Citizen AI: Project Documentation

Team Members

- **Team Leader**: DAYAANITHI S
- **Team Member**: Aakaash N
- **Team Member**: AJAY DEV R
- **Team Member**: Akash T

1. Project Overview

The core purpose of **Citizen AI** is to empower cities and their residents to become more eco-conscious and connected. By leveraging AI and real-time data, this assistant optimizes essential resources like energy, water, and waste. It also offers personalized tips to guide citizens toward sustainable behaviors. For city officials, the assistant acts as a decision-making tool, providing insights, forecasting, and summaries of complex policies to support strategic planning.

Features

- **Conversational Interface**: Allows citizens and officials to interact using natural language.
- **Policy Summarization**: Converts lengthy government documents into concise, actionable summaries.
- **Resource Forecasting**: Estimates future energy, water, and waste usage using historical and real-time data.
- **Eco-Tip Generator**: Recommends personalized daily actions to reduce environmental impact.
- **Citizen Feedback Loop**: Collects and analyzes public input to inform city planning.
- **KPI Forecasting**: Projects key performance indicators to help officials track progress.
- **Anomaly Detection**: Identifies unusual patterns in data to flag potential issues.
- **Multimodal Input Support**: Accepts text, PDFs, and CSVs for analysis and forecasting.
- **Streamlit or Gradio UI**: Provides a user-friendly dashboard for both citizens and officials.

2. Architecture

- **Frontend (Streamlit)**: An interactive web UI with multiple pages for dashboards, file uploads, and a chat interface. Navigation is handled through a sidebar.
- **Backend (FastAPI)**: A REST framework that powers API endpoints for all key functionalities, including document processing and chat interactions.
- **LLM Integration (IBM Watsonx Granite)**: The model is used for natural language understanding and generation, carefully designed to generate summaries and sustainability tips.

- **Vector Search (Pinecone)**: Uploaded policy documents are embedded and stored in Pinecone, allowing for semantic search using natural language queries.
- **ML Modules**: Lightweight ML models from **Scikit-learn** are used for forecasting and anomaly detection. Time-series data is modeled and visualized with **pandas** and **matplotlib**.

3. Setup and Execution

Prerequisites

- Python 3.9 or later
- pip and virtual environment tools
- API keys for IBM Watsonx and Pinecone
- Internet access for cloud services

Installation Process

1. Clone the repository.
2. Install dependencies from requirements.txt.
3. Create a .env file and configure your credentials.
4. Run the backend server using FastAPI.
5. Launch the frontend via Streamlit.
6. Upload data and interact with the modules.

4. API Documentation

The backend offers several APIs, which are documented in Swagger UI:

- POST /chat/ask: Accepts a user query and returns an AI-generated message.
- POST /upload-doc: Uploads and embeds documents in Pinecone.
- GET /search-docs: Returns semantically similar policies to a user's query.
- GET /get-eco-tips: Provides sustainability tips.
- POST /submit-feedback: Stores citizen feedback.

5. User Interface

The interface is designed to be minimalist and functional, with a focus on accessibility. Key elements include a sidebar for navigation, KPI visualizations with summary cards, and real-time form handling.

6. Testing

The project underwent multiple testing 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**: For malformed inputs and other potential issues.