|  |  |
| --- | --- |
| DATE | 10/09/2025 |
| NM ID | NM2025TMID00605 |
| PROJECT NAME | SUSTAINABLE SMART CITY ASSIATANT |
| MAXIMUM MARKS |  |

# 1. Introduction

The Sustainable Smart City Assistance project is designed to provide AI-powered solutions for sustainable living and policy analysis.   
It leverages Natural Language Processing (NLP) models to generate eco-friendly tips and summarize policy documents, helping users, students, and researchers in the field of sustainability.  
  
This project integrates Gradio, Hugging Face Transformers, and Google Colab for seamless deployment, testing, and user interaction.

# 2. Objectives

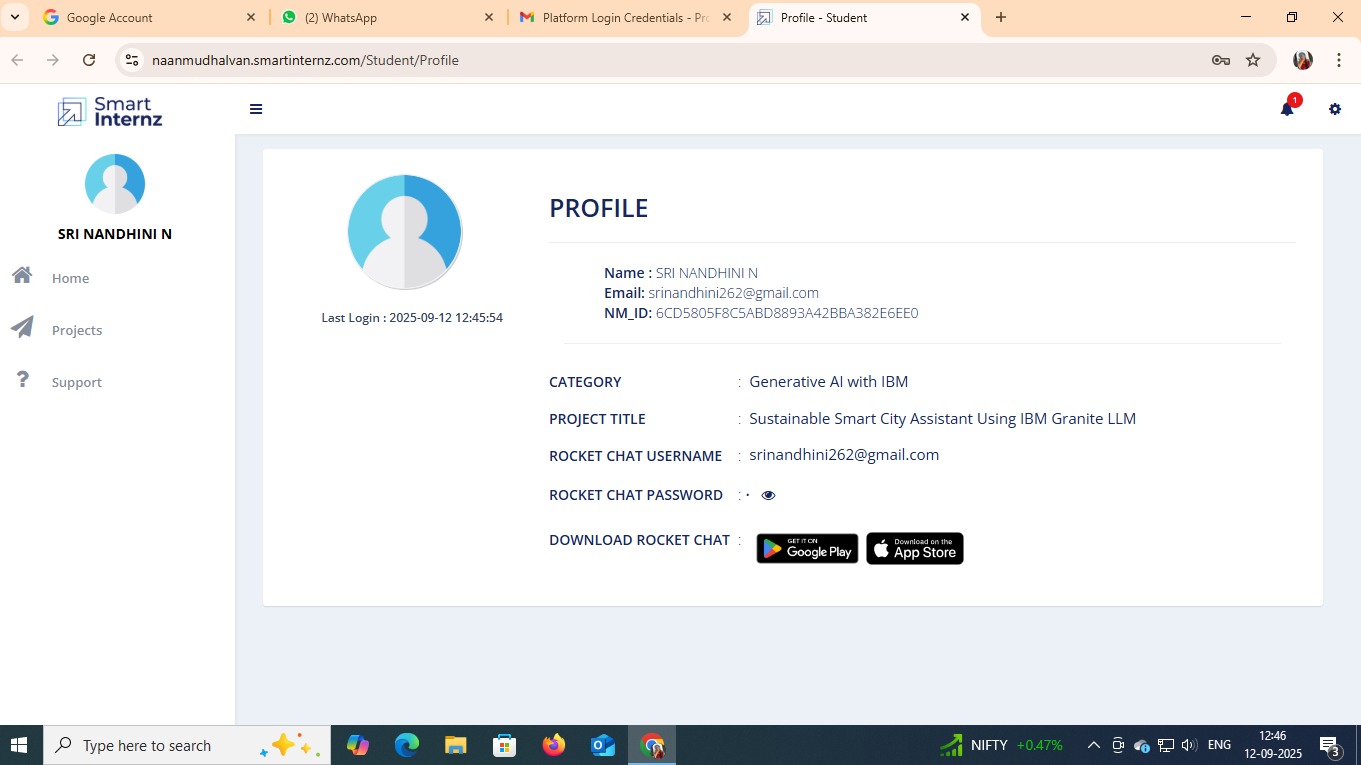
- To assist individuals in adopting eco-friendly lifestyle choices through AI-generated recommendations.  
- To analyze and summarize policy documents for quick understanding of key provisions.  
- To build a user-friendly interface that allows interaction with AI models without requiring deep technical knowledge.  
- To enable cloud-based execution using Google Colab for accessibility and ease of use.

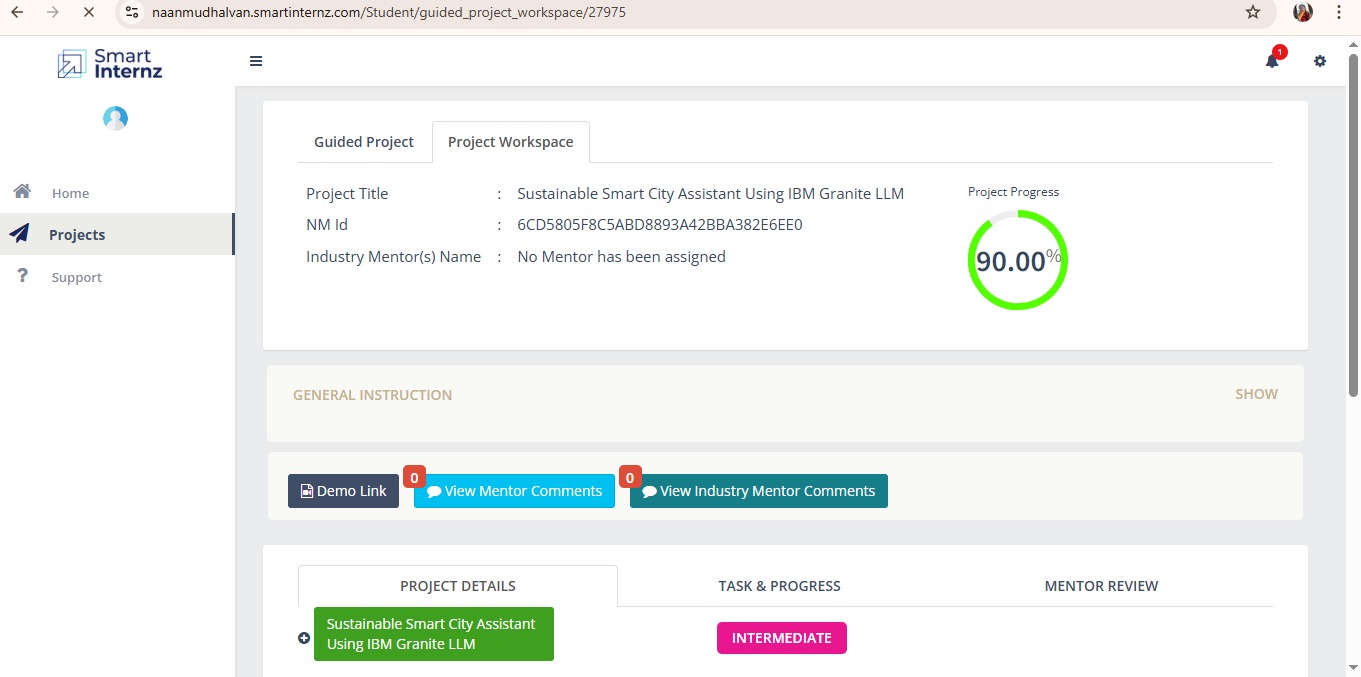
# 3. Key Features

- Eco Tips Generator: Provides practical sustainability tips based on environmental keywords.  
- Policy Summarization: Upload a PDF or paste text to get concise summaries of government or organizational policies.  
- Hugging Face Model Integration: Uses the ibm-granite/granite-3.2-2b-instruct model for text generation and summarization.  
- Gradio Interface: Simple and interactive web-based UI.  
- Google Colab Compatibility: Easy to run on cloud notebooks without local setup.  
- PDF Support: Extracts and processes text from uploaded PDF files using PyPDF2.

# 4. SmartInternz Login

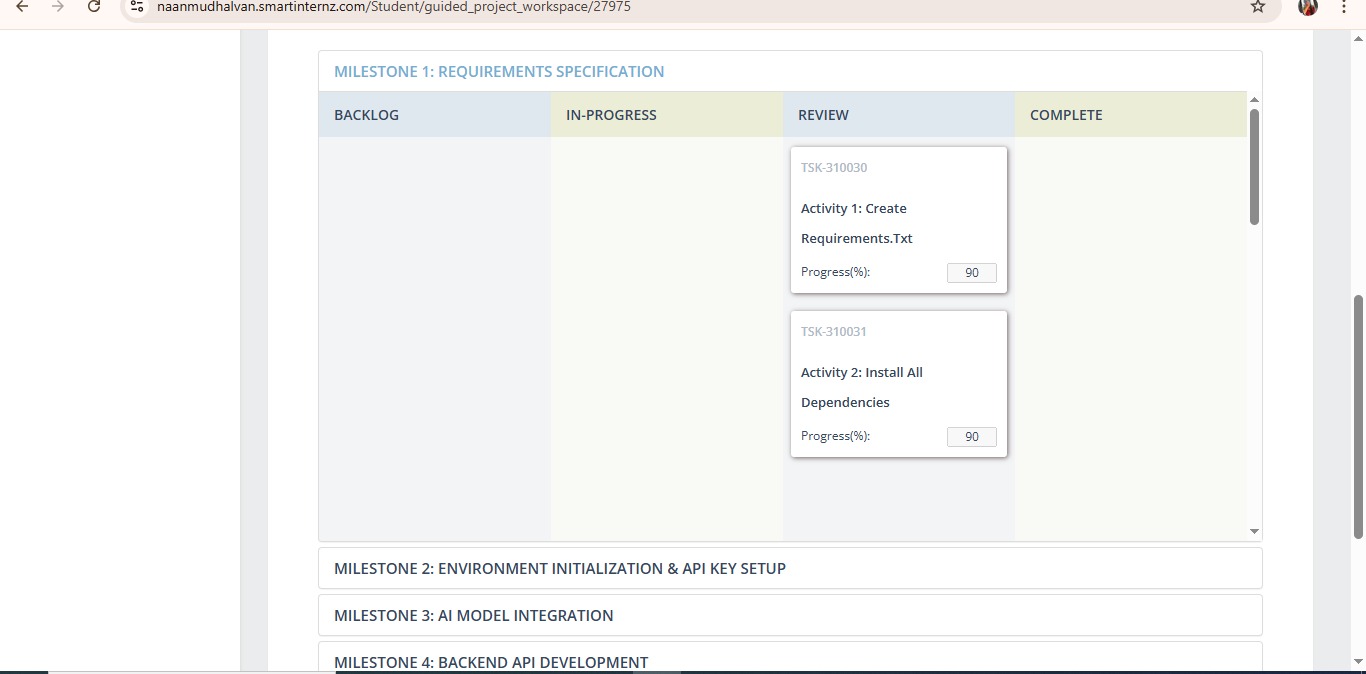
The project can be integrated into SmartInternz for internship or learning purposes. Below is the screenshot of the SmartInternz login and project workspace:



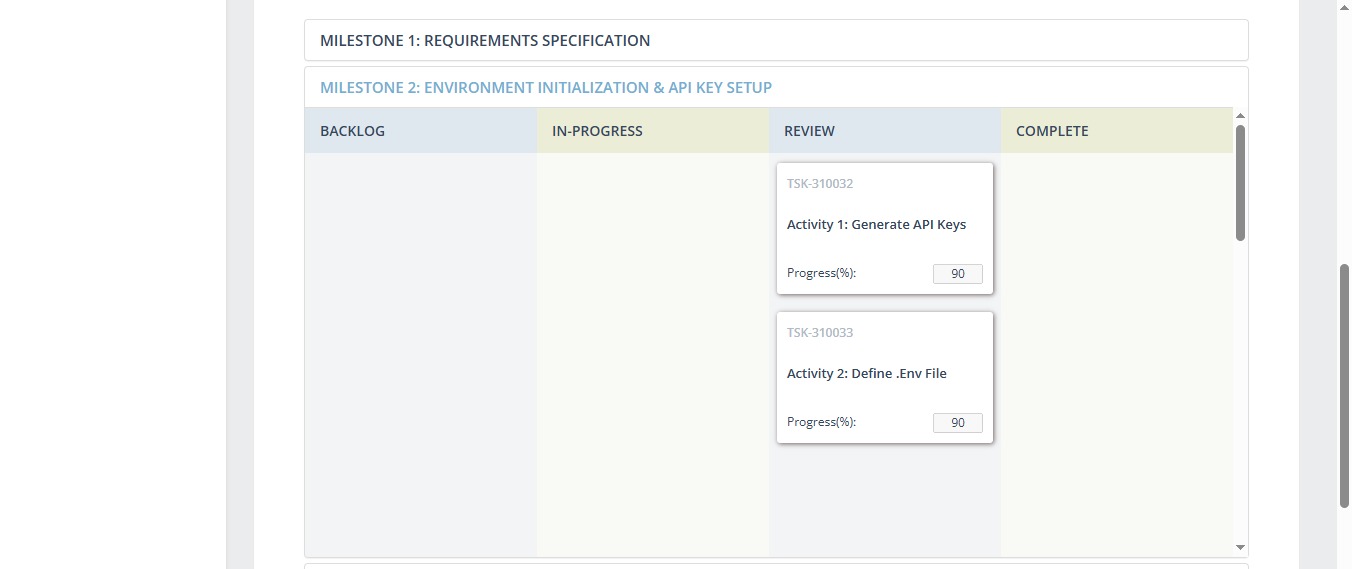


# 5. Milestones

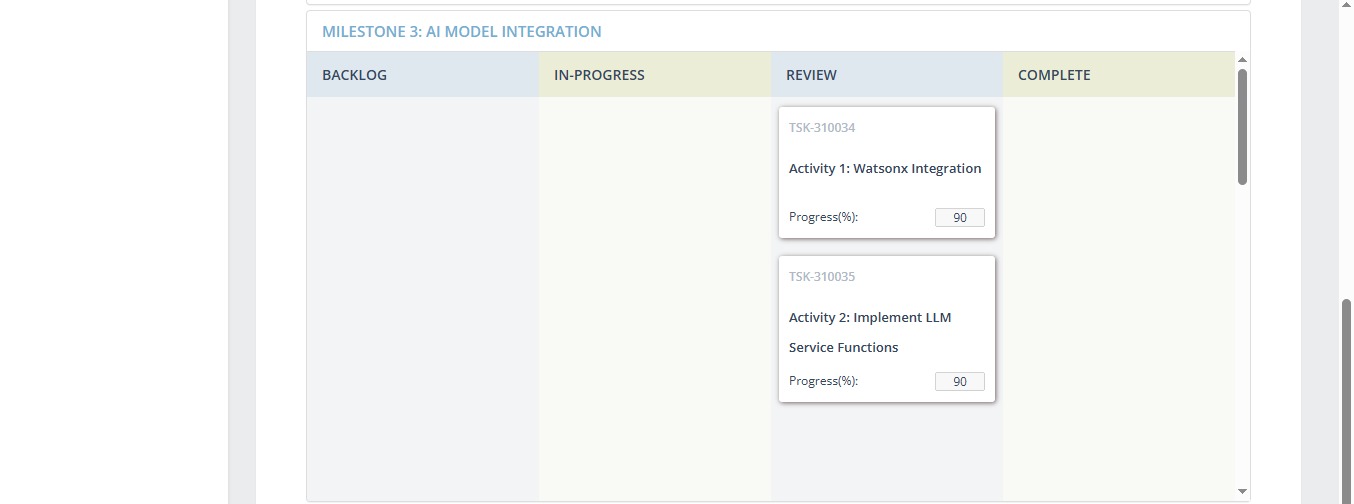
**Milestone 1**

****

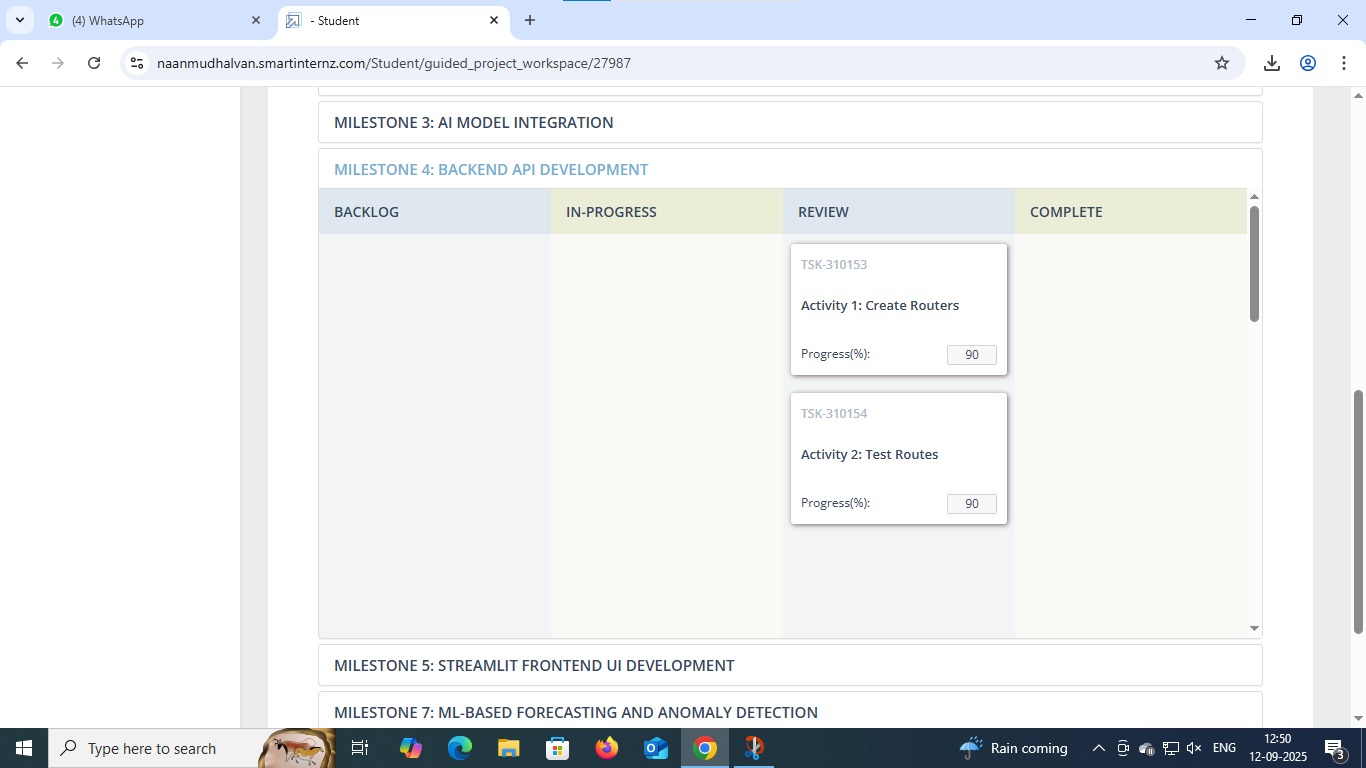
**Milestone 2**

****

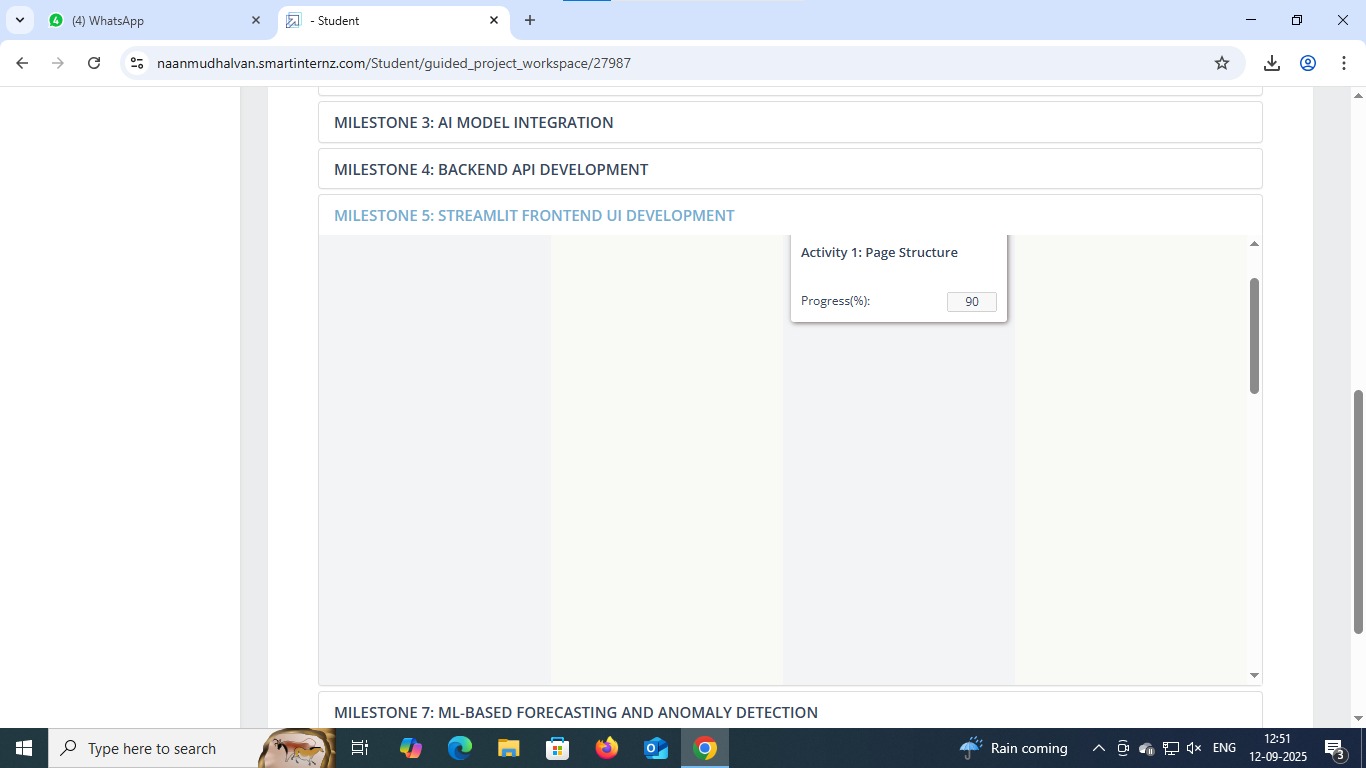
**Milestone 3**

****

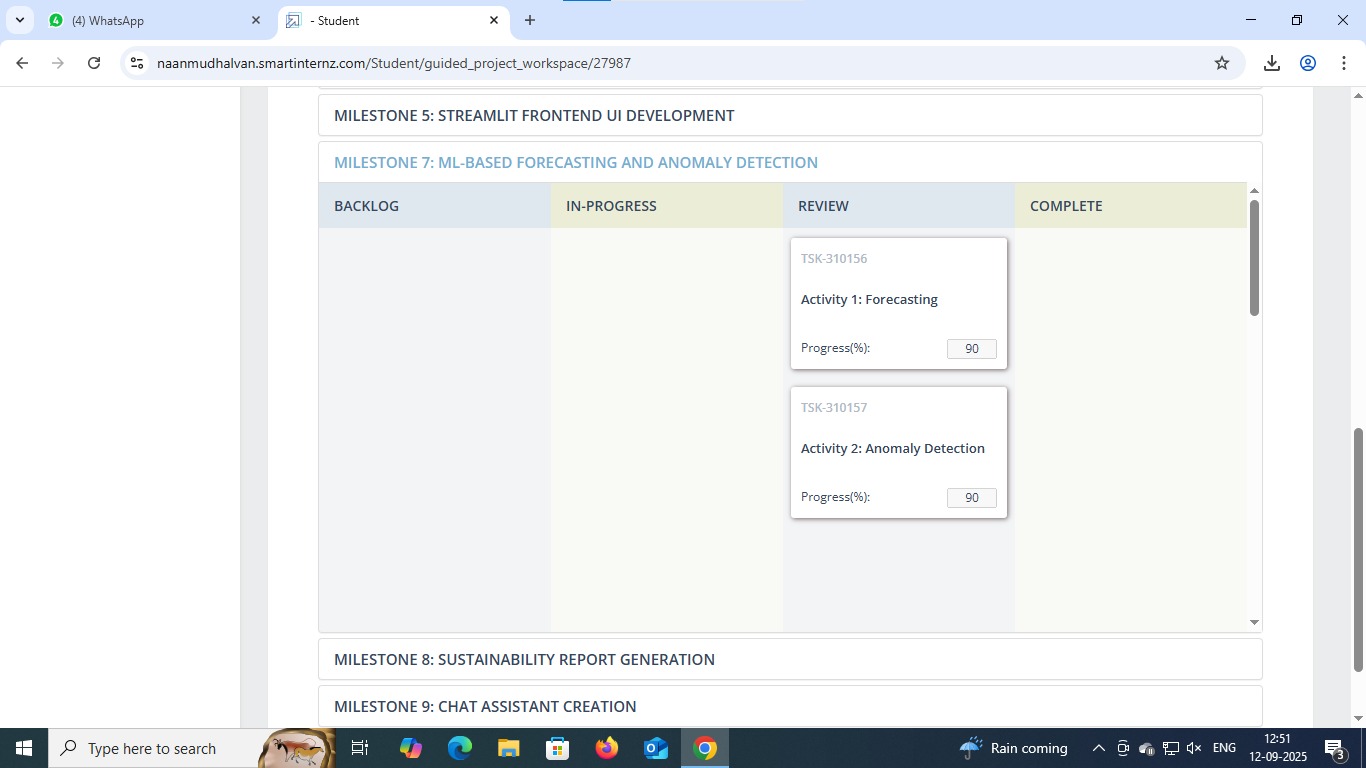
**Milestone 4**

****

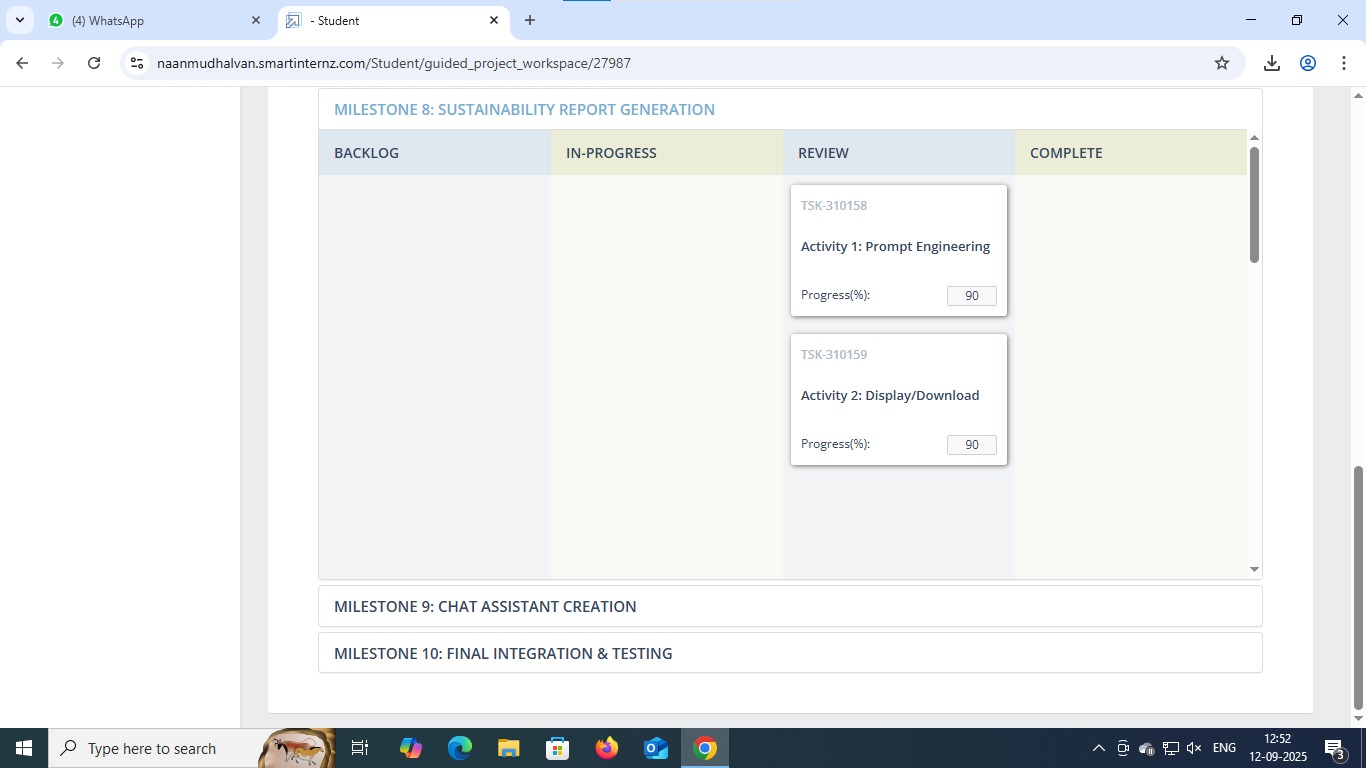
**Milestone 5**

****

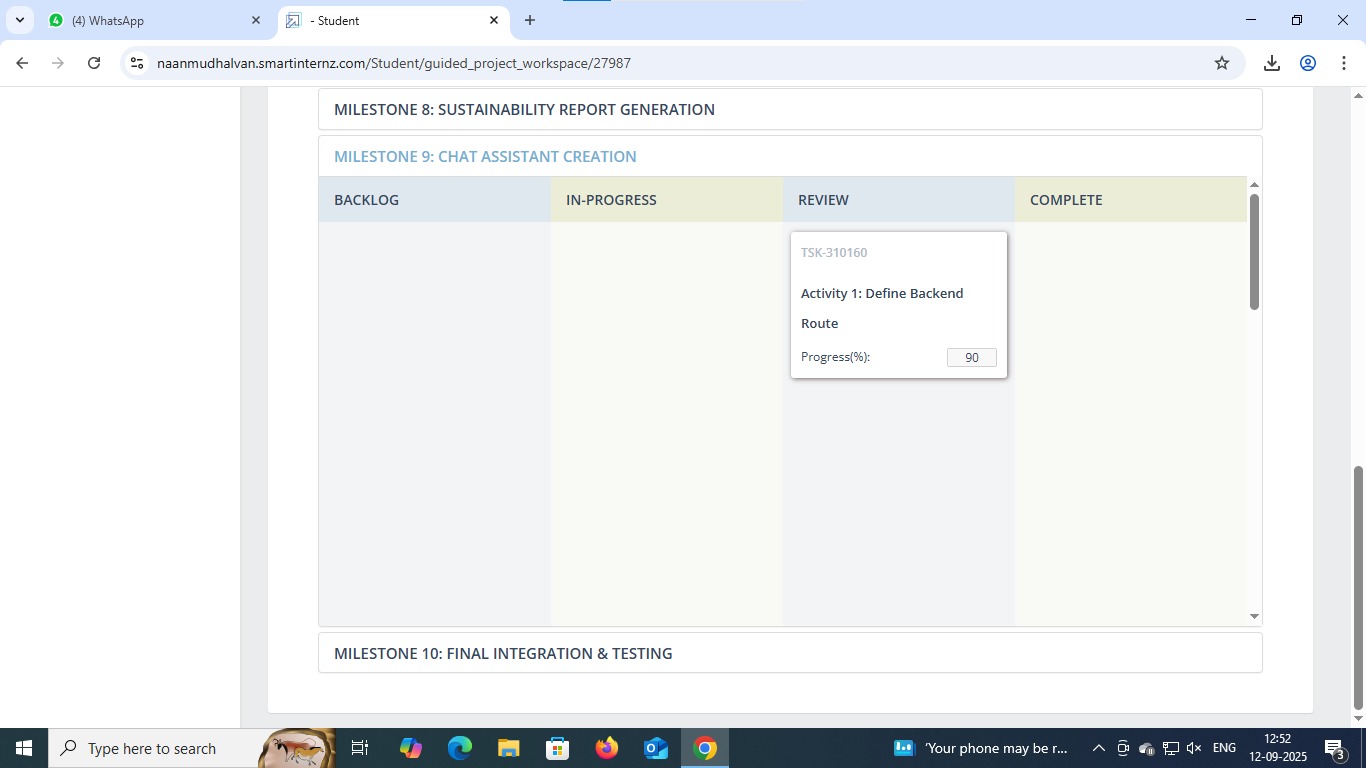
**Milestone 7**

****

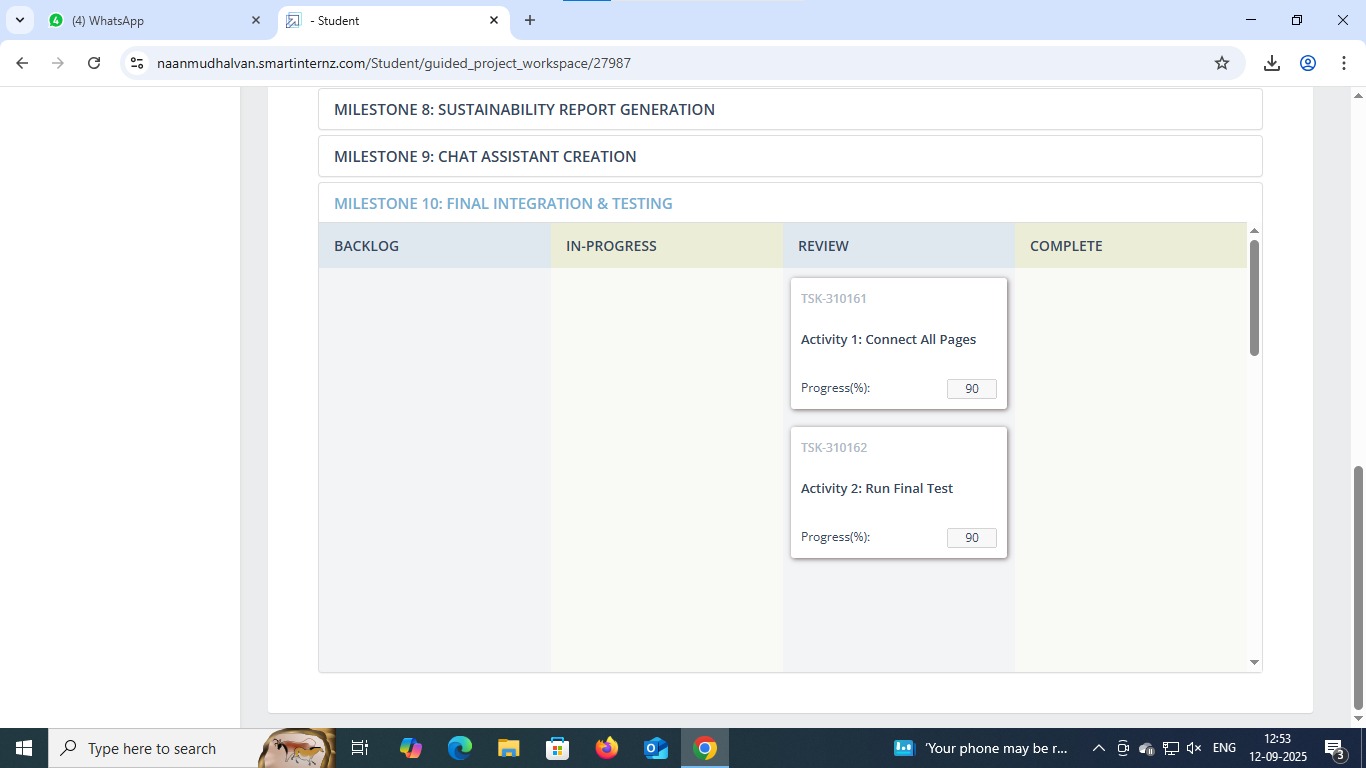
**Milestone 8**

****

**Milestone 9**

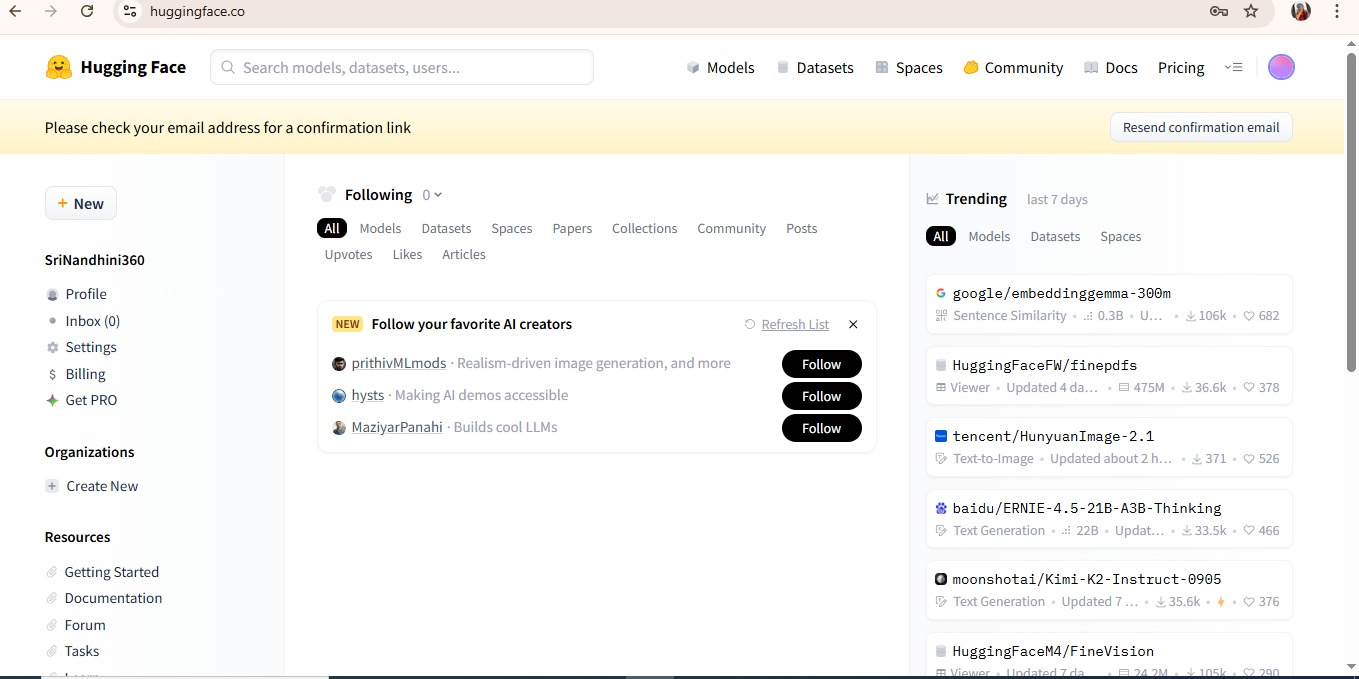
****

**Milestone 10**

****

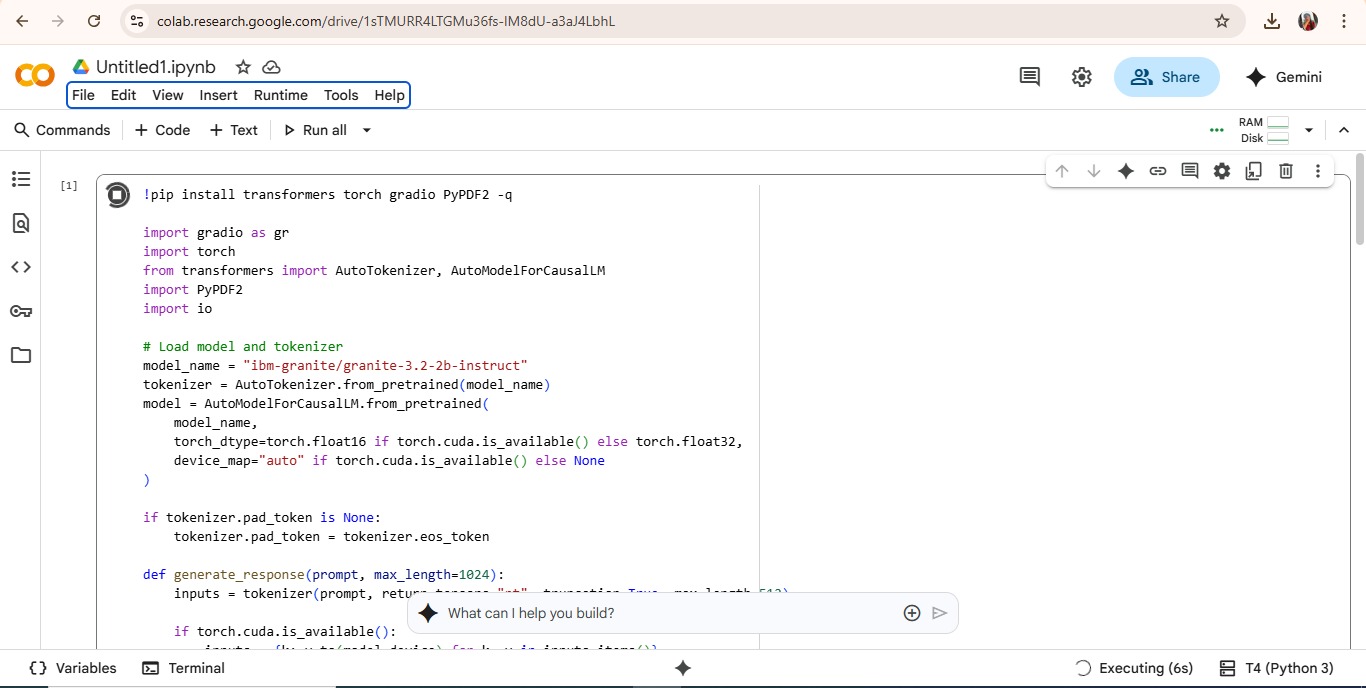
# 6. Hugging Face Login

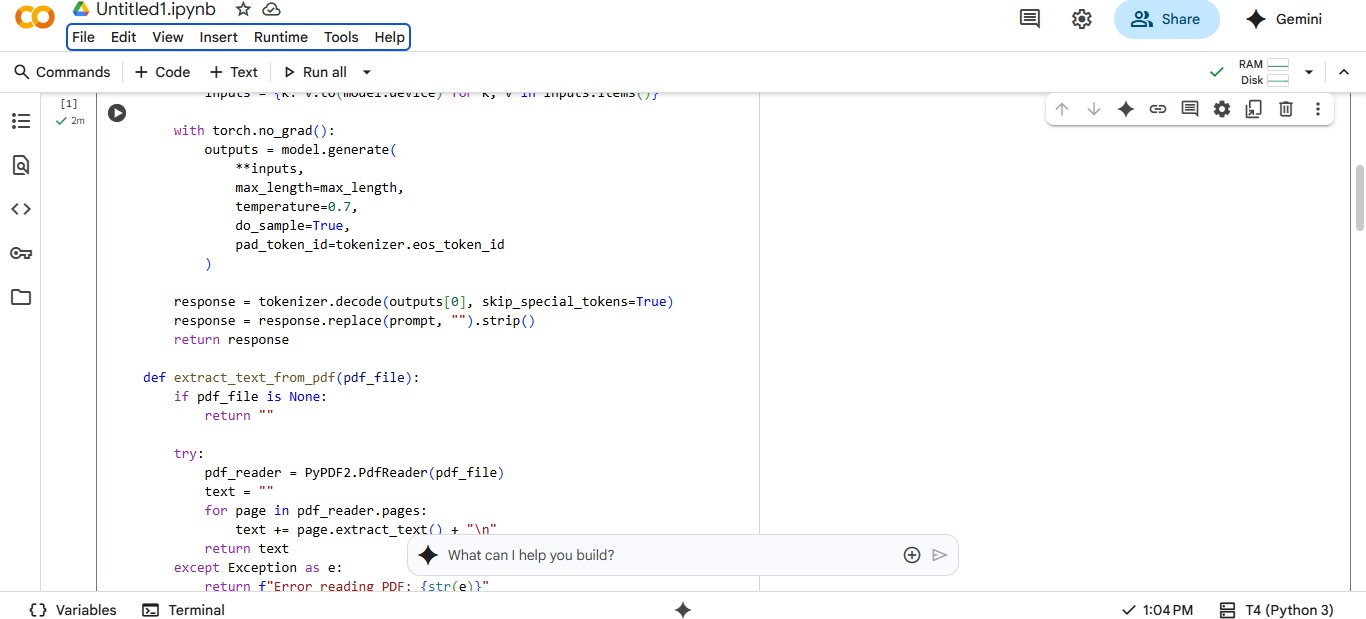
Since the project uses Hugging Face models, login is required. Below is the screenshot of the Hugging Face login page with access token:



# 7. Google Colab Coding

The project can be run on Google Colab with code execution and output. Below are the screenshots of Colab notebook execution:



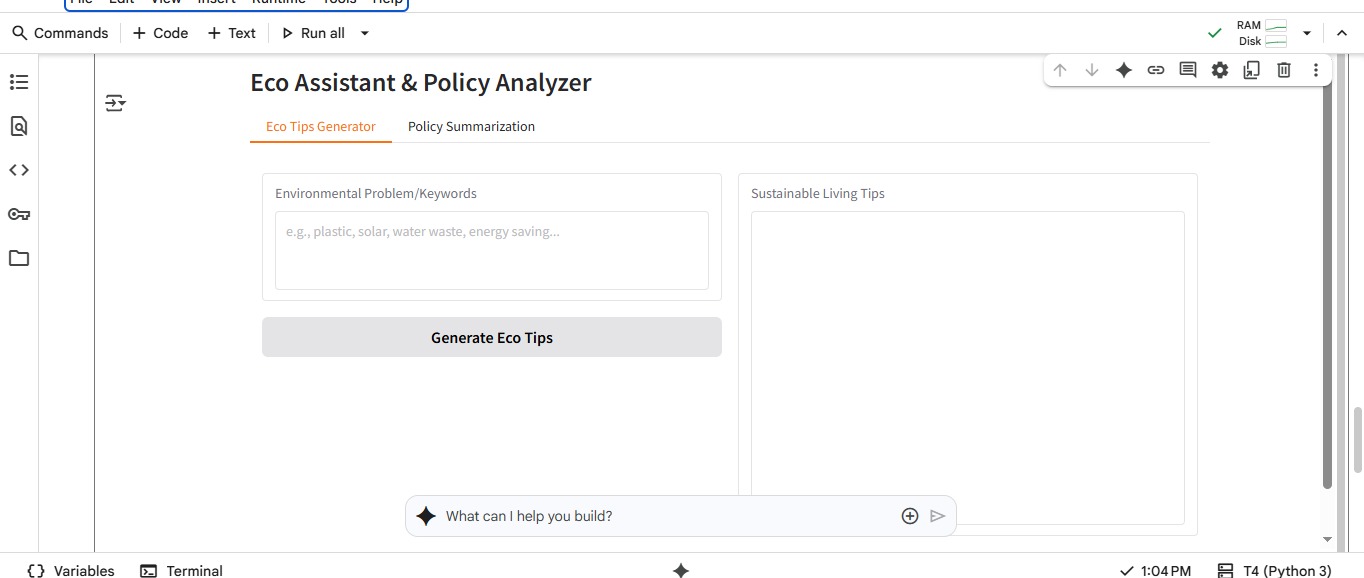
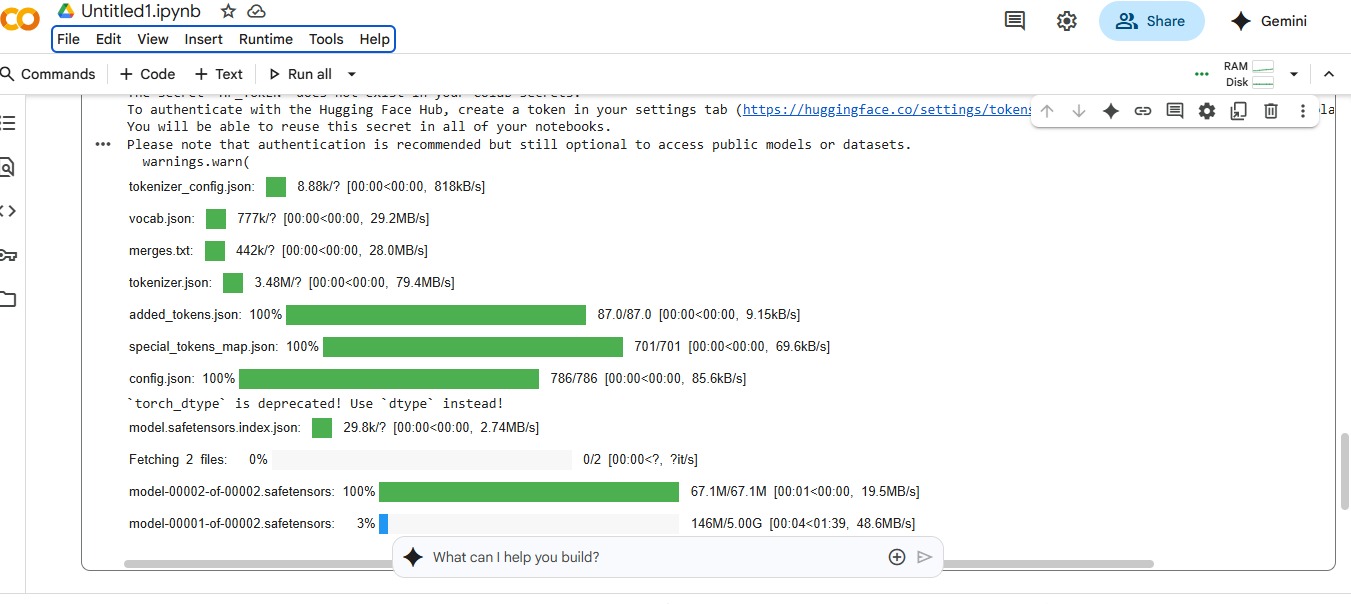






# 8. Coding

!pip install transformers torch gradio PyPDF2 -q  
  
import gradio as gr  
import torch  
from transformers import AutoTokenizer, AutoModelForCausalLM  
import PyPDF2  
import io  
  
# 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  
 )  
  
 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"  
 return text  
 except Exception as e:  
 return f"Error reading PDF: {str(e)}"  
  
def eco\_tips\_generator(problem\_keywords):  
 prompt = f"Generate practical and actionable eco-friendly tips for sustainable living related to: {problem\_keywords}. Provide specific solutions and suggestions:"  
 return generate\_response(prompt, max\_length=1000)  
  
def policy\_summarization(pdf\_file, policy\_text):  
 # Get text from PDF or direct input  
 if pdf\_file is not None:  
 content = extract\_text\_from\_pdf(pdf\_file)  
 summary\_prompt = f"Summarize the following policy document and extract the most important points, key provisions, and implications:\n\n{content}"  
 else:  
 summary\_prompt = f"Summarize the following policy document and extract the most important points, key provisions, and implications:\n\n{policy\_text}"  
  
 return generate\_response(summary\_prompt, max\_length=1200)  
  
# Create Gradio interface  
with gr.Blocks() as app:  
 gr.Markdown("# Eco Assistant & Policy Analyzer")  
  
 with gr.Tabs():  
 with gr.TabItem("Eco Tips Generator"):  
 with gr.Row():  
 with gr.Column():  
 keywords\_input = gr.Textbox(  
 label="Environmental Problem/Keywords",  
 placeholder="e.g., plastic, solar, water waste, energy saving...",  
 lines=3  
 )  
 generate\_tips\_btn = gr.Button("Generate Eco Tips")  
  
 with gr.Column():  
 tips\_output = gr.Textbox(label="Sustainable Living Tips", lines=15)  
  
 generate\_tips\_btn.click(eco\_tips\_generator, inputs=keywords\_input, outputs=tips\_output)  
  
 with gr.TabItem("Policy Summarization"):  
 with gr.Row():  
 with gr.Column():  
 pdf\_upload = gr.File(label="Upload Policy PDF", file\_types=[".pdf"])  
 policy\_text\_input = gr.Textbox(  
 label="Or paste policy text here",  
 placeholder="Paste policy document text...",  
 lines=5  
 )  
 summarize\_btn = gr.Button("Summarize Policy")  
  
 with gr.Column():  
 summary\_output = gr.Textbox(label="Policy Summary & Key Points", lines=20)  
  
 summarize\_btn.click(policy\_summarization, inputs=[pdf\_upload, policy\_text\_input], outputs=summary\_output)  
  
app.launch(share=True)



# 9. Conclusion

The Sustainable Smart City Assistance project demonstrates how AI can support sustainability initiatives by providing actionable eco-tips and simplifying policy understanding.   
With Hugging Face, Google Colab, and Gradio, it ensures easy deployment and accessibility for students, researchers, and organizations working toward sustainable development goals (SDGs).