Citizen AI: Intelligent Citizen Engagement Platform

Generative AI with IBM

Project Description:

Citizen AI uses the Granite model from Hugging Face to give quick, helpful answers about government services and civic issues. It tracks public sentiment and shows simple dashboards for officials to see feedback. This project will be deployed in Google Colab using Granite for easy, low-cost setup and reliable performance.

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.

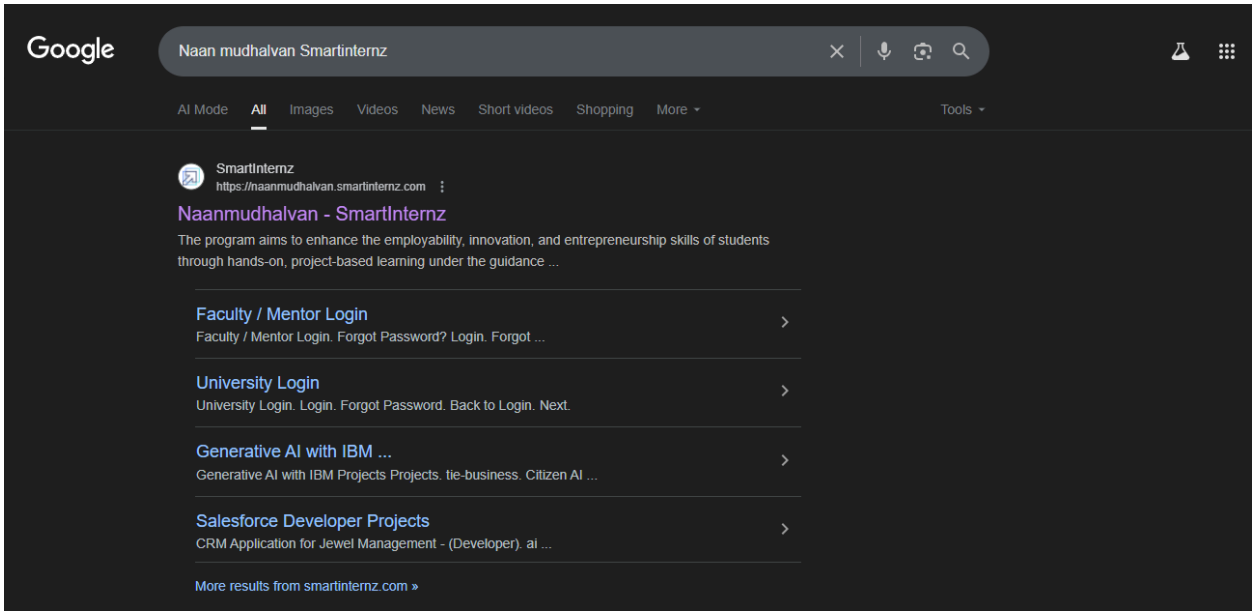
Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application In Google Colab.

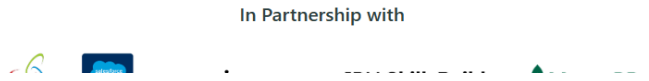
Activity-4: Upload your Project in Github.

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

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

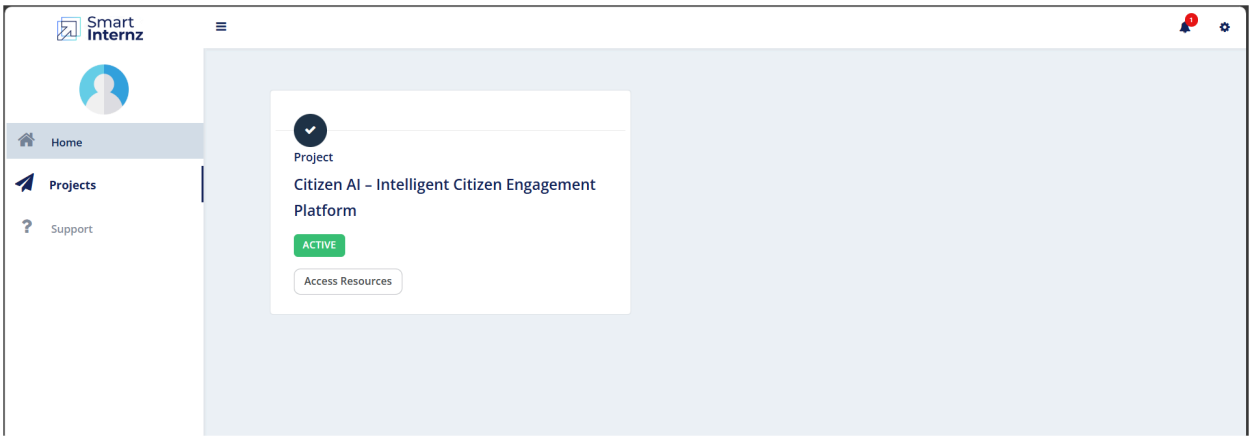


- Then Click on the first link. (Naanmudhalvan Smartinternz) Then login with your details.

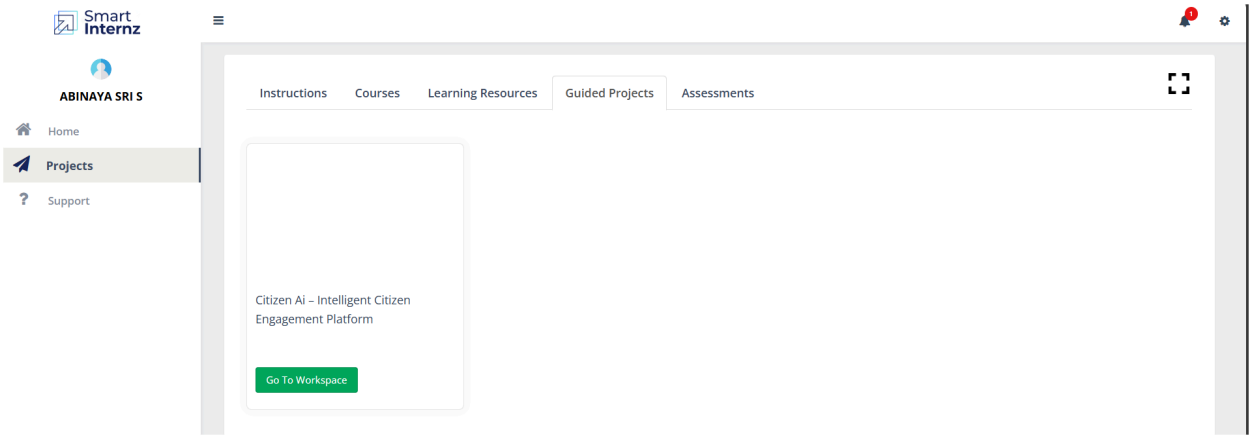


- Then you will be redirected to your account then click on “Projects”

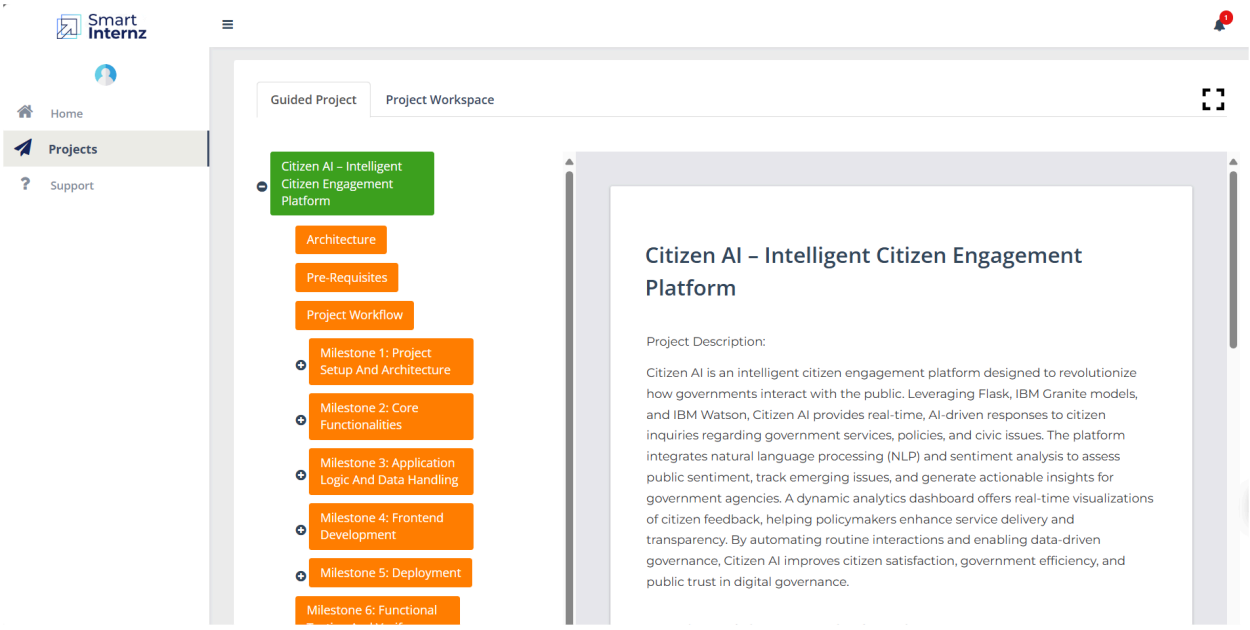
Section. There you can see which project you have enrolled in here it is “Citizen AI”.



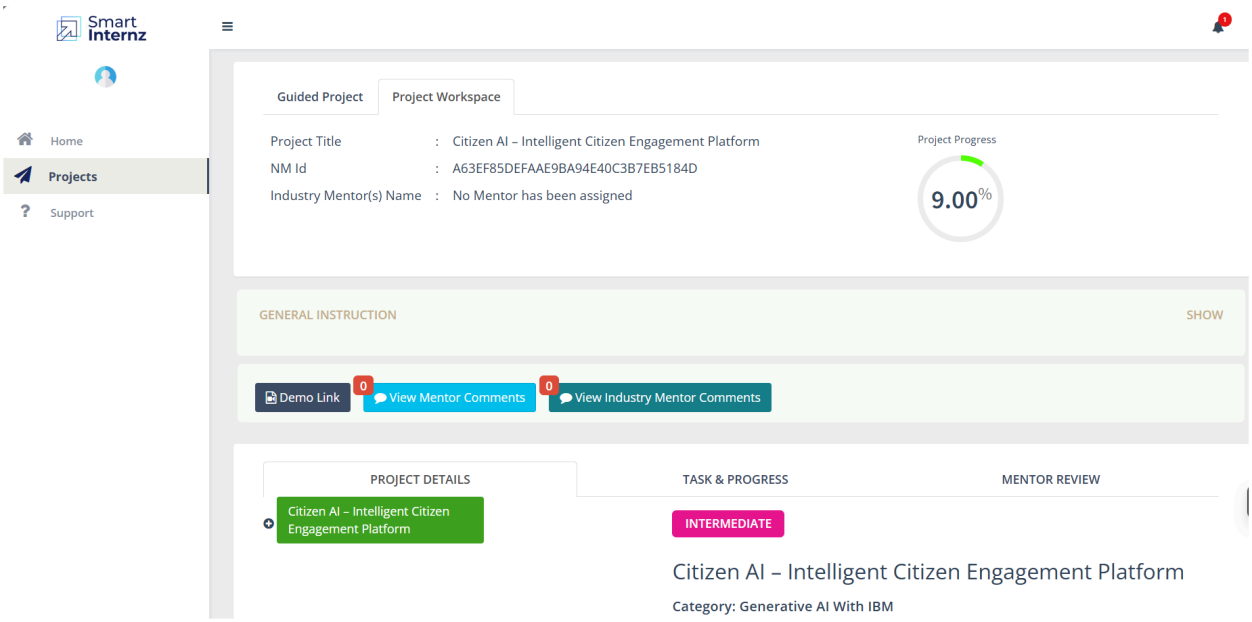
- Then click on “Access Resources” and go to the “Guided Project” Section.



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



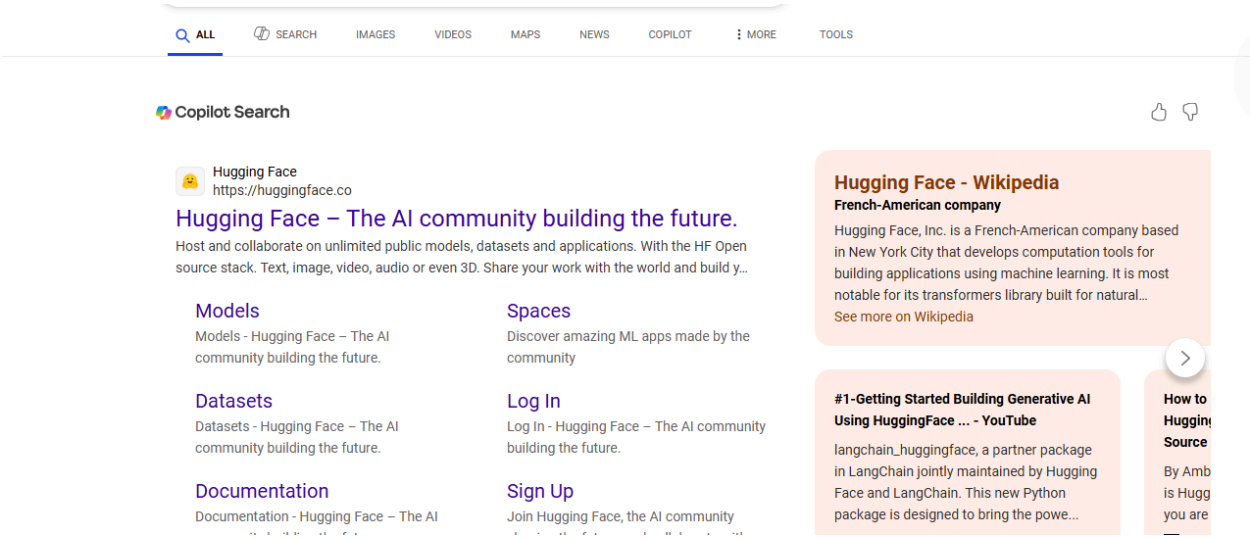
- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.



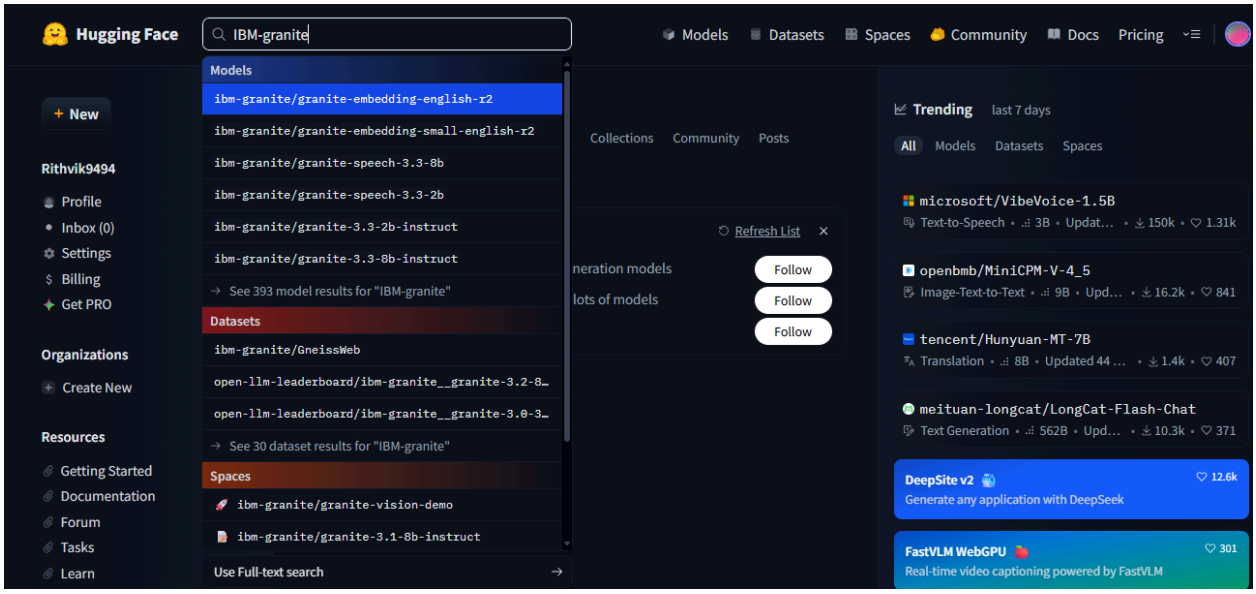
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

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

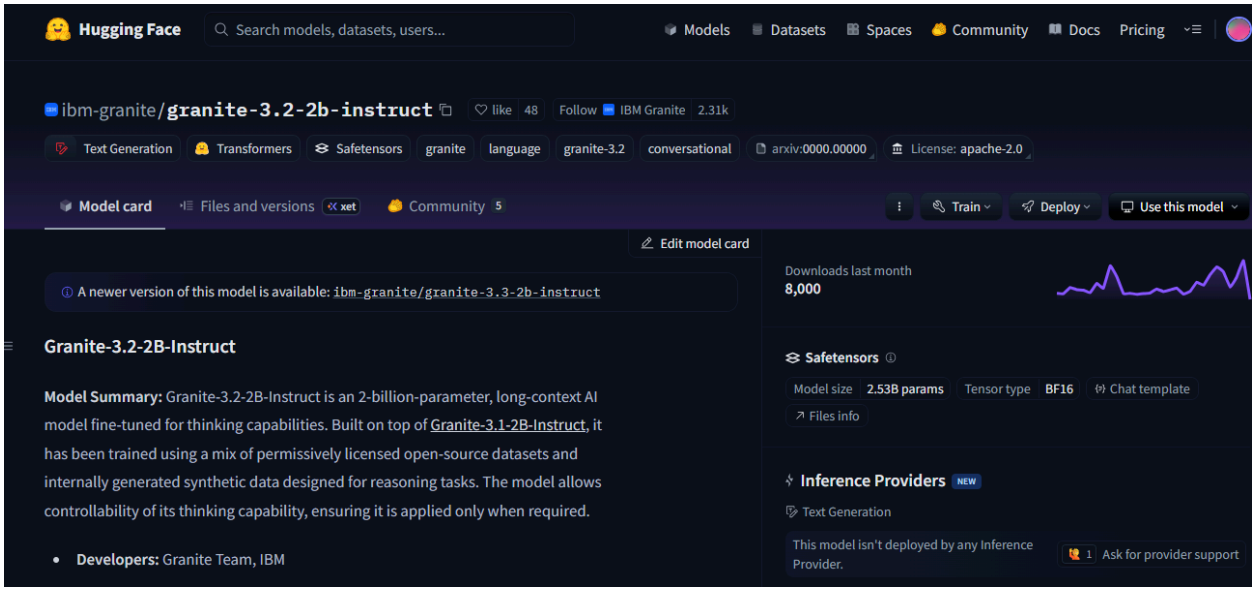
- Search for “Hugging face” in any browser.



- 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.



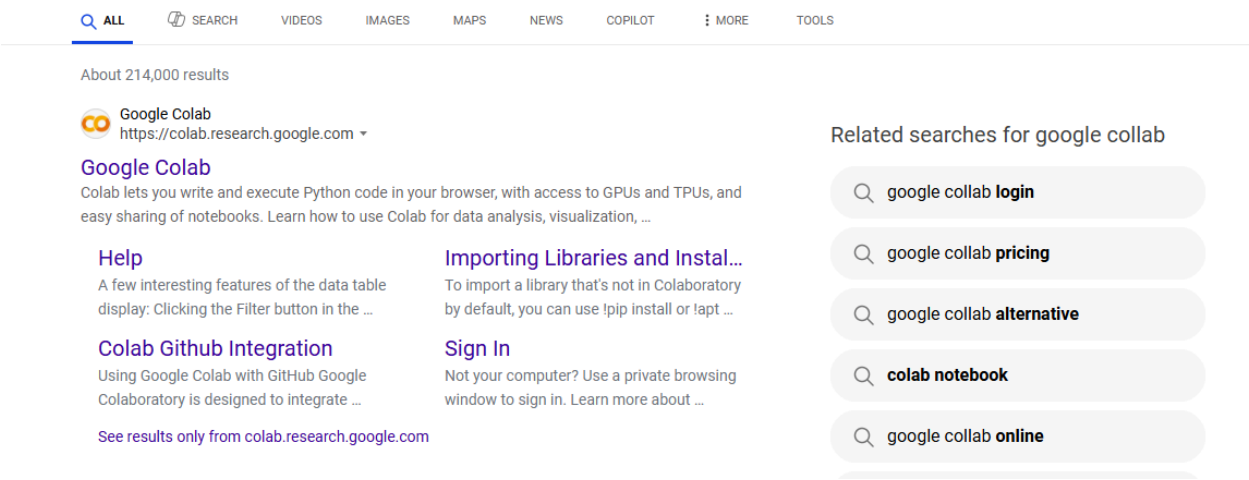
- Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.



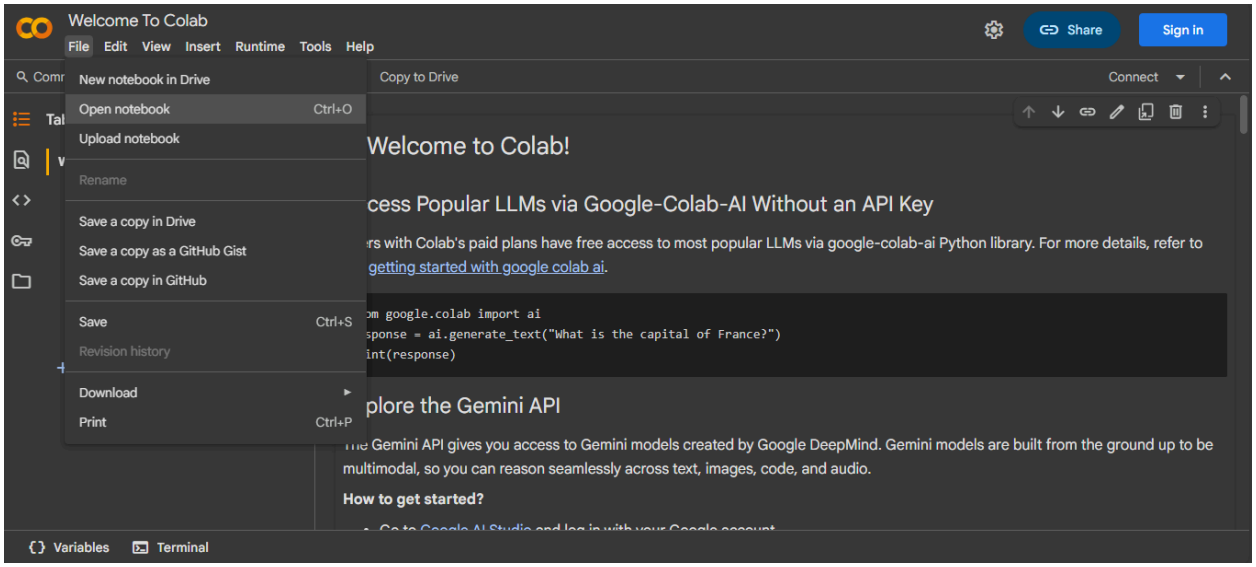
- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

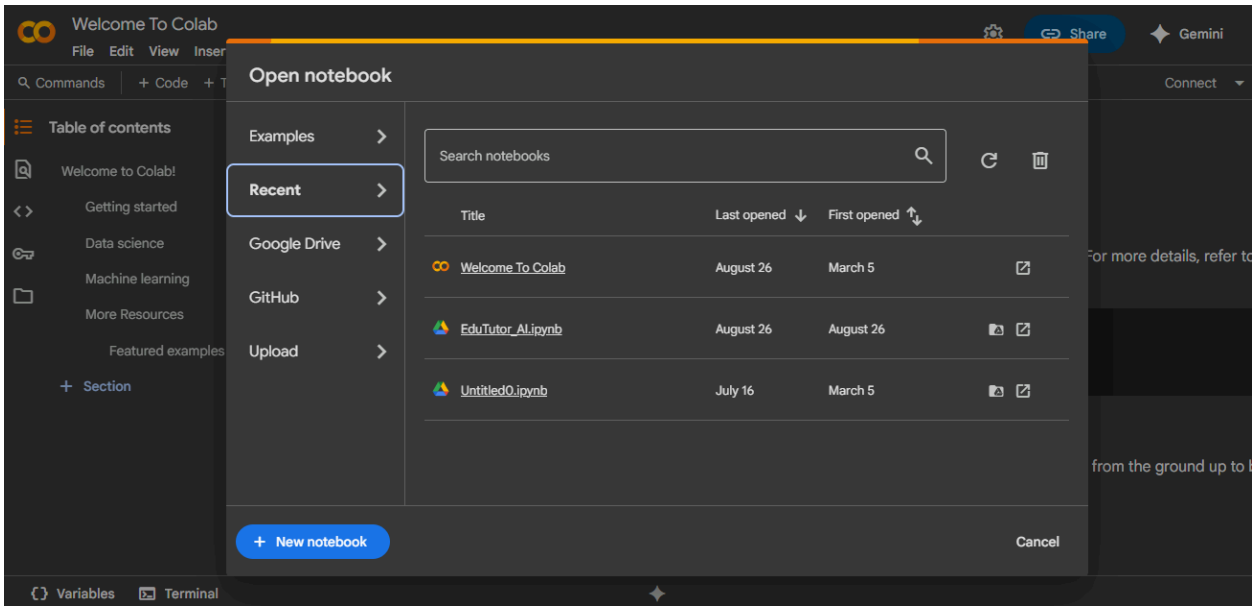
- Search for “Google collab” in any browser.



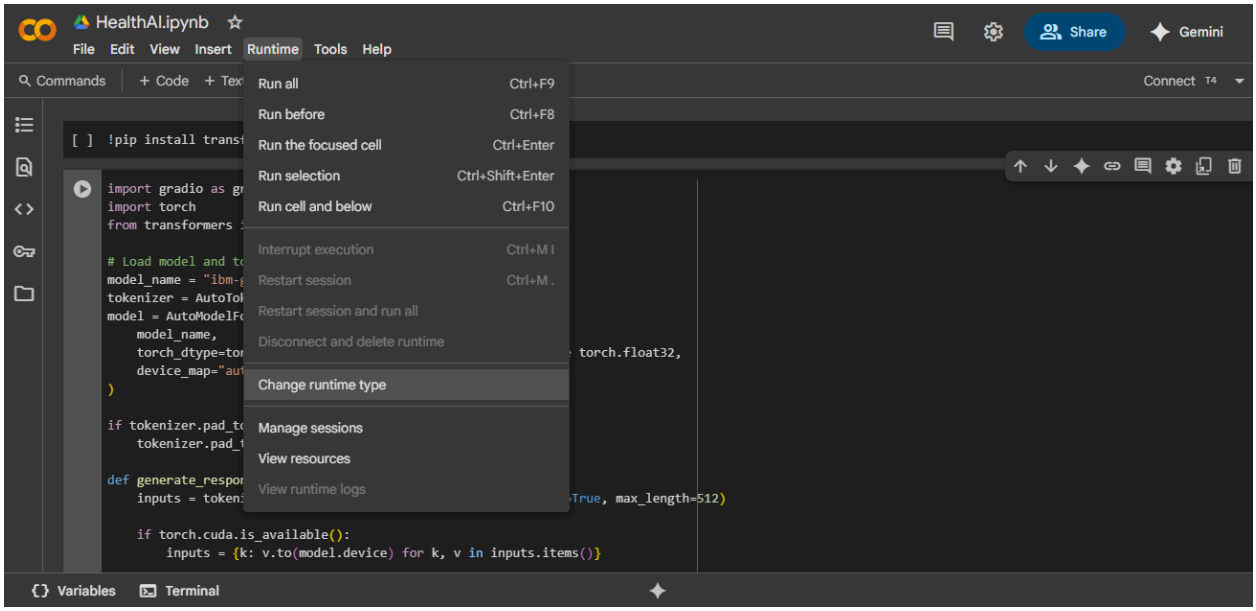
- Click on the first link (Google Colab), then click on “Files” and then “Open Notebook”.



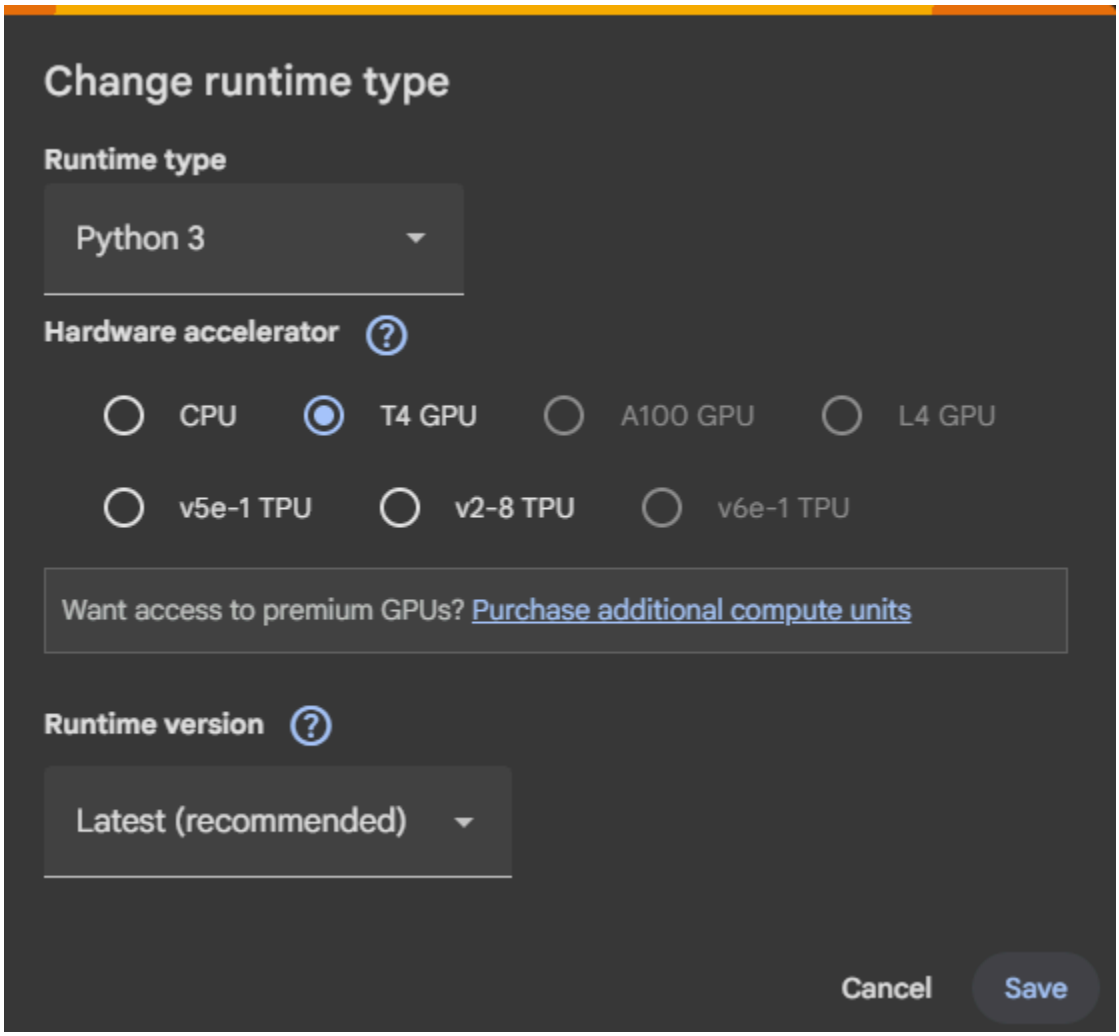
- Click on “New Notebook”



- Change the title of the notebook “Untitled” to “Citizen 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 “`!pip install transformers torch 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,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )
```

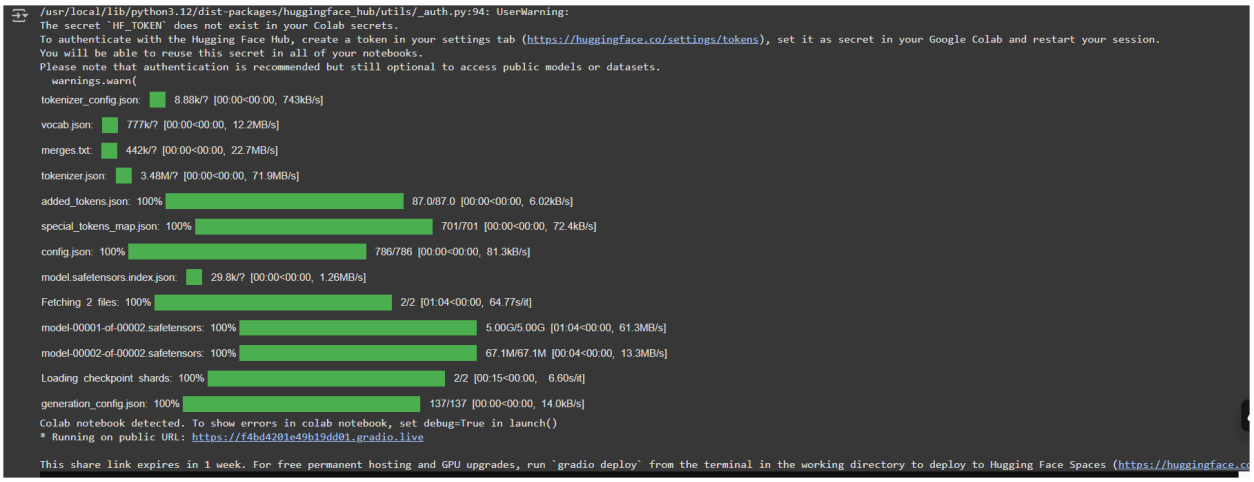
```python

esponse = tokenizer.decode(outputs[0], skip_special_tokens=True)
esponse = response.replace(prompt, "").strip()
eturn response

ity_analysis(city_name):
rompt = f"Provide a detailed analysis of {city_name} including:\n1. Crime Index and safety statistics\n2. Accident rates and traffic safety information\n3. Overall safety assessment\n\nCity: {cit
eturn generate_response(prompt, max_length=1000)

itizen_interaction(query):
rompt = f"As a government assistant, provide accurate and helpful information about the following citizen query related to public services, government policies, or civic issues:\n\nQuery: {query}
eturn generate_response(prompt, max_length=1000)

ate Gradio interface
gr.Blocks() as app:
r.Markdown("# City Analysis & Citizen Services AI")

ith gr.Tabs():
  with gr.TabItem("City Analysis"):
      with gr.Row():
          with gr.Column():
              city_input = gr.Textbox(
                  label="Enter City Name",
                  placeholder="e.g., New York, London, Mumbai...",
                  lines=1
              )
              analyze_btn = gr.Button("Analyze City")

          with gr.Column():
              city_output = gr.Textbox(label="City Analysis (Crime Index & Accidents)", lines=15)

      analyze_btn.click(city_analysis, inputs=city_input, outputs=city_output)
```

```
63
64  with gr.TabItem("Citizen Services"):
65      with gr.Row():
66          with gr.Column():
67              citizen_query = gr.Textbox(
68                  label="Your Query",
69                  placeholder="Ask about public services, government policies, civic issues...",
70                  lines=4
71              )
72              query_btn = gr.Button("Get Information")
73
74          with gr.Column():
75              citizen_output = gr.Textbox(label="Government Response", lines=15)
76
77      query_btn.click(citizen_interaction, inputs=citizen_query, outputs=citizen_output)
78
79 aunch(share=True)
```

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

Output:

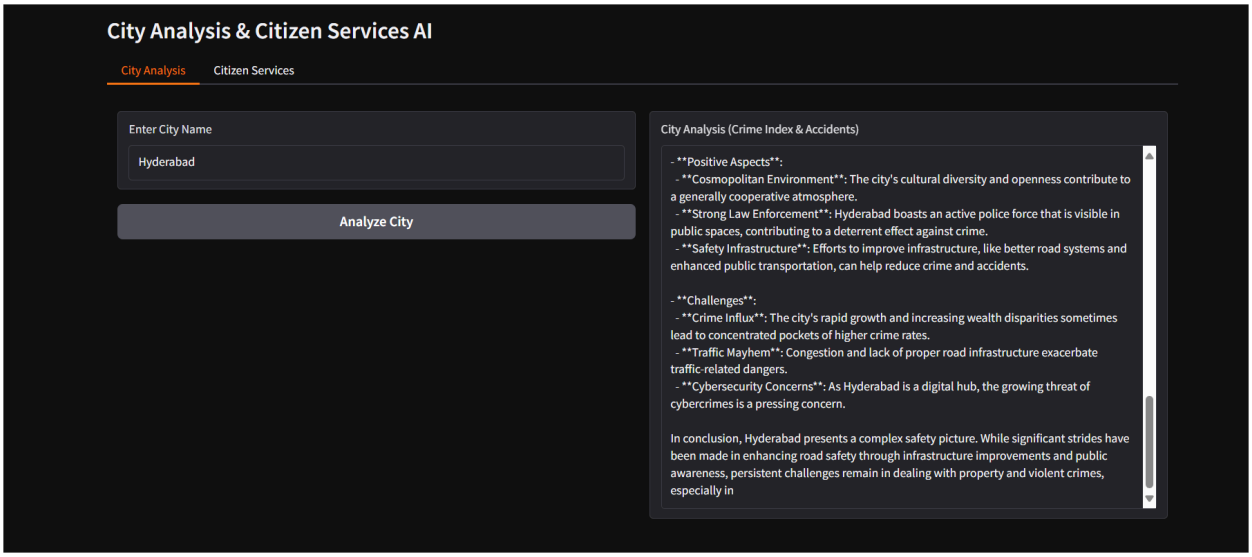
- Now you can see our model is being Downloaded and application is running.

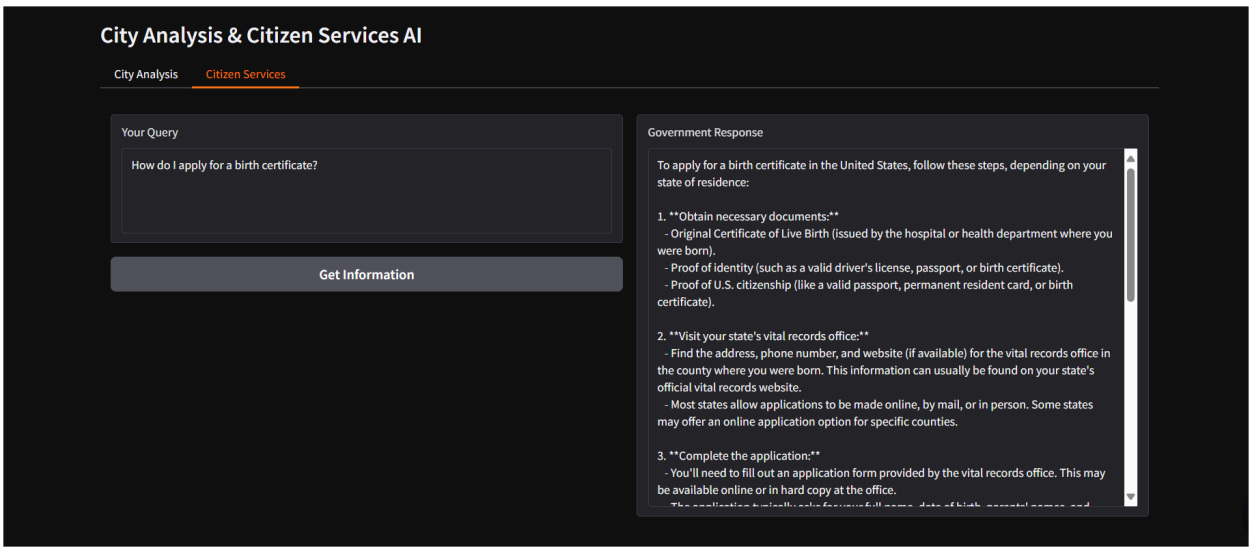


- Click on the URl to open the Gradio Application click on the link.

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://f4bd4201e49b19dd01.gradio.live
```

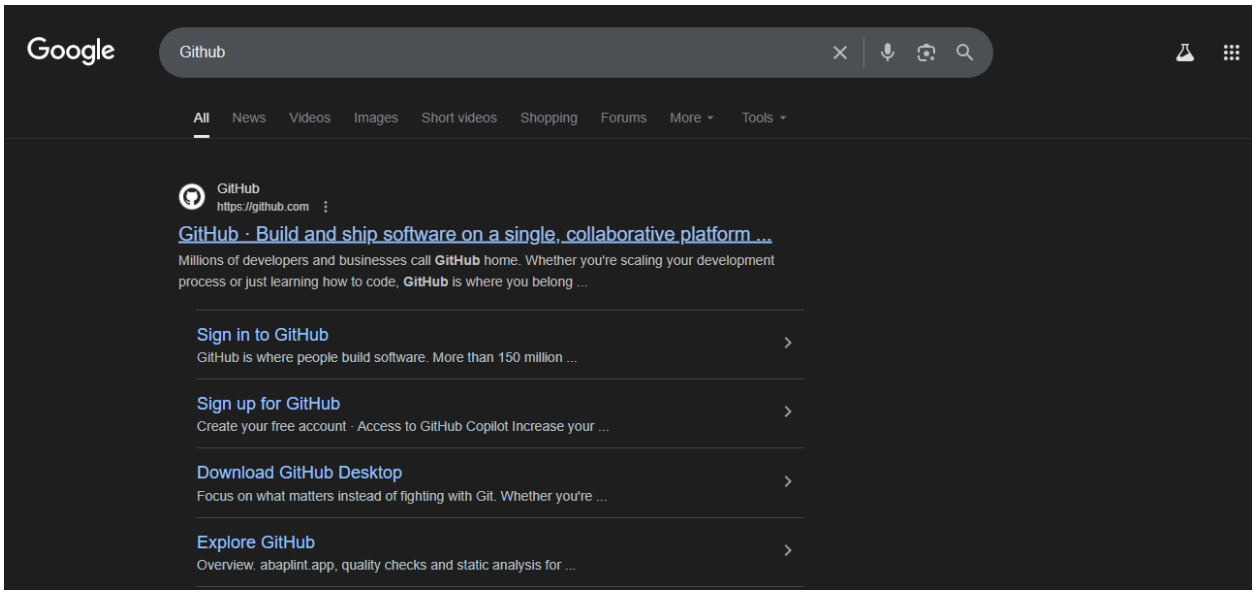
- You can View the Application is the running in the other tab.



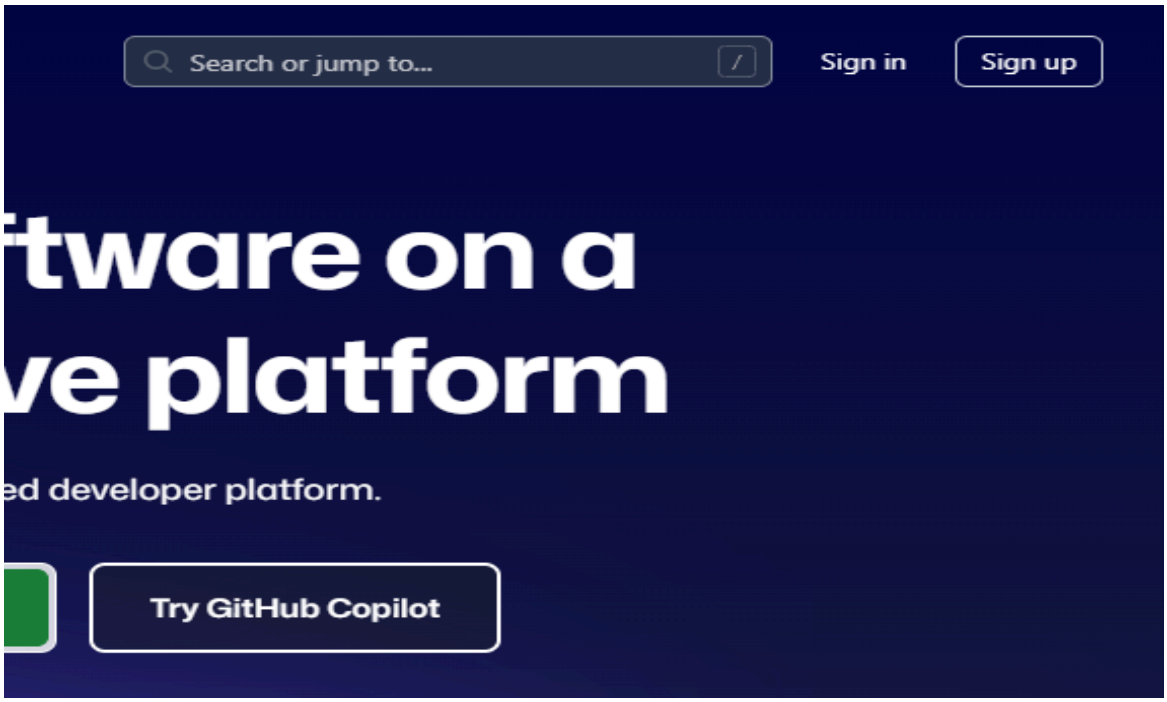


Activity-4: Upload Your Project in GitHub.

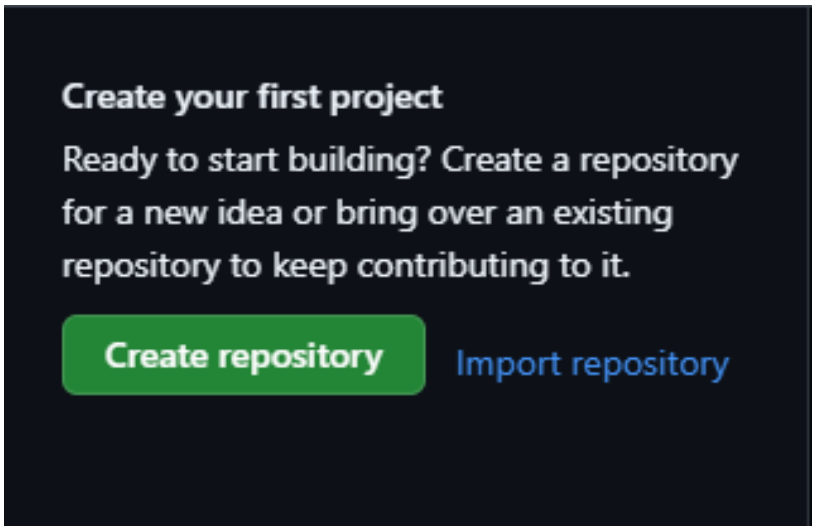
- Search for “GitHub” in any browser, then click on the first link (GitHub).



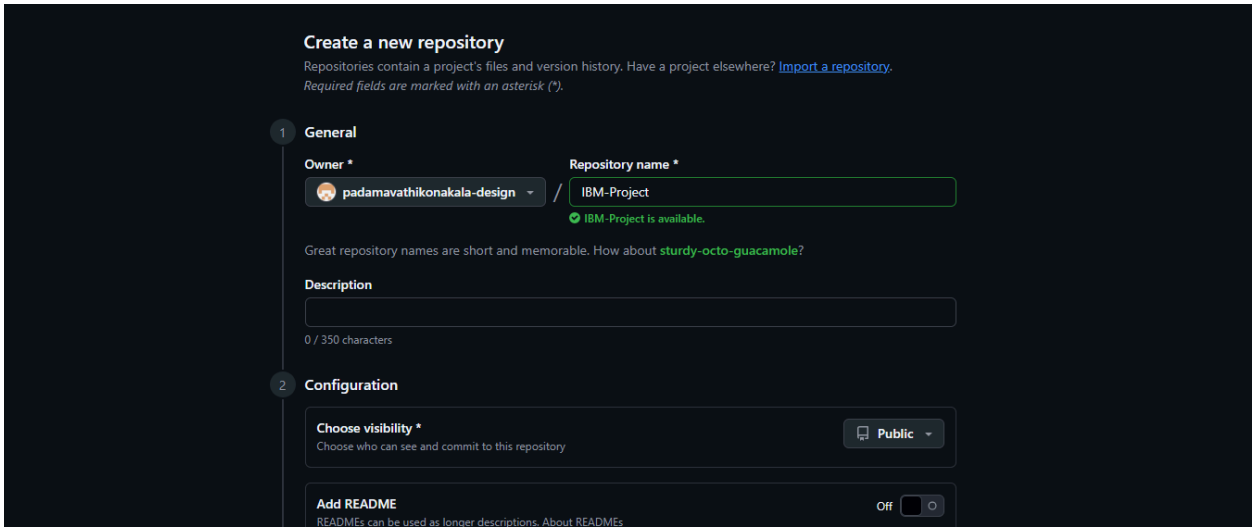
- Then click on “Signup” and create your own account in GitHub. If you already have an account click on “Sign in”



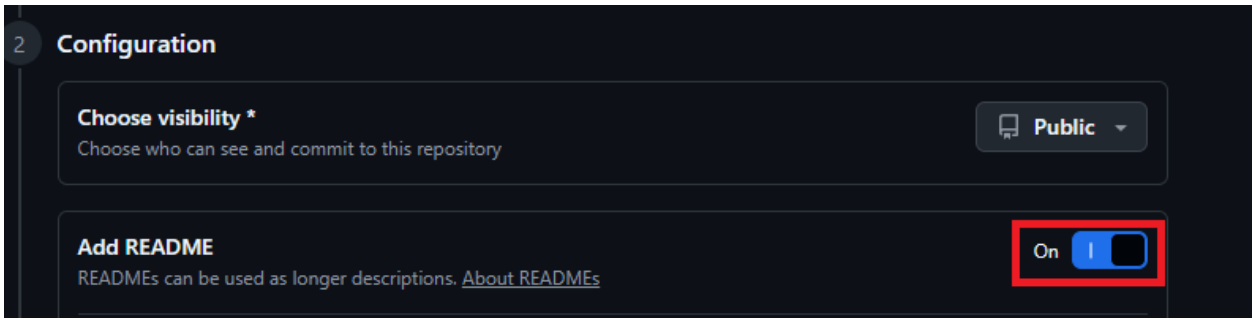
- Click on “Create repository”.



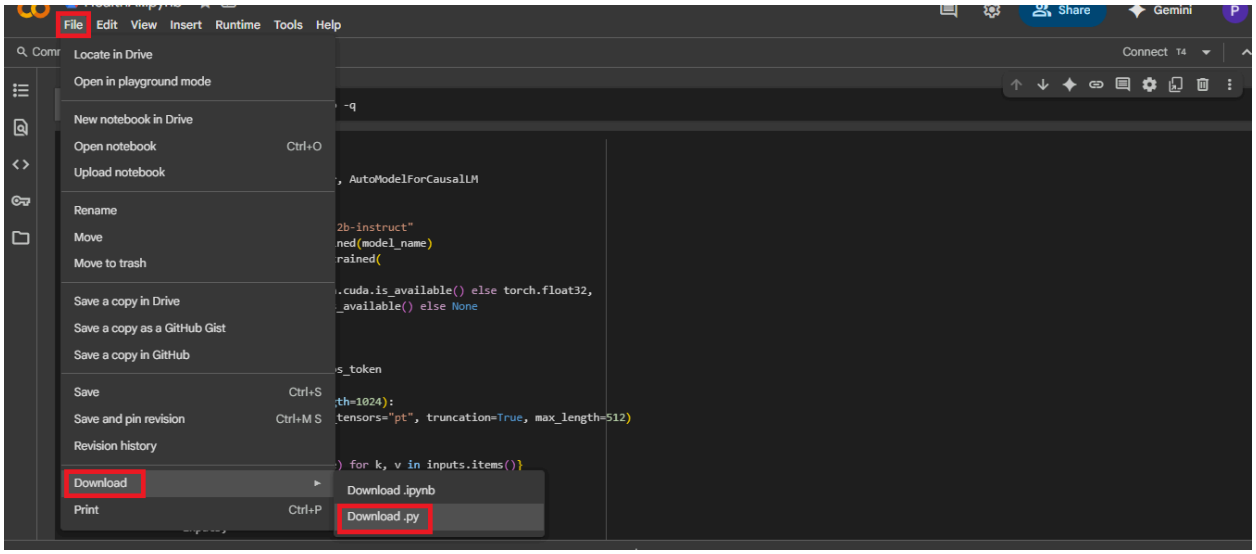
- In “General” Name your repo. (Here I have given “IBM-Project” as my repo name and it is available)



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

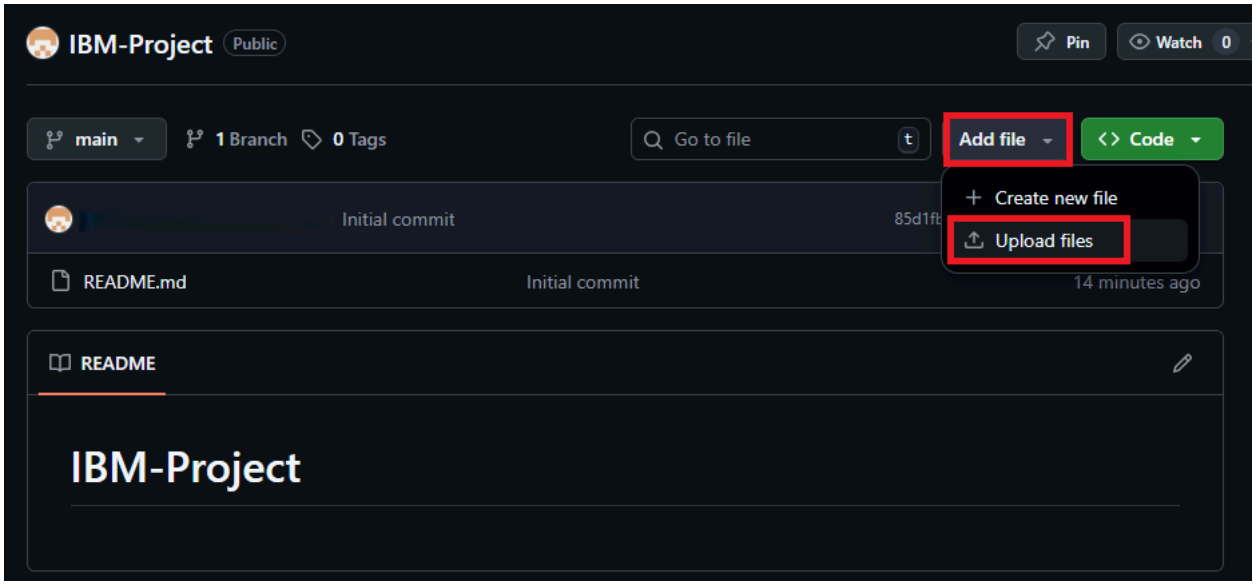


- Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.

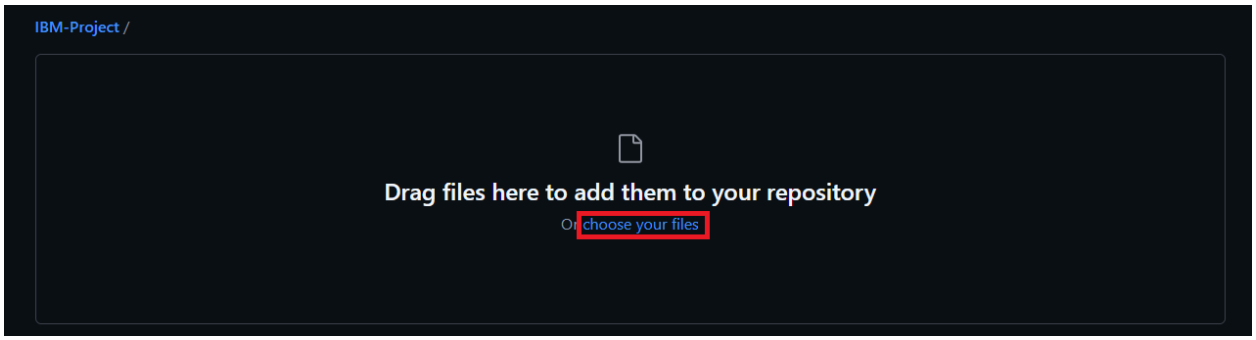


- Then your repository is created, then Click on “Add file” Option. Then Click

“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”.