Sustainable Smart City Assistant Using IBM Granite LLM

1. Introduction

- **Project title** : Smart City AI Assistant
- **Team member** : A.Sivakandan
- **Team member** : R.Sridhar
- **Team member** : S.Sanjay
- **Team member** :S.Saravanan

2. Project overview

- **Purpose:**

 The purpose of this program is to provide an AI-powered eco-assistant and policy analyzer that helps users make environmentally conscious decisions and understand sustainability policies more effectively. It generates actionable eco-friendly living tips tailored to specific environmental problems, empowering individuals to adopt sustainable practices in daily life. Additionally, it allows users to

summarize lengthy policy documents (via PDF upload or text input), extracting key provisions and implications for easy comprehension. By combining eco-awareness guidance with policy intelligence, the program supports individuals, researchers, and policymakers in making informed decisions toward a greener future.

- **Fearture:**

1. Eco Tips Generator

Accepts environmental problem keywords (e.g., plastic, water waste, energy usage).
Generates practical, actionable, and sustainable living tips using AI.
Encourages eco-friendly habits in daily life.

2. Policy Summarization Tool

Upload PDF policy documents or paste text directly.
Extracts and summarizes key points, provisions, and implications.
Helps users quickly understand lengthy or complex documents.

3. PDF Text Extraction

Reads and processes text directly from uploaded PDF files.
Handles multi-page policy documents.

4. AI-Powered Response Generation

Uses IBM Granite AI model for natural language understanding.
Provides context-aware, concise, and user-friendly outputs.

5. User-Friendly Gradio Interface

Tab-based interface with two modules (Eco Tips & Policy Summarization).
Interactive input fields and clear, readable output sections.
One-click button actions for generating results.

6. Cross-Platform Accessibility

Runs locally or online with Gradio share links.
Supports both desktop and mobile use.

3. Architecture

1. Input Layer

User Input Options:
Text Input (keywords for eco tips / policy text).
PDF Upload (policy document).

2. Processing Layer

PDF Extraction Module
Uses PyPDF2 to extract text from uploaded PDF files.
Prompt Construction Module
Dynamically builds AI prompts for either:
Eco Tips generation.
Policy Summarization.

3. AI Model Layer

Tokenizer
Converts user input into tokens using Hugging Face AutoTokenizer.
LLM (Language Model)
IBM Granite-3.2-2B-Instruct model processes the prompt.
Generates natural language output (eco tips or summary).
Post-Processing
Cleans model output.
Removes prompt echoes and formats the response.

4. Output Layer

Eco Tips Module: Displays generated sustainable living suggestions.
Policy Summarization Module: Shows summarized key points & implications.
Error Handling: Returns clear error messages if PDF/text processing fails.

5. Interface Layer (Frontend)

Gradio Blocks Interface with Tabs:
Eco Tips Generator Tab.
Policy Summarization Tab.
Interactive Widgets:
Textbox, File Upload, Buttons, and Output Text Area.

6. Deployment Layer

Runs locally or in cloud environments (Google Colab, Jupyter, server).
Gradio's share=True option for temporary public links.

4. Setup Instructions

1. System Requirements

OS: Windows, macOS, or Linux
Python: 3.9 or later
Hardware:
CPU (works fine)
GPU (recommended for faster inference, supports CUDA)
Internet connection (required to download model + run Gradio app)

2. Install Required Packages

Open a terminal (or Google Colab cell) and run:
pip install torch transformers gradio -q
torch → Deep learning framework for running models.
transformers → Hugging Face library to load IBM Granite model.
gradio → To create the web-based interface.

5. Folder Structure

app.py → Entry point to run the program (contains Gradio UI).

config/ → Stores configs (model name, max tokens, temperature, etc.).

modules/ → Contains all modularized logic: PDF handling, AI responses, eco tips, policy summarization.

assets/ → Store sample input files like test PDFs.

tests/ → For unit and integration testing.

docs/ → Documentation (architecture, setup, features, future enhancements).

6. Running the Application

1.Interface Layer (Frontend)

Gradio Blocks Interface with Tabs:
Eco Tips Generator Tab.

Policy Summarization Tab.
Interactive Widgets:
Textbox, File Upload, Buttons, and Output Text Area.

2.AI Model Layer

Tokenizer
Converts user input into tokens using Hugging Face AutoTokenizer.
LLM (Language Model)
IBM Granite-3.2-2B-Instruct model processes the prompt.
Generates natural language output (eco tips or summary).
Post-Processing
Cleans model output.
Removes prompt echoes and formats the response.

7. API Documentation

The Eco Assistant & Policy Analyzer is an AI-powered application built using Gradio, Transformers, and PyTorch. It provides two main capabilities:

1. Eco Tips Generator – Produces actionable eco-friendly lifestyle suggestions.

2. Policy Summarization – Summarizes lengthy policy documents (via PDF or direct text).

The application is modular, with each function accessible either through the Gradio interface or programmatically via Python imports.

8. Authentication

1. API Key Authentication

Each user is given a unique secret key.
Users must provide this key to access the program.
The program validates the key before processing any request.
Simple to implement and suitable for small apps or internal use.

2. Token-Based Authentication (JWT)

Users log in with a username/password.
The system generates a temporary token (JWT) for session-based access.
Tokens expire after a certain time for added security.
Suitable for web deployment or when multiple users need secure access.

3. OAuth / SSO (Single Sign-On)

Users authenticate via a trusted provider (Google, Microsoft, etc.).
The program receives a secure token from the provider.

Ideal for enterprise or large-scale deployments.

9. User Interface

- Eco Tips Generator

Components:

1. API Key Input (Optional)
Textbox where users enter their API key if authentication is implemented.

2. Keywords Input
Textbox for entering environmental problems or keywords (e.g., “plastic, solar energy”).
Multi-line input to allow multiple keywords.

3. Generate Button
Button labeled “Generate Eco Tips”.
On click, it triggers AI to generate sustainability suggestions.

4. Output Display
Textbox or scrollable area showing the eco-friendly tips.
Multi-line display to handle several suggestions.

2: Treatment Plans

Input (Left Column):
Textbox → Medical Condition (e.g., “diabetes”).
Number Field → Age (default value: 30).
Dropdown → Gender (options: Male, Female, Other).
Textbox → Medical History (e.g., “Previous conditions, allergies, medications”).

A Button → “Generate Treatment Plan”.

Output (Right Column):
A large Textbox labeled “Personalized Treatment Plan”.
Displays AI-generated plan (medication suggestions + home remedies).

- Policy Summarization

Components:

1. API Key Input (Optional)
Textbox for authentication, if enabled.

2. PDF Upload
File upload widget allowing PDF files only.

3. Policy Text Input
Textbox where users can paste policy text directly.
Multi-line input to accommodate lengthy text.

4. Summarize Button
Button labeled “Summarize Policy”.
On click, it triggers the AI summarization module.

5. Output Display
Textbox or scrollable area showing the summarized policy key points.
Clear formatting to highlight main points and implications.

10. Testing

1. PDF Extraction (extract_text_from_pdf)
Test valid single-page PDFs.
Test multi-page PDFs.
Test corrupted or empty PDFs.
Verify text extraction correctness.

2. AI Response Generation (generate_response)
Test with short prompts.
Test with long prompts.
Test for edge cases (empty input, special characters).

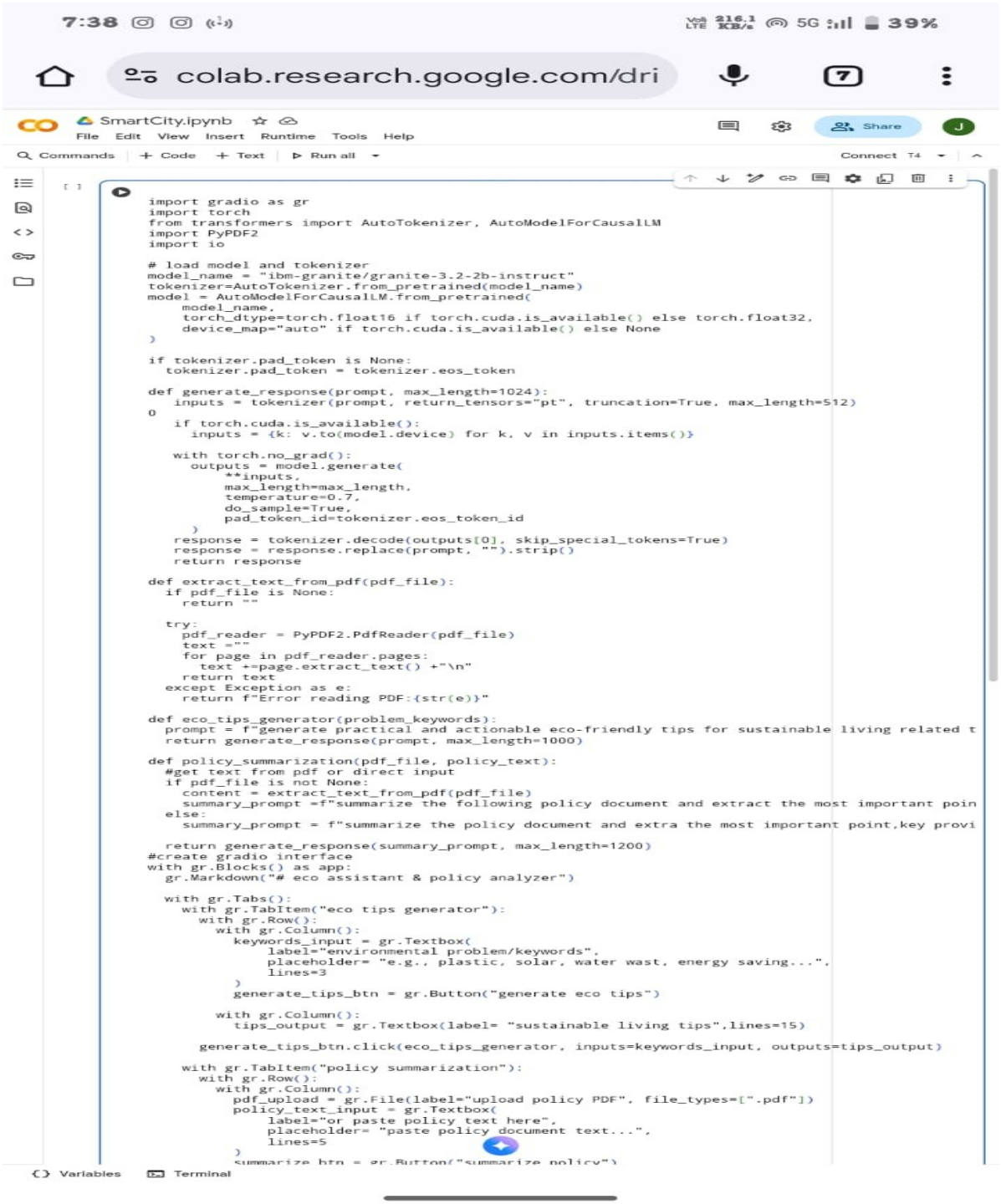
3. Eco Tips Generator (eco_tips_generator)
Test with single keyword.
Test with multiple keywords.
Test empty input handling.

4. Policy Summarization (policy_summarization)
Test with PDF input.
Test with text input.

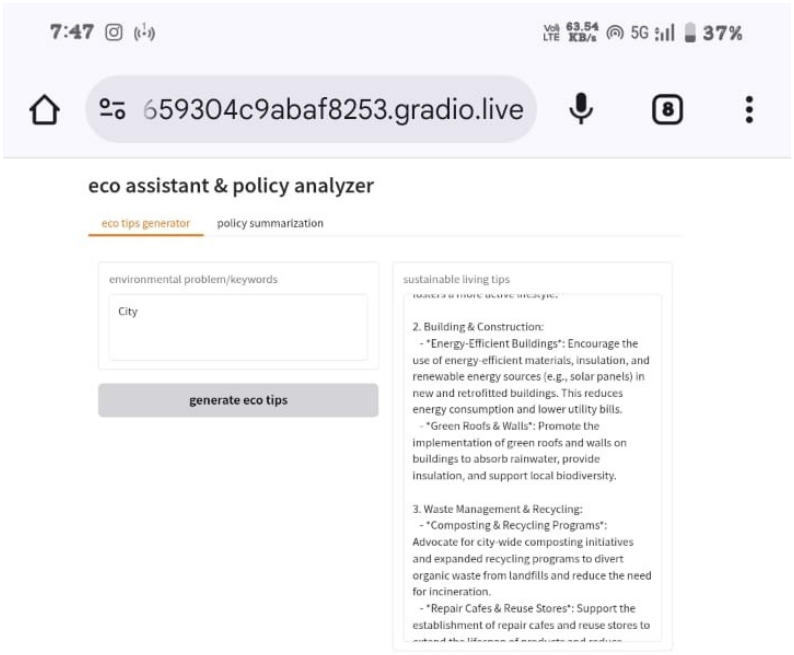
Test missing input scenario.

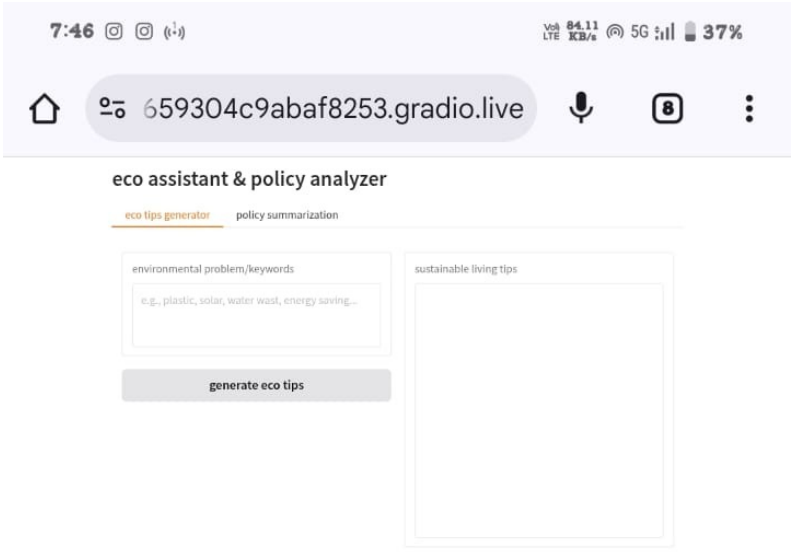
5. Authentication (if implemented)
Test valid API key.
Test invalid API key.
Test missing key.

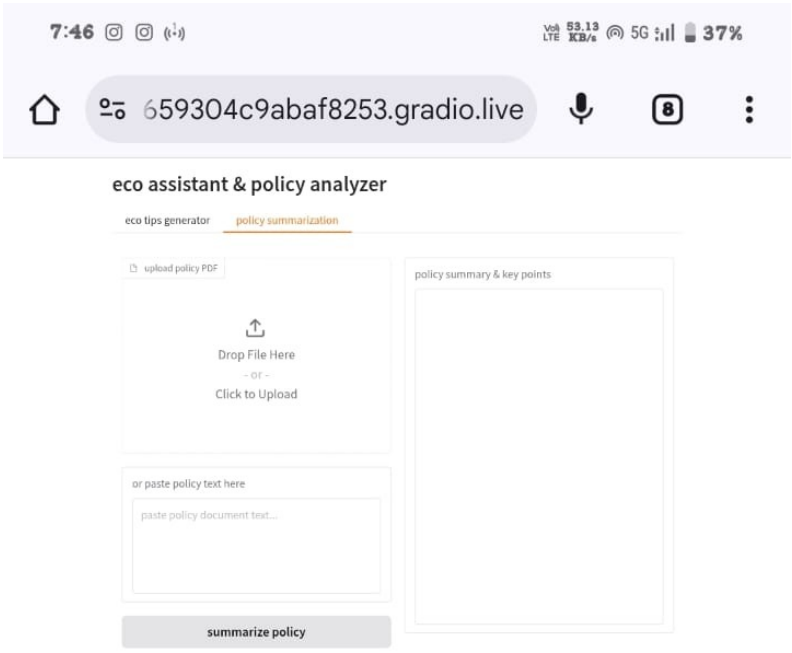
11. screen shots
Input


7:38
39%
colab.research.google.com/dri
SmartCity.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
Connect T4
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)
0
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 t
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 poin
else:
summary_prompt = f"summarize the policy document and extra the most important point,key provi
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 wast, 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")
Variables
Terminal

Output


7:47
37%
659304c9abaf8253.gradio.live
eco assistant & policy analyzer
eco tips generator
policy summarization
environmental problem/keywords
City
generate eco tips
sustainable living tips
2. Building & Construction:
- *Energy-Efficient Buildings*: Encourage the use of energy-efficient materials, insulation, and renewable energy sources (e.g., solar panels) in new and retrofitted buildings. This reduces energy consumption and lower utility bills.
- *Green Roofs & Walls*: Promote the implementation of green roofs and walls on buildings to absorb rainwater, provide insulation, and support local biodiversity.
3. Waste Management & Recycling:
- *Composting & Recycling Programs*: Advocate for city-wide composting initiatives and expanded recycling programs to divert organic waste from landfills and reduce the need for incineration.
- *Repair Cafes & Reuse Stores*: Support the establishment of repair cafes and reuse stores to


7:46
37%
659304c9abaf8253.gradio.live
eco assistant & policy analyzer
eco tips generator
policy summarization
environmental problem/keywords
e.g., plastic, solar, water wast, energy saving...
generate eco tips
sustainable living tips



12. Known Issues

- ◆ PDF Extraction Limitations
- ◆ AI Model Limitations
- ◆ Performance / Resource Issues
- ◆ Authentication Limitations
- ◆ User Interface Issues
- ◆ General Limitations

13. Future enhancement

- ◆ Advanced AI Features

- ◆ User Authentication & Management
- ◆ Enhanced PDF Handling
- ◆ User Interface Improvements
- ◆ Performance & Deployment
- ◆ Analytics & Reporting
- ◆ Integration with External Services