

# Citizen AI: Intelligent Citizen Engagement Platform

*Generative AI with IBM*



## Team members

Saisundar.M  
Kaviyarasan.S  
Vignesh.M  
Sukumar.A  
Dhanush.M

## **Introduction of Citizen AI**

Citizen AI refers to the idea of developing artificial intelligence systems that act responsibly, ethically, and as productive members of society, much like a good citizen. The concept goes beyond just building powerful AI tools; it emphasizes embedding values such as fairness, accountability, transparency, and inclusivity into AI systems so they can contribute positively to human life.

With the increasing role of AI in everyday decision-making—whether in healthcare, education, governance, or business—there is a pressing need to ensure that these systems behave in ways that align with human rights, laws, and societal norms. Citizen AI encourages organizations, governments, and developers to treat AI not merely as a technological tool but as a social entity that should uphold civic responsibilities.

**The main goals of Citizen AI** include:

- Promoting trust and fairness in AI decisions.
- Avoiding biases and discrimination.
- Ensuring transparency in how AI works.
- Supporting responsible innovation for the benefit of society.

**In short**, Citizen AI is about making AI not just smart but also responsible, ensuring it acts as a positive force for individuals, communities, and the world at large.

## **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.

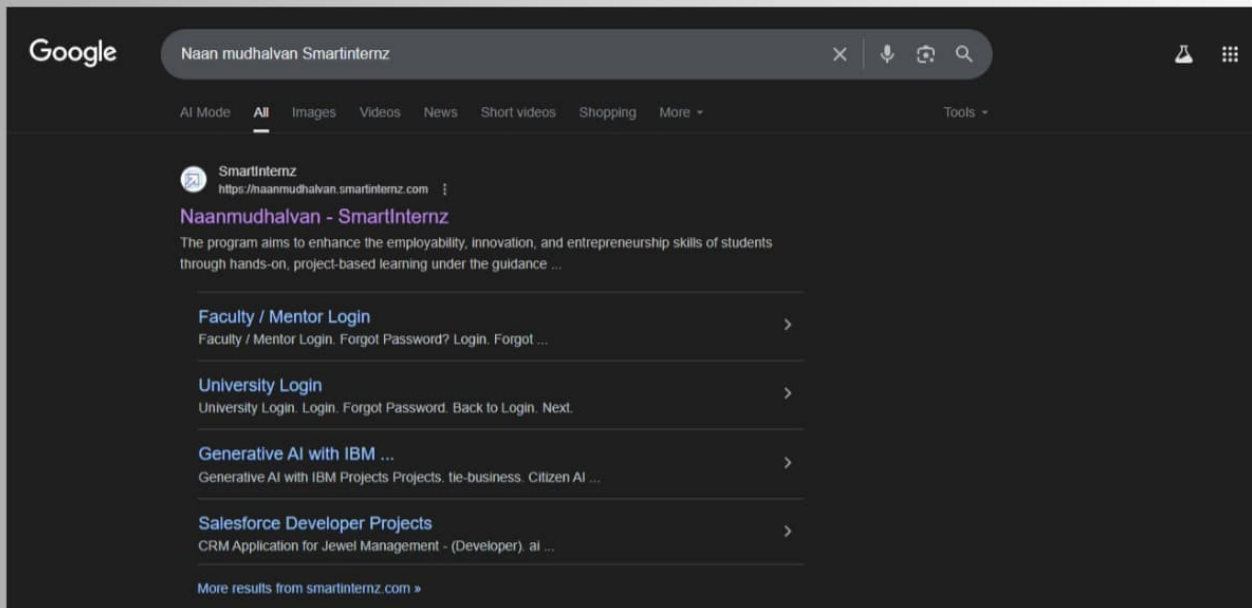
Activity-2: Choosing a IBM Granite Model From Hugging Face.

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.

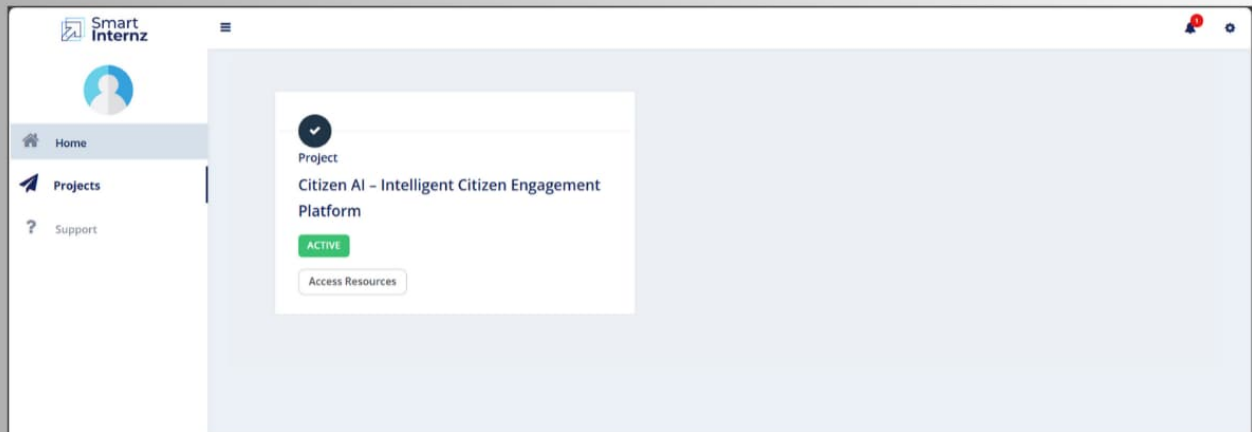


- Then Click on the first link. ([Naanmudhalvan Smartinternz](https://naanmudhalvan.smartinternz.com)) Then login with your details.

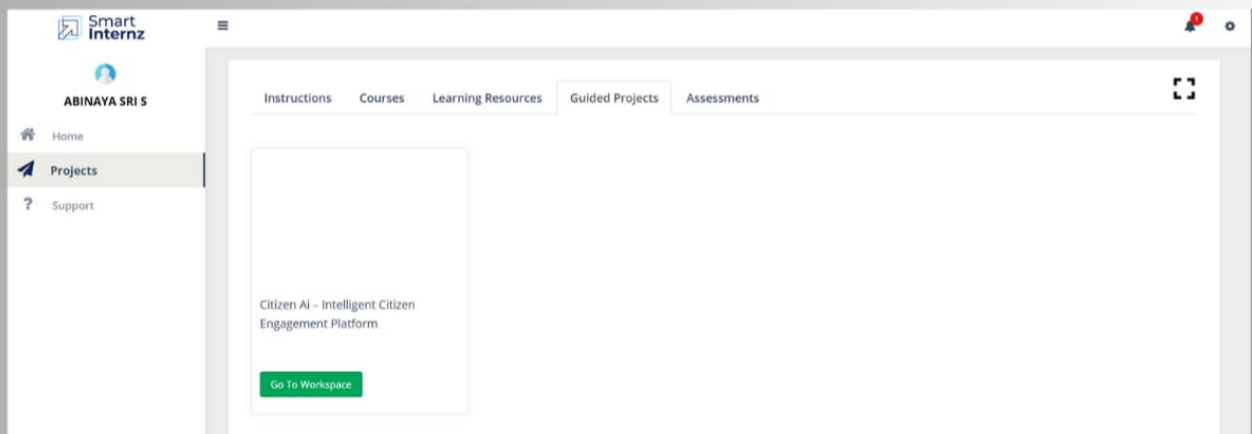


- Then you will be redirected to your account then click on “Projects”

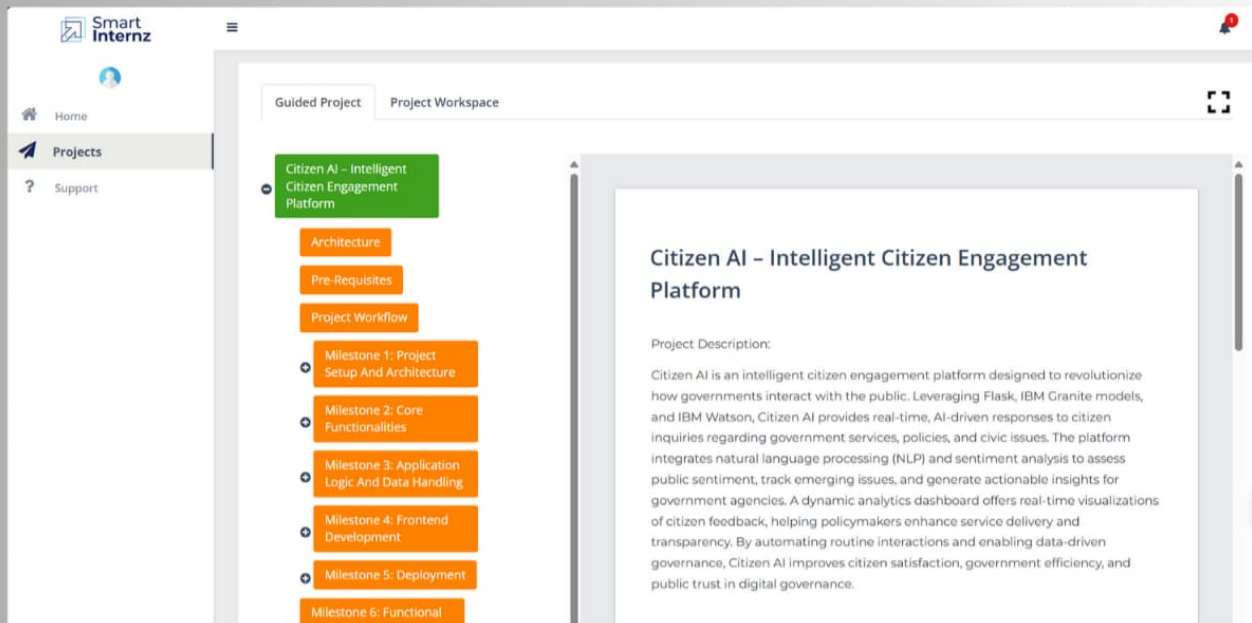
Section. There you can see which project you have enrolled in here it is “Citizen AI”.



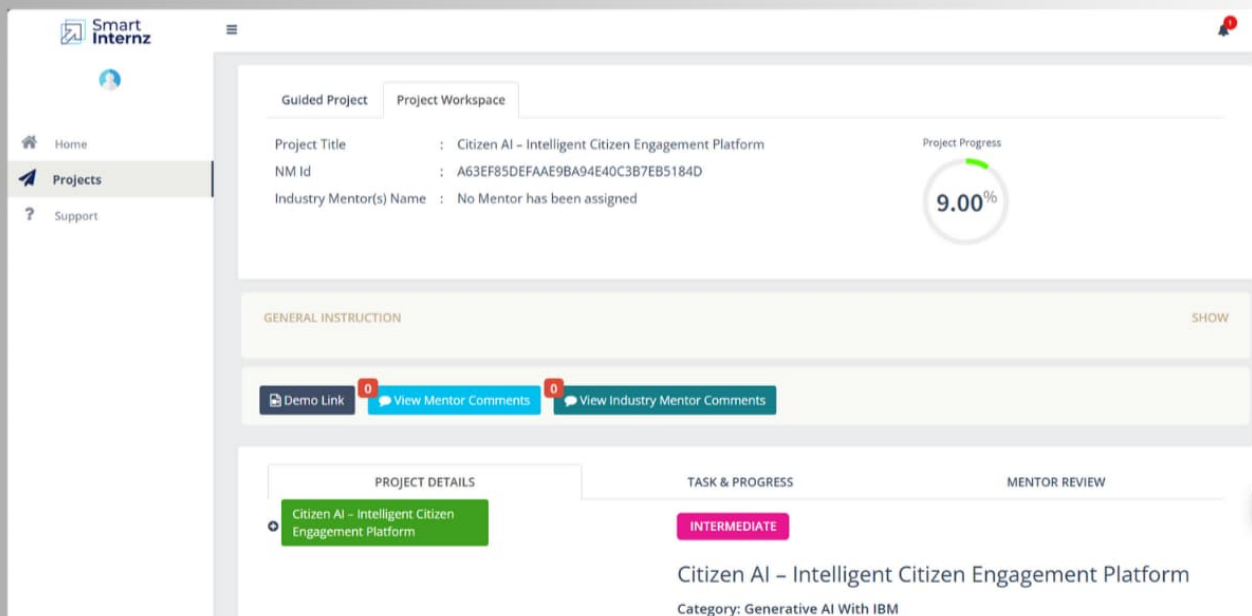
- Then click on “Access Resources” and go to the “Guided Project” Section.



- 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”.

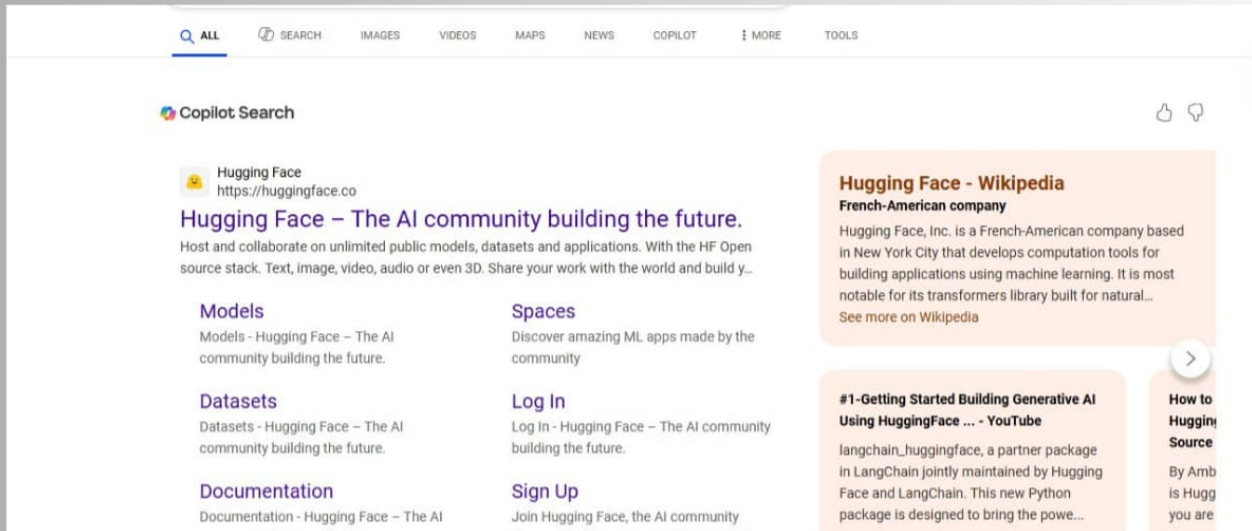


- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

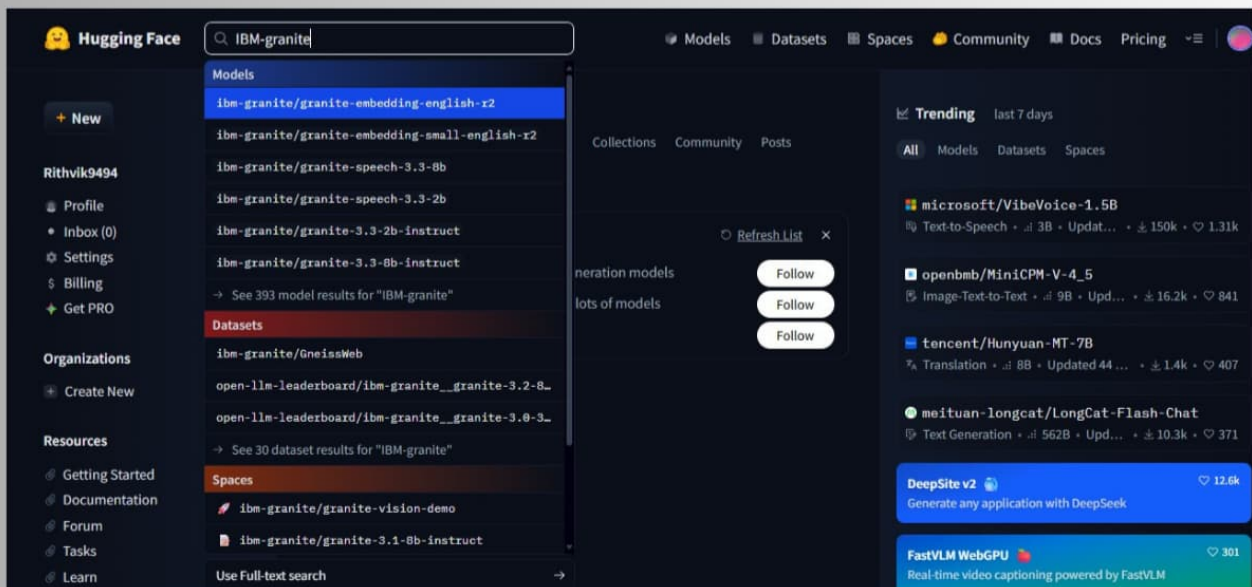


## ACTIVITY-2: CHOOSE A IBM GRANITE MODEL FROM HUGGING FACE.

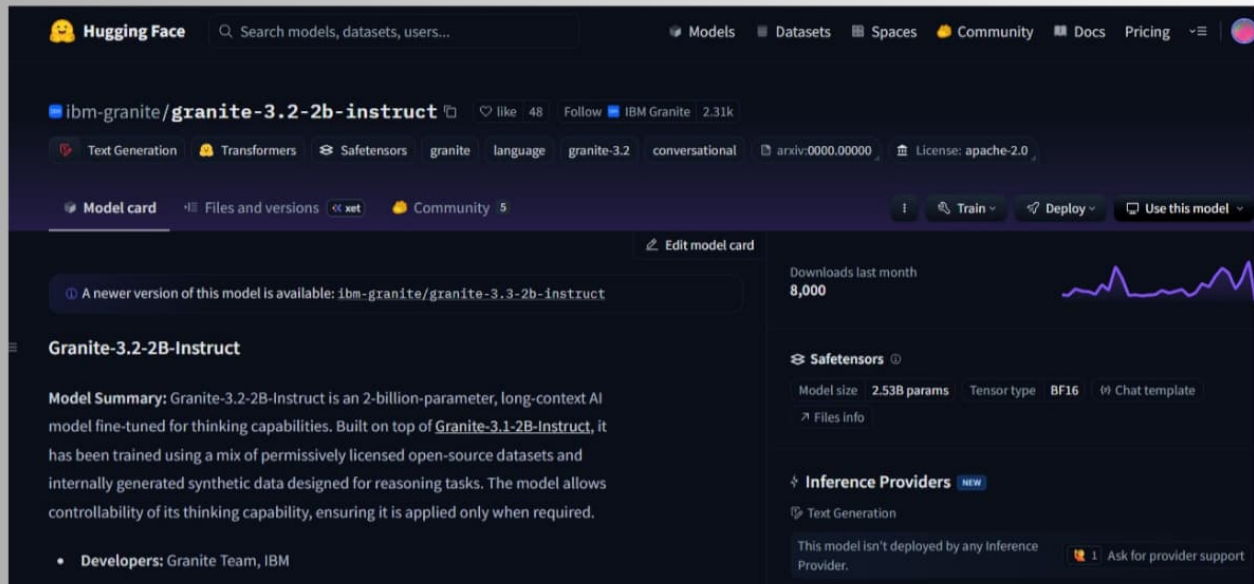
- Search for “Hugging face” in any browser.



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

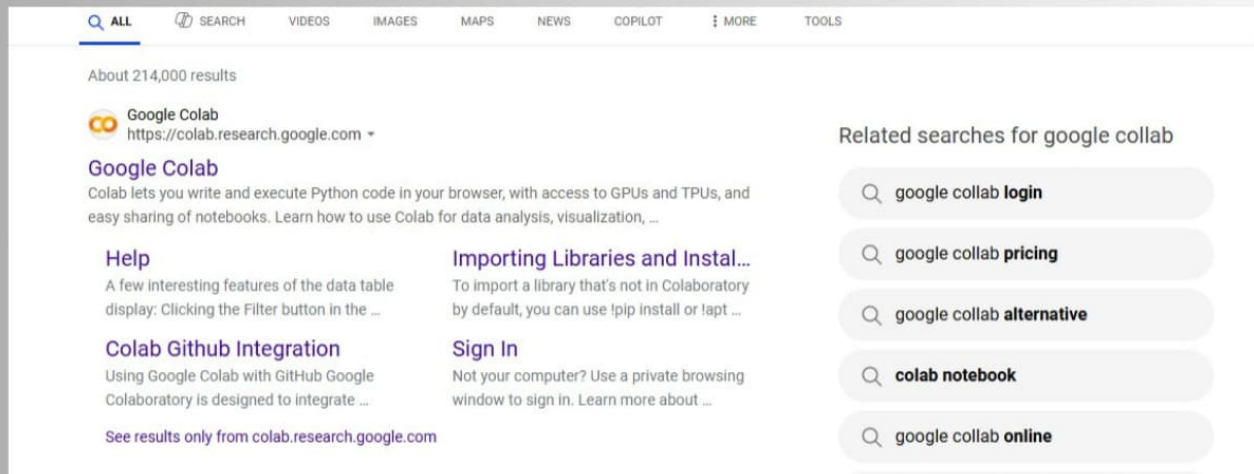


- Now we will start building our project in Google collab.

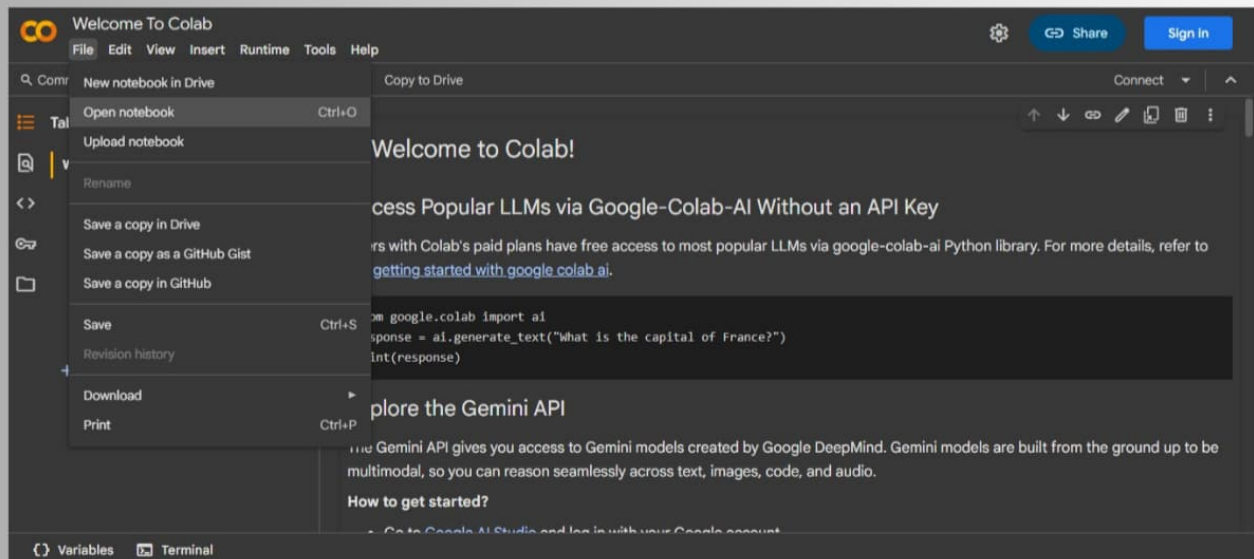


## ACTIVITY-3: RUNNING APPLICATION IN GOOGLE COLLAB.

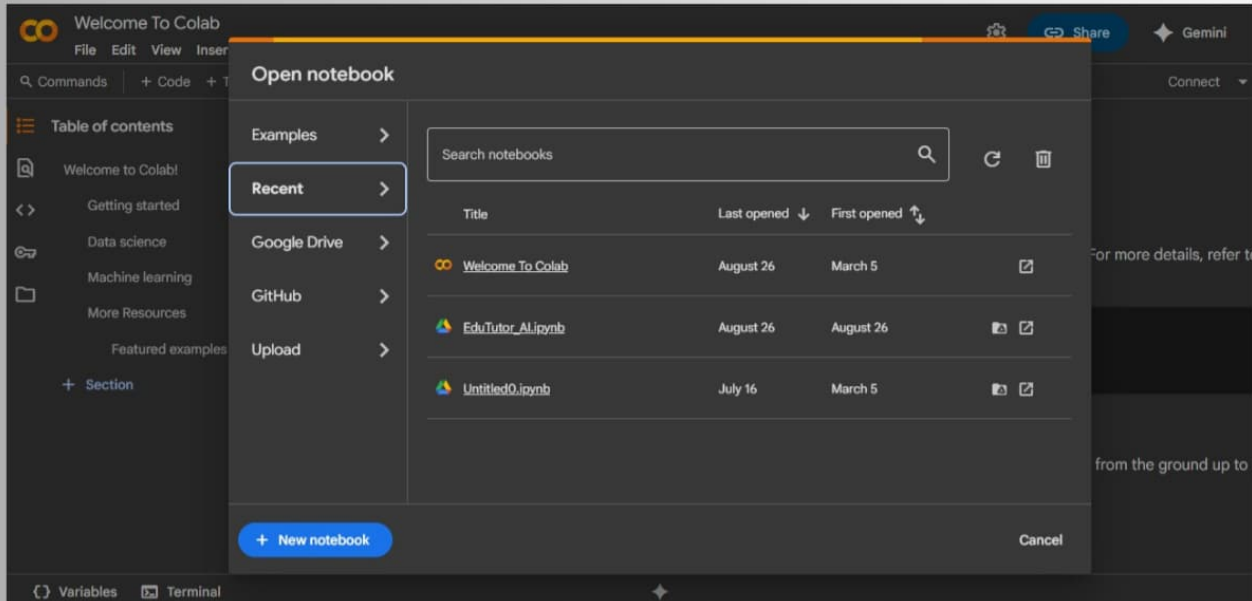
- Search for “Google collab” in any browser.



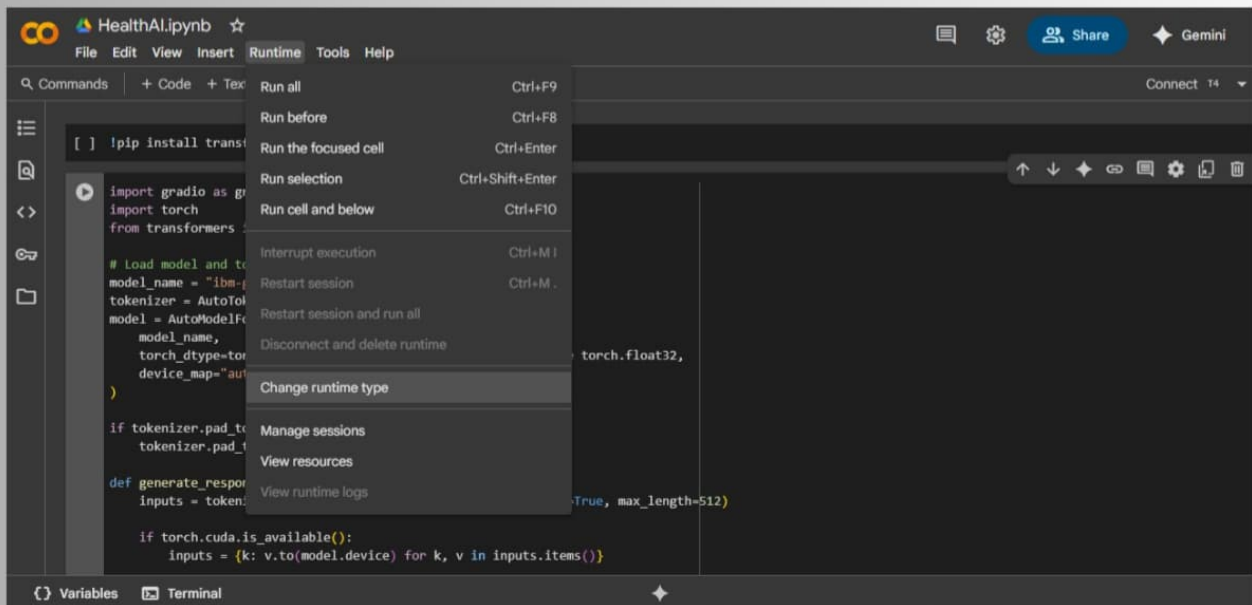
- Click on the first link ([Google Colab](https://colab.research.google.com)), then click on “Files” and then “Open Notebook”.



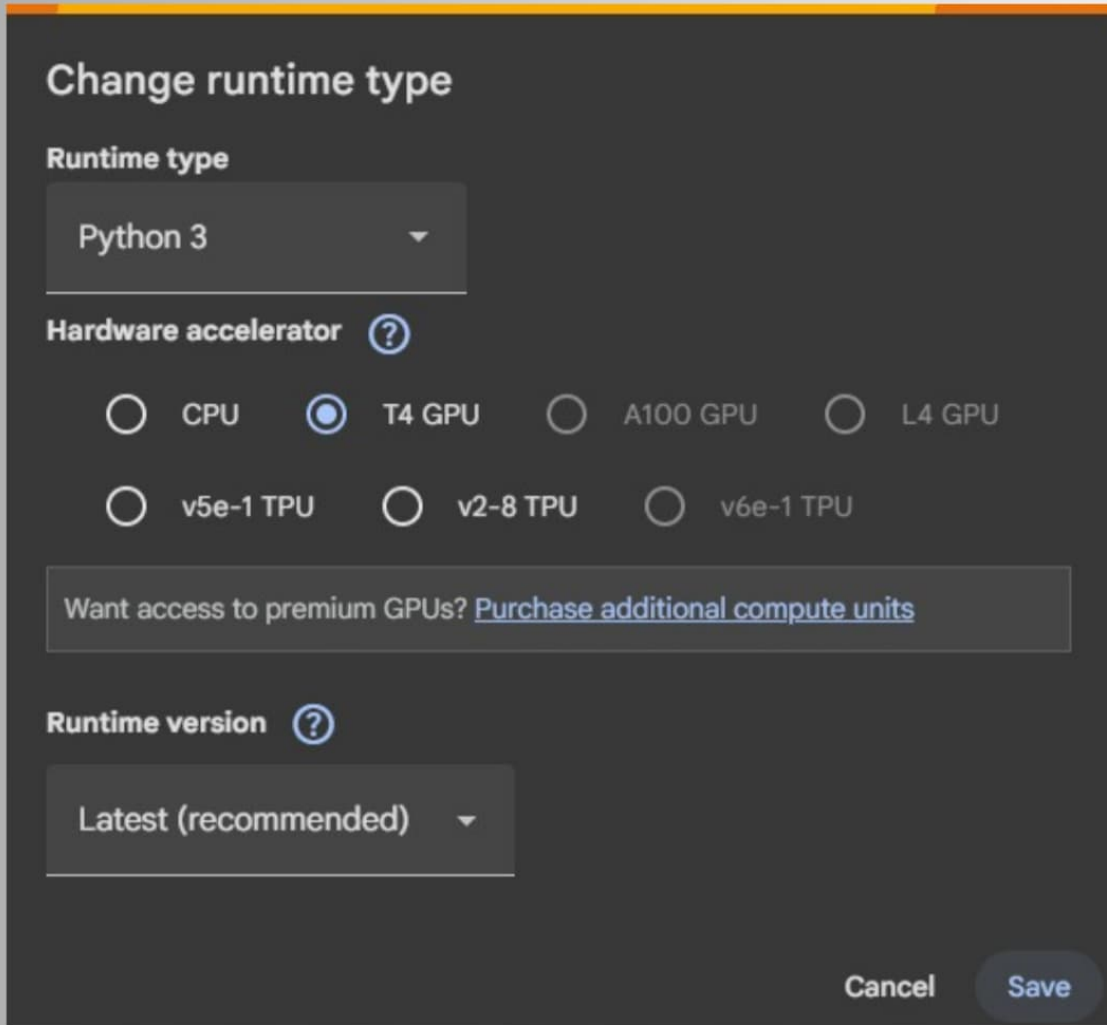
- Click on “New Notebook”



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



- Choose “T4 GPU” and click on “Save”



The dialog box titled "Change runtime type" has a dark background. At the top, the title "Change runtime type" is in white. Below it, the "Runtime type" section shows a dropdown menu with "Python 3" selected. 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](#)". The "Runtime version" section has a question mark icon and a dropdown menu with "Latest (recommended)" selected. At the bottom right, 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

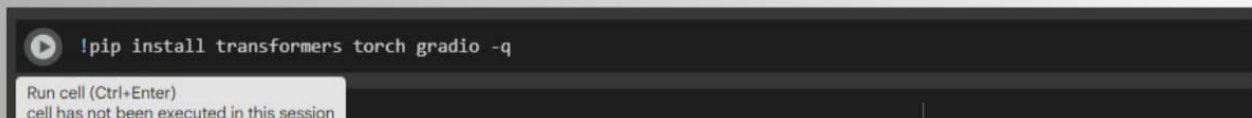
Want access to premium GPUs? [Purchase additional compute units](#)

**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.



A Jupyter Notebook cell with a dark background. The command "!pip install transformers torch gradio -q" is entered in the input area. Below the input area, there is a button that says "Run cell (Ctrl+Enter)" and a message that says "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.

```

1 import gradio as gr
2 import torch
3 from transformers import AutoTokenizer, AutoModelForCausalLM
4
5 # Load model and tokenizer
6 model_name = "ibm-granite/granite-3.2-2b-instruct"
7 tokenizer = AutoTokenizer.from_pretrained(model_name)
8 model = AutoModelForCausalLM.from_pretrained(
9     model_name,
10     torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
11     device_map="auto" if torch.cuda.is_available() else None
12 )
13
14 if tokenizer.pad_token is None:
15     tokenizer.pad_token = tokenizer.eos_token
16
17 def generate_response(prompt, max_length=1024):
18     inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
19
20     if torch.cuda.is_available():
21         inputs = {k: v.to(model.device) for k, v in inputs.items()}
22
23     with torch.no_grad():
24         outputs = model.generate(
25             **inputs,
26             max_length=max_length,
27             temperature=0.7,
28             do_sample=True,
29             pad_token_id=tokenizer.eos_token_id
30 )

```

```

31
32 response = tokenizer.decode(outputs[0], skip_special_tokens=True)
33 response = response.replace(prompt, "").strip()
34 return response
35
36 def city_analysis(city_name):
37     prompt = 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: {city_name}"
38     return generate_response(prompt, max_length=1000)
39
40 def citizen_interaction(query):
41     prompt = 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}"
42     return generate_response(prompt, max_length=1000)
43
44 # Gradio interface
45 gr.Blocks() as app:
46     gr.Markdown("# City Analysis & Citizen Services AI")
47
48     with gr.Tabs():
49         with gr.TabItem("City Analysis"):
50             with gr.Row():
51                 with gr.Column():
52                     city_input = gr.Textbox(
53                         label="Enter City Name",
54                         placeholder="e.g., New York, London, Mumbai...",
55                         lines=1
56                     )
57                     analyze_btn = gr.Button("Analyze City")
58
59                 with gr.Column():
60                     city_output = gr.Textbox(label="City Analysis (Crime Index & Accidents)", lines=15)
61
62             analyze_btn.click(city_analysis, inputs=city_input, outputs=city_output)
63

```

```

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     launch(share=True)

```

Output:

```

/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, 743kB/s]
vocab.json: 777k/ [00:00<00:00, 12.2MB/s]
merges.txt: 442k/ [00:00<00:00, 22.7MB/s]
tokenizer.json: 3.48k/ [00:00<00:00, 71.9MB/s]
added_tokens.json: 100% [00:00<00:00, 6.62MB/s]
special_tokens_map.json: 100% [00:00<00:00, 72.4kB/s]
config.json: 100% [00:00<00:00, 81.3kB/s]
model.safetensors.index.json: 29.9k/ [00:00<00:00, 1.26MB/s]
Fetching 2 files: 100% [00:04<00:00, 64.7kB/s]
model-00001-of-00002.safetensors: 100% [00:04<00:00, 61.3MB/s]
model-00002-of-00002.safetensors: 100% [00:04<00:00, 53.3MB/s]
Loading checkpoint shards: 100% [00:15<00:00, 6.80kB/s]
generation_config.json: 100% [00:00<00:00, 14.0kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://f4bd4201e49b19dd01.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.

```

- 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.



## City Analysis & Citizen Services AI

City Analysis
Citizen Services

Enter City Name  
Chennai

Analyze City

City Analysis (Crime Index & Accidents)

1. Crime Index and Safety Statistics:  
  
Chennai, the capital of Tamil Nadu, India, has seen a mixed performance in terms of crime rates and safety over the past few years. The city's Crime Index, which is a composite measure of crime intensity, is relatively higher compared to national averages. As per the 2019 National Crime Records Bureau (NCRB) data, Chennai ranks 11th among Indian cities in terms of overall crime intensity.  
  
Key crime categories and their percentages in Chennai are:  
- Housebreaking and related offenses: 25.1%

## City Analysis & Citizen Services AI

City Analysis
Citizen Services

Your Query  
Planting trees

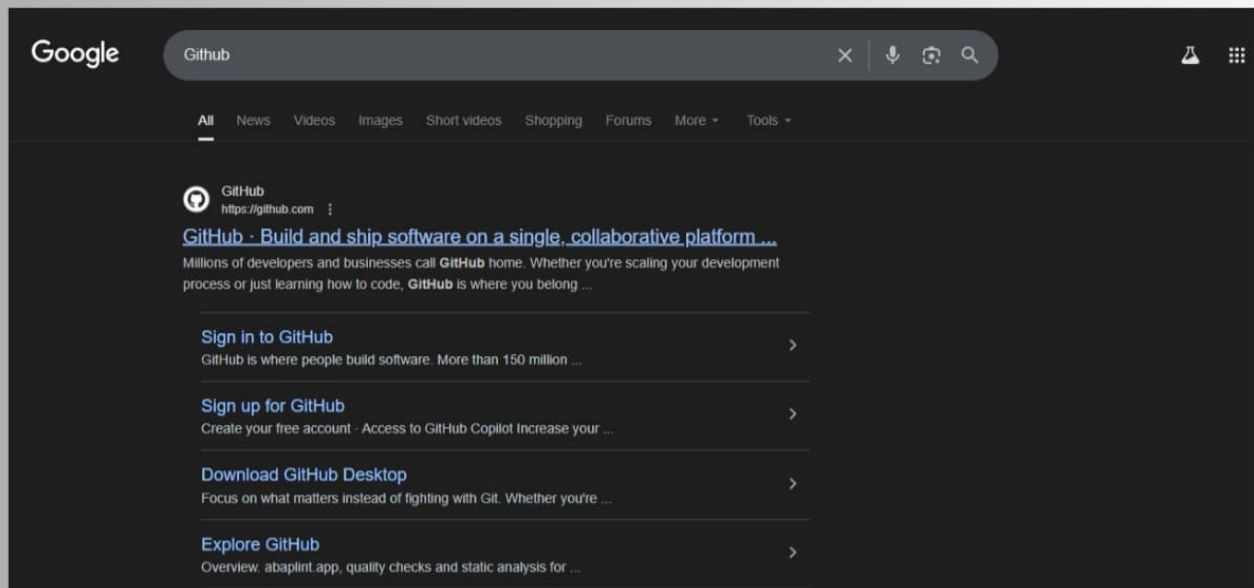
Get Information

Government Response

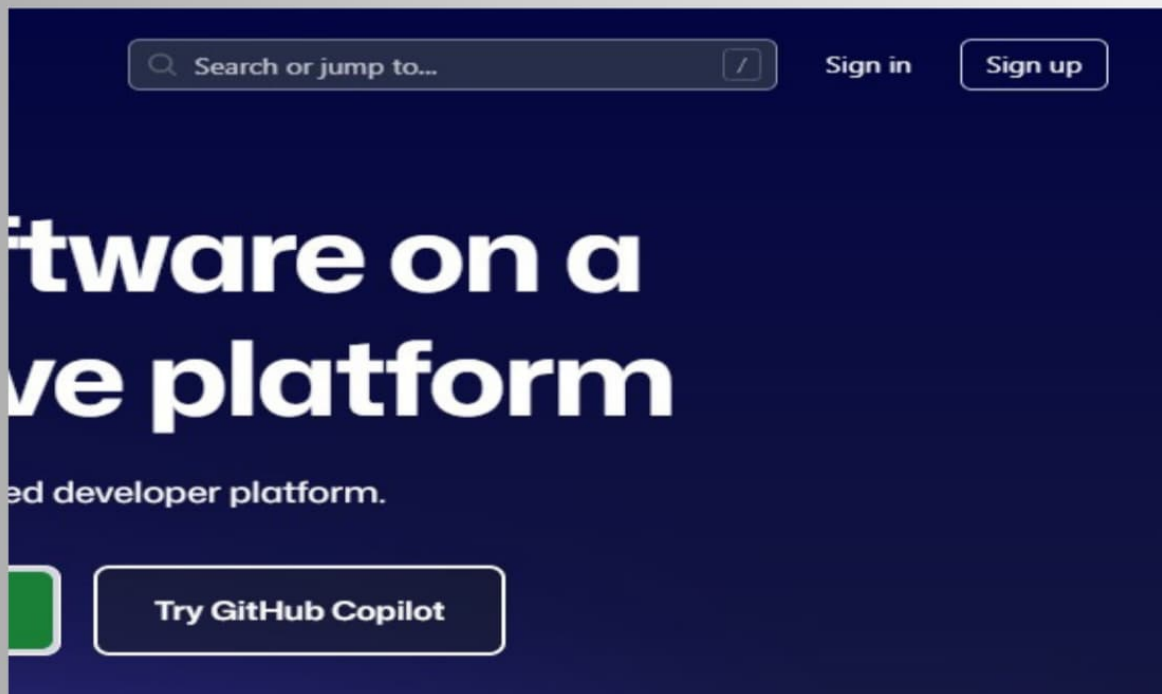
Dear Citizen,  
  
Thank you for your interest in tree planting contributing to our environment's health and sustainability. The government recognizes the importance of this initiative and offers several support mechanisms to facilitate and encourage tree planting across the country. Here are some key points and resources:

### Activity-4: Upload Your Project in GitHub.

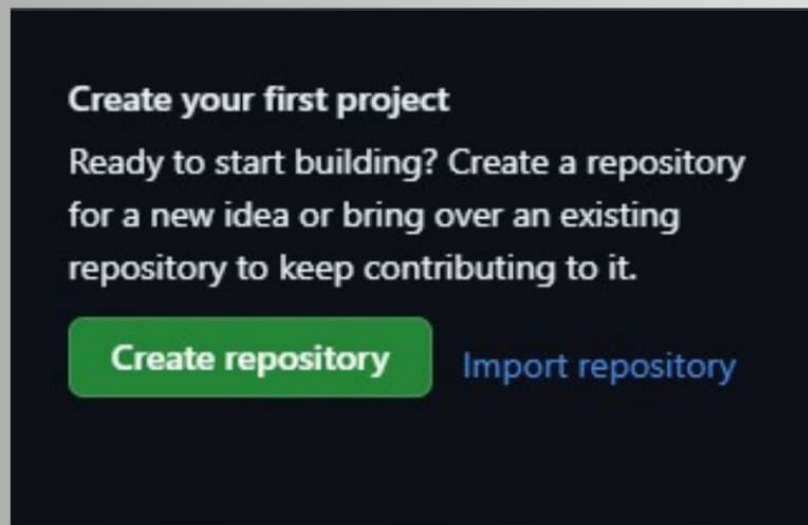
- Search for “GitHub” in any browser, then click on the first link ([GitHub](#)).



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

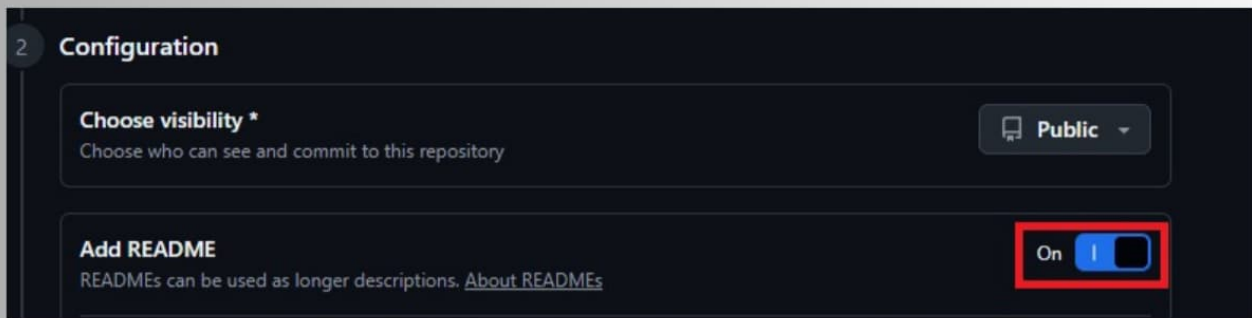


- Click on “Create repository”.

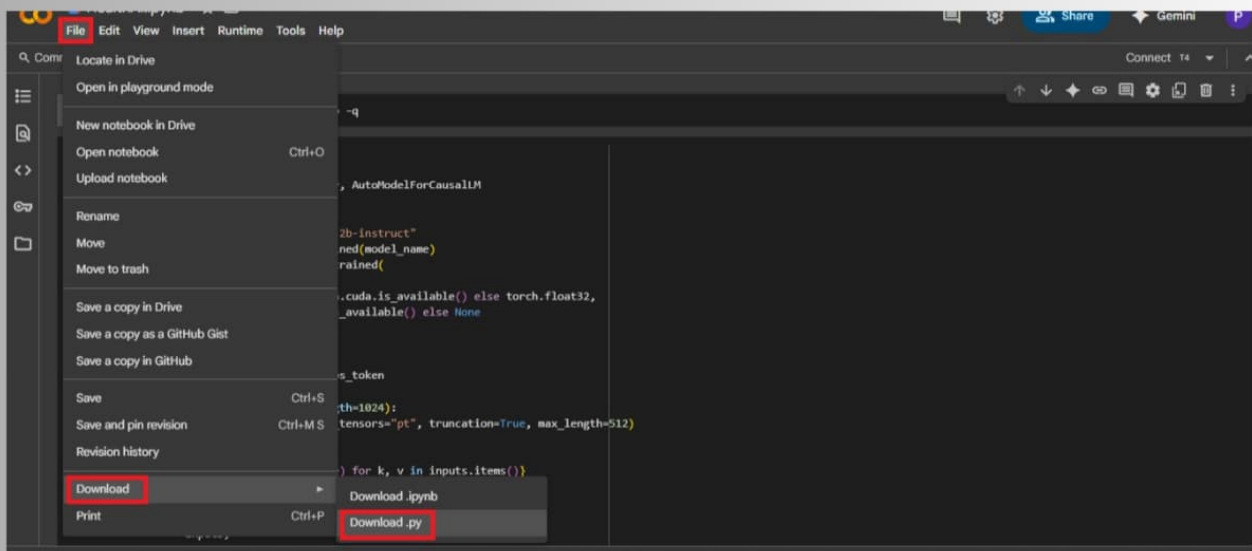


- In “General” Name your repo. (Here I have given “IBM-Project” as my repo name and it is available)

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

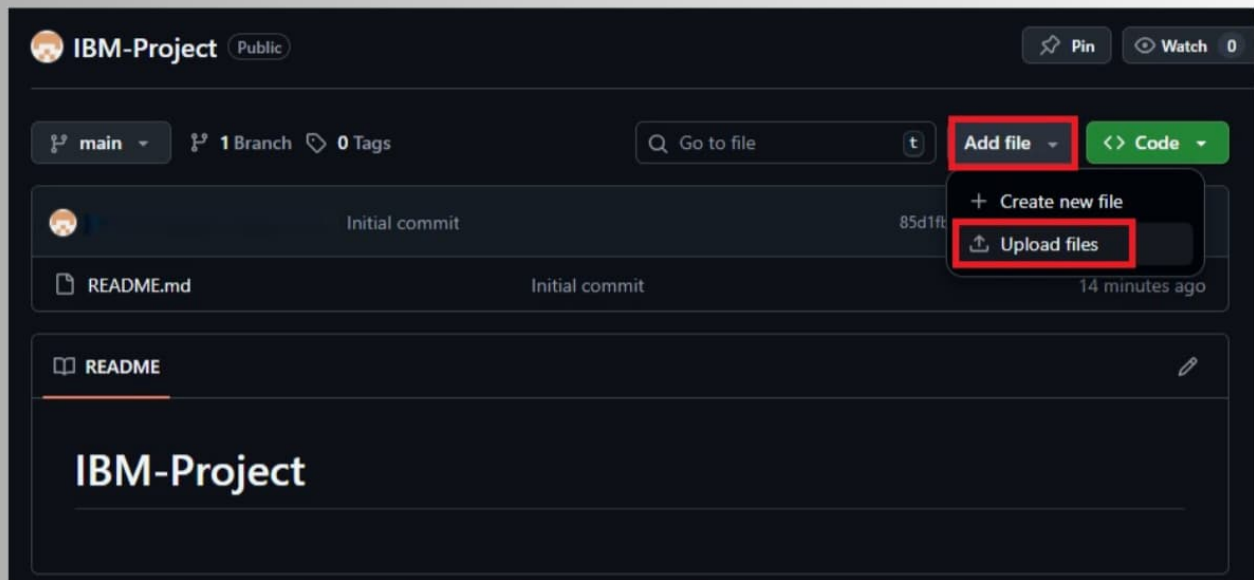


- Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.

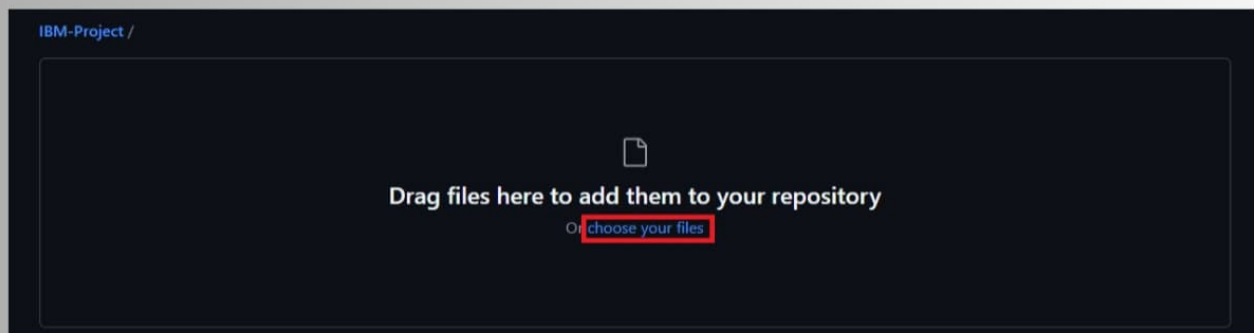


- Then your repository is created, then Click on “Add file” Option. Then Click

“Upload files” to upload your files.

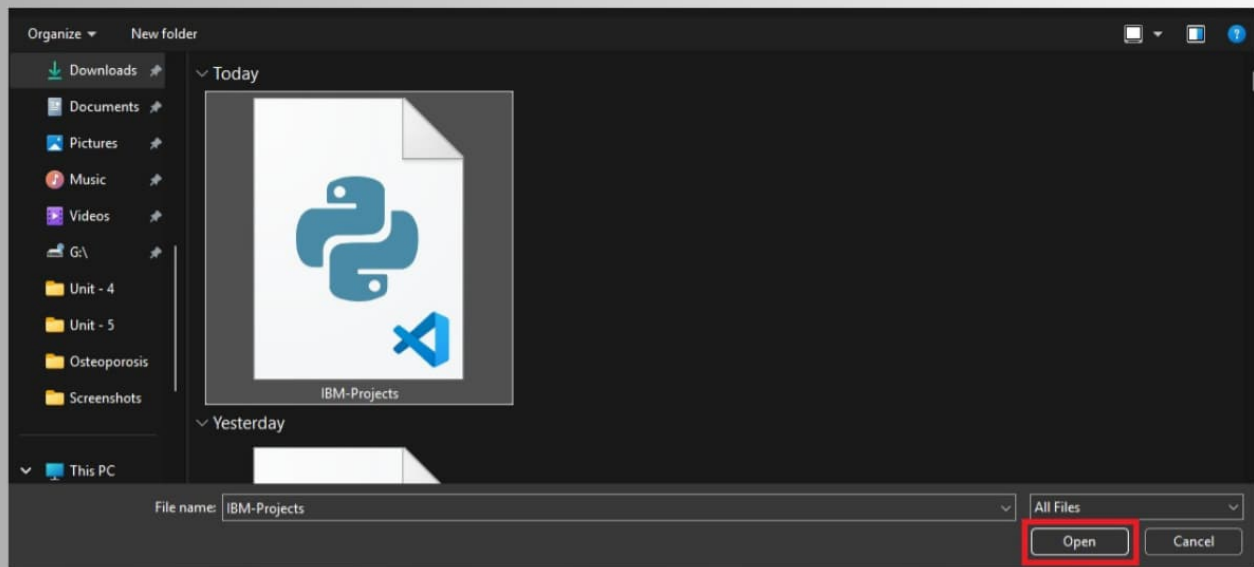


- Click on “choose your files”.



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





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

