Citizen AI: Intelligent Citizen Engagement Platform

1. Introduction

Project Title: Citizen AI: Intelligent Citizen Engagement Platform

Team ID : NM2025TMID10933

Team Leader : Dinesh Si

Team member : Mathan Kumar C

Team member : Perarasu S

Team member : Rapart Thom A

Objective: Citizen AI aims to revolutionize government-citizen interactions by leveraging generative AI technology. The platform addresses the growing need for digital transformation in public services. By automating information delivery, enabling real-time citizen feedback, and resolving inquiries more effectively, the solution enhances transparency, operational efficiency, and public trust. It empowers citizens to easily obtain personalized information about government policies, civic rights, and service procedures.

2. Project Overview

Purpose:

- Improve the communication channels between government and citizens.

- Facilitate easier and faster access to public service information.

- Empower citizens by simplifying governance information.

- Collect structured and actionable citizen feedback to inform policy adjustments.

Features:

1. Conversational Interface: Citizens can engage with the system through a simple, intuitive chatbot. The natural language processing model handles complex query patterns, understands context, and provides follow-up questions or relevant answers.

Example: A citizen may ask, 'How can I renew my driver's license?' The system responds with eligibility criteria, required documents, and a link to the official portal.

2. Sentiment Analysis Dashboard: Uses advanced sentiment algorithms to detect public opinion trends, categorizing feedback into positive, negative, or neutral. This helps government officials monitor public satisfaction and adjust strategies accordingly.

Example: An increase in negative sentiment around public transportation indicates a potential service issue, prompting further investigation.

3. Policy Summarization: Converts dense legal documents and policy papers into human-readable summaries, focusing on key points, eligibility, and procedures. Users can query specific sections without needing legal expertise.

Example: A citizen may input 'What are my rights regarding data privacy under the new law?' and get a concise summary.

4. Resource Navigation & Discovery: Provides an intelligent directory of government departments, service contacts, and online portals. The system dynamically suggests the most relevant resources based on the user's query.

5. Multilingual Support: The system is designed to support multiple languages including English, Hindi, Spanish, and more, allowing non-native speakers to engage effectively with public services.

3. Architecture

The system is architected around scalability, modularity, and fault tolerance principles.

Components:

- **Frontend**: Developed using Gradio, selected for its simplicity and ability to quickly prototype web-based interfaces that include chat and dashboard views.

- **Backend API:** Implemented in Flask, serving as the core communication layer. It validates and routes requests, manages session handling, and integrates with the LLM service.

- **LLM Service:** IBM Granite models serve as the AI engine, delivering accurate and context-aware responses. These models are pre-trained and fine-tuned for policy and civic engagement tasks.

- **Database:** A PostgreSQL database stores user queries, session histories, model responses, and sentiment metrics for historical analysis and reporting.

Data Flow:

Step 1: Citizen submits a query in the Gradio frontend.

Step 2: Query is sent securely to the Flask backend.

Step 3: Backend preprocesses input and interacts with the IBM Granite API.

Step 4: Response is generated by the LLM and returned to the backend.

Step 5: Query-response pairs are stored in the database.

Step 6: The frontend displays the response in an interactive manner.

Step 7: Sentiment and analytics dashboards are updated in real time.

4. Setup Instructions

Prerequisites:

- Install Python 3.8+ and Git.

- Familiarity with creating and managing virtual environments.

- Basic knowledge of Gradio to customize the UI.

- Google Colab account for GPU-based model inference.

- IBM Watsonx API key obtained via the IBM Cloud platform.

Setup Workflow:

1. Clone the GitHub Repository:

- Command: `git clone https://github.com/example/CitizenAI.git`

2. Create and activate virtual environment:

- Command: `python -m venv venv && source venv/bin/activate`

3. Install dependencies:

- `pip install -r requirements.txt`

4. Configure API Key:

- Store API key in `.env` file or system environment variable.

Running the Application:

1. Open Google Colab and select 'T4 GPU' runtime.

2. Mount Google Drive if needed for model storage.

3. Execute necessary pip install commands to install libraries.

4. Clone the repo and run the main script:

- `!python app.py`

5. Gradio provides a public URL for interaction.

5. User Interface

Design Principles:

- Simple, clear interface focusing on usability for all age groups.

- Clean two-panel layout: Chatbot on left, dashboard on right.

Citizen Services Chatbot:

- Text input box with auto-suggestions for popular queries.

- Chat history is persisted across sessions.

City Analytics Dashboard:

- Sentiment Score Cards: Displays sentiment breakdown in graphical format.

- Topic Word Cloud: Highlights frequently asked topics visually.

- Feedback Heatmap: Displays query volume and sentiment geographically using interactive maps.

Service & Policy Lookup:

- Search bar supporting keyword search or direct policy number lookup.

- Links to official websites and contact info are displayed dynamically.

6. Authentication

Current Implementation:

- No user authentication yet – open demo mode.

Planned Security Features:

1. OAuth2 Integration:

- For government officials, leveraging IBM Cloud identity services.

2. JWT Authentication:

- Enables token-based secure user sessions for citizens.

3. Role-Based Access Control (RBAC):

- Citizens: Can interact with chatbot and lookup policies.

- Officials: Can access analytics dashboards, monitor feedback trends.

4. API Key Management:

- API keys stored securely and encrypted using environment variables.

7. Known Issues

1. Hallucinations: Occasional generation of incorrect or misleading information.

2. Rate Limiting: Potential slowdowns during peak usage due to API rate limits.

3. Data Privacy: PII management requires GDPR-compliant solutions.

4. Latency: Processing complex queries may introduce 1-3 seconds delay.

5. No Live Updates: Responses do not reflect latest government updates without external data feeds.

8. Future Enhancements

1. Real-Time Database Integration:

- Query live status of government services.

2. Proactive Notifications:

- Push alerts via SMS or email about policy changes or service outages.

3. Specialized LLM Integrations:

- Add models for legal advice and policy recommendation systems.

4. Mobile Application Development:

- Native apps for Android and iOS with offline mode.

5. Advanced Predictive Analytics:

- Forecast demand for specific services based on trends.

6. Multimodal Interface:

- Allow voice commands and document image upload for automatic processing.

9. Testing

Unit Testing:

- Test individual functions, e.g., prompt formatting, API key handling, error catching.

API Testing:

- Use Postman to simulate different input cases and validate correct responses.

Manual Testing:

- Walkthrough typical user interactions to ensure end-to-end experience remains intuitive.

Edge Case Testing:

- Inputs like empty strings, special characters, or very long paragraphs.

- Attempt security bypass using malformed inputs.

10. Deployment

Local Deployment:

- Docker containers ensure consistency across environments.

Cloud Deployment:

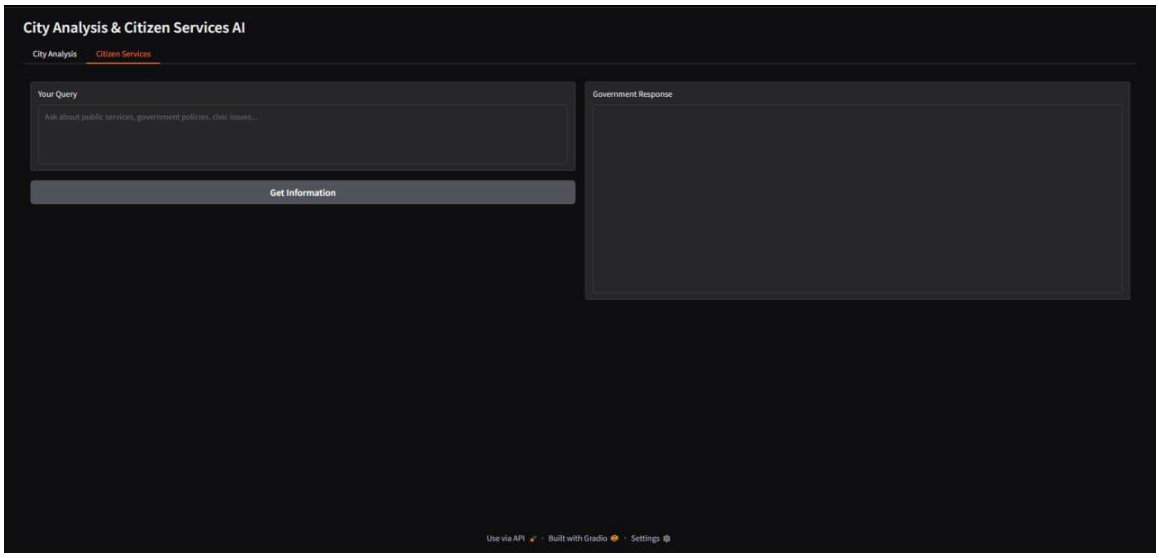
- Google Cloud Run provides auto-scaling based on traffic.

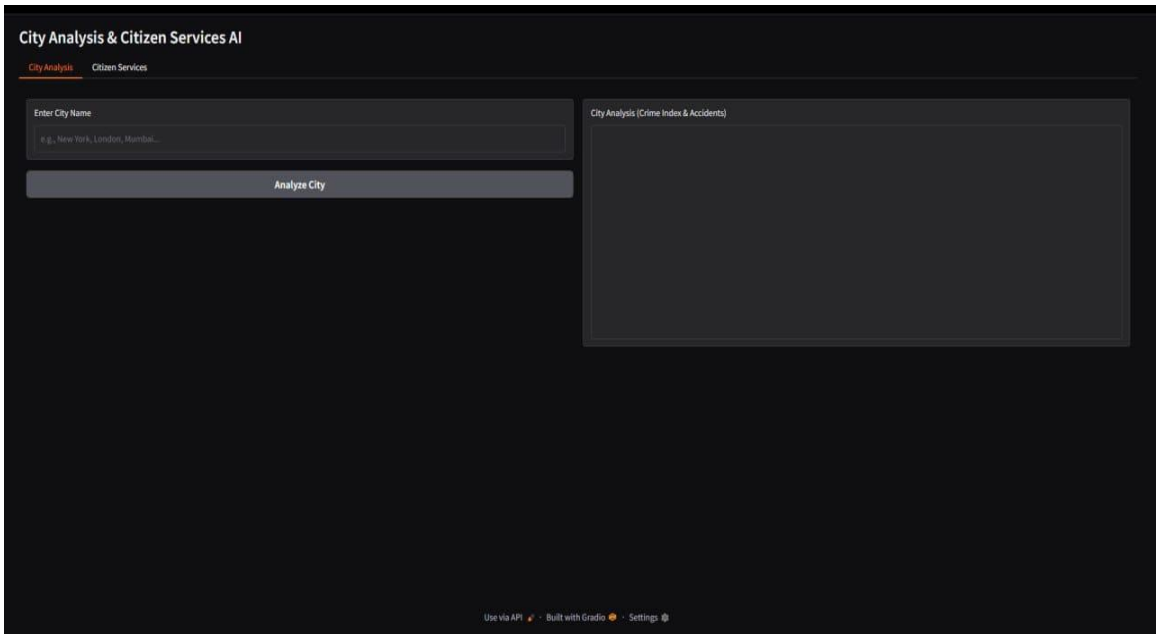
- IBM Kubernetes Engine enables full orchestration for production-grade deployment.

Scalability:

- Stateless design of the LLM Service allows horizontal scaling without session affinity.

11. Screenshots





12. References

- Flask Framework: https://flask.palletsprojects.com/

- IBM Granite Models: https://huggingface.co/ibm/granite-3.2-2b-instruct

- Gradio Framework: https://gradio.app/

- Google Colab Documentation: https://research.google.com/colaboratory/