CITIZEN AI - INTELLIGENCE CITIZEN ENGAGEMENT PLATFORM

1. Introduction:

• Project title: Citizen AI - intelligence citizen engagement platform
• Team member: Sandhiya G S
• Team member: Puvitha S
• Team member: Nivetha S
• Team member: Mohanapriya T

2. Project Overview:

Purpose:

The purpose of **Citizen AI** is to transform the way governments and communities connect by creating a transparent, responsive, and inclusive citizen engagement ecosystem. By combining AI, real-time analytics, and human-centered design, Citizen AI empowers residents to voice their concerns, share ideas, and actively participate in decision-making. For public officials, it offers actionable insights, sentiment analysis, and streamlined communication channels to ensure policies and services align with citizen needs. Ultimately, Citizen AI strengthens trust, promotes accountability, and builds a collaborative environment where every voice matters in shaping the future of governance.

Features:

Conversational Interface

- **Key Point:** Human-like interaction
- **Functionality:** Provides two-way communication where citizens can raise issues, ask questions, and get real-time updates in plain language.

Sentiment & Opinion Analysis

- **Key Point:** Understand citizen mood
- **Functionality:** Uses AI to analyze feedback, social media, and surveys to detect public sentiment and emerging concerns.

Participatory Polls & Surveys

- **Key Point:** Inclusive decision-making
- **Functionality:** Launches quick polls and detailed surveys to collect citizen input on policies, projects, and community needs.

Policy & Project Updates

- **Key Point:** Transparency and accountability
- **Functionality:** Converts complex policy drafts and project progress into simple, digestible updates for citizens.

Grievance & Issue Tracking

- **Key Point:** Streamlined resolution
- **Functionality:** Citizens can submit complaints or service requests, which are tracked and routed to the right departments with progress notifications.

AI-driven Insights for Officials

- **Key Point:** Data-informed governance
- **Functionality:** Generates dashboards showing trends, hotspots of concern, and priority areas for immediate attention.

Community Collaboration Hub

- **Key Point:** Collective problem-solving
- **Functionality:** Provides a platform for citizens to share ideas, co-create initiatives, and participate in hackathons or workshops.

Multilingual & Multimodal Support

- **Key Point:** Inclusivity across diverse populations
- **Functionality:** Supports multiple languages and input formats (voice, text, documents, images) to engage citizens of all backgrounds.

Streamlit or Gradio UI

- **Key Point:** Accessible and easy-to-use design
- **Functionality:** Offers a simple yet powerful dashboard for citizens and officials, ensuring seamless interaction and engagement.

3. Architecture:

- **Frontend (Streamlit):**
  The frontend is developed using Streamlit, providing an intuitive and interactive user interface for citizens and officials. It includes multiple modules such as community dashboards, feedback submission, polls & surveys, grievance tracking, and real-time chat. Navigation is managed through a sidebar using the `streamlit-option-menu` library. Each

component is modularized for easy updates and scalability, ensuring smooth user experience.

- **Backend (FastAPI):**
  FastAPI powers the backend as a high-performance REST framework. It handles API endpoints for citizen queries, sentiment analysis, survey data processing, grievance management, and AI-driven recommendations. Its asynchronous capabilities ensure quick responses, while Swagger integration makes it simple for developers and officials to test and manage APIs.

- **LLM Integration (IBM Watsonx Granite):**
  Granite LLM models from IBM Watsonx are leveraged for natural language understanding and generation. Custom prompt engineering enables functions such as summarizing policy documents, translating government communications into simple language, generating citizen engagement reports, and supporting multilingual responses.

- **Vector Search (Pinecone):**
  All uploaded documents (policies, community reports, citizen charters) are embedded using Sentence Transformers and stored in Pinecone. Semantic search with cosine similarity allows citizens to query documents in plain language, making government information accessible and transparent.

- **ML Modules (Sentiment Analysis & Participation Trends):**
  Machine learning models, built with Scikit-learn and NLP toolkits, are used for sentiment detection and citizen opinion mining. Time-series models track participation levels across polls and surveys, helping officials forecast engagement patterns. Visualizations are generated with pandas and matplotlib for clarity.

- **Analytics Dashboard:**
  A dedicated analytics layer presents KPIs such as top citizen concerns, engagement rates, service resolution times, and satisfaction trends. Interactive charts and filters allow officials to drill down into data for targeted decision-making.

4. Setup Instructions:

Prerequisites:

- Python 3.9 or later
- pip and virtual environment tools
- API keys for IBM Watsonx and Pinecone
- Access credentials for database (if using external citizen feedback DB)
- Internet access for cloud-based services

Installation Process:

1. Clone the repository
2. Install dependencies from `requirements.txt`
3. Create a `.env` file and configure credentials (Watsonx, Pinecone, DB connections)
4. Run the backend server using **FastAPI**
5. Launch the frontend using **Streamlit**
6. Upload documents, submit citizen feedback, or start polls/surveys through the dashboard
7. Explore analytics, reports, and AI-powered insights via the UI

5. Folder Structure:

app/ – Core backend logic built with FastAPI, including routers, models, and integration modules.

app/api/ – Subdirectory for modular API routes such as chat, surveys, feedback, grievance tracking, and document summarization.

ui/ – Frontend components built with Streamlit, including dashboards, survey forms, grievance submission pages, and community interaction panels.

citizen_dashboard.py – Entry script for launching the main Streamlit dashboard interface.

granite_llm.py – Handles communication with IBM Watsonx Granite LLM, powering summarization, multilingual responses, and engagement insights.

document_embedder.py – Converts uploaded government or community documents into embeddings and stores them in Pinecone for semantic search.

sentiment_analyzer.py – Analyzes citizen feedback and survey responses for sentiment, categorizing opinions into actionable insights.

participation_trends.py – Tracks and forecasts citizen participation levels in polls, surveys, and discussions using ML time-series analysis.

grievance_tracker.py – Manages citizen complaints and issue resolution by routing data to appropriate service APIs.

engagement_report_generator.py – Generates AI-driven reports on citizen satisfaction, feedback trends, and community insights for officials.

6. Running the Application:

To start the project:

1. Launch the **FastAPI** server to expose backend endpoints.
2. Run the **Streamlit** dashboard to access the citizen engagement interface.
3. Use the sidebar to navigate between modules like dashboards, surveys, grievance tracking, and feedback forms.
4. Upload documents or datasets, submit feedback, participate in polls, and interact with the AI assistant.
5. All activities are real-time, with backend APIs dynamically updating the frontend and dashboards.

Frontend (Streamlit):
The frontend is developed with Streamlit, providing an interactive web interface with modules such as surveys, grievance tracking, feedback collection, and analytics dashboards. Navigation is managed with the `streamlit-option-menu` library. Each page is modular for scalability and simple updates.

Backend (FastAPI):
FastAPI acts as the REST backend, exposing endpoints for feedback submission, sentiment analysis, chat interaction, document summarization, and survey participation. It is optimized for asynchronous processing and comes with Swagger UI for API inspection and testing.

7. API Documentation:

Backend APIs available include:

- **POST /chat/ask** – Accepts a citizen query and returns an AI-generated response.

- **POST /upload-doc** – Uploads and embeds documents into Pinecone for semantic search.

- **GET /search-docs** – Returns semantically similar policies or reports to a given query.

- **POST /submit-feedback** – Submits citizen feedback for storage and analysis.

- **POST /submit-grievance** – Registers a citizen complaint or issue with tracking ID.

- **GET /get-sentiment-trends** – Provides sentiment insights from citizen feedback over time.

- **GET /get-participation-data** – Returns engagement statistics from polls and surveys.

Each endpoint is tested and documented with Swagger UI, allowing quick inspection and trial during development.

8. Authentication:

This demo version runs in an open environment, but secure deployments Can integrate:

- Token-based authentication (JWT or API keys)
- OAuth2 with IBM Cloud credentials
- Role-based access control (admin, official, citizen, researcher)
- Planned enhancements: user session management and feedback history tracking

9. User Interface:

The design is minimalist and focused on accessibility for non-technical users. It includes:

- Sidebar navigation for quick access to modules
- KPI visualizations with summary cards (engagement rates, top concerns, resolution time)
- Tabbed layouts for chat, surveys, grievances, and feedback
- Real-time form handling for smooth submission
- Downloadable reports in PDF format for transparency
- Multilingual support to reach diverse communities

10. Testing:

Testing was carried out in multiple phases:

- **Unit Testing:** For prompt engineering, feedback analysis, and utility scripts

- **API Testing:** Using Swagger UI, Postman, and automated test scripts

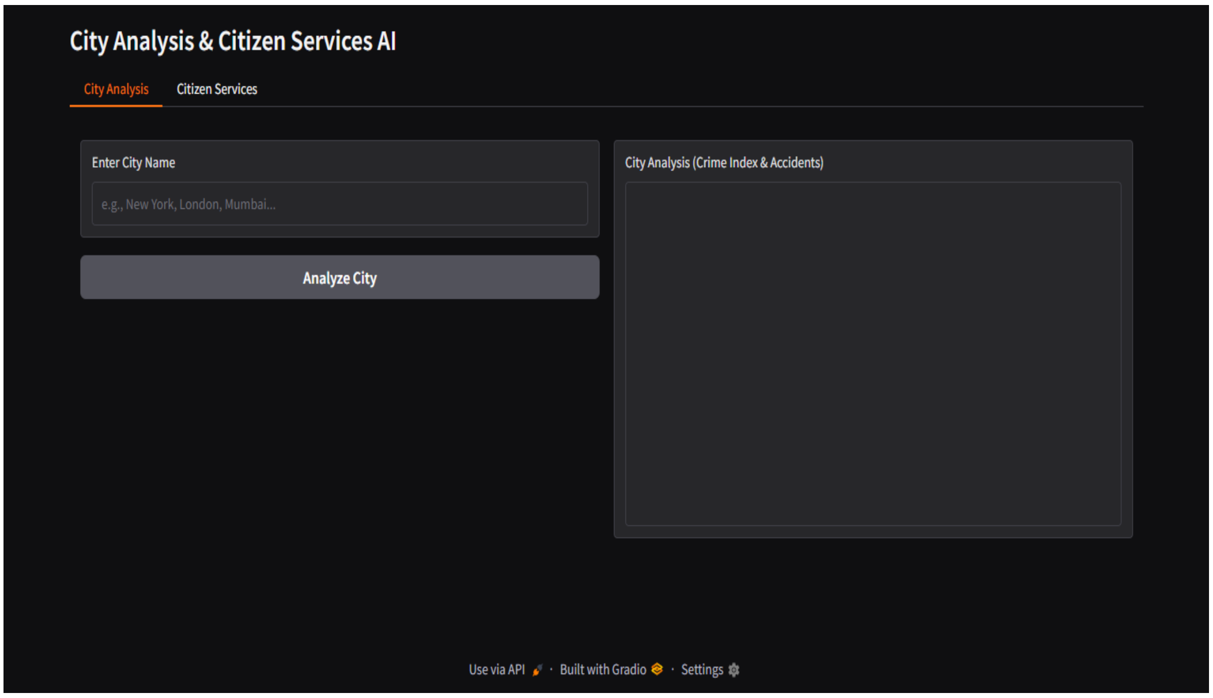
- **Manual Testing:** Covering feedback submission, chat interactions, surveys, and report generation

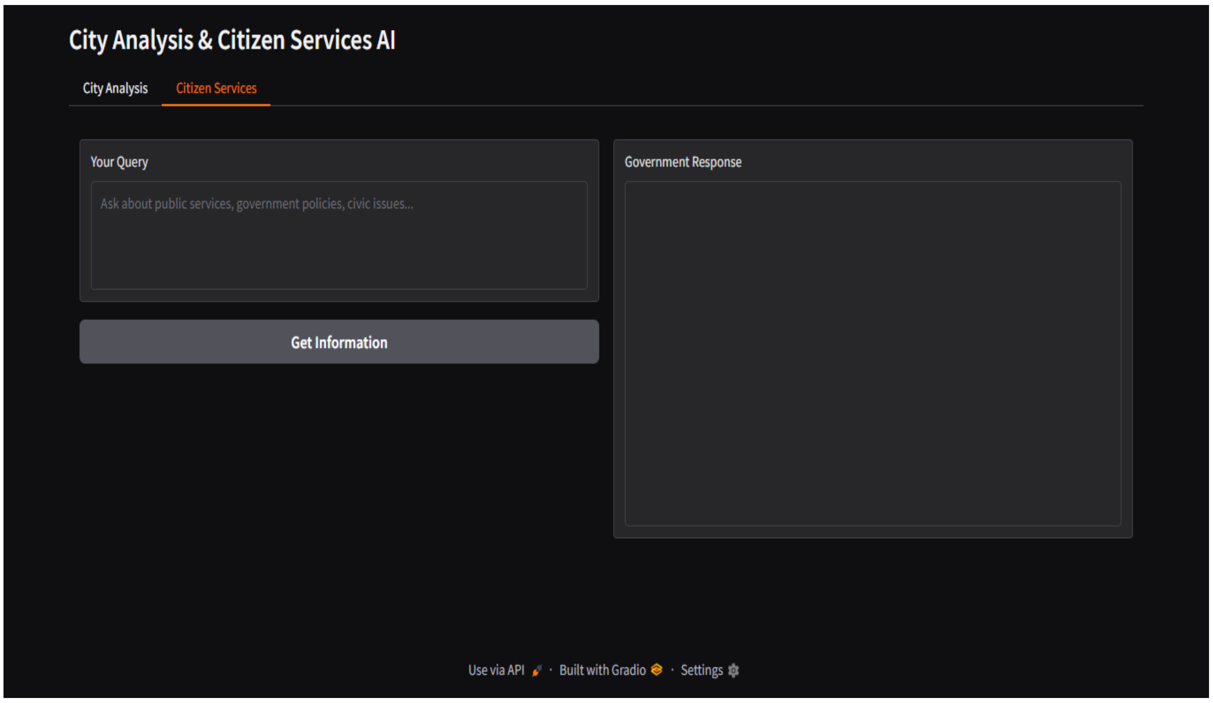
- **Edge Case Handling:** Malformed inputs, duplicate submissions, large document uploads, invalid API keys

Every function was validated to ensure reliability in both offline (local) and connected (API-driven) environments.

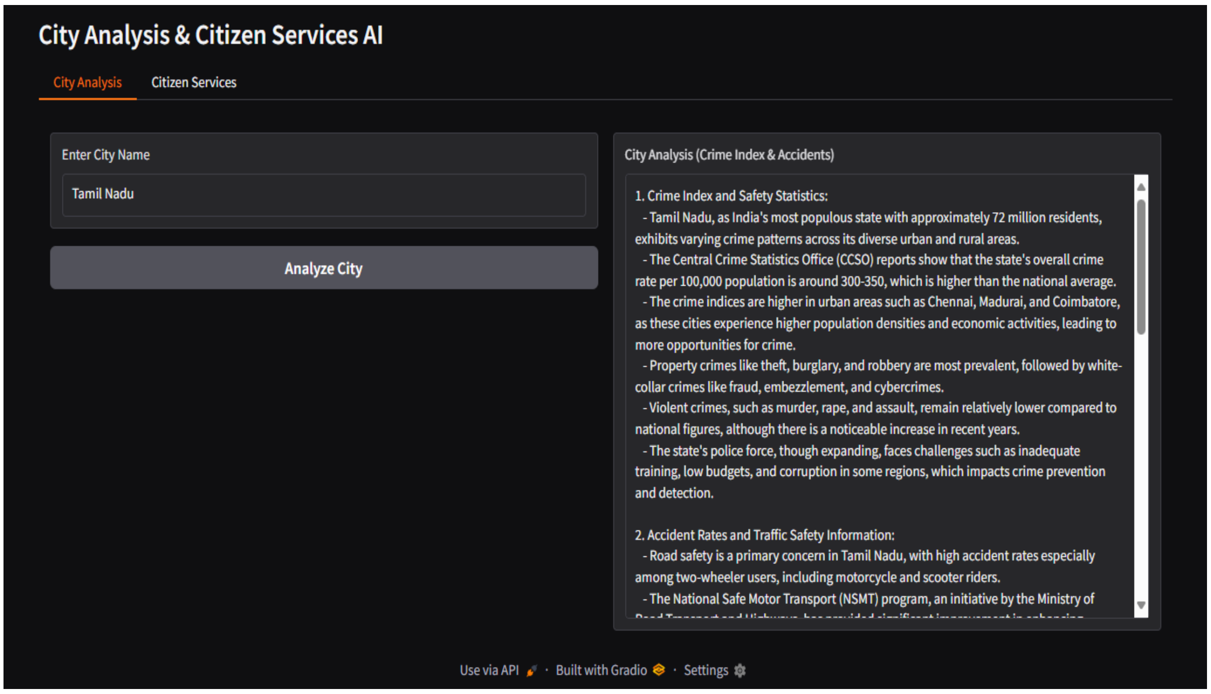
11. Screenshots:

1. Home page:


City Analysis & Citizen Services AI
City Analysis
Citizen Services
Enter City Name
e.g., New York, London, Mumbai...
Analyze City
City Analysis (Crime Index & Accidents)
Use via API · Built with Gradio · Settings


City Analysis & Citizen Services AI
City Analysis
Citizen Services
Your Query
Ask about public services, government policies, civic issues...
Get Information
Government Response
Use via API · Built with Gradio · Settings

2. City Analysis:



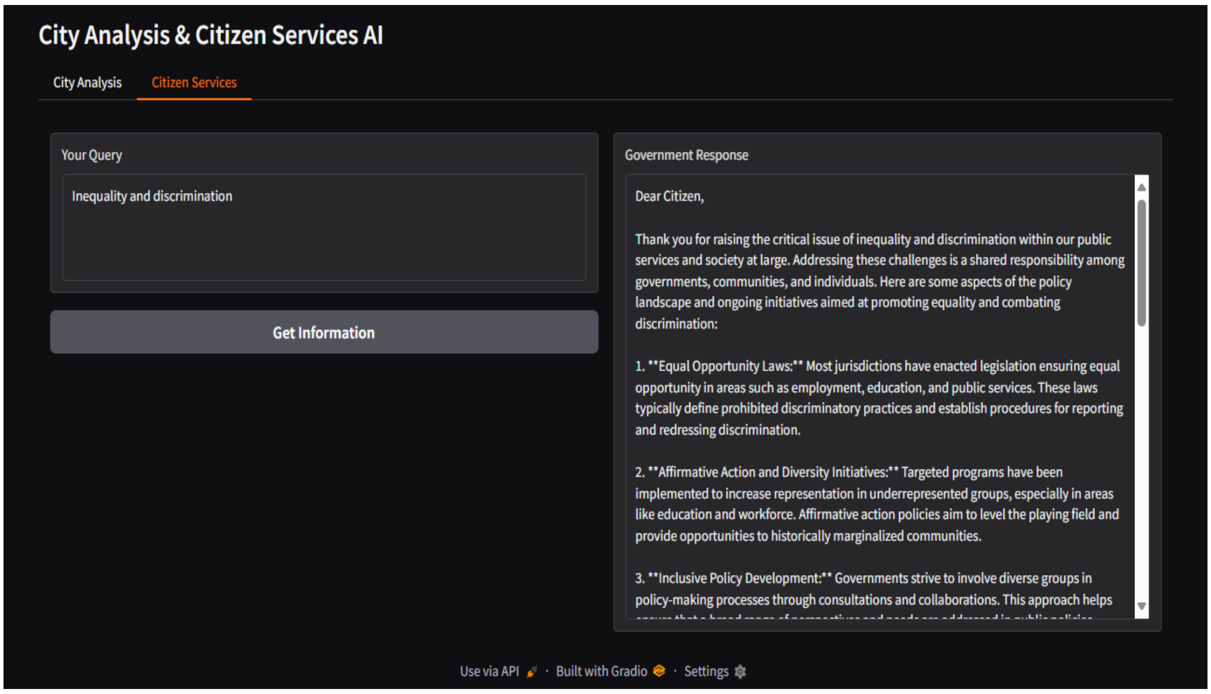
City Analysis & Citizen Services AI

Tamil Nadu

1. Crime Index and Safety Statistics:
 - Tamil Nadu, as India's most populous state with approximately 72 million residents, exhibits varying crime patterns across its diverse urban and rural areas.
 - The Central Crime Statistics Office (CCSO) reports show that the state's overall crime rate per 100,000 population is around 300-350, which is higher than the national average.
 - The crime indices are higher in urban areas such as Chennai, Madurai, and Coimbatore, as these cities experience higher population densities and economic activities, leading to more opportunities for crime.
 - Property crimes like theft, burglary, and robbery are most prevalent, followed by white-collar crimes like fraud, embezzlement, and cybercrimes.
 - Violent crimes, such as murder, rape, and assault, remain relatively lower compared to national figures, although there is a noticeable increase in recent years.
 - The state's police force, though expanding, faces challenges such as inadequate training, low budgets, and corruption in some regions, which impacts crime prevention and detection.

2. Accident Rates and Traffic Safety Information:
 - Road safety is a primary concern in Tamil Nadu, with high accident rates especially among two-wheeler users, including motorcycle and scooter riders.
 - The National Safe Motor Transport (NSMT) program, an initiative by the Ministry of

3. Citizen Services:



City Analysis & Citizen Services AI

Inequality and discrimination

Dear Citizen,

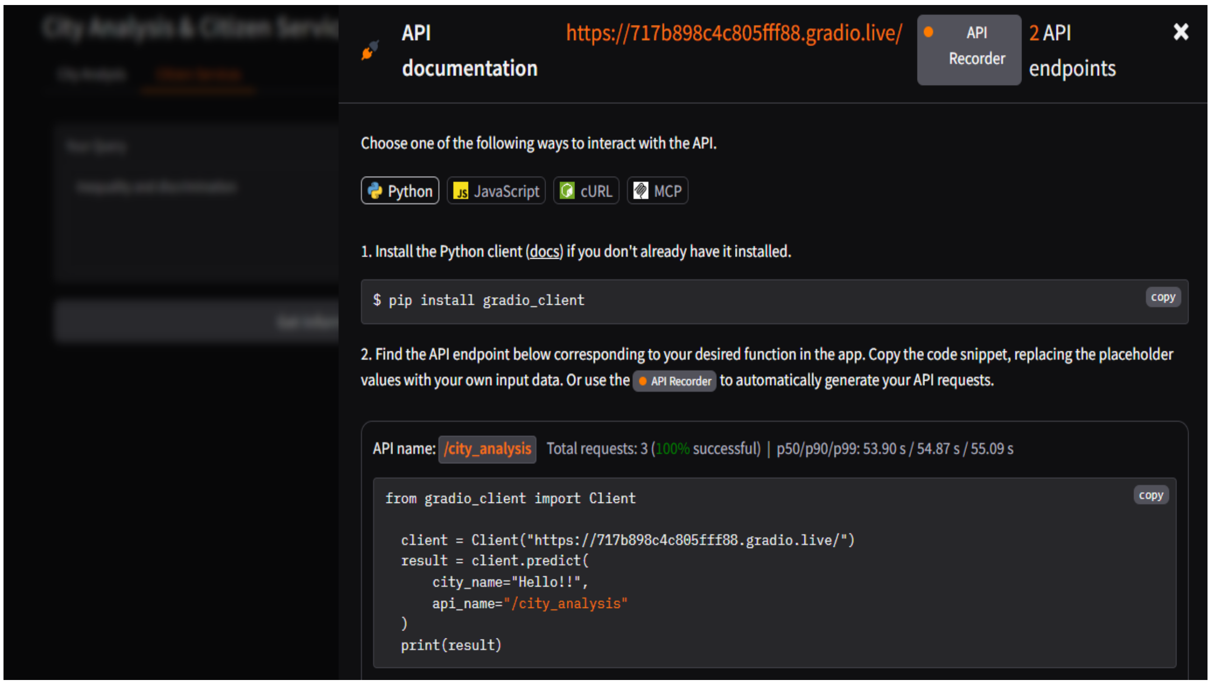
Thank you for raising the critical issue of inequality and discrimination within our public services and society at large. Addressing these challenges is a shared responsibility among governments, communities, and individuals. Here are some aspects of the policy landscape and ongoing initiatives aimed at promoting equality and combating discrimination:

1. **Equal Opportunity Laws:** Most jurisdictions have enacted legislation ensuring equal opportunity in areas such as employment, education, and public services. These laws typically define prohibited discriminatory practices and establish procedures for reporting and redressing discrimination.

2. **Affirmative Action and Diversity Initiatives:** Targeted programs have been implemented to increase representation in underrepresented groups, especially in areas like education and workforce. Affirmative action policies aim to level the playing field and provide opportunities to historically marginalized communities.

3. **Inclusive Policy Development:** Governments strive to involve diverse groups in policy-making processes through consultations and collaborations. This approach helps

4. API Documentation:



Choose one of the following ways to interact with the API.

1. Install the Python client (docs) if you don't already have it installed.

```
$ pip install gradio_client
```

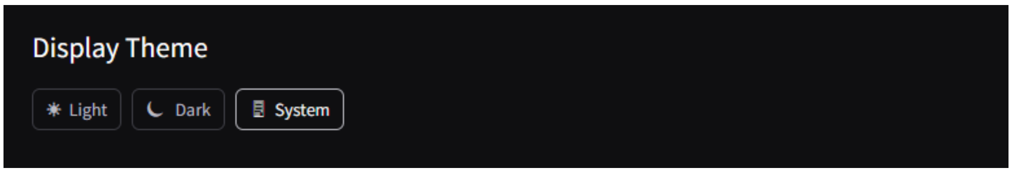
2. Find the API endpoint below corresponding to your desired function in the app. Copy the code snippet, replacing the placeholder values with your own input data. Or use the API Recorder to automatically generate your API requests.

API name: /city_analysis Total requests: 3 (100% successful) | p50/p90/p99: 53.90 s / 54.87 s / 55.09 s

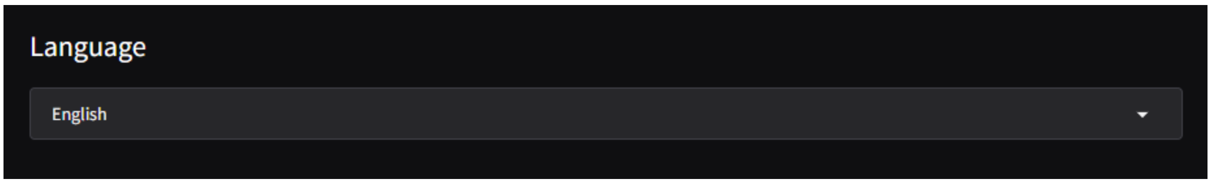
```
from gradio_client import Client

client = Client("https://717b898c4c805fff88.gradio.live/")
result = client.predict(
    city_name="Hello!!",
    api_name="/city_analysis"
)
print(result)
```

5. Settings:

- Display Theme:

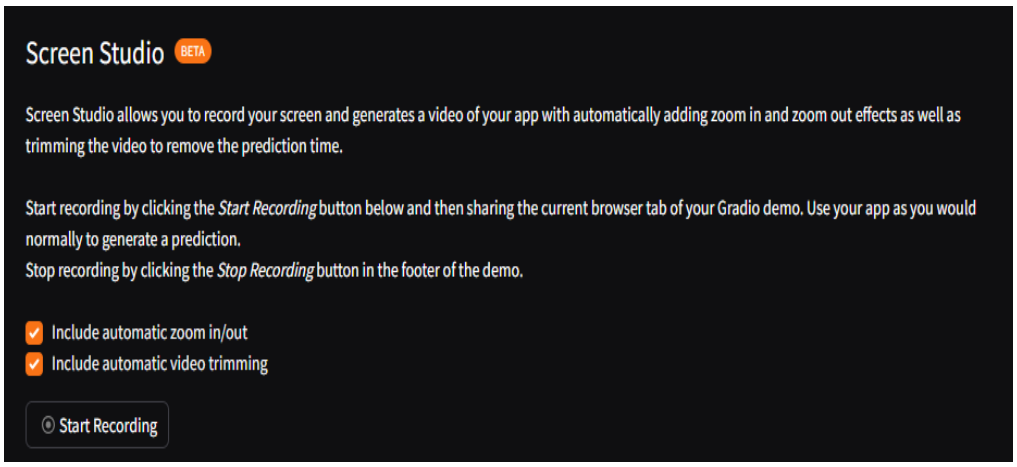


- Language:



Language: English

- Screen Studio



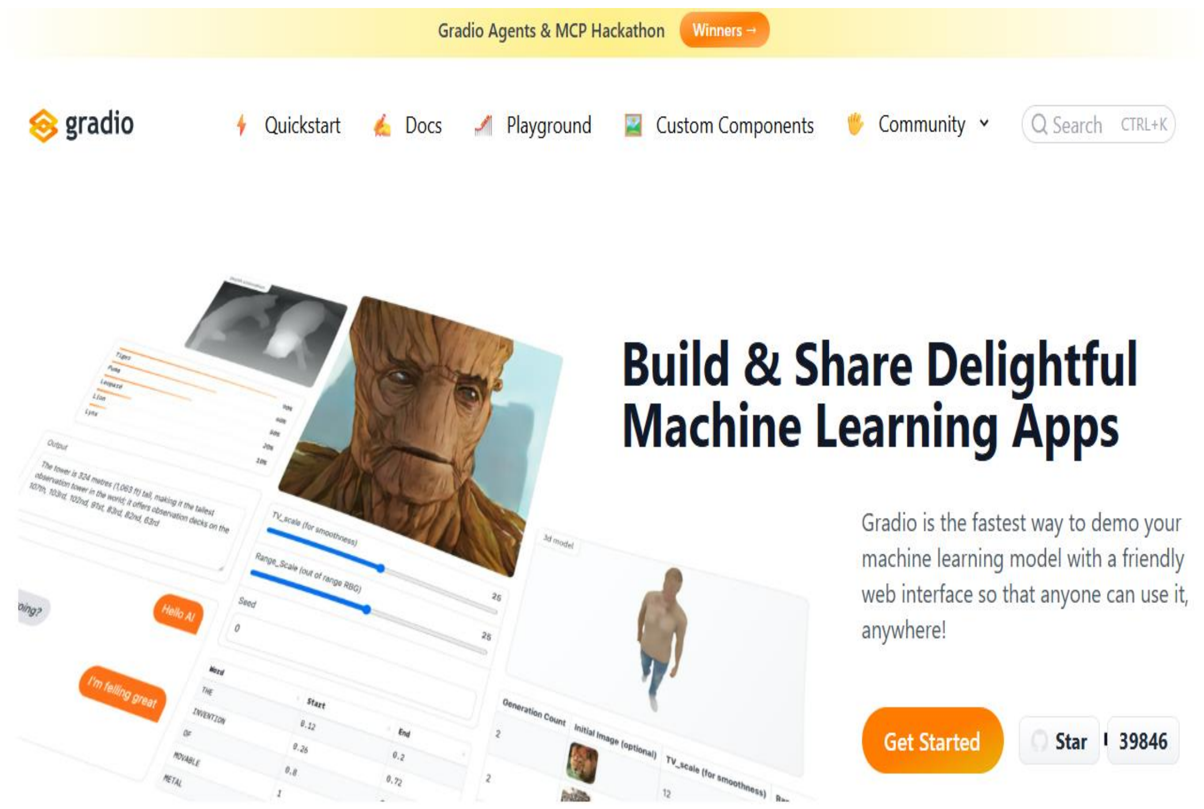
Screen Studio BETA

Screen Studio allows you to record your screen and generates a video of your app with automatically adding zoom in and zoom out effects as well as trimming the video to remove the prediction time.

Start recording by clicking the *Start Recording* button below and then sharing the current browser tab of your Gradio demo. Use your app as you would normally to generate a prediction.
Stop recording by clicking the *Stop Recording* button in the footer of the demo.

✓ Include automatic zoom in/out
✓ Include automatic video trimming

6. Gradio App Lauch:



Gradio Agents & MCP Hackathon Winners →

Build & Share Delightful Machine Learning Apps

Gradio is the fastest way to demo your machine learning model with a friendly web interface so that anyone can use it, anywhere!

12. Known Issues:

LLM May Generate Inaccurate or Fabricated Data:
The model (ibm-granite/granite-3.2-2b-instruct) is not connected to real-time databases or APIs. It may hallucinate statistics like crime rates or accident data, especially for lesser-known cities.
Impact: Users might be misled by incorrect or outdated safety assessments.

Lack of Input Validation:
No input sanitation or validation for city names or queries. Users can enter irrelevant or malformed data (e.g., "123", "!@#", or empty input), which might produce nonsensical outputs or trigger errors.

Fixed Prompt Format Limitations:
Prompts are static and may not handle diverse question types well.
E.g: queries like "How to renew my driving license in Delhi?" might return vague or incomplete information.

Performance on CPU:
If run on CPU (when GPU isn't available), model inference is slow and may timeout or lag.
Impact: Poor user experience on low-resource systems.

No Usage Limits or Rate Control:
The app allows unlimited queries with no throttling, which could lead to:
- Accidental overuse
- Resource exhaustion on free GPUs (e.g., Colab T4)

Impact: Could crash or slow down if multiple users access it via the share = True Gradio link.

No Persistent Session or Chat History:
Responses are not saved or logged, meaning users lose their queries/responses upon refresh.
Impact: Reduces usability for complex interactions or follow-up questions.

No Real Personalization or Context Memory:
The app doesn't remember user interactions or maintain session history. Follow-up questions don’t consider previous context unless explicitly included in the prompt.

Gradio UI Limitations:

Textbox outputs may not format long responses well (no markdown formatting or bullet points). Long responses can be hard to read due to lack of scrollbars or collapsible sections.

Hardcoded Model & Device Settings:

Uses fixed model name and logic:

device_map = "auto" if torch.cuda.is_available() else None

No option to switch models or precision dynamically.

No Error Handling or Debug Messages:

If the model fails to load or inference throws an error (e.g., due to OOM or tokenizer issues), the app may crash silently. No logging or UI-level error messages for user feedback.

Security Risks with Public Deployment:

If deployed using share=True, the Gradio link is public:

Anyone can misuse it (spam, test inappropriate queries).

No CAPTCHA or authentication to limit access.

13. Future enhancement:

Real-Time Data Integration:

Crime and Accident Data APIs: Connect to real-time government APIs or databases (e.g., Open Data Portals, Police or Traffic Authority APIs) for up-to-date statistics.

Multilingual Support:

Enable support for multiple regional languages to make the system accessible to non-English-speaking citizens. Use translation models like MBART, NLLB (No Language Left Behind), or Google Translate API for real-time translation.

Voice Assistant Integration:

Allow voice-based interaction using Gradio’s audio components or integrate with speech-to-text (Whisper) and text-to-speech (TTS) tools. Ideal for low-literacy or visually impaired users.

Geolocation Support:

Use GPS or user-inputted location to auto-detect the city and show localized results. Show nearby government services (e.g., police stations, hospitals) using OpenStreetMap or Google Maps API.

User Feedback & Interaction Loop:

Add a feedback box for users to rate the response or report inaccurate information. Use feedback for model fine-tuning or continuous improvement.

Mobile App Integration:

Build a cross-platform app (Flutter / React Native) for Android/iOS with the same Gradio backend or API service. Ensure offline availability of previously queried responses or safety guidelines.

Security and Authentication:

Add authentication for government employees or city admins to access administrative tools like uploading local data or sending alerts. Use OAuth, JWT, or Firebase Auth.

Emergency Alert System:

Trigger alerts based on user input (e.g., if someone reports a flood, crime, or accident). Integrate with SMS gateways or Push Notification APIs to notify citizens.

Data Visualization:

Visualize crime and safety stats using charts/maps (e.g., Plotly, Altair, or Leaflet for maps). Show trends over time or comparison between cities.

Chat History & Summary:

Keep a chat log and let users download or summarize the interaction for reference. Use summarization models to give concise responses or summaries of long reports.

OpenAI Function Calling / Retrieval-Augmented Generation (RAG):

Use RAG architecture to fetch data from a knowledge base or documents and let the model generate grounded answers.

Integrate Citizen Feedback Analytics:

Allow citizens to submit civic complaints (e.g., potholes, garbage collection delay) and track issue status. Combine with government CRM or ticketing systems (e.g., Municipal portals).