

## **Project Documentation**

### **1. Introduction**

**Project Title:** Citizen AI – Intelligent Citizen Engagement Platform

**Team Leader:** Sandhiya. S

**Team Members:**

**Vanisri. M**

**Satheeswari. R**

**Lakshmi. R**

**Leelavathi. G**

### **2. Project Overview**

**Purpose:**

The purpose of the Citizen AI Intelligent Engagement Platform is to empower citizens and local government officials to enhance communication, participation, and decision-making within their community. Leveraging AI and real-time data, the platform enables personalized citizen services, policy understanding, real-time feedback, and smart resource management. It promotes transparency, efficiency, and proactive engagement for smarter community development.

**Features:**

**Conversational Interface:** Allows citizens and officials to ask questions, report issues, and get updates in plain language.

**Policy Summarization:** Converts government documents into concise, actionable summaries for easier comprehension by the public.

**Resource Usage Analytics:** Tracks local public service usage (e.g., water, electricity) to provide insights and recommendations.

**Eco-Tip Generator:** Suggests daily actions to reduce environmental impact based on citizen behavior.

**Citizen Feedback Loop:** Collects and analyzes public input on local policies and services to inform decision-making.

**KPI Dashboard:** Displays key performance indicators of local services to help officials track progress.

**Anomaly Detection:** Flags unusual patterns in usage data or citizen reports to alert officials.

**Multimodal Input Support:** Accepts text, PDF, and CSV files for analysis.

User-friendly UI (Streamlit or Gradio): Provides an intuitive dashboard for easy interaction.

### 3. Architecture

#### Frontend (Streamlit):

Interactive web interface with dashboards, document uploads, chat interface, feedback forms, and KPI visualization.

#### Backend (FastAPI):

REST API handling document processing, chat interactions, eco-tip generation, feedback management, and vector embedding.

#### LLM Integration (IBM Watsonx Granite):

Used for summarizing policies, generating eco-tips, and answering citizen queries.

#### Vector Search (Pinecone):

Embeds uploaded policy documents and enables semantic search using Sentence Transformers.

#### ML Modules (Forecasting & Anomaly Detection):

Models using Scikit-learn analyze trends and detect anomalies in service usage.

### 4. Setup Instructions

#### Prerequisites:

Python 3.9+

pip and virtual environment tools

API keys for IBM Watsonx and Pinecone

Internet access

#### Installation Process:

Clone the repository

Install dependencies via requirements.txt

Configure credentials in .env file

Start FastAPI backend server

Launch Streamlit frontend

Upload data and interact with modules

## 5. Folder Structure

app/: Backend logic

ui/: Frontend Streamlit pages

document\_embedder.py: Document embeddings

kpi\_forecaster.py: Trend analysis

anomaly\_checker.py: Anomaly detection

report\_generator.py: Report creation

## 6. Running the Application

Start FastAPI backend

Launch Streamlit frontend

Navigate via sidebar

Upload documents, use chat, view reports and summaries

## 7. API Documentation

POST /chat/ask – Handles citizen queries

POST /upload-doc – Upload and embed documents

GET /search-docs – Search relevant policies

GET /get-eco-tips – Get sustainability tips

POST /submit-feedback – Submit citizen feedback

## 8. Authentication

For demonstration: No authentication by default.

Production should use:

Token-based (JWT/API Keys)

OAuth2 (IBM Cloud)

Role-based Access (Admin, Citizen, Official)

## 9. User Interface

Minimalist design with:

Sidebar navigation

KPI summary cards

Tabbed sections for chat, eco-tips, forecasting

Real-time form handling

PDF report download

## 10. Testing

Unit tests for core functions

API tested via Swagger UI and Postman

Manual tests for file upload, chat responses

Edge case handling (large files, invalid inputs)

## 11. Screenshots

