**SUSTAINABLE SMART CITY**

**ASSISTANT USING IBM GRANITE LLM**

Project Documentation

# Introduction:

Project title: Sustainable Smart City Assistant Using IBM Granite LLM

* + Team Leader: KRISHNA S
  + Team Member: SAKTHIVEL P
  + Team Member: SAKTHIVEL P
  + Team Member: SATHISHKUMAR P

# Project Overview:

# Purpose:

The purpose of a Sustainable Smart City Assistant is to empower cities and their residents to thrive in a more eco-conscious and connected urban environment. By leveraging AI and real-time data, the assistant helps optimize essential resources like energy, water and waste, while also guiding sustainable behaviors among citizens through personalized tips and services. For city officials, it serves as a decision- making partner—offering clear insights, forecasting tools, and summarizations of complex policies to support strategic planning. Ultimately,this assistant bridges technology,governance,and community engagement to foster greener cities that are more efficient, inclusive, and resilient.

* + Features:

## Conversational Interface

*Key Point:* Natural language interaction

*Functionality:* Allows citizens and officials to ask questions, get updates, and receive guidance in plain language

## Policy Summarization

*Key Point:* Simplified policy understanding

*Functionality:* Converts lengthy government documents intoconcise, actionable summaries.

## Resource Forecasting

*Key Point:* Predictive analytics

*Functionality:* Estimates future energy, water, and waste usage using historical and real-time data.

## Eco-Tip Generator

*Key Point:* Personalized sustainability advice

*Functionality:* Recommends daily actions to reduce environmental impact based on user behavior.

## Citizen Feedback Loop

*Key Point:* Community engagement

*Functionality:*Collects and analyzes public input toi nform city planning and service improvements.

## KPI Forecasting

*Key Point:* Strategic planning support

*Functionality:* Projects key performance indicators to help officials track progress and plan ahead.

## Anomaly Detection

*Key Point:* Early warning system

*Functionality:* Identifies unusual patterns in sensor or usage data to flag potential issues.

## Multimodal Input Support

*KeyPoint:* Flexible data handling

*Functionality:* Accepts text, PDFs, and CSVs for document analysis and forecasting.

## Streamlit or Gradio UI

*Key Point:* User-friendly interface

*Functionality:* Provides an intuitive dashboard for both citizens and city officials to interact with the assistant.

*3.Architecture:*

*Frontend (Stream lit):*

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

## Backend(FastAPI):

Fast API serves as the backend REST framework that powers API end points for document processing, chat interactions, eco tip generation, report creation,vectorembedding.It is optimized for asynchronous performance and easy Swagger integration.

## 4.Setup Instructions

## Prerequisites:

* Python3.9orlater
* pipandvirtualenvironmenttools
* APIkeysforIBMWatsonxandPinecone
* Internetaccesstoaccesscloudservices

## InstallationProcess:

* Clonetherepository
* Installdependenciesfromrequirements.txt
* Createa.envfileandconfigurecredentials
* RunthebackendserverusingFastAPI
* LaunchthefrontendviaStreamlit
* Uploaddataandinteractwiththemodules

# Folder Structure

app/–Contains all Fast API backend logic including routers, models, and integration modules.

app/api/–Subdirectory for modular API routes like chat, feedback, report, and document vectorization.

ui/–Contains frontend components for Streamlit pages ,card layouts, and form UIs.

smart\_dashboard.py–Entry script for launching the main Streamlit dashboard.

granite\_lim.py– Handles all communication with IBM Watsonx Granite model including summarization and chat.

document\_embedder.py–Converts documents to embeddings and stores in Pinecone.

kpi\_file\_fore caster.py–Forecasts future energy/water trends using regression. anomaly\_file\_checker.py – Flags unusual values in uploaded KPI data.

# Running the Application

To start the project:

* Launch the Fast API server to expose backend endpoints.
* Run the Stream lit dashboard to access the web interface.
* Navigate through pages via the sidebar.

Upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions

## Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web UI with multiple pages including dashboards, file uploads, chat interface, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

## Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for document processing, chat interactions, eco tip generation, report creation, andvectorembedding.Itisoptimizedforasynchronousperformanceandeasy Swagger integration.

# API Documentation

Backend APIs available include:

POST/chat/ask– Accepts a user query and responds with an AI-generated message

POST/upload-doc–Uploads and embeds documents in Pinecone

GET/search-docs–Returns semantically similar policies to the input query

GET/get-eco-tips–Provides sustainability tips for selected topics like energy, water, or waste

POST/submit-feedback–Stores citizen feedback for later review or analytics

Each end point is tested and documented in Swagger UI for quick inspection and trial during development.

# Authentication

Each end point is tested and documented in Swagger UI for quick inspection and trial during development.

This version of the project runs in an open environment for demonstration. However, secure deployments can integrate:

* + Token-based authentication(JWT or API keys)
  + OAuth2withIBMCloudcredentials
  + Role-based access(admin, citizen ,researcher)
  + Plannedenhancementsincludeusersessionsandhistorytracking.8. Authentication

# User Interface

The interface is minimalist and functional, focusing on accessibility for non- technical users. It includes:

Sidebar with navigation

KPI visualizations with summary cards

Tabbed layouts for chat, ecotips, and forecasting Real-time form handling

PDF report download capability

The design prioritizes clarity, speed, and user guidance ewith help texts and intuitive flows.

# Testing

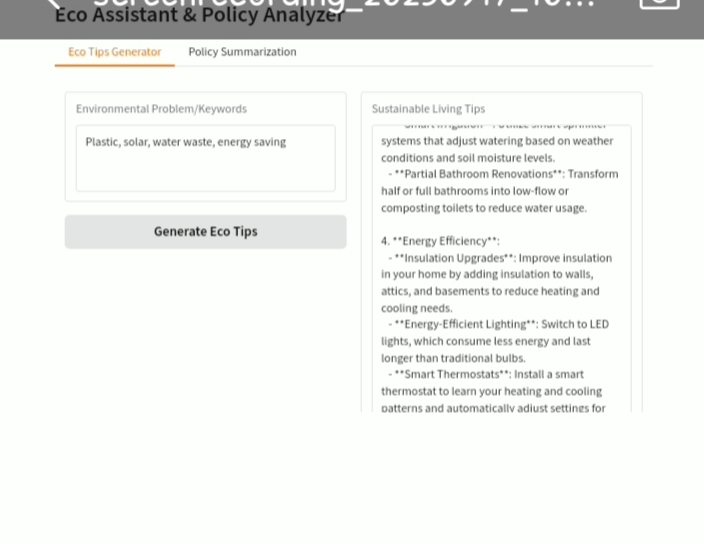
Testing was done in multiple phases:

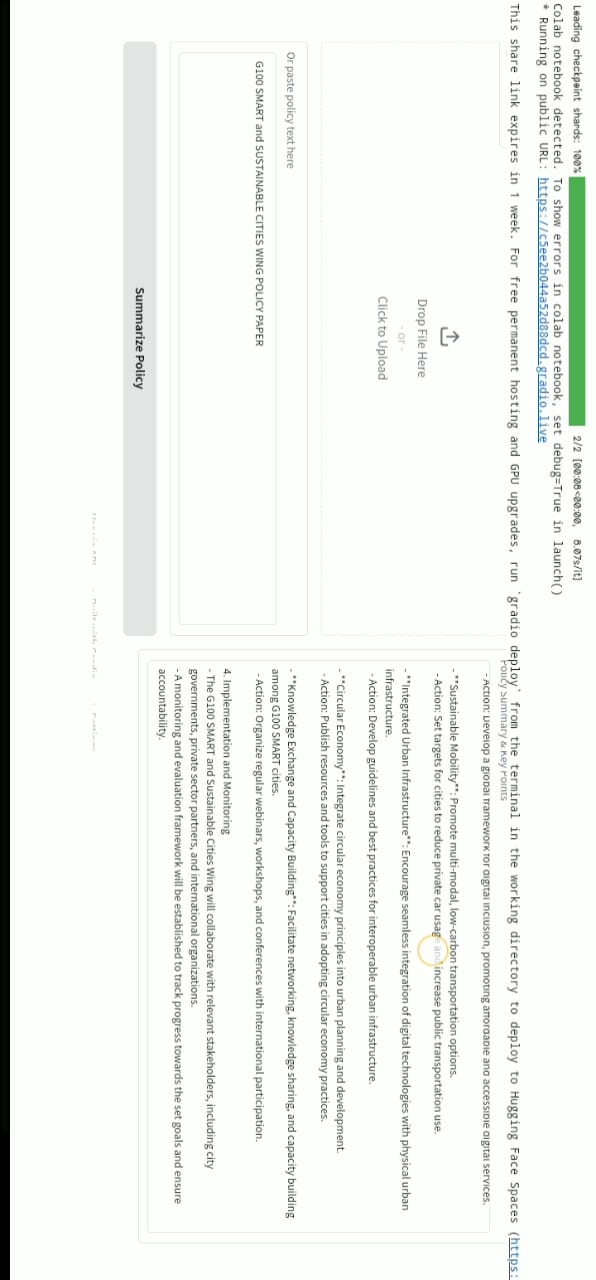
Unit Testing: For prompt engineering functions and utility scripts API Testing: Via Swagger UI, Postman, and test scripts

Manual Testing: For file uploads, chat responses, and output consistency Edge Case Handling: Malformed inputs, large files, invalid API keys

Each function was validated to ensure reliability in both offline and API- connected modes.

**11.screenshot:**





12.Future Enhanement:

Future enhancements for sustainable s mart city projects will involve deeper integration of technlogies a stronger focus on circular economics, and more robust, citizen-centric goverance models.