Citizen AI: Intelligent Citizen Engagement Platform

1. Introduction

• Project title: Citizen AI: Intelligent Citizen Engagement Platform

• Team leader:ASHIK AHAMED R

• Team member:AATHIL MOHAMED T

• Team member:SAKTHIGOPI B

2. Project Overview

Purpose:
The purpose of Citizen AI: Intelligent Citizen Engagement Platform is to strengthen collaboration between citizens and city officials by creating a more transparent, inclusive, and participatory urban environment. By leveraging AI and real-time data, the platform empowers residents to voice opinions, access civic services, and understand policies more easily. For decision-makers, it provides insights, feedback analysis, and forecasting tools that support evidence-based governance. Ultimately, Citizen AI fosters a smarter dialogue between communities and governments, making cities more connected, responsive, and people-centered.

Features:
• Conversational Interface – Enables citizens and officials to communicate in plain language.
• Policy Summarization – Transforms lengthy policy documents into concise insights.
• Citizen Feedback Loop – Collects and analyzes public input for planning and governance.
• Engagement Analytics – Tracks participation levels, trending concerns, and satisfaction metrics.
• KPI Forecasting – Projects civic performance indicators for strategic governance.
• Anomaly Detection – Identifies unusual spikes in citizen concerns or engagement trends.
• Multimodal Input Support – Accepts text, PDFs, and CSVs for analysis.
• User-Friendly Interface – Provides intuitive dashboards for citizens and officials.

3. Architecture

Frontend (Streamlit):
The frontend is built with Streamlit, offering an interactive web UI with dashboards, file uploads, chat, feedback forms, and report viewers. Navigation is handled through a sidebar with modular pages for scalability.

Backend (FastAPI):
FastAPI serves as the backend REST framework powering APIs for document processing, chat, feedback analysis, report creation, and vector embedding. It is optimized for asynchronous performance with Swagger integration.

LLM Integration (IBM Watsonx Granite):
Granite LLM models provide natural language understanding for summarization, chat, and engagement insights.

Vector Search (Pinecone):
Policy documents and feedback data are embedded using Sentence Transformers and stored in Pinecone for semantic search.

ML Modules (Forecasting and Anomaly Detection):

Lightweight ML models forecast engagement KPIs and detect anomalies in participation and sentiment trends.

4. Setup Instructions

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

Installation Process:
1. Clone the repository
2. Install dependencies from requirements.txt
3. Configure credentials in a .env file
4. Run the FastAPI backend server
5. Launch the Streamlit frontend
6. Upload data and interact with the modules

5. Folder Structure

app/ – FastAPI backend logic
app/api/ – API routes for chat, feedback, reports, embeddings
ui/ – Streamlit pages and layouts
dashboard.py – Launches main dashboard
granite_llm.py – Handles Watsonx LLM communication
document_embedder.py – Converts documents to embeddings
kpi_forecaster.py – Forecasts engagement KPIs
anomaly_checker.py – Detects anomalies in citizen data
report_generator.py – Generates AI-based engagement reports

6. Running the Application

• Launch FastAPI backend
• Run Streamlit dashboard
• Navigate via sidebar
• Upload feedback or policy documents
• Interact with chat assistant
• View reports, summaries, predictions
• Real-time backend APIs dynamically update frontend

7. API Documentation

POST /chat/ask – Citizen query → AI-generated response
POST /upload-doc – Uploads and embeds documents
GET /search-docs – Finds semantically similar policies
GET /get-engagement-analytics – Provides citizen engagement insights
POST /submit-feedback – Stores citizen feedback

8. Authentication

• Token-based authentication (JWT/API keys)
• OAuth2 with IBM Cloud credentials
• Role-based access (admin, citizen, researcher)
• Future: sessions and history tracking

9. User Interface

• Sidebar navigation
• KPI visualizations and summary cards
• Tabs for chat, analytics, forecasting
• Real-time feedback forms
• PDF report downloads

10. Testing

• Unit testing for core functions
• API testing with Swagger UI/Postman
• Manual testing for uploads, chat, outputs
• Edge cases: malformed inputs, large files, invalid API keys

11. Screenshots

To be added.

12. Known Issues

To be documented.

13. Future Enhancements

Planned improvements: multi-language support, deeper analytics, voice-enabled interaction.