Sustainable Smart City Assistant Using IBM Granite LLM

Generative AI with IBM

PROJECT DOCUMENTATION

1.Introduction

Project title: Sustainable Smart City Assistant Using IBM Granite LLM

Generative AI with IBM

Team Leader**:** Hari Priya.M

Team Members

1.Girija.S

2.Anusha bai.D

3.Deva Dharshini.V

2.Project Overview

Purpose:

The purpose of the Sustainable Smart City Assistant is to empower cities and citizens with AI-driven insights that promote sustainability, efficiency, and improved quality of life. By leveraging IBM Granite's large language model capabilities, the assistant enables

3.Key Features

Smart Energy Assistant → Optimize energy consumption and suggest conservation methods.

Green Mobility Guide → Recommend public transport, carpooling, EV routes, and cycling paths.

Waste & Water Management Support → Reminders for recycling, efficient water use suggestions.

Citizen Query Handling → Answer questions about services, sustainability tips, and city updates.

Admin Dashboard → Analytics and reports for city officials (air quality, traffic, energy usage).

4.Architecture

Layered Architecture

a. Data Layer

Sources:

IoT Sensors (traffic, air quality, energy meters, water flow).

Public Data (transport schedules, weather forecasts, government APIs).

Citizen Inputs (queries, feedback, complaints).

Function: Collects real-time structured and unstructured data.

5.Setup instructions

Cloud Platform: IBM Cloud account with access to AI/ML services.

LLM Access: IBM Granite model access (API/SDK).

Data Sources: IoT sensor data, public APIs (transport, weather, energy, waste).

Development Environment:

Backend → Node.js / Python (Flask or FastAPI).

Frontend → React (for web), Flutter/Android (for mobile).

Database → IBM Db2 / PostgreSQL for structured data.

APIs: Integration with smart city services (transport, utilities, etc.).

6. Folder structure

backend/ → Handles all API logic for citizens & admins.

ai_integration/ → Dedicated space for IBM Granite API calls, NLP, and AI tasks.

data_layer/ → Connectors for IoT devices, public APIs, and database schema.

frontend/ → Two UIs: one for citizens, one for administrators.

docs/ → Contains reports, architecture diagrams, manuals.

scripts/ → For deployment, database seeding, automation.

config/ → Stores all environment-specific settings.

7. API Documentation

This documentation shows how the API endpoints support both citizens (queries, tips, usage) and administrators (reports, alerts, forecasts) — all powered by IBM Granite AI in the background for NLP, summarization, and predictions.

8.Authentication

User Roles

Citizen → asks queries, gets tips.

Admin → manages reports, alerts.

Method

JWT (JSON Web Token) for login sessions.

API Key for IBM Granite access (kept hidden in backend).

Flow

User logs in → system verifies → issues JWT token.

User sends token with each API request (Authorization: Bearer <token>).

Backend uses Granite API Key to process queries securely.

9.User Interface

Citizen Dashboard

Welcome Screen → Greeting + Quick Access buttons.

Query Box (AI Chat) → Citizens type or voice queries (e.g., "Where's the nearest EV charging station?").

Tips Panel → Daily eco-friendly tips (energy, waste, water).

Quick Actions

Check transport routes

View household energy usage

Waste collection schedule

Water usage insights

Admin Dashboard

Login Page → Secure login with 2FA.

Overview Panel → City-wide sustainability summary (energy, waste, water, air quality).Reports Section → AI-generated reports & predictions (Granite summarization).

Alerts Section → Critical issues (e.g., energy spike, waste overflow).

Management Tools → Approve, configure, or adjust smart city policies.

10.Testing

1. Unit Testing

API Endpoints → Test each REST API (e.g., /citizen/query, /energy/usage).

Granite Integration → Validate AI responses for correctness and clarity.

Authentication → Test JWT token issuance, expiry, and role-based access.

Example:

Input: GET /waste/schedule/A12

Expected: JSON with correct schedule & zone info.

2. Integration Testing

Granite AI + Backend → Ensure queries are processed and responses are meaningful.

IoT Data Sources → Verify integration with transport, energy, and waste systems.

Database Layer → Test data storage (citizen queries, energy reports, water usage logs).

3.functional Testing

Citizen Features

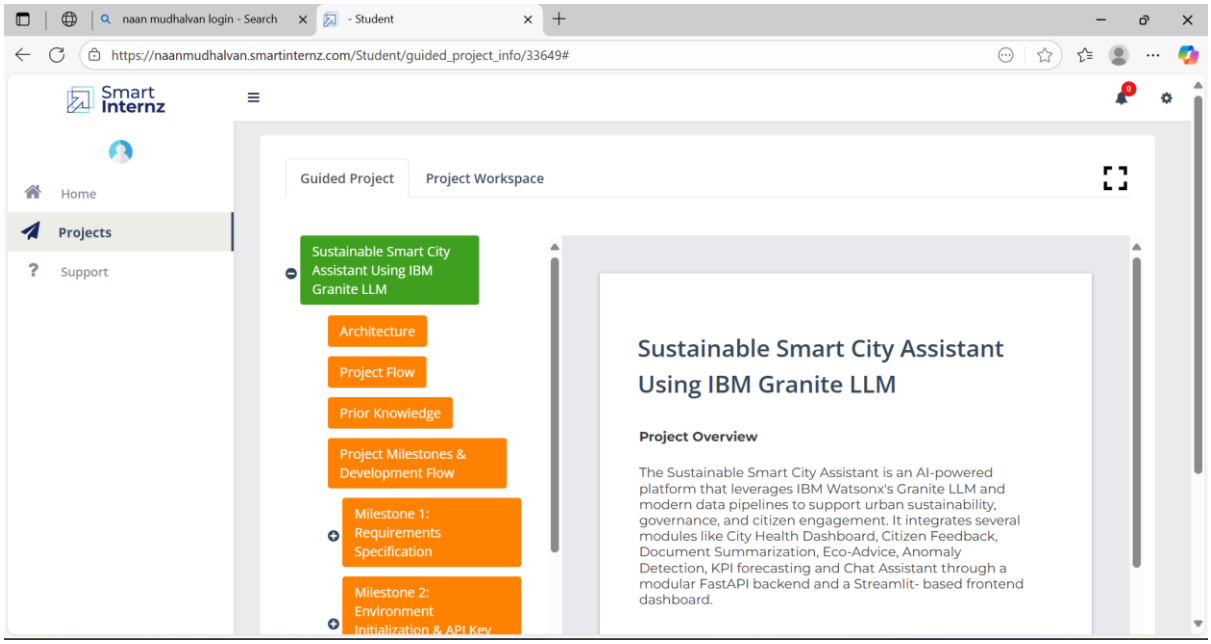
Can query assistant for EV stations, tips, schedules.

Multi-language queries (Granite NLP).

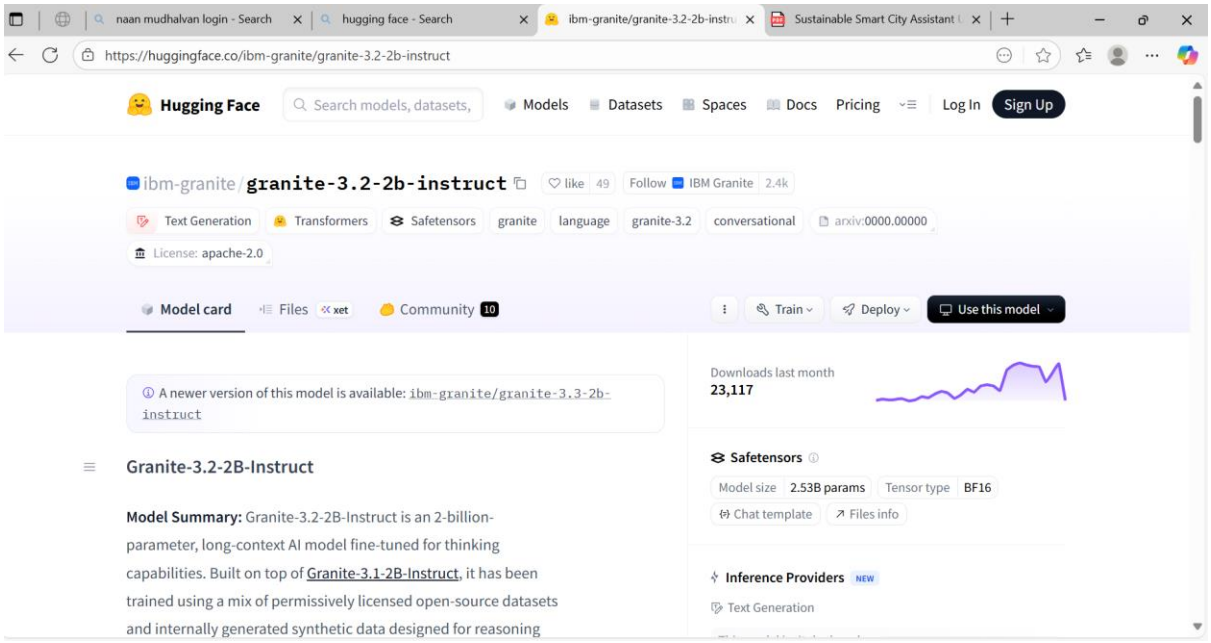
Admin Features

Can generate reports, view alerts, and manage dashboards.

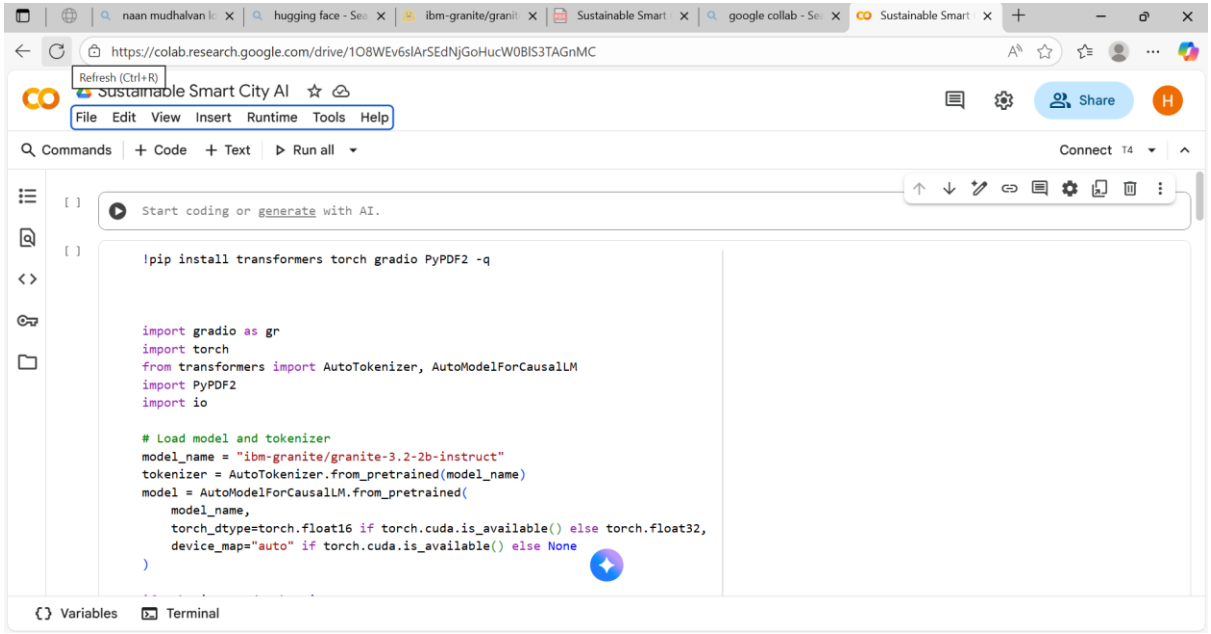
9.Project Screenshots

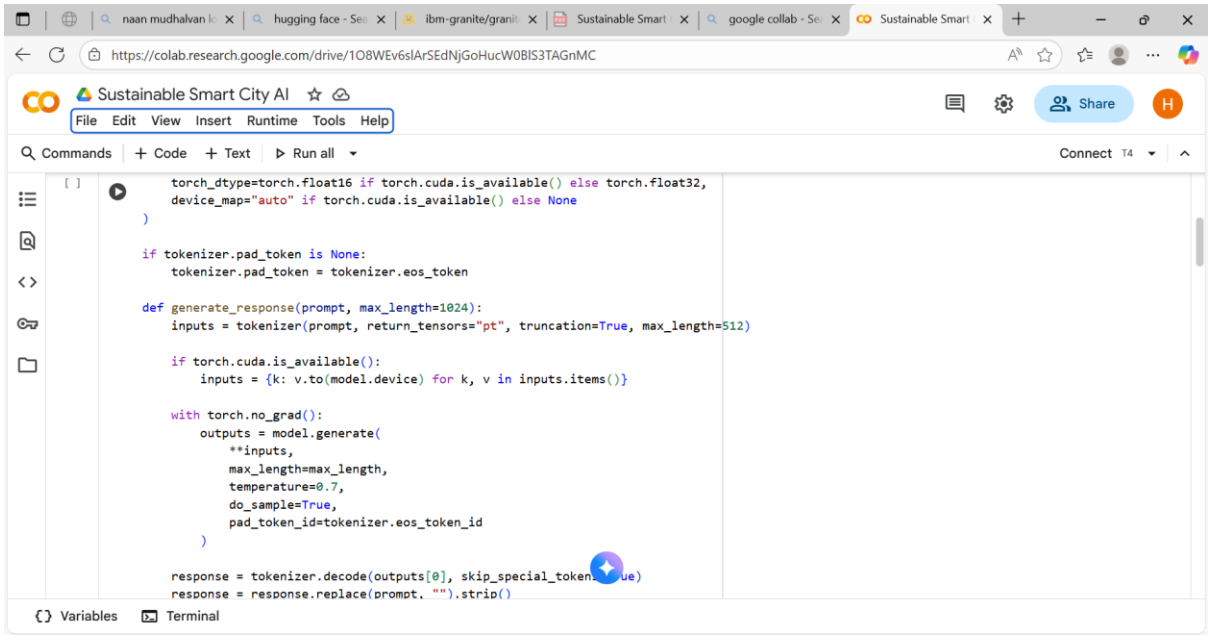


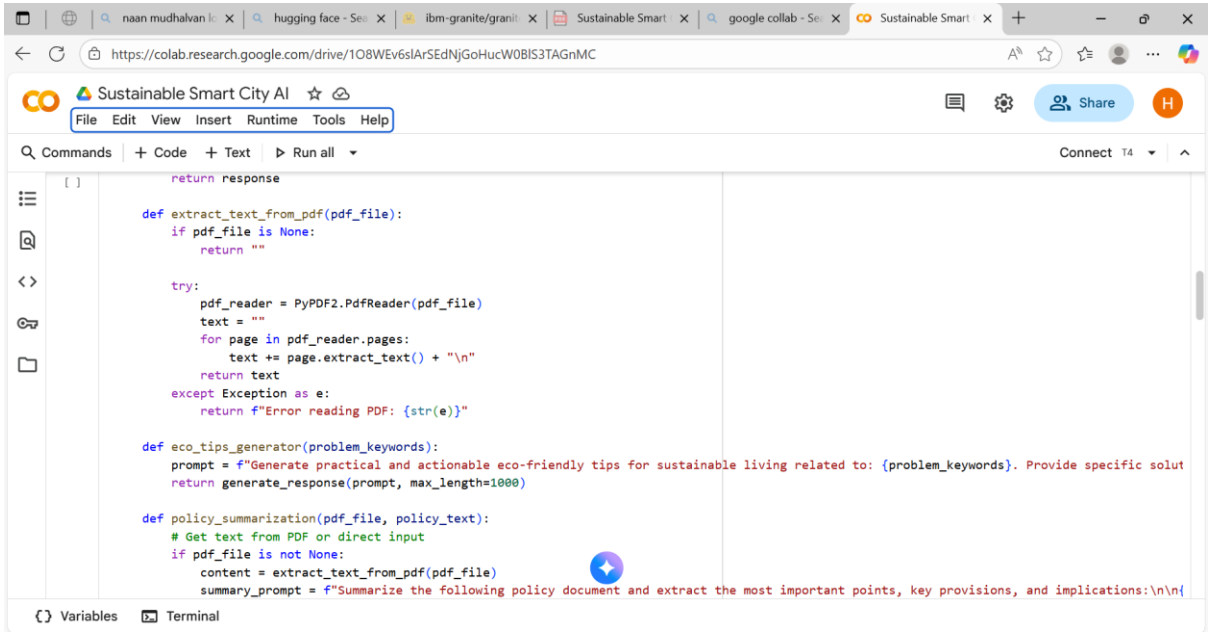
An IBM Granite model was selected from Hugging Face, a platform that hosts and allows collaboration on public AI models and datasets. The ibmgranite/granite-3.2-2b-instruct model was chosen for its performance and efficiency

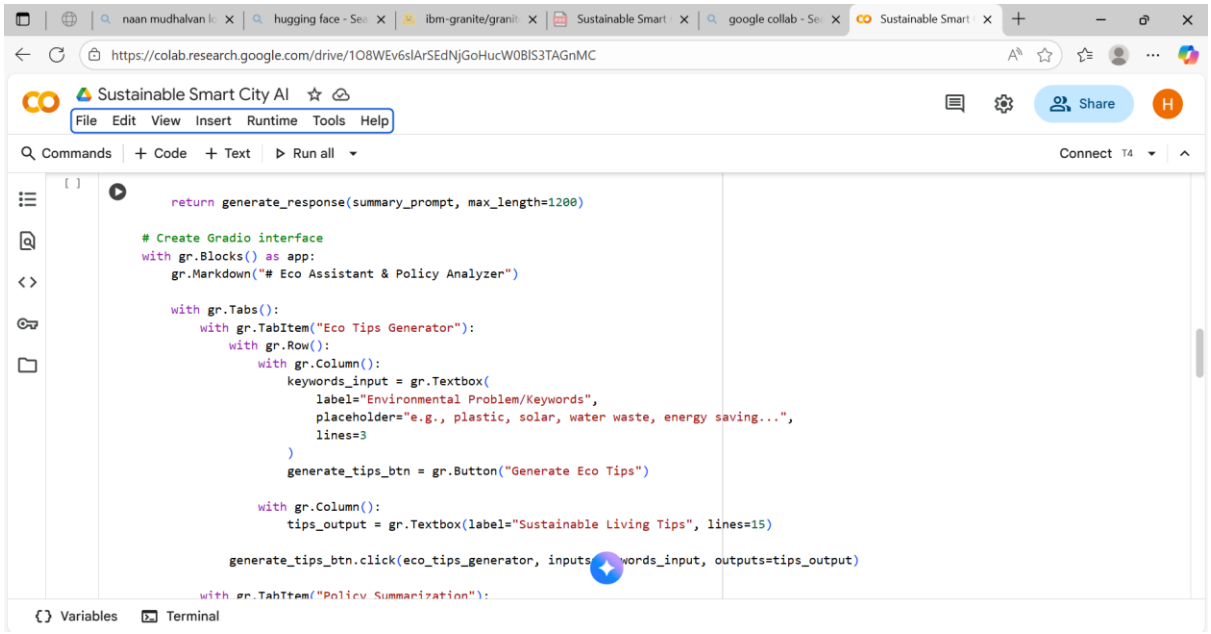


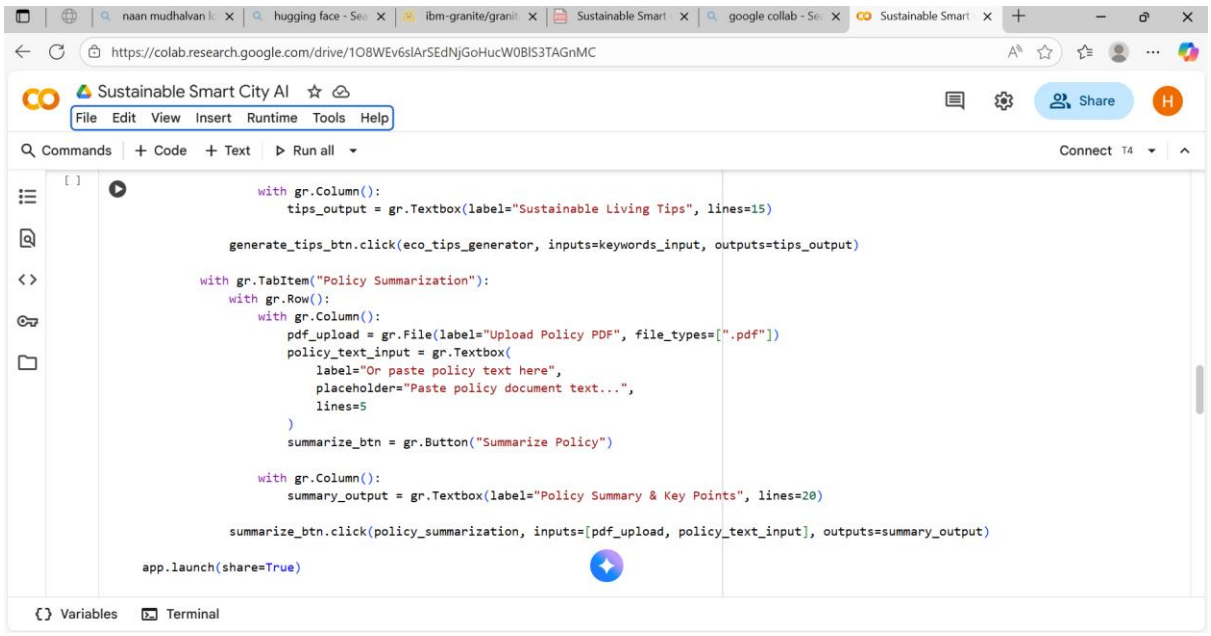
Code for Sustainable Smart City AI

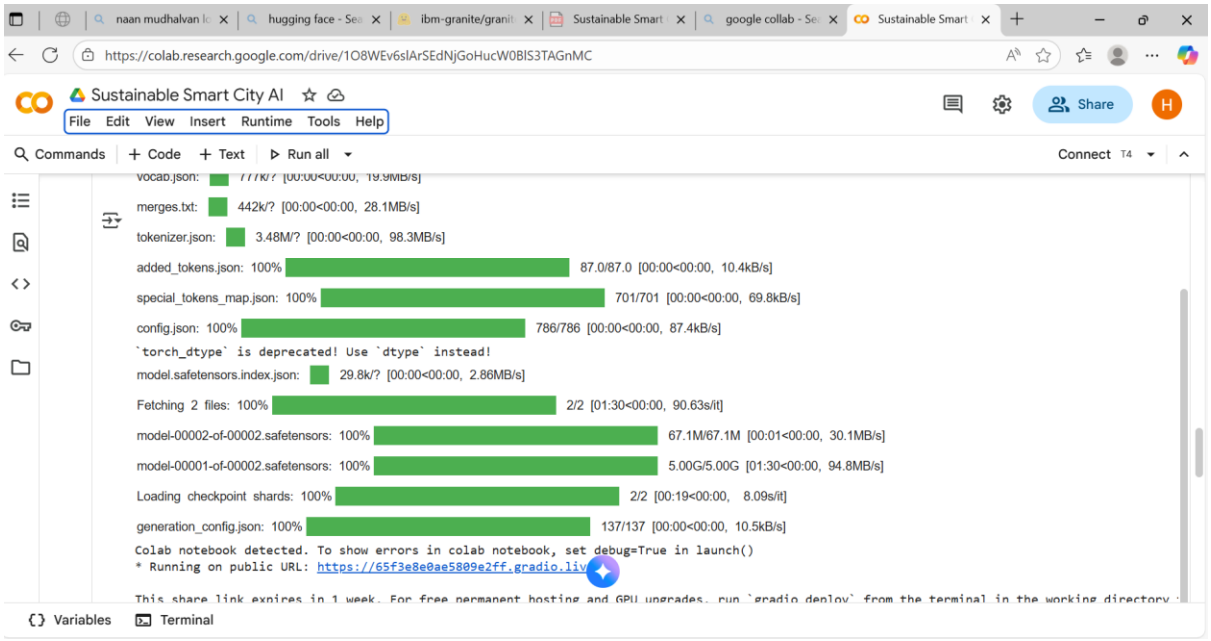



naan mudhalvan
hugging face - Se
ibm-granite/granit
Sustainable Smart
google collab - Se
Sustainable Smart
https://colab.research.google.com/drive/1O8WEv6slArSEdNjGoHucW0BlS3TAGnMC
Sustainable Smart City AI
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
Connect T4
torch_dtype=torch.float16 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_token
response = response.replace(prompt, "").strip()
Variables
Terminal

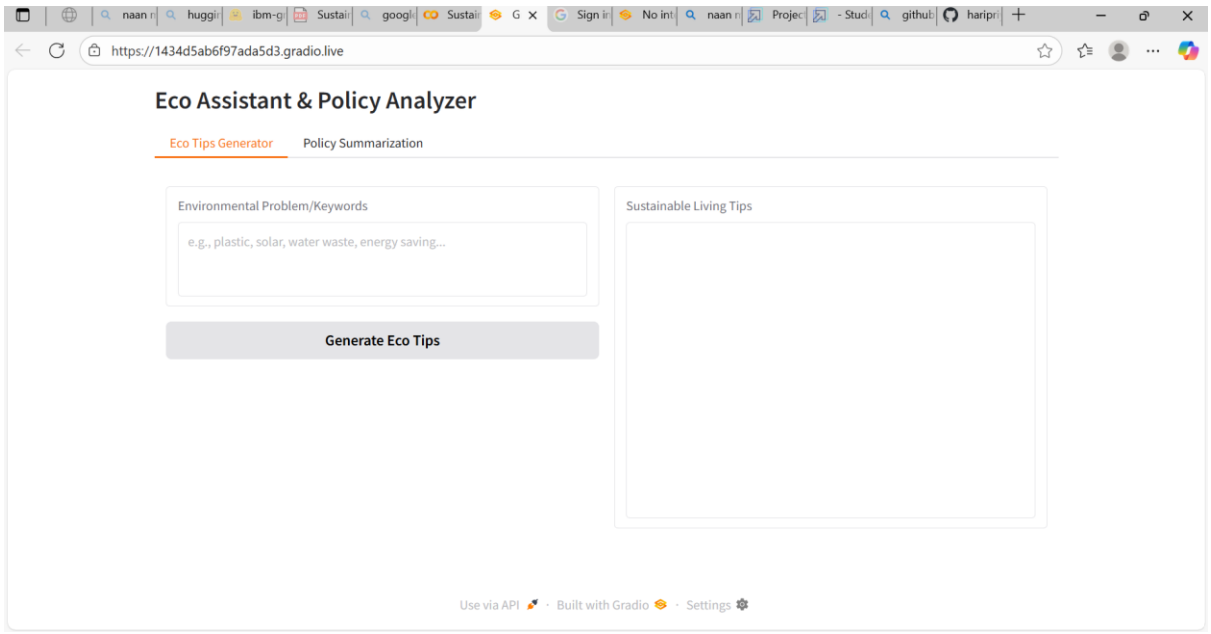

naan mudhalvan
hugging face - Se
ibm-granite/granit
Sustainable Smart
google collab - Se
Sustainable Smart
https://colab.research.google.com/drive/1O8WEv6slArSEdNjGoHucW0BlS3TAGnMC
Sustainable Smart City AI
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
Connect T4
return 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 solut
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{
Variables
Terminal

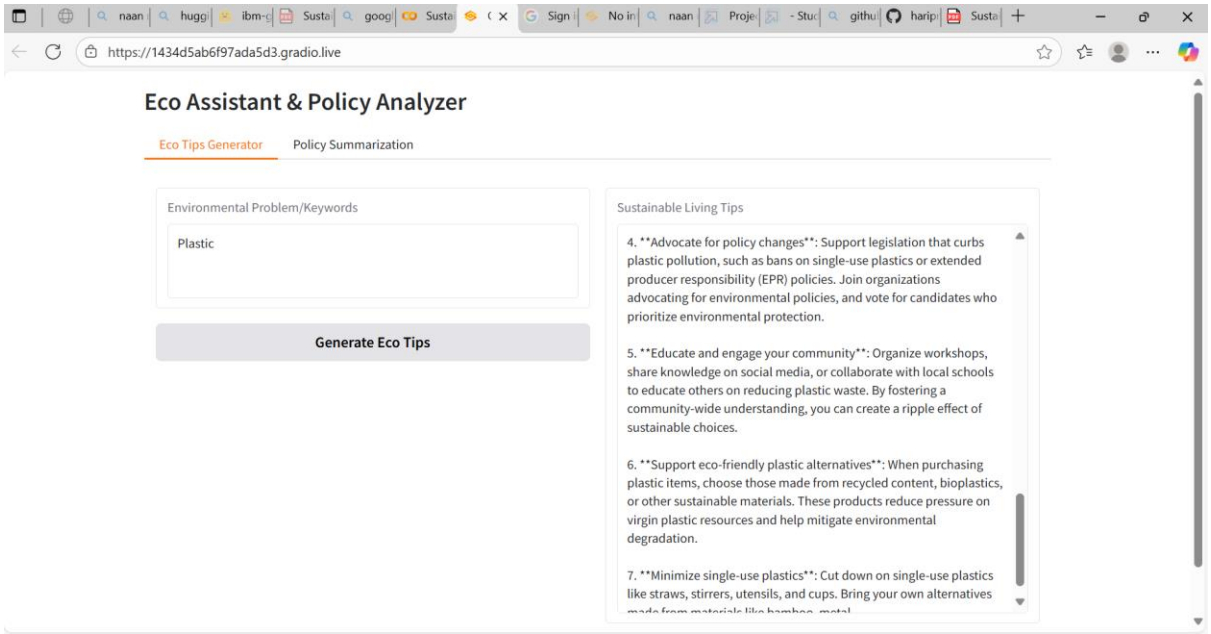

naan mudhalvan
hugging face - Sea
ibm-granite/granit
Sustainable Smart
google collab - Se
Sustainable Smart
https://colab.research.google.com/drive/1O8WEv6slArSEdNjGoHucW0BlS3TAGnMC
Sustainable Smart City AI
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
Connect T4
return 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=
words_input, outputs=tips_output)
with gr.TabItem("Policy Summarization"):
Variables
Terminal


naan mudhalvan
hugging face - Sea
ibm-granite/granit
Sustainable Smart
google collab - Se
Sustainable Smart
https://colab.research.google.com/drive/1O8WEv6slArSEdNjGoHucW0BlS3TAGnMC
Sustainable Smart City AI
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
Connect T4
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)
Variables
Terminal



AI-Powered Quiz generated Screen output.




https://1434d5ab6f97ada5d3.gradio.live
Eco Assistant & Policy Analyzer
Eco Tips Generator
Policy Summarization
Environmental Problem/Keywords
Plastic
Generate Eco Tips
Sustainable Living Tips
4. **Advocate for policy changes**: Support legislation that curbs plastic pollution, such as bans on single-use plastics or extended producer responsibility (EPR) policies. Join organizations advocating for environmental policies, and vote for candidates who prioritize environmental protection.
5. **Educate and engage your community**: Organize workshops, share knowledge on social media, or collaborate with local schools to educate others on reducing plastic waste. By fostering a community-wide understanding, you can create a ripple effect of sustainable choices.
6. **Support eco-friendly plastic alternatives**: When purchasing plastic items, choose those made from recycled content, bioplastics, or other sustainable materials. These products reduce pressure on virgin plastic resources and help mitigate environmental degradation.
7. **Minimize single-use plastics**: Cut down on single-use plastics like straws, stirrers, utensils, and cups. Bring your own alternatives