Sustainable Smartcity Assistant Using IBM Granite

1. Introduction

Project Title:Sustainable Smartcity Assistant Using IBM Granite

Team Members:
Team member 1: Methelas.H.V
Team member 2: Yugesh.J
Team member 3: Mohanraj.R
Team member 4: Leviyan Samuvel.S

2. Project Overview

Purpose:
The purpose of the Sustainable Smartcity Assistant is to provide AI-driven support for urban sustainability management. The system monitors energy usage, transportation, waste management, and air quality, while delivering actionable insights for both citizens and administrators.
Features:
- Conversational Interface: Natural language queries about city services.
- Energy Monitoring: AI-based energy consumption analysis and optimization tips.
- Transport Optimization: Real-time route and traffic suggestions.
- Waste Management: Alerts for collection schedules and recycling tips.
- Air Quality & Environment: Pollution level tracking and eco-recommendations.
- Citizen Services: Guidance on government schemes, utilities, and emergencies.

3. Architecture

Frontend (Gradio): Tab-based dashboard with sections for Energy, Transport, Waste, and Environment.
Backend (Python + Transformers): Processes user queries, integrates IoT data feeds, and generates AI responses using IBM Granite LLM.
LLM Integration (IBM Granite – Hugging Face Model): Handles natural language understanding and generates sustainability recommendations.
Deployment: Can be launched locally or hosted on a cloud server for public access.

4. Setup Instructions

Prerequisites:Python3.9+,pip,Internet connection, optional GPU.
Installation:
- Clone the repository.
- Install dependencies: pip install gradio torch transformers gtts
- Run app.py and open local/public URL.
The interface allows users to query energy, transport, waste, or air quality modules.

5. Procedure

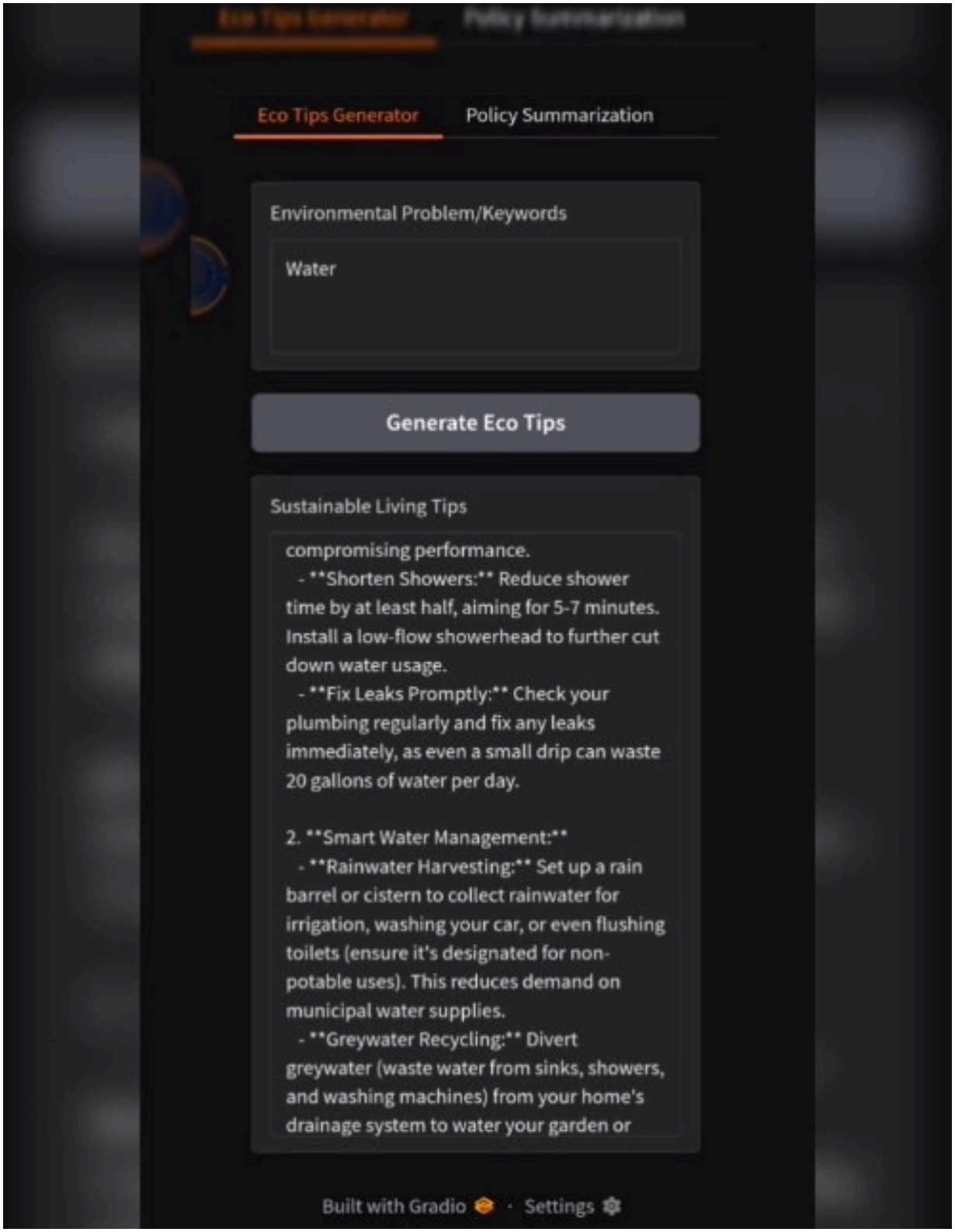
Step 1: User opens the Smartcity Assistant dashboard.
Step 2: Selects a module (Energy, Transport, Waste, Environment).
Step 3: Inputs a query (e.g., 'What is today's air quality index?' or 'Suggest eco-friendly transport routes').
Step 4: The backend processes the input using IBM Granite LLM and IoT datasets.
Step 5: Assistant generates insights and displays them as text and optional audio.
Step 6: Users receive actionable recommendations (e.g., 'Use Metro today, traffic congestion detected on highways').

6.coding

```
    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 ex
tract 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)
```

7. Output

- Textual responses with sustainability recommendations. - Dashboards showing energy usage trends, transport congestion maps, waste collection schedules, and pollution levels. - Voice-enabled suggestions for accessibility. - Example: User asks about air quality → Output: 'AQI = 85 (Moderate). Suggested actions: avoid outdoor exercise, use public transport.'



7. Future Enhancements

- Mobile app integration for citizens.
- AI-driven energy-saving recommendations for households.
- Integration with smart meters, traffic sensors, and weather APIs.
- Predictive analytics for waste reduction and carbon footprint tracking.
- Role-based admin and citizen login system.