**Project Title**

**Project Documentation**

# 1.Introduction

* Project title : Sustainable Smart City Assistant using IBM
* Team member : M. Kiruthiga
* Team member : S. Kiruthiga
* Team member : T. Durga
* Team member : M.K. Baavana

# 2.project overview

* Purpose :

The purpose of the Sustainable Smart City Assistant is to create an AI-powered platform that empowers cities and their citizens to build a greener, more efficient, and connected urban ecosystem. The system leverages artificial intelligence, real-time data, and machine learning to optimize essential resources such as energy, water, and waste, while also promoting sustainable lifestyles among citizens.

For government officials, the assistant provides decision-making support by simplifying complex policy documents, generating forecasts, and highlighting anomalies in resource usage. For citizens, it offers personalized eco-friendly tips and encourages participation through a feedback loop, bridging the gap between governance, technology, and community engagement.

* Objectives
* Encourage sustainable living practices among citizens.
* Provide data-driven insights for city officials.
* Forecast resource usage (energy, water, waste) to enable better planning.
* Detect anomalies in urban infrastructure usage to prevent crises.
* Simplify complex government policies into citizen-friendly summaries.
* Support strategic planning through KPI forecasting and analytics.
* Promote citizen participation in governance through feedback collection.

* Features:
* Conversational Interface – Natural language interaction between users and the system.
* Policy Summarization – Converts lengthy policy documents into clear, actionable summaries.
* Resource Forecasting – Predicts energy, water, and waste consumption using AI models.
* Eco-Tip Generator – Provides personalized sustainability advice for citizens.
* Citizen Feedback Loop – Collects and analyzes input from citizens to improve services.
* KPI Forecasting – Helps officials track progress on sustainability goals.
* Anomaly Detection – Identifies unusual patterns in data to provide early warnings.
* Multimodal Input Support – Accepts text, PDFs, and CSV files for analysis.
* User-Friendly Dashboard – Interactive interface built with Gradio/Streamlit for ease of use.

# 3. Architecture

The Sustainable Smart City Assistant follows a simple modular architecture with three main layers:

1. Frontend (User Interface)

Built using Streamlit or Gradio.

Provides a simple dashboard where users can:

Generate eco-friendly tips.

Upload and summarize policies.

View resource forecasts and reports.

Give feedback to city officials.

1. Backend (Application Layer)

Powered by FastAPI.

Connects the user interface with AI models and machine learning modules.

Handles requests like summarization, eco-tips, forecasting, and feedback collection.

1. AI & ML Modules

LLM Integration (IBM Watsonx Granite): For policy summarization and eco-tips.

Forecasting Models: Predicts energy, water, and waste usage.

Anomaly Detection: Finds unusual patterns in data.

1. Database & Storage

Stores citizen feedback, uploaded documents, and historical data.

Can use MongoDB or PostgreSQL for managing data.

**4. Setup Instructions**

**Prerequisites:**

Python 3.9 or above installed on your computer

Internet connection

Pip (Python package manager)

**Installation Process:**

1. Download or copy the project files to your system.
2. Open Command Prompt or Terminal.
3. Install the required libraries:

Pip install transformers torch gradio PyPDF2 pandas matplotlib scikit-learn fastapi uvicorn

4. Project is now ready to run.

## 5. Folder Structure

app/ → Backend logic (APIs, models, utilities)

api/ → API routes (chat, feedback, summarization)

models/ → Machine learning models (forecasting, anomaly detection)

utils/ → Helper functions

ui/ → Frontend files (Streamlit/Gradio interface)

eco\_tips.py → Eco tips generator

policy\_summary.py → Policy summarization

dashboard.py → Main dashboard

data/ → Sample datasets and uploaded files

requirements.txt → List of dependencies

smart\_dashboard.py → Entry script to run the dashboard

granite\_llm.py → IBM Granite LLM integration

document\_embedder.py → Document/PDF processing

kpi\_file\_forecaster.py → Forecasting module

anomaly\_file\_checker.py → Anomaly detection module

report\_generator.py → AI-generated sustainability reports

README.md → Project overview and usage guide

## 6. Running the Application

To start the project:

* Launch the FastAPI server to expose backend endpoints.
* Run the Streamlit dashboard to access the web interface.
* Navigate through the pages using the sidebar.
* Upload documents (PDF/CSV) or enter text to analyze policies.
* Generate eco-friendly tips using the Eco Tips module.
* View resource forecasts, anomalies, and KPIs.
* Submit citizen feedback through the feedback form.
* Download AI-generated sustainability reports.

All interactions are real-time and use backend APIs to dynamically update the frontend.

## 7. API Documentation

The backend provides several REST API endpoints to support the system.

**POST /chat/ask**

Accepts a user query and responds with an AI-generated answer.

**POST /upload-doc**

Uploads PDF or text documents and processes them for summarization or embedding.

**GET /search-docs**

Returns semantically similar policy documents based on a user’s query.

**GET /get-eco-tips**

Provides sustainability tips for selected topics (e.g., energy, water, waste).

**POST /submit-feedback**

Stores citizen feedback in the database for review and analysis.

Each endpoint is tested and documented in Swagger UI for quick inspection and trial during development.

**Authentication**

This project currently runs in an open environment for demonstration.

However, secure deployments can include:

Token-based authentication (JWT or API keys)

OAuth2 with IBM Cloud credentials

Role-based access (Admin, Citizen, Researcher)

User session management (planned enhancement)

These methods ensure that only authorized users can access sensitive features like policy data, forecasts, and reports.

## 9. User Interface

The interface is designed to be simple, clear, and user-friendly, so both citizens and officials can use it easily.

It includeincludees:

Sidebar Navigation → To move between different modules.

Eco-Tips Tab → Generates daily sustainable living tips.

Policy Summarization Tab → Upload or paste policy text and get key points.

Forecasting Dashboard → Shows energy, water, and waste predictions with charts.

Citizen Feedback Form → Collects user opinions and suggestions.

Report Download Option → Allows downloading AI-generated sustainability reports.

The design focuses on clarity, speed, and accessibility for non-technical users.

## 10. Testing

Testing was carried out in different phases to ensure reliability and performance:

Unit Testing → For core functions like policy summarization, eco-tip generation, and forecasting.

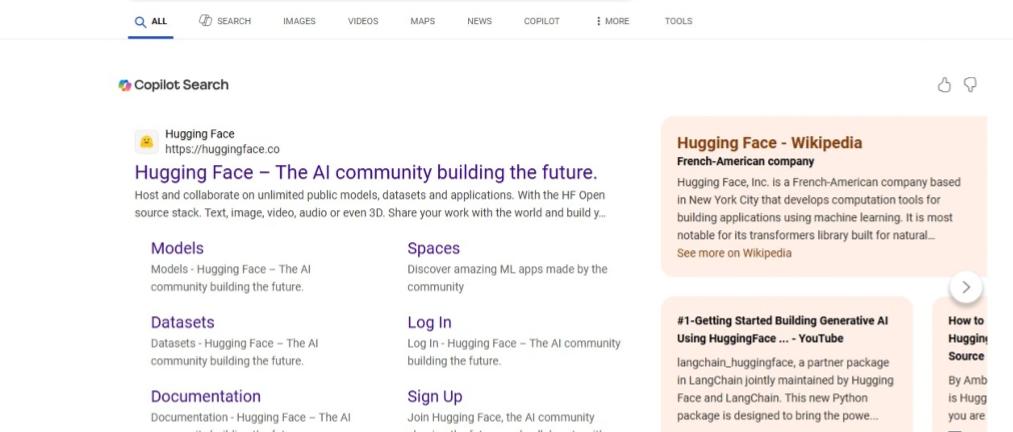
API Testing → Using Swagger UI and Postman to check all backend endpoints.

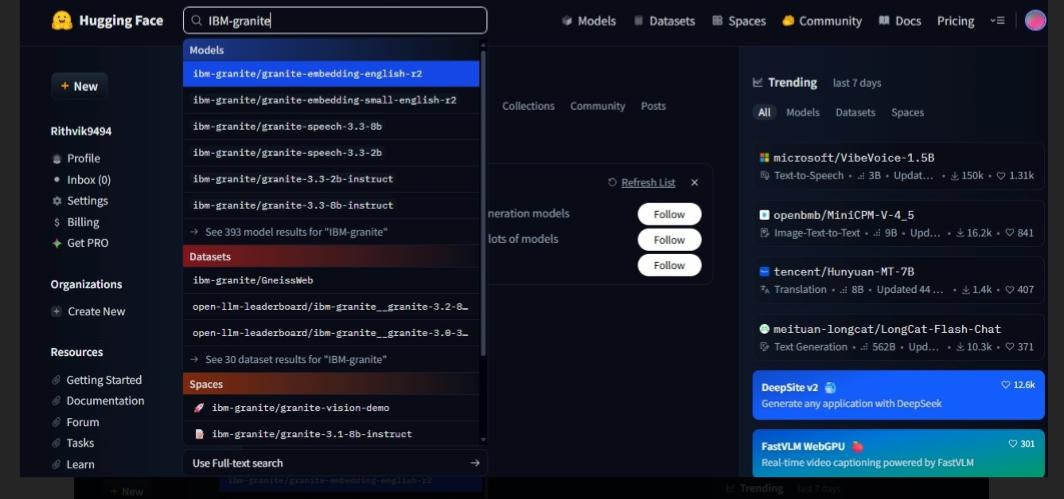
Manual Testing → For file uploads, chat responses, and output accuracy.

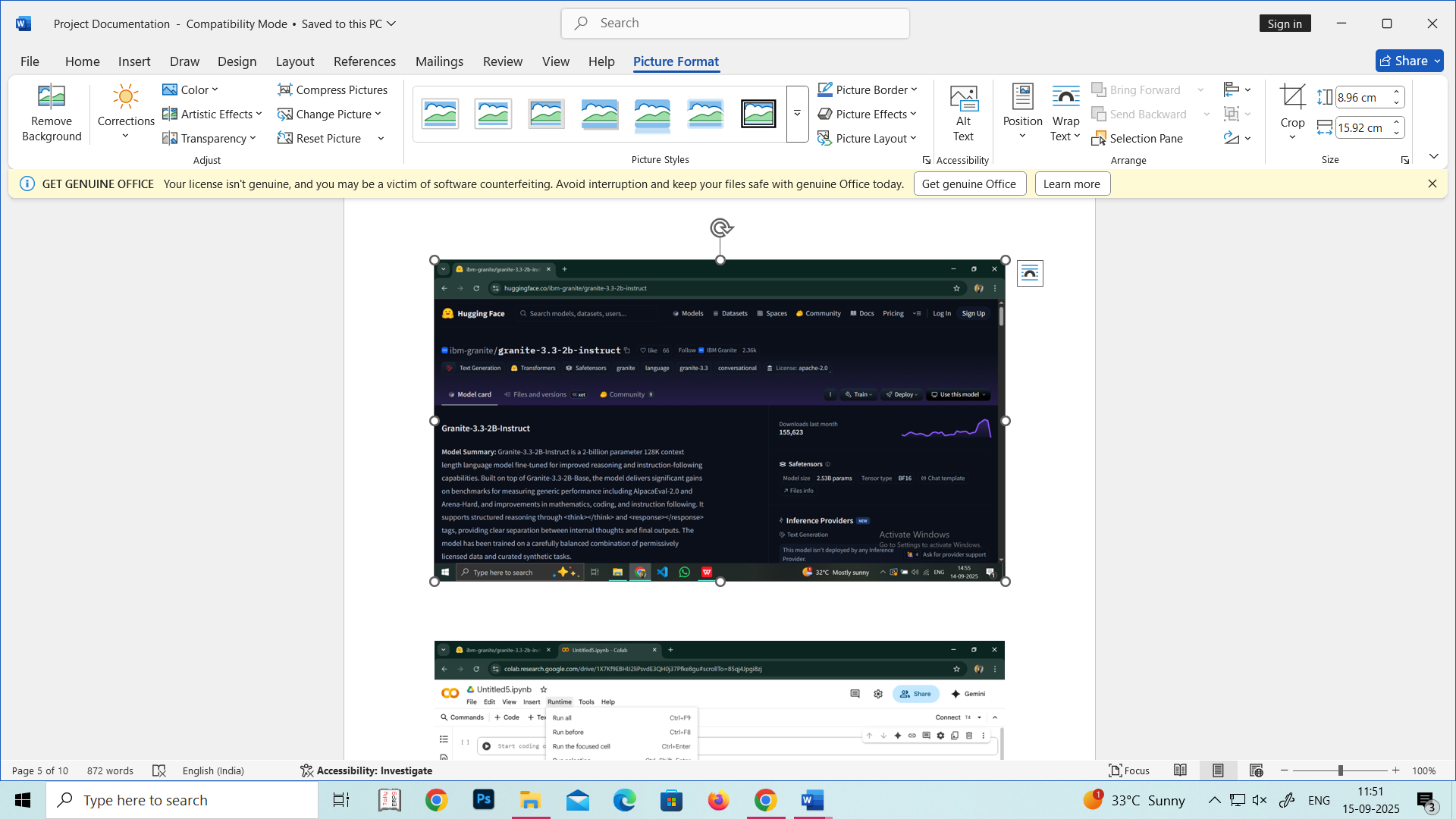
Edge Case Handling → Tested with large files, empty inputs, and invalid data.

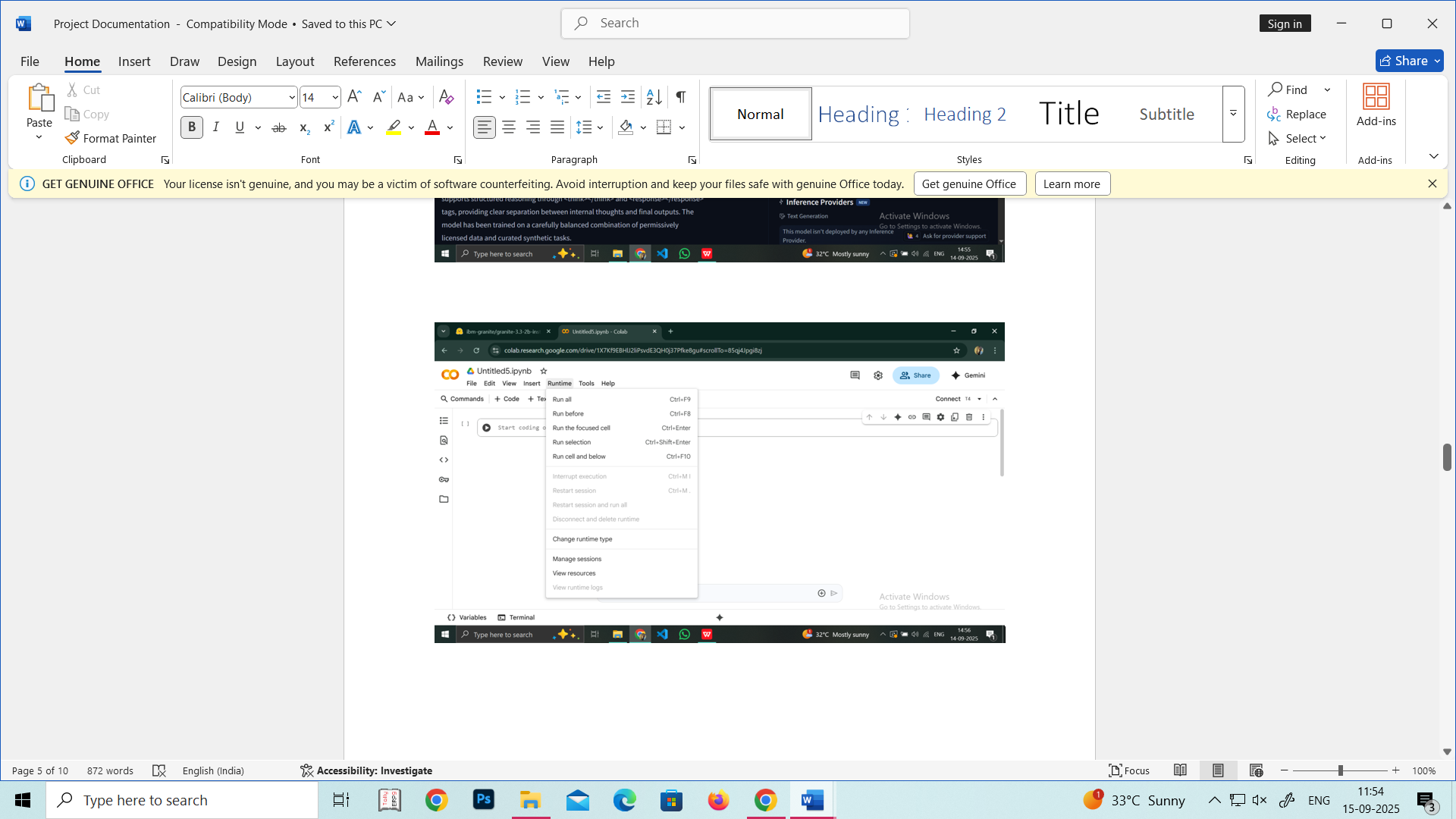
Each module was validated to make sure the system works correctly in both offline and online (API-connected) modes.

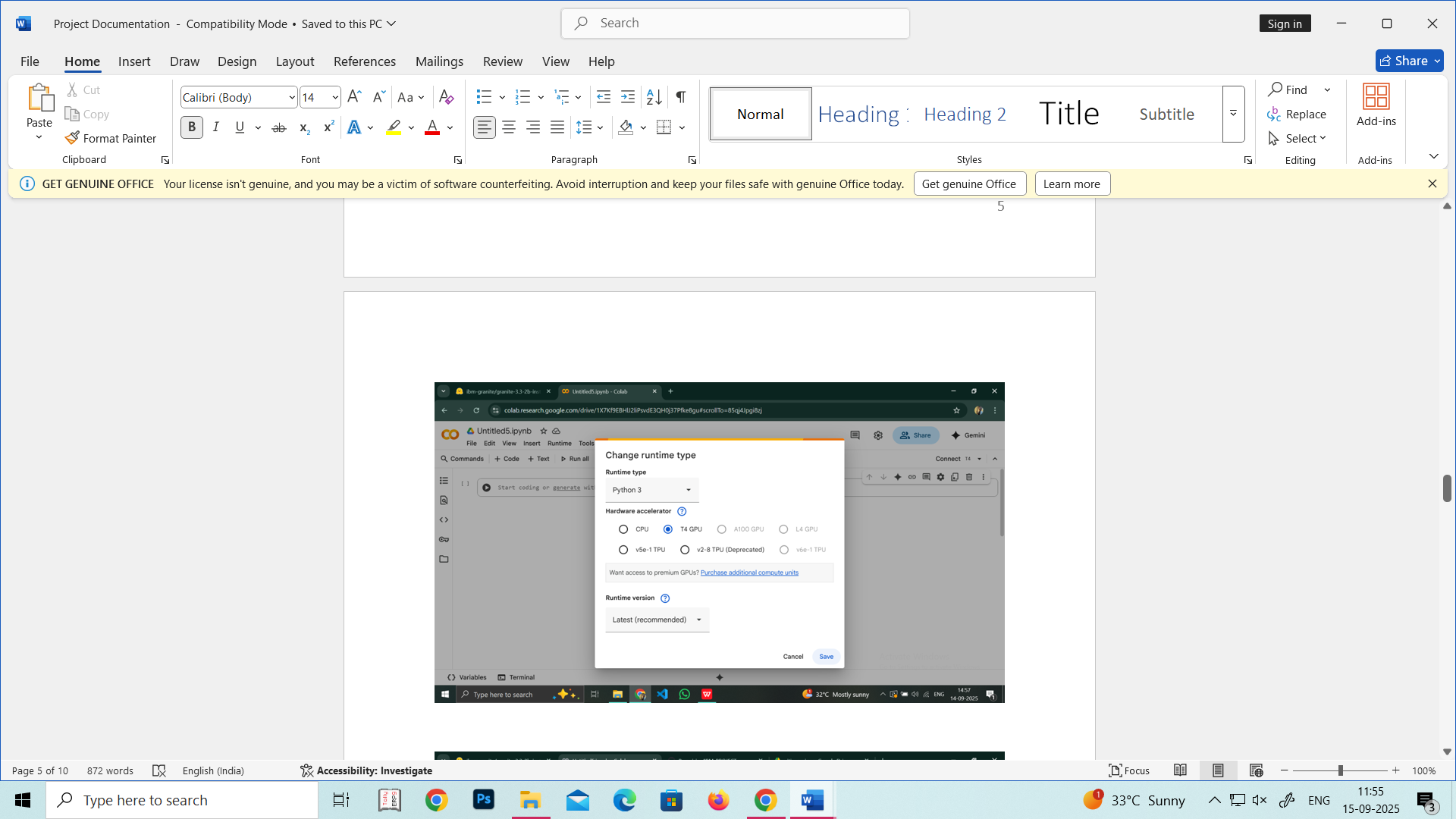
**11.screen shots**

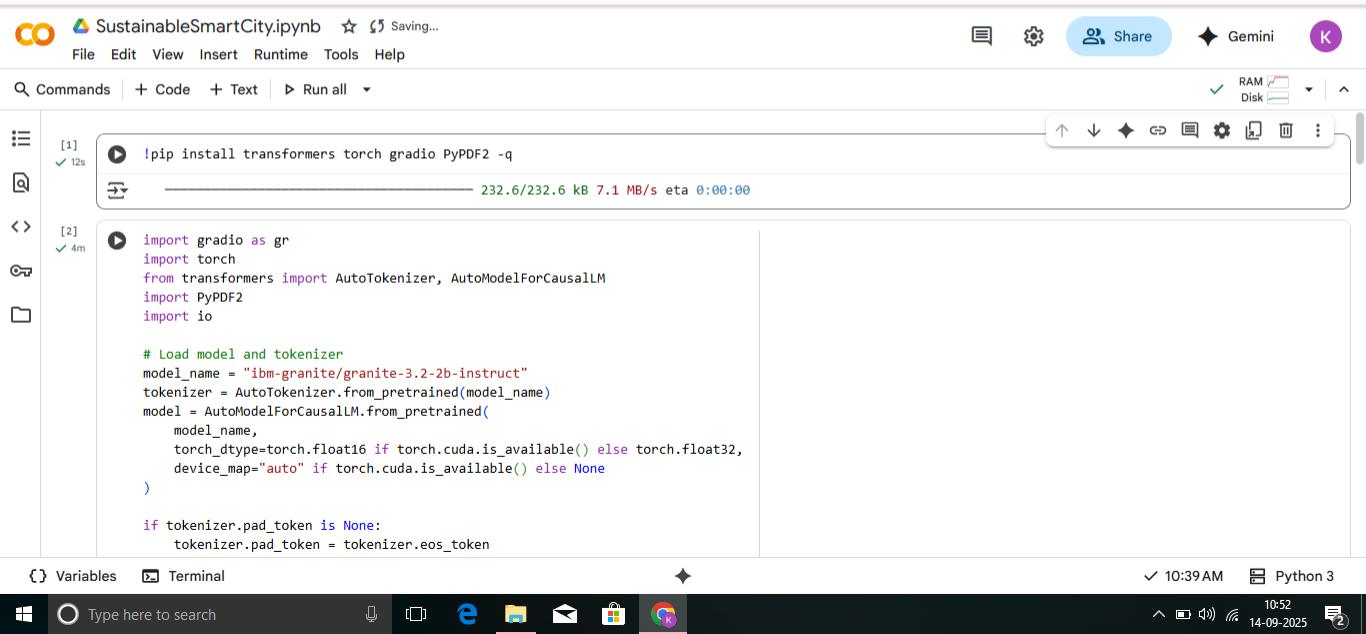
****

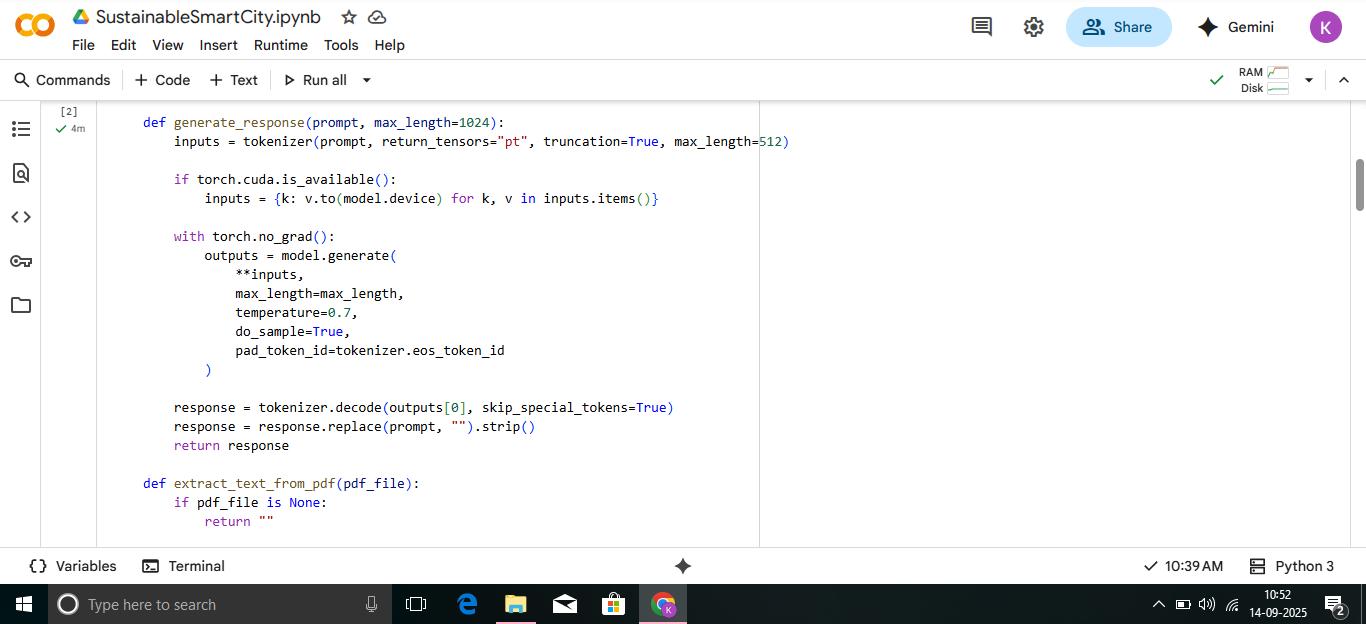
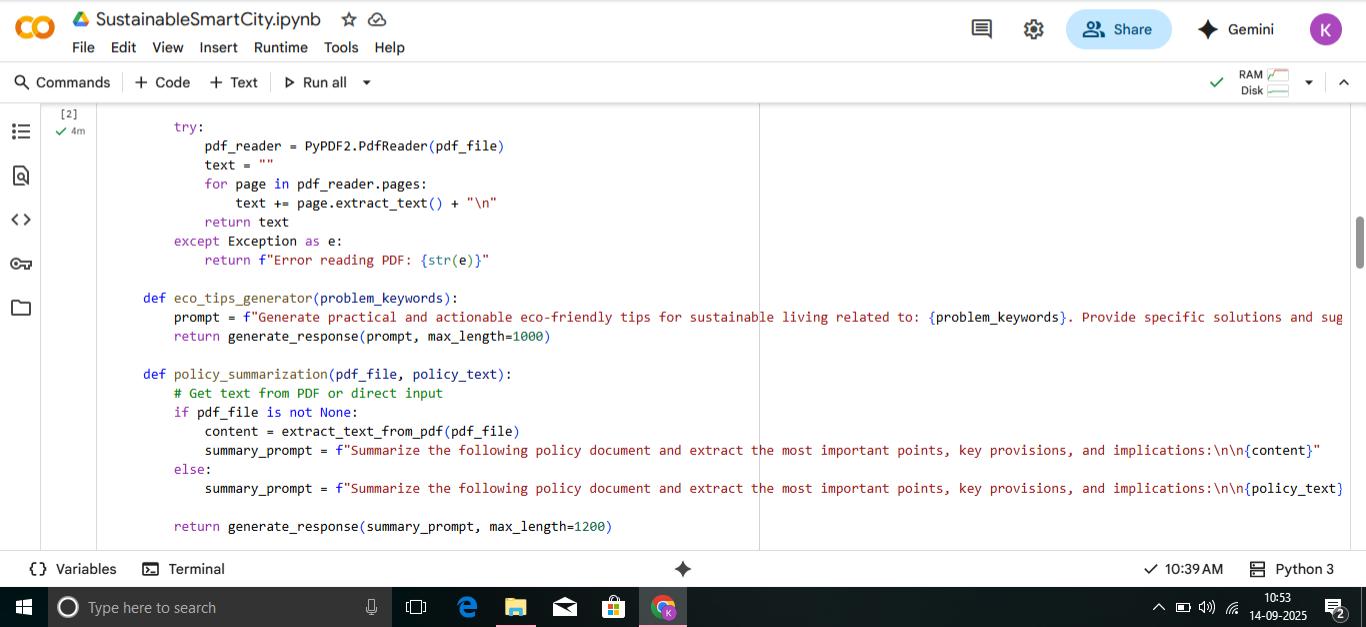
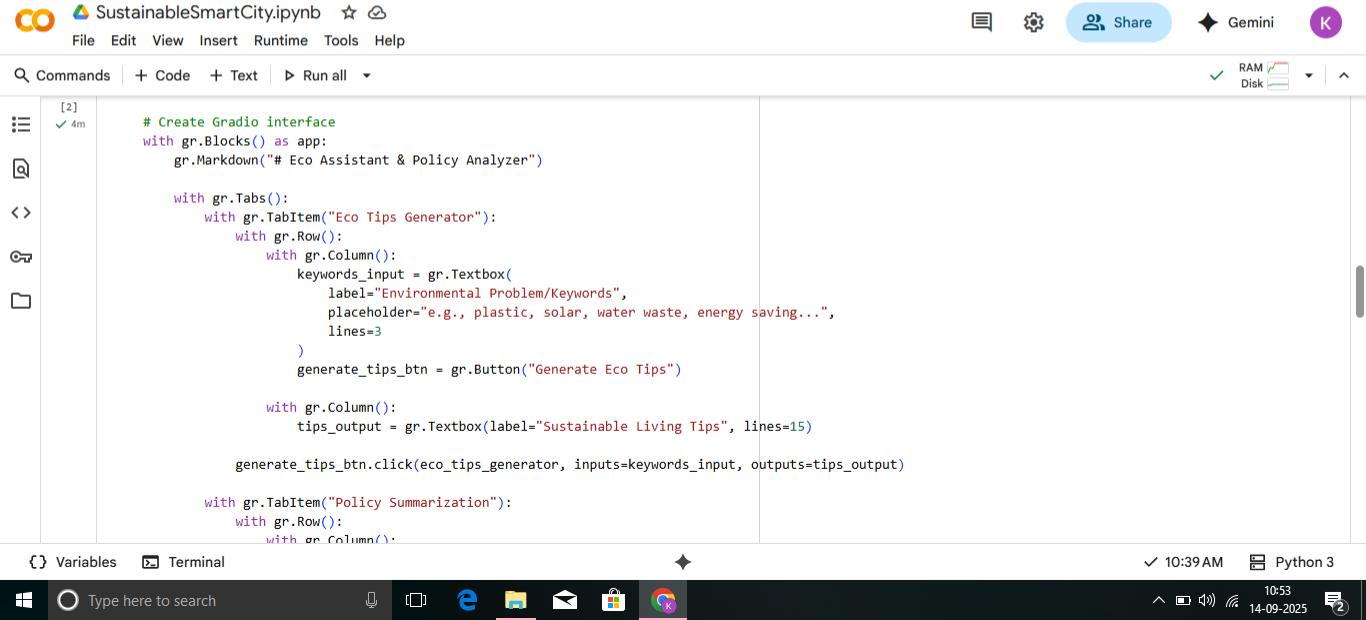
****

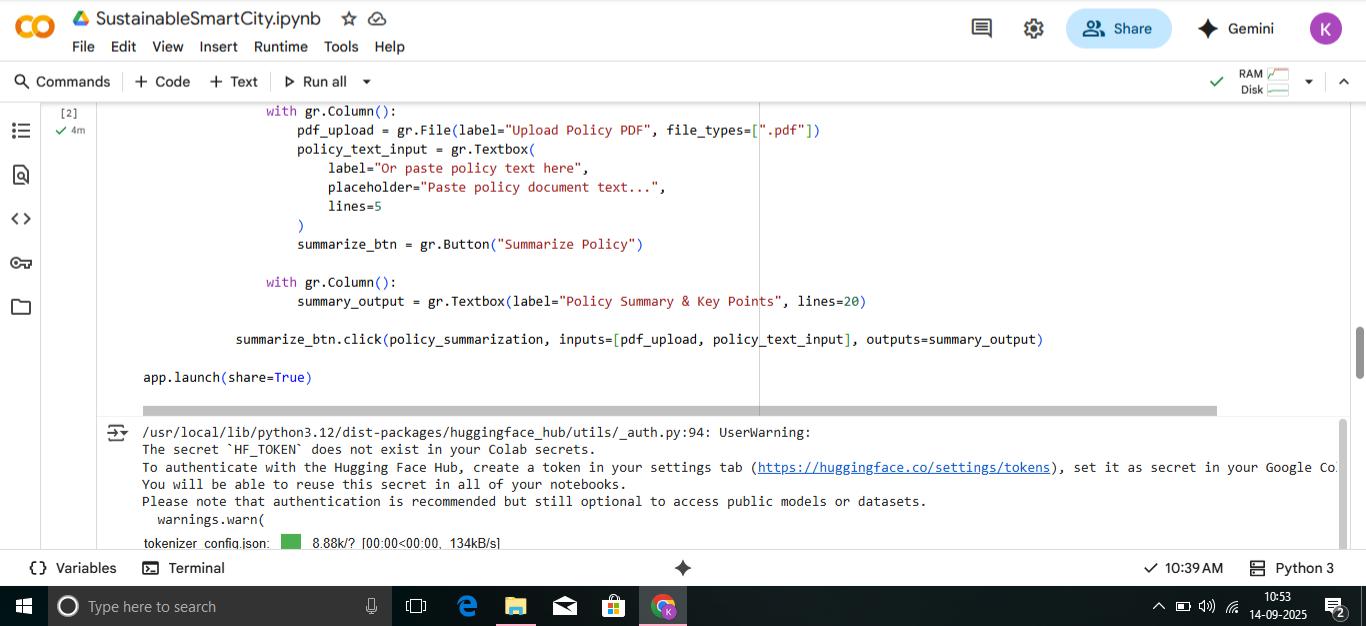
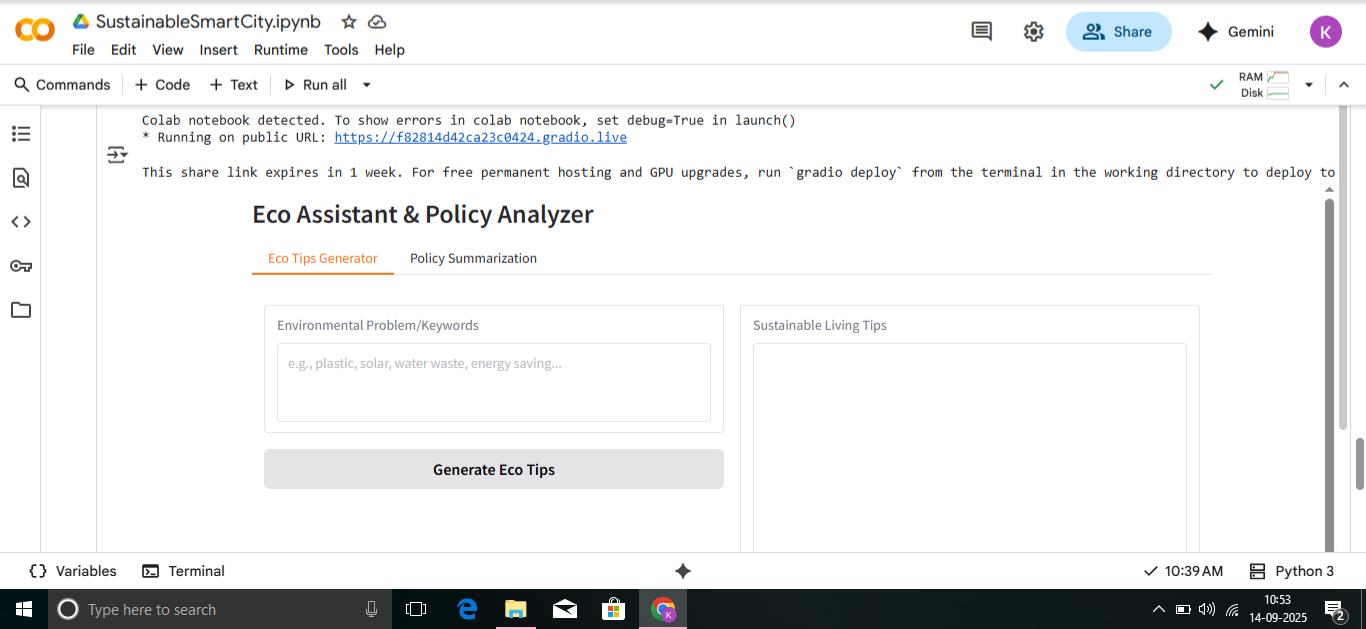
****

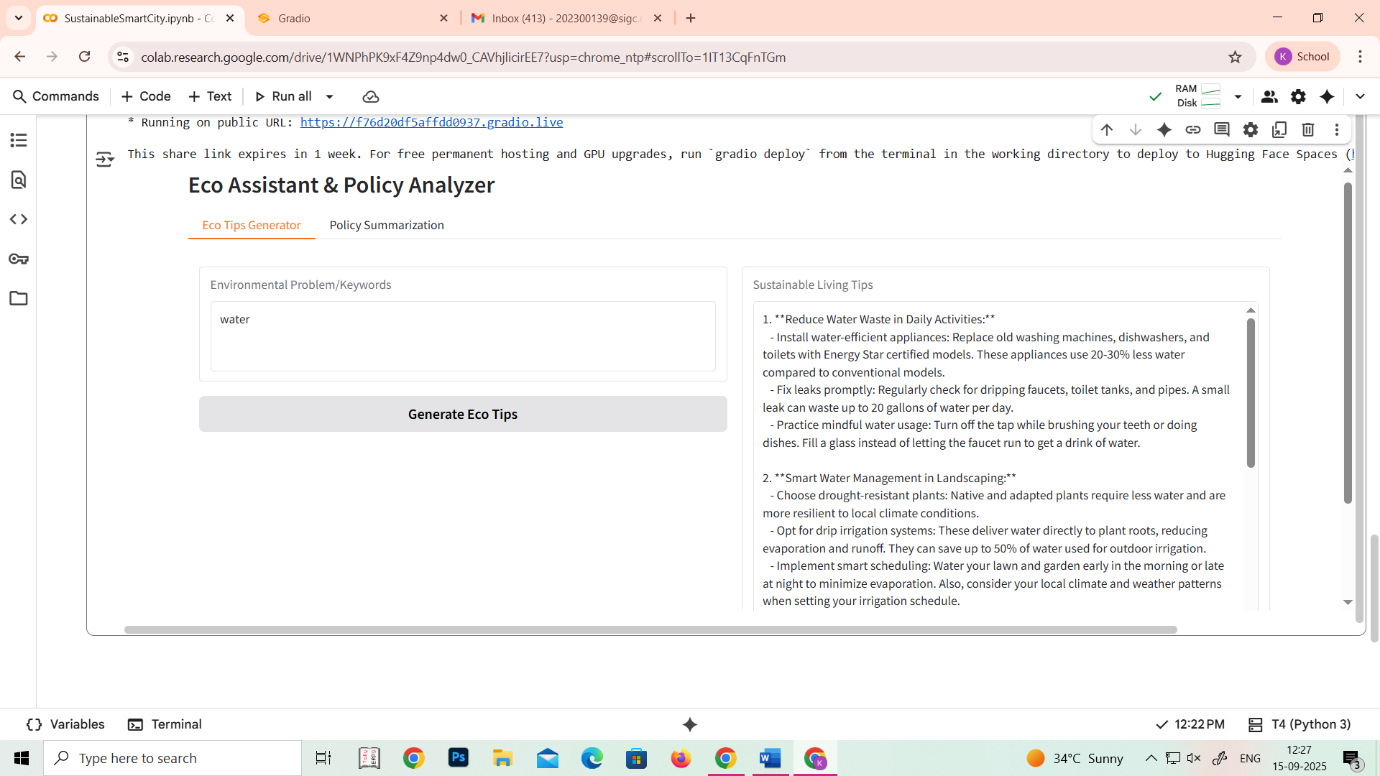
****

****

****

****

****

****

1. **Known Issues**

Large PDF Files → Processing may take extra time or fail for very big documents.

Forecasting Accuracy → Predictions depend on the quality and size of the dataset.

Dependency on Internet → Some modules (LLM, vector search) need internet and API keys.

Limited Offline Mode → AI features like policy summarization require online access.

Basic UI → The current interface is simple and may need improvement for large-scale use.

1. **Future enhancement**

Multi-language support → Provide eco-tips and summaries in regional languages.

Mobile App Integration → Extend the assistant to Android/iOS for wider reach.

Real-time IoT Data → Connect with smart sensors for live monitoring of energy, water, and waste.

Advanced Forecasting Models → Use deep learning (LSTMs/Transformers) for more accurate predictions.

GIS Mapping Integration → Visualize city data on interactive maps.

User Profiles & History → Allow users to track their eco-tips, reports, and contributions.

Cloud Deployment → Full-scale deployment on