Sustainable Smart City Assistant Project Documentation

Team ID:NM2025TMID00827

1.Introduction

• Project Title: Sustainable Smart City Assistant Using IBM Granite LLM

• Team Members:

1. HARSHITHA P R

2. RAMJAN FARHATH S

3. SWATHILAKSHMI G

2. Project Overview

• Purpose:

The Sustainable Smart City Assistant is an AI-driven platform that equips cities and their residents with eco-friendly tools and data-driven insights. Utilizing IBM Watsonx Granite LLM and advanced data pipelines, it enhances the management of energy, water, and waste resources while offering easy-to-understand policy summaries, avenues for citizen feedback, environmental advice, KPI forecasting, and anomaly detection. This platform connects technology, governance, and community involvement to promote greener, more inclusive, and resilient urban spaces.

• Features:

1. Conversational Chat Assistant

Key Point: Natural Language Interaction

Functionality: This module serves as the central AI-driven conversational agent, allowing both citizens and city officials to engage with the system in plain, natural language. Instead of browsing through complex datasets or reports, users can simply ask questions such as "What is the city's current water usage?" or "How can I reduce household energy consumption?".

Benefits: Increases accessibility for non-technical users.

Provides instant, reliable answers without requiring manual data analysis.

Acts as a virtual sustainability advisor, encouraging wider community participation.

Use Case Example: A citizen asks, "How can I reduce my plastic waste at home?" and the assistant responds with personalized eco-tips.

2. Policy Summarization

Key Point: Simplified Comprehension

Functionality: Urban policies and environmental regulations are often lengthy and filled with technical jargon. This feature uses Natural Language Processing (NLP) to break down such documents into concise, easy-to-understand summaries that highlight key actions, deadlines, and responsibilities.

Benefits: Promotes policy transparency by making information accessible to all.

Saves time for both citizens and administrators who need quick policy overviews.

Increases civic engagement by ensuring everyone understands how policies affect them.

Use Case Example: A new waste segregation policy is released; the assistant summarizes it into a one-page, citizen-friendly guide.

3. Resource Forecasting

Key Point: Predictive Analytics

Functionality: Using historical consumption data, climate patterns, and urban growth projections, this feature forecasts future demand for essential resources such as water, energy, and waste management. Predictive models enable city planners to anticipate challenges and allocate resources efficiently.

Benefits: Prevents shortages and overuse of resources.

Enables data-driven planning for infrastructure upgrades.

Improves sustainability by balancing supply and demand.

Use Case Example: Predicting a summer water shortage based on historical drought data, allowing the city to implement conservation measures early.

4. Eco-Tip Generator

Key Point: Sustainable Lifestyle Guidance

Functionality: This personalized advisory system provides practical, everyday eco-friendly suggestions tailored to user preferences and input. Recommendations may include reducing energy use, recycling best practices, or eco-friendly commuting options.

Benefits: Encourages behavioral change toward sustainability.

Provides personalized, actionable steps instead of generic advice.

Strengthens the connection between individual choices and global sustainability goals.

Use Case Example: If a user inputs that they drive to work daily, the assistant may suggest carpooling, cycling, or using public transport for reducing emissions.

5. Citizen Feedback Reporting

Key Point: Instant Issue Reporting

Functionality: This feature allows residents to directly report urban issues—such as water leaks, waste collection delays, or power outages—via the platform. The reports are sent to relevant government departments for quick response and resolution.

Benefits: Creates a direct communication channel between citizens and city officials.

Promotes accountability and responsiveness in governance.

Helps build a real-time issue map for urban maintenance and monitoring.

Use Case Example: A resident notices an overflowing garbage bin and reports it instantly, triggering an alert to the municipal waste department.

6. KPI Forecasting & Anomaly Detection

Key Point: Strategic Insights and Early Alerts

Functionality: This advanced analytics module monitors Key Performance Indicators (KPIs) such as air quality levels, energy consumption, and recycling rates. Using machine learning, it can forecast trends and detect anomalies (unexpected spikes, drops, or irregularities) in real time.

Benefits: Helps administrators respond before small issues escalate into crises.

Provides data-backed decision support for long-term planning.

Builds trust and confidence through transparent reporting.

Use Case Example: Detecting an unexpected spike in air pollution levels in a specific district, prompting authorities to investigate possible industrial violations.

7. Multimodal Input Support

Key Point: Versatile Data Processing

Functionality: The platform is designed to handle multiple data input formats—including text, PDF, and CSV—ensuring that city data, citizen reports, and third-party datasets can all be processed efficiently.

Benefits: Enhances flexibility in data management.

Saves time by supporting direct uploads without needing format conversions.

Expands the scope of datasets usable for forecasting and policy analysis.

Use Case Example: A government official uploads a CSV of last year's energy usage, and the system instantly generates consumption trends and forecasts.

8. Streamlit Dashboard

Key Point: Intuitive User Interface

Functionality: The Streamlit-based dashboard provides an easy-to-use, interactive interface for visualizing city data, sustainability reports, and environmental insights. Users can explore charts, trend lines, and interactive widgets to understand the city's sustainability performance.

Benefits: Makes complex data visually clear and actionable.

Supports decision-making with evidence-based visualizations.

Offers transparency by giving both citizens and officials access to the same data.

Use Case Example: The city council uses the dashboard during a meeting to present a real-time water consumption forecast to stakeholders.

Use Case Scenarios

Policy Search & Summarization:

Scenario: A municipal planner uploads a lengthy and complex city policy document—such as a new waste management regulation or an urban zoning guideline—into the assistant.

Process:The assistant ingests the document (in PDF, Word, or text format).It applies natural language processing (NLP) techniques to identify the core objectives, deadlines, and citizen responsibilities.It then generates a clear, concise summary that highlights the most relevant points.

Outcome:The planner quickly gains a high-level overview of the policy without reading hundreds of pages.Citizens and stakeholders can also access citizen-friendly summaries, making governance more transparent and inclusive.

Benefit: Saves time, improves comprehension, and ensures wider civic engagement with policy decisions.

Citizen Feedback Reporting:

Scenario: A resident notices a burst pipe in their neighborhood, leading to water wastage. Instead of navigating bureaucratic channels, they open the assistant and report the issue directly.

Process: The resident submits a short description or even an image of the issue.The assistant categorizes the report (e.g., Water Infrastructure → Burst Pipe).It automatically tags the incident with relevant metadata such as location, severity, and urgency.The case is then routed to the appropriate municipal department for resolution.

Outcome: The issue is logged, categorized, and prioritized in real time.Authorities can monitor issue trends through the dashboard, enabling data-driven decision-making.

Benefit: Improves city responsiveness, fosters citizen trust, and ensures faster resolution of urban problems.

KPI Forecasting:

Scenario: A city administrator wants to plan for water supply management in the upcoming year. They upload last year's water consumption dataset (in CSV format) into the assistant.

Process: The assistant analyzes historical water usage patterns.It applies machine learning models to identify seasonal trends, peak usage times, and anomalies.The system then generates predictive forecasts for the coming months or years.Forecasts are displayed through interactive dashboards, with charts highlighting projected demand versus available resources.

Outcome: Administrators can anticipate resource shortages or demand spikes in advance.The city can implement preventive conservation measures (e.g., public awareness campaigns, rationing strategies, infrastructure upgrades).

Benefit: Ensures sustainable resource management, reduces waste, and supports long-term urban planning.

3. Architecture

Frontend (Streamlit):

Description: The assistant's user-facing layer is built using Streamlit, offering a modular, interactive dashboard.

Features Provided: Conversational Chat Module: Citizens and officials can interact with the assistant using natural language queries.

Feedback Submission: Residents can submit reports on city issues (e.g., infrastructure damage, water leaks).

KPI Visualization: Decision-makers access real-time dashboards showing energy usage, water consumption, and waste statistics.

Policy Search: Users can quickly locate and summarize specific policies using semantic search.

Eco-Tips Section: Displays personalized sustainability advice.

Anomaly Detection Alerts: Highlights unusual data trends such as sudden water surges or energy spikes.

Benefit: Provides a simple, intuitive, and visually rich interface for citizens, administrators, and policymakers.

Backend (FastAPI):

Description: The backend is built on FastAPI, a high-performance web framework for Python, which exposes a set of RESTful APIs to support frontend operations.

Core Responsibilities:

File Handling: Supports uploads of text, CSV, and PDF documents for analysis.

ML Forecasting & Anomaly Detection: Calls machine learning pipelines to generate predictions and detect irregularities.

LLM Integration: Connects to IBM Watsonx Granite for conversational AI, policy summarization, and eco-advice.

Data Routing: Ensures smooth communication between different system modules.

Benefit: Provides a scalable, secure, and modular API layer, enabling integration with third-party services or future city applications.

LLM Integration (IBM Watsonx Granite):

Description: The platform integrates IBM's Watsonx Granite LLM, which acts as the intelligence layer for text processing and human-like interaction.

Capabilities: Policy Summarization: Converts complex policy documents into short, clear summaries.

Eco-Tip Generation: Provides personalized sustainability recommendations based on user inputs.

Sustainability Reporting: Creates simplified environmental reports for citizens and administrators.

Conversational AI: Handles natural language queries for quick information retrieval.

Benefit: Bridges the gap between raw data and human understanding, ensuring accessibility and engagement.

Vector Search (Pinecone):

Description: To handle semantic policy search, the system uses Pinecone, a vector database optimized for embedding-based retrieval.

Functionality: Documents are converted into vector embeddings using NLP models.Pinecone stores and indexes these embeddings for efficient semantic search.

Enables queries like “Find all policies related to renewable energy incentives” and returns the most relevant results.

Benefit: Makes policy information discoverable and contextual, moving beyond keyword search to concept-based retrieval.

ML Modules:

Description: The assistant incorporates machine learning pipelines built with scikit-learn, pandas, and matplotlib.

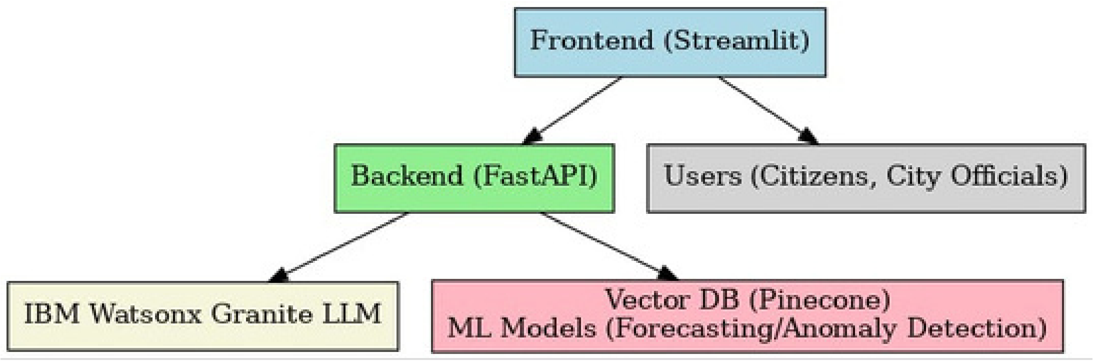
Core Functions:

Forecasting: Predicts future trends for water consumption, energy usage, and waste generation.

Anomaly Detection: Identifies unusual spikes or drops in datasets that may indicate leaks, inefficiencies, or misuse.

Visualization: Uses matplotlib to create graphs, trend lines, and KPI charts for decision-makers.

Benefit: Transforms historical and real-time data into actionable insights, supporting proactive city management.



4. Setup Instructions

Prerequisities

- Python 3.9 or higher
-
- FastAPI and Streamlit
-
- API keys for IBM Watsonx and Pinecone
-
- scikit-learn, pandas, and matplotlib
-
- Internet connectivity
-

Installation Process:

- **Clone the project repository**
-
- **Install required packages from requirements.txt**
-
- **Set up API credentials in the .env file**
-
- **Start the FastAPI backend server**
-
- **Launch the Streamlit frontend interface**
-
- **Upload documents or data to begin using the various modules**
-

5. Folder Structure

app/ – Contains the FastAPI backend logic

app/api/ – Houses routers for chat, feedback, eco tips, policies, and KPI endpoints

ui/ – Includes Streamlit frontend components

smart_dashboard.py – Main script to launch the dashboard

granite_llm.py – Functions for LLM services such as summaries, eco tips, and chat document_embedder.py – Handles conversion of documents into embeddings

kpi_file_forecaster.py – Performs forecasting of urban KPIs

anomaly_file_checker.py – Detects anomalies within datasets

report_generator.py – Generates sustainability reports

6. Running the Application

- Launch the FastAPI backend
-
- Start the Streamlit dashboard
-
- Use the sidebar for navigation
-
- Upload policy documents or KPI datasets
-
- Engage with the chat assistant and eco-friendly tools
-
- Access forecasts, anomaly detections, and sustainability reports
-

7. API Documentation

POST /chat/ask – Generates AI-powered responses

POST /upload-doc – Uploads and creates embeddings for documents

GET /search-docs – Performs semantic search on policies

GET /get-eco-tips – Retrieves sustainability tips

POST /submit-feedback – Records citizen feedback

8. Authentication

Token-Based Authentication (JWT or API Keys):

- Provides stateless, secure communication between client and server.
-
- Tokens carry user identity and permissions to avoid repeated logins.
-
- API keys allow controlled access for third-party integrations.
-

OAuth2 Integration with IBM Cloud:

- Enables enterprise-grade authentication with IBM IAM services.
-
- Supports Single Sign-On (SSO) across multiple IBM Cloud services.
-
- Enhances compliance with industry security standards.
-

Role-Based Access Control (RBAC):

- **Admins:** Manage dashboards, configure modules, monitor anomalies, and oversee citizen reports.
-
- **Citizens:** Limited to chat, eco-tips, policy summaries, and feedback submissions.
-
- **Researchers:** Access to datasets, KPIs, and forecasting outputs without admin privileges.
-

Session Management (Upcoming):

- Tracks active sessions across multiple devices.
-
- Auto-logout on inactivity or suspicious login attempts.
-
- History Tracking (Upcoming):
-
- Maintains audit logs of queries, feedback, and system access.
-
- Allows personalization (e.g., recommending eco-tips based on past activity).
-

9. User Interface

Sidebar Navigation with Themed Icons:

- Quick access to chat, dashboards, feedback forms, and reports.
-
- Consistent iconography for intuitive navigation.
-

KPI Visualizations with Summary Cards:

- Real-time data displayed in charts, graphs, and trend lines.
-
- Summary cards highlight critical insights (e.g., Water Usage: 15% above average).
-

Chat Assistant (AI-Powered):

- Provides conversational access to policies, eco-tips, and sustainability data.
-
- Supports natural language queries from both citizens and officials.
-

Feedback Forms with Categorized Issue Reporting:

- Citizens can log issues like waste management, water leaks, or energy outages.
-
- Auto-tagging by category ensures proper routing to authorities.
-

Display of Policy Summaries and Eco-Tips:

- Summaries generated by the LLM for better accessibility.
-
- Personalized eco-tips displayed based on user profile or activity.
-

Report Generation & Downloads:

- Users can create PDF/Excel reports from KPIs, feedback logs, or forecasts.
-
- Supports sharing with stakeholders in official meetings.
-

10. Testing

Unit Tests (Backend & ML Modules):

- Validates functionality of APIs, forecasting models, and anomaly detectors.
-
- Ensures code reliability and maintainability.
-

API Testing (Swagger UI & Postman):

- Interactive documentation with Swagger UI for developers.
-
- Postman scripts to test endpoints under different scenarios.
-

Manual Testing:

- Chat assistant tested for accuracy and contextual understanding.
-
- Policy search checked for relevance and summarization quality.
-
- Forecasting tested with different datasets (short-term vs. long-term).
-

Edge Case Handling:

- Large file uploads tested for performance and stability.
-
- Malformed inputs checked for graceful error handling.
-
- Invalid API keys tested to ensure proper authentication failure responses.
-

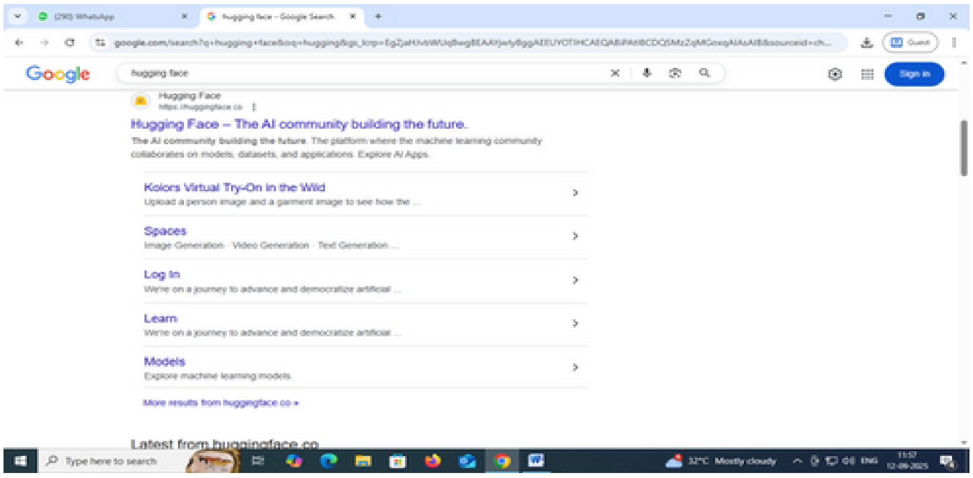
Integration Testing:

- Ensures smooth communication between frontend, backend, LLM, Pinecone, and ML modules.
-
- Load & Stress Testing:
-
- Simulates multiple concurrent users accessing the system.
-
- Ensures scalability under heavy workloads.
-

User Acceptance Testing (UAT):

- Collects feedback from city officials and citizens on usability.
-
- Validates whether system meets real-world expectations.
-

11. Screenshots


(290) WhatsApp
hugging face - Google Search
google.com/search?q=hugging+face&oq=hugging&gs_lcrp=EgZjaHJvbWUqBwgBEAAYjwIyBggAEEUYOTIHCAEQABiPAtIBCDQ5MzZqMGoxqAIAsAIB&sourceid=ch...
Guest
hugging face
Sign in

Hugging Face
https://huggingface.co

Hugging Face – The AI community building the future.

The AI community building the future. The platform where the machine learning community collaborates on models, datasets, and applications. Explore AI Apps.

Kolors Virtual Try-On in the Wild
Upload a person image and a garment image to see how the ...

Spaces
Image Generation · Video Generation · Text Generation ...

Log In
We're on a journey to advance and democratize artificial ...

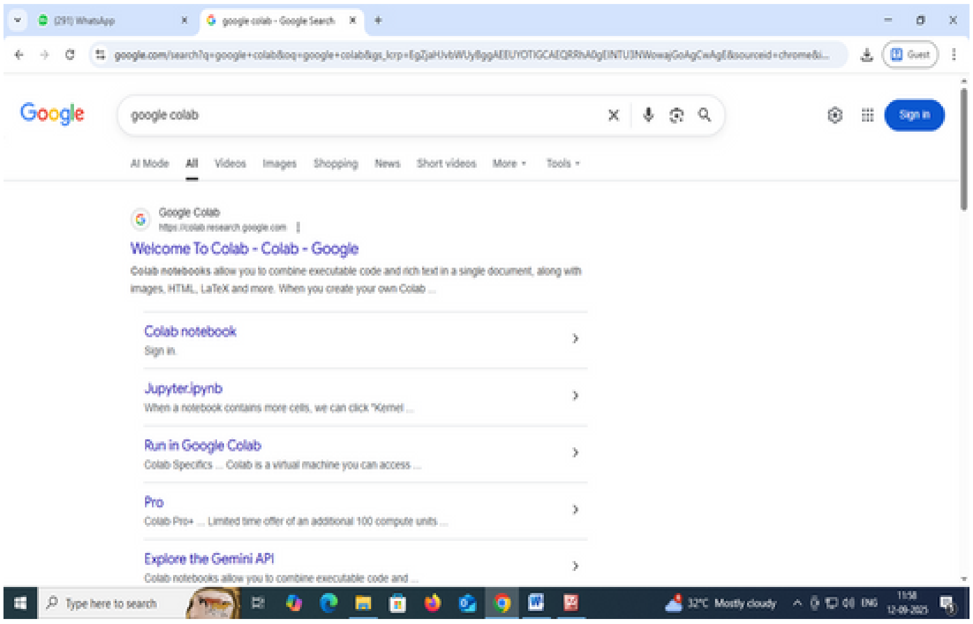
Learn
We're on a journey to advance and democratize artificial ...

Models
Explore machine learning models.

More results from huggingface.co »

Latest from huggingface.co

Type here to search
32°C Mostly cloudy
ENG
11:57
12-09-2025


(291) WhatsApp
google colab - Google Search
google.com/search?q=google+colab&oq=google+colab&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIGCAEQRRhA0gEINTU3NWowajGoAgCwAgE&sourceid=chrome&i...
Guest
google colab
Sign in
AI Mode
All
Videos
Images
Shopping
News
Short videos
More
Tools

Google Colab
https://colab.research.google.com

Welcome To Colab - Colab - Google

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab ...

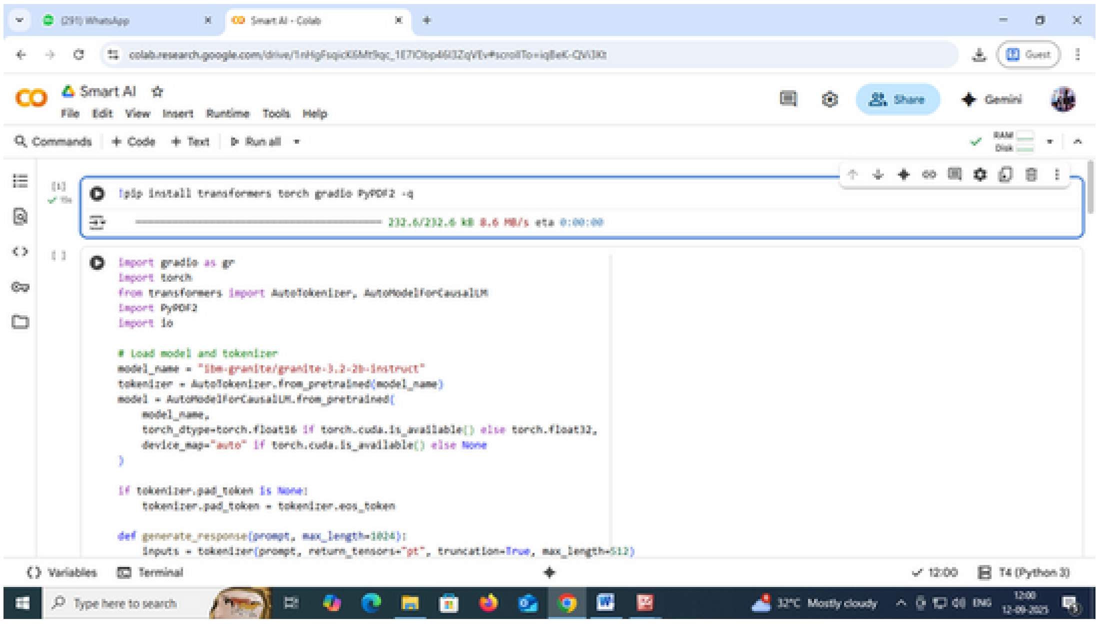
Colab notebook
Sign in.

Jupyter.ipynb
When a notebook contains more cells, we can click "Kernel ...

Run in Google Colab
Colab Specifics ... Colab is a virtual machine you can access ...

Pro
Colab Pro+ ... Limited time offer of an additional 100 compute units ...

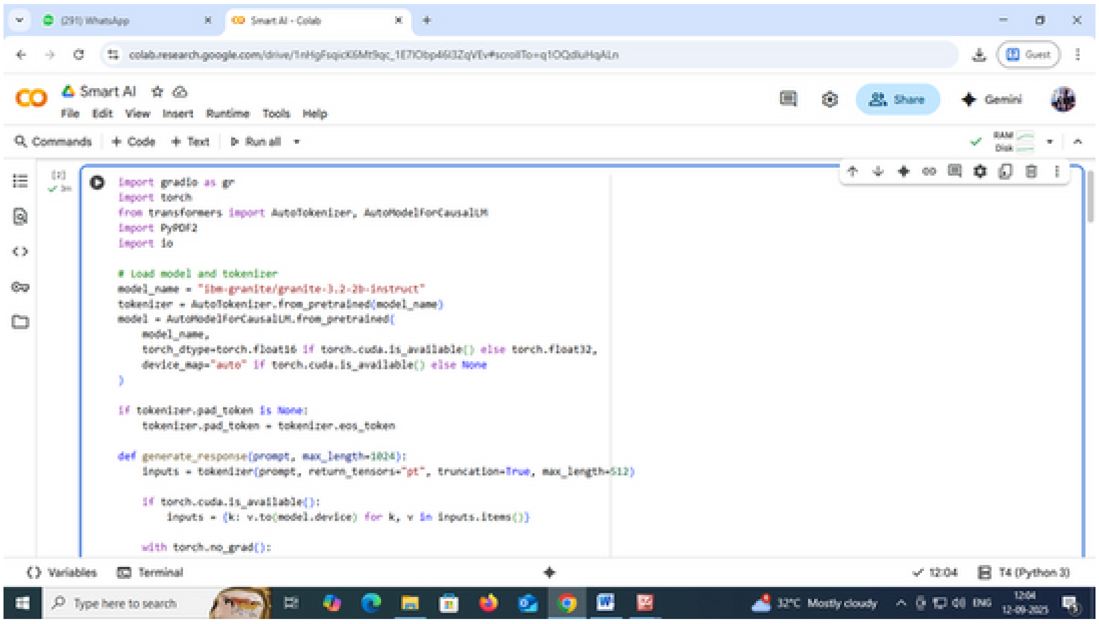
Explore the Gemini API
Colab notebooks allow you to combine executable code and ...



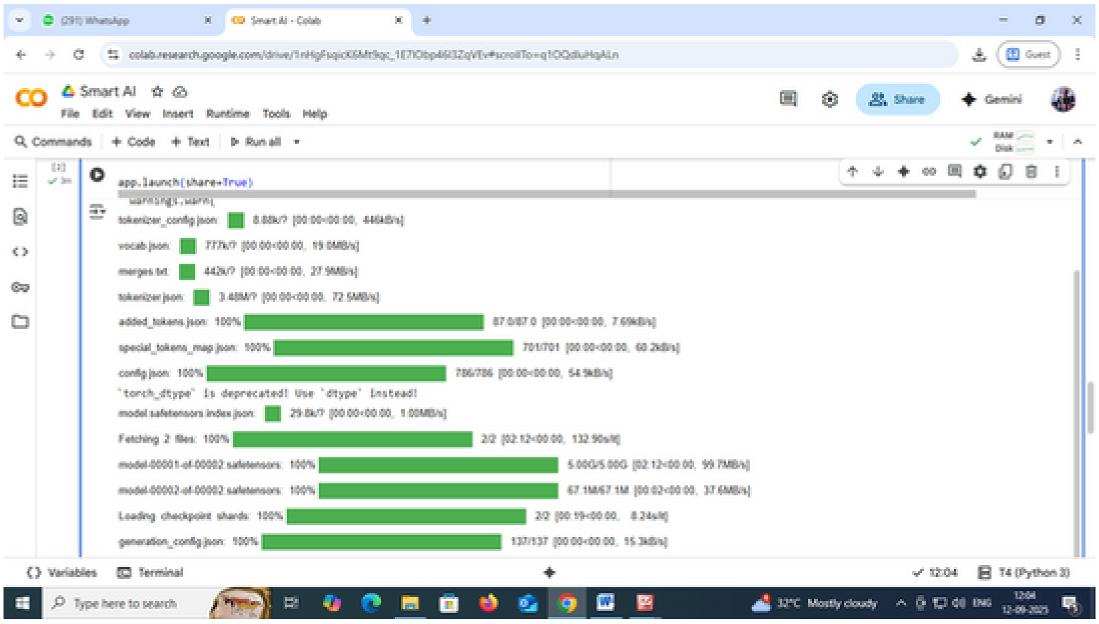
```
!pip install transformers torch gradio PyPDF2 -q
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 8.6 MB/s eta 0:00:00
```

```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
import io

# 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)
```



```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
import io

# 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():
```



```
app.launch(share=True)
  warnings.warn(
tokenizer_config.json: 8.88k/? [00:00<00:00, 446kB/s]
vocab.json: 777k/? [00:00<00:00, 19.0MB/s]
merges.txt: 442k/? [00:00<00:00, 27.9MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 72.5MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 7.69kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 60.2kB/s]
config.json: 100% 786/786 [00:00<00:00, 54.9kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 1.00MB/s]
Fetching 2 files: 100% 2/2 [02:12<00:00, 132.90s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [02:12<00:00, 99.7MB/s]
model-00002-of-00002.safetensors: 100% 67.1M/67.1M [00:02<00:00, 37.6MB/s]
Loading checkpoint shards: 100% 2/2 [00:19<00:00, 8.24s/it]
generation_config.json: 100% 137/137 [00:00<00:00, 15.3kB/s]
```

12. Known Issues

1. Support for Limited Languages

Currently, the system primarily supports English (and a few widely used languages).

This can restrict accessibility for citizens in multilingual regions where local dialects or native languages are commonly spoken.

Impact: Limits inclusivity for non-English speakers.

Reduces adoption in culturally diverse urban populations.

Planned Resolution: Expansion of multilingual NLP models.Incorporation of translation APIs to support regional languages.

2. Requires Reliable Cloud Connectivity

The assistant is cloud-hosted and depends heavily on a stable internet connection.Areas with poor connectivity or limited bandwidth may face issues such as delayed responses or failure in real-time forecasting.

Impact: Reduced usability in rural or underdeveloped urban zones.Inaccessibility during network outages or emergencies.

Planned Resolution: Development of offline-first features with local caching.Lightweight mobile-friendly version with minimal data dependency.

3. Subject to API Quota Restrictions

The system integrates third-party services (e.g., LLMs, vector databases, cloud APIs) that impose usage limits.

Excessive requests or peak-time activity may trigger rate limiting or temporary service restrictions.

Impact: Delays in processing citizen feedback or policy searches.Inconsistent availability for high-volume users (e.g., city administrators during crises).

Planned Resolution: Implement request optimization and caching mechanisms.Explore higher-tier subscription plans or on-premise deployments to reduce dependency on quotas.

13. Future Enhancement

1.Support for multiple languages:

This involves designing the system to handle a wide range of languages, enabling users from different linguistic backgrounds to interact with it seamlessly. It includes localization of the interface, translation of content, and support for regional formats such as date, time, and numeric values. Multi-language support improves accessibility, user engagement, and usability across diverse populations.

2.Integration with IoT devices and city sensors:

This feature enables the platform to connect and communicate with a variety of Internet of Things (IoT) devices and sensors deployed throughout a city. These may include environmental sensors (air quality, temperature, noise), traffic monitors, energy meters, and smart lighting systems. Integrating these devices allows real-time data collection, enhances situational awareness, and supports informed decision-making in urban management and sustainability efforts.

3.Advanced anomaly detection with deep learning:

Using deep learning algorithms, the system can identify unusual patterns or deviations in data that may indicate faults, inefficiencies, or emerging problems. This automated anomaly detection goes beyond traditional rule-based systems by learning from historical data to detect subtle or complex anomalies. It helps proactively address issues such as equipment failures, security threats, or environmental hazards.

4.Mobile-optimized dashboard interface:

This ensures that the dashboard, which displays key data, analytics, and controls, is fully responsive and optimized for mobile devices like smartphones and tablets. A mobile-optimized interface enhances usability and accessibility for users on the go, enabling them to monitor and manage systems anytime, anywhere with intuitive navigation and clear visualization.

5.Official/doctoral verification of eco policies:

This involves the formal validation and certification of environmental policies and practices by recognized authorities, such as government agencies or academic institutions. Doctoral verification implies rigorous research-based assessment of the policies' effectiveness and compliance with scientific standards. This verification increases credibility, transparency, and accountability in implementing and reporting on eco-friendly initiatives.