Sustainable Smart City Assistant Using IBM Granite LLM

Project Title: Sustainable Smart City Assistant Using IBM Granite LLM

- **Team ID:** LTVIP2025TMID32124
- **Team Size:** 4 members
- **Team Leader:** Chintapalli Swetha
- **Team Members:** Bhatta Pujitha, Bonthu Deekshith Naidu, Battula Bala Sai

1. INTRODUCTION:

1.1 Project Overview Sustainable Smart City Assistant is an AI-driven citizen engagement platform built with IBM Granite LLM (via Watsonx) and modern cloud technologies. It aims to improve urban governance, public awareness, and sustainability by offering:

- Real-time policy summarization
- Citizen grievance reporting
- Eco-friendly recommendations
- Key performance forecasting
- Anomaly detection and traffic insights

1.2 Purpose To empower citizens and administrators with intelligent tools for:

- Understanding and summarizing public policies
- Reporting civic issues seamlessly
- Receiving eco-lifestyle guidance
- Forecasting urban KPIs
- Detecting anomalies (e.g., sudden traffic spikes, water leakage)

2.IDEATION PHASE:

2.1 Problem Statement: Modern cities lack smart platforms for transparent policy communication, real-time civic engagement, and personalized sustainability suggestions. Citizens often remain unaware of policies or eco-practices, and grievance redressal is slow and inefficient. Additionally, digital

illiteracy and lack of user-friendly platforms make it difficult for a large portion of the population to access vital information. The absence of a centralized system that integrates civic data, policy updates, environmental alerts, and public feedback leads to fragmented governance. Moreover, without predictive analytics and AI-driven insights, cities struggle to respond proactively to issues such as traffic congestion, energy waste, and environmental degradation.

2.2 Empathy Map Canvas Users: Citizens, Urban Planners, Smart City Officers Needs: Policy clarity, complaint system, eco-education, traffic updates, performance monitoring Pain Points: Policy overload, slow issue resolution, low eco-awareness, fragmented data

2.3 Brainstorming (Expanded with UI-Based Features):

1. Policy Search & Summarization:
   - Upload or paste policy documents (PDF/Text)
   - Get AI-generated summaries, key highlights, and bullet points
   - Ask questions about the document (Q&A mode)
   - Ideal for citizens, students, and policymakers
2. Citizen Reports:
   - Submit complaints using text, voice, or image (with OCR)
   - Auto-categorization using LLM prompts (e.g., “Garbage issue” or “Streetlight not working”)
   - Status tracking with email/SMS updates
   - Location tagging for effective routing
3. KPI Forecasting:
   - Forecast Key Performance Indicators like:
     - Traffic density
     - Electricity or water usage
     - Waste generation trends
   - Uses historical CSV/PDF data + LLM + basic regression
4. Eco Tips Engine:

- AI suggests eco-friendly lifestyle tips based on:
    - User location or interest (energy, waste, water, transport)
    - Time of day, season, or local events
- Smart nudges like “Turn off your geyser for 1 hour to save 2kWh”

5. Anomaly Detection:

- Detect sudden spikes in urban behavior like:
    - Unusual traffic patterns
    - Sharp rise in electricity/water usage
    - Excessive complaints in a zone
    - Uses uploaded sensor/CSV data + IBM LLM analysis

6. AskMe – AI Chat Assistant:

- Ask any smart city–related question
- Understand uploaded PDFs or policies
- Example: “What does the 2024 Water Act say about recycling?”
- Previous chat history saved in dashboard

7. Traffic & Routes Insights:

- View traffic congestion summaries
- Ask for “best route to reduce carbon footprint”
- Based on integrated city data and API suggestions

8. User Dashboard:

- Total complaints made
- Policies read/summarized
- Eco tips viewed or liked
- Chat interaction history

3.REQUIREMENT ANALYSIS:

3.1 Customer Journey:

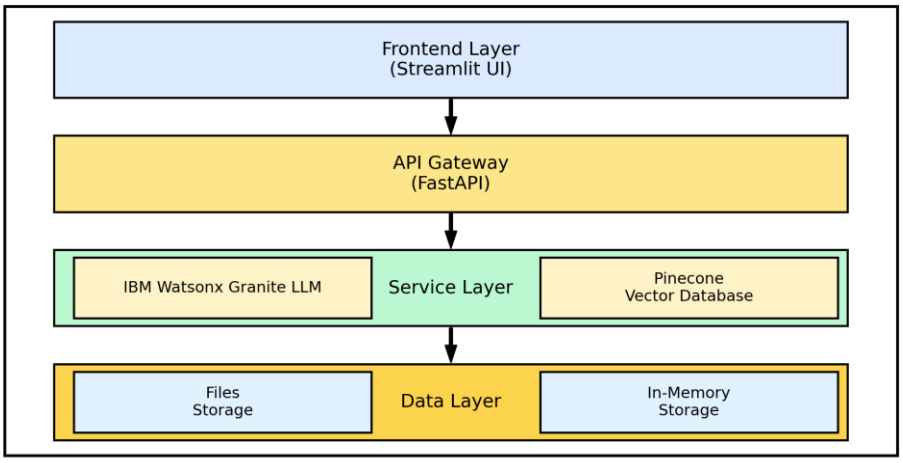
- User logs in securely (Firebase/JWT)

- Chooses a module: Policy, Report, Eco, Chat, etc.
- Uploads or enters relevant data
- IBM Granite LLM processes the input
- Output is presented interactively
- Dashboard updates and stores interaction

3.2 Requirements:

- ✓ Flask with multiple panels
- ✓ FastAPI backend with modular routers
- ✓ IBM Watsonx LLM via LangChain
- ✓ Storage: Local JSON or NoSQL

3.3 Data Flow Diagram:



3.4 **Tech Stack Layer Tools**:

```
# Cell 1: Install dependencies and setup
!pip install transformers torch accelerate huggingface_hub pandas numpy sci

# Import required libraries
import os
import torch
import pandas as pd
import numpy as np
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from datetime import datetime, timedelta
import json
import re
from flask import Flask, request, jsonify
from flask_cors import CORS
import threading
import time
from pyngrok import ngrok
```

4. PROJECT DESIGN:

4.1 Problem–Solution Fit The assistant solves urban issues by:

- Giving access to policies in understandable form
- Automating issue reporting
- Offering daily sustainability tips
- Forecasting civic KPIs
- Detecting irregularities in urban data

4.2 **Proposed Solutions:**

📜 Policy Search & Summarization

📢 Citizen Grievance Reports

🌿 Eco Tips Engine

📈 KPI Forecasting

⚠️ Anomaly Detector

🚦 Traffic & Routing Support

🤖 AskMe Chatbot

📊 Citizen Dashboard

4.3 **Architecture:**

- All frontend interactions → FastAPI backend
- Backend uses LangChain → IBM Granite model
- Output from LLM rendered in real-time in each module
- Activity logged in dashboards for insights

4.2 Key Configuration Files

- **Environment Configuration (.env)**
- # IBM Watsonx Configuration
- HUGGINGFACE_API_KEY=your_ibm_api_key_here
- HUGGING FACE_ID=your_project_id_here
- HUGGINGFACE_MODEL_ID=ibm/granite-13b-instruct-v2
- INDEX_NAME=smartcity-policies
- # Application Settings
- DEBUG=True
- API_HOST=127.0.0.1
- API_PORT=5000
- FRONTEND_PORT=8501

5. **PROJECT PLANNING & SCHEDULING Phase**:

- The project followed an Agile development lifecycle with clearly defined sprints and deliverables.
- Daily stand-up meetings were held to track progress and address blockers.
- A modular development approach ensured scalable and independent development of each feature.

- Gantt-style planning was used to map tasks and durations.
- Phase Duration Tasks
- Ideation → 2 days → Feature design and brainstorming.
- Backend Development → 5 days → Create API routes for modules: policy, reports, eco tips, chatbot, traffic insights.
- LLM Setup → 2 days → Integrate IBM Granite via LangChain and test prompts.
- Frontend UI → 4 days → Build responsive Streamlit interface with role-based views.
- Testing → 2 days → Perform unit, integration, and system testing.
- Documentation → 1 day → Create reports, architecture diagrams, and manuals.
- Deployment → 1 day → Host the complete solution on Streamlit/Vercel platform.
- Buffer Time 1 day Bug fixes and last-minute enhancements
- Roles Involved:
- Frontend Developer: UI/UX development
- Backend Developer: FastAPI integration and service modules
- LLM Engineer: Prompt tuning, IBM Watsonx setup
- QA Tester: Functional and performance testing
- Project Lead: Coordination, scheduling, and integration

6. FUNCTIONAL & PERFORMANCE TESTING:

- **Policy analysis speed**: Tested by uploading varying-sized documents and measuring summarization latency.
- **Complaint submission time**: Simulated multiple user reports using text, voice, and OCR formats to validate fast response and correct categorization.
- **KPI forecast accuracy vs. known values**: Verified with historical data files (CSV/PDF) using Linear Regression and compared prediction vs. actual trends.

- **Traffic/route query latency**: API integration response time measured to ensure fast route insights.
- **Chat response speed**: AskMe AI assistant performance tested across complex user queries.
- **Concurrent module usage**: Stress-tested simultaneous use of chatbot, resource tips, and reporting tools by multiple users.
- **Dashboard consistency**: Assured real-time update of statistics post every interaction (quiz, eco tip, report, chat).
- **Error resilience**: Introduced incorrect formats to validate robustness of document parser and LLM stability.

7. RESULTS:

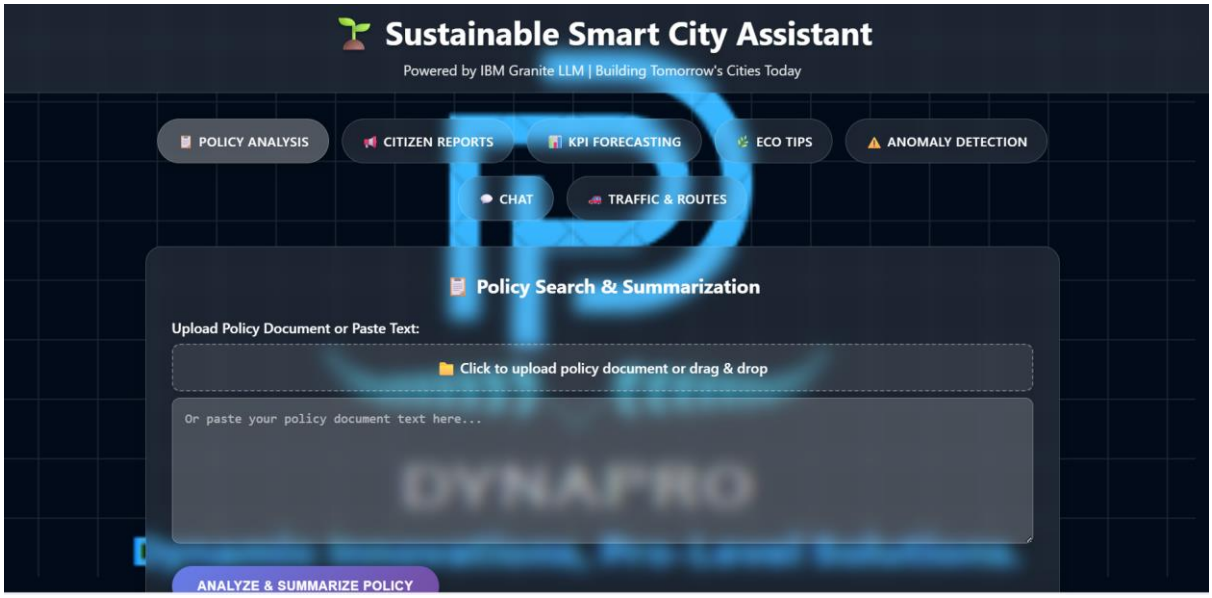


Fig: DashBoard

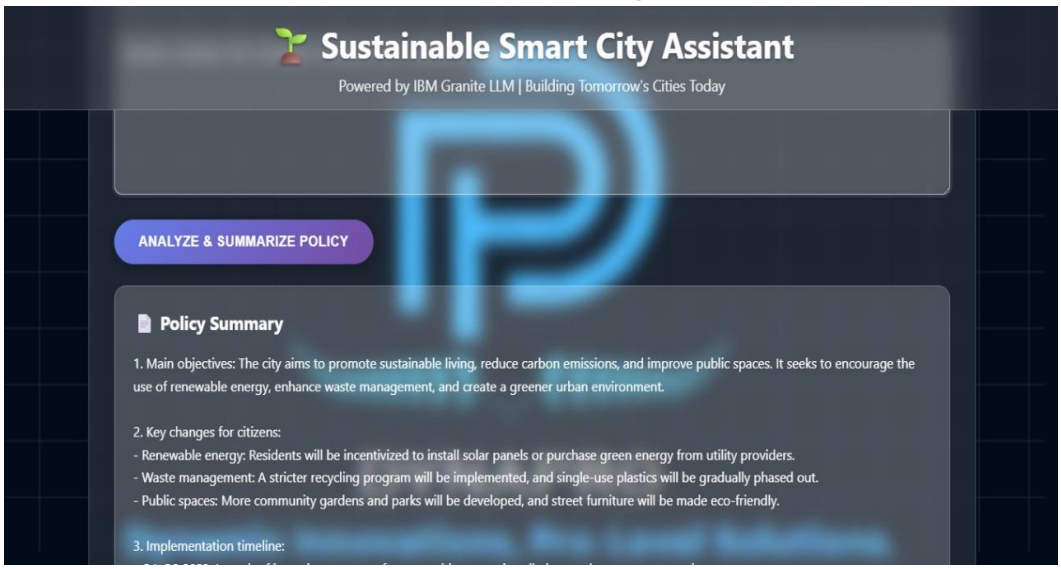


Fig: Policy Analyser.

Sustainable Smart City Assistant Using IBM Granite LLM

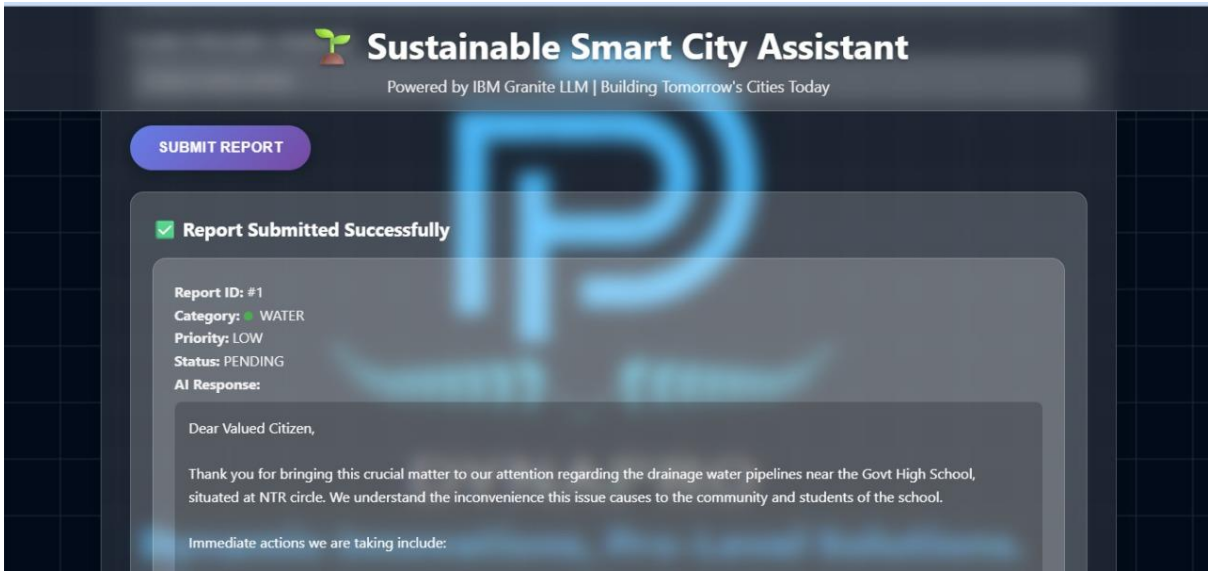


Fig:Citizen feedback

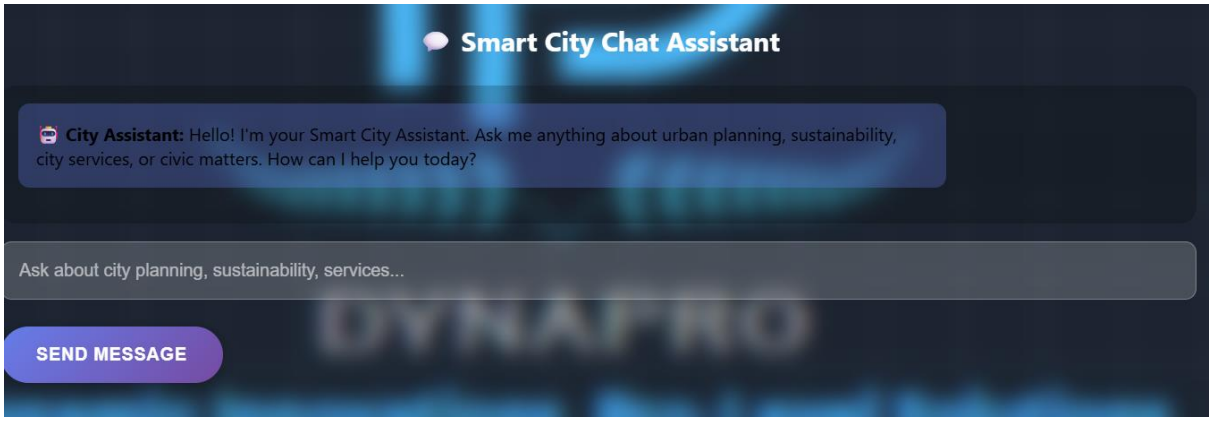


Fig:Chat Assistant

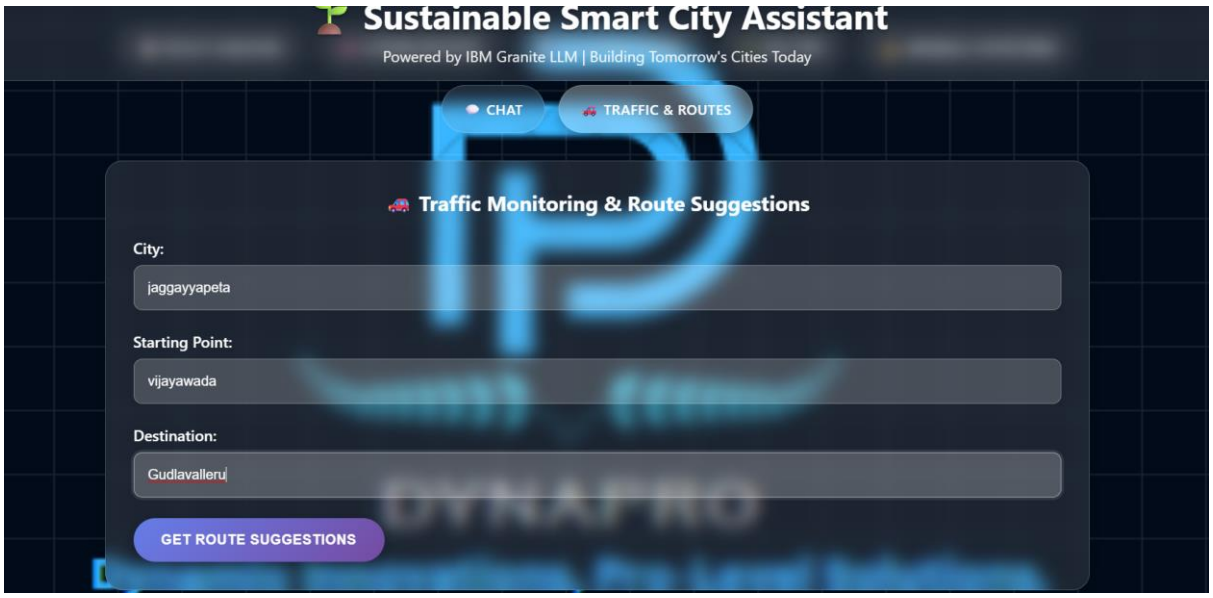


Fig:Route Suggestion

8. **ADVANTAGES & DISADVANTAGES**:

📉 Advantages:

- AI-powered personalization
- Easy document summarization
- Multi-service in one portal
- Live urban data integration
- Empowers both citizens & officials

⚠️ Disadvantages:

- LLM requires cloud/internet access
- Integration complexity with real-time data APIs
- No offline access

9. **CONCLUSION**:

The Sustainable Smart City Assistant transforms citizen engagement by merging AI with urban services. With modules for policy insight, civic reporting, eco-guidance, and analytics, it bridges the gap between smart governance and informed citizens.

10. **FUTURE SCOPE**:

- 🎧 Voice-based interactions for hands-free accessibility.
- 📱 Mobile App version to enhance user engagement on the go.
- 🌐 Multilingual AI support to serve diverse populations.
- 📡 IoT sensor data integration for real-time environment and traffic monitoring.
- 📊 Admin panel with heatmaps and detailed reports for decision-makers.
- 🔔 Smart alert system via SMS/email for emergencies and updates.
- 📍 GIS-based map interface to visualize complaint zones, eco stats, and traffic.
- 🧠 AI-powered trend prediction for urban planning and emergency preparedness.

- 🧾 Integration with e-Governance portals for seamless citizen service experience.
- 📤 Exportable analytics and citizen feedback reports for policy refinement.

11.APPENDIX:

- GitHub: https://github.com/Deekshith6049/my-web-app.git
- Demo-link: https://drive.google.com/file/d/1NU1fw_hiIxYTnTDTKeedvH04QqEUHZb4/view?usp=sharing