

Sustainable Smart City Assistant Using IBM Granite LLM

Generative AI with IBM



Project Description:

Sustainable Smart City Assistant uses the Granite model from Hugging Face to help with city sustainability, governance, and citizen engagement. It includes quick tools for a City Health Dashboard, citizen feedback, document summaries and eco tips. This project will be deployed in Google Colab using Granite for easy setup and smooth 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.

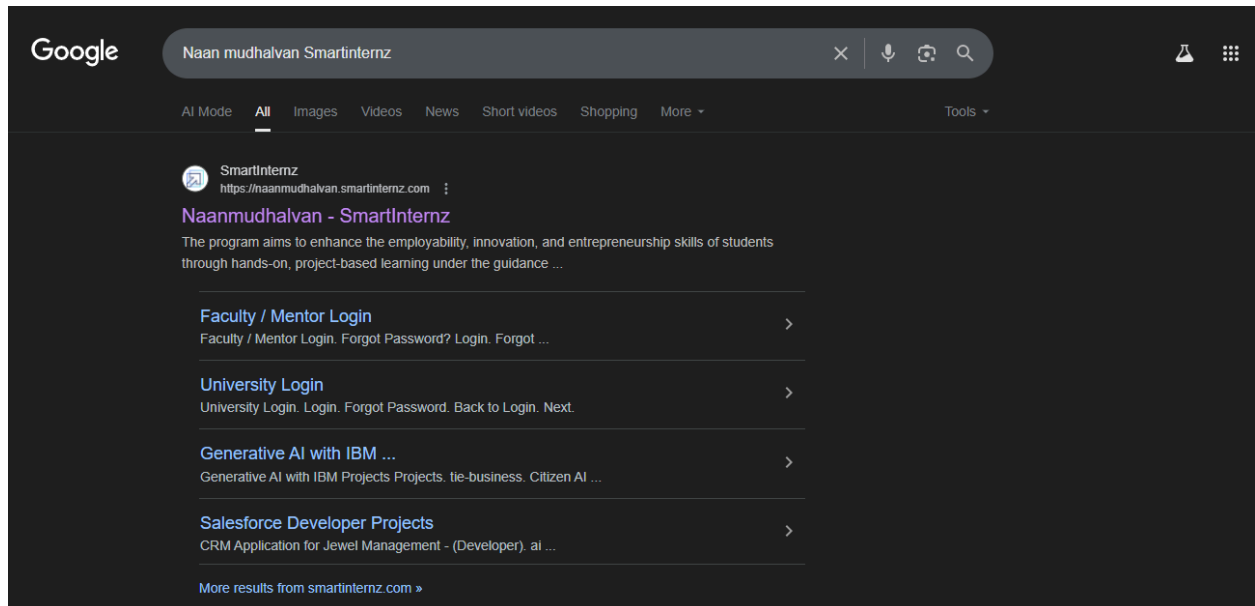
Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application In Google Colab.

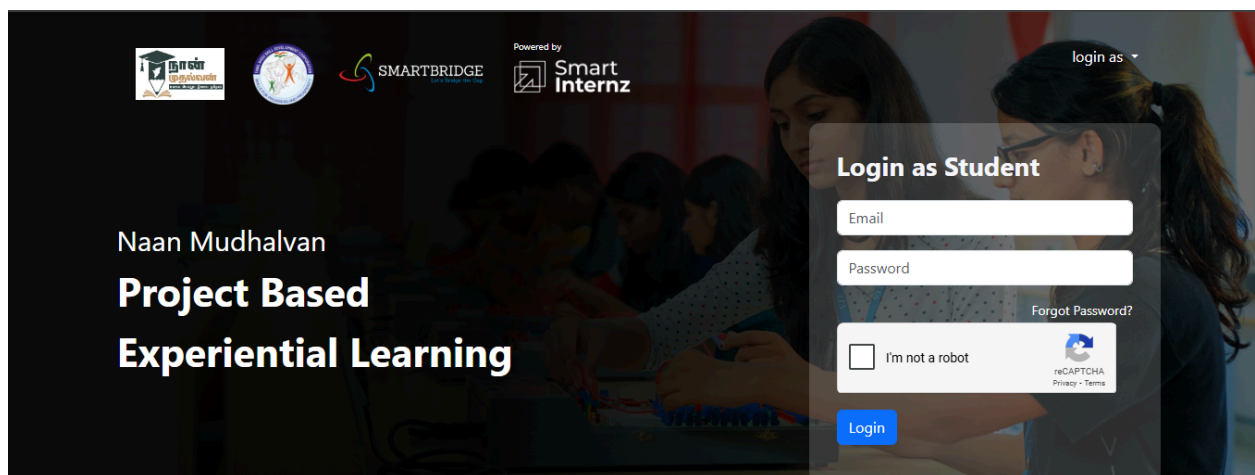
Activity-4: Upload your Project in Github.

Activity-1: Exploring Naan Mudhalavan Smart Interz Portal.

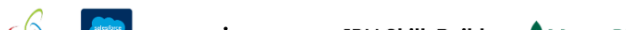
- Search for “Naan Mudhalavan Smart Interz” Portal in any Browser.



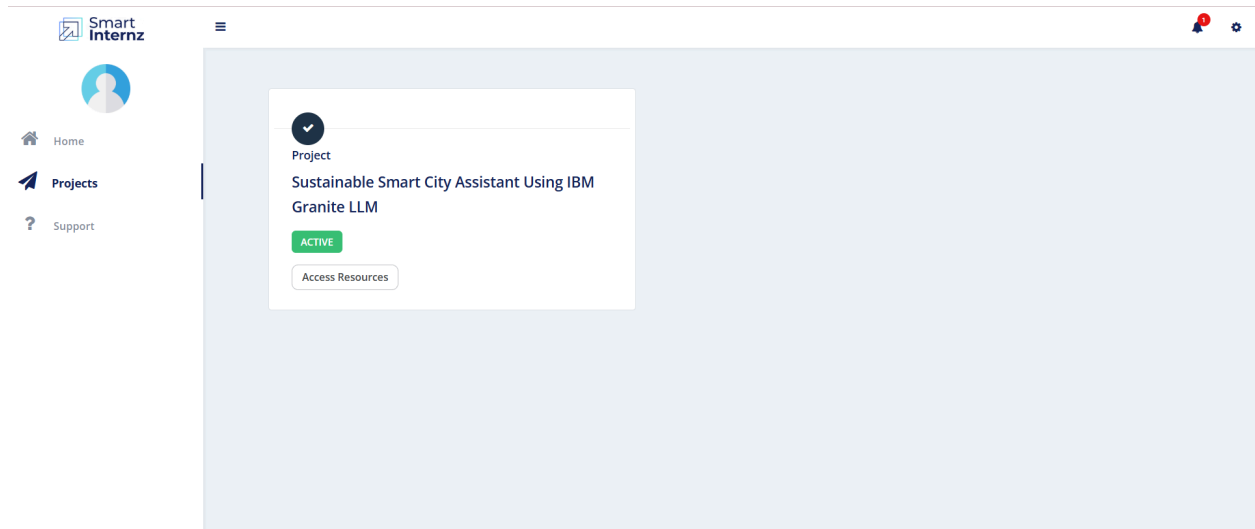
- Then Click on the first link. ([Naanmudhalvan Smartinternz](https://naanmudhalvan-smartinternz.com)) Then login with your details.



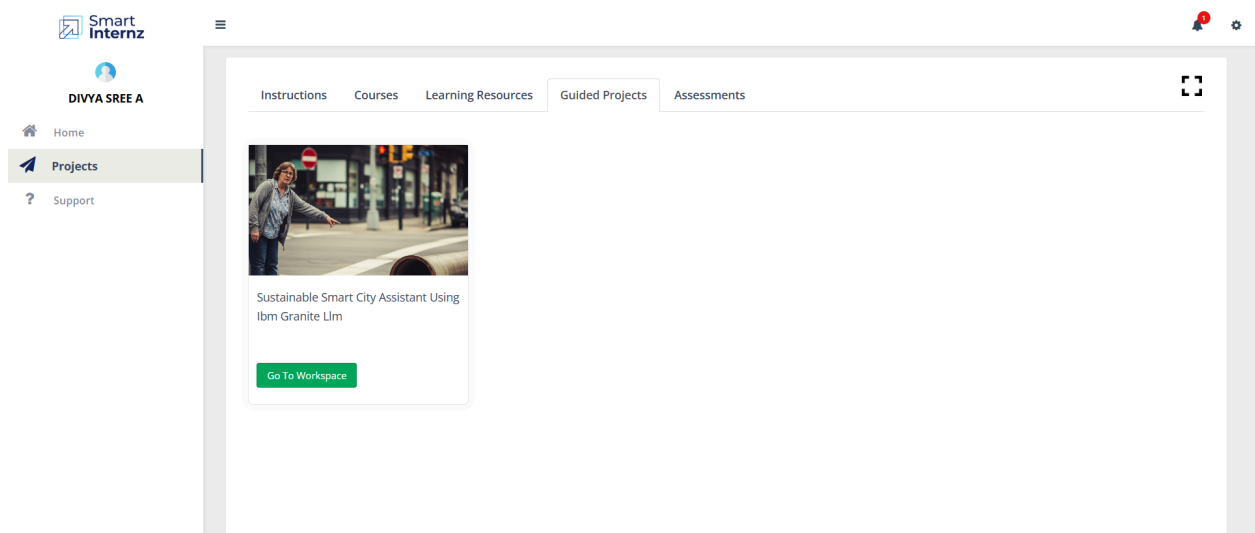
In Partnership with



- 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”.

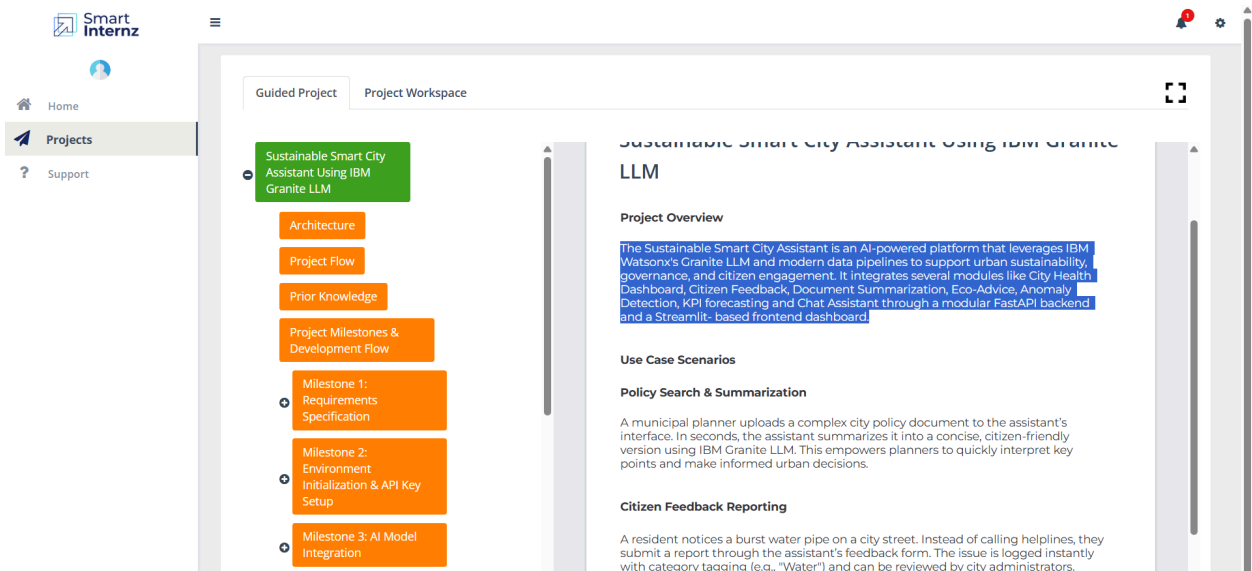


- Then click on “Access Resources” and go to the “Guided Project” Section.

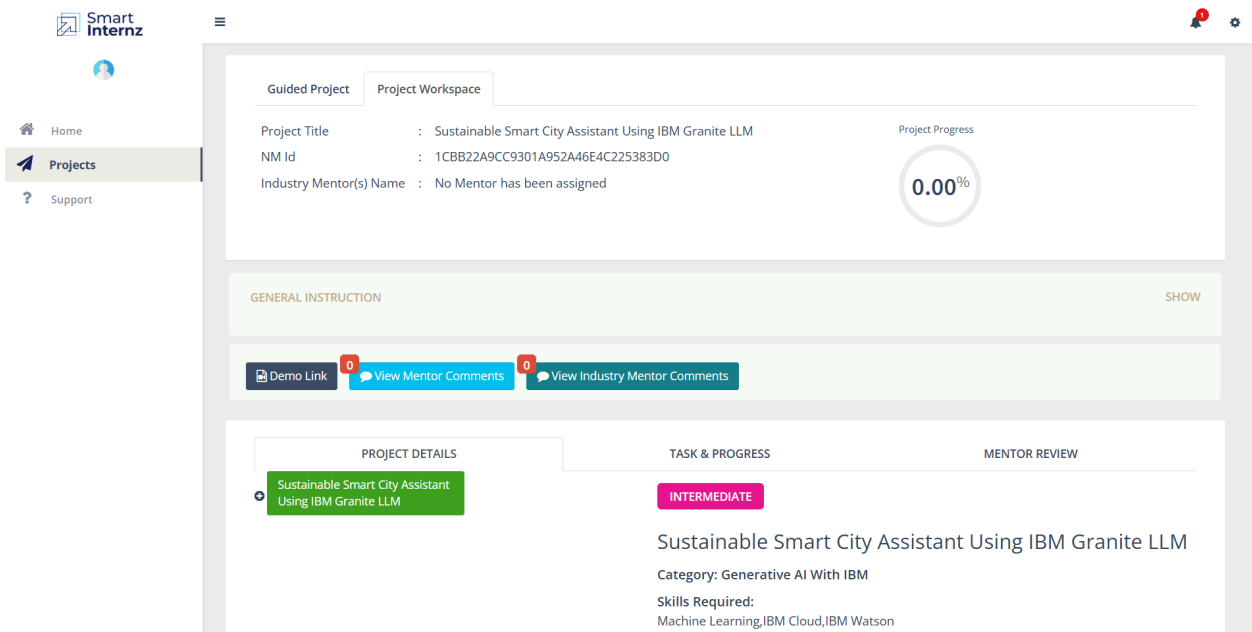


- Click on the “Go to workspace” section. Then you can find the detailed

explanation of Generative AI Project using IBM Watsonx API key.



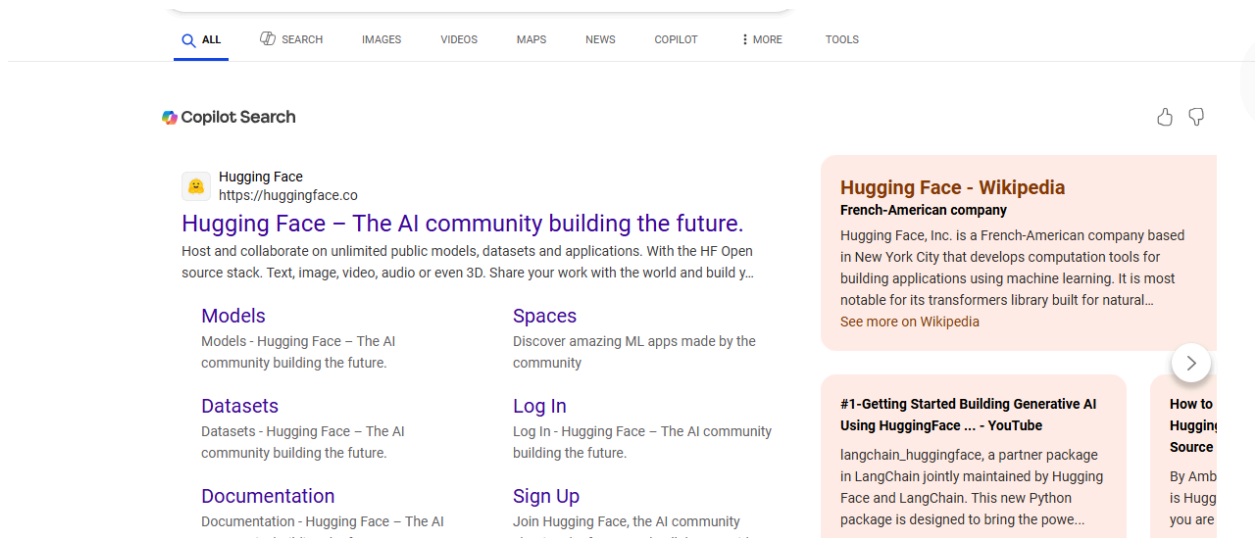
- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.



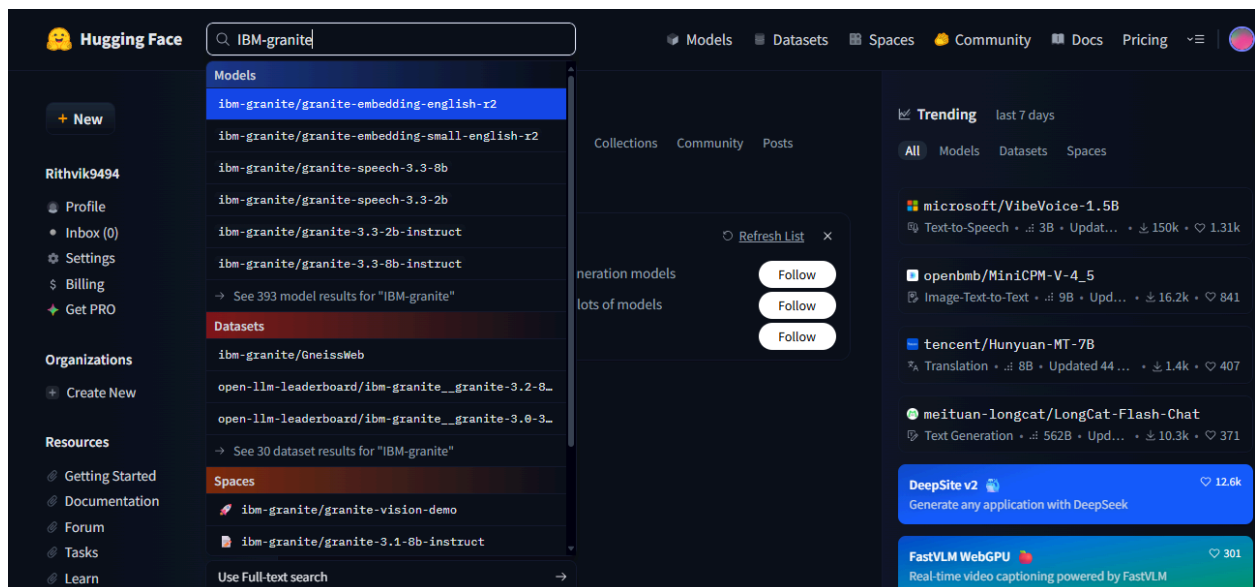
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project.

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.

The screenshot shows the Hugging Face interface for the model `ibm-granite/granite-3.2-2b-instruct`. The header includes the Hugging Face logo, a search bar, and navigation links for Models, Datasets, Spaces, Community, Docs, and Pricing. The model's name is displayed with a folder icon, a like count of 48, and a follow button for the user `IBM Granite` with 2,31k followers. Below this, a row of tags includes Text Generation, Transformers, Safetensors, granite, language, granite-3.2, conversational, arxiv:0000.00000, and License: apache-2.0. A secondary navigation bar shows Model card, Files and versions, and Community tabs. The main content area features a notification about a newer version, the model name **Granite-3.2-2B-Instruct**, and a detailed **Model Summary** describing it as a 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. To the right, a sidebar displays the download count (8,000), Safetensors details (2.53B params, BF16 tensor type), and Inference Providers information.

ibm-granite/granite-3.2-2b-instruct like 48 Follow IBM Granite 2,31k

Text Generation Transformers Safetensors granite language granite-3.2 conversational arxiv:0000.00000 License: apache-2.0

Model card Files and versions Community

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

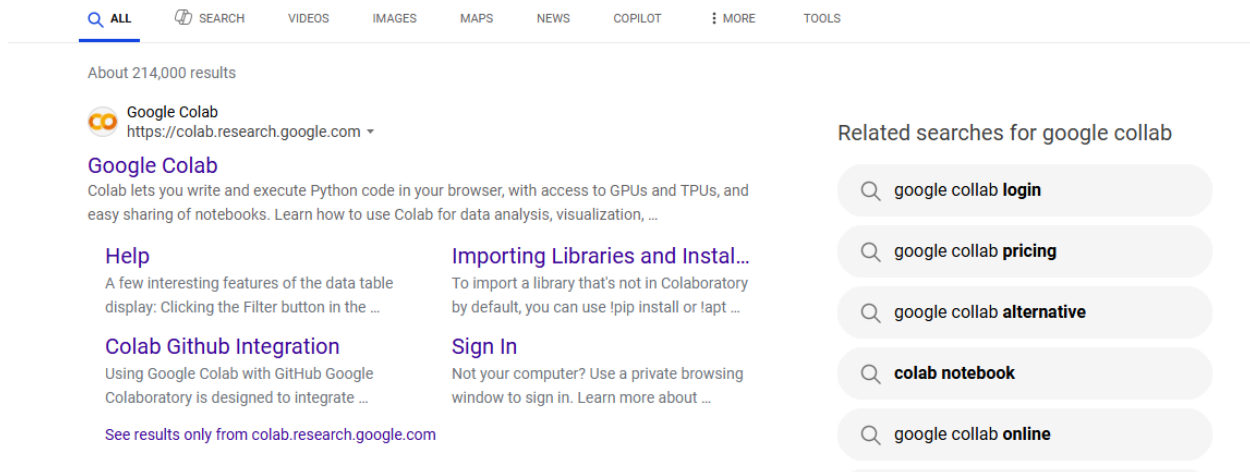
Text Generation

This model isn't deployed by any Inference Provider. Ask for provider support

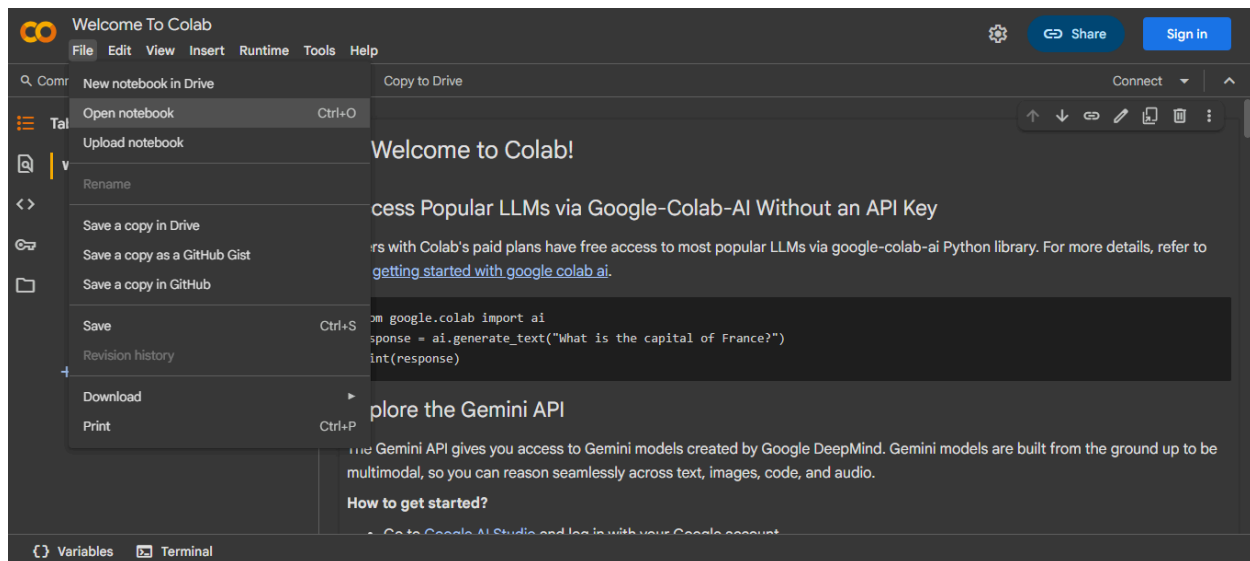
- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

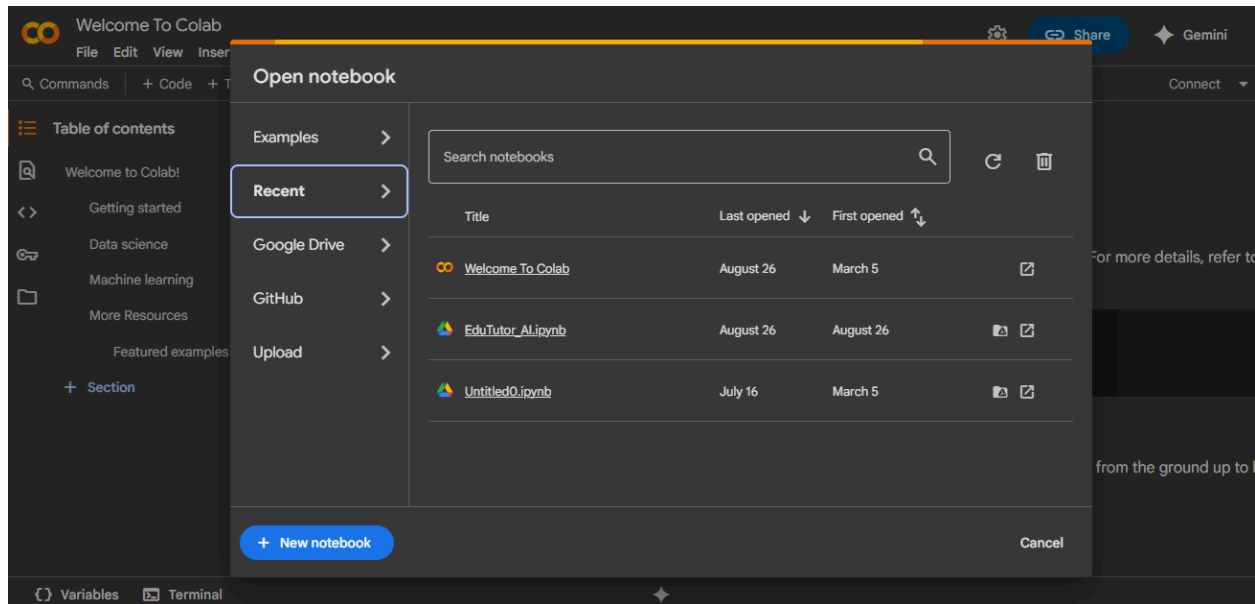
- Search for “Google collab” in any browser.



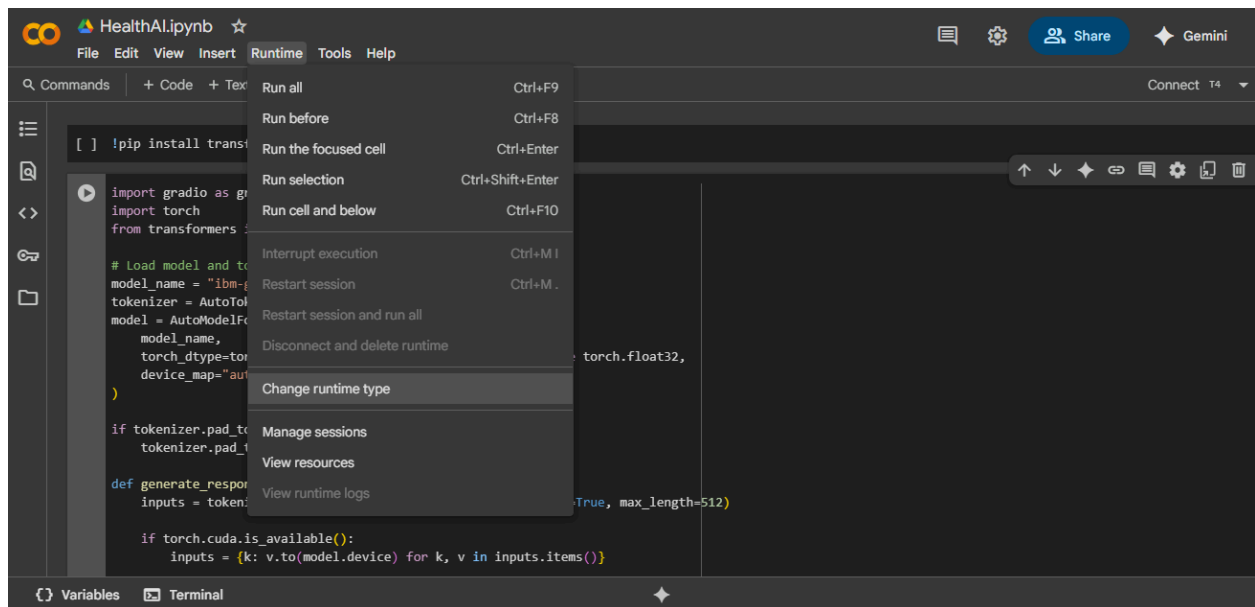
- Click on the first link ([Google Colab](https://colab.research.google.com)), then click on “Files” and then “Open Notebook”.



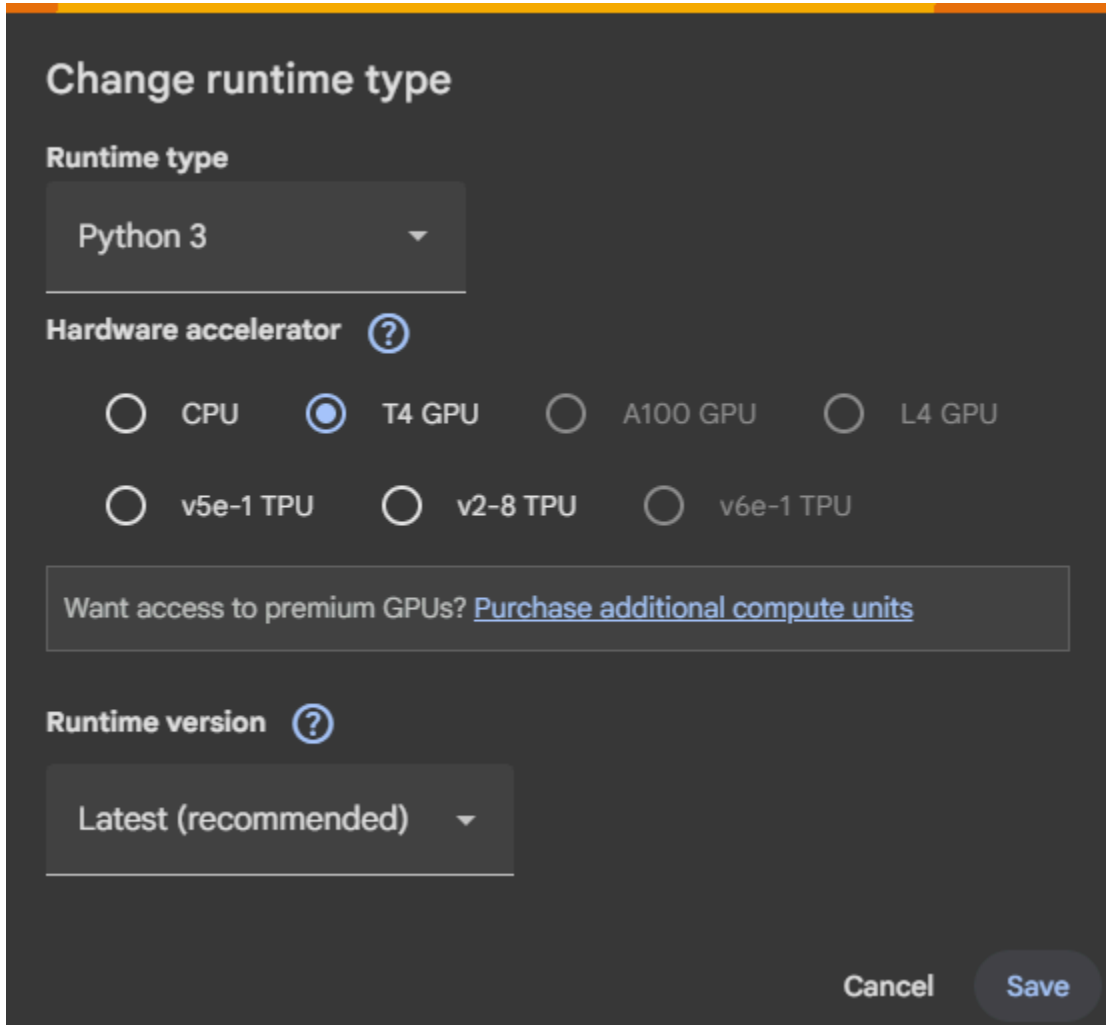
- Click on “New Notebook”



- 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”



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 PyPDF2 -q`”. To install the required libraries to run our application.

```
[ ] !pip install transformers torch gradio PyPDF2 -q
```



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

```

1
2 import gradio as gr
3 import torch
4 from transformers import AutoTokenizer, AutoModelForCausalLM
5 import PyPDF2
6 import io
7
8 # Load model and tokenizer
9 model_name = "ibm-granite/granite-3.2-2b-instruct"
10 tokenizer = AutoTokenizer.from_pretrained(model_name)
11 model = AutoModelForCausalLM.from_pretrained(
12     model_name,
13     torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
14     device_map="auto" if torch.cuda.is_available() else None
15 )
16
17 if tokenizer.pad_token is None:
18     tokenizer.pad_token = tokenizer.eos_token
19
20 def generate_response(prompt, max_length=1024):
21     inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
22
23     if torch.cuda.is_available():
24         inputs = {k: v.to(model.device) for k, v in inputs.items()}
25
26     with torch.no_grad():
27         outputs = model.generate(
28             *inputs,
29             max_length=max_length,
30             temperature=0.7,
31             do_sample=True,
32             pad_token_id=tokenizer.eos_token_id
33         )
34
35     response = tokenizer.decode(outputs[0], skip_special_tokens=True)
36     response = response.replace(prompt, "").strip()
37     return response

```

```

38
39 def extract_text_from_pdf(pdf_file):
40     if pdf_file is None:
41         return ""
42
43     try:
44         pdf_reader = PyPDF2.PdfReader(pdf_file)
45         text = ""
46         for page in pdf_reader.pages:
47             text += page.extract_text() + "\n"
48         return text
49     except Exception as e:
50         return f"Error reading PDF: {str(e)}"
51
52 def eco_tips_generator(problem_keywords):
53     prompt = f"Generate practical and actionable eco-friendly tips for sustainable living related to: {problem_keywords}. Provide specific solutions and suggestions:"
54     return generate_response(prompt, max_length=1000)
55
56 def policy_summarization(pdf_file, policy_text):
57     # Get text from PDF or direct input
58     if pdf_file is not None:
59         content = extract_text_from_pdf(pdf_file)
60         summary_prompt = f"Summarize the following policy document and extract the most important points, key provisions, and implications:\n\n{content}"
61     else:
62         summary_prompt = f"Summarize the following policy document and extract the most important points, key provisions, and implications:\n\n{policy_text}"
63
64     return generate_response(summary_prompt, max_length=1200)
65
66 # Create Gradio interface
67 with gr.Blocks() as app:
68     gr.Markdown("# Eco Assistant & Policy Analyzer")
69
70     with gr.Tabs():
71         with gr.TabItem("Eco Tips Generator"):
72             with gr.Row():
73                 with gr.Column():
74                     keywords_input = gr.Textbox(
75                         label="Environmental Problem/Keywords",
76                         placeholder="e.g., plastic, solar, water waste, energy saving...",
77                         lines=3
78                     )
79                     generate_tips_btn = gr.Button("Generate Eco Tips")
80
81                 with gr.Column():
82                     tips_output = gr.Textbox(label="Sustainable Living Tips", lines=15)
83
84             generate_tips_btn.click(eco_tips_generator, inputs=keywords_input, outputs=tips_output)
85
86         with gr.TabItem("Policy Summarization"):
87             with gr.Row():
88                 with gr.Column():
89                     pdf_upload = gr.File(label="Upload Policy PDF", file_types=[".pdf"])
90                     policy_text_input = gr.Textbox(
91                         label="Or paste policy text here",
92                         placeholder="Paste policy document text...",
93                         lines=5
94                     )
95                     summarize_btn = gr.Button("Summarize Policy")
96
97                 with gr.Column():
98                     summary_output = gr.Textbox(label="Policy Summary & Key Points", lines=20)
99
100             summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summary_output)
101
102 app.launch(share=True)
103
104

```

- You can find the code here in this link: [Sustainable Smart City Assistant](#)

OUTPUT:

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

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

* Running on public URL: <https://5475fe6c096b7ff650.gradio.live>

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

Eco Assistant & Policy Analyzer

Eco Tips Generator

Policy Summarization

Environmental Problem/Keywords

e.g., plastic, solar, water waste, energy saving...

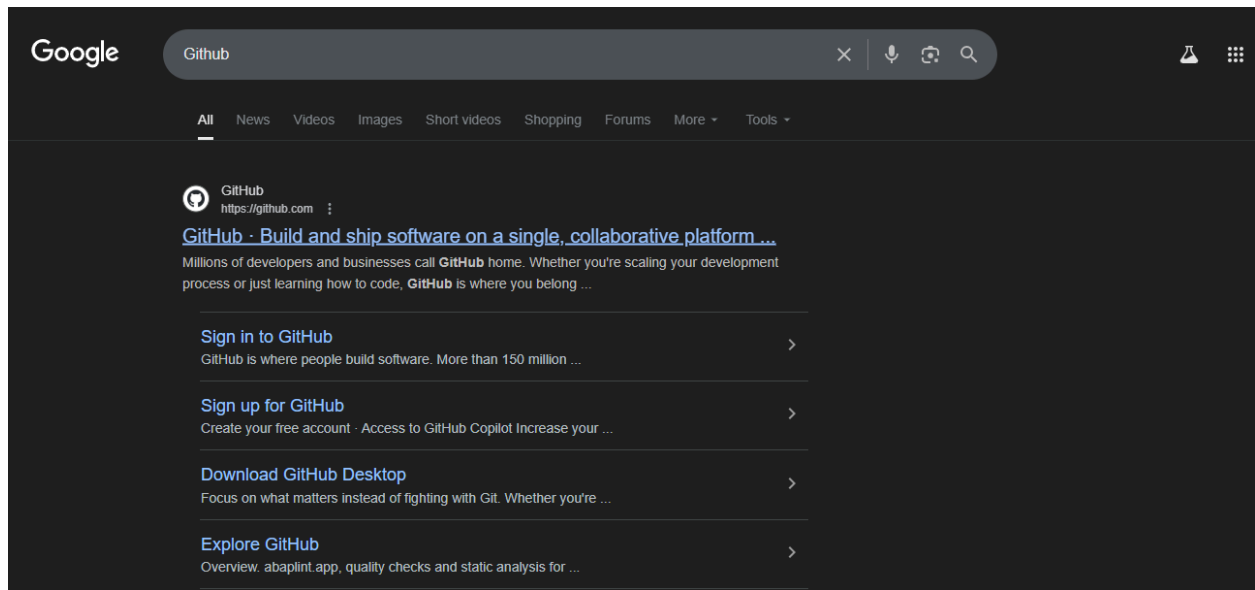
Generate Eco Tips

Sustainable Living Tips

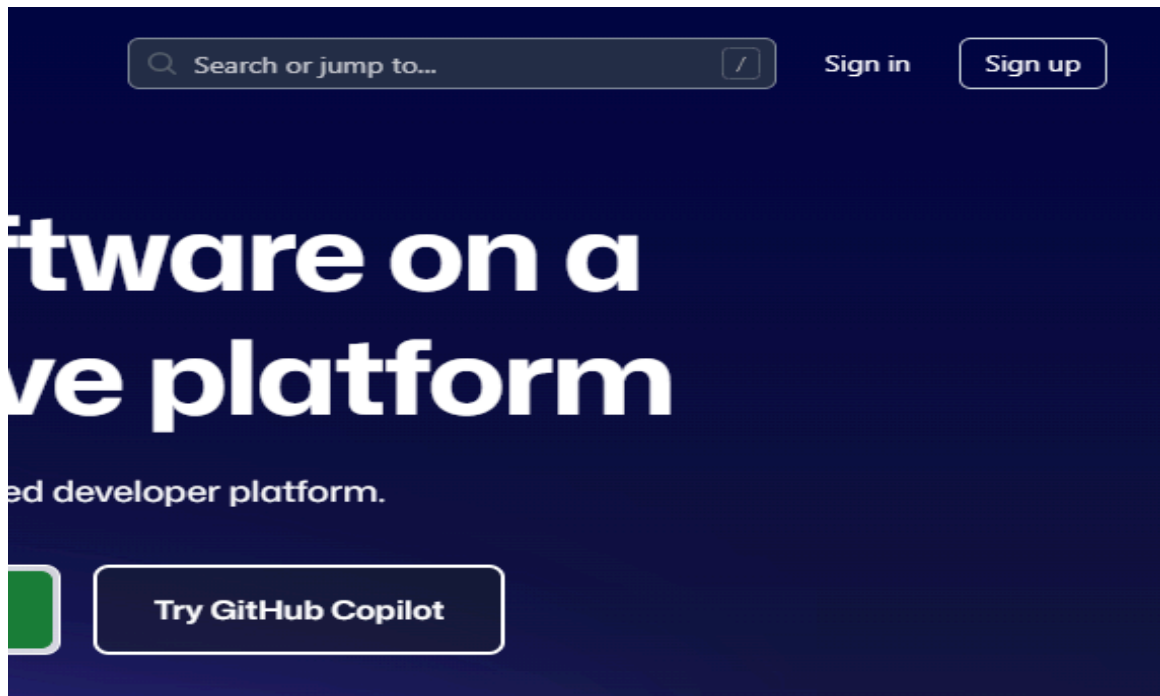
- You can View the Application is the running in the other tab

Activity-4: Upload Your Project in GitHub.

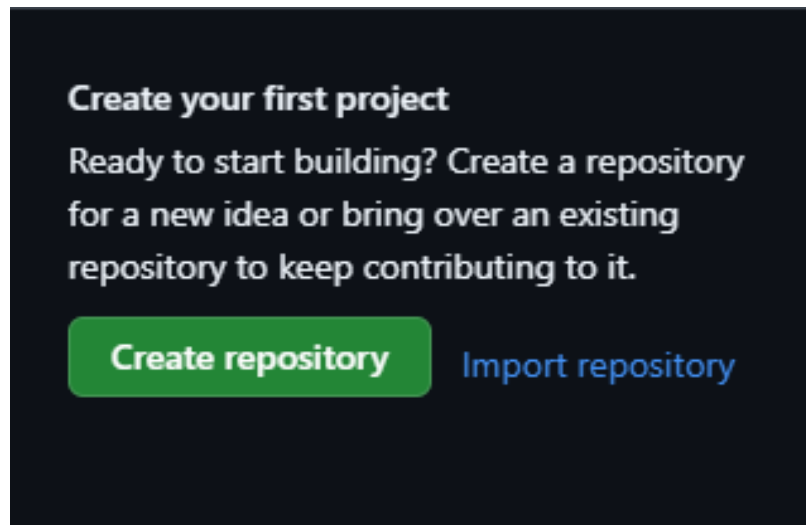
- Search for “GitHub” in any browser, then click on the first link ([GitHub](#)).



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

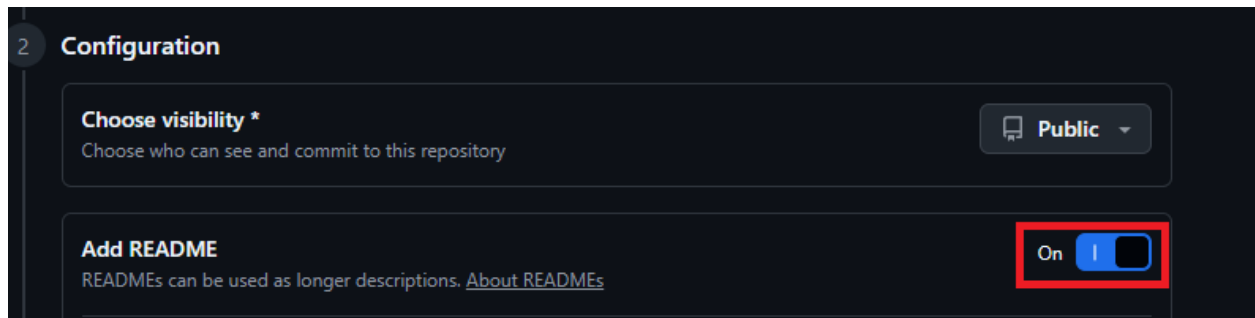


- Click on “Create repository”.

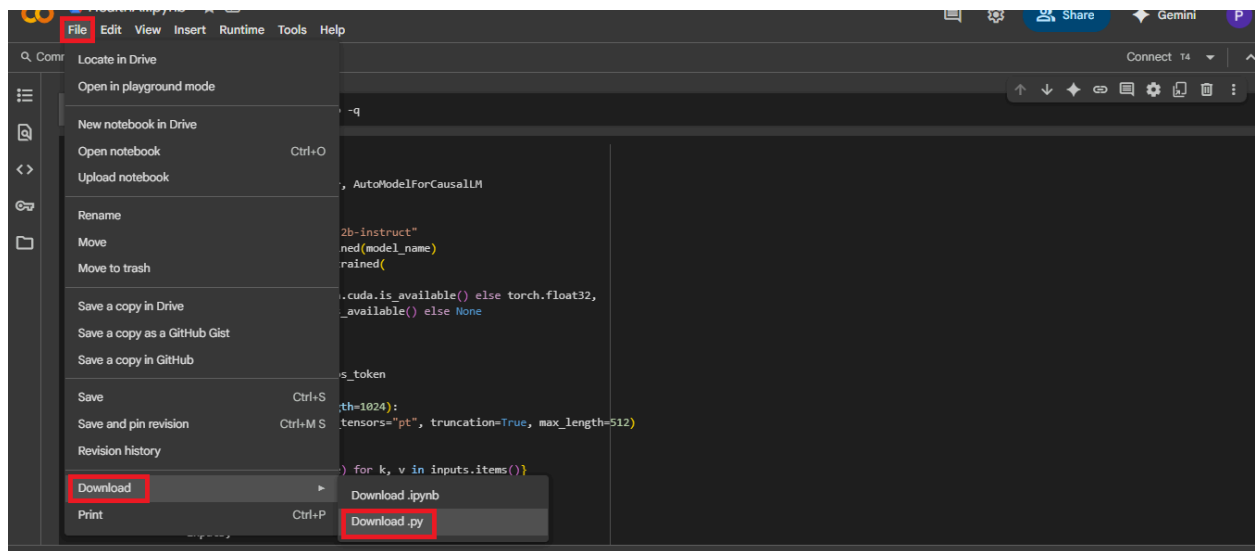


- In “General” Name your repo. (Here I have given “IBM-Project” as my repo name and it is available)

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

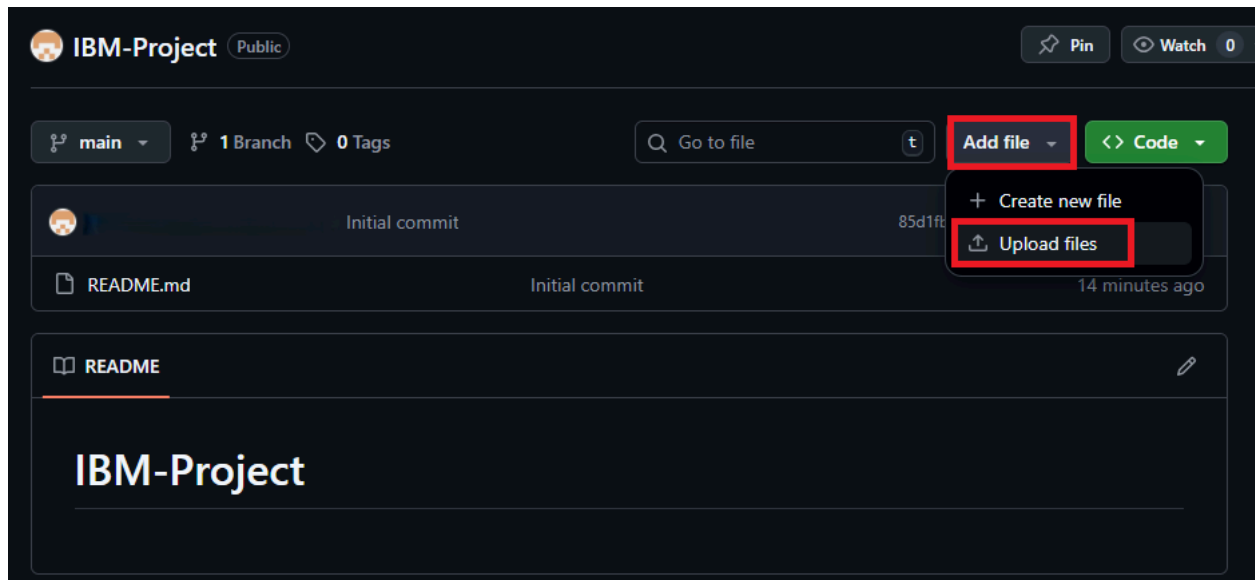


- Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.

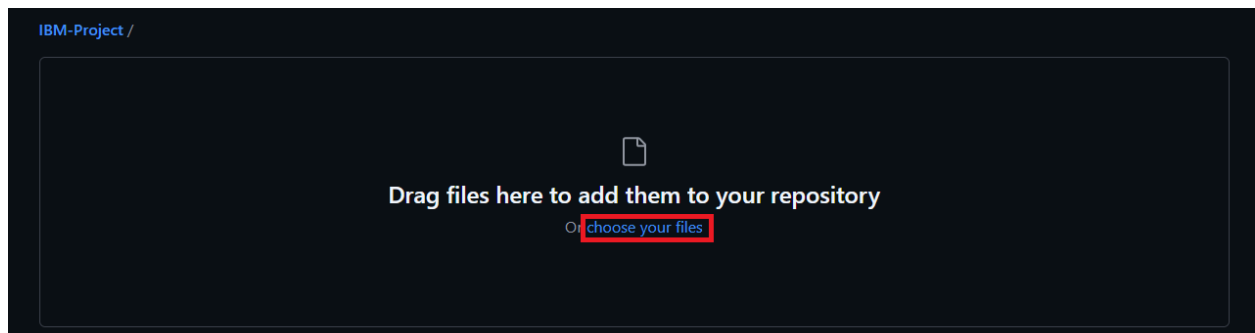


- Then your repository is created, then Click on “Add file” Option. Then Click

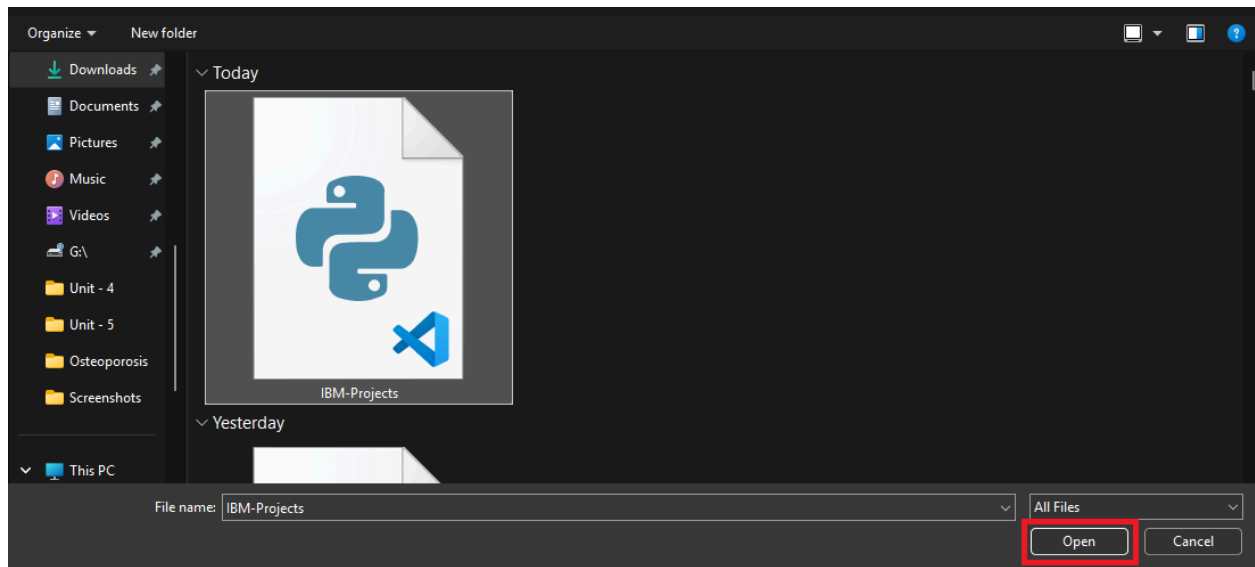
“Upload files” to upload your files.



- Click on “choose your files”.



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



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

