SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM

Project Documentation done by: Malar

1.Introduction:

- •Sustainable Smart City
- •Team Leader : MANOGARI V
- •Team member : MADHUMITHA G
- •Team member : MAHALAKSHIMI K
- •Team member : Malar Ranjani M

2.Project overview:

* Core Vision

To create an AI-powered digital assistant that serves as a single point of contact for citizens and city officials to access information, automate tasks, and make data-driven decisions that promote urban sustainability and improve quality of life.

* Key Objectives

· Empower Citizens: Make sustainable living easier and more accessible.
· Optimize Operations: Help city government manage resources more efficiently.
· Improve Decision-Making: Provide data-driven insights for urban planning.
· Increase Engagement: Foster a collaborative relationship between citizens and their city.

* Target Users & Functionality

A. For Citizens (Public Chatbot & Mobile App):
· Waste Management Guide ("Waste Wizard"):
· Answers questions on recycling, compost, and trash rules via chat (e.g., "Can I recycle this plastic wrapper?").
· Sends personalized collection day reminders and alerts for schedule changes.
· Sustainable Mobility Planner:
· Provides integrated, multi-modal travel routes (public transit, bike-share, walking).
· Locates EV charging stations and provides real-time availability and pricing.
· Calculates carbon footprint savings for chosen routes.
· Resource Conservation Helper:
· Analyzes anonymized utility (water, energy) usage to provide personalized conservation tips.
· Connects users to rebate programs for energy-efficient appliances.
· Civic Engagement Portal:
· Reports issues like potholes, broken streetlights, or illegal dumping via chat and image upload.
· Informs users about local community events, farmers' markets, and public meetings.

3.Architecture:

Core Concept: A secure, AI-powered assistant that uses city data to promote sustainability via a conversational interface.

*User Layer:
· Interfaces: Public Web Chat, Mobile App, City Official Dashboard.

*Orchestration Layer:
· Backend Server: Manages user requests, security, and conversation state.
· Key Task: Constructs intelligent prompts for the LLM.

*AI Core (IBM watsonx.ai Platform):
· IBM Granite LLM: The reasoning engine. Its strengths are:

· Code Generation: Excels at translating user requests into API calls and data queries.
· Enterprise Security: Deployed securely on IBM Cloud, ensuring data privacy and compliance.

*Action & Data Layer:
· Action Broker: Executes the API calls decided by the LLM (e.g., fetch transit data, check recycling rules).
· Data Ecosystem: Connects to city APIs (Transport, Waste, Energy IoT sensors) and external services (Maps).
· Vector Database (For Accuracy): Stores official city documents. Used to retrieve facts and ground the LLM's responses, preventing hallucinations.
*How It Works:
- A user asks a question (e.g., "How do I recycle electronics?").
- The backend sends the query + context to Granite.
- Granite decides if it can answer or needs data.
- The Action Broker calls the required API (e.g., waste management database).
- Granite synthesizes the data into a clear, natural language answer.
- The response is delivered to the user.

4.Setup Instruction:
PREREQUISITES
✓Governance & Planning
✓ Technical Prerequisites
INSTALLATION PROCESS
✓Set Up the IBM watsonx.ai Environment
✓Backend Application Setup
✓Data Layer Configuration
✓Deployment
✓Frontend Integration
✓Testing & Validation

5. Folder Structure:
- app.py - Main application file that:
- Initializes the Gradio interface
- Sets up the model and tokenizer
- Defines the application workflow and UI components

*requirements.txt - Ensures consistent environment setup by specifying exact package versions needed
* README.md - Documentation that explains:
.How to install and run the application
· What the application does
· How to use both features (Eco Tips and Policy Analysis)
* models/ - Optional directory to cache the pretrained model locally rather than downloading each time
*utils/ - Modularizes functionality for better code organization:
· pdf_processor.py handles all PDF-related operations
· model_handler.py manages model loading and text generation
*static/ - Contains assets that enhance the UI/UX:

· Custom CSS to style the Gradio interface
· Images for branding and visual appeal

* templates/ - For future expansion if converting to a web framework like Flask/FastAPI
* tests/ - Ensures code reliability through automated testing:
  · Verifies PDF text extraction works correctly
  · Tests that model generates appropriate responses.

6. Running the Application

1. Python (3.8 or higher): The most common language for these projects.
   · Download from python.org.
   · Verify installation: python --version or python3 --version
2. Pip (Python Package Manager): Usually comes with Python.
   · Verify: pip --version or pip3 --version
3. IBM Cloud Account & API Key:
   · Go to IBM Cloud and create a free account.
   · Create an API key for yourself (Search for "IBM Cloud API keys" in the console).
   · You need the Project ID for your Watsonx.ai service.
   · Go to your IBM Cloud Resource List, find your Watsonx.ai service, and copy its GUID (a long unique string). This is often used as the project_id.
4. The Application Code:
   · This is likely in a GitHub repository. You need to clone or download the code to your computer.
   · Example: git clone <repository-url>

7.API Documentation

•POST /api/chat-To ask questions and get informative, context-aware answer about sustainable urban living.

•POST /api/analyze/policy-To simulate and get a summary of the potential economic, environmental, and social impacts of a proposed city policy.

•POST /api/generate/report-To automatically generate reports (e.g., Annual Sustainability Report, Carbon Footprint Analysis) from structured data.

•POST /api/analyze/sentiment-To process large volumes of text feedback and summarize the main complaints, suggestions, and public sentiment.

•POST /api/optimize/:resource (e.g., /api/optimize/energy)-To get specific, actionable recommendations for optimizing a particular city resource.

8.Authentication

1. Purpose: Secure Role-Based Access
2. Primary Method: JWT (JSON Web Tokens)
3. Key API Endpoints
4. Role-Based Access Control (RBAC)
5. Environment Configuration
6. Integration with IBM Granite

9.User Interface

The UI transforms the complex AI and data capabilities into a simple, actionable, and engaging experience for everyone in the city.

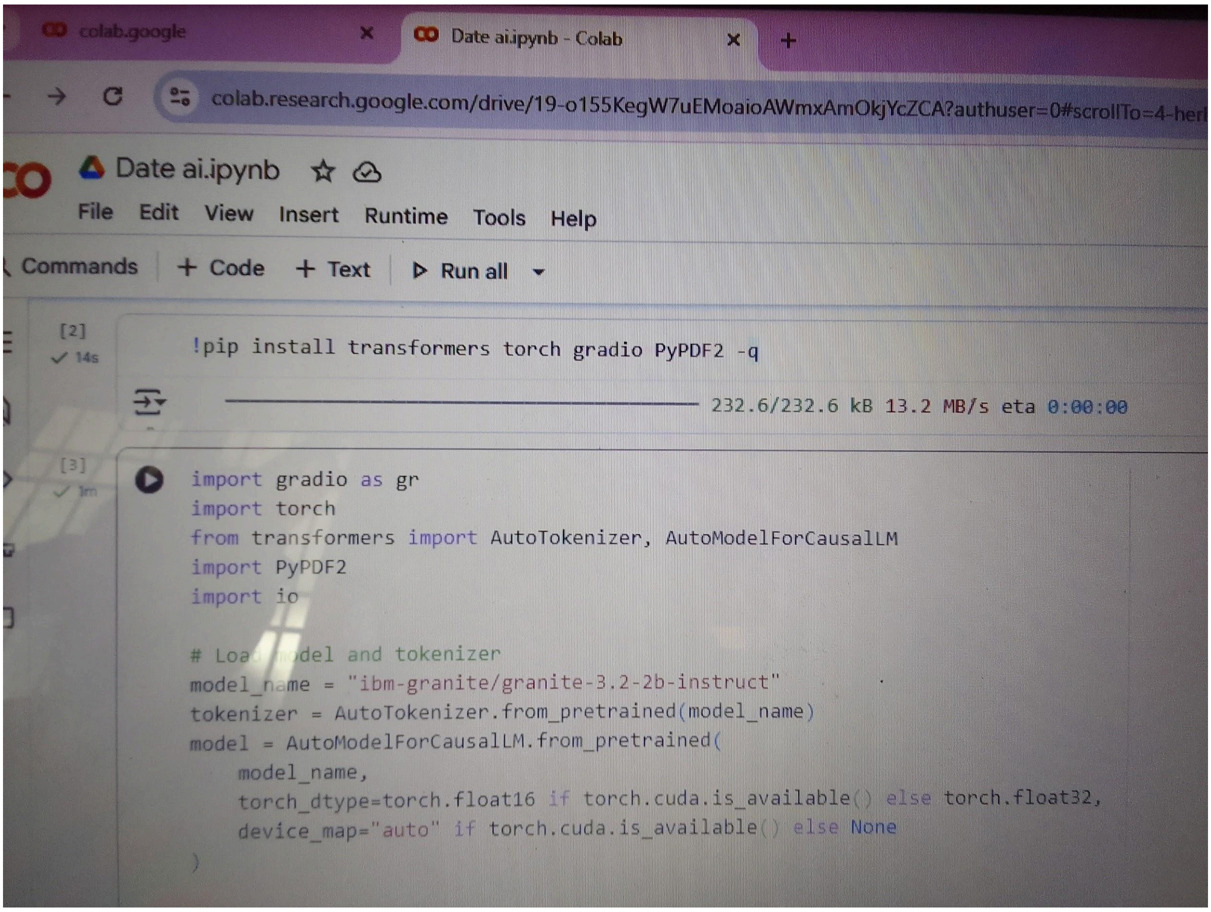
10.Testing

Testing was done in different phases:

Phase 1: Requirements Analysis
Phase 2: Data Collection & Validation
Phase 3: System Integration Testing
Phase 4: Functional Testing
Phase 5: Performance Testing
Phase 6: Security Testing
Phase 7: User Acceptance Testing (UAT)
Phase 8: Pilot Deployment Testing
Phase 9: Sustainability Impact Assessment
Phase 10: Regression & Maintenance testing
Phase 11: Compliance Testing
Phase 12: Disaster Recovery Testing

Screenshot:

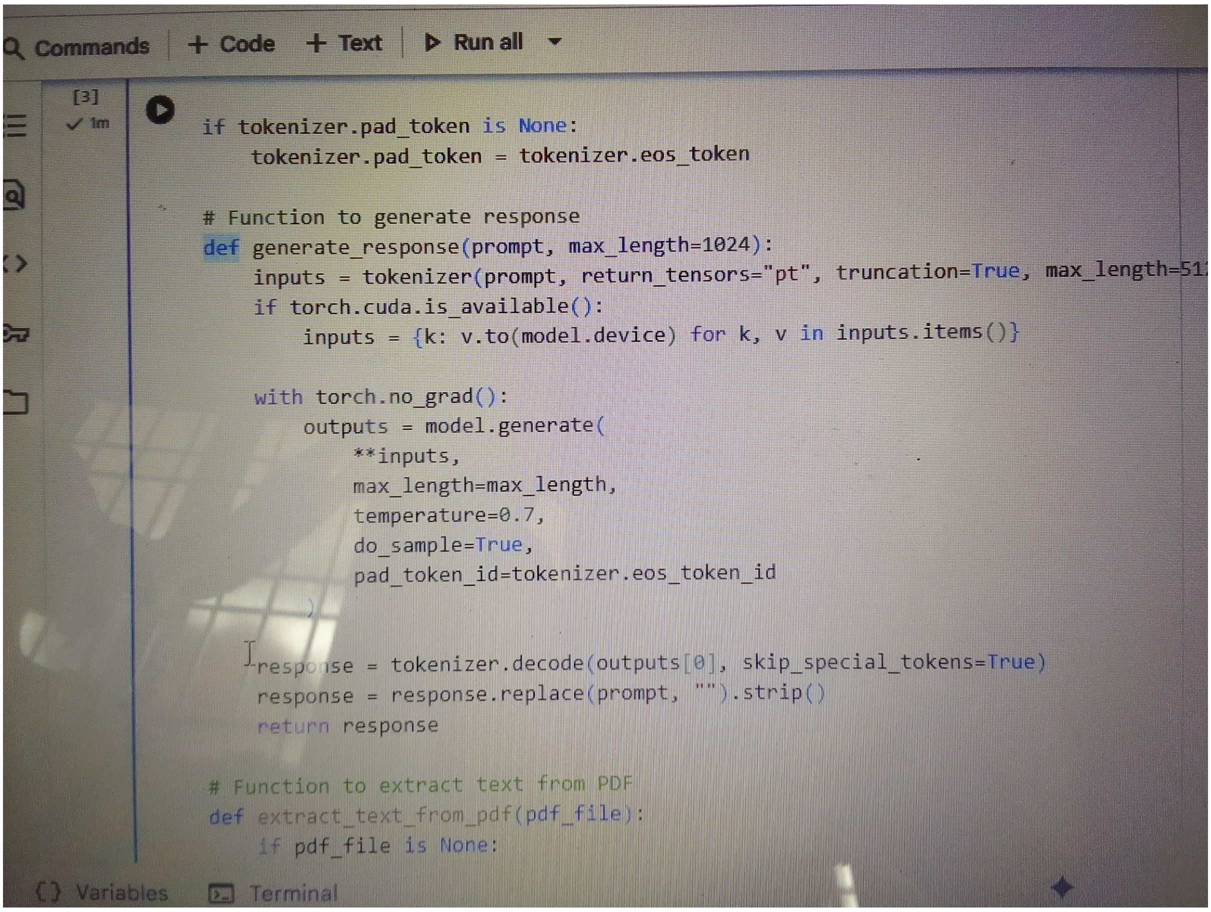
Program:



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

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 13.2 MB/s eta 0:00:00
```

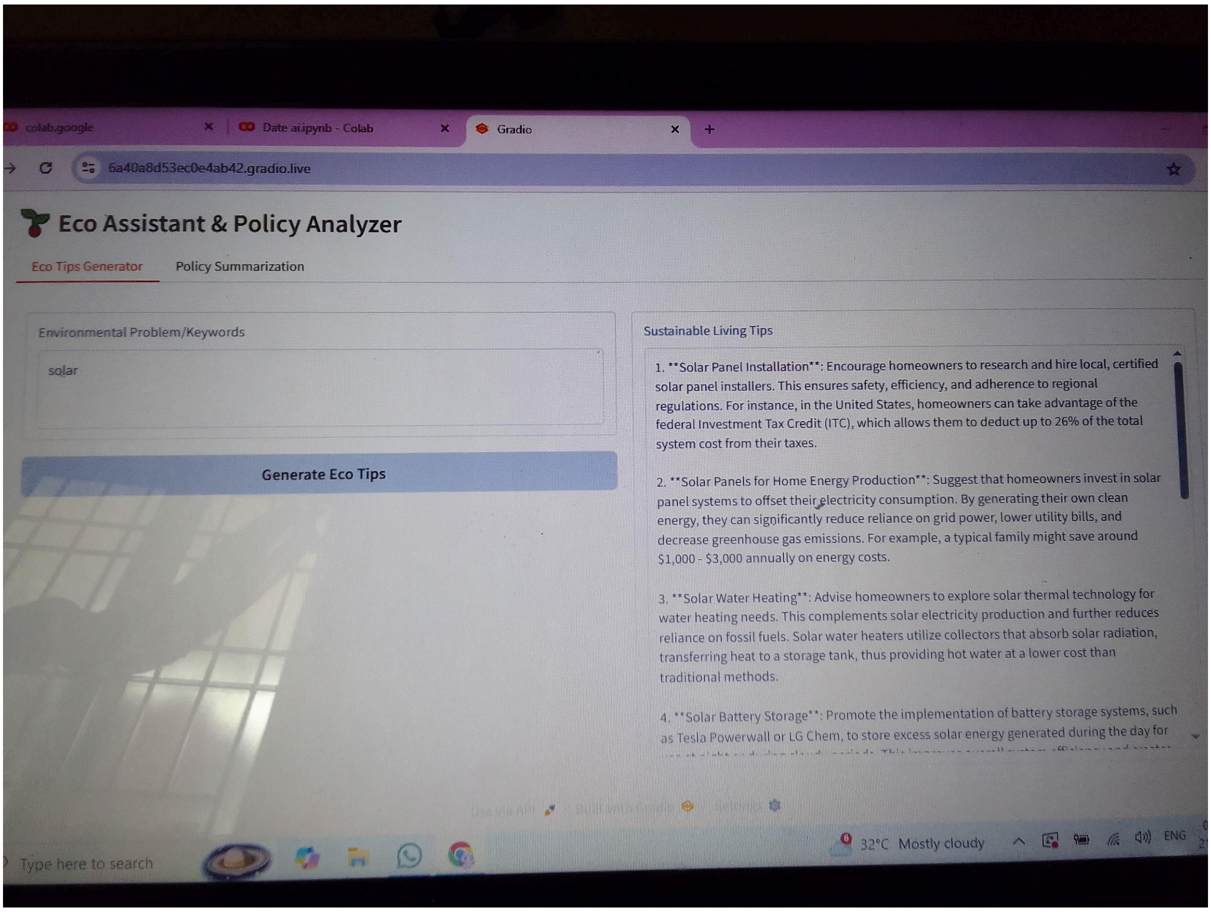
```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
import io

# 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

# Function to generate response
def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=51
    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

# Function to extract text from PDF
def extract_text_from_pdf(pdf_file):
    if pdf_file is None:
```

Output:

- Eco tips generator:



Eco Assistant & Policy Analyzer

Environmental Problem/Keywords

solar

Sustainable Living Tips

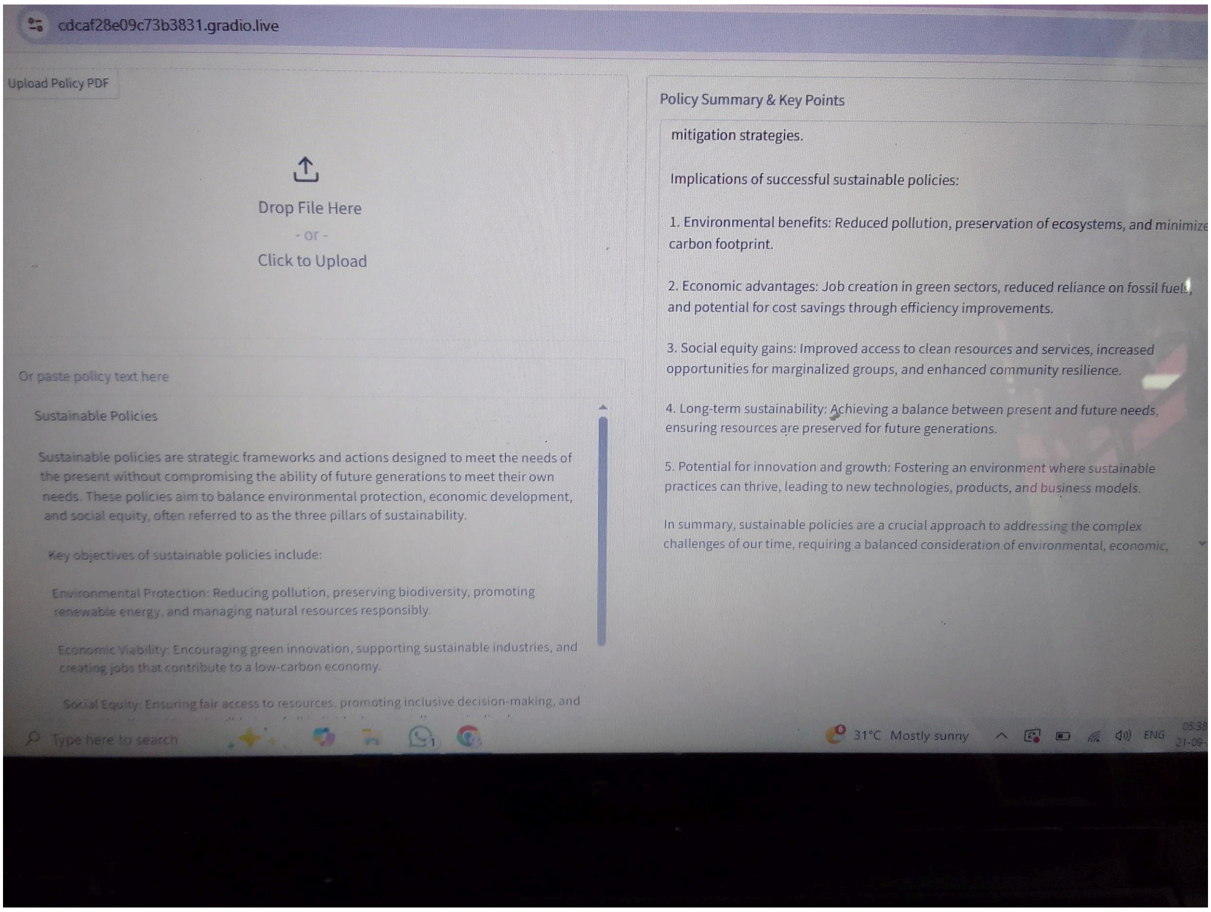
1. **Solar Panel Installation**: Encourage homeowners to research and hire local, certified solar panel installers. This ensures safety, efficiency, and adherence to regional regulations. For instance, in the United States, homeowners can take advantage of the federal Investment Tax Credit (ITC), which allows them to deduct up to 26% of the total system cost from their taxes.

2. **Solar Panels for Home Energy Production**: Suggest that homeowners invest in solar panel systems to offset their electricity consumption. By generating their own clean energy, they can significantly reduce reliance on grid power, lower utility bills, and decrease greenhouse gas emissions. For example, a typical family might save around $1,000 - $3,000 annually on energy costs.

3. **Solar Water Heating**: Advise homeowners to explore solar thermal technology for water heating needs. This complements solar electricity production and further reduces reliance on fossil fuels. Solar water heaters utilize collectors that absorb solar radiation, transferring heat to a storage tank, thus providing hot water at a lower cost than traditional methods.

4. **Solar Battery Storage**: Promote the implementation of battery storage systems, such as Tesla Powerwall or LG Chem, to store excess solar energy generated during the day for

- Policy summarisation:



Or paste policy text here

Sustainable Policies

Sustainable policies are strategic frameworks and actions designed to meet the needs of the present without compromising the ability of future generations to meet their own needs. These policies aim to balance environmental protection, economic development, and social equity, often referred to as the three pillars of sustainability.

Key objectives of sustainable policies include:

Environmental Protection: Reducing pollution, preserving biodiversity, promoting renewable energy, and managing natural resources responsibly.

Economic Viability: Encouraging green innovation, supporting sustainable industries, and creating jobs that contribute to a low-carbon economy.

Social Equity: Ensuring fair access to resources, promoting inclusive decision-making, and

Policy Summary & Key Points

mitigation strategies.

Implications of successful sustainable policies:

1. Environmental benefits: Reduced pollution, preservation of ecosystems, and minimize carbon footprint.

2. Economic advantages: Job creation in green sectors, reduced reliance on fossil fuels, and potential for cost savings through efficiency improvements.

3. Social equity gains: Improved access to clean resources and services, increased opportunities for marginalized groups, and enhanced community resilience.

4. Long-term sustainability: Achieving a balance between present and future needs, ensuring resources are preserved for future generations.

5. Potential for innovation and growth: Fostering an environment where sustainable practices can thrive, leading to new technologies, products, and business models.

In summary, sustainable policies are a crucial approach to addressing the complex challenges of our time, requiring a balanced consideration of environmental, economic,