Sustainable Smart City Assistant Using IBM Granite LLM

Project Overview

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- based frontend dashboard.

Use Case Scenarios

Policy Search & Summarization

A municipal planner uploads a complex city policy document to the assistant’s interface. In seconds, the assistant summarizes it into a concise, citizen-friendly version using IBM Granite LLM. This empowers planners to quickly interpret key points and make informed urban decisions.

Citizen Feedback Reporting

A resident notices a burst water pipe on a city street. Instead of calling helplines, they submit a report through the assistant’s feedback form. The issue is logged instantly with category tagging (e.g., "Water") and can be reviewed by city administrators.

KPI Forecasting

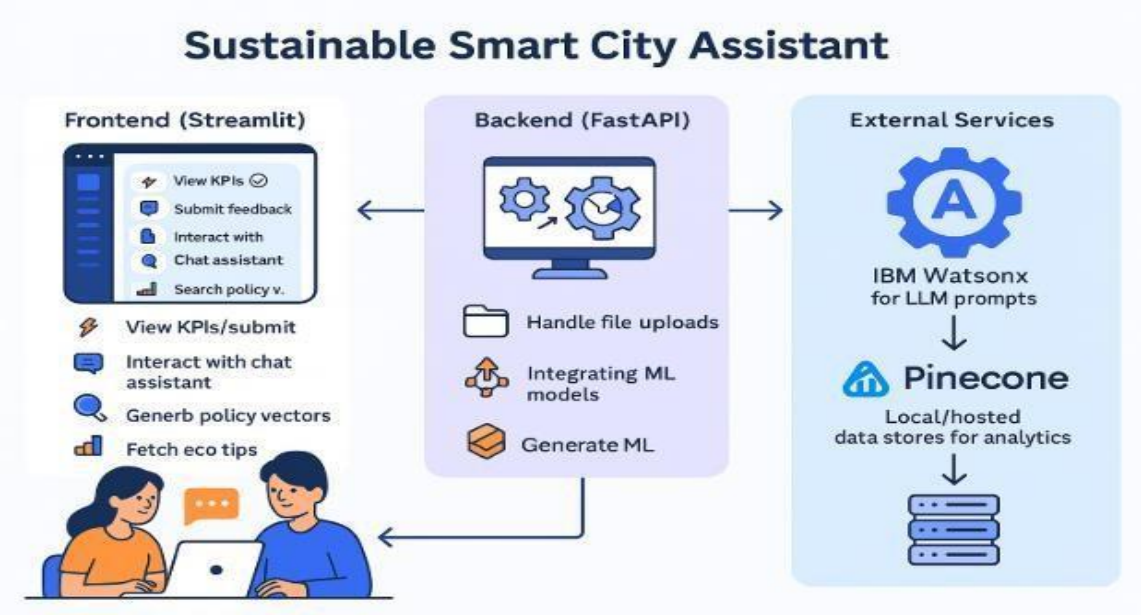
A city administrator uploads last year’s water usage KPI CSV. The assistant forecasts next year’s consumption using built-in machine learning. This data is used in planning budgets and infrastructure upgrades.

Architecture

The architecture consists of three main layers:

1. Frontend (Streamlit): Provides a modular dashboard with options to view KPIs, submit feedback, interact with chat assistant, search policy vectors, generate reports, and fetch eco tips.
2. Backend (FastAPI): Manages API requests, handles file uploads, and integrates with ML models, Pinecone, and Watsonx Granite LLM.
3. External Services: IBM Watsonx for LLM prompts, Pinecone for vector indexing &

semantic search, and local/hosted data stores for analytics.



Project Flow

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.

Prior Knowledge

You must have the prior knowledge of the following topics to complete this project:

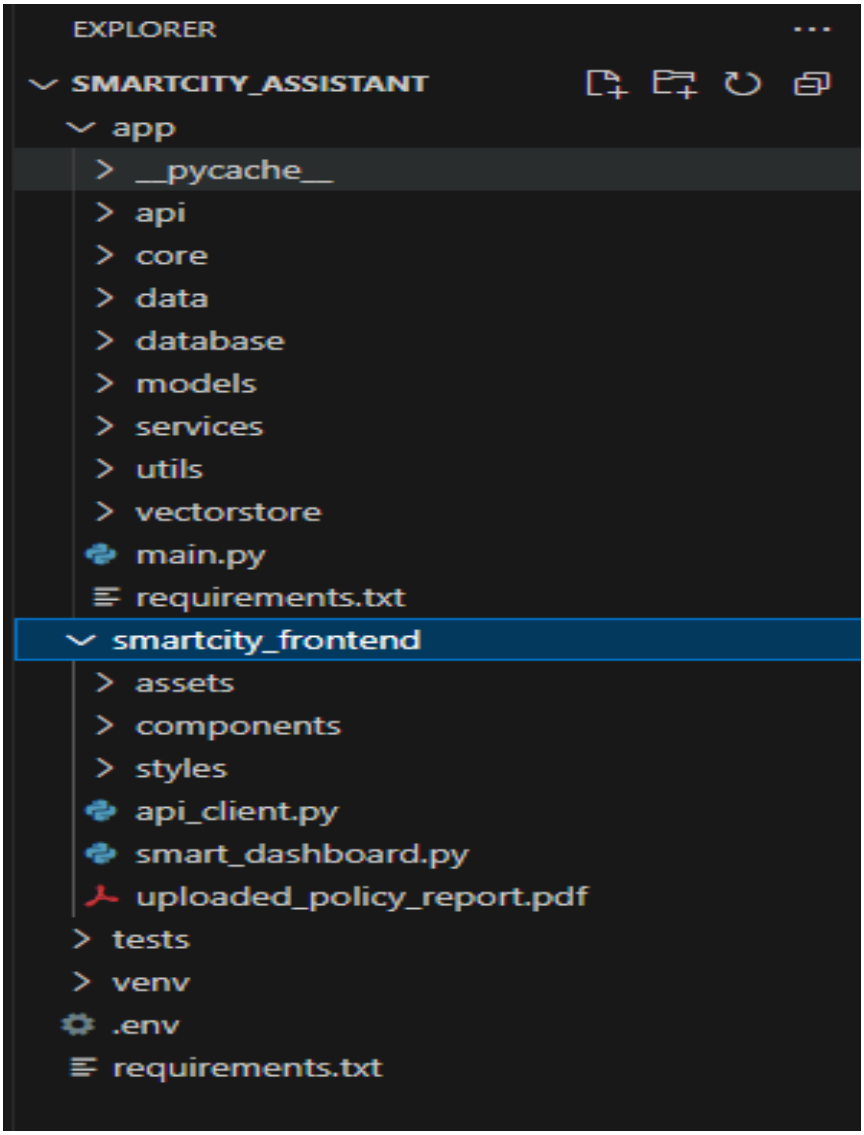
- Generative AI Concepts
- NLP: https://www.tutorialspoint.com/natural_language_processing/index.htm
- Generative AI: https://en.wikipedia.org/wiki/Generative_artificial_intelligence
- IBM Watsonx Granite: https://cloud.ibm.com/watsonx/overview
- LangChain: https://docs.langchain.com/docs/
- Pinecone: https://docs.pinecone.io/docs/overview
- FastAPI: https://fastapi.tiangolo.com/

Streamlit: https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/

Project Milestones & 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

```
.env
WATSONX_API_KEY=pAjcxi3DfOg687VOmCe3_Q8TmRKSDlp9wulXo52qwNn5
WATSONX_PROJECT_ID=f371addd-61dd-4ff0-882d-571db5a32aea
WATSONX_URL=https://eu-de.ml.cloud.ibm.com
WATSONX_MODEL_ID=ibm/granite-13b-instruct-v2
PINECONE_API_KEY=pcsk_22YGb3_9RY8BMqaUZN55nkxUA7nR7ZyhBnKA1LjW44XtRpkeo43rCmj2yW4HrPhQfQbafu
PINECONE_ENV=us-east-1
INDEX_NAME=smartcity-policies
```

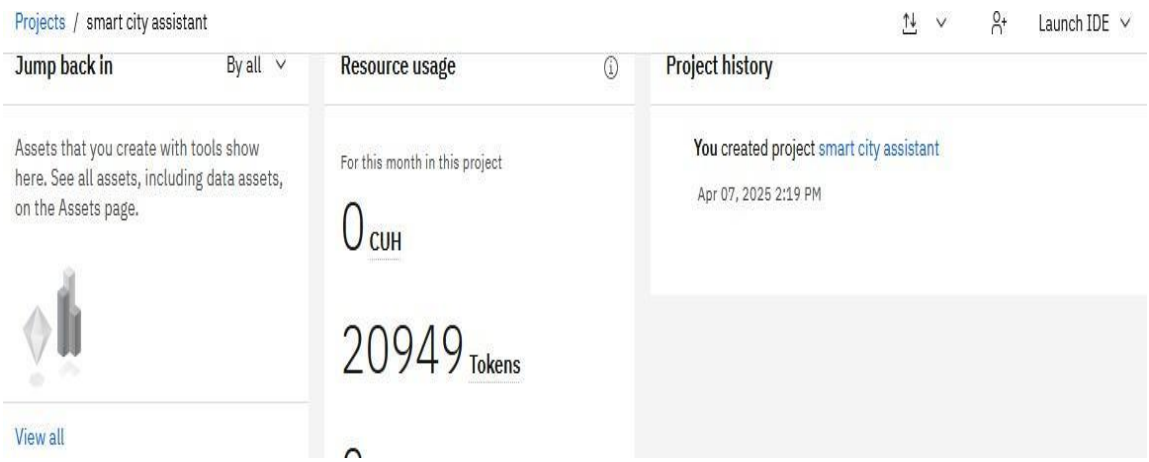
Config.py file

```
app > core > config.py > Settings > Config
class Settings(BaseSettings):


    # Watsonx configs
    WATSONX_API_KEY: str = "pAjcxi3DfOg687VOmCe3_Q8TmRKSDlp9wulXo52qwNn5"
    WATSONX_PROJECT_ID: str = "f371addd-61dd-4ff0-882d-571db5a32aea"
    WATSONX_URL: str = "https://eu-de.ml.cloud.ibm.com"
    WATSONX_MODEL_ID: str = "ibm/granite-13b-instruct-v2"


    class Config:
        env_file = ".env"
        extra = "allow"

settings = Settings()

```

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

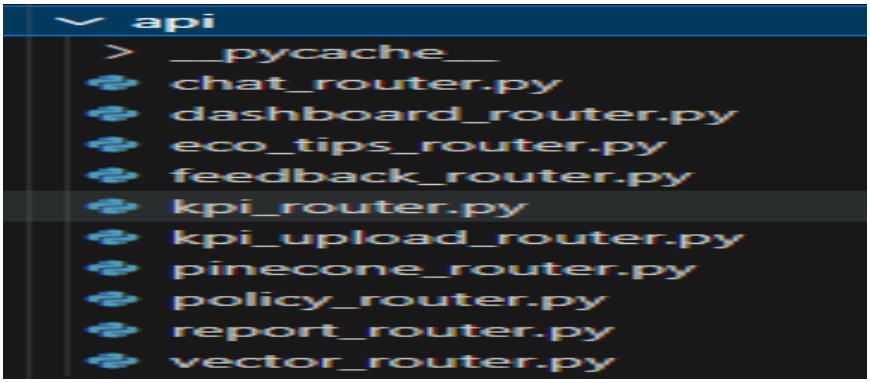


Endpoint Testing:
Validated /chat, /policy/summarize, and /get-eco-tips FastAPI routes using Swagger UI.

Phase 3 – Backend API Routers
API Routes Implemented:
Developed modular routers:

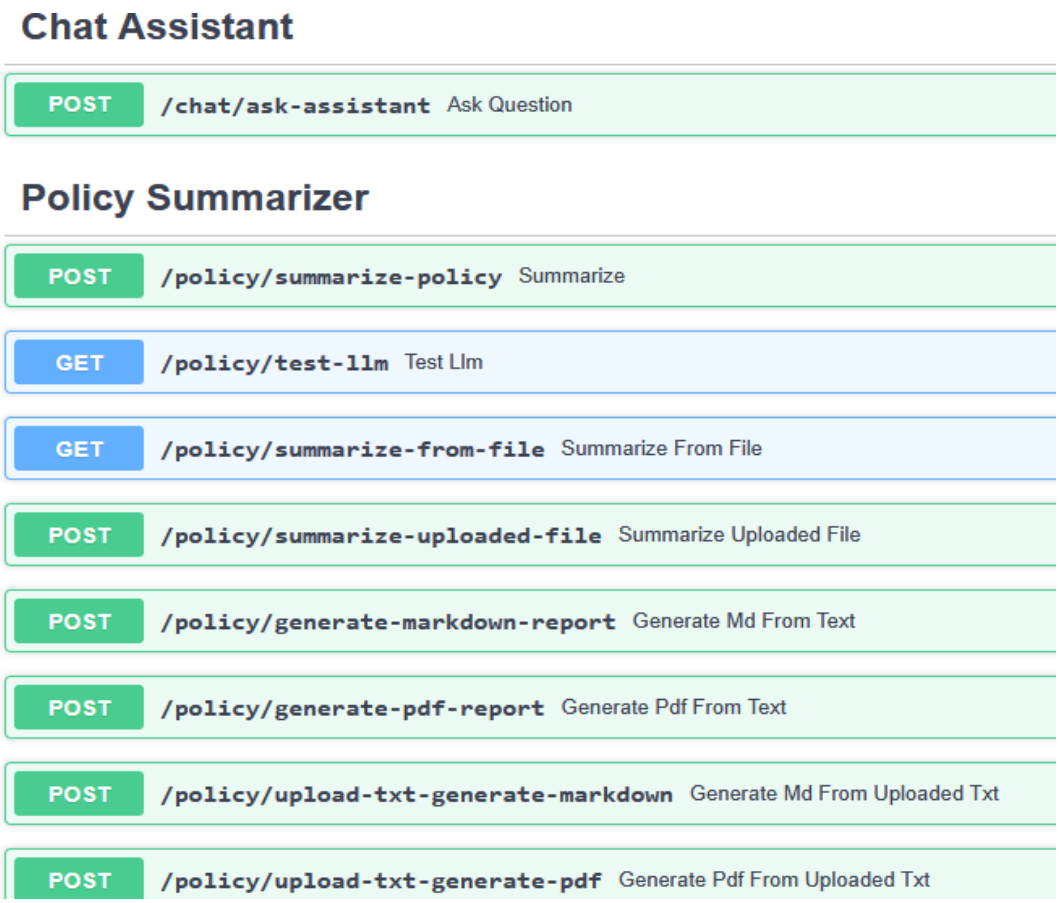
- chat_router.py

- feedback_router.py
- eco_tips_router.py
- kpi_upload_router.py
- anomaly_checker.py
- vector_router.py, etc.



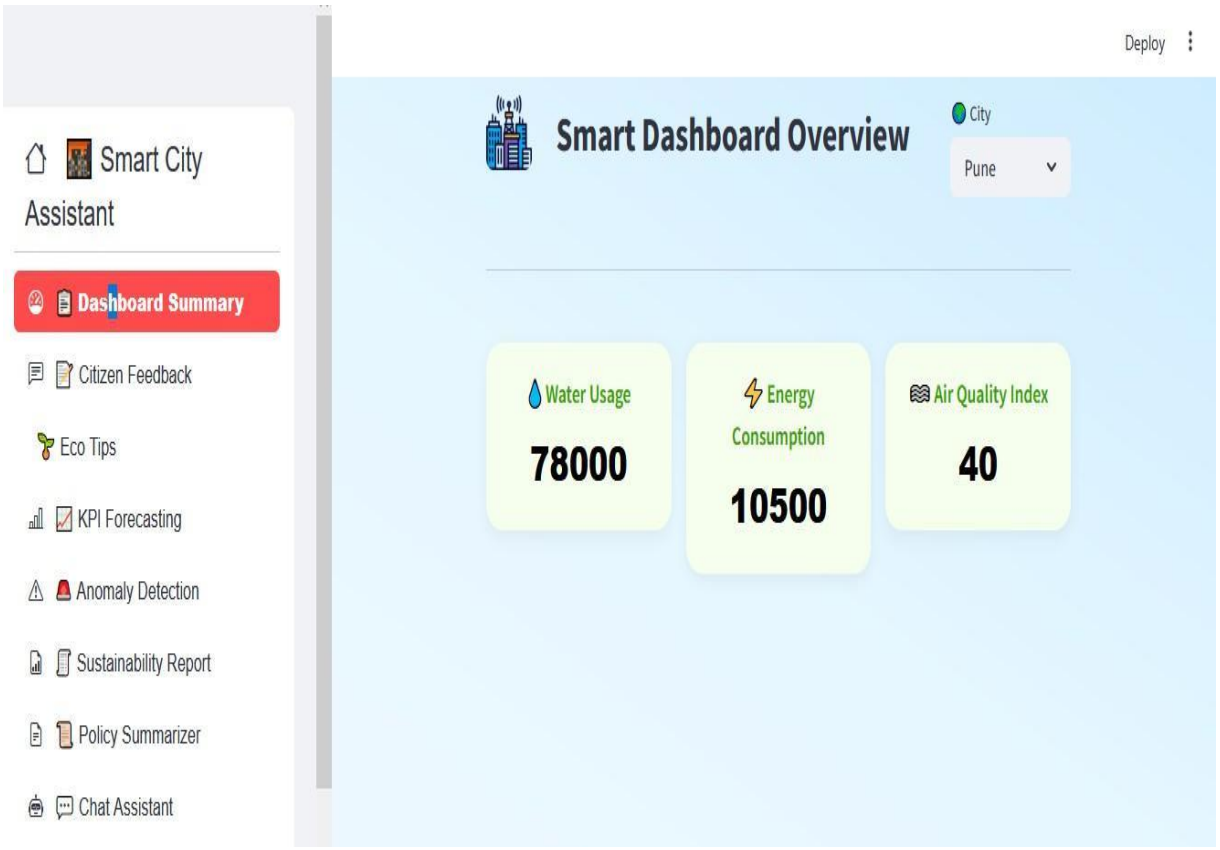
Testing & Validation:
Each route tested for:

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



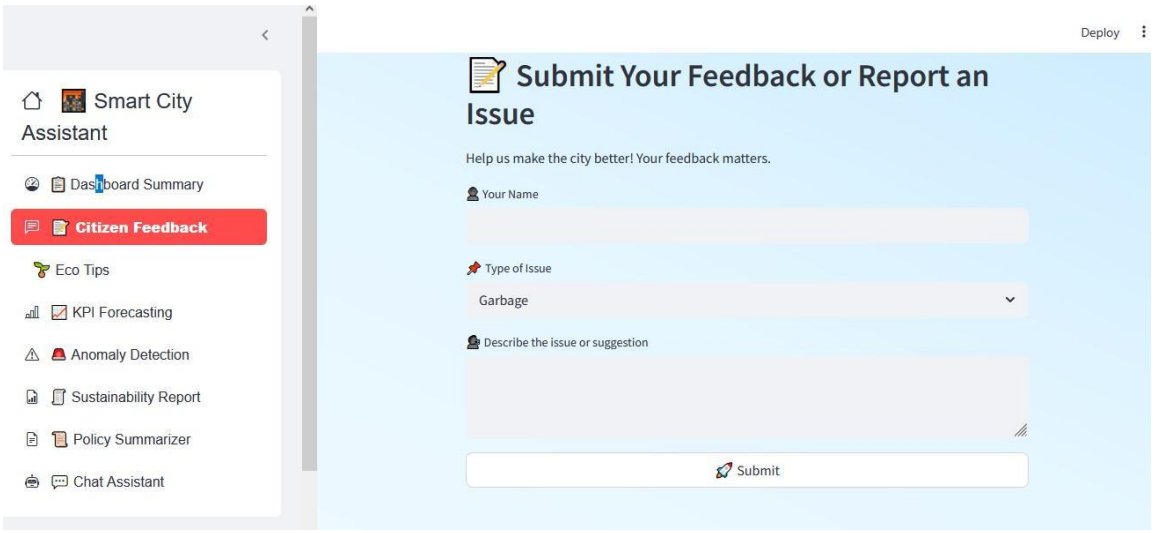
Phase 4 – Frontend UI Design

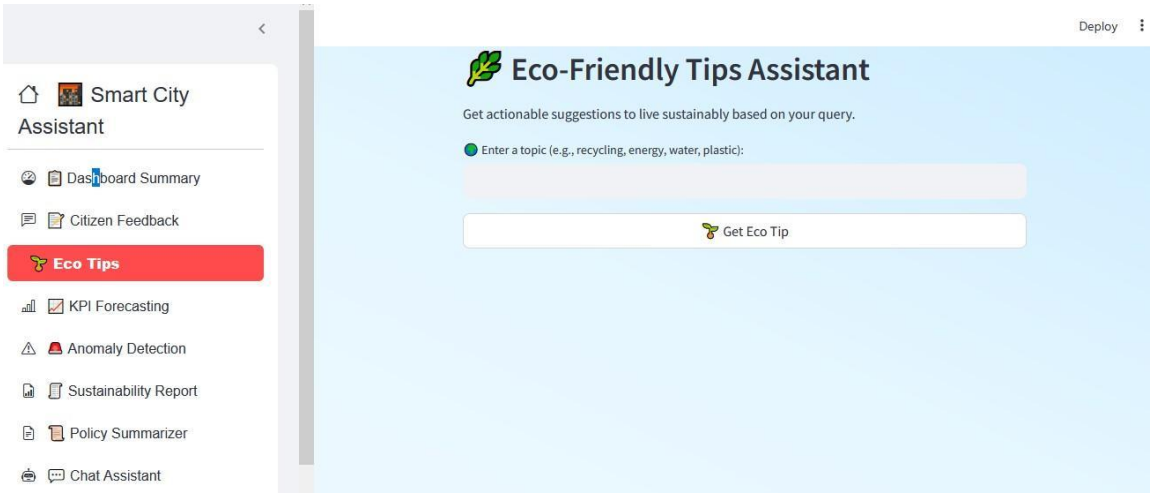
Streamlit UI Structure Implemented:
Created central file smart_dashboard.py with conditional rendering for each module using sidebar navigation.



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.

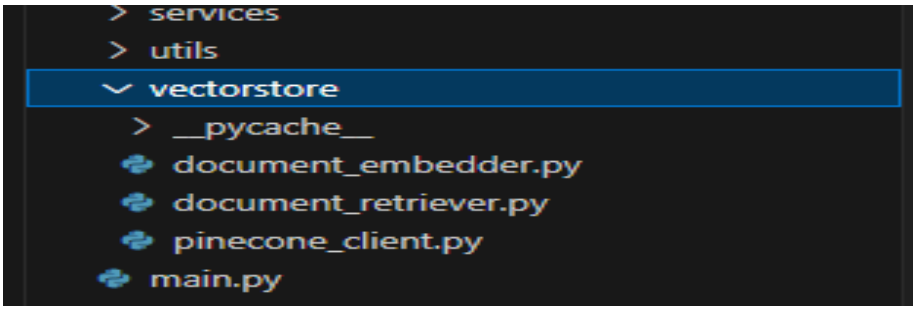




UI Enhancements Done:
Gradient backgrounds
Icon-rich sidebar using streamlit-option-menu
Rounded buttons, font styles, padding fixes

Phase 5 – Pinecone & Document Embedding

Embedding Logic Built:
Created document_embedder.py and document_retriever.py using sentence-transformers.



Phase 6 – Report Generation & Deployment

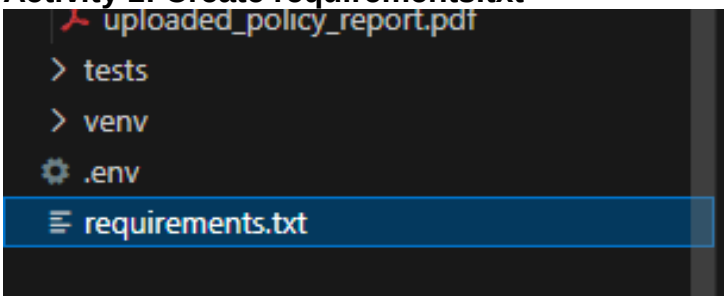
Granite LLM Report Generator:
report_generator.py takes city name and KPI data, generates detailed city sustainability report using Granite LLM prompts.
Markdown & PDF Support:
Output formatted to text block for copy/paste or PDF download (optional).
End-to-End Integration Testing:
Final dashboard tested 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.

Activity 1: Create requirements.txt

Activity 1: Create requirements.txt



Define the required libraries:

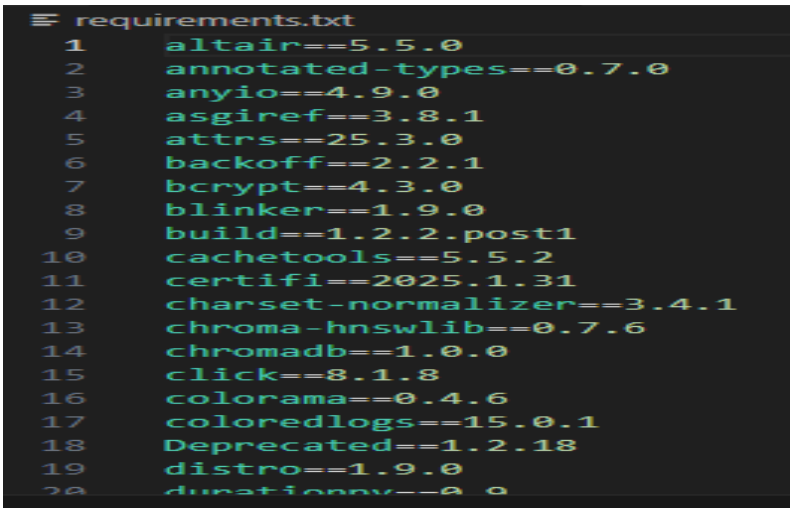
- streamlit: For building interactive dashboard interfaces

- 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

Activity 2: Install all dependencies

bash

pip install -r requirements.txt



Milestone 2: Environment Initialization & API Key Setup

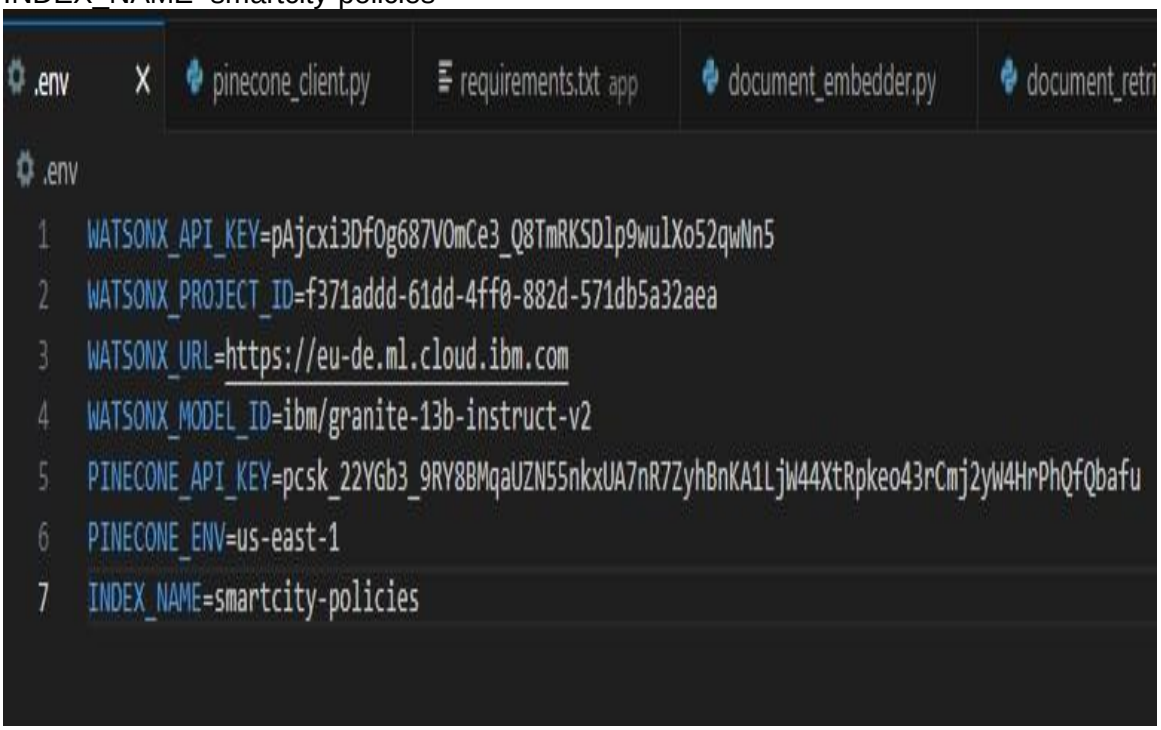
Objective: Configure and secure external service credentials (Watsonx & Pinecone).

Activity 1: Generate API Keys

- Watsonx Granite credentials from IBM Cloud dashboard
- Pinecone API key and environment from https://app.pinecone.io

Activity 2: 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



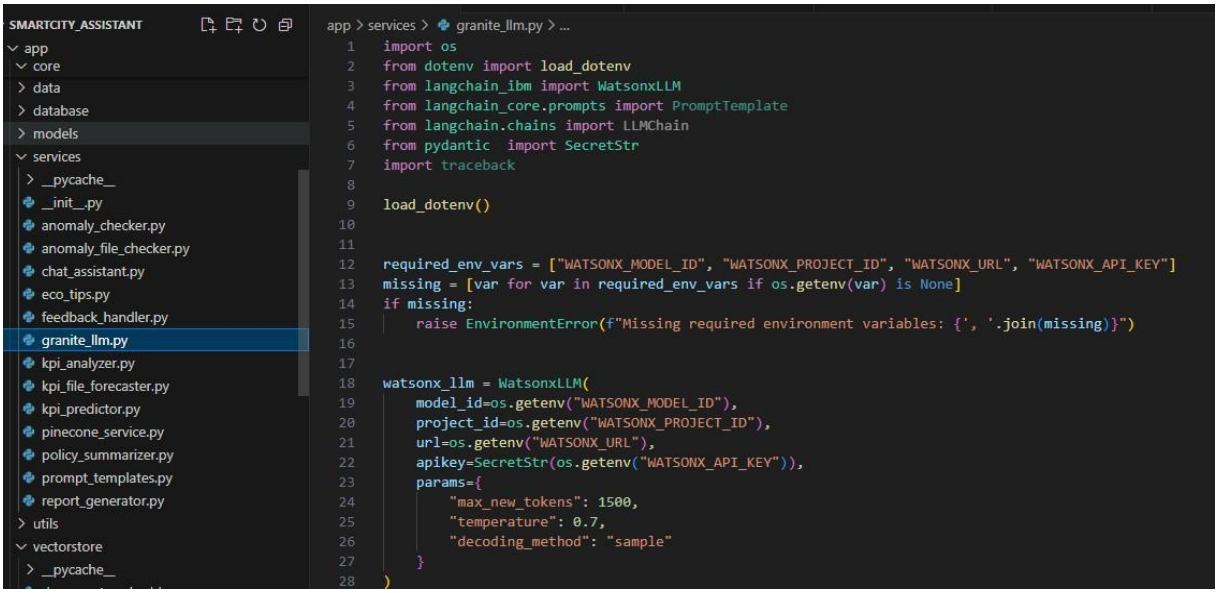
Milestone 3: AI Model Integration

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

Activity 1: Watsonx Integration

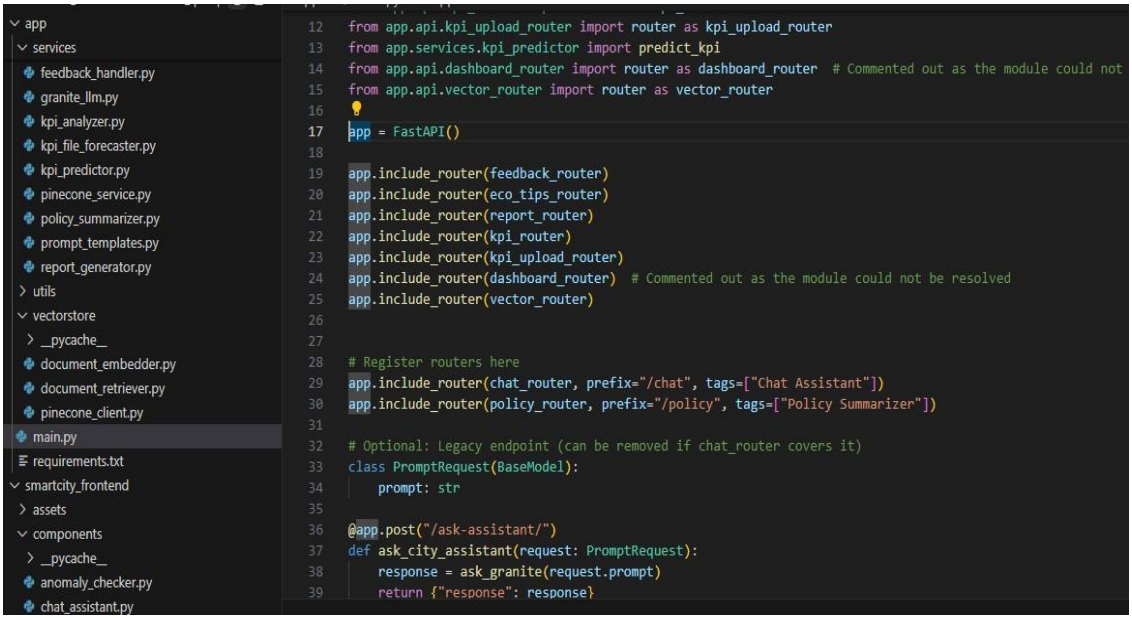
- Load env variables using python-dotenv

- Set up granite_llm.py to handle summarization, chat, eco tips, and sustainability reports
- Test LLM endpoints using dummy prompts



Activity 2: 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



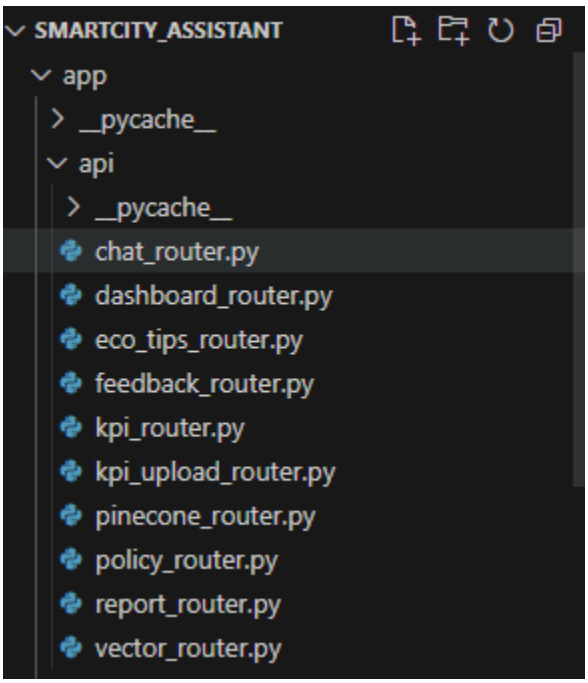
Milestone 4: Backend API Development

Objective: Build modular RESTful API routes using FastAPI.

Activity 1: Create Routers

Modules created in app/api/:

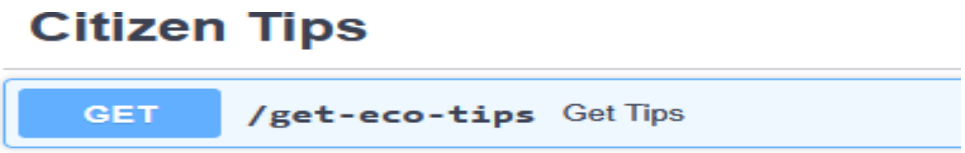
- chat_router.py
- policy_router.py
- eco_tips_router.py
- feedback_router.py
- report_router.py
- vector_router.py
- kpi_upload_router.py
- dashboard_router.py



Activity 2: 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.

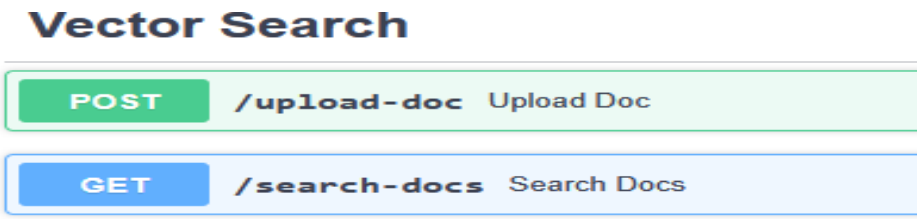
Activity 1: Page Structure

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

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

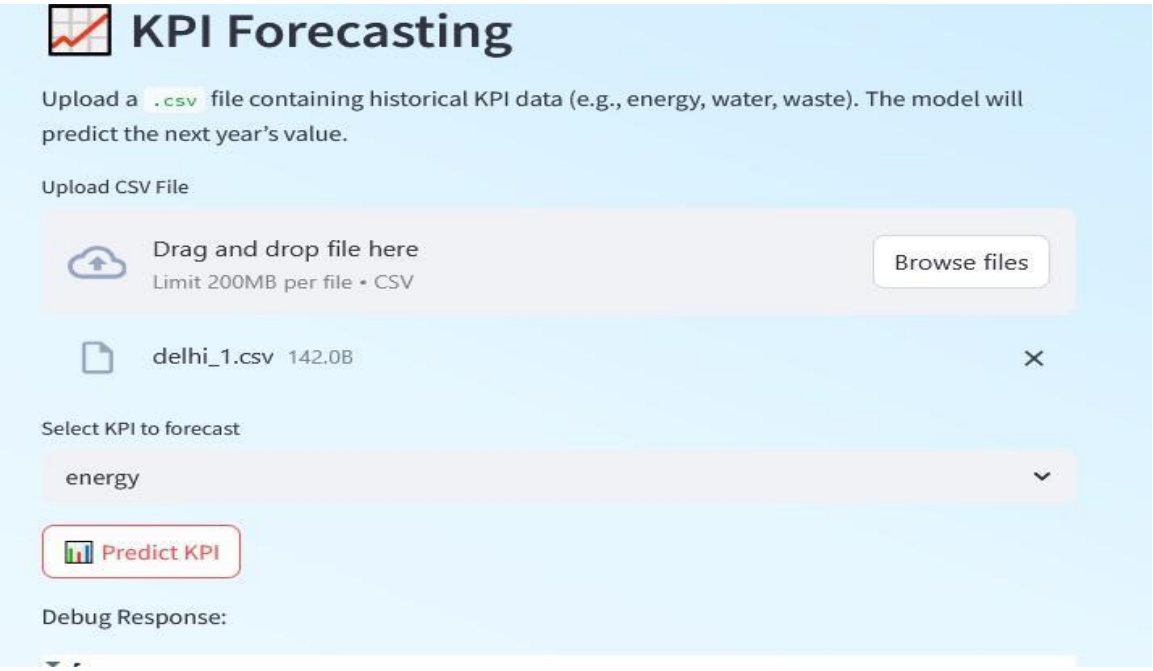
Objective: Analyze uploaded CSV files and predict future trends or irregularities.

Activity 1: Forecasting

- Use Linear Regression in kpi_file_forecaster.py
- Predict water/energy use based on past data
- Display forecast on dashboard

app > services > kpi_file_forecaster.py > ...

```python
import pandas as pd
from sklearn.linear_model import LinearRegression
from io import StringIO

def forecast_from_uploaded_csv(content: str, kpi: str) -> dict:
    df = pd.read_csv(StringIO(content))

    if 'year' not in df.columns or kpi not in df.columns:
        return {"error": "Missing 'year' or KPI column in uploaded file."}

    model = LinearRegression()
    model.fit(df[['year']], df[[kpi]])

    next_year = df['year'].max() + 1
    prediction = model.predict(pd.DataFrame([[next_year]], columns=["year"]))[0]


    return {
    "predicted_year": int(next_year),                    # Cast to int
    "kpi": str(kpi),                                     # Ensure string
    "predicted_value": float(round(prediction[0], 2))  # Access first element from NumPy array

}
```

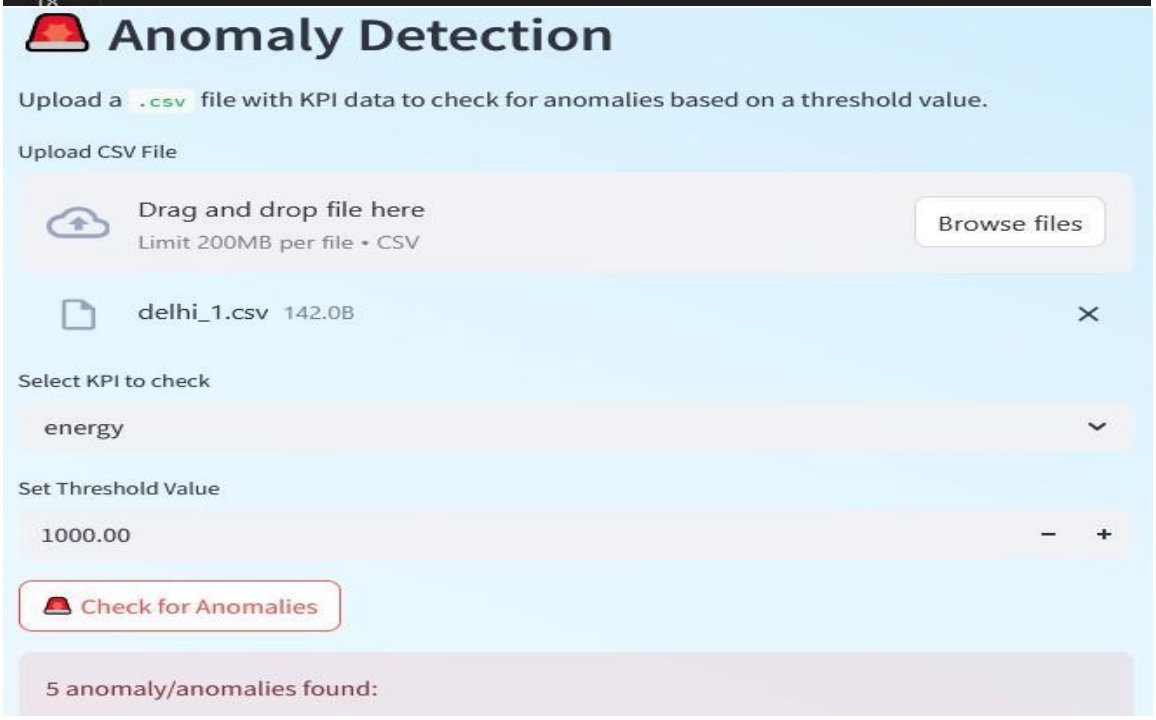


Activity 2: Anomaly Detection

- anomaly_file_checker.py flags abnormal spikes
- Display results in tabular or colored badge format

app > services > anomaly_file_checker.py > ...

```python
import pandas as pd
from io import StringIO

def detect_anomaly_in_uploaded_csv(content: str, kpi: str, threshold: float) -> dict:
    df = pd.read_csv(StringIO(content))

    if kpi not in df.columns:
        return {"error": f"KPI '{kpi}' not found in file."}

    anomalies = df[df[kpi] > threshold]

    return {
        "kpi": kpi,
        "threshold": threshold,
        "anomaly_count": len(anomalies),
        "anomalies": anomalies.to_dict(orient="records")
    }
```



Milestone 8: Sustainability Report Generation

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

Activity 1: Prompt Engineering

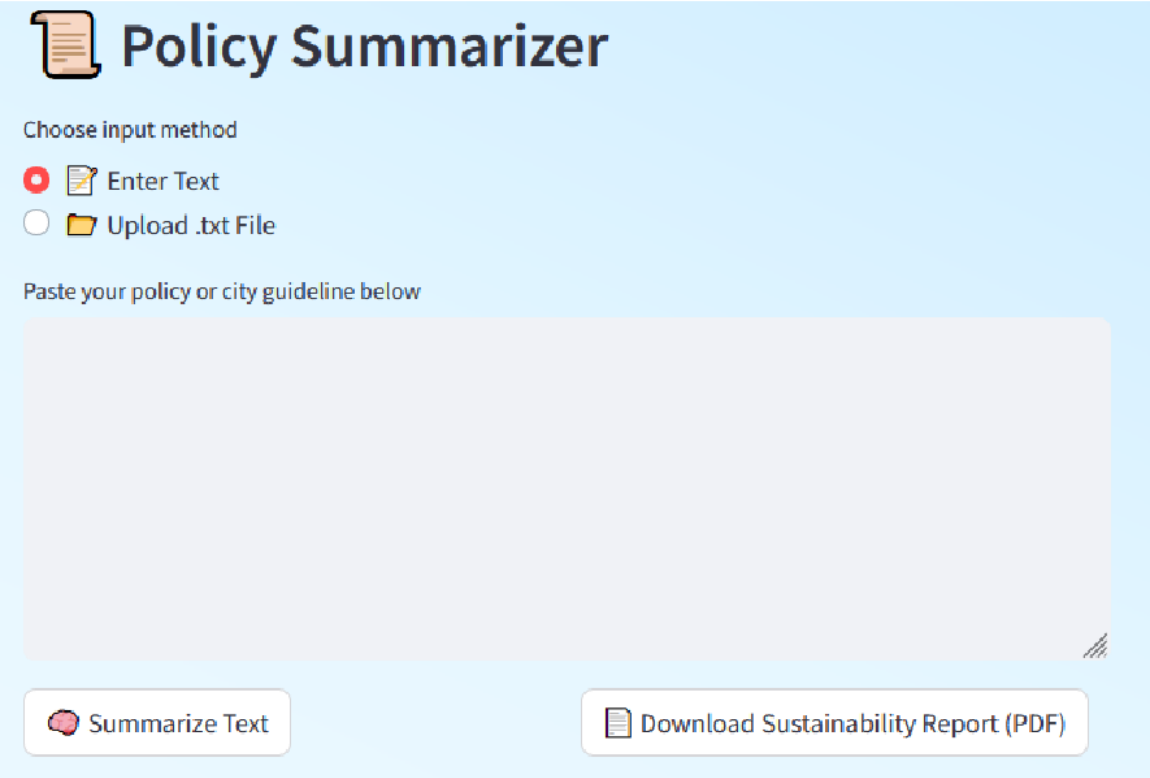
- report_generator.py uses a custom prompt to generate an AI-written report from KPI inputs

Activity 2: Display/Download

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

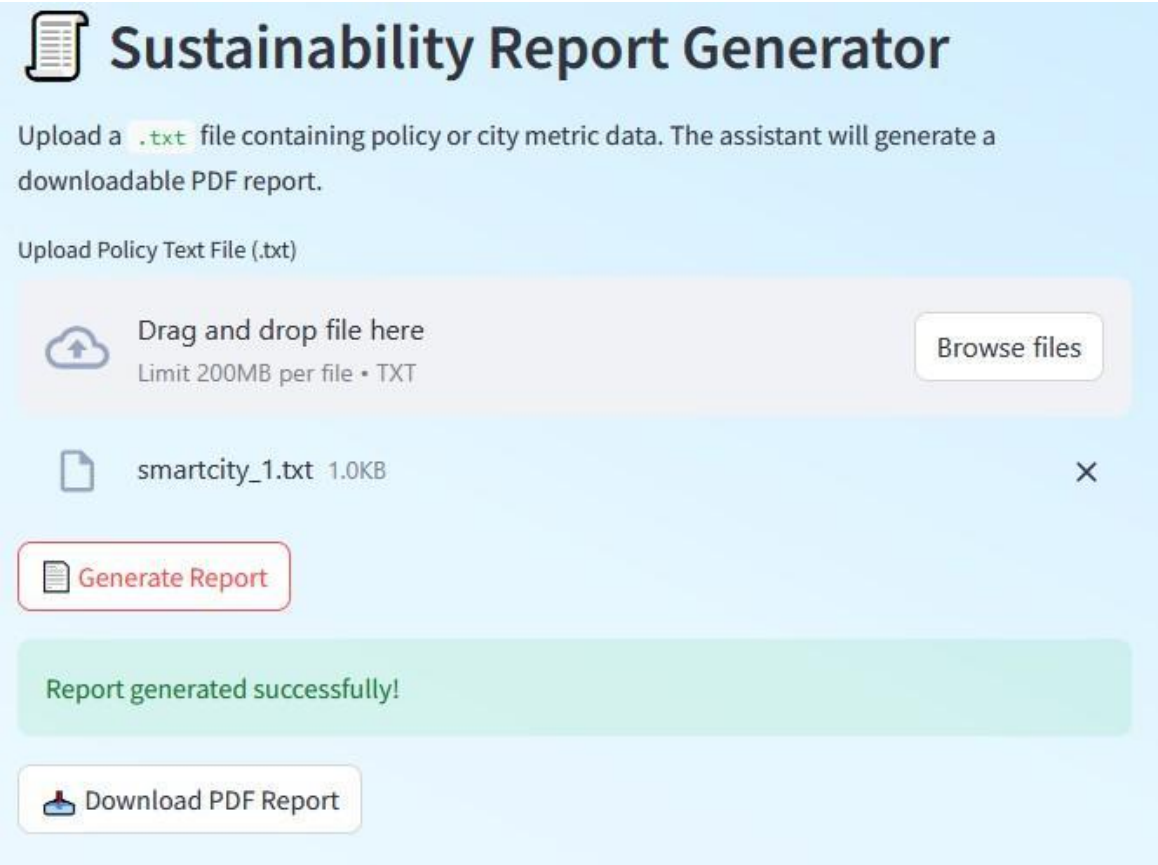
app > services > policy_summarizer.py > ...

```python
from app.services.granite_llm import ask_granite

def summarize_policy(text: str) -> str:
    prompt = (
        "You are a smart city assistant AI. Read the following policy text carefully, "
        "and generate a concise summary of minimum 50 words using simple language for citizens to understar
        f"Policy Text:\n{text}\n\nSummary:"
    )
    return ask_granite(prompt)

```



app > services > report_generator.py > ...

```python
from app.services.granite_llm import ask_granite
import os
from datetime import datetime
from pathlib import Path
from fpdf import FPDF

def generate_sustainability_report(content: str) -> str:
    prompt = (
        "You are a sustainability analyst. Based on the following document or metrics, "
        "generate a concise sustainability report suitable for city planners:\n\n"
        f"{content}\n\n"
        "Structure the report with clear headings like Summary, Challenges, Recommendations."
    )
    return ask_granite(prompt)

def generate_markdown_report(summary: str) -> str:
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"sustainability_report_{timestamp}.md"
    path = Path("app/data") / filename

    with open(path, "w", encoding="utf-8") as f:
        f.write("# Sustainability Report\n\n")
        f.write(summary)

```

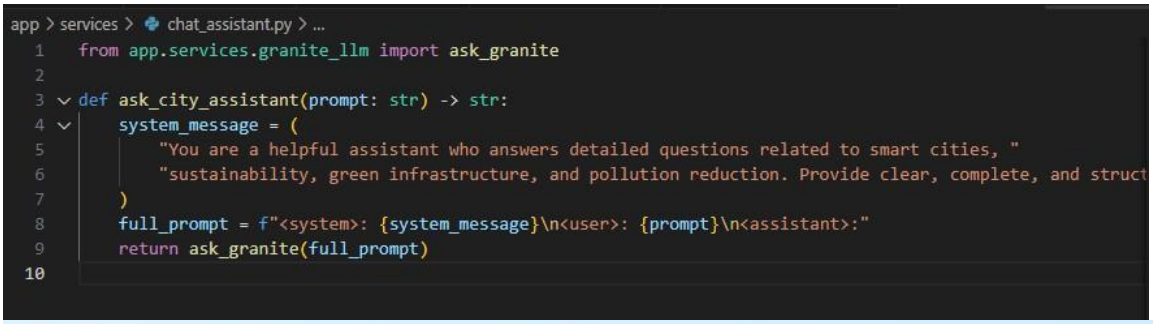


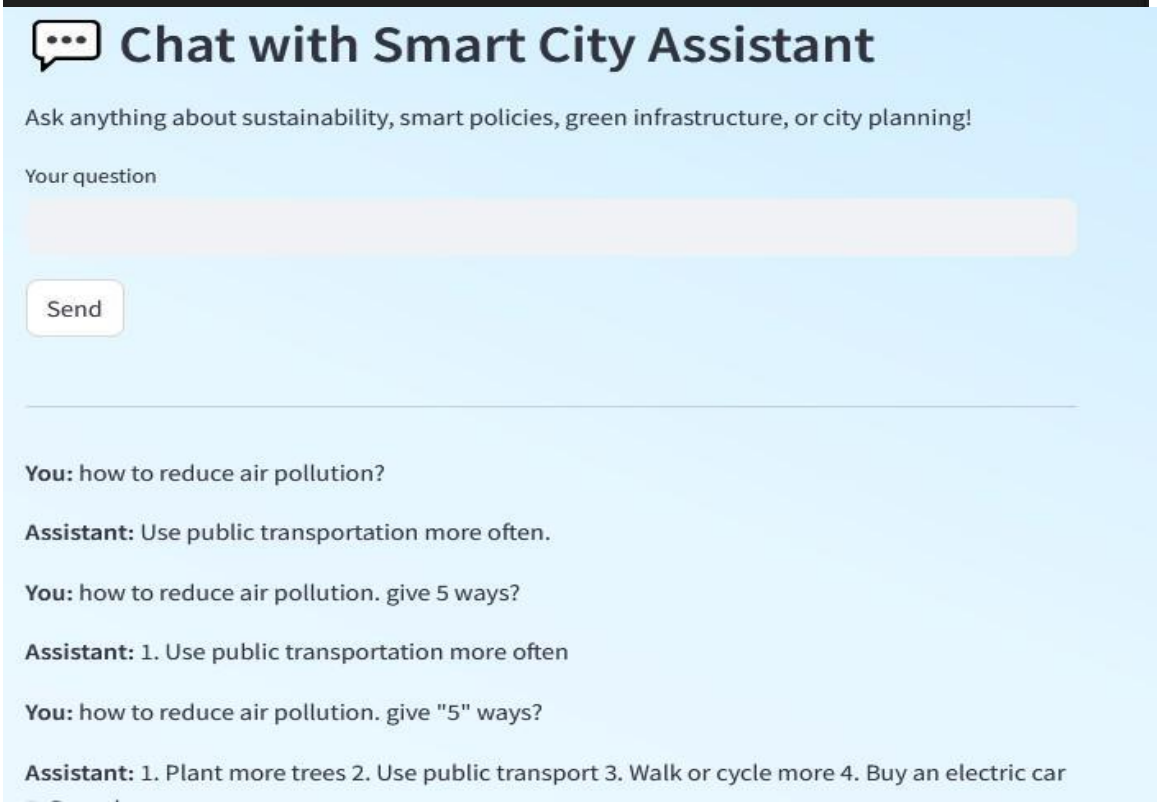
Milestone 9: Chat Assistant Creation

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.

Activity 1: 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.

Activity 1: Connect All Pages

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

Activity 2: Run Final Test

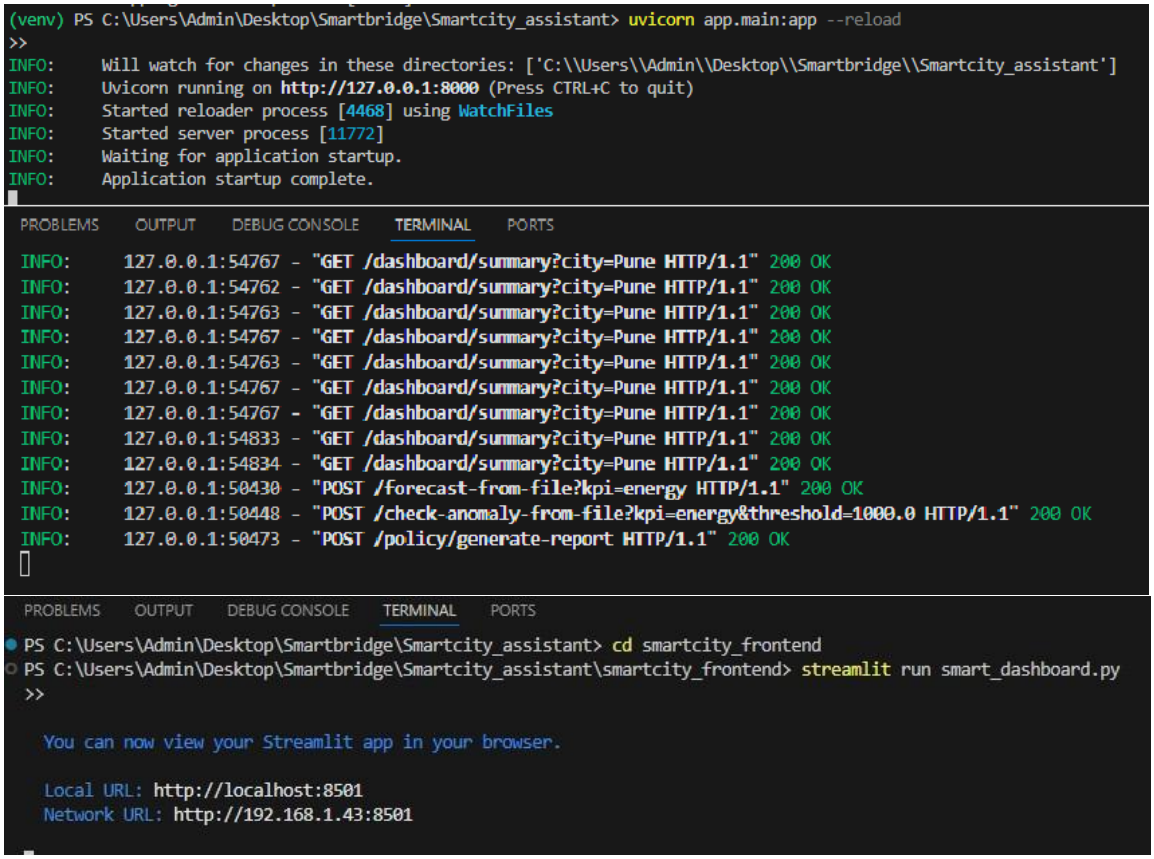
uvicorn

app.main:app

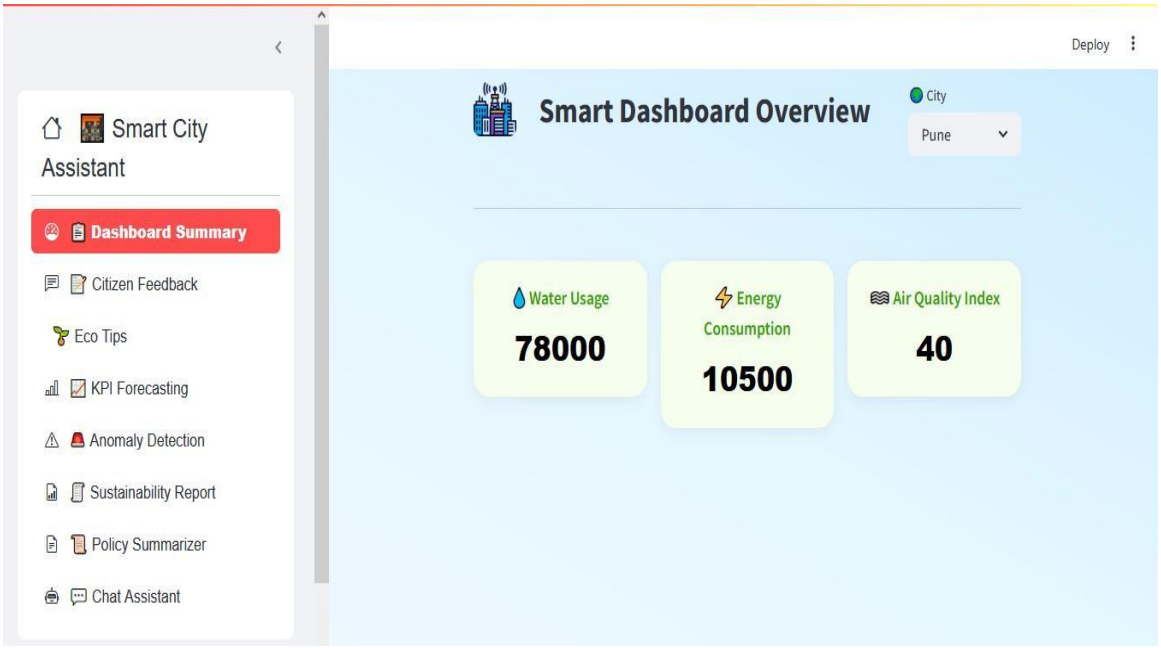
--reload

streamlit run

smart_dashboard.py

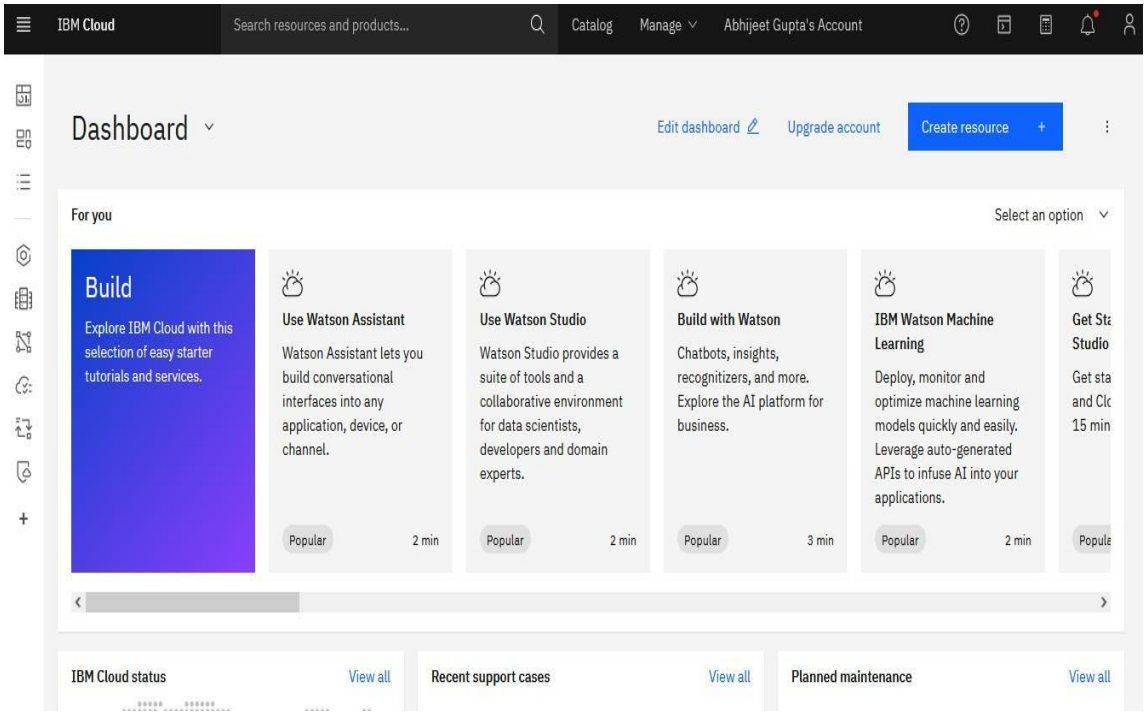


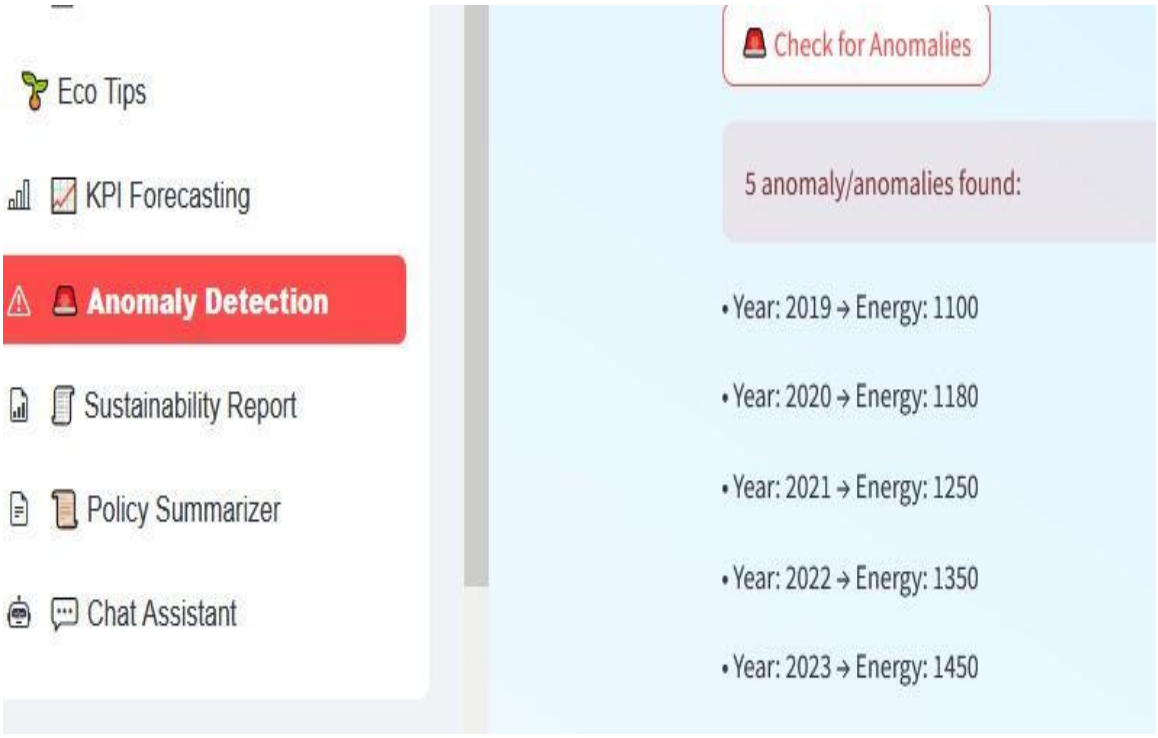
Screenshots / Outputs:



Policy Summarizer

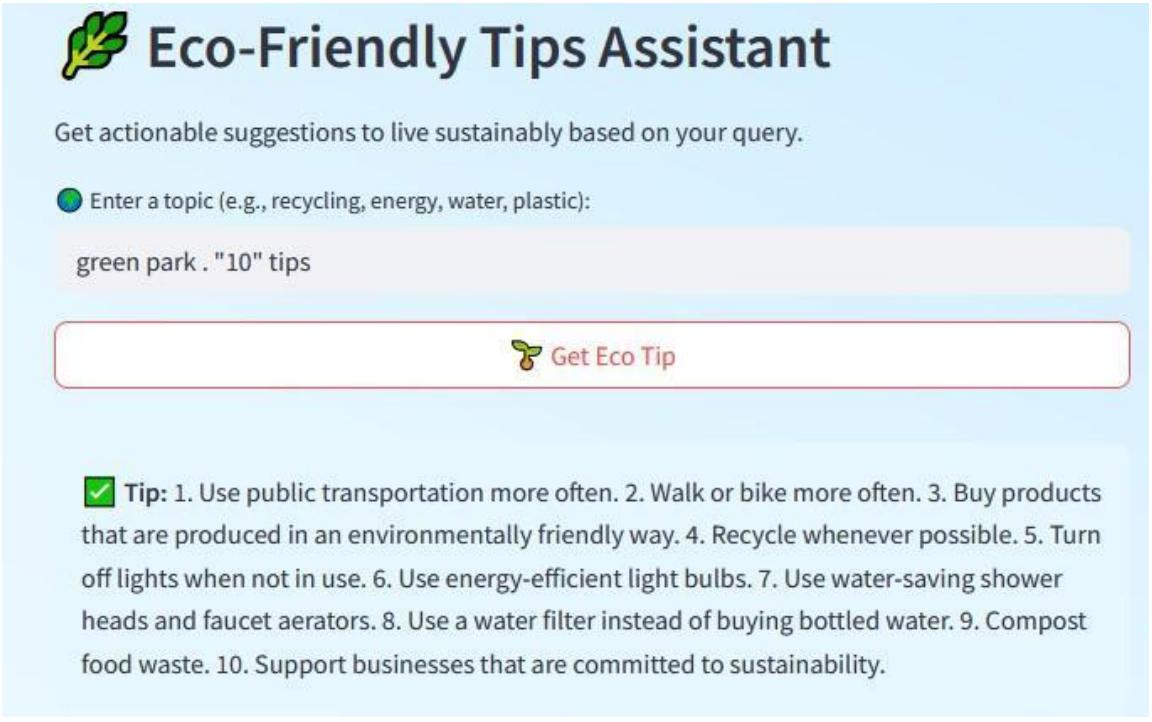
POST	/policy/summarize-policy	Summarize
GET	/policy/test-llm	Test Llm
GET	/policy/summarize-from-file	Summarize From File
POST	/policy/summarize-uploaded-file	Summarize Uploaded File
POST	/policy/generate-markdown-report	Generate Md From Text
POST	/policy/generate-pdf-report	Generate Pdf From Text
POST	/policy/upload-txt-generate-markdown	Generate Md From Uploaded Txt
POST	/policy/upload-txt-generate-pdf	Generate Pdf From Uploaded Txt
POST	/policy/generate-report	Generate Report From Uploaded Txt


IBM Cloud
Search resources and products...
Catalog
Manage
Abhijeet Gupta's Account
Dashboard
Edit dashboard
Upgrade account
Create resource
For you
Select an option
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Use Watson Assistant
Watson Assistant lets you build conversational interfaces into any application, device, or channel.
Popular
2 min
Use Watson Studio
Watson Studio provides a suite of tools and a collaborative environment for data scientists, developers and domain experts.
Popular
2 min
Build with Watson
Chatbots, insights, recognitizers, and more. Explore the AI platform for business.
Popular
3 min
IBM Watson Machine Learning
Deploy, monitor and optimize machine learning models quickly and easily. Leverage auto-generated APIs to infuse AI into your applications.
Popular
2 min
IBM Cloud status
View all
Recent support cases
View all
Planned maintenance
View all


Eco Tips
KPI Forecasting
Anomaly Detection
Sustainability Report
Policy Summarizer
Chat Assistant
Check for Anomalies
5 anomaly/anomalies found:
• Year: 2019 → Energy: 1100
• Year: 2020 → Energy: 1180
• Year: 2021 → Energy: 1250
• Year: 2022 → Energy: 1350
• Year: 2023 → Energy: 1450

app > data > {} feedback_log.json > {} 7


```
[
    {
        "timestamp": "2025-04-09T14:20:13.013642",
        "user": "abhijeet",
        "category": "waste management",
        "message": "Garbage bins in sector 5 are overflowing every day."
    },
    {
        "timestamp": "2025-04-10T07:01:13.630209",
        "user": "Vaisali",
        "category": "Energy",
        "message": "streets lights are On during Daylight"
    },
    {
        "timestamp": "2025-04-10T11:51:23.871276",
        "user": "Ankit",
        "category": "Water",
        "message": "water supply not available in sector 5 . noida"
    },
```


Predict KPI
Debug Response:
{
"predicted_year" : 2024
"kpi" : "energy"
"predicted_value" : 1528.67
}
Forecasted ENERGY for next year: 1528.67