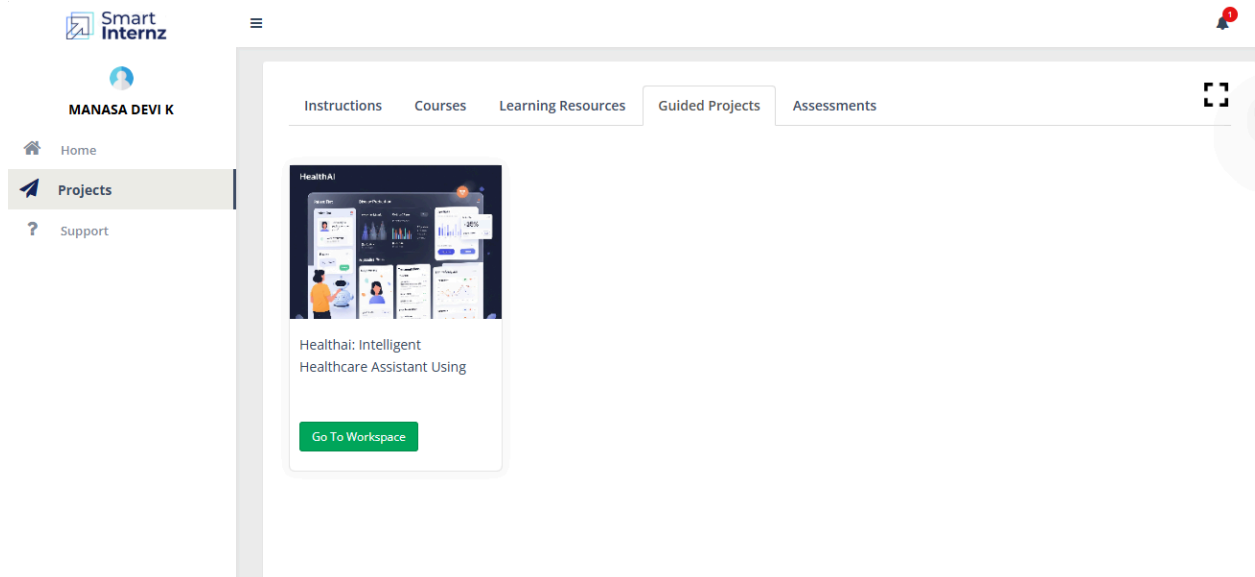
In Partnership with



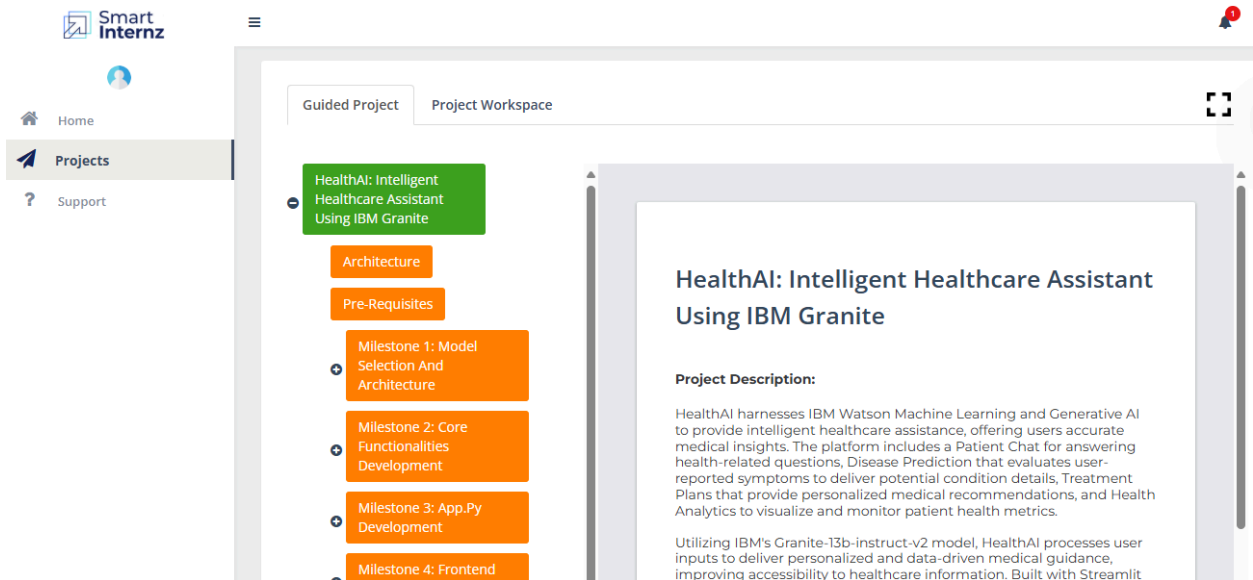
- Then you will be redirected to your account then click on “Projects” Section. There you can see which project you have enrolled in here it is “Health AI”.



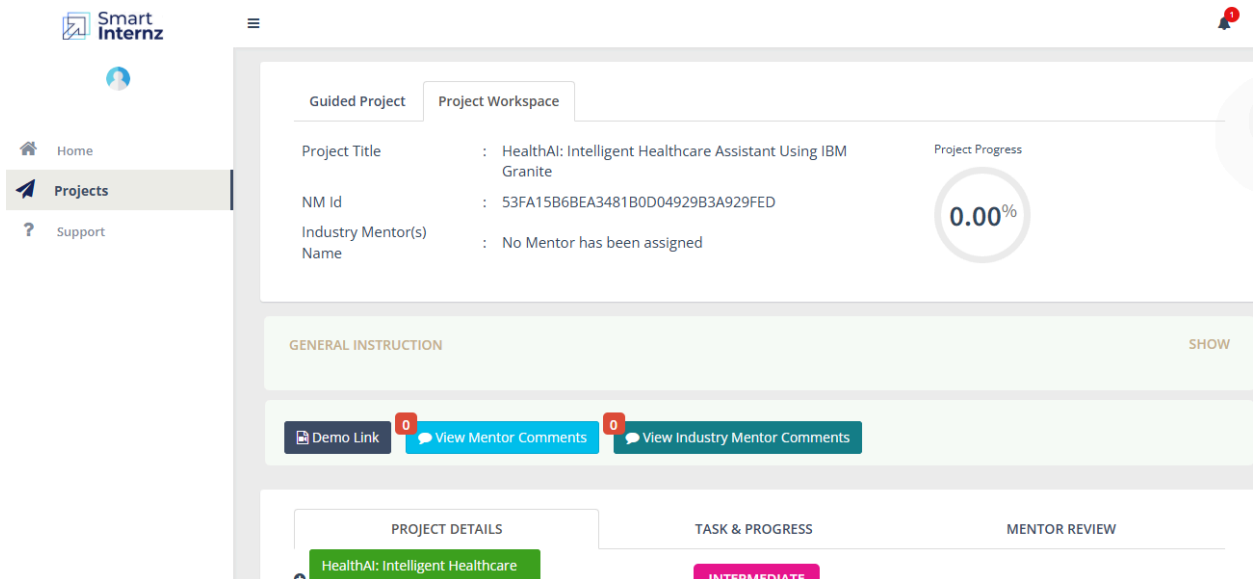
- Then click on “Access Resources” and go to the “Guided Project” Section.



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



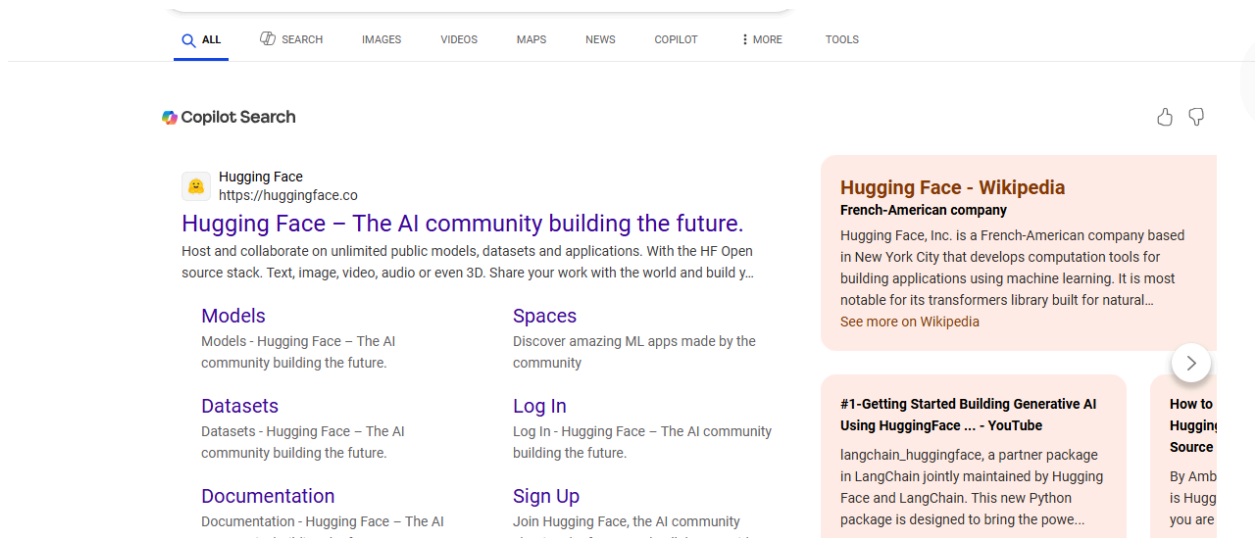
- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.



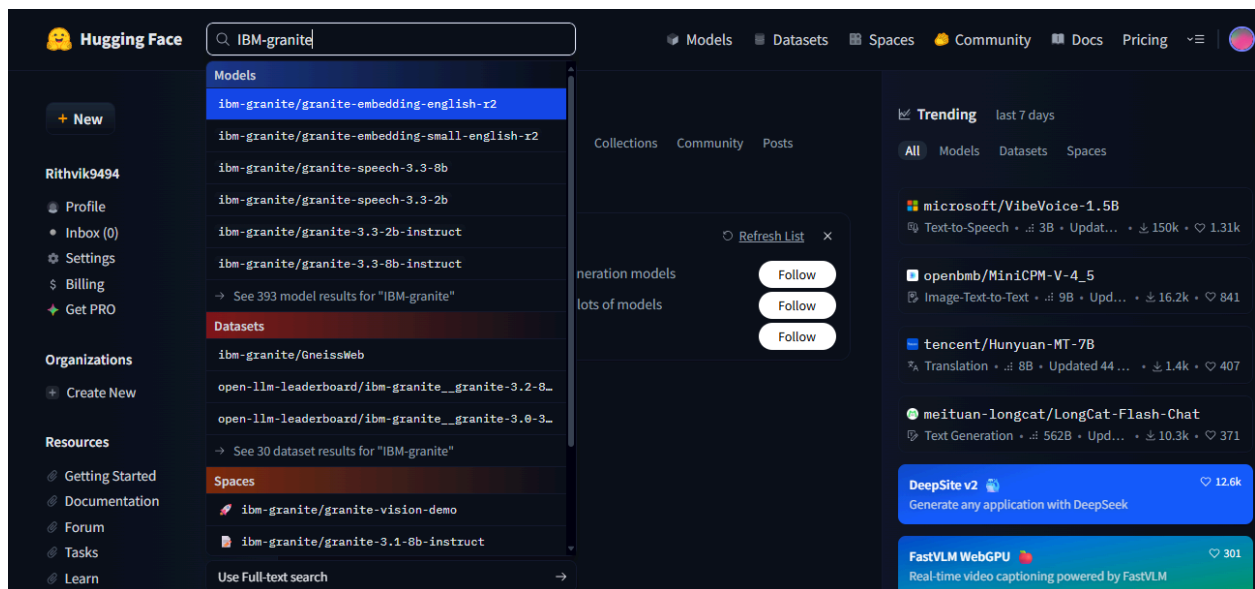
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

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

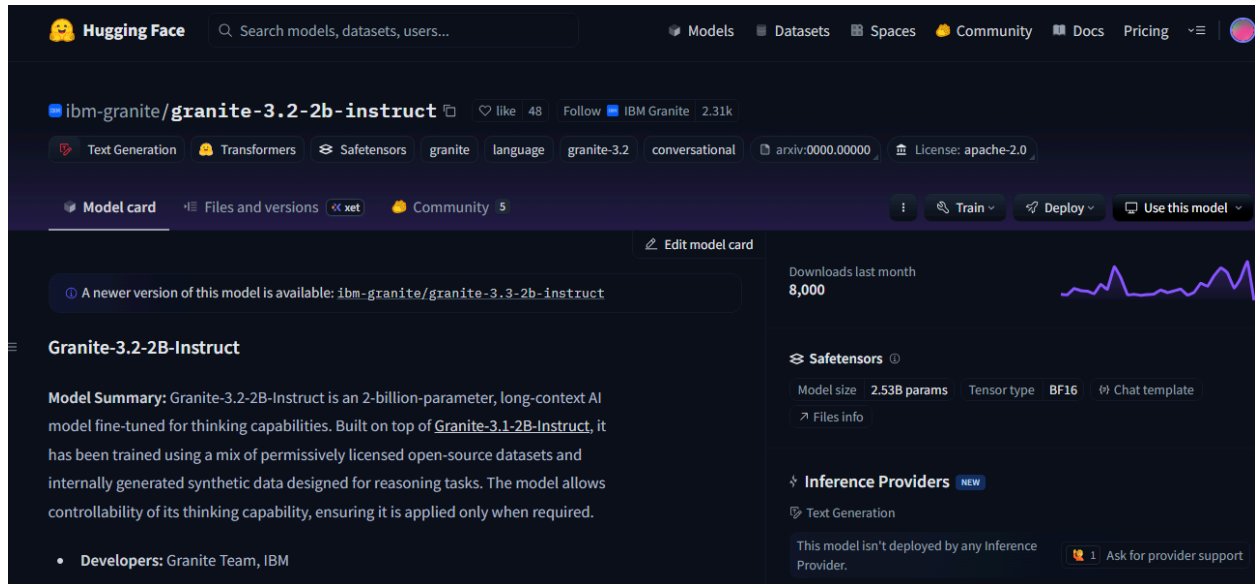
- Search for “Hugging face” in any browser.



- Then click on the first link ([Hugging Face](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.



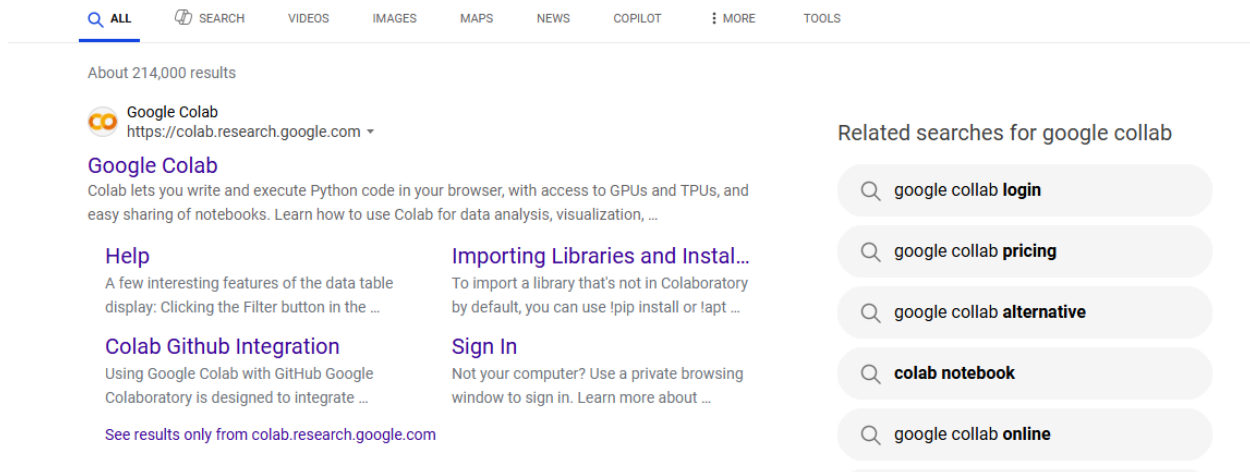
- Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.



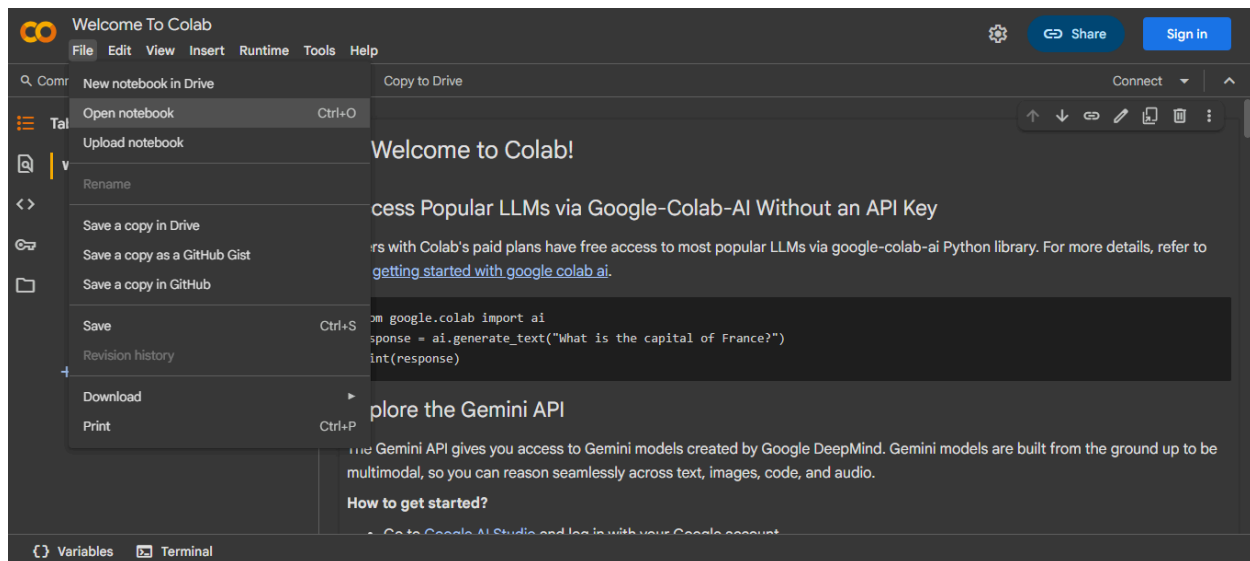
- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

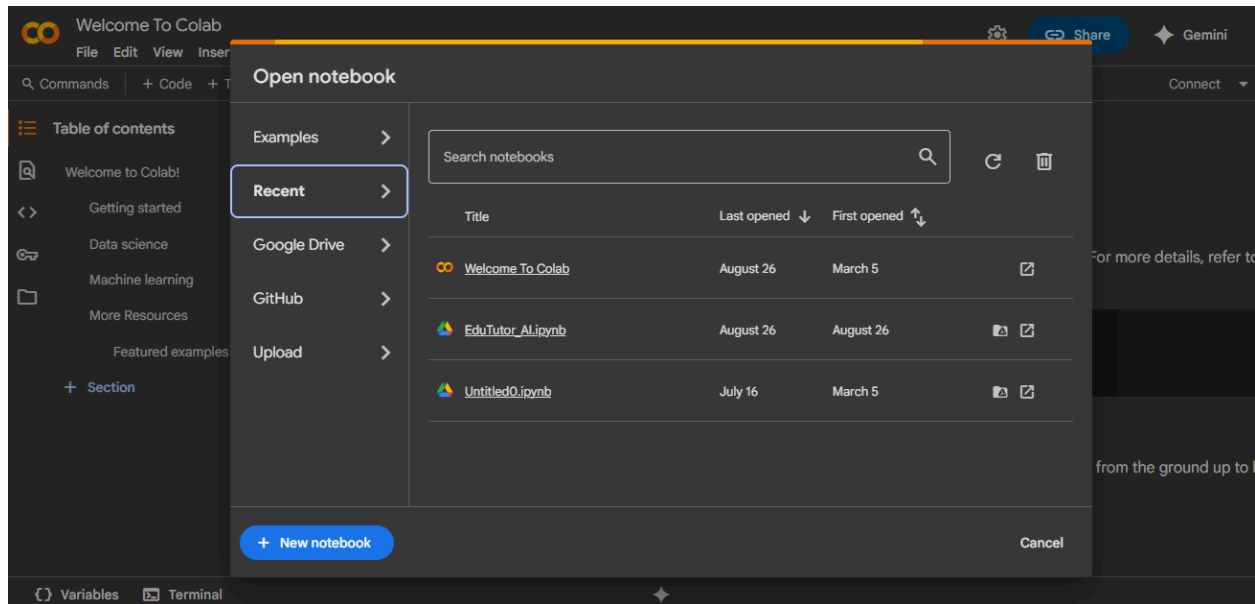
- Search for “Google collab” in any browser.



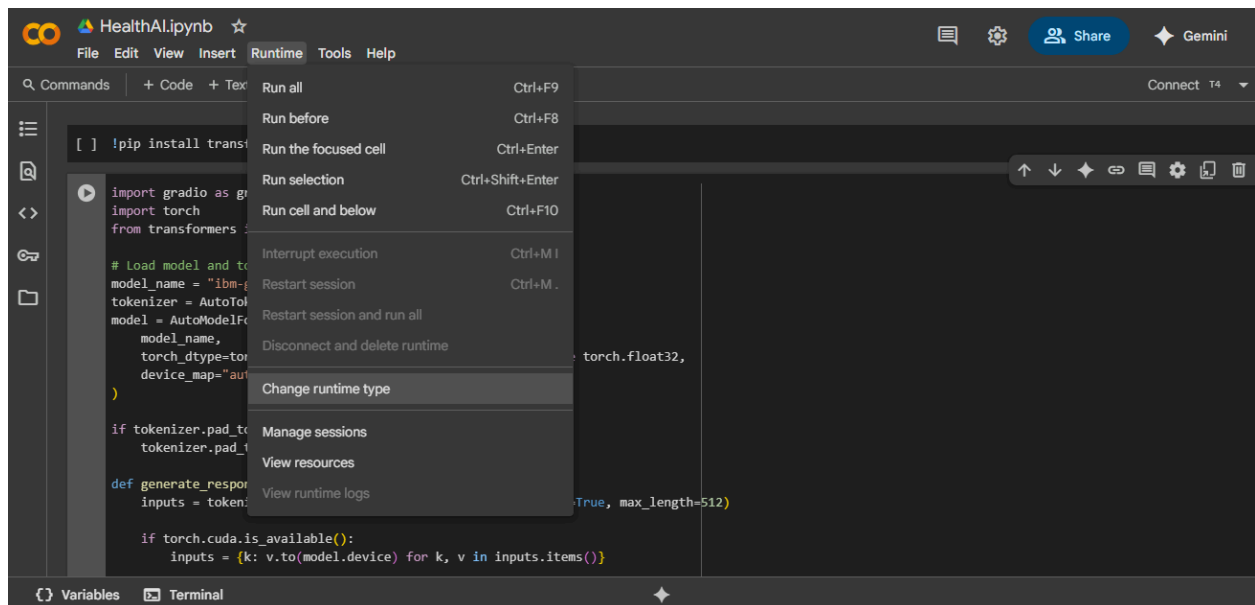
- Click on the first link ([Google Colab](https://colab.research.google.com)), then click on “Files” and then “Open Notebook”.



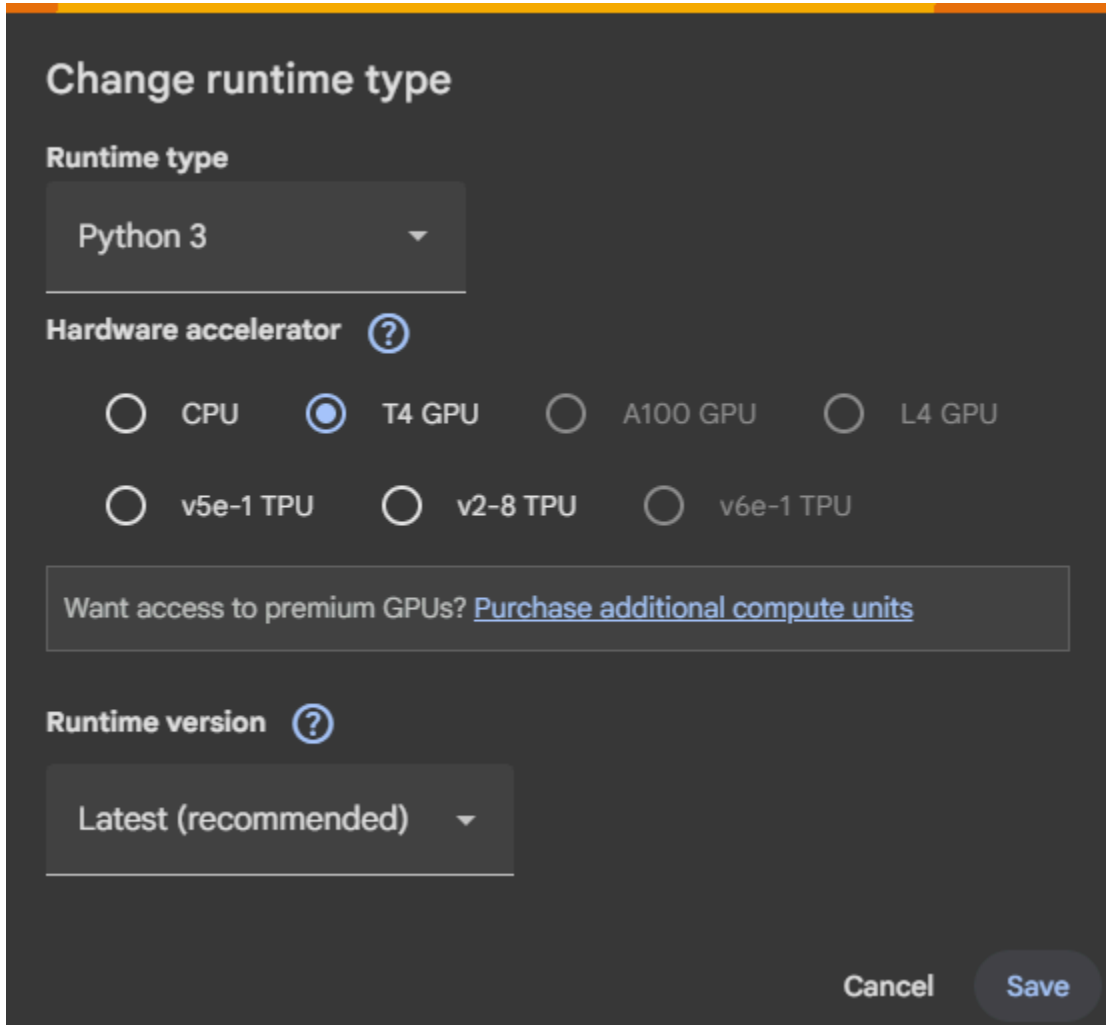
- Click on “New Notebook”



- 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”



The image shows a 'Change runtime type' dialog box with a dark background. At the top, the title 'Change runtime type' is in white. Below it, the 'Runtime type' is set to 'Python 3' in a dropdown menu. The 'Hardware accelerator' section has a question mark icon and several radio button options: 'CPU', 'T4 GPU' (which is selected), 'A100 GPU', 'L4 GPU', 'v5e-1 TPU', 'v2-8 TPU', and 'v6e-1 TPU'. Below these options is a text box that says 'Want access to premium GPUs? [Purchase additional compute units](#)'. At the bottom, the 'Runtime version' is set to 'Latest (recommended)' in a dropdown menu. In the bottom right corner, there are 'Cancel' and 'Save' buttons.

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) ▼

Cancel Save

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



The image shows a Jupyter notebook cell with a dark background. The cell contains the command '!pip install transformers torch gradio -q'. Below the command, there is a status bar that says 'Run cell (Ctrl+Enter)' and 'cell has not been executed in this session'.

```
!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 next cell.

```

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,

```

```

            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 consulting a healthcare professional."
    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.\nPatient Info: {condition}, {age}, {gender}, {medical_history}"
    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.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.

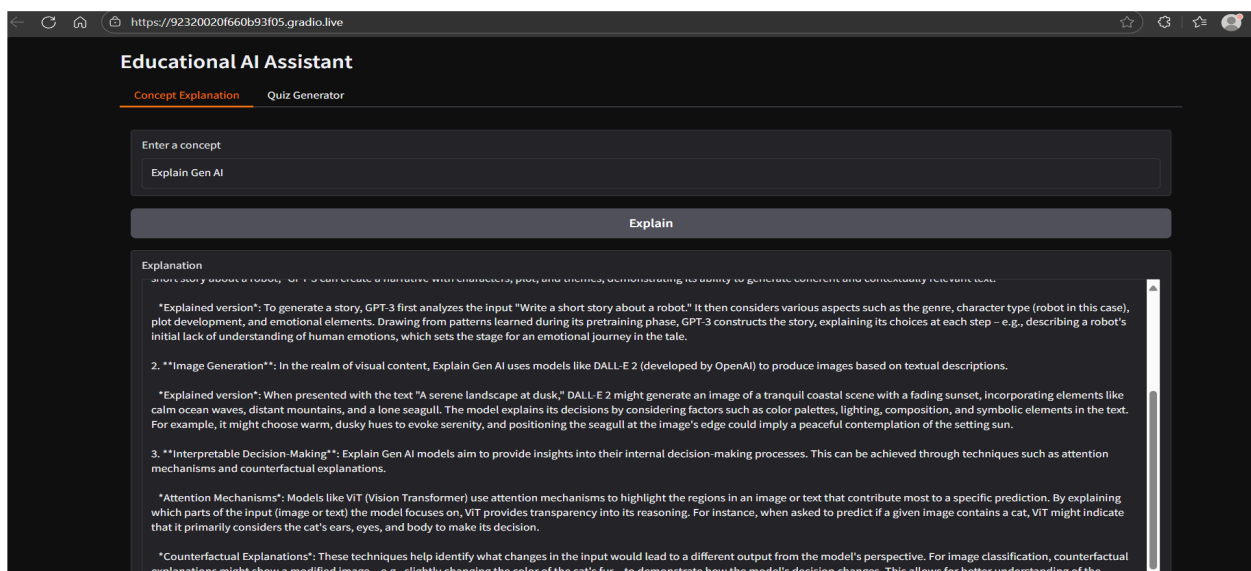
```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret 'HF_TOKEN' does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(

tokenizer_config.json: 8.88k/? [00:00<00:00, 695kB/s]
vocab.json: 777k/? [00:00<00:00, 30.9MB/s]
merges.txt: 442k/? [00:00<00:00, 23.4MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 84.3MB/s]
added_tokens.json: 100% [00:00<00:00, 87.0/87.0 [00:00<00:00, 8.14kB/s]
special_tokens_map.json: 100% [00:00<00:00, 701/701 [00:00<00:00, 50.9kB/s]
config.json: 100% [00:00<00:00, 786/786 [00:00<00:00, 48.8kB/s]
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.54MB/s]
Fetching 2 files: 100% [00:21<00:00, 2/2 [02:21<00:00, 141.84s/it]
model-00001-of-00002.safetensors: 100% [00:21<00:00, 5.00GiB/5.00G [02:21<00:00, 50.7MB/s]
model-00002-of-00002.safetensors: 100% [00:02<00:00, 67.1M/67.1M [00:02<00:00, 37.0MB/s]
Loading checkpoint shards: 100% [00:25<00:00, 2/2 [00:25<00:00, 10.58s/it]
generation_config.json: 100% [00:00<00:00, 137/137 [00:00<00:00, 10.5kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://92320020f660b93f05.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run 'gradio deploy' from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

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

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://92320020f660b93f05.gradio.live
```

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



Educational AI Assistant

Concept Explanation

Quiz Generator

Enter a topic

Gen AI

Generate Quiz

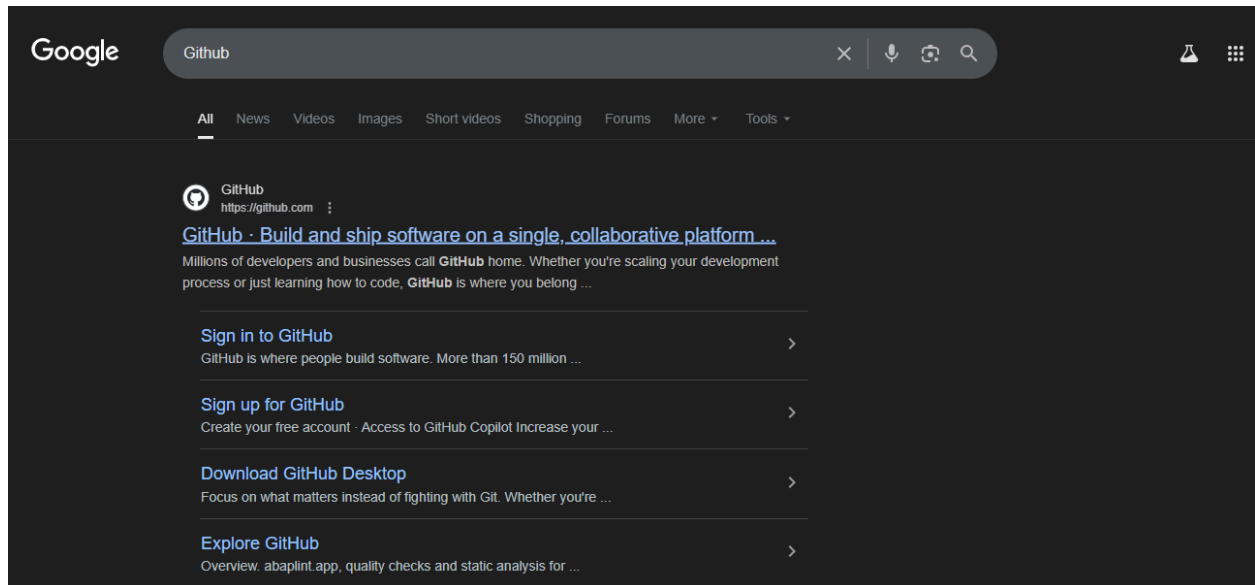
Quiz Questions

1. **Multiple Choice:** What is the primary function of Generative Artificial Intelligence (Gen AI)?
A) Data analysis
B) Content creation
C) Decision-making
D) Coding
2. **True or False:** Gen AI models can learn and improve without human intervention, a concept known as "unsupervised learning."
3. **Short Answer:** Describe a real-world application of Gen AI in generating text, such as a news article or a poem.
4. **Multiple Choice:** Which of the following is NOT a type of Generative Adversarial Network (GAN)?
A) DCGAN (Deep Convolutional GAN)
B) WGAN (Wasserstein GAN)
C) Flow-based GAN
D) Radial basis function GAN
5. **True or False:** As Gen AI continues to evolve, there are growing concerns about its potential misuse for creating deepfakes and other malicious content.

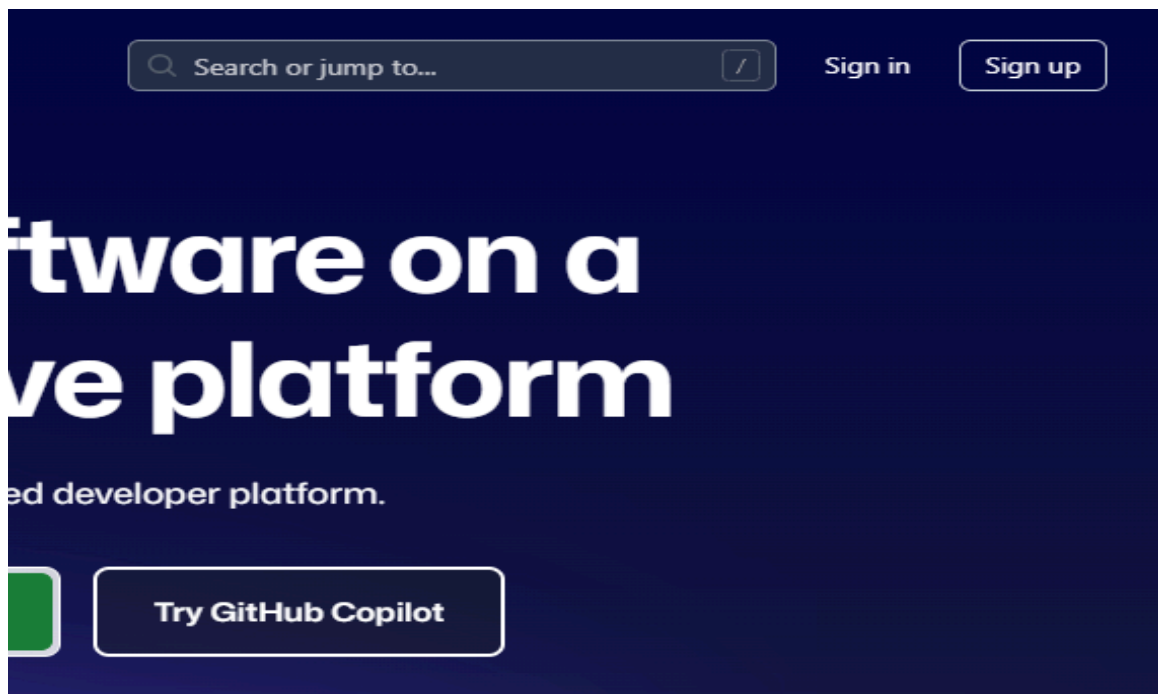
ANSWERS:

Activity-4: Upload Your Project in GitHub.

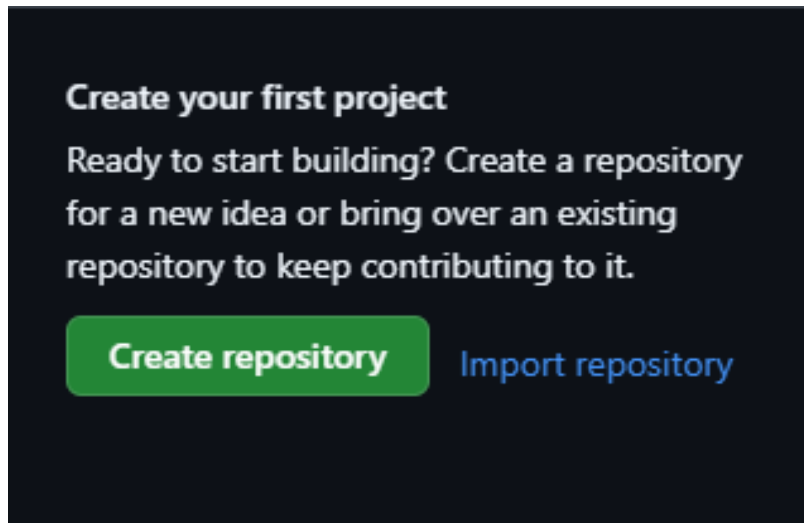
- Search for “GitHub” in any browser, then click on the first link ([GitHub](https://github.com)).



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



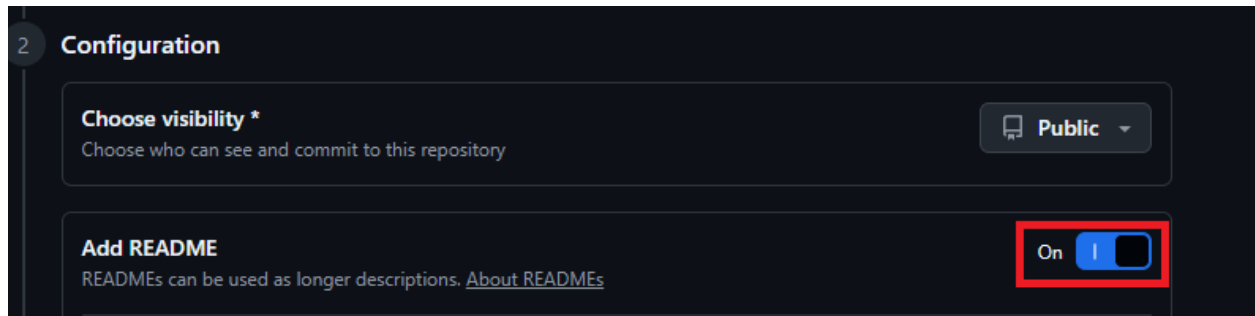
- Click on “Create repository”.



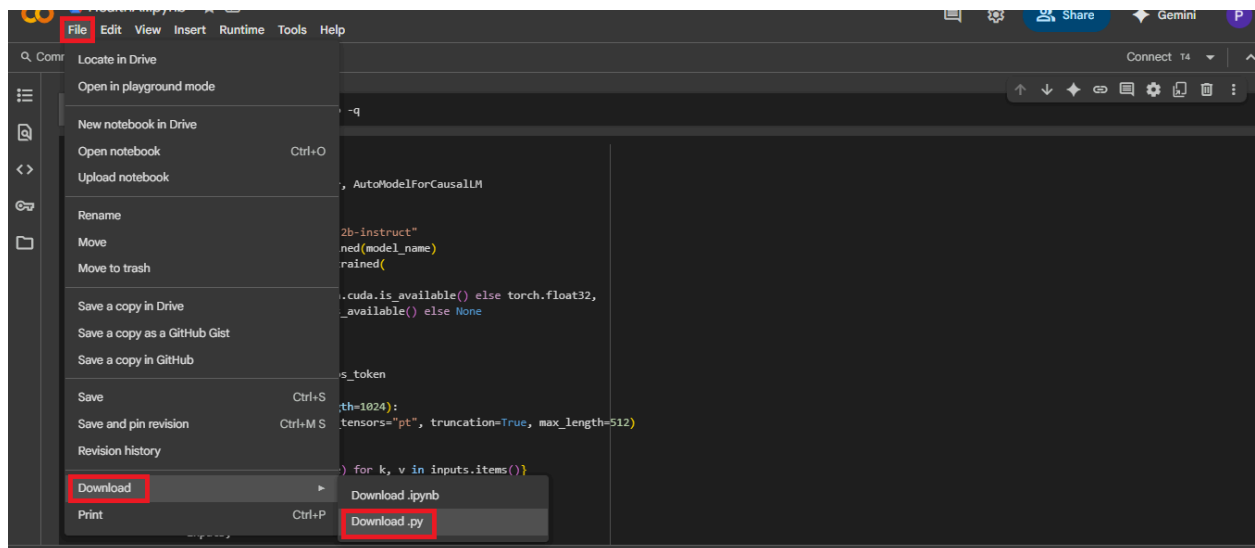
- In “General” Name your repo. (Here I have given “IBM-Project” as my repo name and it is available)

A screenshot of the GitHub 'Create a new repository' form. The form has two sections: '1 General' and '2 Configuration'. In the 'General' section, the 'Owner' is 'padamavathikonakala-design' and the 'Repository name' is 'IBM-Project', with a green checkmark indicating it is available. There is a text input for 'Description' with a character count of '0 / 350 characters'. In the 'Configuration' section, the 'Choose visibility' is set to 'Public' and the 'Add README' toggle is turned 'Off'.

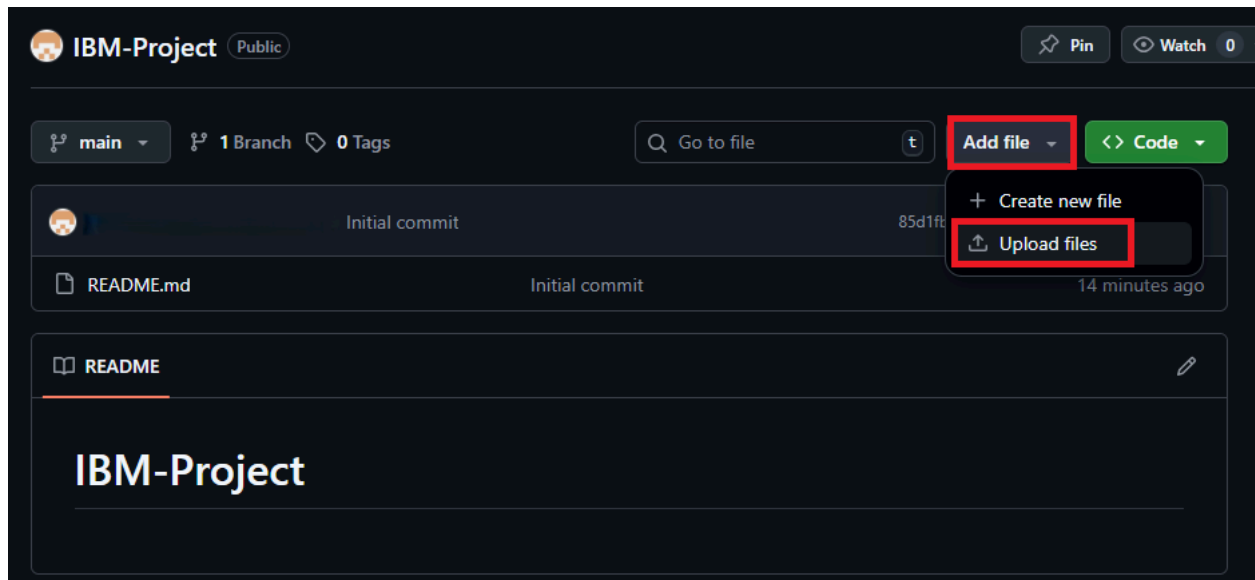
- In “Configurations” Turn On “Add readme” file Option.



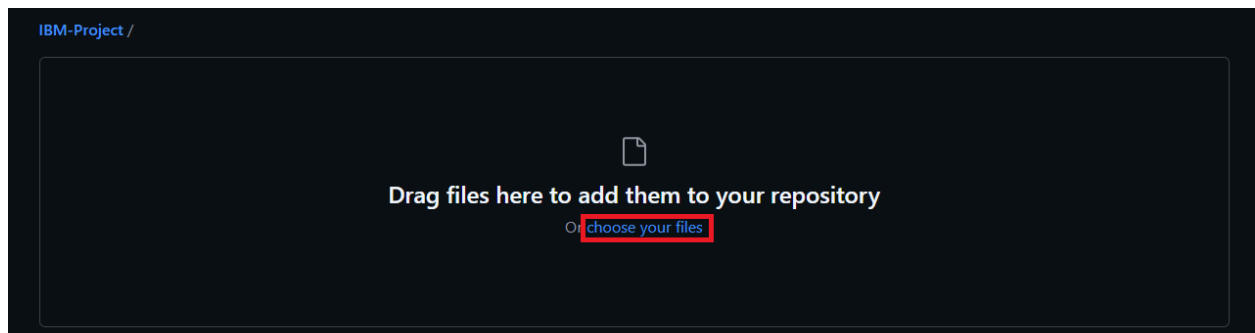
- Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.



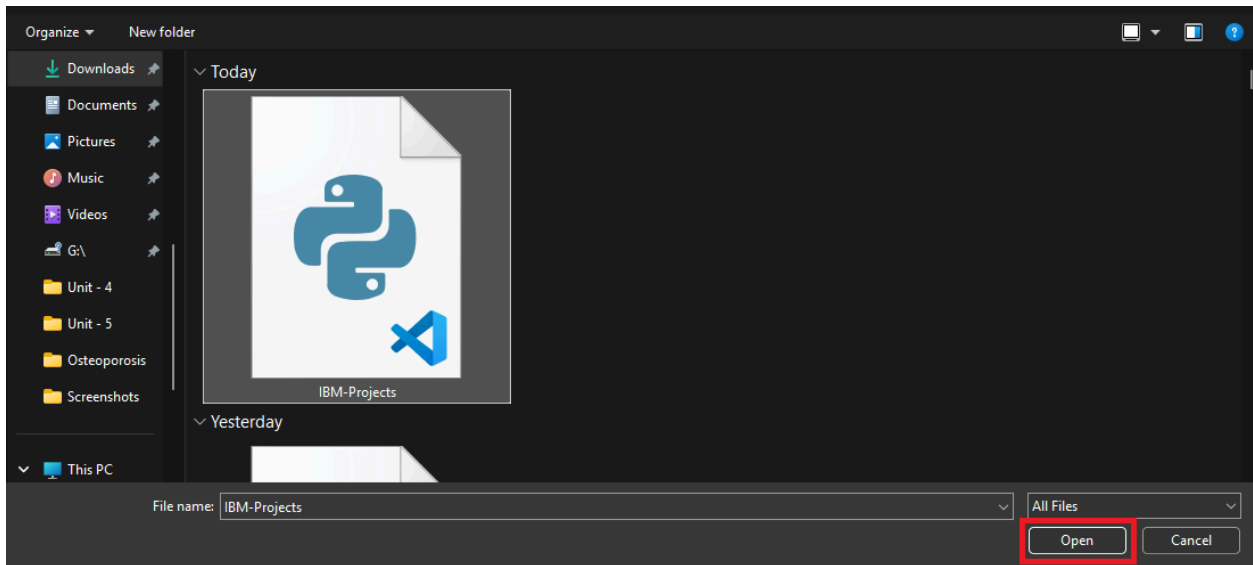
- Then your repository is created, then Click on “Add file” Option. Then Click “Upload files” to upload your files.



- Click on “choose your files”.



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



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

