**Sustainable Smart City Assistant using IBM Granite**

**Category:** Cloud Application Development  
**Skills Required:** Python, IBM Granite / Hugging Face Transformers, Gradio, PyTorch/Accelerate, Scikit-Learn, Pandas, NumPy

**Project Description**

The Sustainable Smart City Assistant leverages **IBM Granite Instruct** models to help city stakeholders—from policy teams to citizens—make faster, greener, and data-informed decisions. The platform provides:

* **Policy Summarization:** Convert lengthy municipal policies into concise, plain-language briefs.
* **Citizen Feedback Intake:** Capture and acknowledge citizen issues for downstream triage.
* **KPI Forecasting:** Forecast sustainability KPIs (e.g., waste diversion, EV uptake) from historical data.
* **Eco Tips Generator:** On-demand, actionable micro-interventions for households and businesses.
* **Anomaly Detection:** Flag outliers in sensor or operational streams (e.g., water leaks, energy spikes).
* **Sustainability Q&A Chat:** Granite-powered answers to climate and urban-sustainability questions.
* **Carbon Footprint Calculator:** Quick daily CO₂-equivalent estimation for transport, power, and waste.

Built with **Gradio** for a clean, multi-tab UI and powered by **IBM Granite** via transformers, the app is easy to run in Colab or deploy in the cloud. Security guidance and responsible-use guardrails are provided below.

**Scenarios**

1. **Rapid Policy Briefing**

* **Action:** Urban planner uploads a draft solid-waste policy (TXT) or pastes the text.
* **Outcome:** A readable summary highlighting goals, compliance requirements, and proposed timelines.

1. **Citizen Issue Logging**

* **Action:** Resident reports “overflowing public bins near Ward 14, evenings.”
* **Outcome:** Instant acknowledgement with category tagging (e.g., Waste → Collection → Overflow).

1. **City KPI Forecast**

* **Action: Analyst uploads a CSV with year,value for recycling rate.**
* **Outcome:** Model fits a baseline regression and predicts next year’s value for planning.

1. **Household Eco Nudges**

* **Action:** User enters a keyword like “plastic,” “solar,” or “water.”
* **Outcome:** Three actionable tips (behavioral + low-cost interventions).

1. **Sensor Anomaly Alert**

* **Action:** Operations team uploads value readings from water-pressure sensors.
* **Outcome:** Outlier windows are flagged (>2σ), enabling fast field inspection.

1. **Sustainability Q&A**

* **Action:** “How can we reduce peak-hour grid load in summer?”
* **Outcome:** Granite returns practical strategies (demand response, TOU tariffs, cool roofs).

1. **Personal Carbon Snapshot**

* **Action:** User provides daily commute km, electricity kWh, and waste kg.
* **Outcome:** Estimated CO₂ breakdown and total—useful for awareness campaigns.

**Technical Architecture**

**Prerequisites**

* Python 3.10+
* Libraries: gradio, transformers, torch, accelerate, scikit-learn, pandas, numpy
* Hardware: GPU recommended (≥8 GB VRAM) for faster Granite inference; CPU works with slower generation
* Internet (first model pull)

**Model & Libraries**

* **LLM:** ibm-granite/granite-3.3-2b-instruct (via Hugging Face transformers pipeline)
* **Classical ML:** LinearRegression for baseline KPI forecasting
* **Data:** pandas/numpy for transforms; simple σ-rule for anomaly detection

**System Design**  
Input (text / CSV) → Granite inference or analytics → Formatting → Gradio UI (Tabs) → Optional export

**Project Structure**

**smart-city/**

**app.ipynb                 # Colab/Notebook version (provided)**

**app.py                    # (optional) script form**

**data/                     # sample CSVs (optional)**

**README.md**

**Core Functionalities (mapped to your code)**

1. **Policy Summarization**
   * Reads .txt or textbox; prompts Granite: “Summarize in simple terms.”
   * Output: concise summary for planners and council briefings.
2. **Citizen Feedback Intake**
   * Textbox + acknowledgement.
   * (Extension-ready): auto-tag by category/ward, create triage tickets.
3. **KPI Forecasting**
   * CSV with two columns: year,value (ints/floats).
   * Fits LinearRegression on historical trend; predicts year+1.
4. **Eco Tips Generator**
   * Prompted tips (3 bullets) for a sustainability keyword.
5. **Anomaly Detection**
   * CSV must include column value.
   * Flags points where |value − μ| > 2σ; returns a compact table.
6. **Sustainability Q&A Chat**
   * Single-turn Q&A with Granite; temperature tuned for helpfulness.
7. **Carbon Footprint Calculator**
   * Simple factors (per day):
     + Transport ≈ **0.12 kg CO₂ / km**
     + Electricity ≈ **0.82 kg CO₂ / kWh**
     + Waste ≈ **0.50 kg CO₂ / kg**
   * Returns a transparent breakdown + total.

**Data Handling & Formats**

* **Policy Summarizer:** .txt or pasted text; no storage by default.
* **KPI Forecasting:** CSV format:  
   year,value

2018,24.5

2019,26.0

...

* **Anomaly Detection:** CSV format (minimum):  
   value

42

45

101

...

* **Session:** Ephemeral; for production add DB or object storage with consent.

**Responsible AI & Safety**

* **Scope clarity:** Assistant provides guidance, **not** legal or engineering approval.
* **Data privacy:** Avoid PII; anonymize ward/location as needed; no raw citizen IDs in logs.
* **Bias & hallucinations:** Keep prompts explicit; consider retrieval (future work) to ground answers on city docs; add disclaimers in UI.
* **Energy use:** Prefer small Granite models for routine tasks; batch generations; cache summaries.

**How to Run (Google Colab)**

1. **Open a new Colab notebook** and set **Runtime → Change runtime type → GPU (T4/A100 if available)**.
2. **Install deps** (your code already does this):  
   **!pip install gradio transformers torch scikit-learn pandas numpy accelerate**
3. **Paste your full code cell** (as provided) into Colab.
4. **Run all cells.** First model download may take a few minutes.
5. **App link:** After app.launch(share=True), Colab prints a public URL. Click to open the multi-tab UI.
6. **Test quickly:**
   * *Policy Summarization:* paste a short paragraph.
   * *KPI Forecasting:* upload a tiny CSV (year,value).
   * *Anomaly Detection:* upload a value column CSV.
   * *Eco Tips / Chat / Footprint:* try the defaults.

**Deployment (Optional)**

* **Containerize:** Minimal Dockerfile with Python base + requirements.
* **Hosting:** IBM Cloud Code Engine / Cloud Run / any GPU host.
* **Networking:** Terminate TLS; restrict admin endpoints.
* **Secrets:** Environment variables for any tokens; do not hardcode.
* **Monitoring:** Basic logs + request metrics; capture latency and token usage.

**KPIs & Evaluation**

* **User Engagement:** # feedback submissions, completion rate.
* **Operational Impact:** # anomalies caught before escalation.
* **Policy Efficiency:** average summary length vs. comprehension survey.
* **Greener Behavior:** click-through/adoption on eco-tips; estimated CO₂ reduced.
* **Model Quality:** human evaluation of Q&A helpfulness (Likert), factuality checks.

**Roadmap / Future Work**

* **RAG over City Docs:** Ground Q&A and summarization on verified PDFs/portals.
* **Auto-Triage:** Route citizen feedback to the right department with confidence scores.
* **Time-Series Models:** Move from linear regression to Prophet/LSTM for KPIs.
* **Real-time Streams:** Kafka/MQ ingestion for sensors; live anomaly dashboards.
* **Localization:** Multi-language UX for residents.
* **Accessibility:** WCAG-compliant UI, voice I/O, SMS bot for low-connectivity users.

**Quick Reference (Modules ↔ Tabs)**

* Policy Summarization → policy\_summarizer\_v2
* Citizen Feedback → citizen\_feedback
* KPI Forecasting → kpi\_forecasting
* Eco Tips → eco\_tips
* Anomaly Detection → detect\_anomaly
* Chat Assistant → chat\_assistant
* Carbon Footprint → carbon\_footprint