**CITIZEN AI: INTELLIGENT CITIZEN**

**ENGAGEMENT PLATFORM**

**PROJECT DOCUMENTATION**

**INTRODUCTION:**

**PROJECT TITLE :** **CITIZEN AI**

**TEAM LEADER** : BAGULIKA L

**TEAM MEMBERS**: ELAVARASU PANDIYAN E

KALAI SELVAN R

SURYA S

# 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 into concise, 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 to inform 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

**Key Point:**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:

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, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

## LLM Integration (IBM Watsonx Granite):

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

## 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.

## ML Modules (Forecasting and Anomaly Detection):

Lightweight ML models are used for forecasting and anomaly detection using Scikit-learn. Time-series data is parsed, modeled, and visualized using pandas and matplotlib.

# Setup Instructions

## 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

# 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 Stream lit pages, card layouts, and form UIs.

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

granite\_llm.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\_forecaster.py – Forecasts future energy/water trends using regression. anomaly\_file\_checker.py – Flags unusual values in uploaded KPI data.

report\_generator.py – Constructs AI-generated sustainability reports.



