Sustainable Smart City Assistant

Project Title: Sustainable Smart City Assistant Using IBM Granite LLM

Team Information

- Team ID: LTVIP2025TMID38464
- Team Size: 4 members
- Team Leader: Akshai Mutya
- Team Members: Chitturi Abhishek
  Ganisetti Vasantha Lakshmi
  Koruprolu Bangarraju

Table of Contents

---

1. Project Overview

1.1 Problem Statement

The "Sustainable Smart City Assistant Using IBM Granite LLM" is an AI-powered solution aimed at enhancing urban sustainability and governance. It leverages IBM's Granite Large Language Model to process and analyze city data efficiently. The assistant supports modules like policy summarization, citizen feedback analysis, KPI forecasting, and eco-friendly recommendations. It provides real-time insights for administrators to make data-driven decisions. Ultimately, it aims to create smarter, greener, and more responsive urban environments.

1.2 Key Problems Identified:

1. **Limited Citizen Engagement**
    - Citizens lack accessible channels to provide feedback on city services
    - Government struggles to collect and analyze citizen input systematically
    - Communication gap between city officials and residents
    - Delayed response to community concerns and suggestions
2. **Environmental Monitoring Gaps**
    - Insufficient real-time monitoring of air quality, water usage, and energy consumption
    - Lack of predictive analytics for environmental trends
    - Limited awareness among citizens about sustainable practices
    - Difficulty in tracking progress toward sustainability goals
3. **Policy Document Management Issues**
    - Large volumes of policy documents difficult to search and retrieve
    - Time-consuming manual processes for policy analysis
    - Lack of semantic search capabilities for relevant information
    - Limited accessibility for citizens to understand policy implications
4. **Data-Driven Decision Making Deficits**
    - Absence of integrated KPI monitoring systems
    - Limited forecasting capabilities for city metrics
    - Reactive rather than proactive city management
    - Insufficient anomaly detection for early intervention
5. **Resource Optimization Challenges**
    - Inefficient allocation of city resources
    - Limited visibility into departmental performance
    - Lack of comprehensive reporting mechanisms
    - Difficulty in measuring sustainability impact

1.3 Target Audience Impact

- City Officials: Need intelligent tools for data-driven decision making
- Citizens: Require accessible platforms for engagement and information
- Policy Makers: Need efficient document management and analysis tools
- Environmental Agencies: Require real-time monitoring and reporting capabilities

1.4 Proposed Solution

1.4.1 Introduction

The *Sustainable Smart City Assistant* is an AI-driven solution designed to help city administrators and citizens foster environmentally conscious, data-informed urban development. Powered by IBM Granite LLM, the system uses advanced language understanding to analyze city-wide data, engage citizens, and support sustainable decision-making. By integrating multiple smart modules, the assistant transforms raw civic data into actionable insights and eco-friendly practices, aiming to build greener and more responsive cities.

1.4.2 Objectives

- **Primary Goal**:

  - To develop an intelligent assistant that leverages IBM Granite LLM for enhancing sustainable urban governance.

  - To automate key administrative tasks like policy summarization, citizen feedback analysis, and KPI forecasting for smart cities.

- **Secondary Goals**:

  - To encourage eco-friendly habits among citizens through AI-generated sustainability tips.

  - To build a real-time, interactive interface enabling smooth communication between citizens and administrators.

  - To ensure scalability and accessibility of the system using cloud-native technologies like IBM Cloud and Streamlit.

1.4.3 Key Features

- **Interactive Chat Assistant** powered by IBM Watsonx Granite LLM
- **Real-time KPI Dashboard** with sustainability metrics
- **Citizen Feedback System** for community engagement
- **Policy Document Search** using semantic vector search
- **Eco-friendly Tips Generator** for environmental awareness
- **Anomaly Detection** for unusual patterns in city data
- **Forecasting Models** for predictive analytics

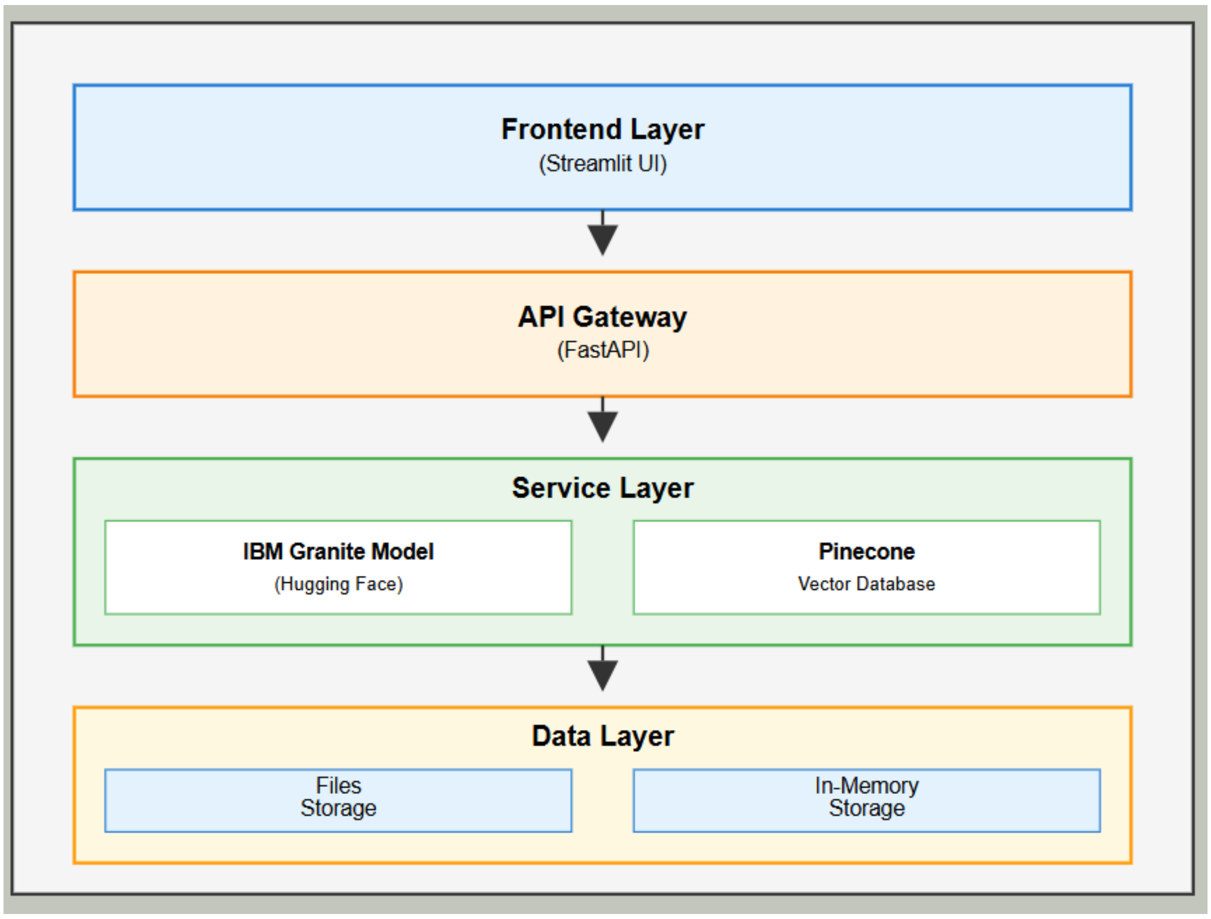
- **Comprehensive Report Generation** for city officials

---

2. System Architecture

2.1 Architecture Overview

The system follows a **microservices architecture** with clear separation between frontend and backend components:



2.2 Component Interaction Flow

1. **User Input**: Users interact through Streamlit dashboard
2. **API Gateway**: FastAPI routes requests to appropriate services
3. **AI Processing**: IBM HuggingFace processes natural language queries
4. **Vector Search**: Pinecone handles semantic document retrieval
5. **Data Analysis**: ML models process KPI data for insights
6. **Response**: Results rendered in user-friendly interface

---

3. Technology Stack

3.1 Backend Technologies

- **FastAPI**: Modern web framework for building APIs
- **Python 3.8+**: Core programming language
- **Pydantic**: Data validation and settings management
- **Uvicorn**: ASGI server for FastAPI

3.2 Frontend Technologies

- **Streamlit**: Interactive web application framework
- **Streamlit-Option-Menu**: Enhanced navigation components
- **Custom CSS**: Styling and UI enhancements

3.3 AI and ML Technologies

- **IBM Granite With HuggingFace**: Enterprise-grade AI platform
- **Granite LLM**: Large language model for text generation
- **Sentence Transformers**: Text embedding generation
- **Scikit-learn**: Machine learning algorithms
- **Pandas**: Data manipulation and analysis

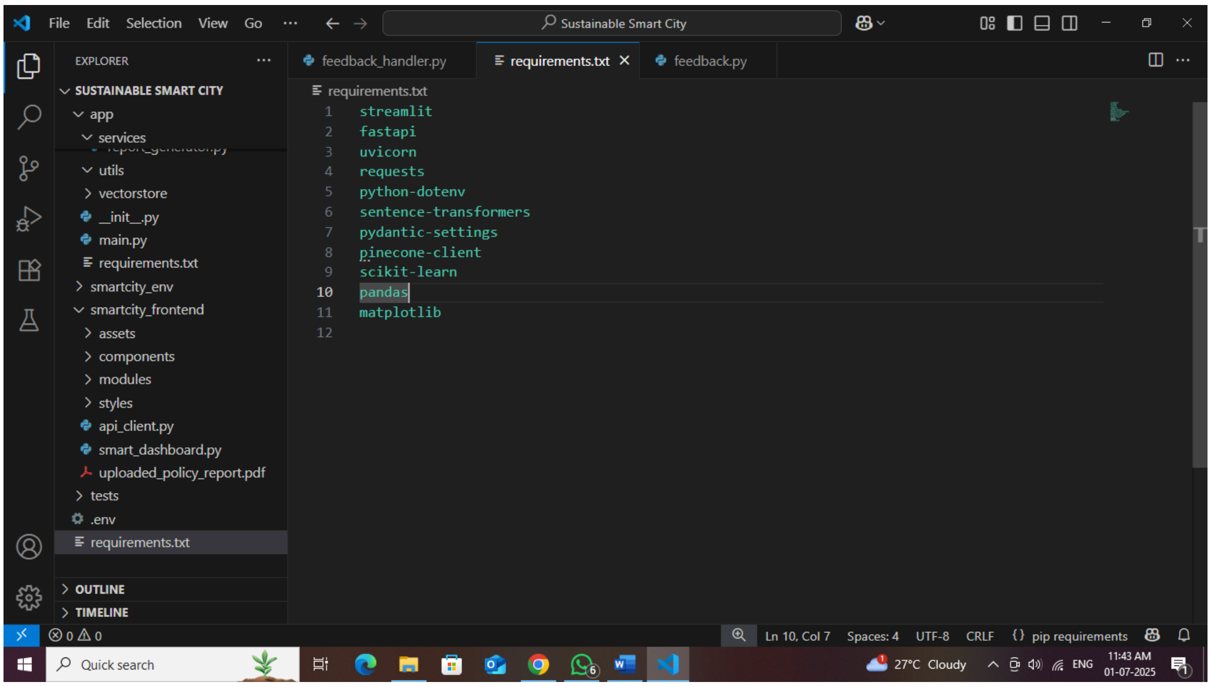
3.4 Database and Storage

- **Pinecone**: Vector database for semantic search
- **File System**: Local storage for documents and data
- **In-Memory Storage**: Session-based data management

---

4. Project Structure

4.1 Directory Organization



```
SUSTAINABLE SMART CITY
  app
    services
    utils
    vectorstore
    __init__.py
    main.py
    requirements.txt
  smartcity_env
  smartcity_frontend
    assets
    components
    modules
    styles
    api_client.py
    smart_dashboard.py
    uploaded_policy_report.pdf
  tests
  .env
  requirements.txt
```

```
streamlit
fastapi
uvicorn
requests
python-dotenv
sentence-transformers
pydantic-settings
pinecone-client
scikit-learn
pandas
matplotlib
```

4.2 Key Configuration Files

Environment Configuration (.env)

Based on the details configuration model of project ID ,url,pincone key,id,metric,and IBM Granite model.

```
# Application Settings
DEBUG=True
API_HOST=127.0.0.1
API_PORT=5000
FRONTEND_PORT=8501
```

---

5. Implementation Details

5.1 Backend Implementation

5.1.1 FastAPI Main Application

The main application (app/main.py) serves as the entry point for all API requests:

- Configures CORS middleware for cross-origin requests
- Includes all router modules for modular organization
- Provides health check endpoints
- Handles error responses and logging

5.1.2 IBM Watsonx Integration

The granite_llm.py service handles all interactions with IBM Watsonx:

- **Authentication**: Secure API key management
- **Text Generation**: Multiple specialized functions for different use cases
- **Error Handling**: Robust error management with fallbacks
- **Response Processing**: Clean formatting of AI responses

Key Functions:

- ask_granite(): General chat functionality
- generate_summary(): Policy document summarization
- generate_eco_tip(): Environmental recommendations
- generate_city_report(): Comprehensive sustainability reports

5.1.3 Vector Database Integration

The pinecone_client.py manages semantic search capabilities:

- **Index Management**: Automatic index creation and configuration
- **Vector Operations**: Update and query operations

- **Embedding Integration**: Works with sentence-transformers
- **Error Recovery**: Handles connection issues gracefully

5.2 Frontend Implementation

5.2.1 Main Dashboard

The smart_dashboard.py serves as the central hub:

- **Navigation System**: Sidebar with option menu for page routing
- **Responsive Design**: Optimized for different screen sizes
- **Custom Styling**: CSS enhancements for professional appearance
- **State Management**: Session state handling for user interactions

5.2.2 Component Architecture

Each UI component is modularized for reusability:

- **Summary Cards**: Styled KPI display components
- **Chat Interface**: Real-time conversation handling
- **Forms**: User input collection and validation
- **Data Visualization**: Charts and metrics display

---

6. Development Workflow

6.1 Phase-by-Phase Development

Phase 1: Project Initialization

Objectives Completed:

- Created modular folder structure
- Implemented environment configuration
- Set up dependency management
- Established coding standards

Key Deliverables:

- Project structure with clear separation of concerns
- Configuration management system
- Development environment setup

Fig: Project Folder Structure

Phase 2: IBM HuggingFace Integration

Objectives Completed:

- Configured HuggingFace API credentials

- Implemented LLM service wrapper
- Created specialized AI functions
- Tested endpoint connectivity

Key Deliverables:

- Granite LLM service implementation
- API testing and validation
- Error handling mechanisms

Phase 3: Backend API Development

Objectives Completed:

- Developed modular router architecture
- Implemented RESTful API endpoints
- Added request/response validation
- Created Swagger documentation

Key Deliverables:

- 8 comprehensive API routers
- Input validation and error handling
- Auto-generated API documentation

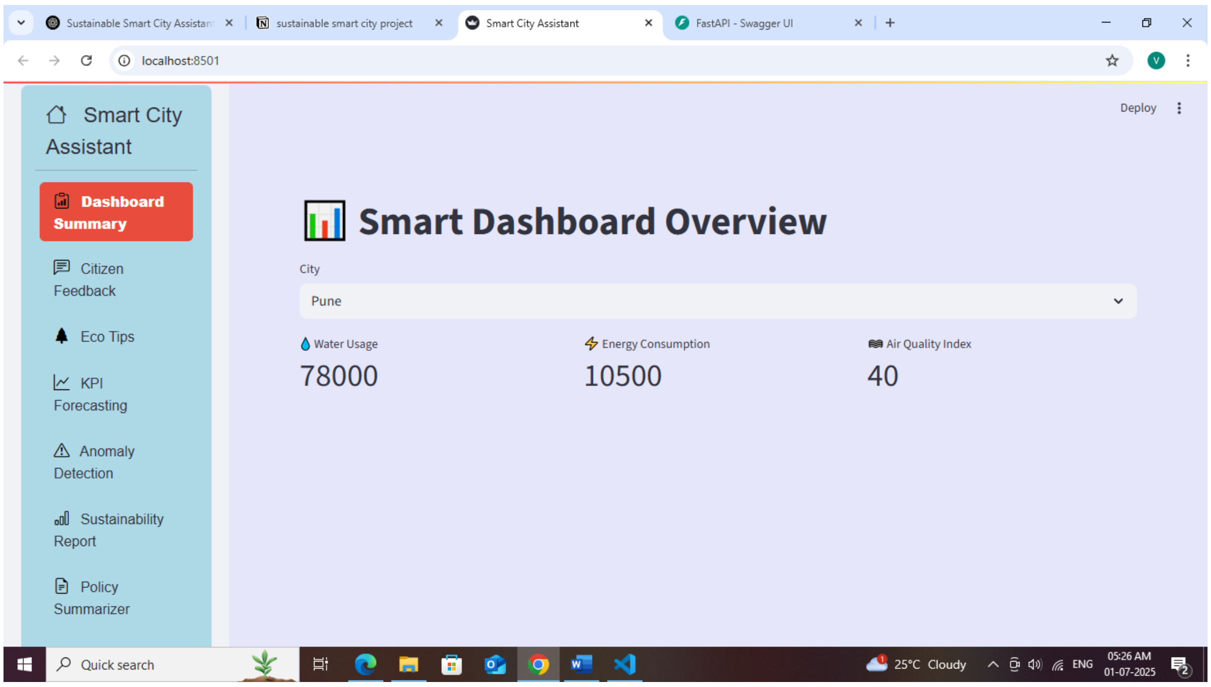
Phase 4: Frontend UI Design

Objectives Completed:

- Built responsive Streamlit interface
- Created reusable UI components
- Implemented custom styling
- Added interactive navigation

Key Deliverables:

- Professional dashboard interface
- Component-based architecture
- Enhanced user experience features
- Component – based architecture of IBM granite and llm language models
- Pinecone credentials for organizing keys and summary data
- IBM Granite watsonsx key and model efficient usage.



Smart Dashboard Overview

City: Pune

Water Usage	Energy Consumption	Air Quality Index
78000	10500	40

Fig: Frontend Main Page

Phase 5: Vector Database Integration

Objectives Completed:

- Configured Pinecone vector database
- Implemented document embedding
- Created semantic search functionality
- Added retrieval mechanisms

Key Deliverables:

- Document embedding pipeline
- Semantic search capabilities
- Policy document management

Phase 6: Machine Learning Features

Objectives Completed:

- Implemented KPI forecasting
- Added anomaly detection
- Created predictive models
- Built data analysis pipelines

Key Deliverables:

- Forecasting algorithms
- Anomaly detection system
- Data visualization components

Phase 7: Report Generation

Objectives Completed:

- AI-powered report generation
- Custom prompt engineering
- Multi-format output support
- Dynamic content creation

Key Deliverables:

- Automated report generation
- Customizable report templates
- Export functionality

Phase 8: Integration and Testing

Objectives Completed:

- End-to-end system integration
- Comprehensive testing suite
- Performance optimization
- User acceptance testing

Key Deliverables:

- Fully integrated system
- Testing documentation
- Performance benchmarks

---

7. Setup and Installation

7.1 Prerequisites

- Python 3.8 or higher
- IBM Granite model with HuggingFace access
- Git for version control

7.2 Installation Steps

Step 2: Dependency Installation

```
# Install required packages
pip install -r requirements.txt
```

Step 3: Configuration

1. **Create .env file** with your API credentials
2. **Configure IBM Watsonx** settings

3. **Set up Pinecone** vector database
4. **Verify configurations** using test scripts

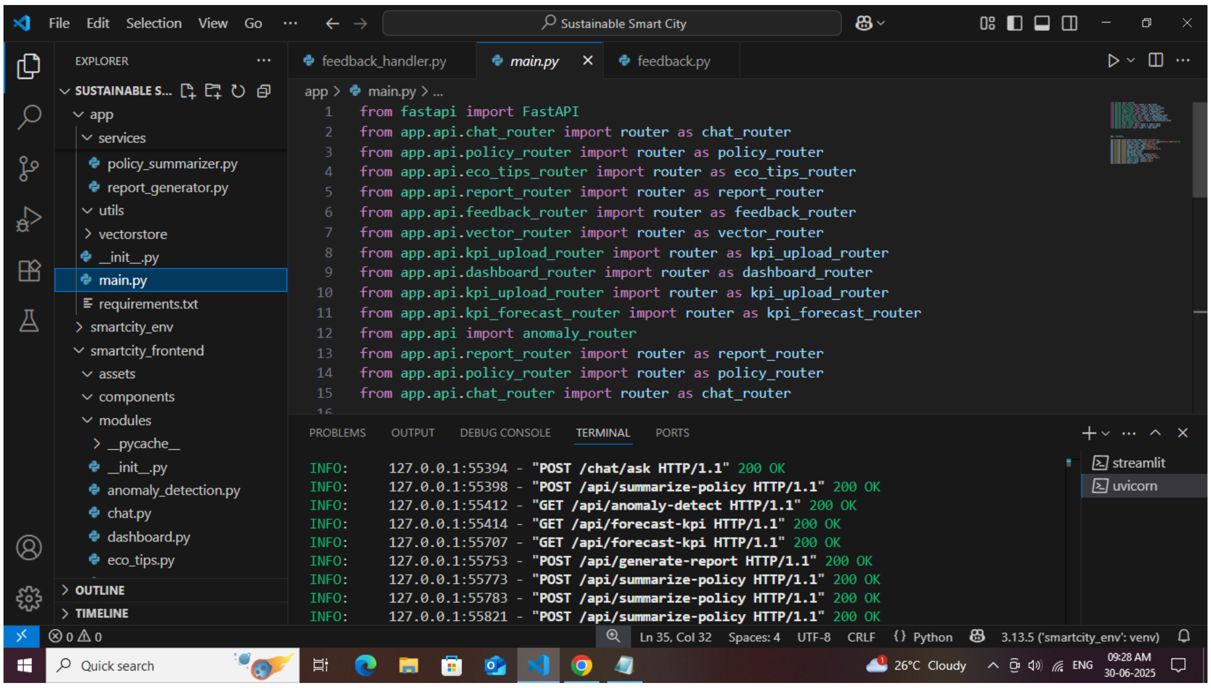
Step 4: Application Launch

# Method 1: Use the launcher (recommended)

python run_app.py

# Method 2: Manual startup

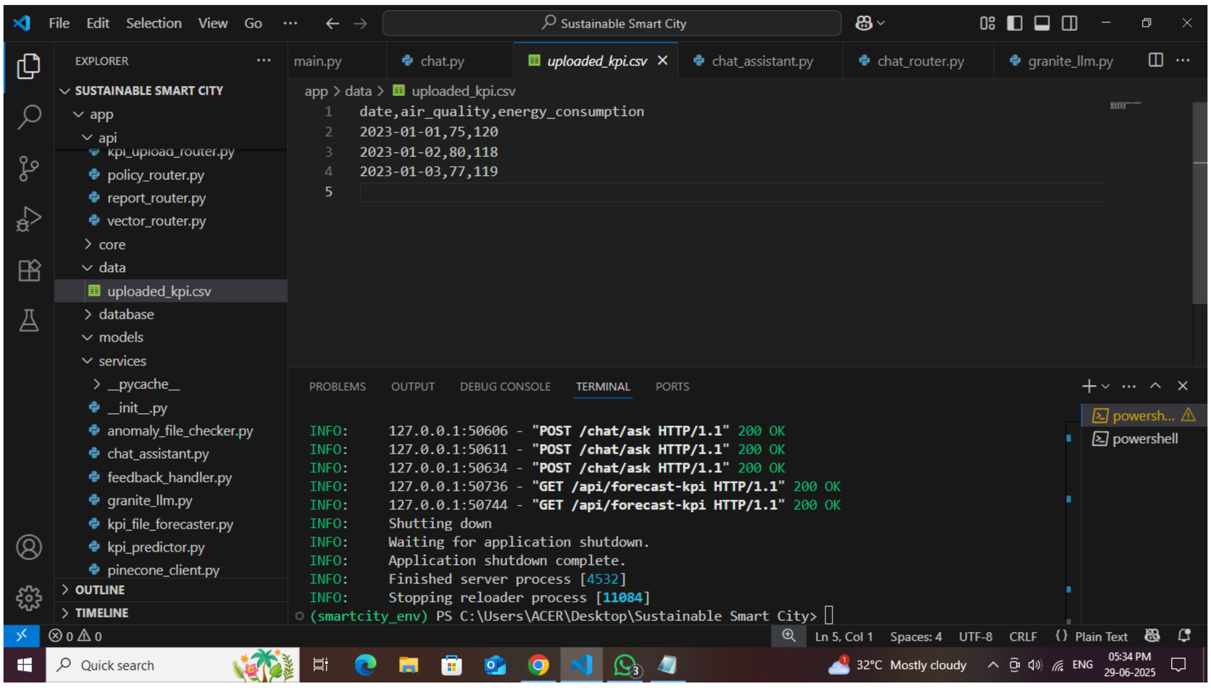
# Terminal 1 – Backend :::



```
app > main.py > ...
from fastapi import FastAPI
from app.api.chat_router import router as chat_router
from app.api.policy_router import router as policy_router
from app.api.eco_tips_router import router as eco_tips_router
from app.api.report_router import router as report_router
from app.api.feedback_router import router as feedback_router
from app.api.vector_router import router as vector_router
from app.api.kpi_upload_router import router as kpi_upload_router
from app.api.dashboard_router import router as dashboard_router
from app.api.kpi_upload_router import router as kpi_upload_router
from app.api.kpi_forecast_router import router as kpi_forecast_router
from app.api import anomaly_router
from app.api.report_router import router as report_router
from app.api.policy_router import router as policy_router
from app.api.chat_router import router as chat_router
```

```
INFO:     127.0.0.1:55394 - "POST /chat/ask HTTP/1.1" 200 OK
INFO:     127.0.0.1:55398 - "POST /api/summarize-policy HTTP/1.1" 200 OK
INFO:     127.0.0.1:55412 - "GET /api/anomaly-detect HTTP/1.1" 200 OK
INFO:     127.0.0.1:55414 - "GET /api/forecast-kpi HTTP/1.1" 200 OK
INFO:     127.0.0.1:55707 - "GET /api/forecast-kpi HTTP/1.1" 200 OK
INFO:     127.0.0.1:55753 - "POST /api/generate-report HTTP/1.1" 200 OK
INFO:     127.0.0.1:55773 - "POST /api/summarize-policy HTTP/1.1" 200 OK
INFO:     127.0.0.1:55783 - "POST /api/summarize-policy HTTP/1.1" 200 OK
INFO:     127.0.0.1:55821 - "POST /api/summarize-policy HTTP/1.1" 200 OK
```

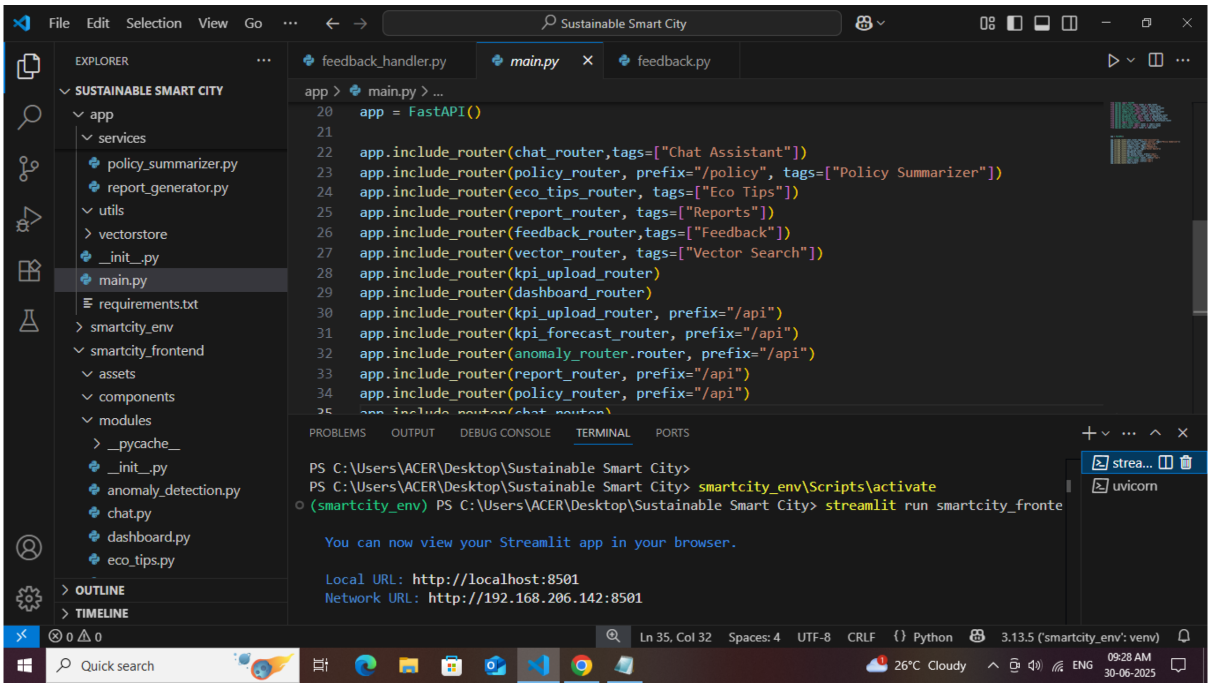
In backend I upload KPI Data as this image depicts :



```
app > data > uploaded_kpi.csv
date,air_quality,energy_consumption
2023-01-01,75,120
2023-01-02,80,118
2023-01-03,77,119
```

```
INFO:     127.0.0.1:50606 - "POST /chat/ask HTTP/1.1" 200 OK
INFO:     127.0.0.1:50611 - "POST /chat/ask HTTP/1.1" 200 OK
INFO:     127.0.0.1:50634 - "POST /chat/ask HTTP/1.1" 200 OK
INFO:     127.0.0.1:50736 - "GET /api/forecast-kpi HTTP/1.1" 200 OK
INFO:     127.0.0.1:50744 - "GET /api/forecast-kpi HTTP/1.1" 200 OK
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [4532]
INFO:     Stopping reloader process [11084]
(smartcity_env) PS C:\Users\ACER\Desktop\Sustainable Smart City>
```

# Terminal 2 - Frontend::: https://localhost:5000/api



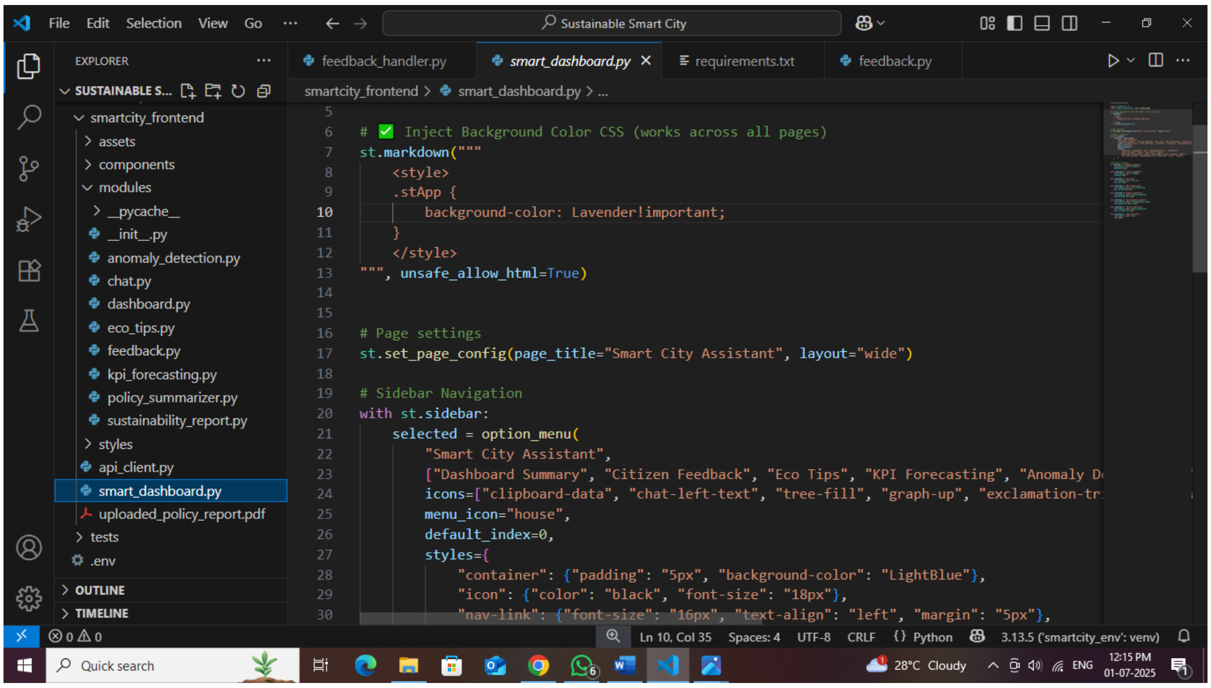
```python
app = FastAPI()

app.include_router(chat_router,tags=["Chat Assistant"])
app.include_router(policy_router, prefix="/policy", tags=["Policy Summarizer"])
app.include_router(eco_tips_router, tags=["Eco Tips"])
app.include_router(report_router, tags=["Reports"])
app.include_router(feedback_router,tags=["Feedback"])
app.include_router(vector_router, tags=["Vector Search"])
app.include_router(kpi_upload_router)
app.include_router(dashboard_router)
app.include_router(kpi_upload_router, prefix="/api")
app.include_router(kpi_forecast_router, prefix="/api")
app.include_router(anomaly_router.router, prefix="/api")
app.include_router(report_router, prefix="/api")
app.include_router(policy_router, prefix="/api")
```

```
PS C:\Users\ACER\Desktop\Sustainable Smart City>
PS C:\Users\ACER\Desktop\Sustainable Smart City> smartcity_env\Scripts\activate
(smartcity_env) PS C:\Users\ACER\Desktop\Sustainable Smart City> streamlit run smartcity_fronte

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.206.142:8501
```

Fig : Application Launch

UPLOAD POLICY DATA:

Upload policy data in frontend :



```python

# ✅ Inject Background Color CSS (works across all pages)
st.markdown("""
    <style>
    .stApp {
        background-color: Lavender!important;
    }
    </style>
""", unsafe_allow_html=True)


# Page settings
st.set_page_config(page_title="Smart City Assistant", layout="wide")

# Sidebar Navigation
with st.sidebar:
    selected = option_menu(
        "Smart City Assistant",
        ["Dashboard Summary", "Citizen Feedback", "Eco Tips", "KPI Forecasting", "Anomaly D
        icons=["clipboard-data", "chat-left-text", "tree-fill", "graph-up", "exclamation-tr
        menu_icon="house",
        default_index=0,
        styles={
            "container": {"padding": "5px", "background-color": "LightBlue"},
            "icon": {"color": "black", "font-size": "18px"},
            "nav-link": {"font-size": "16px", "text-align": "left", "margin": "5px"},
```

8. Features and Functionality

8.1 Dashboard Overview

Description: Central hub displaying key city sustainability metrics **Components**:

- Real-time KPI cards (Air Quality, Water Usage, Energy Consumption, Waste Recycling)
- City selection dropdown for comparative analysis
- Interactive metric visualization
- Trend indicators and alerts

8.2 Chat Assistant

Description: AI-powered conversational interface for city-related queries **Capabilities**:

- Natural language processing using Granite LLM
- Context-aware responses about urban sustainability
- Multi-turn conversation support
- Real-time response generation

8.3 Citizen Feedback System

Description: Community engagement platform for citizen input **Features**:

- Categorized feedback submission
- Contact information collection
- Feedback tracking and management
- Response analytics and reporting

8.4 Eco-Tips Generator

Description: Personalized environmental recommendations **Functionality**:

- Topic-based tip generation
- AI-powered content creation
- Actionable sustainability advice
- Category-specific recommendations

8.5 Policy Document Search

Description: Semantic search through policy documents **Technology**:

- Vector-based document retrieval
- Natural language query processing
- Relevance ranking and scoring

- Document summarization capabilities

8.6 KPI Analysis and Forecasting

Description: Predictive analytics for city metrics **Features**:

- Historical data analysis
- Trend forecasting algorithms
- Anomaly detection systems
- Performance benchmarking

8.7 Report Generation

Description: Automated sustainability reporting **Capabilities**:

- AI-generated comprehensive reports
- Custom prompt engineering
- Multi-format output support
- Data-driven insights and recommendations

---

9. API Documentation

9.1 Chat Endpoints

POST /api/chat/ask

Description: Submit chat queries to AI assistant

Request Body: {"message": "string"}

Response: {"response": "string", "status": "string"}

9.2 Feedback Endpoints

POST /api/feedback/submit

Description: Submit citizen feedback

```
Request Body: {
  "name": "string",
  "category": "string",
  "message": "string",
}
```

Response: {"message": "string", "status": "string", "feedback_id": "string"}

GET /api/feedback/list

Description: Retrieve all feedback entries

Response: {"feedback": [...], "count": number}

9.3 Eco-Tips Endpoints

GET /api/eco-tips/generate?topic={topic}

Description: Generate eco-friendly tips for specified topic

Parameters: topic (string) - The topic for which to generate tips

Response: {"tips": "string", "topic": "string", "status": "string"}

9.4 Policy Endpoints

POST /api/policy/summarize

Description: Summarize uploaded policy documents

Request Body: {"text": "string"}

Response: {"summary": "string", "status": "string"}

POST /api/vector/search

Description: Semantic search through policy documents

Request Body: {"query": "string", "top_k": number}

Response: {"results": [...], "status": "string"}

9.5 KPI Endpoints

POST /api/kpi/upload

Description: Upload KPI data for analysis

Request Body: Form data with CSV file

Response: {"message": "string", "analysis": {...}, "status": "string"}

POST /api/kpi/forecast

Description: Generate forecasting for KPI data

Request Body: {"data": [...], "periods": number}

Response: {"forecast": [...], "metrics": {...}, "status": "string"}

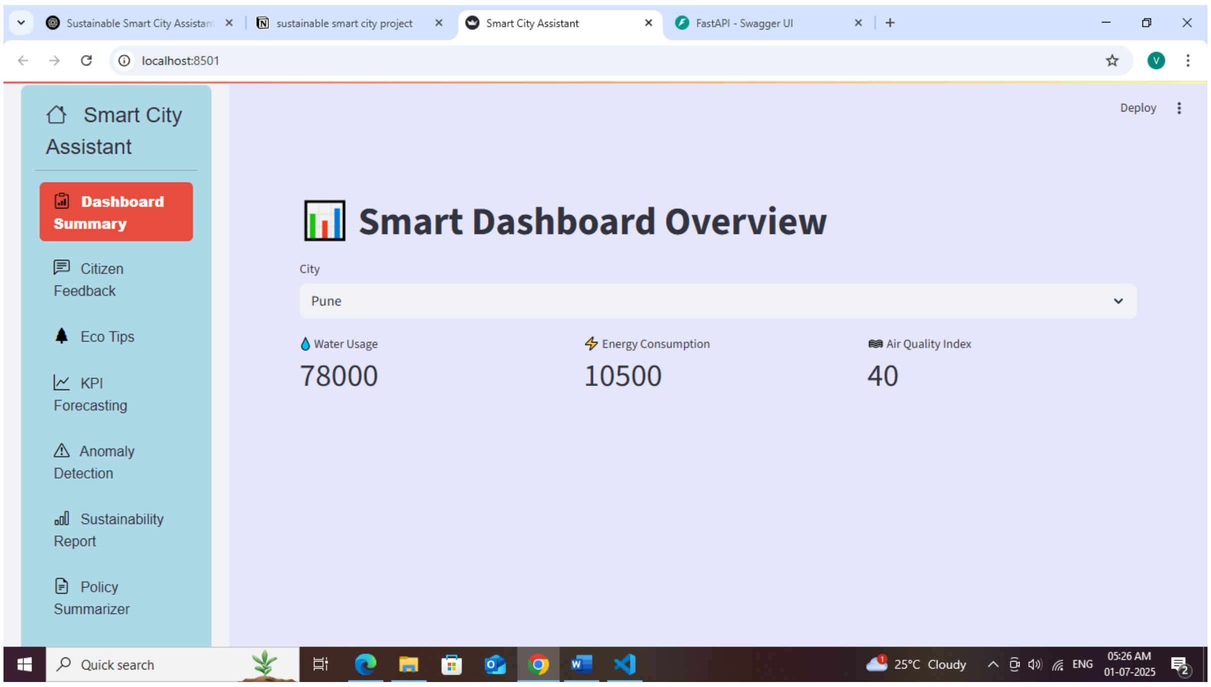
---

10. Screenshots and Results

10.1 Dashboard Interface

The main dashboard provides an intuitive overview of city sustainability metrics:

- **KPI Cards**: Visually appealing metric displays with trend indicators
- **City Selection**: Dropdown for comparative analysis across different cities
- **Real-time Updates**: Dynamic data refresh capabilities
- **Responsive Design**: Optimized for desktop and mobile viewing

- 

📊 Smart Dashboard Overview

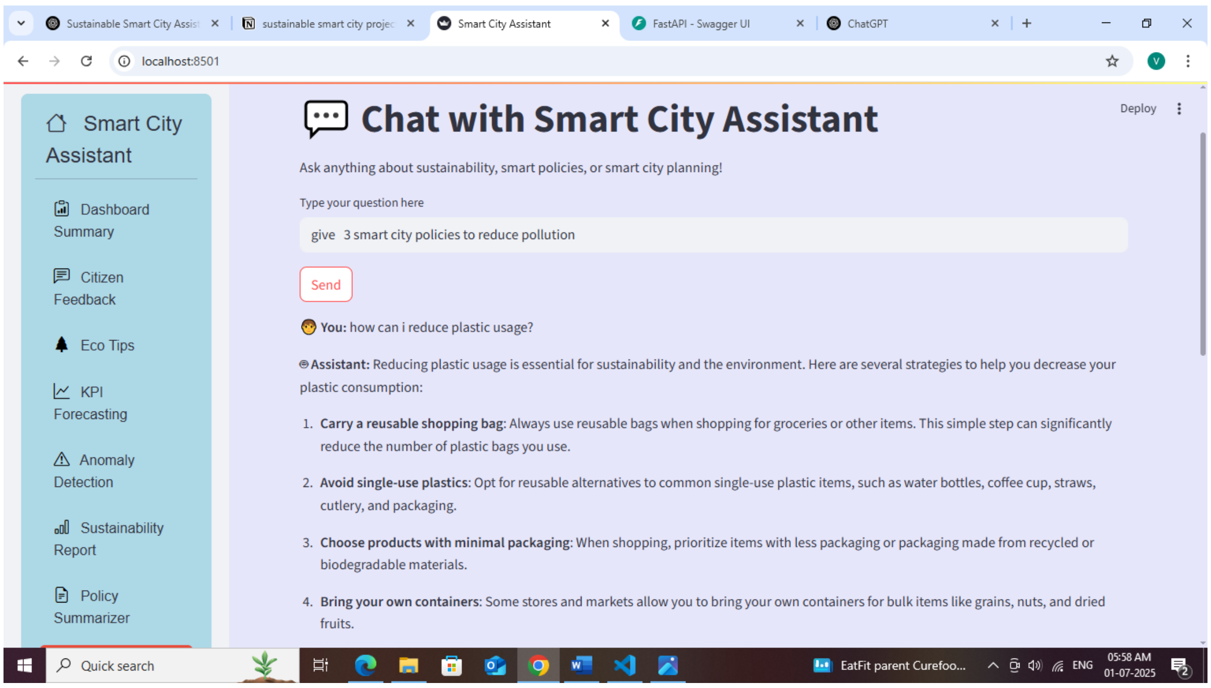
City

Pune

Water Usage	Energy Consumption	Air Quality Index
78000	10500	40

Fig: Dashboard

- **Chat Assistant eco friendly answer due to use of IBM Granite LLM :**
- **ibm/granite-3-3-8b-instruct**



💬 Chat with Smart City Assistant

Ask anything about sustainability, smart policies, or smart city planning!

Type your question here

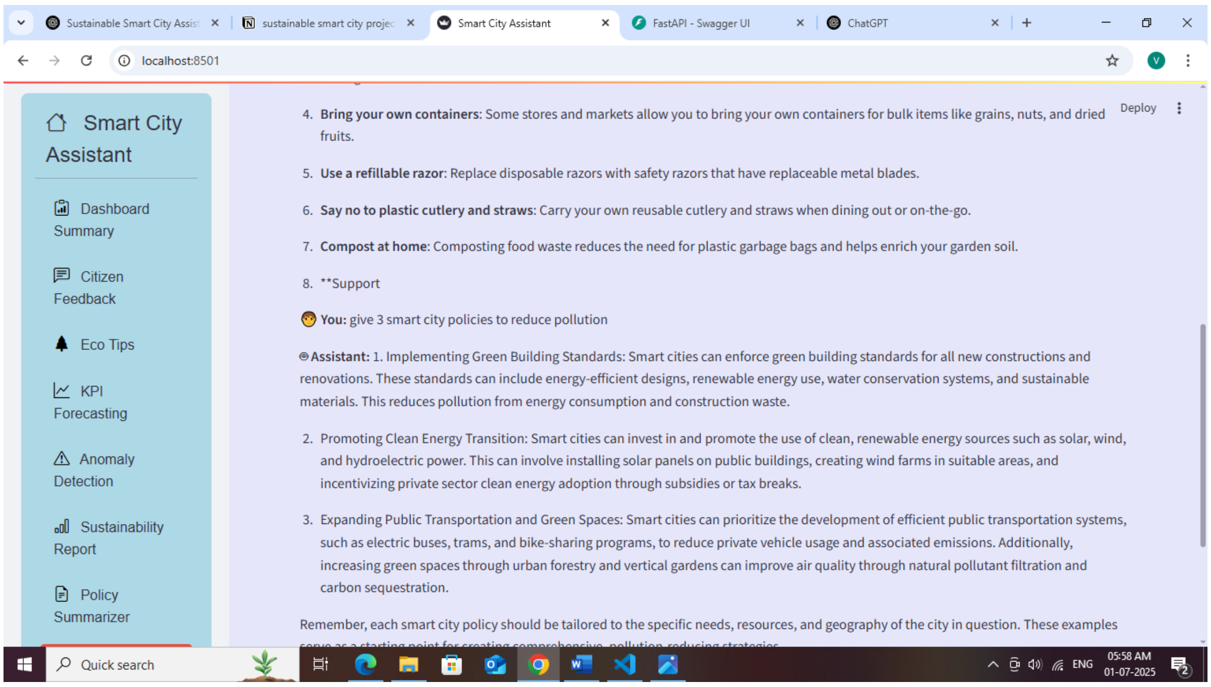
give 3 smart city policies to reduce pollution

You: how can i reduce plastic usage?

Assistant: Reducing plastic usage is essential for sustainability and the environment. Here are several strategies to help you decrease your plastic consumption:

1. **Carry a reusable shopping bag**: Always use reusable bags when shopping for groceries or other items. This simple step can significantly reduce the number of plastic bags you use.
2. **Avoid single-use plastics**: Opt for reusable alternatives to common single-use plastic items, such as water bottles, coffee cup, straws, cutlery, and packaging.
3. **Choose products with minimal packaging**: When shopping, prioritize items with less packaging or packaging made from recycled or biodegradable materials.
4. **Bring your own containers**: Some stores and markets allow you to bring your own containers for bulk items like grains, nuts, and dried fruits.

10.2 Chat Assistant Interface :



4. **Bring your own containers**: Some stores and markets allow you to bring your own containers for bulk items like grains, nuts, and dried fruits.
5. **Use a refillable razor**: Replace disposable razors with safety razors that have replaceable metal blades.
6. **Say no to plastic cutlery and straws**: Carry your own reusable cutlery and straws when dining out or on-the-go.
7. **Compost at home**: Composting food waste reduces the need for plastic garbage bags and helps enrich your garden soil.
8. **Support

You: give 3 smart city policies to reduce pollution

Assistant: 1. Implementing Green Building Standards: Smart cities can enforce green building standards for all new constructions and renovations. These standards can include energy-efficient designs, renewable energy use, water conservation systems, and sustainable materials. This reduces pollution from energy consumption and construction waste.

2. Promoting Clean Energy Transition: Smart cities can invest in and promote the use of clean, renewable energy sources such as solar, wind, and hydroelectric power. This can involve installing solar panels on public buildings, creating wind farms in suitable areas, and incentivizing private sector clean energy adoption through subsidies or tax breaks.
3. Expanding Public Transportation and Green Spaces: Smart cities can prioritize the development of efficient public transportation systems, such as electric buses, trams, and bike-sharing programs, to reduce private vehicle usage and associated emissions. Additionally, increasing green spaces through urban forestry and vertical gardens can improve air quality through natural pollutant filtration and carbon sequestration.

Remember, each smart city policy should be tailored to the specific needs, resources, and geography of the city in question. These examples

- Chat with Smart City Assistant is quite interesting.

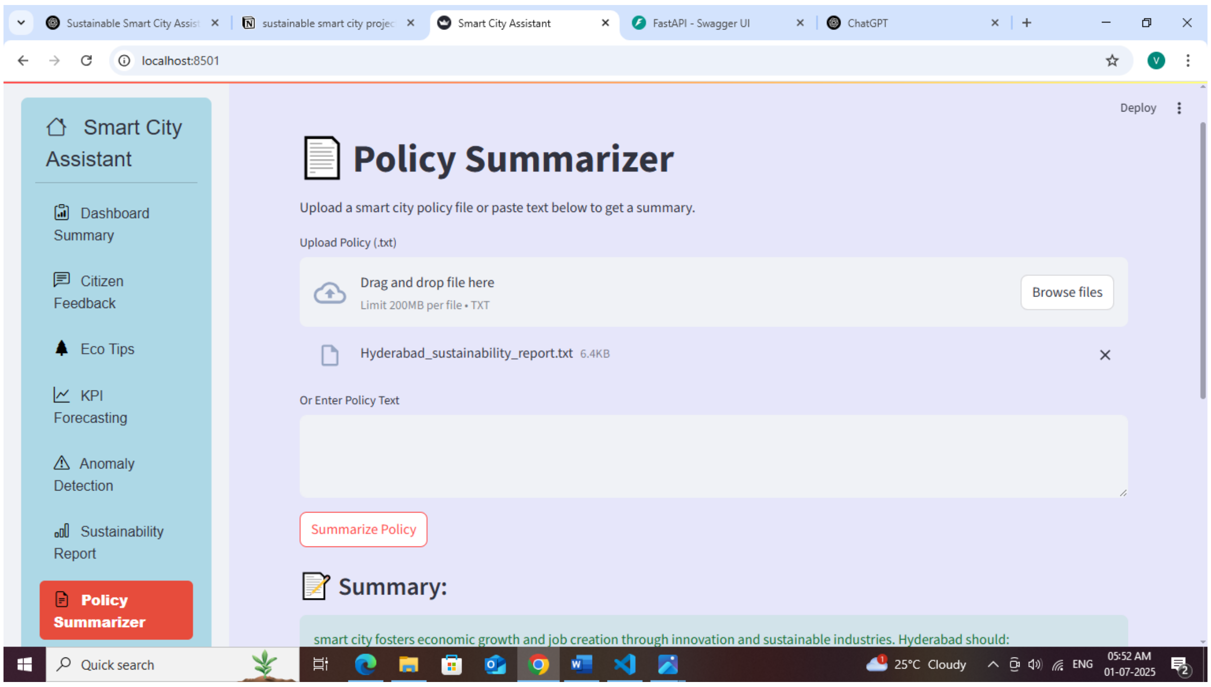
The conversational AI interface demonstrates:

- **Natural Language Processing**: Advanced query understanding
- **Context Awareness**: Maintains conversation context
- **Professional Responses**: Well-formatted, informative answers
- **Interactive Design**: User-friendly chat bubble interfac**Fig: Chat Assistant**

10.3 Policy Search Results

The semantic search functionality displays:

- **Relevance Ranking**: Results ordered by semantic similarity
- **Document Snippets**: Contextual preview of relevant content
- **Search Accuracy**: High precision in document retrieval
- **User Experience**: Fast and intuitive search interface
- User explore through policy search : approximate results.



Policy Summarizer

Upload a smart city policy file or paste text below to get a summary.

Upload Policy (.txt)

Drag and drop file here
Limit 200MB per file • TXT

Browse files

Hyderabad_sustainability_report.txt 6.4KB

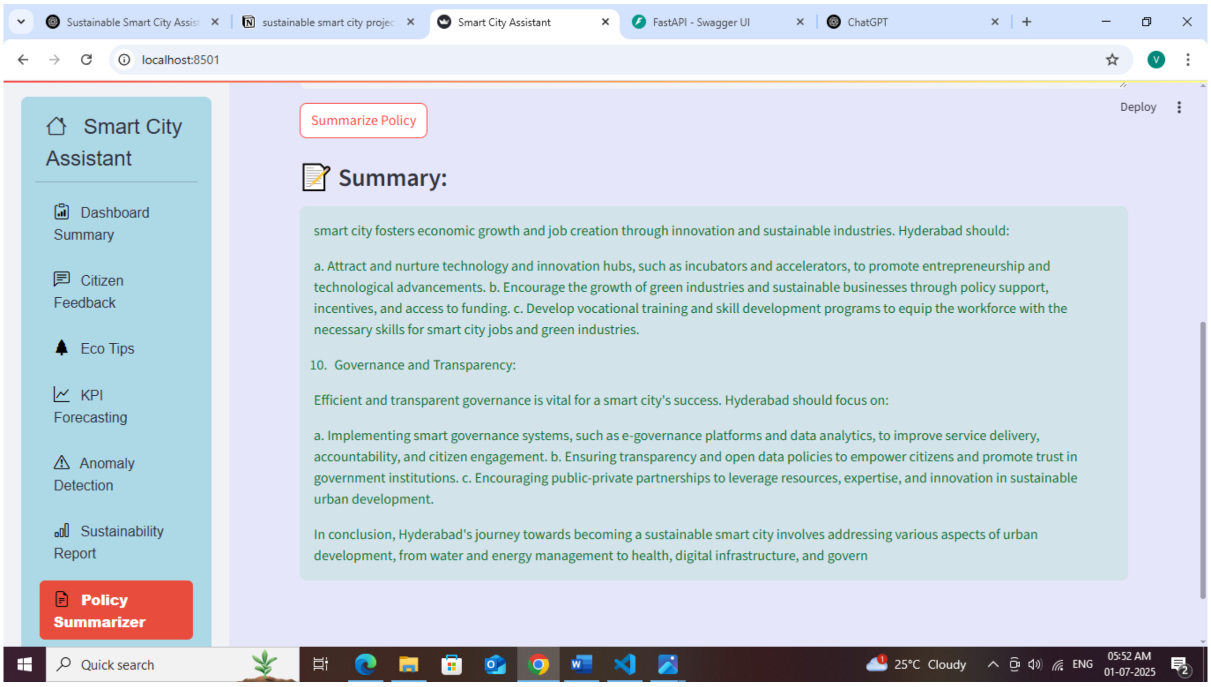
Or Enter Policy Text

Summarize Policy

Summary:

smart city fosters economic growth and job creation through innovation and sustainable industries. Hyderabad should:

Fig: Policy Search Results



Summary:

smart city fosters economic growth and job creation through innovation and sustainable industries. Hyderabad should:

a. Attract and nurture technology and innovation hubs, such as incubators and accelerators, to promote entrepreneurship and technological advancements. b. Encourage the growth of green industries and sustainable businesses through policy support, incentives, and access to funding. c. Develop vocational training and skill development programs to equip the workforce with the necessary skills for smart city jobs and green industries.

10. Governance and Transparency:

Efficient and transparent governance is vital for a smart city's success. Hyderabad should focus on:

a. Implementing smart governance systems, such as e-governance platforms and data analytics, to improve service delivery, accountability, and citizen engagement. b. Ensuring transparency and open data policies to empower citizens and promote trust in government institutions. c. Encouraging public-private partnerships to leverage resources, expertise, and innovation in sustainable urban development.

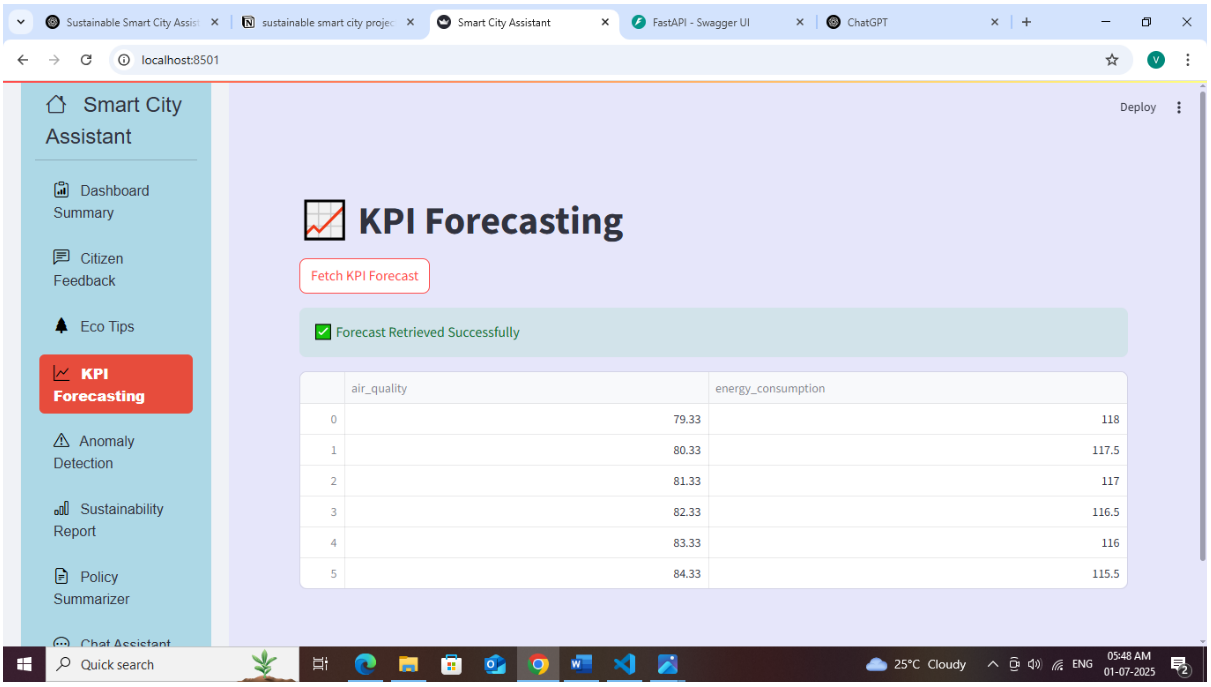
In conclusion, Hyderabad's journey towards becoming a sustainable smart city involves addressing various aspects of urban development, from water and energy management to health, digital infrastructure, and govern

- Policy summarizer gave summary both pdf format and as well as text format also

KPI Forecasting :

KPI Forecasting stands for **Key Performance Indicator Forecasting**. It uses **historical data** and **machine learning** models like **Linear Regression** to **predict future trends** in critical city metrics such as:

- Water Usage
- Energy Consumption
- Air Quality Index (AQI)



KPI Forecasting

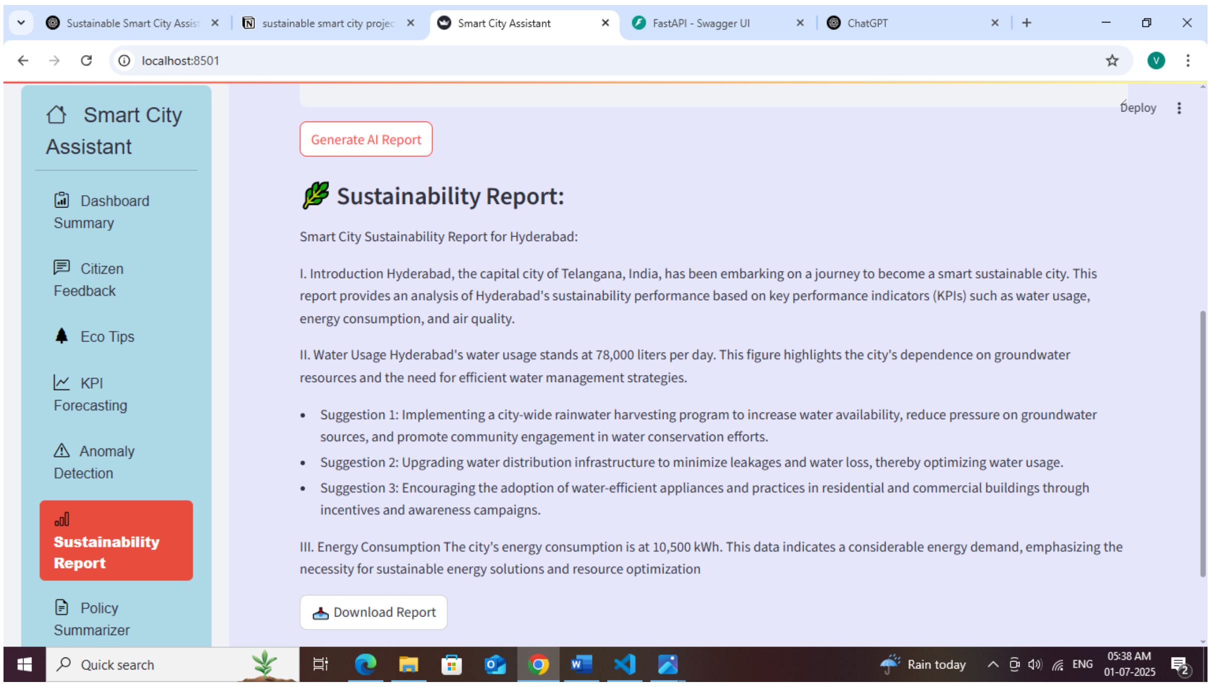
Forecast Retrieved Successfully

	air_quality	energy_consumption
0	79.33	118
1	80.33	117.5
2	81.33	117
3	82.33	116.5
4	83.33	116
5	84.33	115.5

- **Sustainability Report :**
- A Sustainability Report is an AI-generated summary that highlights the city's environmental performance and sustainable practices based on user input, metrics, and feedback.
- It provides an overview of how the city is progressing toward its smart and eco-friendly goals

To generate a **detailed, readable, and shareable** report that:

- Summarizes energy & water usage trends
- Reflects citizen feedback and actions taken
- Highlights green initiatives (e.g., solar panels, waste segregation)
- Assesses air quality, transport, and city infrastructure



Generate AI Report

Sustainability Report:

Smart City Sustainability Report for Hyderabad:

I. Introduction Hyderabad, the capital city of Telangana, India, has been embarking on a journey to become a smart sustainable city. This report provides an analysis of Hyderabad's sustainability performance based on key performance indicators (KPIs) such as water usage, energy consumption, and air quality.

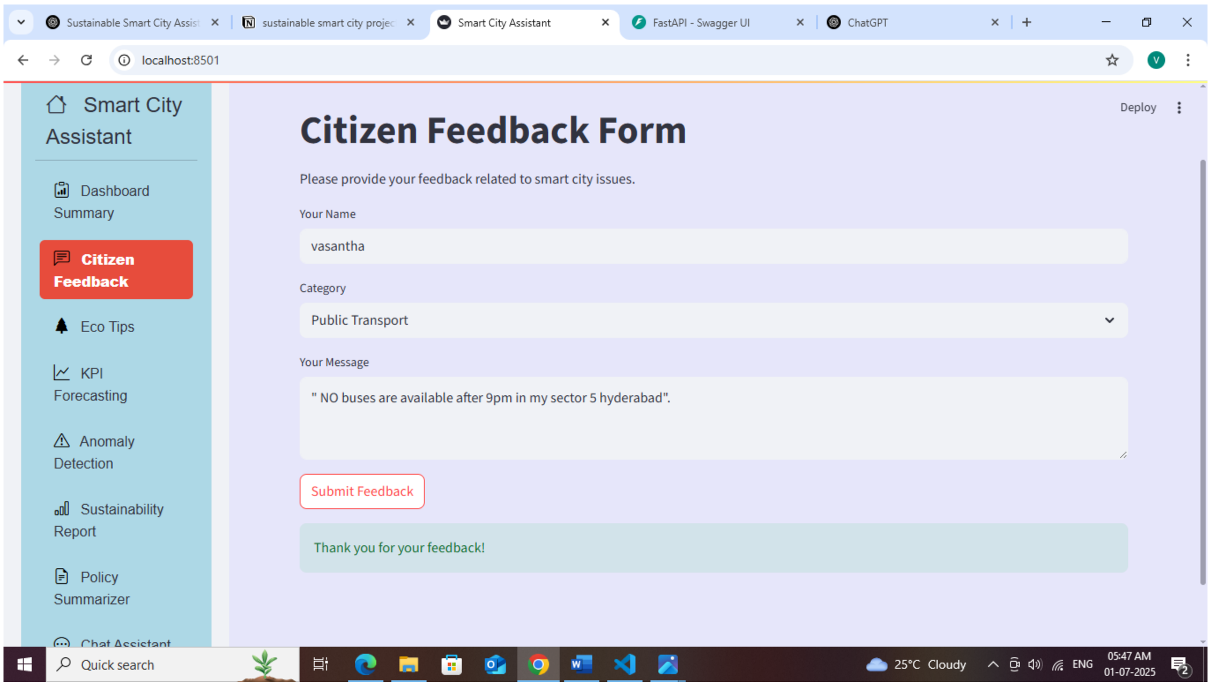
II. Water Usage Hyderabad's water usage stands at 78,000 liters per day. This figure highlights the city's dependence on groundwater resources and the need for efficient water management strategies.

- Suggestion 1: Implementing a city-wide rainwater harvesting program to increase water availability, reduce pressure on groundwater sources, and promote community engagement in water conservation efforts.
- Suggestion 2: Upgrading water distribution infrastructure to minimize leakages and water loss, thereby optimizing water usage.
- Suggestion 3: Encouraging the adoption of water-efficient appliances and practices in residential and commercial buildings through incentives and awareness campaigns.

III. Energy Consumption The city's energy consumption is at 10,500 kWh. This data indicates a considerable energy demand, emphasizing the necessity for sustainable energy solutions and resource optimization

Download Report

- **Citizen Feedback :**
- The **Citizen Feedback** module allows residents to **report issues** and **share suggestions** directly free access this website
  It's a two-way communication channel for creating a more **responsive and people-centric smart city**.



Citizen Feedback Form

Please provide your feedback related to smart city issues.

Your Name: vasantha

Category: Public Transport

Your Message: " NO buses are available after 9pm in my sector 5 hyderabad".

Submit Feedback

Thank you for your feedback!

11. Challenges and Solutions

11.1 Technical Challenges

Challenge 1: API Integration Complexity with IBM Granite (via Hugging Face) and Pinecone

Problem: Integrating multiple external services—such as IBM Granite models (via Hugging Face) and Pinecone vector DB—posed difficulties due to differing authentication mechanisms, SDK structures, and response formats.

Solution:

 Developed **modular wrapper services** for:

- IBM Granite model (accessed via Hugging Face transformers interface or REST API)
- Pinecone vector search engine

 Implemented **unified error handling** using middleware to catch, log, and gracefully manage API-specific failures.

 Introduced **fallback strategies**, such as:

- Switching to a backup embedding model when the Granite model API exceeds rate limits or returns errors
- Using cached results for Pinecone in the event of a timeout

 Standardized response formats using internal data classes (Pydantic / JSON schema) to normalize outputs regardless of source

 Authenticated using Hugging Face tokens for IBM Granite and API keys for Pinecone, abstracted behind a common security interface

Challenge 2: Real-time Data Processing

Problem: Handling real-time data updates and maintaining synchronization between frontend and backend.

Solution:

- Implemented asynchronous request handling
- Added caching mechanisms for frequently accessed data
- Created efficient data serialization processes
- Optimized API response times

Challenge 3: Vector Database Management

Problem: Managing document embeddings and ensuring efficient semantic search performance.

Solution:

- Optimized embedding generation pipeline
- Implemented batch processing for large documents
- Added index management utilities
- Created automated cleanup processes

11.2 Design Challenges

Challenge 1: User Experience Consistency

Problem: Maintaining consistent UI/UX across different components and pages.

Solution:

- Developed reusable component library
- Implemented standardized styling guidelines
- Created consistent navigation patterns
- Added responsive design principles

Challenge 2: Performance Optimization

Problem: Ensuring fast loading times and smooth user interactions.

Solution:

- Implemented lazy loading for components
- Optimized API calls and caching
- Minimized redundant operations
- Added loading indicators for better UX

---

12. Future Enhancements

12.1 Planned Features

1. **Mobile Application**: Native mobile app for iOS and Android
2. **Real-time Notifications**: Push notifications for critical alerts
3. **Advanced Analytics**: Machine learning-powered insights dashboard
4. **Multi-language Support**: Internationalization and localization
5. **Integration APIs**: Third-party service integrations (weather, traffic, etc.)

12.2 Scalability Improvements

1. **Database Migration**: Transition to PostgreSQL/MongoDB for production
2. **Microservices Architecture**: Further decomposition of services
3. **Load Balancing**: Implement horizontal scaling capabilities
4. **Caching Layer**: Redis implementation for improved performance

5. **Container Deployment**: Docker and Kubernetes deployment

12.3 AI Enhancements

1. **Custom Model Training**: Fine-tuned models for specific city domains
2. **Multi-modal AI**: Image and document processing capabilities
3. **Predictive Analytics**: Advanced forecasting algorithms
4. **Natural Language Understanding**: Enhanced query processing
5. **Automated Insights**: Proactive recommendations and alerts

---

13. Conclusion

13.1 Project Success Metrics

The Sustainable Smart City Assistant project has successfully achieved its primary objectives:

- **Functional AI Integration**: Successfully integrated IBM Watsonx Granite LLM for intelligent responses
- **Comprehensive Feature Set**: Delivered all 8 planned features with full functionality
- **User-Friendly Interface**: Created an intuitive and responsive web application
- **Scalable Architecture**: Implemented modular design for future enhancements
- **Technical Excellence**: Maintained high code quality and documentation standards

13.2 Key Achievements

1. **Advanced AI Integration**: Seamless integration of enterprise-grade AI services
2. **Full-Stack Development**: Complete end-to-end application development
3. **Modern Technology Stack**: Utilization of cutting-edge frameworks and tools
4. **Professional UI/UX**: High-quality user interface with custom styling
5. **Comprehensive Documentation**: Detailed technical and user documentation

13.3 Impact and Value

The project demonstrates significant value in:

- **Urban Governance**: Providing intelligent tools for city management
- **Citizen Engagement**: Facilitating better communication between citizens and government
- **Environmental Sustainability**: Promoting eco-friendly practices and awareness
- **Data-Driven Decision Making**: Enabling evidence-based policy decisions
- **Technology Innovation**: Showcasing practical AI applications in public sector

13.4 Lessons Learned

1. **API Integration Complexity**: The importance of robust error handling and fallback mechanisms
2. **User Experience Design**: The critical role of intuitive interface design in adoption
3. **Modular Architecture**: The benefits of well-structured, maintainable code organization
4. **Documentation Importance**: The value of comprehensive documentation for project sustainability
5. **Continuous Testing**: The necessity of thorough testing throughout development cycles

13.5 Final Recommendations

For organizations looking to implement similar AI-powered city management solutions:

1. **Start with Clear Objectives**: Define specific use cases and success metrics
2. **Invest in Quality APIs**: Choose reliable, enterprise-grade AI services
3. **Prioritize User Experience**: Focus on intuitive design and user feedback
4. **Plan for Scalability**: Design architecture with future growth in mind
5. **Maintain Documentation**: Keep comprehensive technical and user documentation
6. **Implement Monitoring**: Add comprehensive logging and monitoring systems
7. **Plan for Security**: Implement robust security measures from the beginning

The Sustainable Smart City Assistant represents a successful implementation of AI technology in urban governance, demonstrating the potential for intelligent systems to enhance city management and citizen engagement while promoting environmental sustainability.

Github Link : https://github.com/lakshmivasanthaa/Sustainable-Smart-City-Assistant-Using-IBM-Granite-LLM