**SUSTAINABLE SMART CITY ASSISTANT USING IBM GRANITE LLM**

**1.Introduction**

**Project title: sustainable smart city**

**Team members: L.Saniya Florence**

**Team members: S.Gobika**

**Team members: V. Sneha**

**Team members: S .Mathisha**

**2.project overview**

* **Purpose**

The purpose of a Sustainable Smart City Assistant is to empower cities and the 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, citizens through personalized tips and services. For city official ,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:

Natural language interaction, allows citizens and officials to ask questions, get updates, and receive guidance in plain language.

Policy Summarization:

Simplified policy understanding, converts lengthy government documents into concise, actionable summaries.

Predictive analytics, estimates feature energy, water, and waste usage using historical and real-time data.

Eco-Tip Generator:

Personalized sustainability advice, recommends daily actions to reduce environmental impact based on user behavior.

Citizen Feedback Loop:

Community engagement, collects and analyzes public input to inform city planning and service improvements.

KPI Forecasting:

Strategic planning support, projects key performance indicators to help official track progress and plan ahead.

Anomaly Detection:

Early warning system, identifies unusual patterns in sensor or usage data to flag potential issues.

Multi model Input Support:

Flexible data handling, accepts text, PDFs and CSVs for document analysis and forecasting.

Stream lit or Gradio UI:

User-friendly interface, provides on intuitive dashboard for both citizens and city officials to interact with the assistant.

**3.Archtecture**

Frontend(Stream lit):

The frontend is build with stream lit, offering on 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, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

LLM Integration(IBM watsonx Granit):

Granite LLM models from IBM watsonx are used for natural language understanding and generation. Prompts are carefully designed to generate summaries, sustainability tips and repots.

Vector search(Pinecone):

Uploaded policy documents are embedded using Sentence Transformers and stored in pinecone. Semantic search is implemented using cosine similarity to allow users to search documents using natural language queries.

**4.Setup Insructions:**

**Prerequisites:**

* Python 3.9 or later
* Pip and virtual environment tools
* API keys for IBM Watsonx and Pinecone
* Internet access to access cloud services

**Installation Process:**

* Clone the repository
* Install dependencies from requirements.txt
* Create a .env file and configure credentials
* Run the backend server using Fast API
* Launch the frontend via Stream lit
* Upload data and interact with the modules

**5.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.
* All interaction are real –time and use backend APIs to dynamically update the frondend.

**6.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

**7.Authentication:**

* Token-based authentication(JWT or API keys)
* OAuth2 with IBM Cloud credentials
* Role-based access(admin, citizen, researcher)
* Planned enhancements include user sessions and history tracking.8.Authentication

**8.User interface:**

* Sidebar with navigation
* KPI visualizations with summary cards
* Tabbed layout for chat, eco tips and forecasting
* Real-time from handling
* PDF report download capability

**9.Testing**

Testing was done in multiple phases:

Unit Testing : For prompt engineering functions utility scripts

API Testing: Via Swagger UI, Postman,and test script

Edge Case Handling: Malformed inputs, large files, invalid API keys