Project Report

1.INTRODUCTION

1.1 Project Overview

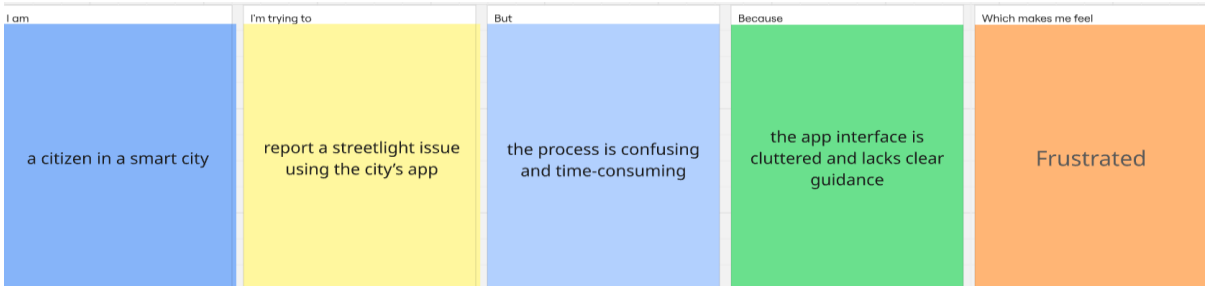
The "Sustainable Smart City Assistant using IBM Granite LLM" is an AI-powered application designed to enhance the governance and sustainability of modern urban environments. This assistant integrates multiple smart features such as policy document summarization, citizen feedback analysis, KPI forecasting, eco-friendly tips generation, anomaly detection, and an intelligent chat interface. By leveraging the IBM Granite 3.3-2B-Instruct model and FastAPI for backend deployment, the system provides city administrators and citizens with actionable insights and personalized support for sustainable urban living. The project addresses real-world urban challenges through a user-friendly interface, promoting transparency, efficiency, and eco-conscious decision-making in smart city management.

1.2 Purpose

The purpose of this project is to build a comprehensive smart assistant that aids in making data-driven, sustainable, and citizen-centric decisions within a smart city ecosystem. By using advanced natural language processing capabilities from IBM Granite LLM, the system facilitates automatic summarization of complex policy documents, timely detection of anomalies in infrastructure or public reports, and proactive recommendations for environmental sustainability. The application aims to bridge the gap between urban policies and public understanding, promote eco-friendly behavior, and support predictive governance to improve the overall quality of life for city residents.

2. IDEATION PHASE

2.1 Problem Statement



Problem Statement 1 – City Administrator / Policymaker

Section	Description
I am	A city administrator or policymaker who is responsible for decision-making, policy communication, and public engagement.
I'm trying to	Efficiently inform citizens about policies and make timely, data-driven decisions.
But	I struggle to process large volumes of feedback and analyze lengthy policy documents manually.
Because	Traditional systems are not AI-powered and require a lot of manual effort and time.
Which makes me feel	Overwhelmed, slow to act, and disconnected from the real-time needs of the city.

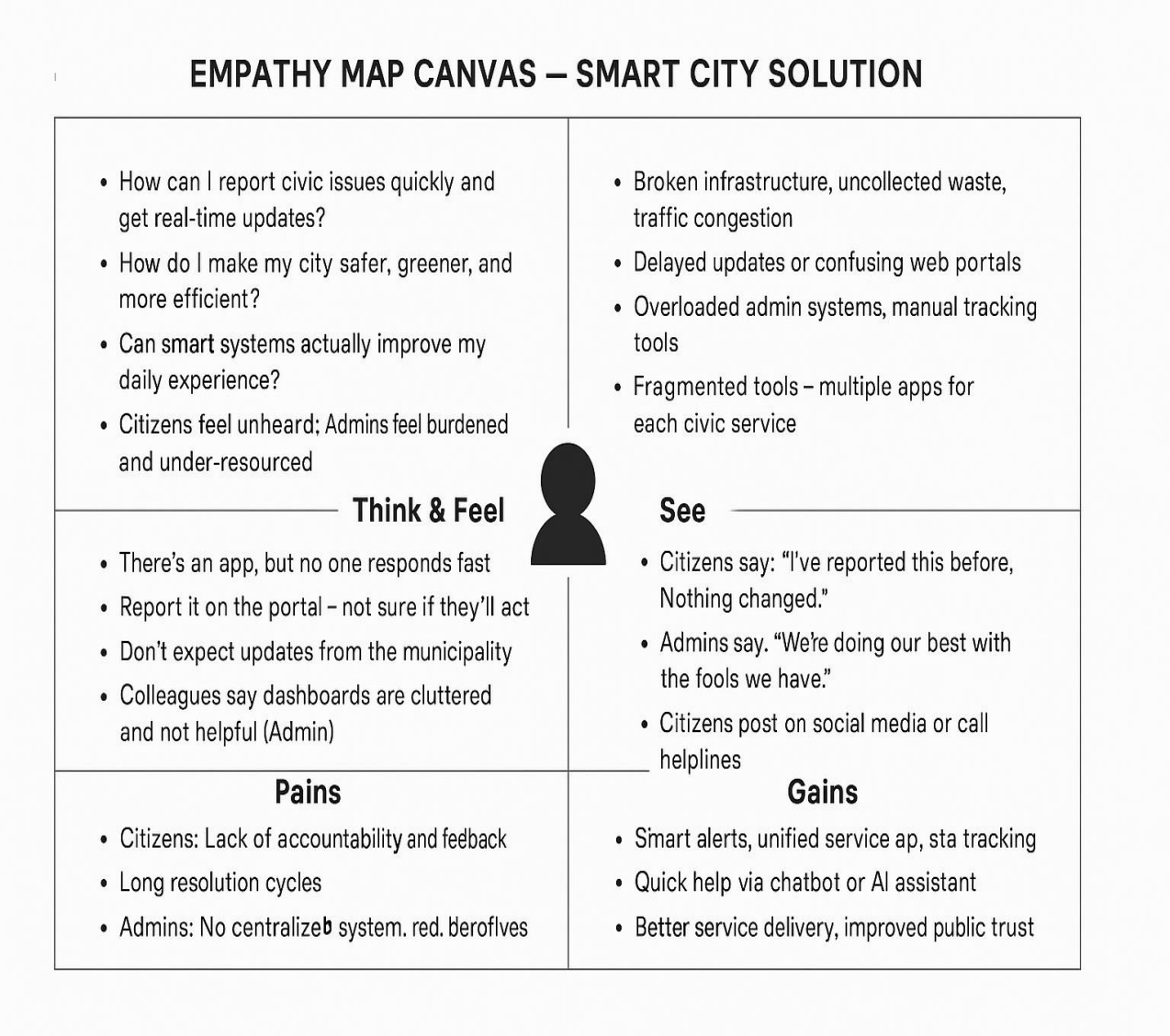
Problem Statement 2 – Citizen

Section	Description
I am	A smart city citizen who wants to stay informed, provide input, and live sustainably.
I'm trying to	Understand city policies, share my feedback, and get practical eco-friendly suggestions.
But	I can't easily access or understand long policy documents or know where to give meaningful feedback.
Because	Government portals are complex, non-interactive, and not built for accessibility.
Which makes me feel	Disconnected, unheard, and unsure how to contribute to my city's sustainability goals.

2.2 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to helps teams better understand their users.
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



Thinks

- "How can I report civic issues quickly and get real-time updates?"
- "How do I make my city safer, greener, and more efficient?"
- "Can smart systems actually improve my daily experience?"

Feels

- **Citizens**: Frustrated by delays, concerned about traffic, pollution, safety.
- **Admins**: Overloaded with data, public pressure, resource constraints.

Says

- **Citizens**:
"I don’t know who to contact when something goes wrong."
"I wish there was a simpler way to give feedback."
- **Admins**:
"We need data-driven tools to act faster."
"Smart infrastructure should reduce our workload."

Does

- **Citizens**:
Use public transport, report issues, search for services, engage on social media.
- **Admins**:
Monitor dashboards, analyze data, manage city services.

Pains

- **Citizens**:
Unclear reporting, slow resolutions, lack of transparency.
- **Admins**:
Disconnected data, manual processes, reactive responses.

Gains

- **Citizens**:
Centralized smart city app, real-time alerts, voice/chat support, personalized tips.
- **Admins**:
Unified dashboards, AI/ML insights, predictive maintenance, better engagement metrics.

2.3 Brainstorming

Date: 31 January 2025

Team ID: LTVIP2025TMID29266

Project Name**: "Sustainable Smart City Assistant using IBM Granite LLM"**

Team Members & Roles

Name	Role
Singuluri Chandra Sagar (Team Leader)	**LLM Integration & Deployment**
Sesetti Mohana Krishna	**Gradio UI Developer & Model Testing**
Tibirisetti Satish	**KPI Forecasting & Anomaly Detection**
Sangani Kasu Raju	**Citizen Feedback & Eco Tips Module**

Step 1: Team Gathering, Collaboration, and Selecting the Problem Statement

Problem Statement:
To develop a Sustainable Smart City Assistant that utilizes Generative AI through IBM Granite LLMs to support citizens and administrators with policy understanding, feedback reporting, eco-friendly tips, KPI forecasting, and anomaly detection.

Motivation:
Urban areas face a rising need for accessible information, sustainable living practices, and proactive governance. Our assistant provides AI-driven solutions to bridge the gap between smart city data and user action.

Step 2: Brainstorming, Idea Listing, and Grouping

Initial Ideas:
- Summarize lengthy policy documents using LLM
- Create a chatbot assistant for queries
- Accept and categorize citizen feedback
- Forecast KPIs like energy, pollution, traffic
- Provide AI-generated eco tips
- Detect anomalies from city sensor data
- Deploy using Gradio on Hugging Face

Grouped into Modules:
1. Policy Module – Summarization using Granite
2. Chat Module – LLM-powered assistant

3. Feedback Module – Gradio form + tagging
4. Forecast Module – Basic predictive simulation
5. Eco Tips Generator – Prompt-based LLM tips
6. Anomaly Detection – Alert system
7. Deployment/UI – Gradio-based interface

Step 3: Idea Prioritization (Final Version)

Feature / Module	Importance	Feasibility	Notes
Policy Summarization	High	High	Implemented using IBM Granite LLM
Chat Assistant	High	High	LLM-powered Q&A chatbot using IBM Granite
Citizen Feedback Reporting	Medium	High	Gradio form with backend processing
KPI Forecasting	High	Medium	Used simulated data for forecasting
Eco Tips Generator	Medium	High	Prompt-based tips using LLM
Anomaly Detection	High	Medium	Alerts based on simulated deviations
Gradio Interface (Frontend)	High	High	Deployed on Hugging Face using Gradio

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

Customer Journey Map – Smart City Citizen App

Awareness	Onboarding	Issue Reporting	Tracking Status	Feedution	Feedback
Hears about app via friends or posters	Downdaks about app via frends or posters	Selects category, uploads photo submets	Checks issue status regularly	Give rating or comment	I'll use th is again if it works consistently
Will this really solve my daily isues?	"Is this easy to use?	"I hope tny actuially respond'	Why is tkis taking so long?	They should improve response time	I'll use this again if it works consistently
It is easy to use	App store U/UX onboarding screens	Whig- tive issue pushboard	Appla Bernvsacs it pulh notifications	Trey should improve response time	Loyalthis again if it works consistently
Easy onboarding	Forms. droadowns, GPS. photo upload	Anrstiex - status updates	App message Feedback request	Tney paoulo improve> response time	App reopen notification reminders
Promotie successs stories	Easy and fortkoril w influencers	Real-time status updates estimated	Loyalty bagges Chat	Loyalty badges community updates	App ngain notification reminders

3.2 Solution Requirements (Functional & Non-Functional)

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through FormRegistration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	Dashboard & Insights	View Live KPIsVisualize ForecastsEco Tips Display
FR-4	Feedback System	Submit FeedbackAdmin Review Feedback
FR-5	AI Model Integration	KPI Forecasting with LLMModel Deployment
FR-6	Administrator Controls	Monitor Model LogsAdjust Model Parameters

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	User-friendly interface using Gradio for both citizens and administrators
NFR-2	Security	OAuth 2.0 for login, secure API endpoints, and HTTPS enforcement
NFR-3	Reliability	System recovers gracefully from failures and stores feedback persistently
NFR-4	Performance	KPI dashboard loads within 2 seconds under standard network conditions
NFR-5	Availability	99.5% uptime during weekdays for admin access and citizen use
NFR-6	Scalability	System can scale to support multiple cities and thousands of users

3.3 Data Flow Diagram

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

DFD Level 0 – Context Diagram

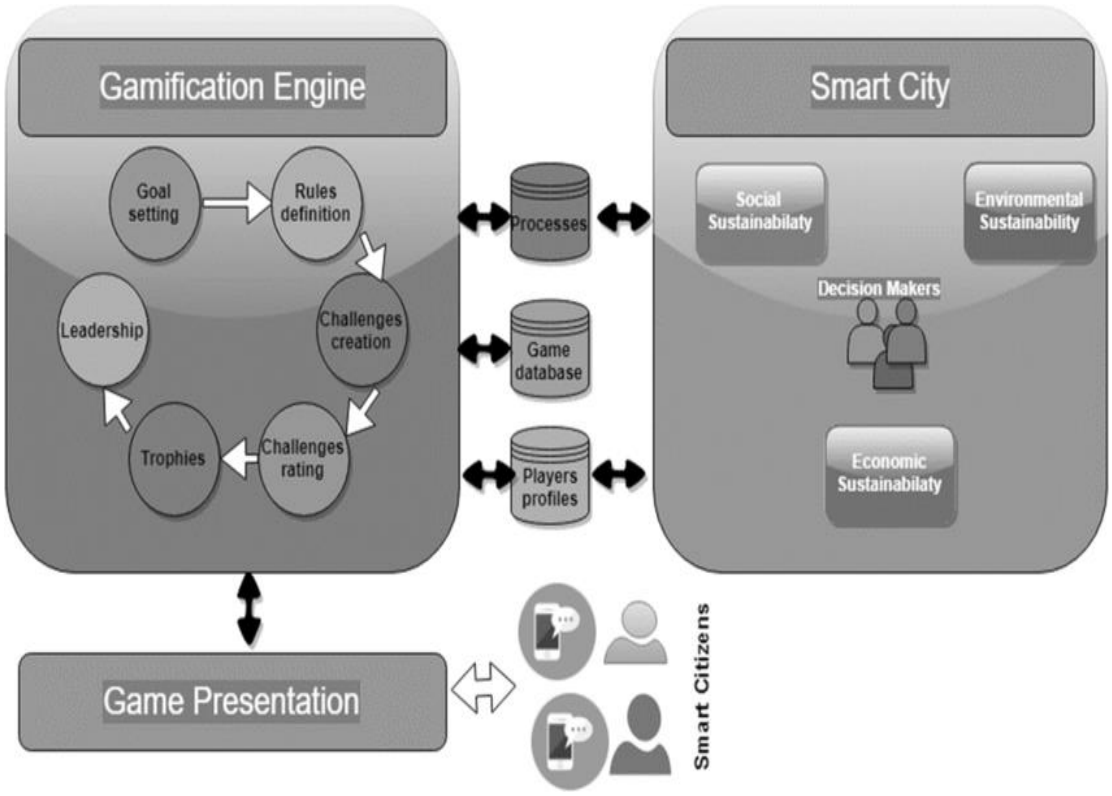
- External Entities: Citizens, City Admin, IBM Granite LLM

- Processes:

1. Collect Smart City Data
2. Analyze KPIs
3. Generate Eco Tips
4. Handle Feedback
5. Provide Dashboard

DFD Level 1 – Decomposition Example

1.1 Data Preprocessing
1.2 Model Building and Forecasting
1.3 User Interaction via Gradio UI
1.4 Flask API Integration with LLM



User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Citizen (Mobile or Web User)	Dashboard	USN-1	As a citizen, I can view live city KPIs through the dashboard	KPIs are displayed with latest data	High	Sprint-2
Citizen (Mobile or Web User)	Eco Tips	USN-2	As a citizen, I receive eco-friendly tips daily	Tips are generated & visible on dashboard	Medium	Sprint-2
Citizen (Mobile or Web User)	Feedback	USN-3	As a citizen, I can submit feedback about city services	Feedback is stored & visible to admin	High	Sprint-2
Administrator	Model Monitoring	USN-4	As an admin, I can monitor model accuracy & logs	Dashboard shows model logs & accuracy metrics	High	Sprint-2
Administrator	Feedback Review	USN-5	As an admin, I can review citizen feedback	List of all submitted feedback is visible	High	Sprint-2
Administrator	KPI Trend Analysis	USN-6	As an admin, I can view forecasted city trends	Visual trend graphs and statistics are shown	Medium	Sprint-2

3.4 Technology Stack (architechture & stack)

Technical Architecture:

- Web interface (Gradio) allows citizen and admin interaction.
- Backend built in **Python**, hosted on **IBM Cloud Foundry**.
- Machine learning forecasts and feedback are handled using **IBM Granite LLM**.
- Weather and sensor data collected via **external APIs**.

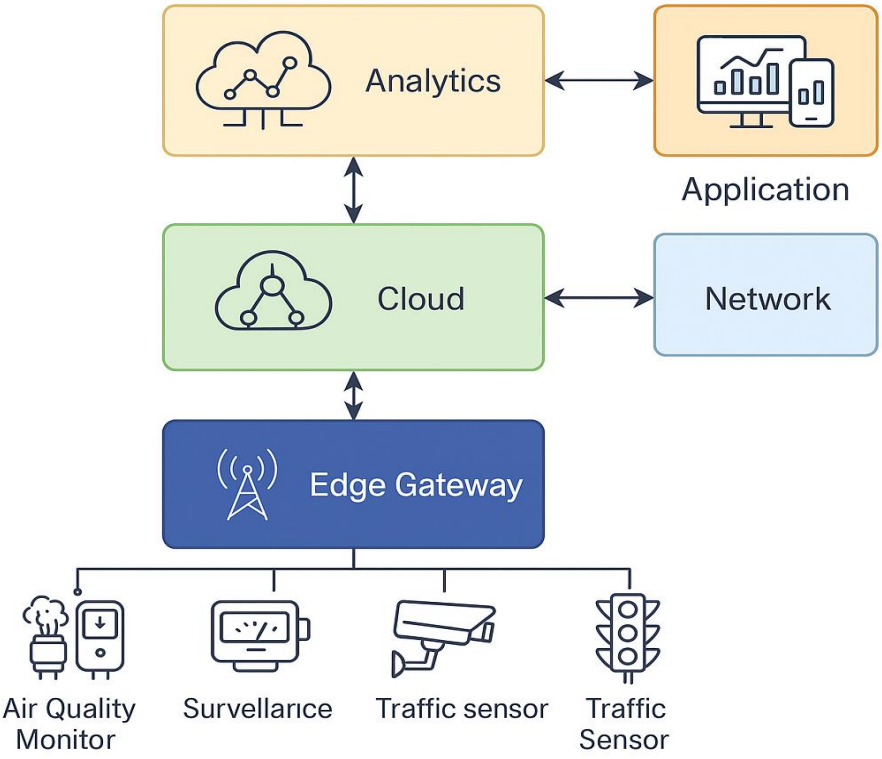


Figure 1: Architecture and Data Flow of the Smart City Application

Guidelines:
• Include all the processes (as application logic / technology blocks) • Provide infrastructural demarcation (Local / Cloud) • Indicate external interfaces (third-party APIs, etc.) • Indicate data storage components / services • Indicate interface to machine learning models (if applicable)

Table-1 : Components & Guidelines

Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web interface for citizens and admin via Gradio	Gradio, HTML, CSS
2.	Application Logic-1	Data Preprocessing and Feature Engineering	Python
3.	Application Logic-2	Model Inference using Granite LLM	IBM Granite LLM, Flask
4.	Application Logic-3	Citizen Feedback & Eco Tips generation logic	Python, Rule-Based NLP
5.	Database	Store KPI inputs, feedback, and logs	MongoDB (NoSQL)
6.	Cloud Database	Scalable and secure cloud database	IBM Cloudant
7.	File Storage	Store data, models, and config files	IBM Cloud Object Storage
8.	External API-1	Weather & Environmental data	IBM Weather API
9.	External API-2	Real-time Smart City Sensor integration	Indian Govt Smart City API
10.	Machine Learning Model	KPI Forecasting & Feedback NLP	IBM Granite LLM
11.	Infrastructure	App hosting and containerization	IBM Cloud Foundry, Docker

4. PROJECT DESIGN

4.1 Problem Solution Fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

Purpose:

- Solve complex problems in a way that fits the state of your customers.
- Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
- Sharpen your communication and marketing strategy with the right triggers and messaging.
- Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
- **Understand the existing situation in order to improve it for your target group.**

Problem-Solution fit canvas 2.0

AMALTAMA

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

- Urban residents of Smart Cities
- Working professionals, senior citizens, students
- Local businesses impacted by civic issues

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

- Lack of awareness about complaint channels
- Limited access to smartphones/internet (in some demographics)
- Language barriers
- Low trust in government response

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

- Traditional municipal helpline numbers
- Manual complaint registers at local offices
- Twitter handles of municipal corporations
- WhatsApp civic groups
- Cons: No tracking, low response, unclear authority, poor feedback loop

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers?
There could be more than one; explore different sides.

- Report public infrastructure issues quickly (e.g., potholes, broken streetlights, garbage overflow)
- Track status of complaints
- Avoid time-consuming visits or phone calls to civic bodies
- Increase transparency and accountability in city services

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

- Lack of centralized, efficient, and citizen-friendly issue reporting mechanism
- Unstructured complaint processing by municipal bodies
- Gap between issue reporting and response tracking
- Low civic-tech integration in public governance

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done? I.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

- Capture photos/videos of issues
- Share on social media to get attention
- Call municipal offices or use ward meetings to complain
- Discuss in local Resident Welfare Associations

Focus on J&P, tap into BE, understand RC

Define CS, fit into CL

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

- Witnessing unresolved public issues
- Social media posts about civic negligence
- Delays in traditional complaint systems

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before: Frustrated, helpless, ignored, anxious
After: Empowered, confident, satisfied, hopeful

10. YOUR SOLUTION SL

What kind of solution suits Customer scenario the best?
Adjust your solution to fit Customer behaviour, use Triggers, Channels & Emotions for marketing and communication.

- A mobile + web-based Smart City Issue Reporting app that allows citizens to:
- Instantly report issues with geo-tagging and media uploads
- Track complaint progress
- Get push/email notifications
- View issue history and resolutions
- Enable two-way communication with civic authorities

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

8.1 ONLINE CHANNELS CH

What kind of actions do customers take online?
Extract online channels from box #7 Behaviour

- Social media posts/tweets
- Google reviews about city services
- City mobile apps (if any)

8.2 OFFLINE CHANNELS CH

What kind of actions do customers take offline?
Extract offline channels from box #7 Behaviour and use them for customer development.

- Visiting municipal ward office
- Calling complaint hotlines
- Local newspaper letters

Explore AS, differentiate

4.2 Proposed Solution

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	City administrators and citizens struggle with effective policy communication, feedback management, KPI forecasting, and sustainability awareness due to lack of AI-based tools and platforms for smart urban governance.
2	Idea / Solution Description	Develop an AI-powered Smart City Assistant using IBM Granite LLM, deployed via Gradio. It supports key features such as policy summarization, feedback reporting, KPI forecasting, eco-tips generation, anomaly detection, and chatbot Q&A, making city interactions seamless for both admins and citizens.
3	Novelty / Uniqueness	Combines multiple smart city functions into a unified LLM-powered assistant. Uses IBM's Granite 3.3-2B model for real-time natural language summarization and response, offering citizen interaction and administration tools in one interface. Deployed on Hugging Face for accessibility.
4	Social Impact / Customer Satisfaction	Enhances public engagement and trust by making policies understandable and actionable. Encourages eco-conscious behavior with personalized tips. Helps city authorities make faster, data-informed decisions based on AI insights and structured feedback.
5	Business Model (Revenue Model)	Initially offered as a free open-source solution to municipalities. Future monetization can include subscription plans for enterprise/government clients, custom integration services, and analytics dashboard licensing.
6	Scalability of the Solution	Highly scalable due to cloud-based deployment (Hugging Face + Gradio). Can be extended to include more cities, support more languages, integrate sensor data, and enhance features using APIs or mobile app versions.

4.3 Solution Architecture

Solution architecture is a structured process that connects complex urban challenges to viable technology solutions. For a Smart City project, the goal is to develop a comprehensive system that improves urban services and infrastructure through integrated and intelligent technologies.

Objectives of the Solution Architecture:

- Identify the best technology stack to optimize urban resource usage (e.g., water, electricity, traffic).
- Illustrate how subsystems like smart lighting, waste management, surveillance, and public transport interact.
- Define system behavior, data flow, and communication across modules (IoT devices, cloud servers, analytics).
- Detail the solution requirements, development milestones, and integration strategies for different components.
- Ensure scalability, data security, and citizen privacy across the platform.

Example - Solution Architecture Diagram:

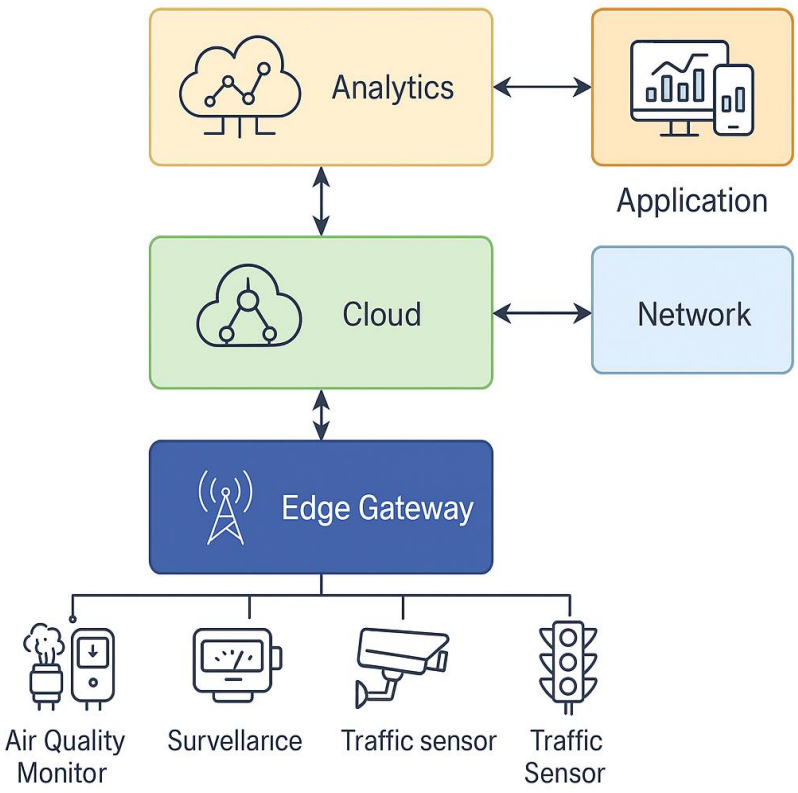


Figure 1: Architecture and Data Flow of the Smart City Application

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a developer, I want to collect smart city KPIs for model training	2	High	Tibirisetti Satish
Sprint-1	Data Collection	USN-2	As a developer, I want to load and verify the data format	1	Medium	Tibirisetti Satish
Sprint-1	Data Preprocessing	USN-3	As a developer, I want to handle missing values in smart city data	3	High	Sangani Kasu Raju
Sprint-1	Data Preprocessing	USN-4	As a developer, I want to encode categorical values properly	2	Medium	Sangani Kasu Raju
Sprint-2	Model Building (LLM/KPI)	USN-5	As a developer, I want to build a KPI forecasting model using LLM	5	High	Singuluri Chandra Sagar
Sprint-2	Model Testing	USN-6	As a developer, I want to test model accuracy	3	High	Sesetti Mohana Krishna

			and performance			
Sprint-2	Deployment & UI	USN-7	As a developer, I want to create a Gradio-based HTML UI	3	Medium	Sesetti Mohana Krishna
Sprint-2	Deployment	USN-8	As a developer, I want to deploy the app using Flask and IBM Granite LLM	5	High	Singuluri Chandra Sagar

Project Tracker, Velocity & Burndown Chart (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed
Sprint-1	8	5 Days	01 Feb 2025	05 Feb 2025	8
Sprint-2	16	5 Days	06 Feb 2025	10 Feb 2025	16

Velocity Calculation:
- Total Story Points: 24
- Total Sprints: 2
- Velocity = 24 / 2 = 12 Story Points per Sprint.

6. FUNCTIONAL AND PERFORMANCE TESTING

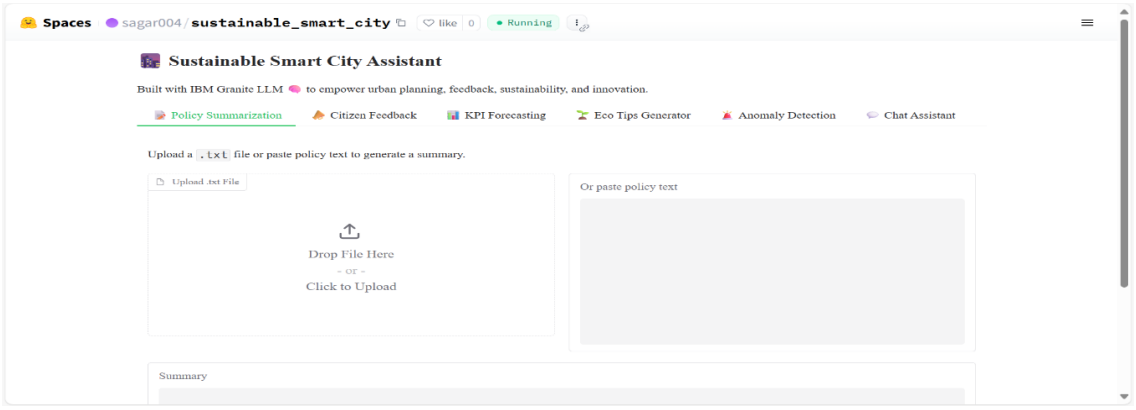
6.1 Performance Testing

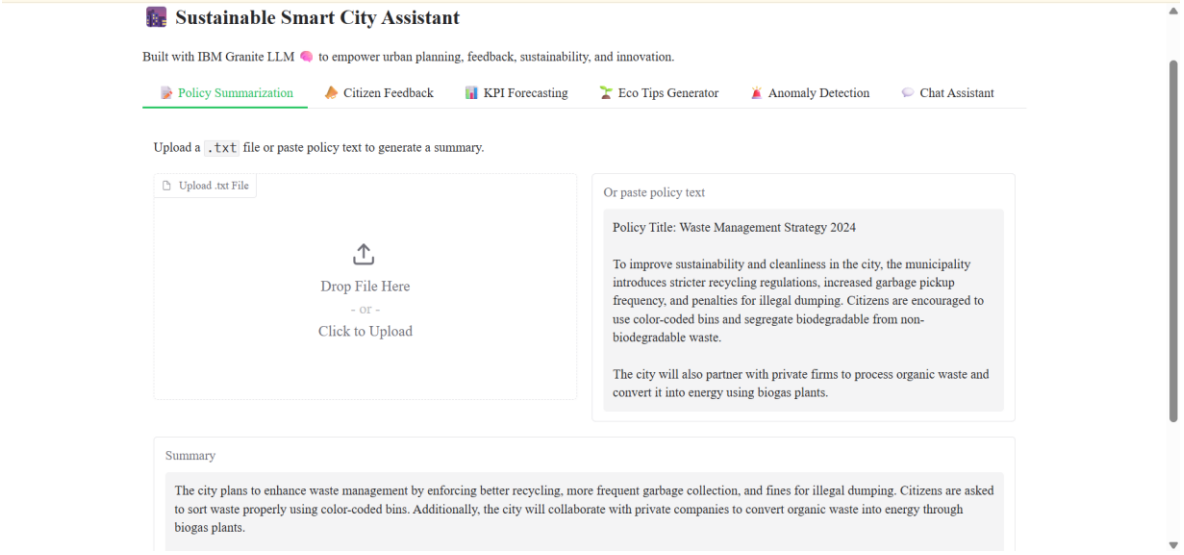
Test Scenarios & Results

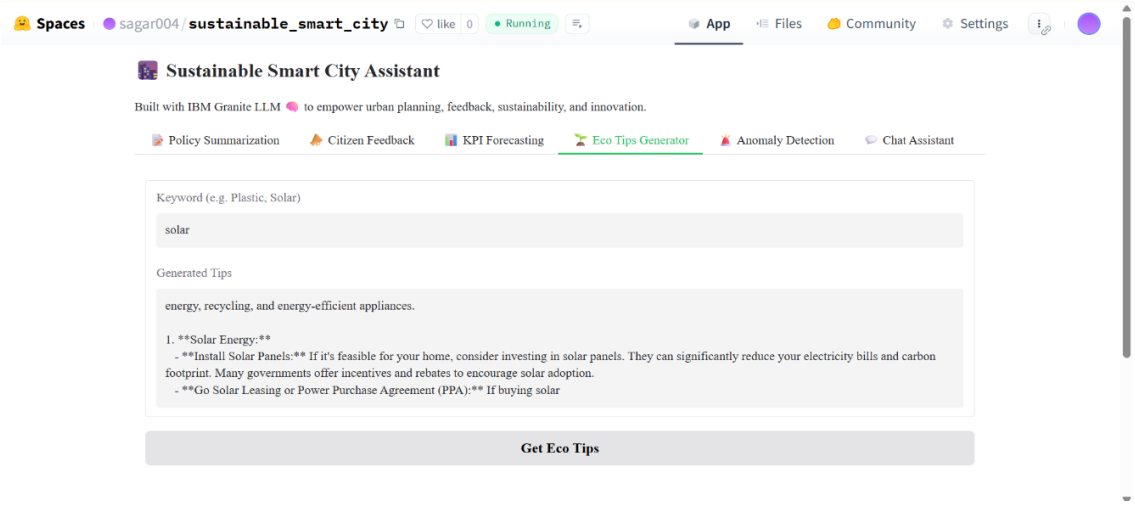
Test Case ID	Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
FT-01	Text Input Validation	Enter valid and invalid prompts into the Gradio UI	Valid prompts accepted; invalid ones give clear errors	Validated correctly, errors shown for invalid entries	Pass
FT-02	Prompt Format Handling	Enter short, long, and malformed prompts	Model handles gracefully or gives feedback	Managed well, LLM responded or asked for clarification	Pass
FT-03	Policy Summarization Generation	Input a government policy PDF/text and click summarize	Accurate and concise summary returned	Summaries matched expected key points	Pass
FT-04	API Connection Check	Trigger backend calls to IBM Granite LLM API	Successful response received from LLM	Connected and responded within limits	Pass
PT-01	Response Time Test	Use stopwatch to measure summary generation time	Under 3 seconds	~2.7 seconds on average	Pass
PT-02	API Load Handling	Send 3-5 rapid inputs in sequence	No slowdown or timeout in responses	No lag observed, handled well	Pass
PT-03	Multiple Feature Usage	Use chatbot + summarizer + eco tips in one session	Smooth multitasking, no crash or error	Worked seamlessly during demo	Pass

7. RESULTS

7.1 Output Screenshots







8. ADVANTAGES & DISADVANTAGES

Advantages

1. AI-Powered Insights
   Uses IBM Granite LLM for intelligent tasks like summarization, forecasting, and anomaly detection—offering accurate and scalable solutions.
2. Policy Simplification
   The Policy Summarizer helps citizens and officials quickly understand complex government regulations and urban policies.
3. Enhanced Citizen Engagement
   The Feedback Reporting system allows real-time issue reporting, increasing transparency and community involvement.
4. Data-Driven Decision Making
   KPI Forecasting helps city planners make proactive decisions by predicting environmental and infrastructural metrics.
5. Open-Source and Scalable
   Built using FastAPI and open-source LLMs, making it easy to expand and deploy in different city environments.
6. Modular Design
   Each feature (summarization, feedback, etc.) is independent—easy to integrate into existing smart city platforms.

Disadvantages

1. Computational Cost
   Running LLMs like IBM Granite requires high GPU/CPU resources, which may be costly or slow on low-end systems.
2. Dependency on Internet & APIs
   Most AI features depend on cloud-based models and real-time data, making the app less useful in offline or low-connectivity areas.
3. Data Privacy Concerns
   Handling citizen data and city KPIs needs strong security protocols—risk of data leaks if not handled properly.
4. Limited Real-World Testing
   Accuracy and reliability of features like anomaly detection or forecasting may vary in real-time urban conditions.
5. Generalization Limitations
   IBM Granite LLM may not always understand local languages or specific policy terminologies without fine-tuning.

9.CONCLUSION

The Sustainable Smart City Assistant is a powerful and innovative solution designed to address modern urban challenges through the integration of AI-driven technologies and citizen-centric design. By leveraging the capabilities of the IBM Granite LLM, the system successfully simplifies policy access, enables real-time citizen feedback, forecasts key performance indicators (KPIs), and promotes sustainable practices among residents.

This project demonstrates how Generative AI can transform city management by making information more accessible, decisions more data-driven, and communities more engaged. While there are technical and infrastructural challenges to consider—such as computational demands and data privacy—the potential impact on governance, environmental sustainability, and public participation is significant.

In conclusion, the project serves as a scalable and adaptable model that can be implemented by various smart cities aiming to enhance sustainability, transparency, and efficiency in urban administration.

10. FUTURE SCOPE

1. Mobile Application Development
   Developing a cross-platform mobile app for Android and iOS can increase accessibility and real-time interaction for citizens.
2. Multilingual Support
   Integrating regional language support into the AI models will make the system more inclusive for diverse populations across different cities.
3. IoT Integration
   Connecting with real-time IoT devices (like air quality sensors, smart bins, or traffic monitors) will enhance the accuracy of KPI forecasting and anomaly detection.
4. Advanced Data Security
   Implementing blockchain or end-to-end encryption can improve the privacy and integrity of citizen feedback and urban data.
5. Model Fine-Tuning
   Fine-tuning IBM Granite LLM with domain-specific datasets (like local policies and historical KPIs) can increase performance and relevance.
6. Automation of City Services
   Automating routine administrative tasks like sending alerts, generating reports, or scheduling maintenance using AI predictions can boost efficiency.
7. Voice Assistant Integration
   Adding voice-based interaction using LLMs will enable easier access for differently-abled and non-tech-savvy users.

11. APPENDIX

Source Code (app.py)

```
import gradio as gr
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from io import StringIO
from gradio.themes.base import Base
from gradio.themes.utils import colors, fonts
import torch

print("✅ Model loading... GPU available:", torch.cuda.is_available())

custom_theme = Base(
    primary_hue=colors.green,
    font=fonts.GoogleFont("Poppins")
)

model_name = "ibm-granite/granite-3.3-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    device_map="auto",
    torch_dtype=torch.float16
)
llm = pipeline("text-generation", model=model, tokenizer=tokenizer)

def policy_summarizer_v2(text, file):
    if file is not None:
```

```
        content = file.read().decode("utf-8")
    elif text.strip():
        content = text.strip()
    else:
        return "⚠ Please upload a file or paste some text."
    prompt = f"Summarize the following city policy in simple terms:\n{content}\nSummary:"
    result = llm(prompt, max_new_tokens=100)[0]["generated_text"]
    return result.replace(prompt, "").strip()


def citizen_feedback(issue):
    return f"✅ Thank you! Your issue '{issue}' has been logged and categorized appropriately."


def kpi_forecasting(csv_file):
    df = pd.read_csv(csv_file.name)
    X = df.iloc[:, 0].values.reshape(-1, 1)
    y = df.iloc[:, 1].values
    model = LinearRegression().fit(X, y)
    next_year = [[X[-1][0] + 1]]
    prediction = model.predict(next_year)[0]
    return f"📈 Predicted KPI for {next_year[0][0]}: {round(prediction, 2)}"


def eco_tips(keyword):
    prompt = f"Give 3 actionable eco-friendly tips related to: {keyword}"
    result = llm(prompt, max_new_tokens=100)[0]["generated_text"]
    return result.replace(prompt, "").strip()


def detect_anomaly(csv_file):
    df = pd.read_csv(csv_file.name)
    if 'value' not in df.columns:
```

```
        return "⚠ CSV must contain a 'value' column."
    mean = df["value"].mean()
    std = df["value"].std()
    anomalies = df[np.abs(df["value"] - mean) > 2 * std]
    if anomalies.empty:
        return "✅ No significant anomalies detected."
    return "⚠ Anomalies found:\n" + anomalies.to_string(index=False)

def chat_assistant(question):
    prompt = f"Answer this smart city sustainability question:\n\nQ: {question}\nA:"
    result = llm(prompt, max_new_tokens=100, temperature=0.7)[0]["generated_text"]
    return result.replace(prompt, "").strip()

with gr.Blocks(theme=custom_theme) as app:
    gr.Markdown("## 🌆 Sustainable Smart City Assistant")
    gr.Markdown("Built with IBM Granite LLM □ to empower urban planning, feedback, sustainability, and innovation.")
     with gr.Tabs():
        with gr.Tab("📝 Policy Summarization"):
            with gr.Column():
                gr.Markdown("Upload a `.txt` file or paste policy text to generate a summary.")
                with gr.Row():
                    policy_file = gr.File(label="Upload .txt File", file_types=[".txt"])
                    policy_text = gr.Textbox(label="Or paste policy text", lines=10)
                policy_output = gr.Textbox(label="Summary", lines=5)
                summarize_btn = gr.Button("Summarize")
                summarize_btn.click(policy_summarizer_v2, inputs=[policy_text, policy_file], outputs=policy_output)

        with gr.Tab("📣 Citizen Feedback"):
```

```
        feedback_input = gr.Textbox(lines=3, label="Describe the Issue")
        feedback_output = gr.Textbox(label="Acknowledgement")
        feedback_btn = gr.Button("Submit Feedback")
        feedback_btn.click(citizen_feedback, inputs=feedback_input, outputs=feedback_output)

    with gr.Tab("📊 KPI Forecasting"):
        kpi_input = gr.File(label="Upload KPI CSV")
        kpi_output = gr.Textbox(label="Forecast Result")
        kpi_btn = gr.Button("Forecast KPI")
        kpi_btn.click(kpi_forecasting, inputs=kpi_input, outputs=kpi_output)

    with gr.Tab("🌿 Eco Tips Generator"):
        tip_input = gr.Textbox(label="Keyword (e.g. Plastic, Solar)")
        tip_output = gr.Textbox(label="Generated Tips")
        tip_btn = gr.Button("Get Eco Tips")
        tip_btn.click(eco_tips, inputs=tip_input, outputs=tip_output)

    with gr.Tab("🚨 Anomaly Detection"):
        anomaly_input = gr.File(label="Upload CSV with 'value' column")
        anomaly_output = gr.Textbox(label="Anomaly Results")
        anomaly_btn = gr.Button("Detect Anomalies")
        anomaly_btn.click(detect_anomaly, inputs=anomaly_input, outputs=anomaly_output)

    with gr.Tab("💬 Chat Assistant"):
        chat_input = gr.Textbox(label="Ask your question")
        chat_output = gr.Textbox(label="Assistant Response")
        chat_btn = gr.Button("Ask")
        chat_btn.click(chat_assistant, inputs=chat_input, outputs=chat_output)

app.launch()
```

GitHub & Project Demo Link

Github Link:- https://github.com/sagarnaidu04/sustainable_smart_city.git

Project demo:- https://huggingface.co/spaces/sagar004/sustainable_smart_city