TEAM MEMBERS:

Mythili.K S

Manjusri.A

Mathu sree.T

Narmadha.P

CONTENTS

SmartSDLC – AI-Enhanced Software Development Lifecycle

Project Description

The SmartSDLC project is an AI-enhanced platform designed to automate and streamline the Software Development Lifecycle (SDLC) using advanced technologies like IBM Watsonx, FastAPI, LangChain, and Streamlit. It integrates generative AI to handle key SDLC phases, including requirement analysis, code generation, test case creation, bug fixing, and documentation. The platform features a user-friendly interface that allows users to upload PDFs, generate structured requirements, and transform natural language prompts into functional code. It also includes an AI-powered chatbot for real-time assistance and support. The backend, built with FastAPI, efficiently processes API requests, while the frontend, developed using Streamlit, offers a visually appealing and interactive dashboard. The system is modular, scalable, and secure, featuring a robust authentication mechanism and seamless integration between AI models and user inputs.

Objectives:

- Automatically generate and run test cases.
- Identify defects in code through AI-powered code reviews.
- Predict possible bugs before deployment.
- Ensure continuous monitoring after release to catch issues quickly.
- Adapt quickly to changing requirements
- Ensure compliance and security checks automatically
- Auto-generate documentation and test cases

Abstract:

The Smart Software Development Life Cycle (Smart SDLC) represents an evolution of traditional software development methodologies by integrating intelligent technologies such as AI, machine learning, automation, and data analytics into each phase of the SDLC. This modern approach enhances efficiency, decision-making, and adaptability by enabling predictive planning, automated testing, real-time monitoring, and continuous integration and delivery. Smart

SDLC aims to reduce time-to-market, improve software quality, and increase responsiveness to changing requirements, making it an essential paradigm in the age of digital transformation.

Future Enhancement:

- AI tools can **listen to what users want** and turn that into proper software requirements.
- They can **spot missing or confusing parts** in the early stages.

🎨 2. **Better Software Design Help**

- AI can help **create system designs automatically**.
- It can suggest the **best structure** based on what worked in past projects.

💻 3. **AI Writing Code**

- Smart tools can **write parts of the code for developers**.
- They also **check the code quality** as it's written.

🧪 4. **Automatic Testing**

- AI can **create test cases by itself**.
- If something changes in the software, the tests **adjust automatically** without manual updates. 🚀 5. **Faster and Smarter Deployment**
- The software can **launch automatically** when it's ready.
- If something goes wrong, it can **roll back changes** on its own.

📊 5. **Live Monitoring and Feedback**

- AI watches the software in real time to **find issues quickly**.

- It learns from how users use the software and **suggests improvements**.

🔧 6. **Easy Maintenance**

- AI can **predict bugs** before they happen.
- Virtual assistants can help developers **find and fix issues faster**

🔐 7. **Built-in Smart Security**

- The system checks for **security risks constantly**.
- It also **suggests fixes** before the problems get serious.

🔁 8. **Reusing Old Work**

- The system remembers old solutions so teams don't have to **start from scratch** every time.
- It **recommends ready-made parts** to use again.

🤝 9. **Smarter Teamwork**

- AI can help assign tasks to the **right team members**.
- It can suggest which **development method** (Agile, Waterfall, etc.) fits the project best.

Project workflow:

Activity-1: Exploring Naan Mudhalvan Smart Interz Portal.

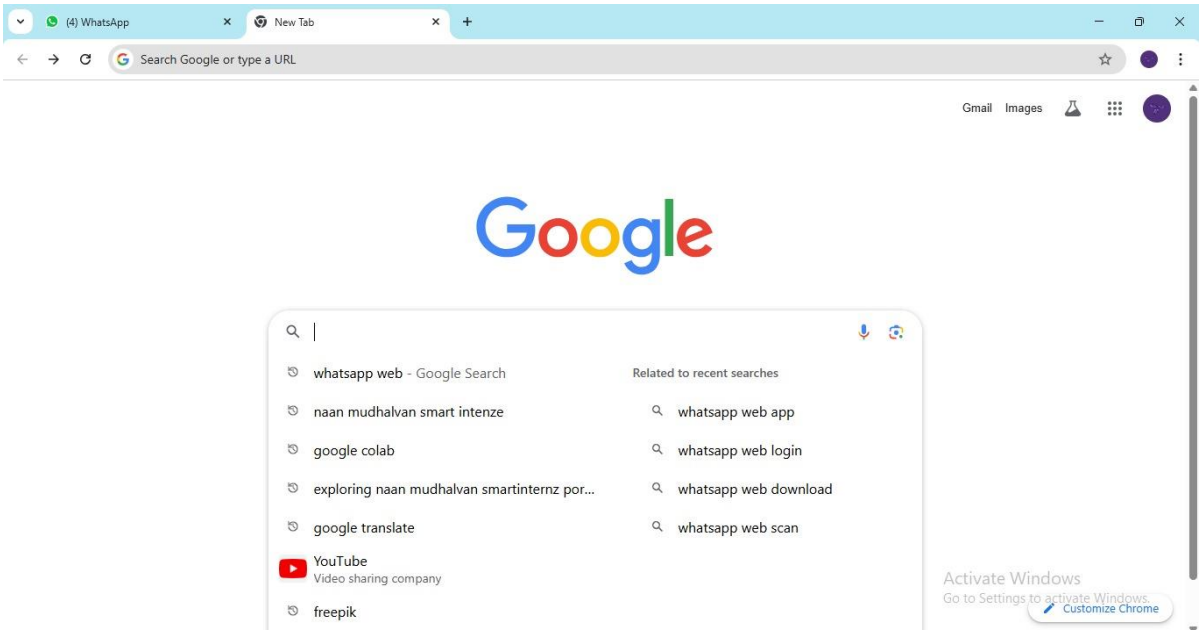
Activity-2: Choosing a IBM Granite Model From Hugging Face.

Activity-3: Running Application in Google Colab.

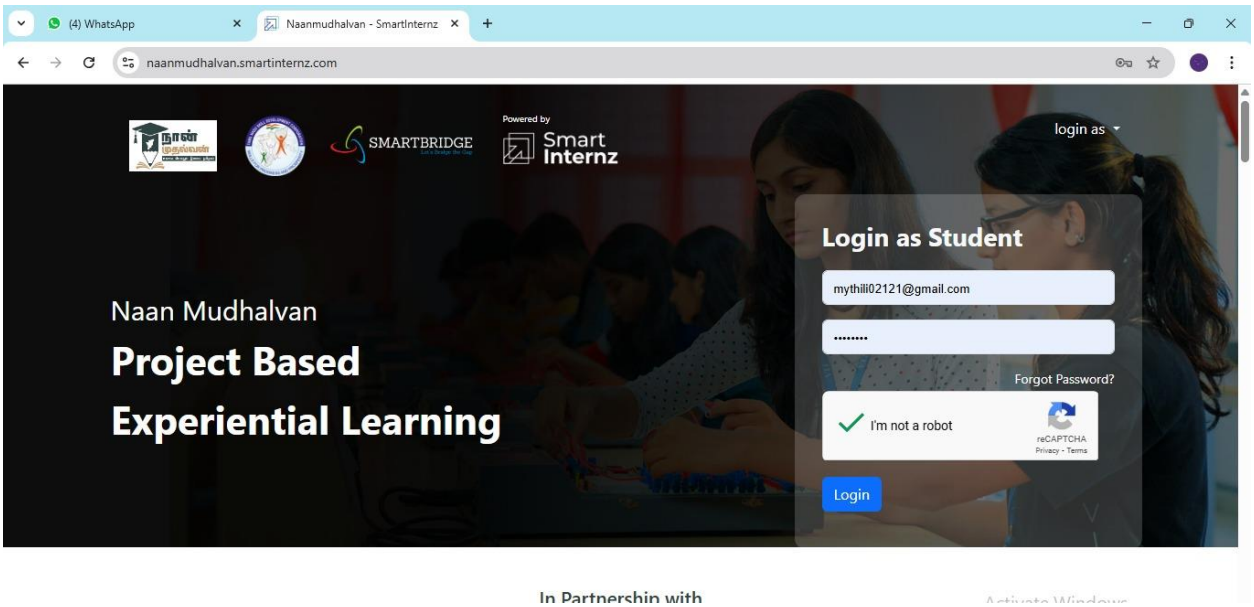
Activity-4: Upload your Project in GitHub

Activity-1: Exploring Naan Mudhalvan Smart Interz Portal.

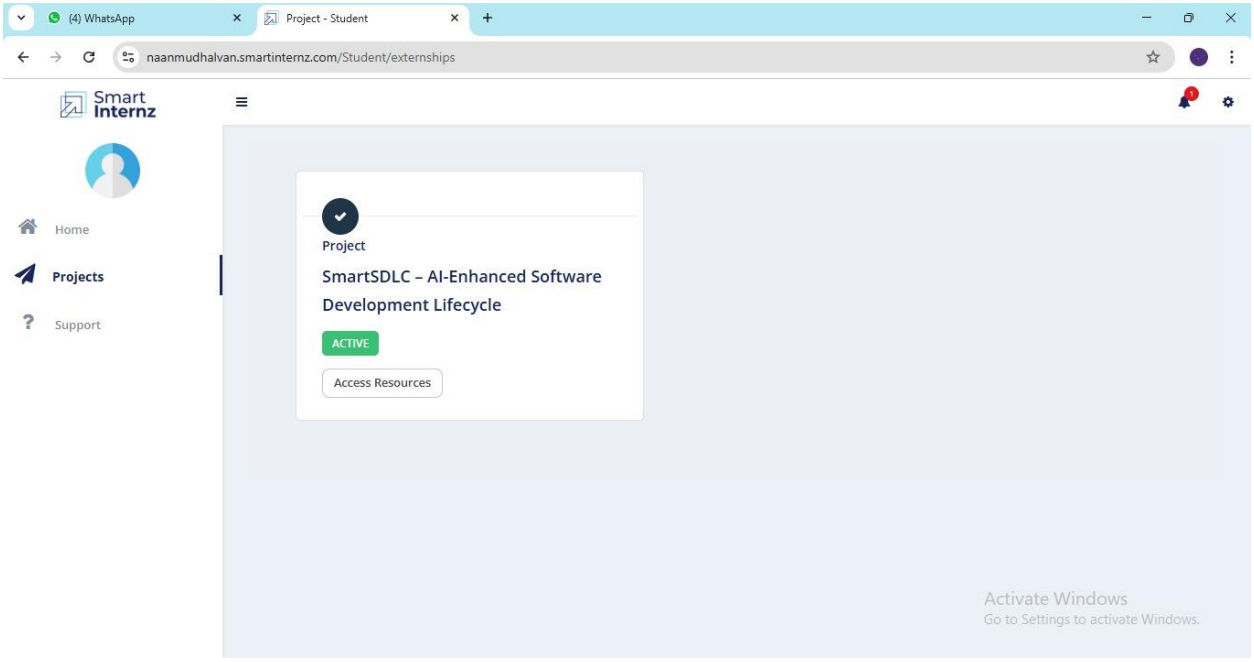
- Search for “Naan Mudhalvan Smart Interz” Portal in any Browser



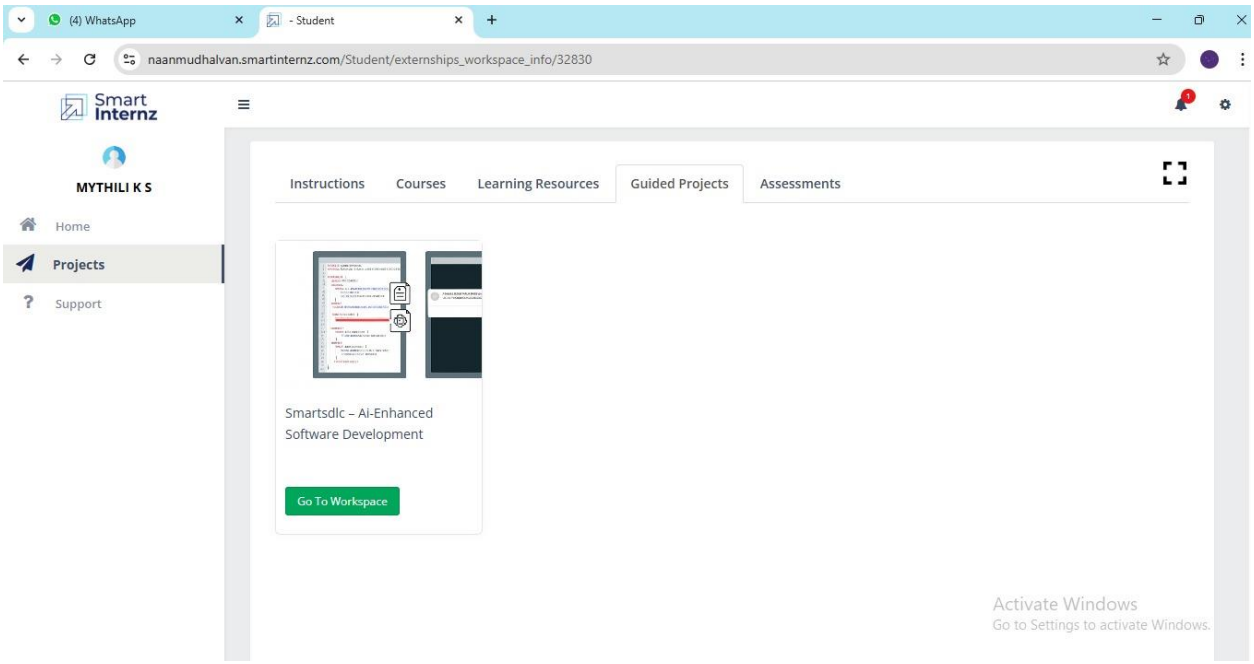
- Then Click on the first link. (Naanmudhalvan Smartinternz) Then login with your details



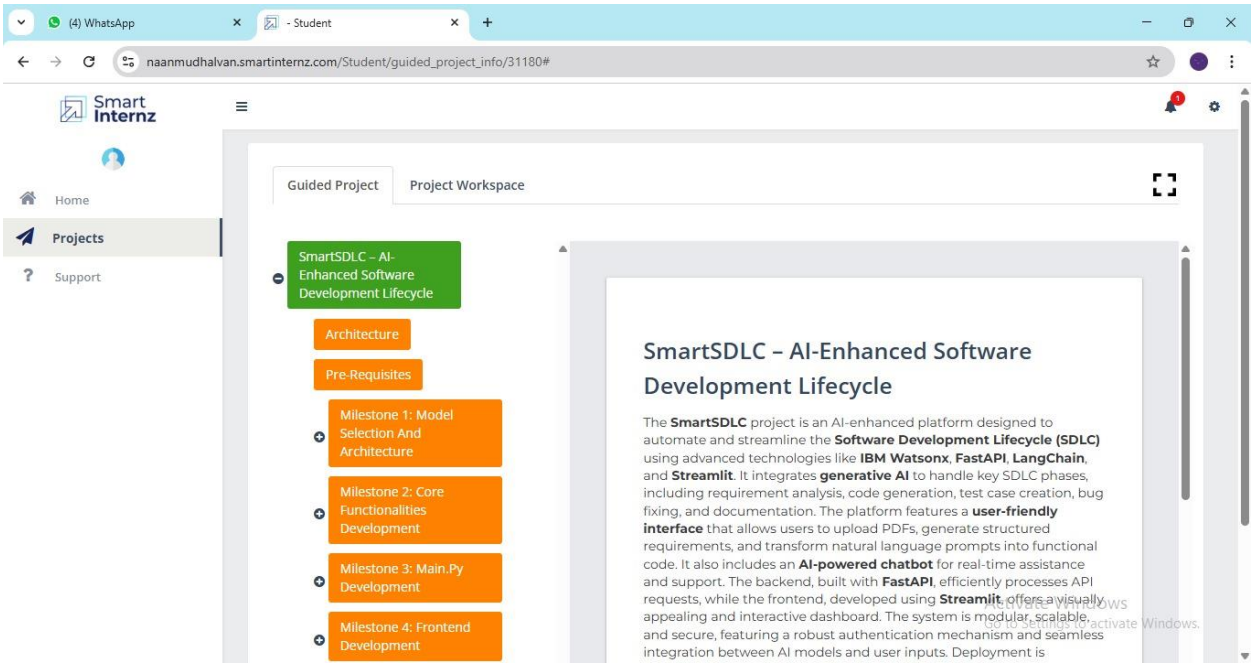
- Then you will be redirected to your account then click on “Projects” Section. There you can see which project you have enrolled in here it is “SmartSDLC – AI-Enhanced Software Development Lifecycle”.



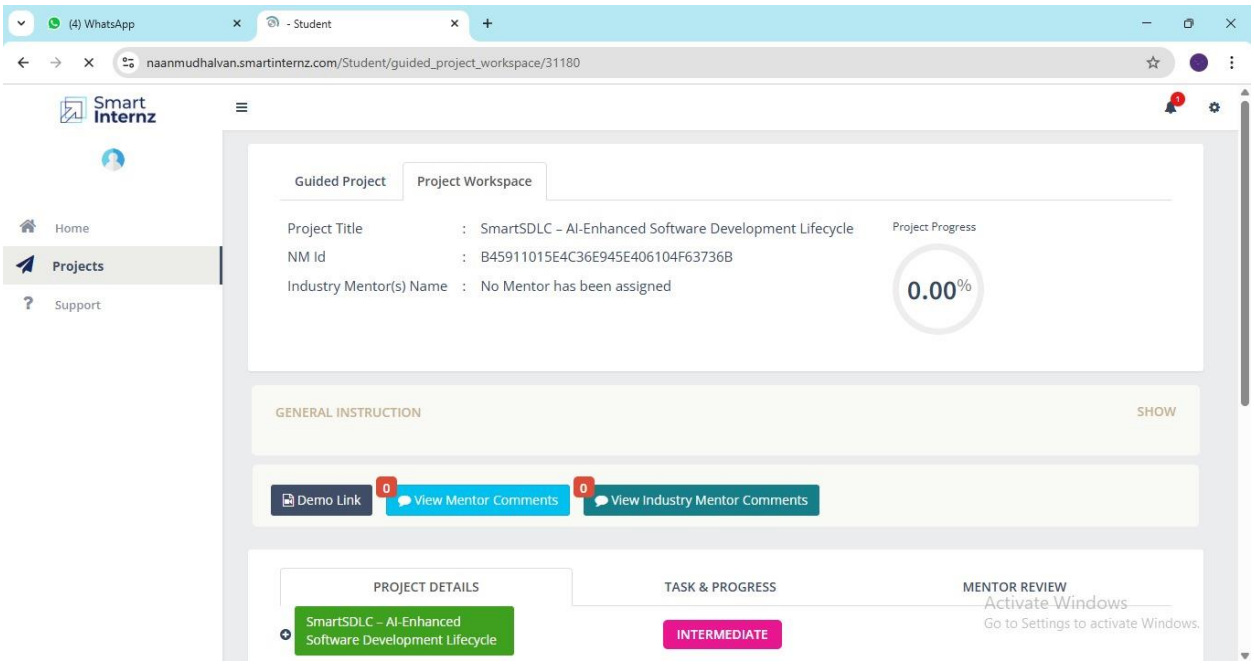
- Then click on “Access Resources” and go to the “Guided Project” Section



- Click on the “Go to workspace” section. Then you can find the detailed 3 explanation of Generative AI Project using IBM WatsonX API key



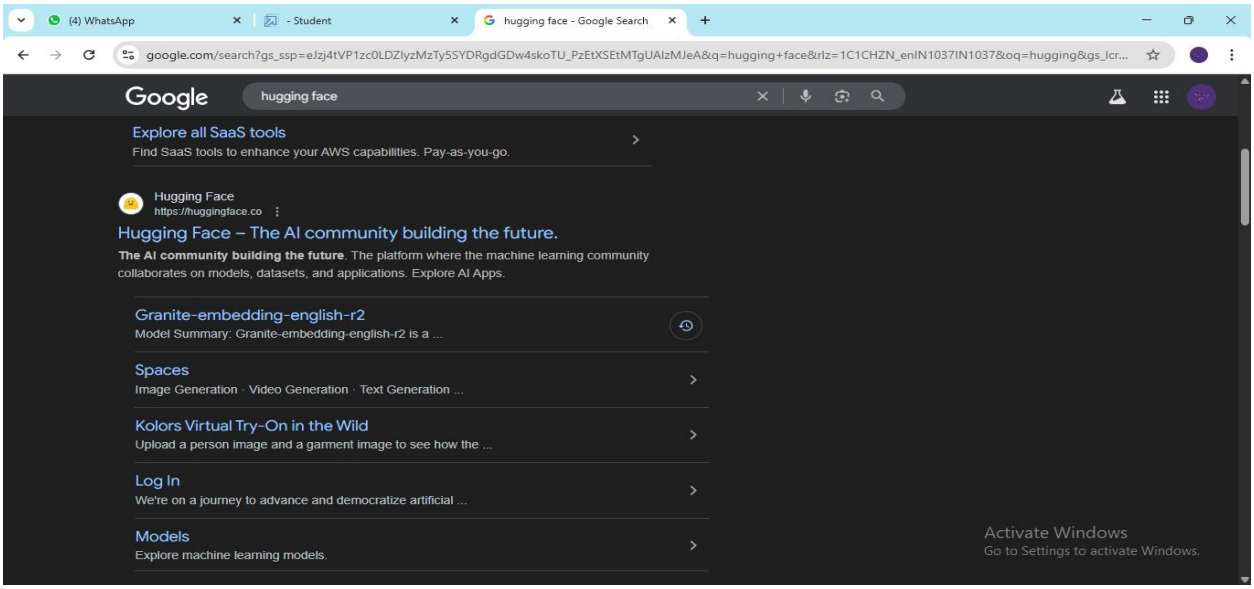
- Click on “Project Workspace”, there you can find your project progress and Place to upload “Demo link”.



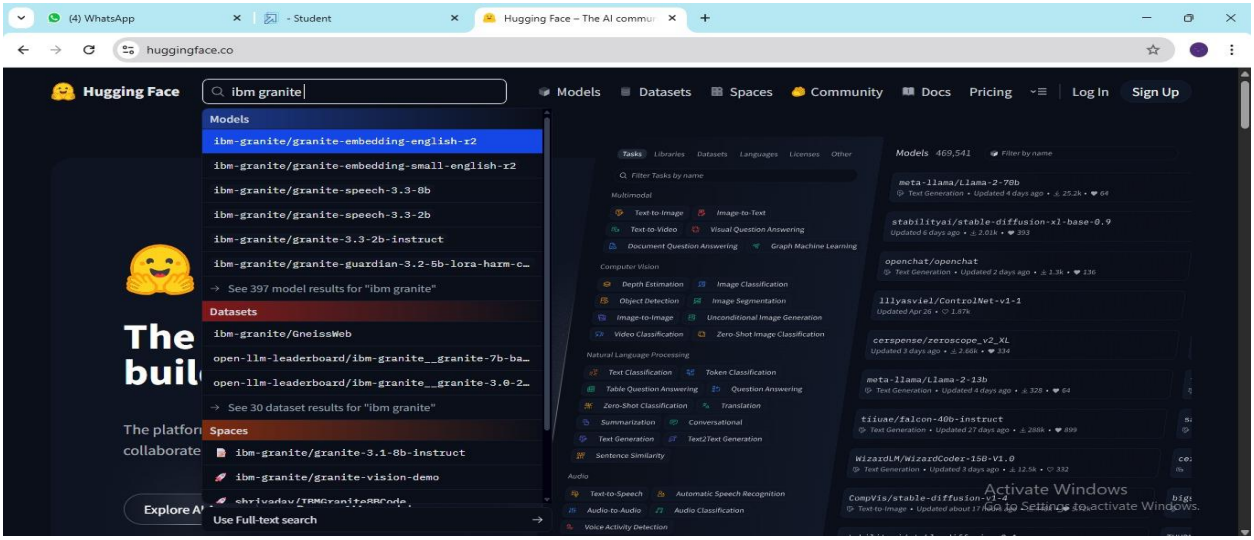
- Now we have gone through portal understanding, now lets find a IBM granite model from hugging face to integrate in our project

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

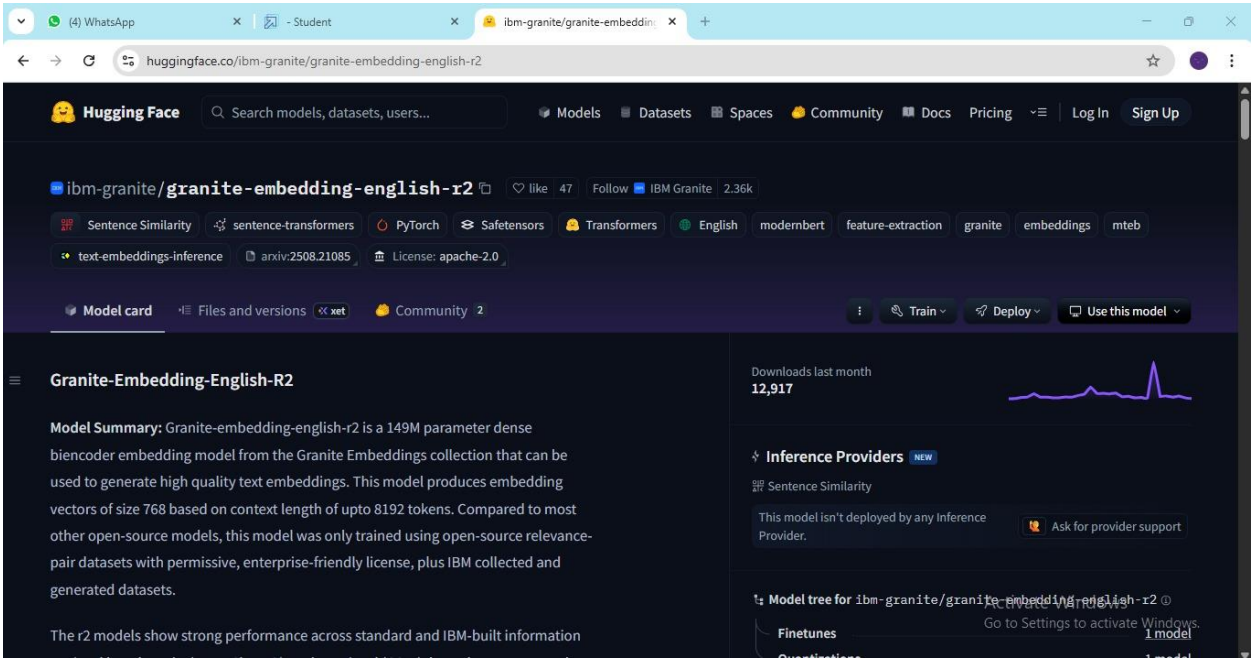
- Search for “Hugging face” in any browser.



- 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



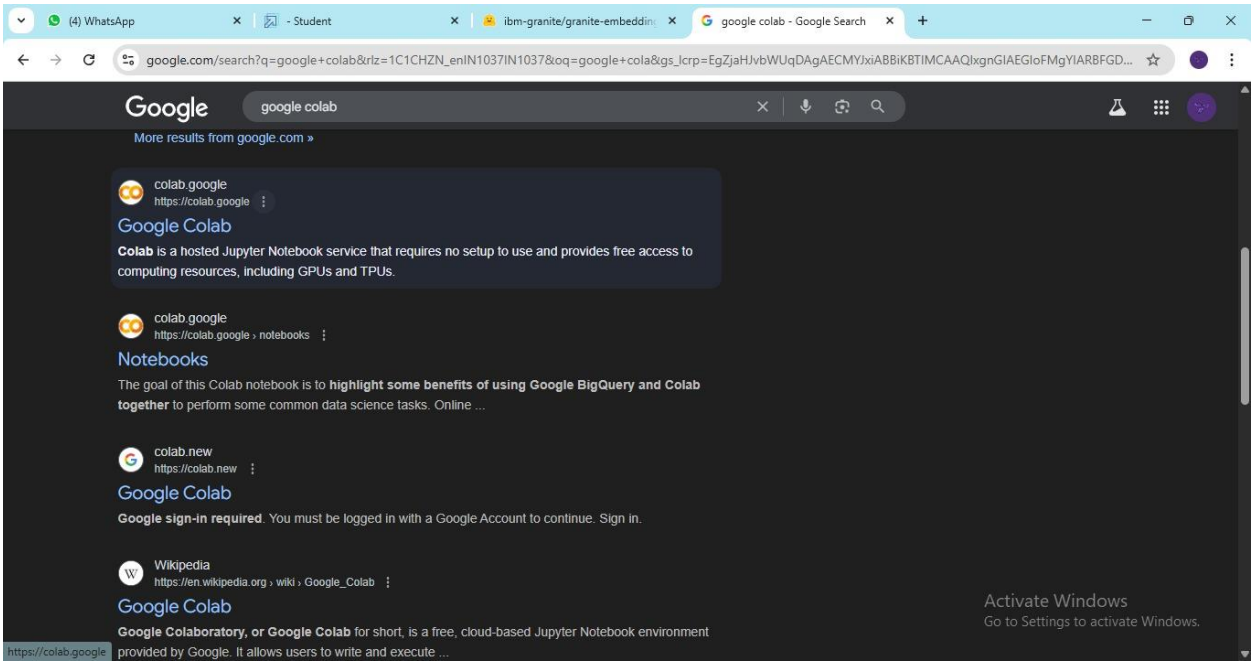
- Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.



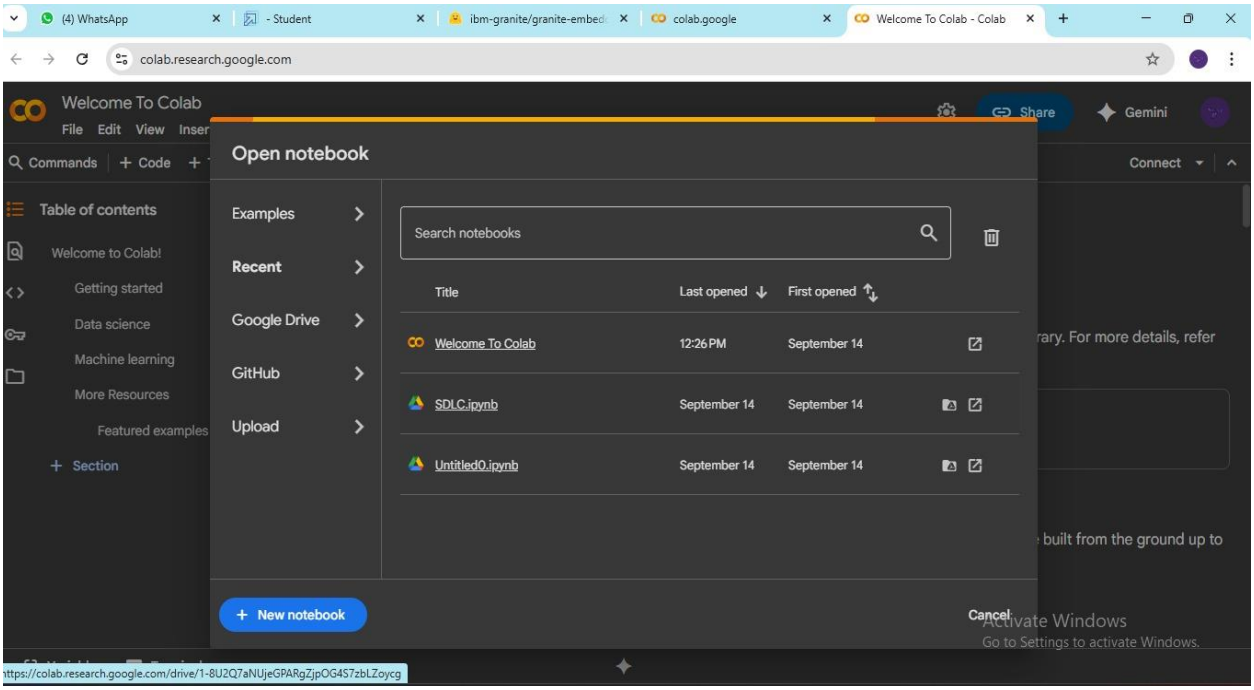
- Now we will start building our project in Google collab.

Activity-3: Running Application in Google Collab.

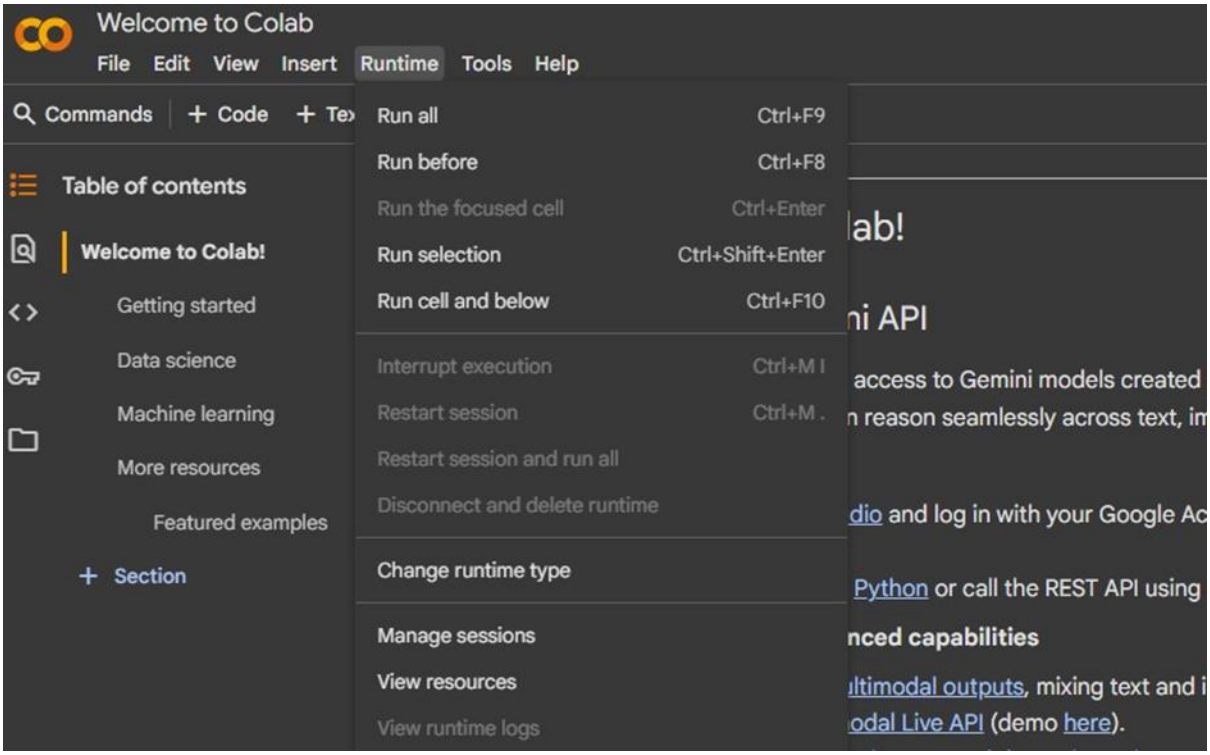
- Search for “Google collab” in any browser.



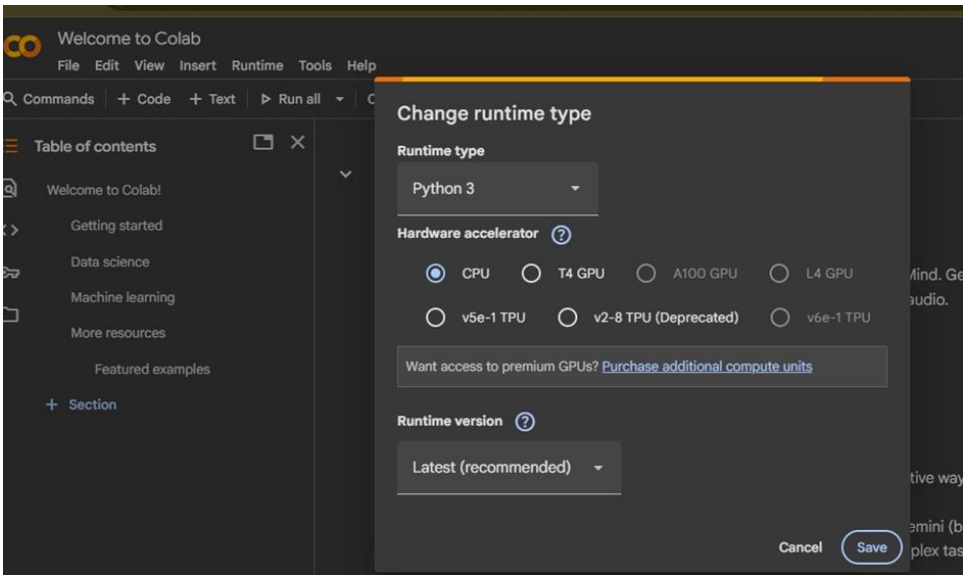
- Click on the first link (Google Colab), then click on “Files” and then “Open Notebook
- Click on “New Notebook”



➢ 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”.



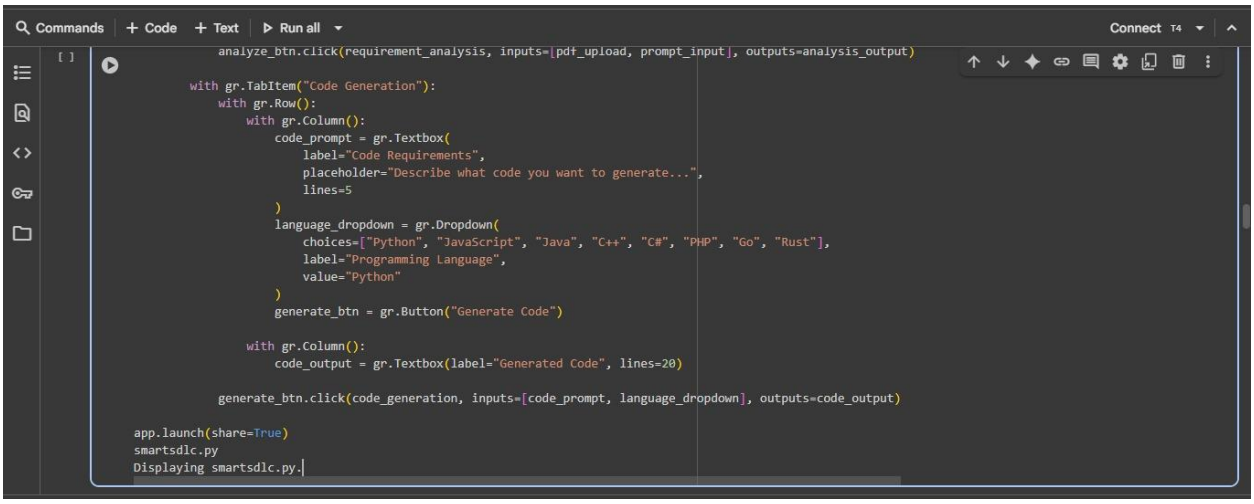
- Then run this command in the first cell "!pip install transformers torch gradio PyPDF2 -q". To install the required libraries to run our application.

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

```
    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
        )

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

def extract_text_from_pdf(pdf_file):
    if pdf_file is None:
        return ""

    try:
        pdf_reader = PyPDF2.PdfReader(pdf_file)
        text = ""
        for page in pdf_reader.pages:
            text += page.extract_text() + "\n"
```

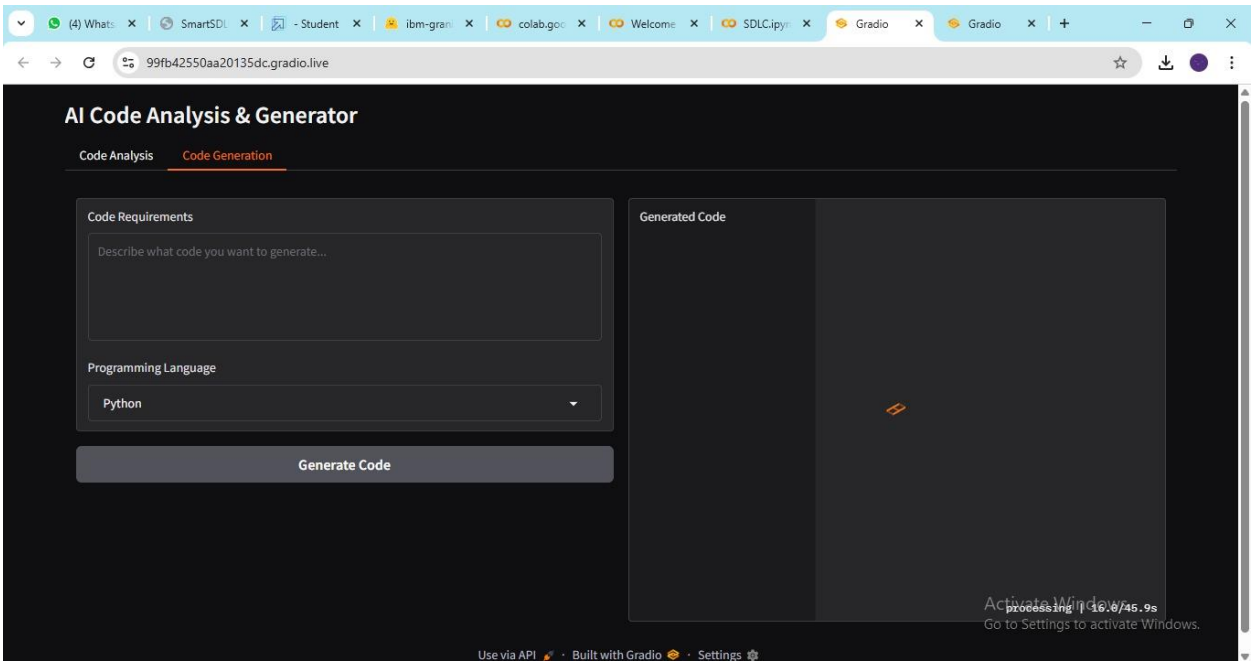
```
        try:
            pdf_reader = PyPDF2.PdfReader(pdf_file)
            text = ""
            for page in pdf_reader.pages:
                text += page.extract_text() + "\n"
            return text
        except Exception as e:
            return f"Error reading PDF: {str(e)}"

    def requirement_analysis(pdf_file, prompt_text):
        # Get text from PDF or prompt
        if pdf_file is not None:
            content = extract_text_from_pdf(pdf_file)
            analysis_prompt = f"Analyze the following document and extract key software requirements. Organize them into functional requirements, non-functional
        else:
            analysis_prompt = f"Analyze the following requirements and organize them into functional requirements, non-functional requirements, and technical sp

        return generate_response(analysis_prompt, max_length=1200)

    def code_generation(prompt, language):
        code_prompt = f"Generate {language} code for the following requirement:\n\n{prompt}\n\nCode:"
        return generate_response(code_prompt, max_length=1200)

    # Create Gradio interface
    with gr.Blocks() as app:
        gr.Markdown("# AI Code Analysis & Generator")
```

```
    with gr.Tabs():
        with gr.TabItem("Code Analysis"):
            with gr.Row():
                with gr.Column():
                    pdf_upload = gr.File(label="Upload PDF", file_types=[".pdf"])
                    prompt_input = gr.Textbox(
                        label="Or write requirements here",
                        placeholder="Describe your software requirements...",
                        lines=5
                    )
                    analyze_btn = gr.Button("Analyze")

                with gr.Column():
                    analysis_output = gr.Textbox(label="Requirements Analysis", lines=20)

            analyze_btn.click(requirement_analysis, inputs=[pdf_upload, prompt_input], outputs=analysis_output)

        with gr.TabItem("Code Generation"):
            with gr.Row():
                with gr.Column():
                    code_prompt = gr.Textbox(
                        label="Code Requirements",
                        placeholder="Describe what code you want to generate...",
                        lines=5
                    )
```



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

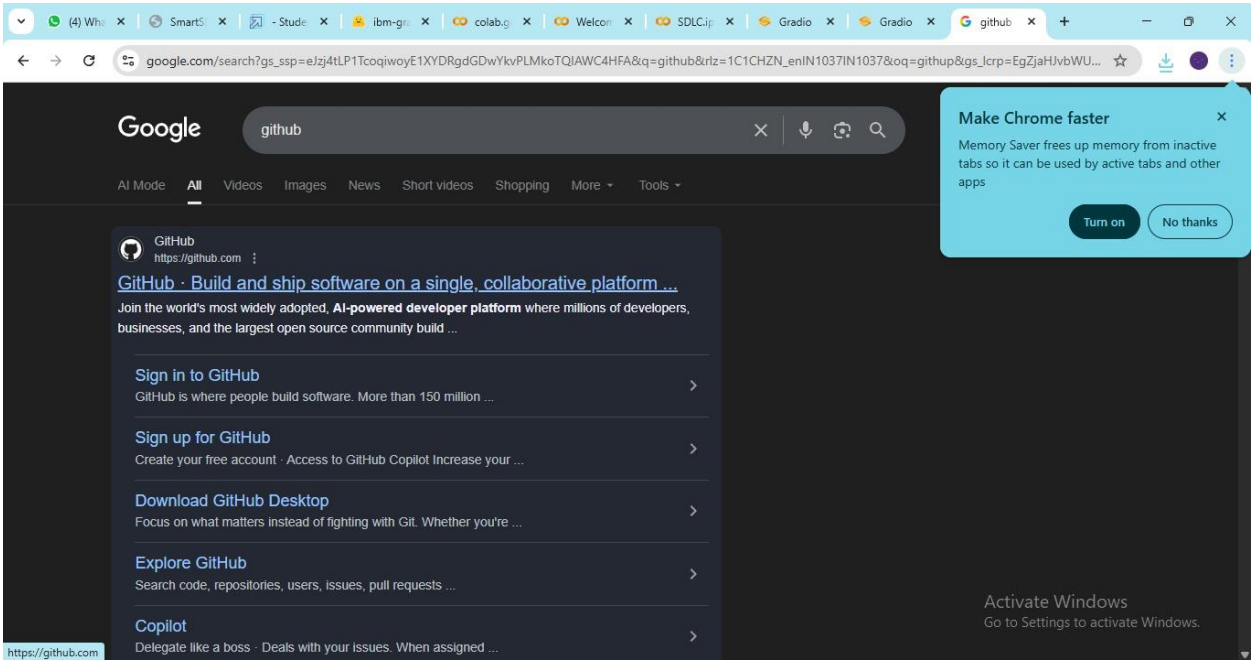
- Now you can see our model is being Downloaded and application is running
- Click on the URl to open the Gradio Application click on the link: https://f4888bb452623b8fc4.gradio.live



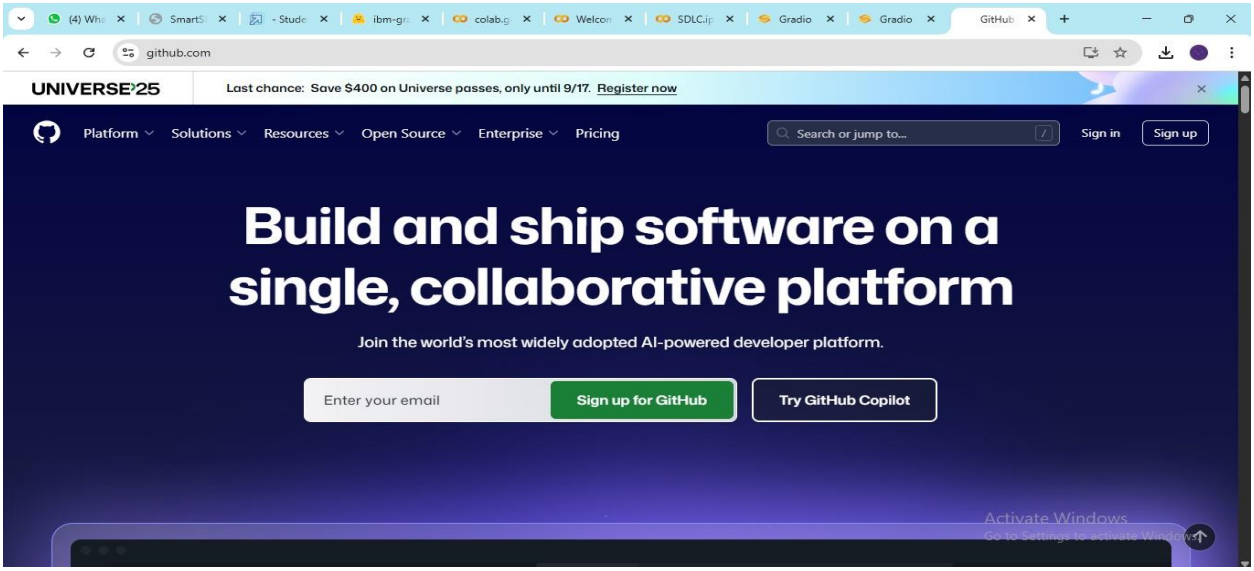
- You can View the Application is the running in the other tab

Activity-4: Upload Your Project in GitHub.

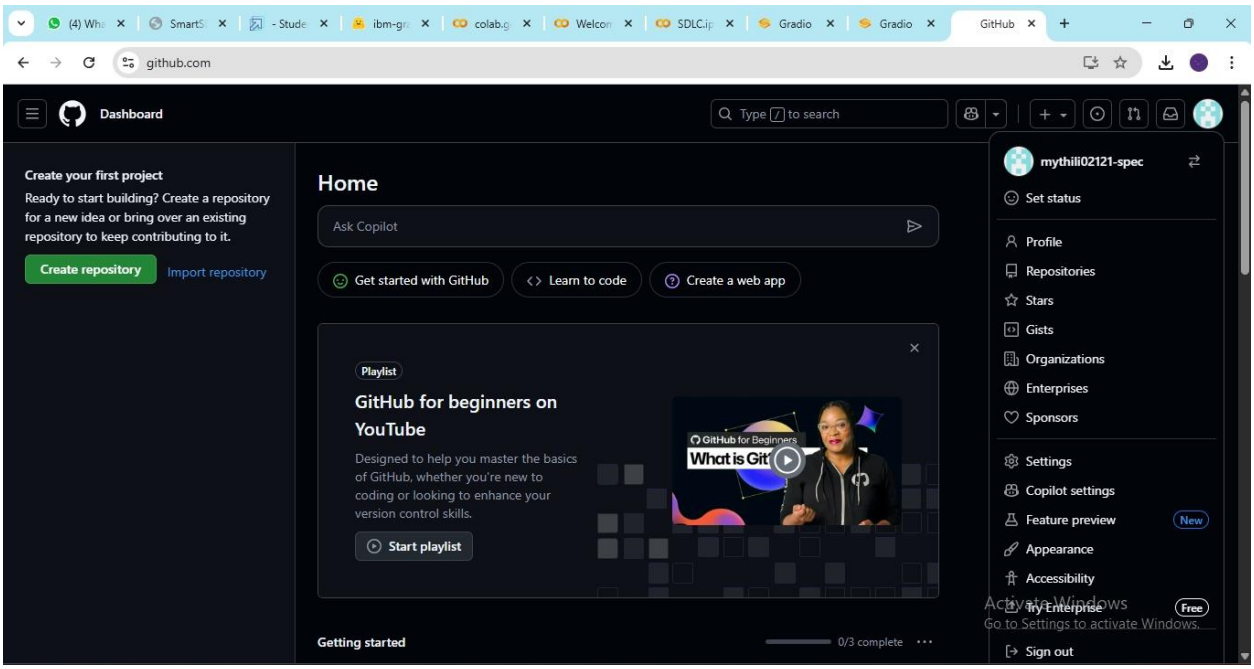
- Search for “GitHub” in any browser, then click on the first link (GitHub)



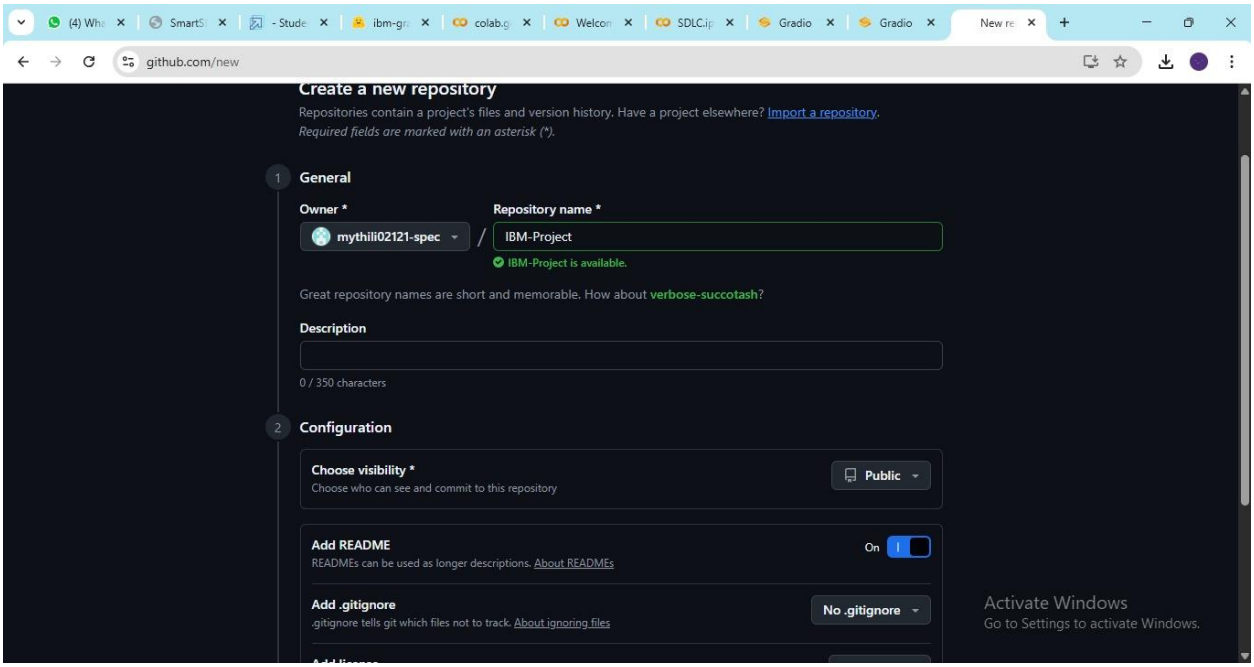
- Then click on “Signup” and create your own account in GitHub. If you already have an account click on “Sign in”.



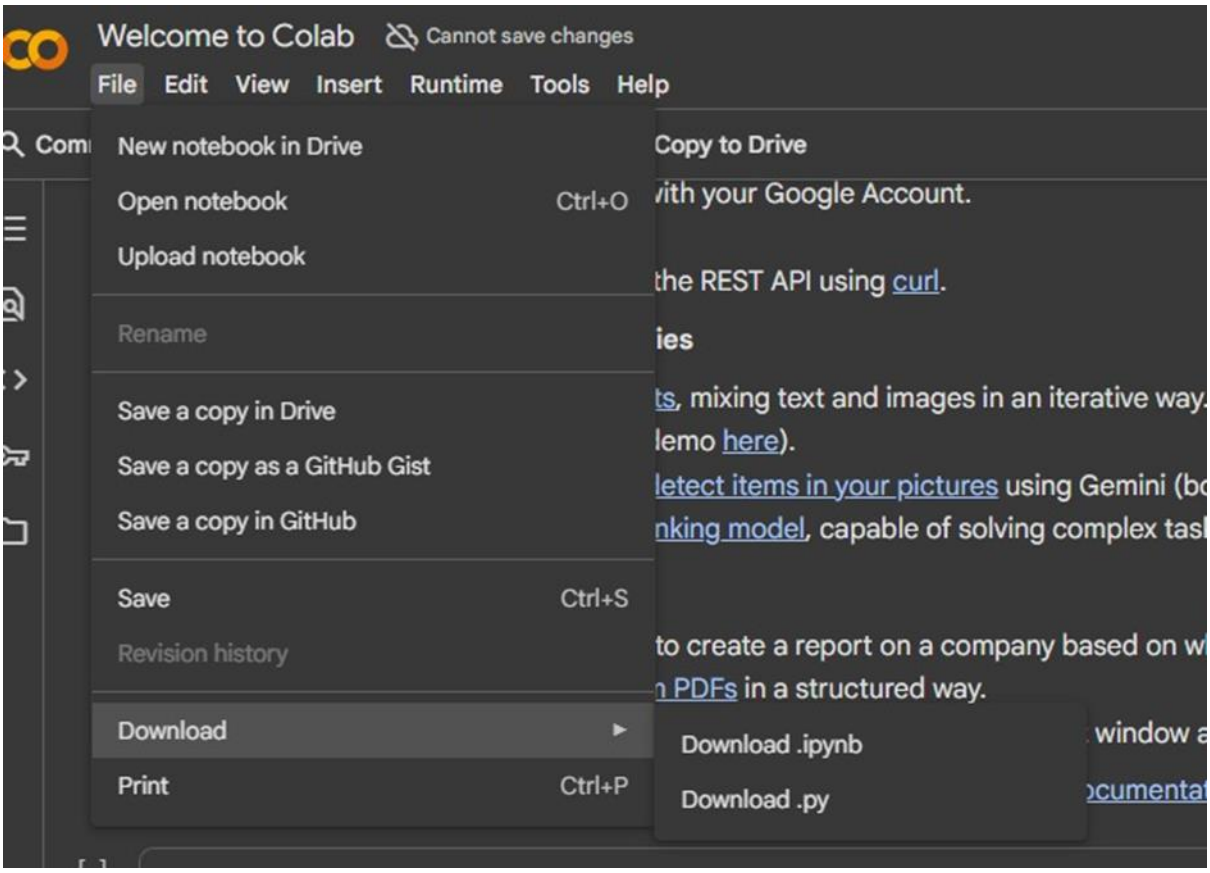
- Click on “Create repository”.



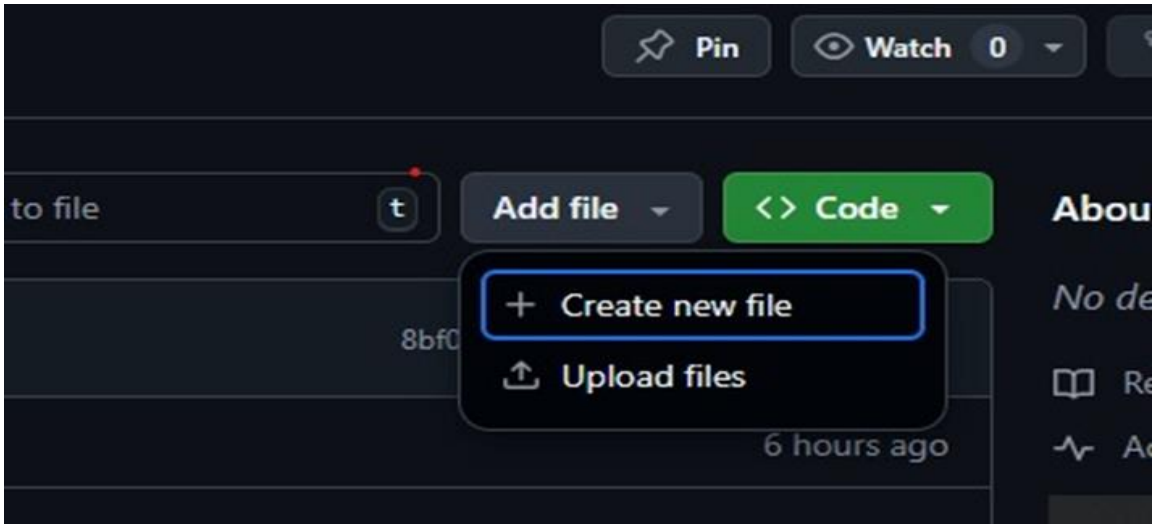
- In “General” Name your repo. (Here I have given “IBM-Project” as my repo name and it is available).
- In “Configurations” Turn On “Add readme” file Option.



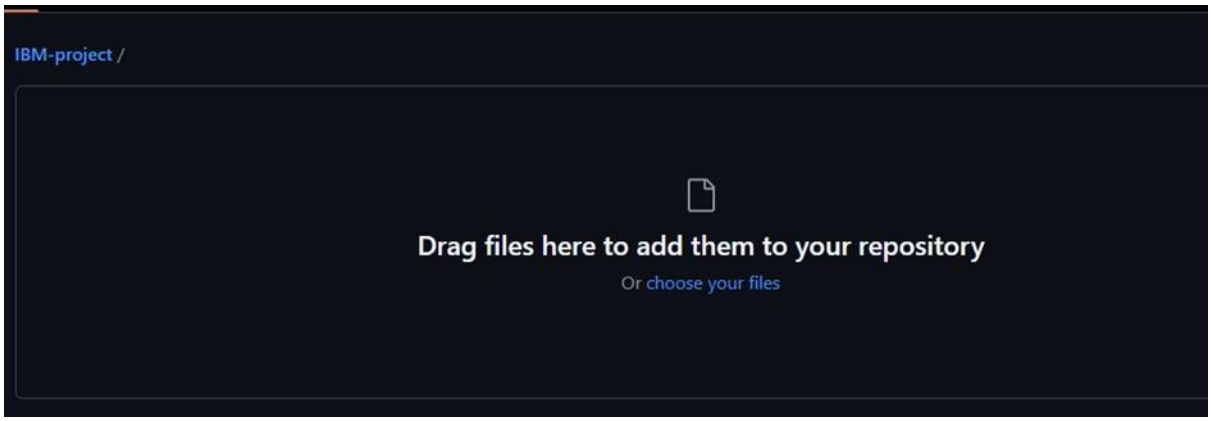
- Now Download your code from Google collab by Clicking on “File”, then Goto “Download” then download as “.py”.



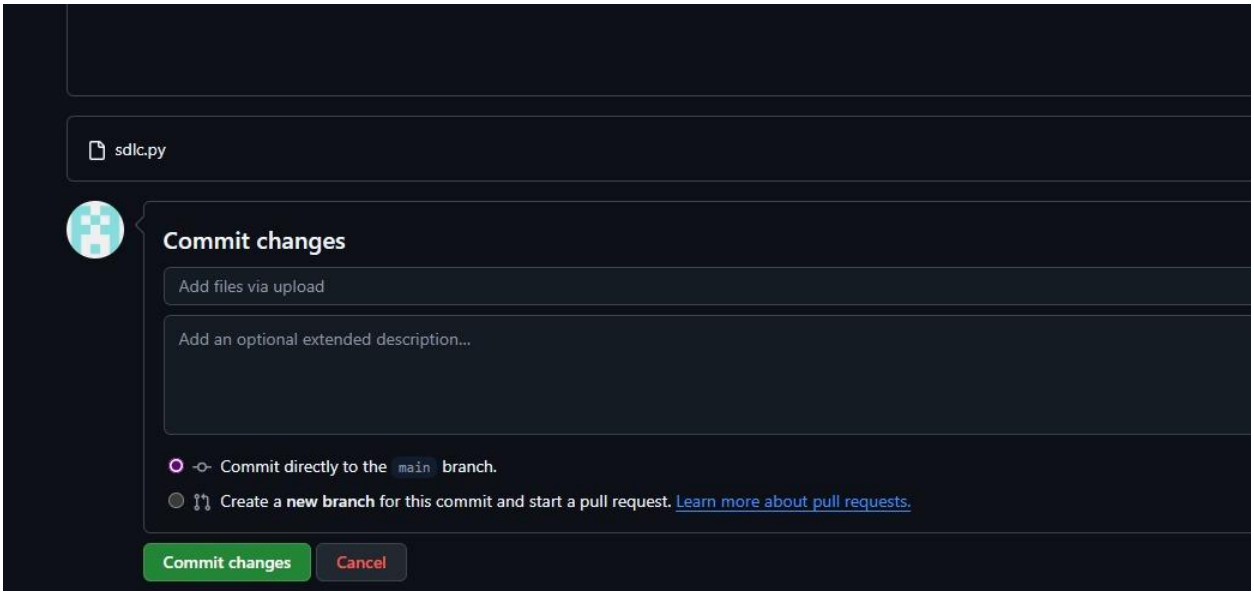
- Then your repository is created, then Click on “Add file” Option. Then Click 15 “Upload files” to upload your files.



- Click on “choose your files”.



- Choose your project file and click on “Open”.
- After your file has Uploaded Click on “Commit changes”.



- Successfully uploaded code into GitHub.

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