Citizen AI: Intelligent Citizen Engagement Platform

Generative AI with IBM

PROJECT DOCUMENTATION

1.Introduction

Project title: Intelligent Citizen Engagement Platform

Team leader: Dharshini. U

1. Team member: Abinaya. B
2. Team member: Abinaya. S
3. Team member: Archana. M

2. Project Overview

The Intelligent Citizen Engagement Platform (ICEP) is a digital solution designed to enhance the interaction between citizens and government authorities. It leverages AI, data analytic s, and multi-channel communication to streamline service delivery, ensure transparency, and improve responsiveness. By integrating online services, grievance readdress, feedback collection, and real-time updates, the platform empowers citizens with easy access to government services while enabling authorities to make data-driven decisions for efficient governance.

3. Key Features

1. Unified Service Access

Single-window portal/mobile app for accessing multiple government services.

2. AI-powered Virtual Assistant

Chatbot/voice bot for instant query resolution, service navigation, and updates.

3. Grievance Redressal System

Easy complaint registration, tracking, and automated escalation.

4. Real-time Notifications & AlertsSMS, email, and app notifications for service status, deadlines, and emergencies.

5. Citizen Feedback & Surveys

Integrated feedback forms, polls, and sentiment analysis to assess satisfaction.

6. Data-driven Insights

Analytics dashboard for authorities to track service usage, complaints, and trends.

7. Multilingual & Inclusive Support

Services available in regional languages, with accessibility features for all citizens

8. Secure Digital Transactions

Online payment gateway for bills, taxes, and government fees with data protection

9. Location-based Services

Geo-enabled features like nearest service centers, local announcements, and events.

10. Omnichannel Engagemen

Seamless interaction across web, mobile app, WhatsApp, call centers, and kiosks.

4.Scenarios:

1. Citizen Access – A resident logs into the platform via mobile app, web portal, or kiosk to access government services.

2. Smart Assistance – An AI chatbot/voice bot guides them in their preferred language to the required service.

3. Service Delivery – The citizen applies, pays, or registers a grievance online with instant confirmation/receipt.

4. Feedback & Tracking – The system allows real-time status tracking and collects quick feedback on the service.

5. Government Insights – Authorities view analytics dashboards to monitor requests, complaints, and citizen satisfaction for better decision-making.

5.Project Flow – Citizen AI

1. . Login / Onboarding – Citizen registers using mobile or digital ID.
2. Service Selection – Chooses service (bills, certificates, grievances).
3. AI Guidance – Chatbot/voice bot helps fill details and submit request.
4. Processing – Request auto-routed to the right department.
5. Updates – Citizen gets real-time status alerts (SMS, app, email).
6. Completion – Service delivered digitally or scheduled in-person.
7. Feedback – Citizen gives quick rating/suggestion.
8. Analytic s – Government dashboard shows trends and satisfaction.

6.Architecture

1. User Layer – Mobile app, web portal, kiosks, call center, chatbot/voice bot.

2. Application Layer – Service modules (bills, certificates, grievances), notifications, feedback, scheduling.

3. Integration Layer – API gateway, middleware, secure payment systems.

4. Data & Intelligence Layer – Citizen database, AI/ML, analytics dashboards, security controls.

5. Infrastructure Layer – Cloud hosting, cybersecurity, backup & disaster recovery.

7.Setup instructions

1. Infrastructure Setup – Deploy cloud servers with security and backup.

2. Database Configuration – Create citizen data repository and link govt. databases.

3. Integration Layer – Configure API gateway for department and payment systems.

4. Application Deployment – Install service modules, chatbot, notification engine.

5. User Access Setup – Launch web portal, mobile app, kiosks, and call center.

6. Testing & Security – Run end-to-end testing with encryption and role-based access.

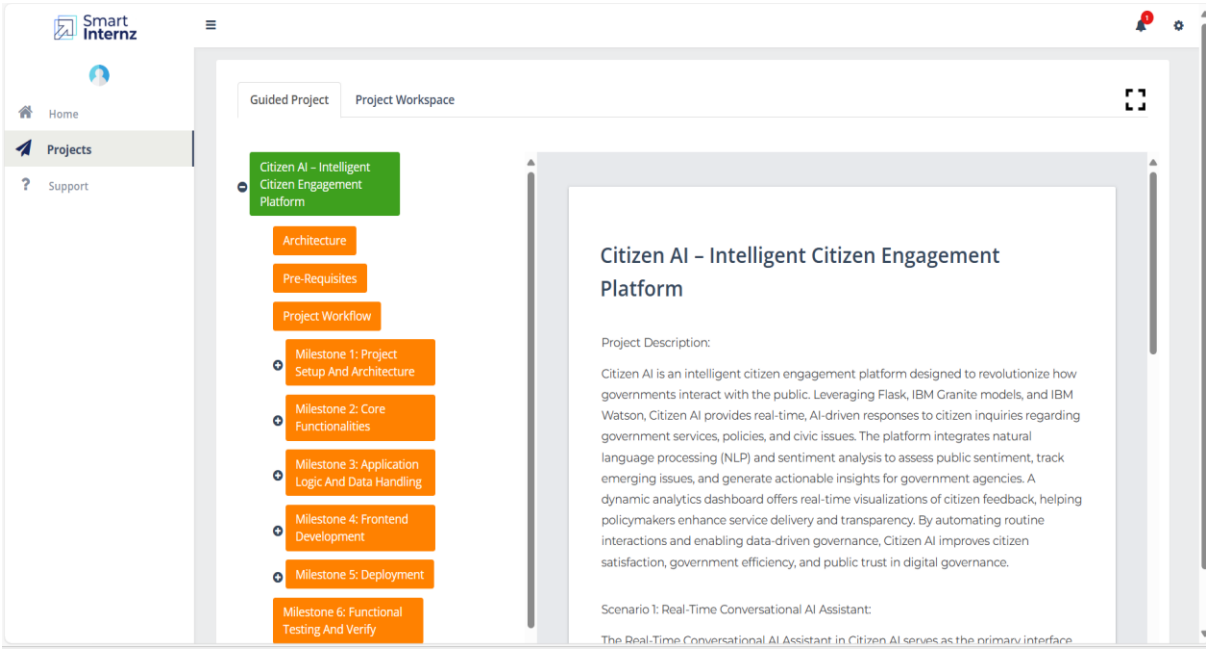
7. Go-Live & Training – Train staff, onboard citizens, and roll out in phases.

8.Milestones

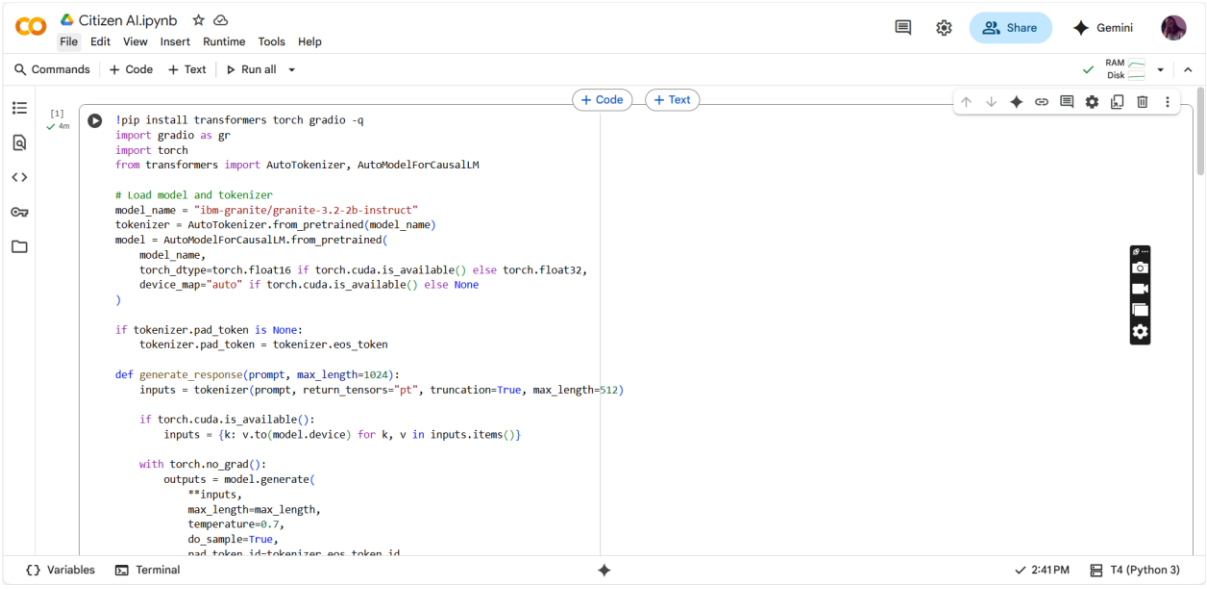
1. Requirement Gathering – Collect citizen and government needs.
2. System Design – Finalize architecture, modules, and integration plan.
3. Infrastructure Setup – Deploy cloud, databases, and security framework.
4. Module Development – Build services (grievance, payments, certificates).
5. Integration – Connect with government databases and payment gateways.
6. Testing Phase – Functional, security, and user acceptance testing.
7. Pilot Launch – Roll out in selected zones for feedback.
8. Full Deployment – City-wide platform launch.
9. Training & Adoption – Educate staff and citizens for smooth use.
10. Monitoring & Improvement – Continuous analytics and feature upgrades.

Would you like me to compress this to just 5 key

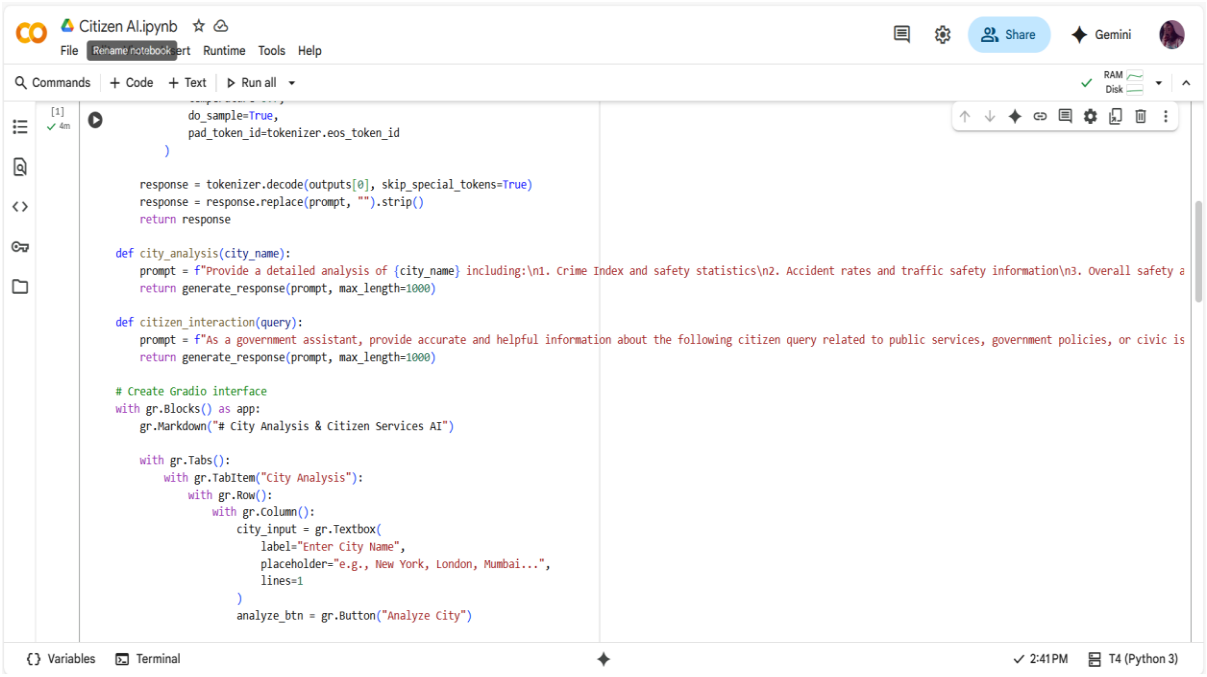
9.Project screenshots



10.Coding

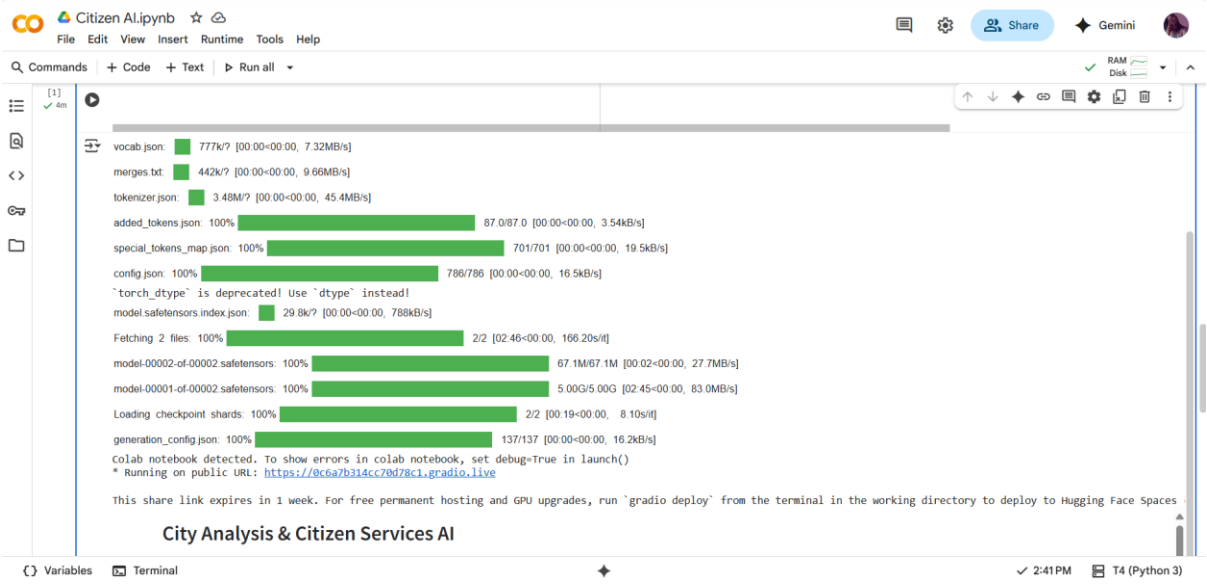


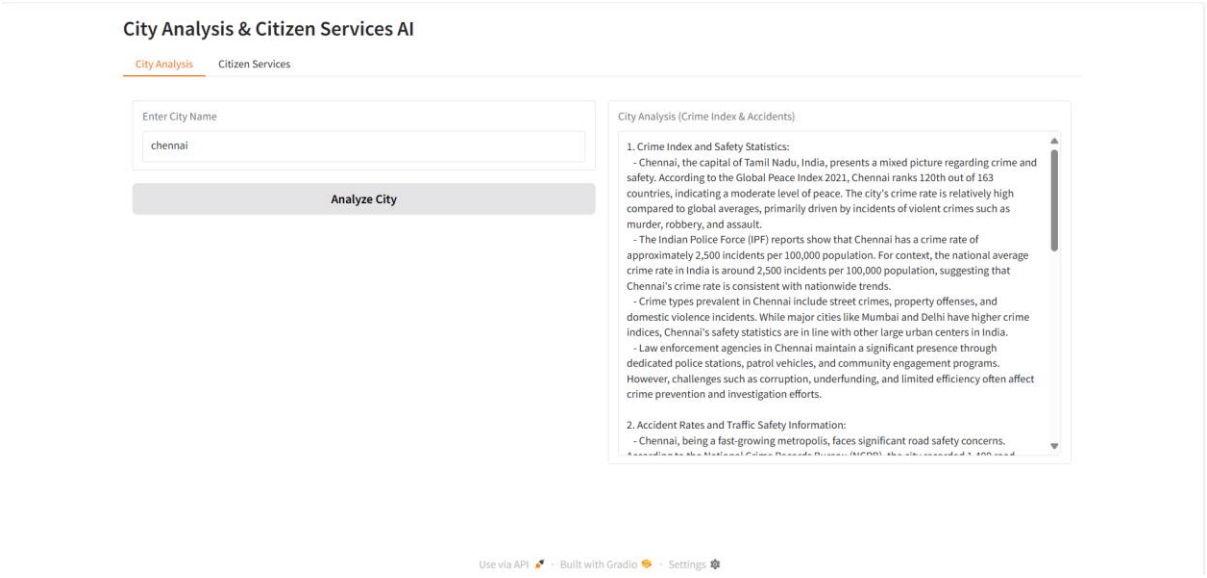
```
!pip install transformers torch gradio -q
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
```

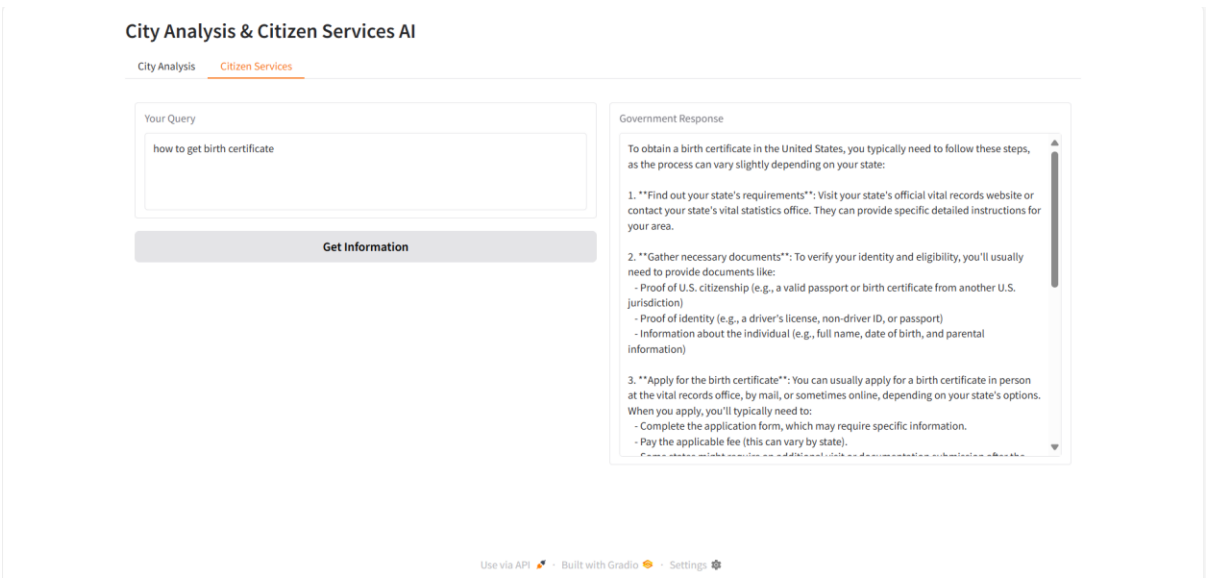


```
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response

def city_analysis(city_name):
    prompt = f"Provide a detailed analysis of {city_name} including:\n1. Crime Index and safety statistics\n2. Accident rates and traffic safety information\n3. Overall safety a
    return generate_response(prompt, max_length=1000)

def citizen_interaction(query):
    prompt = f"As a government assistant, provide accurate and helpful information about the following citizen query related to public services, government policies, or civic is
    return generate_response(prompt, max_length=1000)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# City Analysis & Citizen Services AI")

    with gr.Tabs():
        with gr.TabItem("City Analysis"):
            with gr.Row():
                with gr.Column():
                    city_input = gr.Textbox(
                        label="Enter City Name",
                        placeholder="e.g., New York, London, Mumbai...",
                        lines=1
                    )
                    analyze_btn = gr.Button("Analyze City")
```

11.Output


Citizen AI.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
vocab.json: 777k/? [00:00<00:00, 7.32MB/s]
merges.txt: 442k/? [00:00<00:00, 9.66MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 45.4MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 3.54kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 19.5kB/s]
config.json: 100% 786/786 [00:00<00:00, 16.5kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 788kB/s]
Fetching 2 files: 100% 2/2 [02:46<00:00, 166.20s/it]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:02<00:00, 27.7MB/s]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:45<00:00, 83.0MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 8.10s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 16.2kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://0c6a7b314cc70d78c1.gradio.live
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
City Analysis & Citizen Services AI
Variables
Terminal
2:41 PM
T4 (Python 3)


City Analysis & Citizen Services AI
City Analysis
Citizen Services
Enter City Name
chennai
Analyze City
City Analysis (Crime Index & Accidents)
1. Crime Index and Safety Statistics:
- Chennai, the capital of Tamil Nadu, India, presents a mixed picture regarding crime and safety. According to the Global Peace Index 2021, Chennai ranks 120th out of 163 countries, indicating a moderate level of peace. The city's crime rate is relatively high compared to global averages, primarily driven by incidents of violent crimes such as murder, robbery, and assault.
- The Indian Police Force (IPF) reports show that Chennai has a crime rate of approximately 2,500 incidents per 100,000 population. For context, the national average crime rate in India is around 2,500 incidents per 100,000 population, suggesting that Chennai's crime rate is consistent with nationwide trends.
- Crime types prevalent in Chennai include street crimes, property offenses, and domestic violence incidents. While major cities like Mumbai and Delhi have higher crime indices, Chennai's safety statistics are in line with other large urban centers in India.
- Law enforcement agencies in Chennai maintain a significant presence through dedicated police stations, patrol vehicles, and community engagement programs. However, challenges such as corruption, underfunding, and limited efficiency often affect crime prevention and investigation efforts.
2. Accident Rates and Traffic Safety Information:
- Chennai, being a fast-growing metropolis, faces significant road safety concerns.
Use via API
Built with Gradio
Settings


City Analysis & Citizen Services AI
City Analysis
Citizen Services
Your Query
how to get birth certificate
Get Information
Government Response
To obtain a birth certificate in the United States, you typically need to follow these steps, as the process can vary slightly depending on your state:
1. **Find out your state's requirements**: Visit your state's official vital records website or contact your state's vital statistics office. They can provide specific detailed instructions for your area.
2. **Gather necessary documents**: To verify your identity and eligibility, you'll usually need to provide documents like:
- Proof of U.S. citizenship (e.g., a valid passport or birth certificate from another U.S. jurisdiction)
- Proof of identity (e.g., a driver's license, non-driver ID, or passport)
- Information about the individual (e.g., full name, date of birth, and parental information)
3. **Apply for the birth certificate**: You can usually apply for a birth certificate in person at the vital records office, by mail, or sometimes online, depending on your state's options. When you apply, you'll typically need to:
- Complete the application form, which may require specific information.
- Pay the applicable fee (this can vary by state).
Use via API
Built with Gradio
Settings