Project Title

Citizen AI:Intelligent Citizen Engagement Platform

Project Documentation

1. Introduction

Project Title: Citizen AI

Team Leader: [AJAY G]

Team Member: [BALU S]

Team Member: [GULAM AHAMED RAZA Y]

Team Member: [ABDUL SALAM A]

2. Project Overview

- Purpose:

Citizen AI is designed to improve governance and community engagement by providing citizens with an accessible platform to interact with government policies, services, and updates. It leverages AI to deliver accurate information, summarize documents, collect citizen feedback, and assist officials in decision-making. The system empowers both citizens and authorities by ensuring transparency, inclusivity, and real-time communication.

- Features:

Conversational Interface

Key Point: Natural language Q&A

Functionality: Citizens can ask questions about policies, schemes, or services and receive AI-powered responses.

Policy Summarization

Key Point: Simplified understanding

Functionality: Converts lengthy documents into concise summaries.

Citizen Feedback Collection

Key Point: Community engagement

Functionality: Collects opinions and suggestions from citizens for governance improvements.

Service Recommendation

Key Point: Personalized assistance

Functionality: Suggests government schemes and services based on user profile.

Data Analytics for Officials

Key Point: Informed decisions

Functionality: Provides insights from citizen feedback and queries.

Multimodal Support

Key Point: Flexible inputs

Functionality: Accepts text, PDFs, and voice queries.

User-Friendly Dashboard

Key Point: Accessibility

Functionality: Intuitive design for both citizens and government officials.

3. Architecture

Frontend (Streamlit/React): Interactive dashboard with chat, document upload, and visualization panels.

Backend (FastAPI): API-driven backend for chat, feedback collection, and report generation.

LLM Integration: AI model for natural language processing, summarization, and recommendations.

Vector Database (Pinecone/FAISS): Stores embedded documents for semantic search.

Analytics Module: Processes citizen feedback and generates visual reports for policymakers.

4. Setup Instructions

Prerequisites:

- Python 3.9+
- Virtual environment setup
- API keys for AI/Vector DB services
- Internet connection

Installation Steps:

1. Clone repository

2. Install dependencies from requirements.txt

3. Configure .env with credentials

4. Run backend server (uvicorn)

5. Launch frontend (streamlit run app.py)

6. Interact with chatbot, upload docs, and view analytics

5. Folder Structure

```
citizen_ai/
├── app/                # Backend API (FastAPI)
│   ├── routes/         # API routes (chat, feedback, reports)
│   ├── models/         # Data models
│   └── services/       # Business logic
│
├── ui/                 # Frontend (Streamlit)
│   ├── pages/          # Dashboard pages
│   └── components/     # UI components
│
├── embeddings/         # Document embeddings storage
├── analytics/          # Data processing & reports
├── main.py             # Entry point for backend
├── dashboard.py        # Entry point for frontend
```

6. Running the Application

1. Start FastAPI backend.

2. Run Streamlit dashboard.

3. Upload documents or submit queries.

4. View AI-generated summaries, reports, and citizen feedback analytics.

7. API Documentation

POST /chat/ask → Query the AI assistant

POST /upload-doc → Upload policy or scheme document

GET /search-docs → Search relevant documents

GET /recommend-service → Get suggested schemes

POST /submit-feedback → Store citizen feedback

8. Authentication

Token-based authentication (JWT/API keys)
Role-based access: Citizen, Official, Admin

Future scope: OAuth2 integration with government digital IDs

9. User Interface

Sidebar navigation

Chat window for Q&A

Policy/document summarization tab

Feedback form

Analytics dashboard with visual charts

Downloadable reports

10. Testing

Unit Testing: Core AI functions & utilities

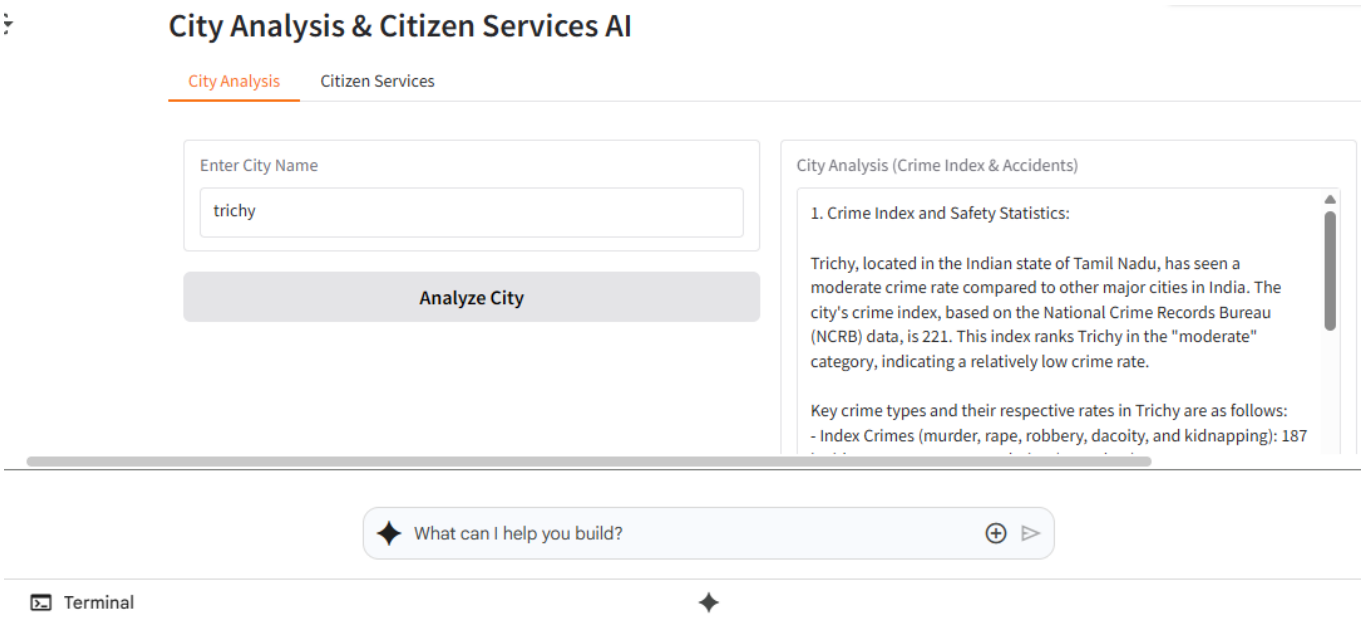
API Testing: Postman & Swagger UI

Manual Testing: User interactions & document handling

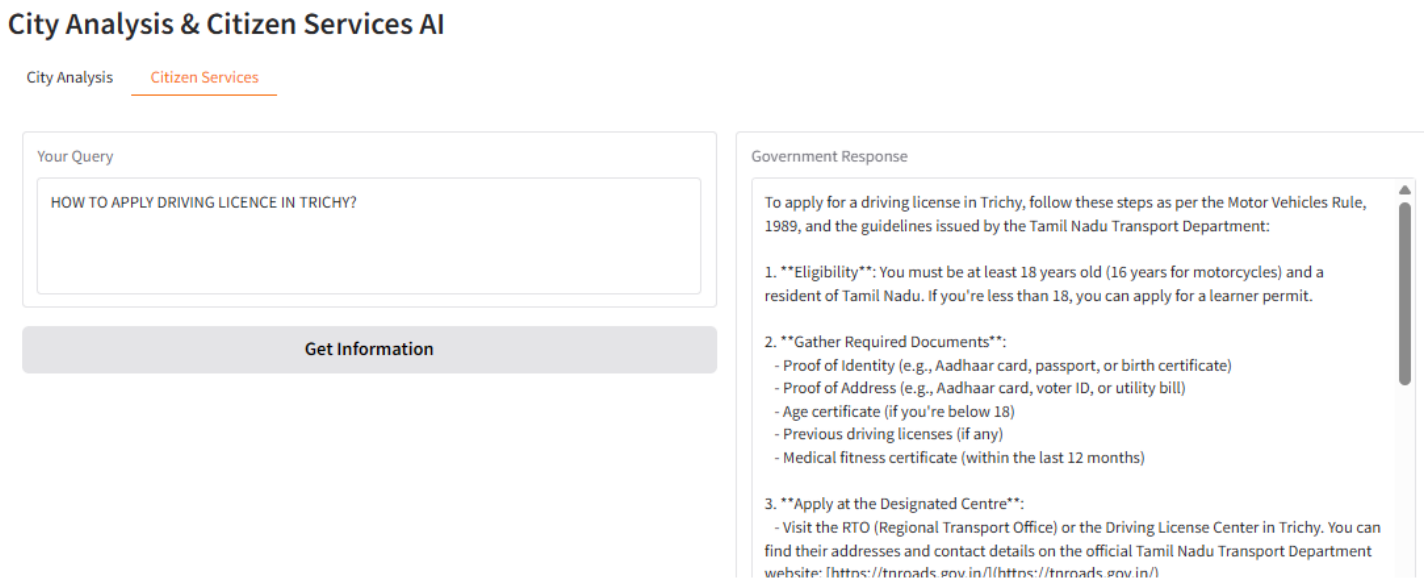
Edge Cases: Malformed input, large documents, invalid keys

11. Screenshots :

SCREENSHOT 1:



SCREENSHOT 2:



12. Known Issues

Limited support for regional languages (planned enhancement)

Requires stable internet connection

Large document embedding may slow response

13. Future Enhancements

Multilingual support

Mobile app integration

Voice-based interaction

Blockchain-enabled feedback transparency

Predictive analytics for policy planning