Sustainable Smart City Assistant

**Team Information:**

**Team ID:** LTVIP2025TMID60337

**Team Size:** 4

**Team Leader:** Ganjapu Nanda Gopala Krishna Chari

**Team member:** Konda Bhargavi

**Team member:** Chinagola Venkata Padmavathi

**Team member:** Anusha

### Platform: IBM Cloud (Smart City Assistant)

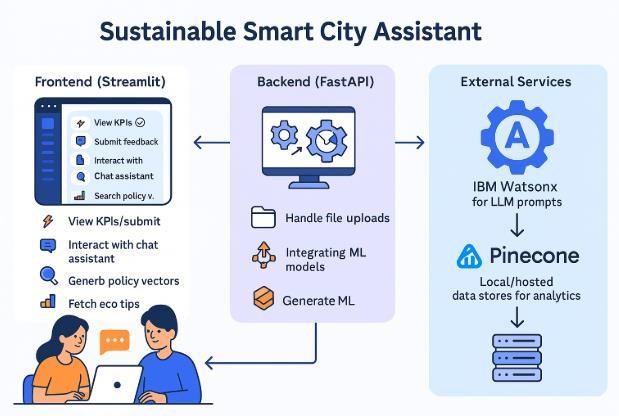
**Skills Required:**

* Python
* IBM Cloud

**Project Description: -**

The Sustainable Smart City Assistant is an AI-powered platform that leverages IBM Watsonx's Granite LLM and modern data pipelines to support urban sustainability, governance, and citizen engagement. It integrates several modules like City Health Dashboard, Citizen Feedback, Document Summarization, Eco-Advice, Anomaly Detection, KPI forecasting and Chat Assistant through a modular FastAPI backend and a Streamlit.

**Technical Architecture**



**Pre-requisites: -**

Duration: 1 Hrs

Skill Tags:

1. Streamlit Framework Knowledge
2. IBM Watson Machine Learning
3. Python Programming Proficiency
4. Data Visualization Libraries
5. Version Control with Git

**Project Workflow: -**Project Flow for Sustainable Smart city Assistant

1. **User Input:**

Users interact with the Streamlit frontend dashboard, where they can:

* Submit textual prompts (for chat or policy summaries).
* Upload policy documents (.txt, .csv) for summarization and vector search.
* Choose a city to view real-time KPIs (water usage, air quality, energy).
* Submit citizen feedback (name, category, message).
* Ask sustainability queries via chat interface.
* Search for eco-friendly tips by entering a topic keyword.
* UI Components Involved:
* smart\_dashboard.py, feedback\_form.py, chat\_assistant.py, eco\_tips.py, summary\_card.py

**2.Backend Processing (FastAPI): -**

* Each input request is sent to corresponding FastAPI endpoints, where:
* Feedback is stored and categorized through feedback\_router.py.
* KPI .csv files are forecasted using internal ML models in kpi\_file\_forecaster.py.
* Text prompts (from chat, summarizer, eco tips) are sent to IBM Granite LLM using the granite\_llm.py service.
* Anomaly detection is applied to uploaded datasets using statistical checks.

**Key Backend Components: -**

* vector\_router.py, chat\_router.py, kpi\_upload\_router.py, granite\_llm.py, pinecone\_client.py

**3.AI Response Generation: -**

* The Watsonx Granite LLM processes chat queries, summaries, eco tips, and generates human-like natural language responses.
* ML models forecast future KPIs or detect anomalies in uploaded files.
* Pinecone retrieves the most relevant policy document chunks using semantic search powered by vector similarity.
* **Output Formats:** JSON objects containing text summaries, search results, KPIs, anomaly alerts.

**4.Frontend Display: -**

The Streamlit frontend dynamically renders:

* KPI data in visually enhanced cards(summary\_card.py).
* AI-generated responses (chat, eco tips) directly in user input sections.
* Policy search results in readable formats.
* Submission success or errors through toast messages (e.g., feedback success).
* Frontend Enhancements Done: Rounded input cards, Gradient background, Icon-rich sidebar, Themed buttons and layout improvements.

**5.User Interaction: -**

**Users are able to: -**

* + Switch cities and compare urban KPIs dynamically.
  + Ask follow-up queries in the chat assistant.
  + Generate policy summaries and sustainability reports.
  + Continuously explore eco tips with varied topics.
  + Interact with updated dashboard metrics in real time.

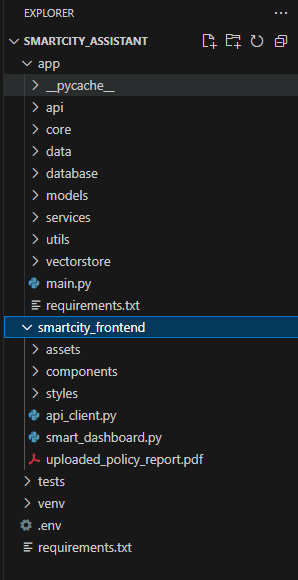
**Real-time interaction enabled by: -**

* + FastAPI + Streamlit two-way binding with updated backend JSON responses

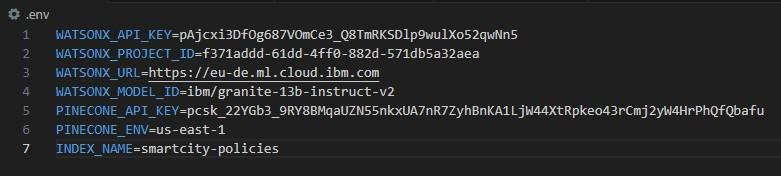
**Development Flow:**

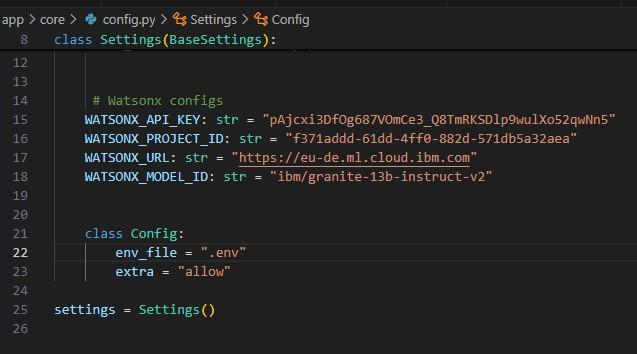
**Phase 1 – Project Initialization**

Modular Folder Structure Defined: Created separate folders for app/api, services, vectorstore,

core, frontend/components, and utils for organized and scalable development.

**Environment Setup:**

* .env file created with keys for Pinecone and Watsonx. config.py loads environment variables securely using pydantic.
* .env file

**Config.py file**

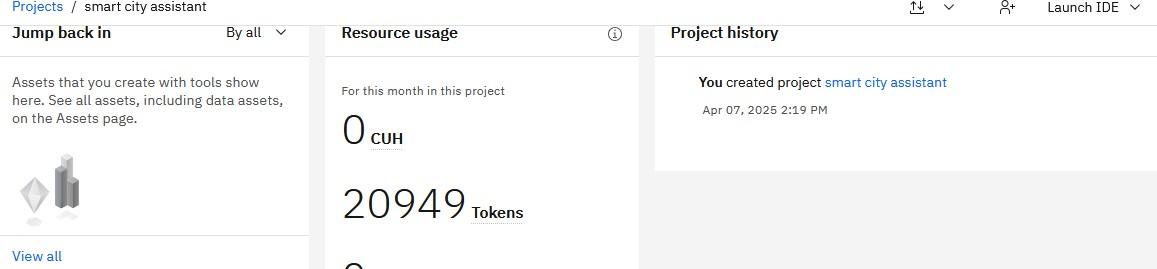
**Pinecone Initialization:**

pinecone\_client.py written to initialize the Pinecone vector index (smartcity-policies). Ensured creation with correct dimension=384 matching embedding model.

**Phase 2 – IBM Watsonx Integration**

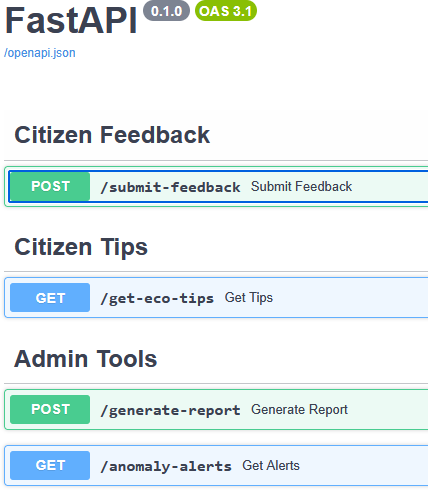
Watsonx Key & Model Configuration: Set up .env with:

WATSONX\_API\_KEY, PROJECT\_ID, MODEL\_ID



**Endpoint Testing:**

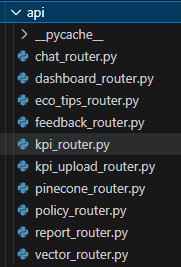
Validated /chat, /policy/summarize, and /get-eco-tips FastAPI routes using Swagger UI.



**Phase 3 – Backend API Routers API Routes Implemented:**

**Developed modular routers:**

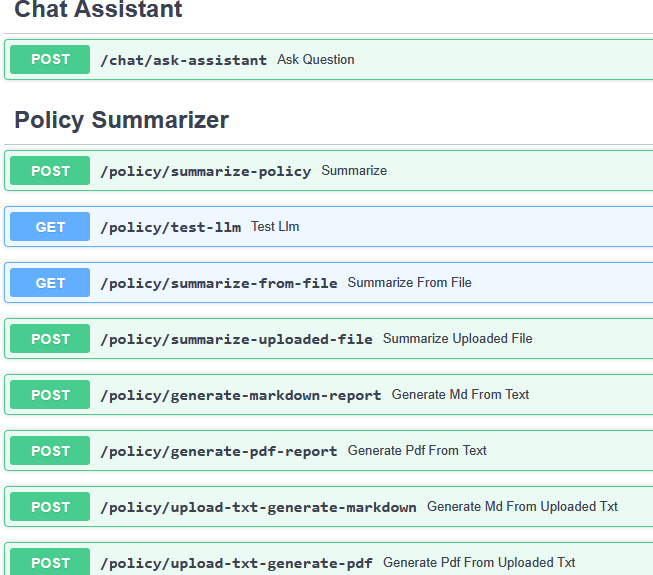
* chat\_router.py
* feedback\_router.py
* eco\_tips\_router.py
* kpi\_upload\_router.py
* anomaly\_checker.py
* vector\_router.py, etc.



**Testing & Validation:**

**Each route tested for:**

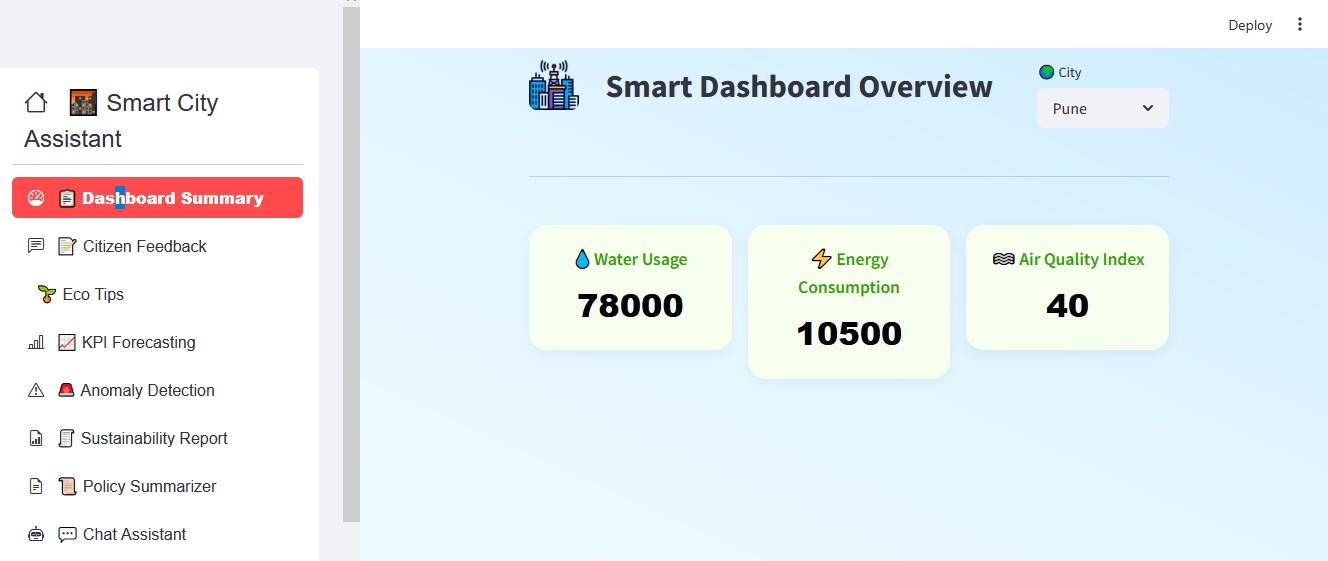
* JSON payload correctness
* File upload parsing
* Error handling & logging
* Swagger auto-documentation generation



**Phase 4 – Frontend UI Design**

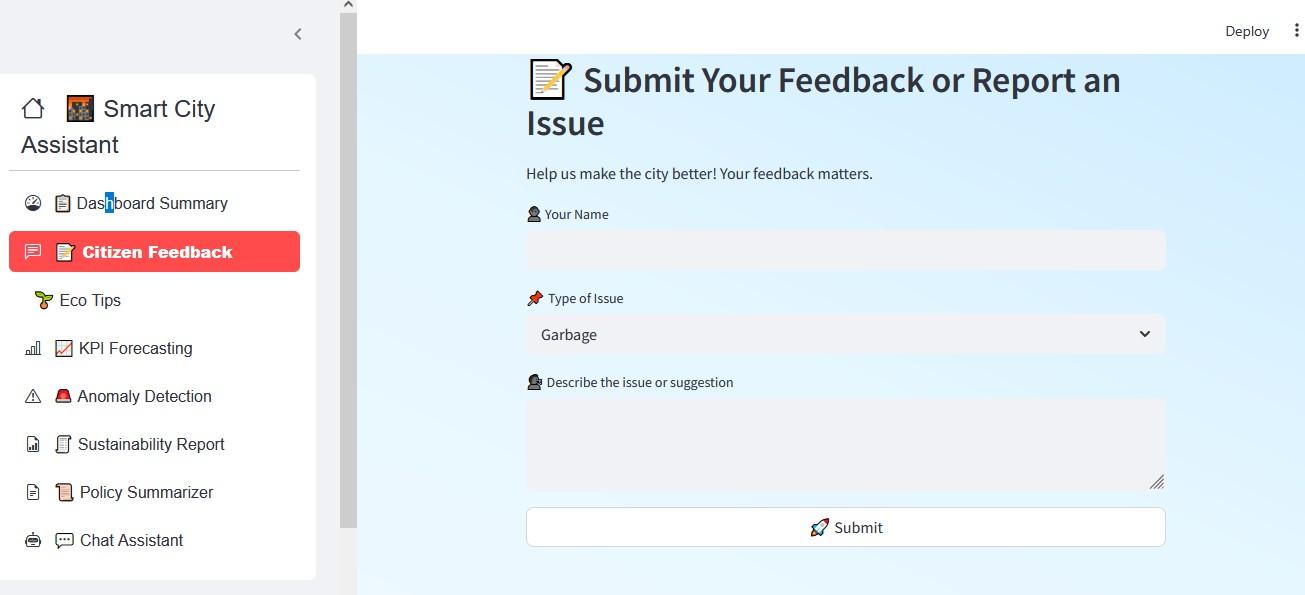
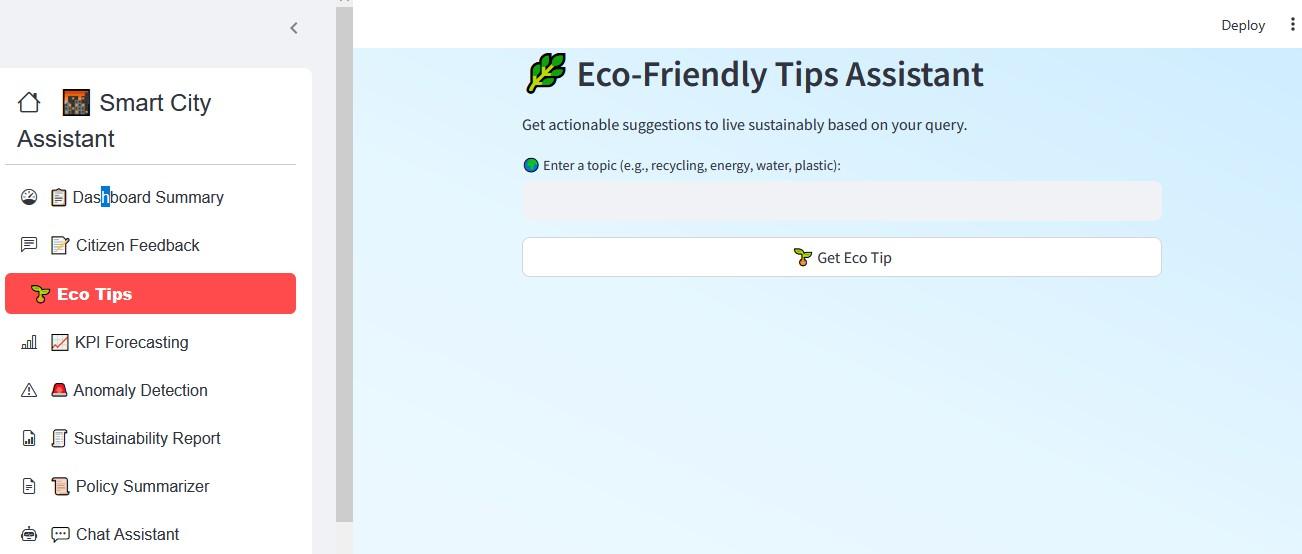
**Streamlit UI Structure Implemented:**

Created central file smart\_dashboard.py with conditional rendering for each module using sidebar navigation.



**Component Development:**

Developed reusable Streamlit components: summary\_card.py – Beautiful KPI cards chat\_assistant.py – Text prompt and AI reply feedback\_form.py, eco\_tips.py, report\_generator.py, etc.



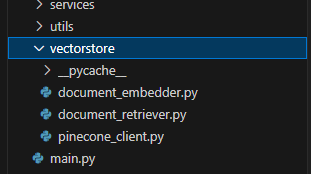
**UI Enhancements Done:**

Gradient backgrounds

Icon-rich sidebar using streamlit-option-menu Rounded buttons, font styles, padding fixes

Phase 5 – Pinecone & Document Embedding

Embedding Logic Built:

Created document\_embedder.py and document\_retriever.py using sentence-transformers.

**Phase 6 – Report Generation & Deployment**

**Granite LLM Report Generator:**

report\_generator.py takes city name and KPI data, generates detailed city sustainability report using Granite LLM prompts.

**Markdown & PDF Support:**

Output formatted to text block for copy/paste or PDF download (optional).

**End-to-End Integration Testing:**

**Final dashboard tested on all 8 features:** KPI dashboard, feedback form, policy summarization, eco tips, chat, anomaly check, vector search, report generation

**Milestone 1: Requirements Specification**

**Objective:** Establish the foundational libraries and packages for both frontend and backend to ensure reproducibility and easy environment setup.

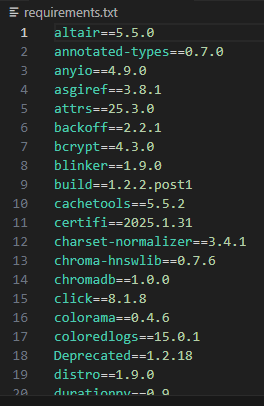
**Create requirements.txt**

**Define the required libraries:**

* streamlit: For building interactive dashboard interfaces
* fastapi: Backend API framework for rapid development
* uvicorn: ASGI server to run FastAPI
* requests: For API communication from frontend
* python-dotenv: Manage environment variables
* sentence-transformers: Text embedding model
* pydantic-settings: Handle configuration management
* pinecone-client: For semantic document search
* scikit-learn, pandas: For anomaly detection and forecasting
* matplotlib: For report visualizations

### **Install all dependencies**

bash

pip install -r requirements.txt

**Milestone 2: Environment Initialization & API Key Setup**

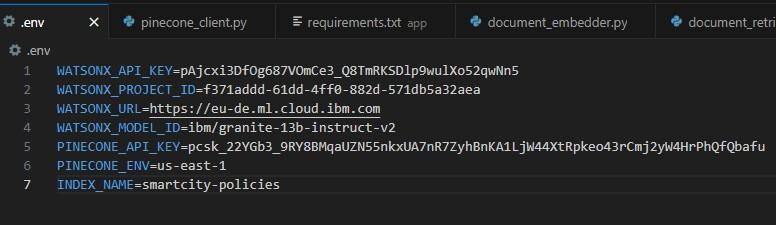
**Objective:** Configure and secure external service credentials (Watsonx & Pinecone).

**Generate API Keys**

* Watsonx Granite credentials from IBM Cloud dashboard
* Pinecone API key and environment from [https://app.pinecone.io](https://app.pinecone.io/)

### **Define .env File**

## **Create a .env file to hold:** WATSONX\_API\_KEY=your\_ibm\_api\_key WATSONX\_PROJECT\_ID=your\_project\_id WATSONX\_URL=https://your-region.ml.cloud.ibm.com WATSONX\_MODEL\_ID=ibm/granite-13b-instruct-v2 PINECONE\_API\_KEY=your\_pinecone\_key PINECONE\_ENV=your\_pinecone\_env INDEX\_NAME=smartcity-policies

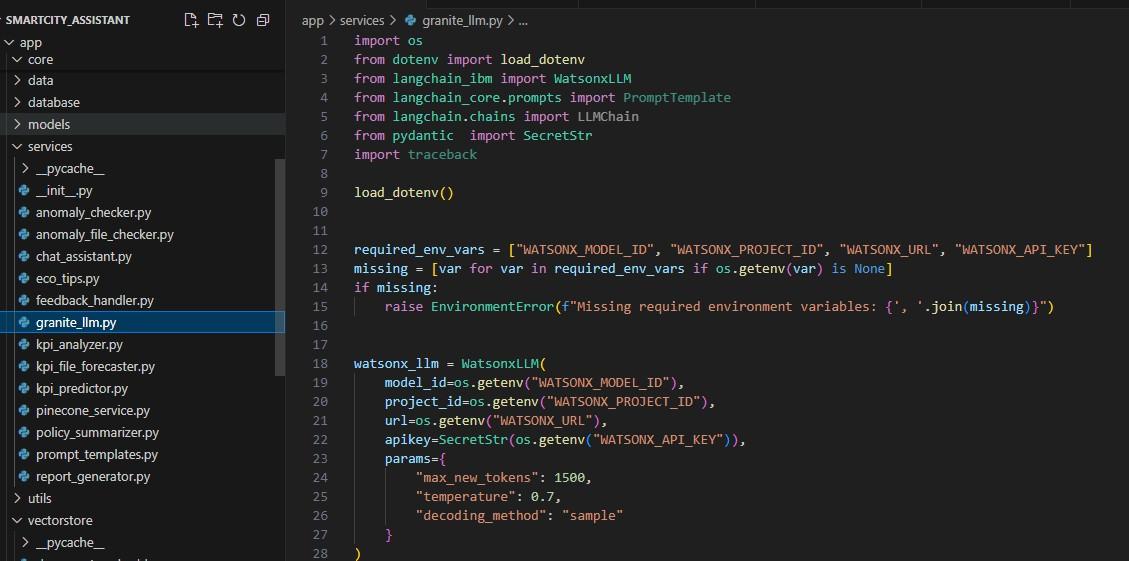
****

**Milestone 3: AI Model Integration**

**Objective:** Integrate Watsonx Granite LLM with a centralized service layer.

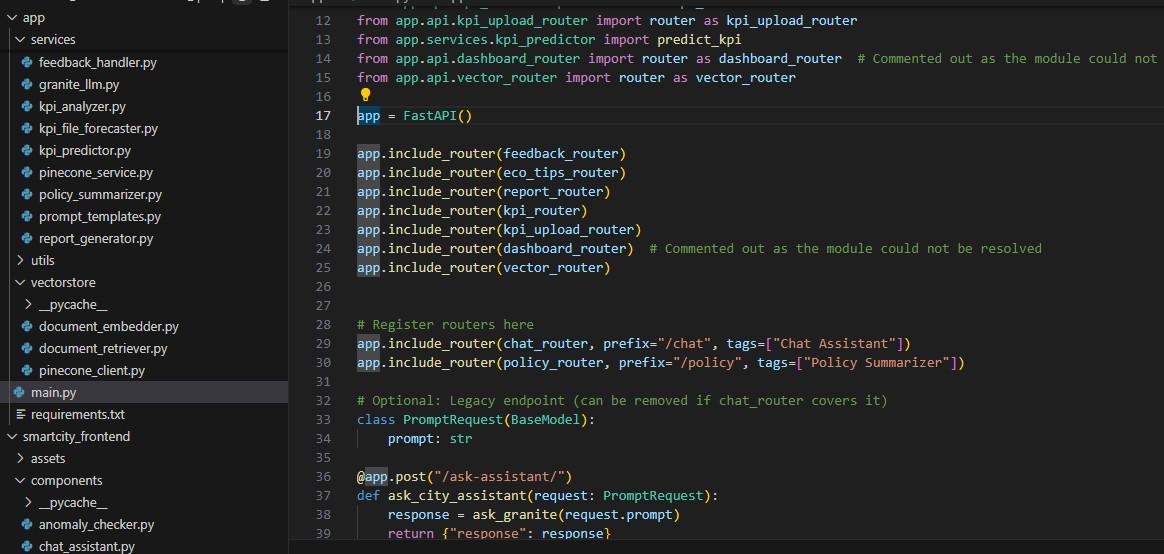
**Watsonx Integration**

* Load env variables using python-dotenv
* Set up granite\_llm.py to handle summarization, chat, eco tips, and sustainability reports
* Test LLM endpoints using dummy prompts



**Implement LLM Service Functions**

* ask\_granite(prompt) for chat
* generate\_summary(text) for policy summarization
* generate\_eco\_tip(topic) for environmental suggestions
* generate\_city\_report(kpi\_data) for sustainability reports

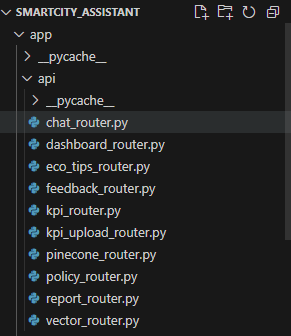


**Milestone 4: Backend API Development**

**Objective:** Build modular RESTful API routes using FastAPI.

**Create Routers**

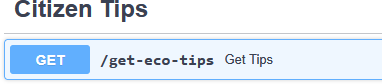
**Modules created in app/api/:**

* chat\_router.py
* policy\_router.py
* eco\_tips\_router.py
* feedback\_router.py
* report\_router.py
* vector\_router.py
* kpi\_upload\_router.py
* dashboard\_router.py

**Test Routes**

**Use Swagger UI to validate:**

* POST /upload-doc
* GET /search-docs
* GET /get-eco-tips?topic=energy
* POST /submit-feedback



**Milestone 5: Streamlit Frontend UI Development**

**Objective:** Design a user-friendly dashboard for real-time interaction.

**Page Structure:-**

* Sidebar navigation using streamlit-option-menu
* Separate pages for: Dashboard, Feedback, Eco Tips, Chat, Policy Search, Anomaly Checker, KPI Forecasting

**Build UI Components:-**

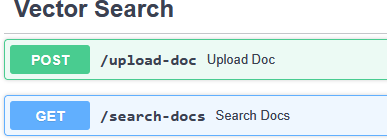
* summary\_card.py: Stylish KPI boxes
* chat\_assistant.py, feedback\_form.py: Form input UIs
* eco\_tips.py, policy\_summarizer.py: Prompt + result blocks
* report\_generator.py: PDF report generation

### **Milestone 6:Pinecone Semantic Search Integration**

**Objective:** Embed uploaded documents and enable semantic policy search.

Document Embedding

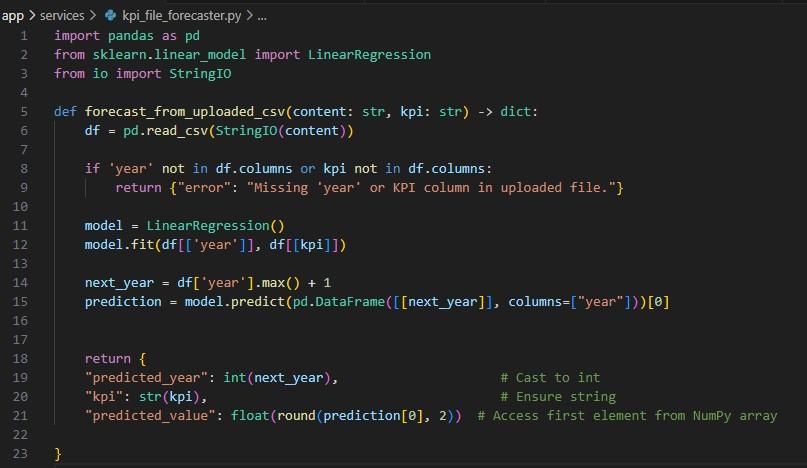
* Use sentence-transformers model (MiniLM) to convert .txt into 384-d vectors
* Store documents in Pinecone via document\_embedder.py

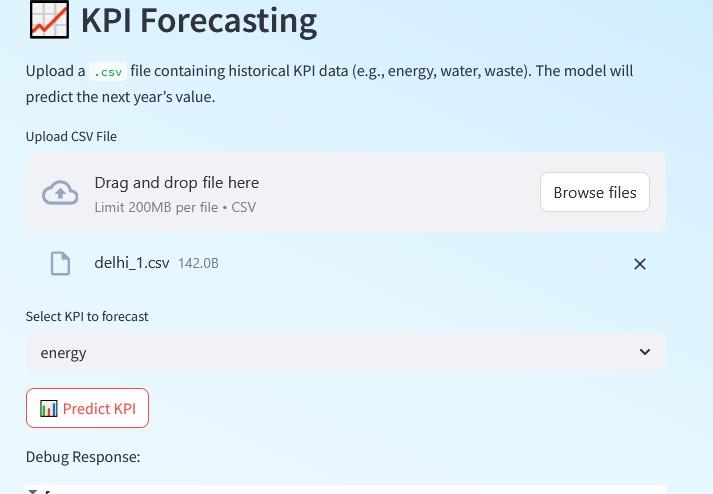


**Milestone 7:ML-based Forecasting and Anomaly Detection**

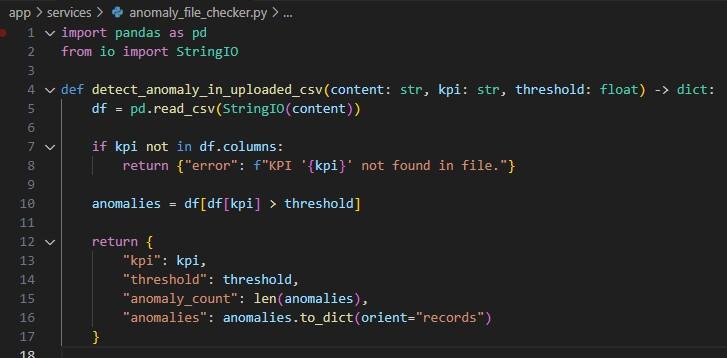
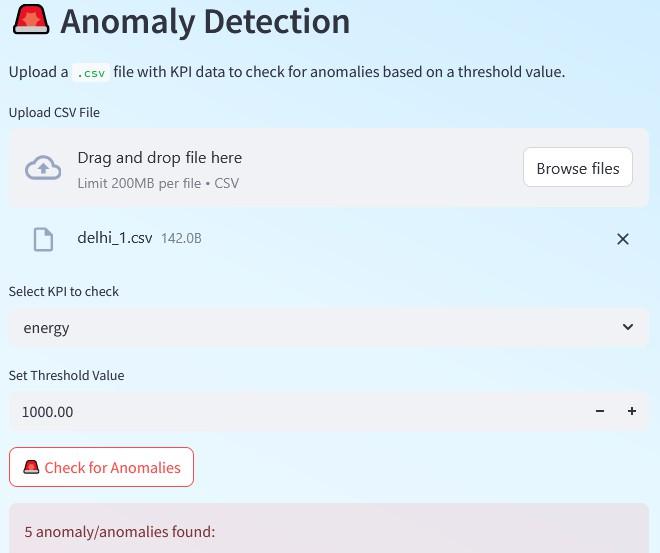
**Objective:** Analyze uploaded CSV files and predict future trends or irregularities.

**Forecasting**

* Use Linear Regression in kpi\_file\_forecaster.py
* Predict water/energy use based on past data
* Display forecast on dashboard

  
  
**Anomaly Detection:**

* anomaly\_file\_checker.py flags abnormal spikes
* Display results in tabular or colored badge format

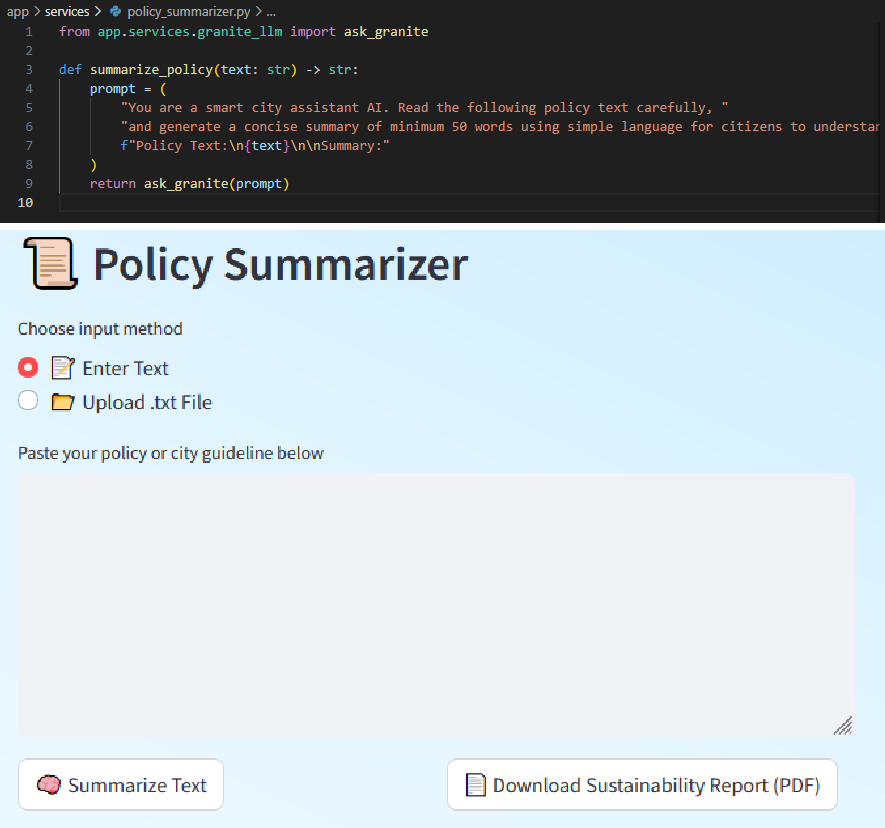
  
  
  
  
**Milestone 8:Sustainability Report Generation:**

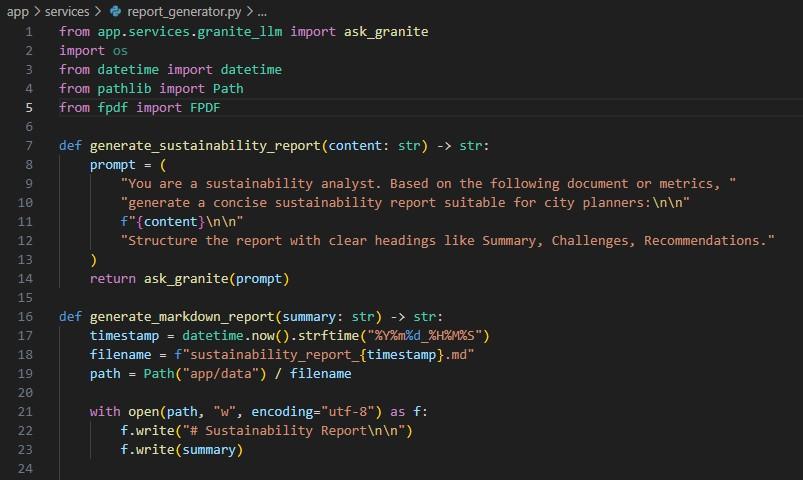
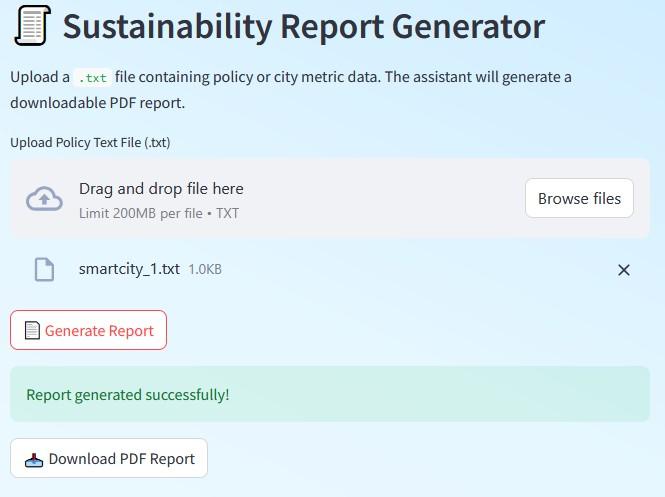
**Objective:** Generate a city-wise AI-powered sustainability summary.

**Prompt Engineering:**

* report\_generator.py uses a custom prompt to generate an AI-written report from KPI inputs

**Display/Download:**

* Render AI report on frontend
* Optionally provide markdown/PDF output

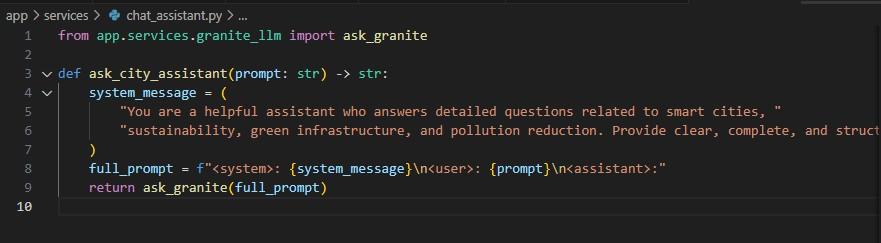


**Milestone 9: Chat Assistant Creation**

Objective: Build an interactive chat module where users can ask AI-driven questions related to sustainability, city governance, and smart living powered by IBM Watsonx Granite LLM.

**Define Backend Route**

* Create chat\_router.py in the app/api/ directory.
* Endpoint /chat/ask accepts a prompt string as input.
* Calls the ask\_granite() function from granite\_llm.py to generate the response.


**Milestone 10: Final Integration & Testing**

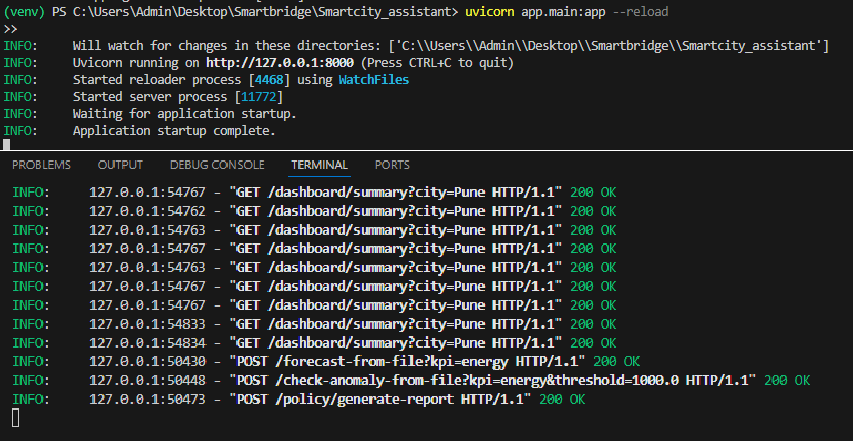
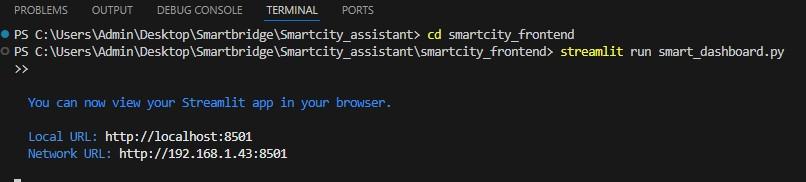
**Objective**: Ensure smooth interaction across modules.

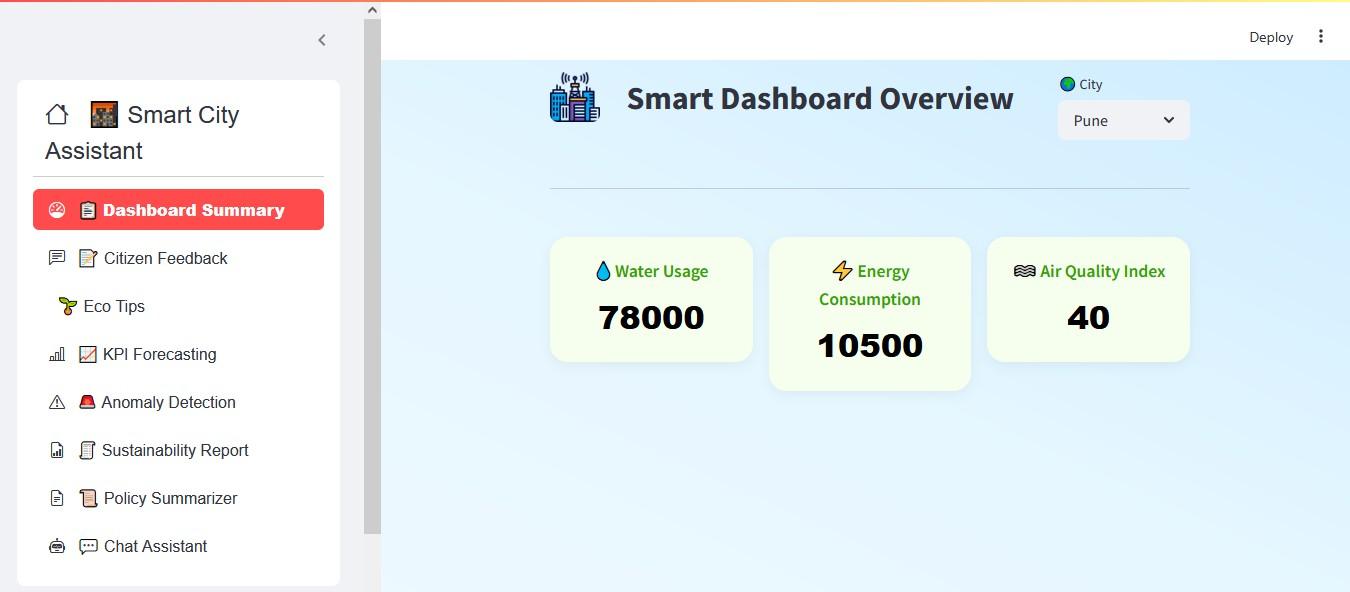
**Connect All Pages:**

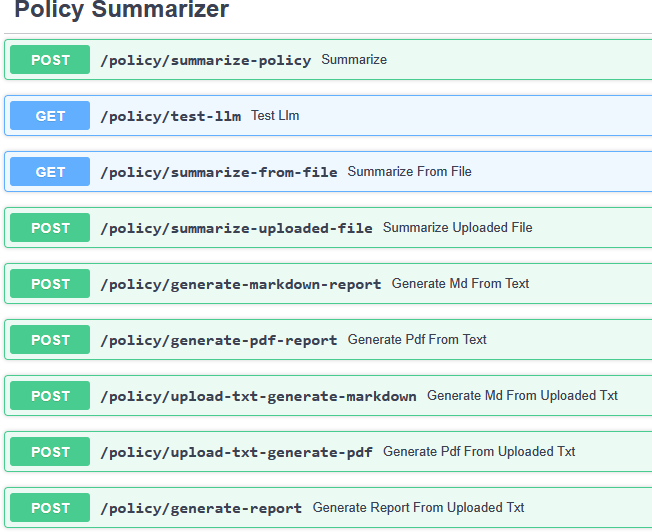
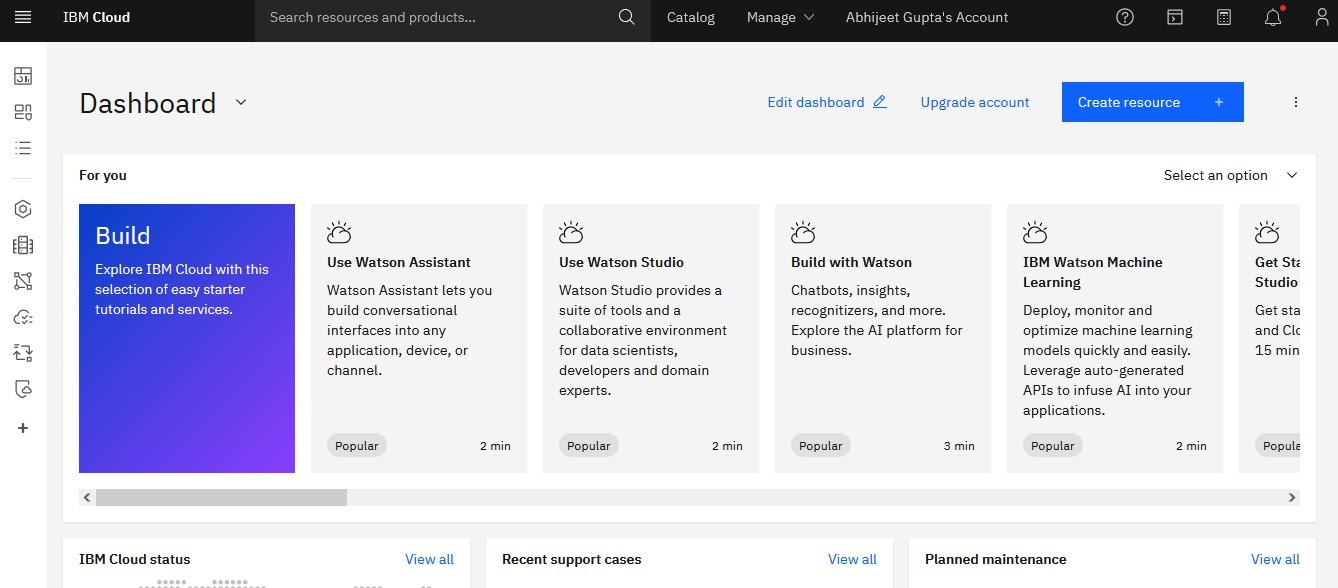
* Navigation working via sidebar
* Real-time API interactions tested

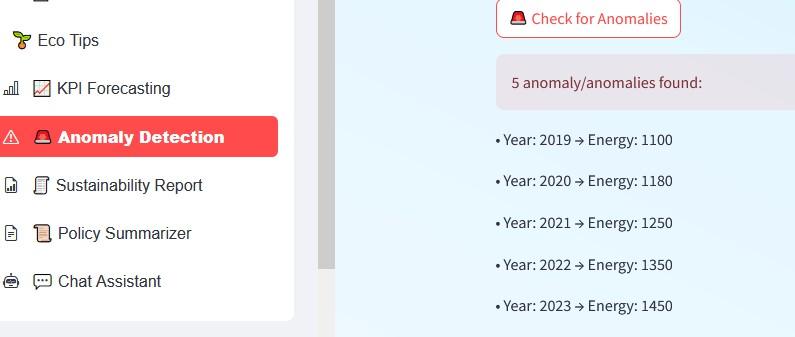
**Run Final Test:**

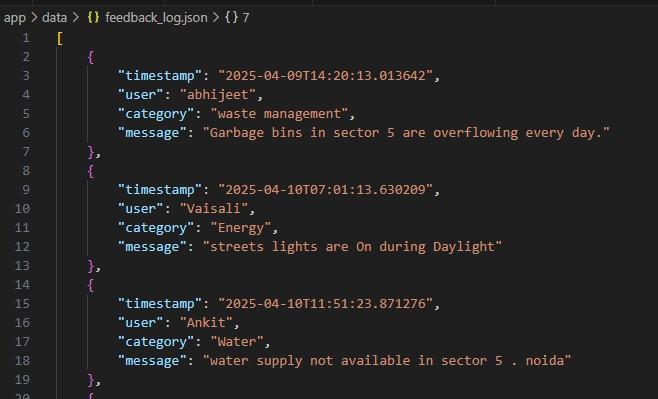
uvicorn app.main:app --reload streamlit run smart\_dashboard.py



**Screenshots / Outputs:**









Conclusion:-

The Sustainable Smart City Assistant is an AI-powered platform that leverages IBM Watsonx's Granite LLM and modern data pipelines to support urban sustainability, governance, and citizen engagement. It integrates several modules like City Health Dashboard, Citizen Feedback, Document Summarization, Eco-Advice, Anomaly Detection, KPI forecasting and Chat Assistant through a modular FastAPI backend and a Streamlit.IBM Watson enables cities to integrate various operational systems, such as traffic management, waste disposal, and energy distribution, into a single cohesive framework