**Smart City Sustainability Assistant – Project Documentation**

**Team members:**

S.Shalini

M.Nesapriya

R.santhiya

P.sharmila

**1. Introduction**

This project is titled *Smart City Sustainability Assistant*. It was developed by a team of four members who collaborated to create a system that promotes eco-friendly urban living. The assistant helps both citizens and city officials make better environmental decisions using artificial intelligence and real-time data.

**2. Project Overview**

The purpose of this project is to help cities become smarter and more sustainable. It does this by using AI to optimize how resources like energy, water, and waste are used. It also gives helpful tips to citizens on how to live more sustainably and assists city officials with important information for planning. The assistant can summarize long policy documents, predict future resource needs, and identify unusual data patterns. Overall, it connects technology, governance, and the community to build greener and more efficient cities.

Some of the main features include a chat assistant that understands natural language, a policy summarizer that simplifies government documents, and forecasting tools that predict resource use. It can also give personalized eco-tips, collect citizen feedback, detect anomalies in data, and allow flexible inputs like text, PDFs, and CSV files. The user interface is built using Streamlit or Gradio, making it simple and easy to use.

**3. Architecture**

The frontend is created using Streamlit or Gradio, providing a web interface with different pages. Users can upload files, view dashboards, chat with the assistant, and submit feedback. The backend uses FastAPI, which handles data processing and communicates with AI models. This setup makes the app fast and responsive.

The assistant uses IBM Watsonx’s Granite language models for tasks like answering questions, summarizing documents, and giving tips. Documents are stored and searched using Pinecone, which allows users to search for related content using simple language. Forecasting and anomaly detection are done using lightweight machine learning models that analyze uploaded data and show results clearly.

**4. Setup Instructions**

To use the project, make sure you have Python 3.9 or higher, internet access, and API keys for IBM Watsonx and Pinecone. First, clone the project repository, install all required libraries, and add your API credentials in a .env file. Then, run the backend server using FastAPI and start the frontend using Streamlit. Once everything is running, you can upload documents, interact with the assistant, and use all available features.

**5. Folder Structure**

The backend code is stored in the app folder, which includes API routes and functions for tasks like chatting, summarizing documents, and generating reports. The ui folder contains the frontend files used for building the web interface. The main Streamlit file is called smart\_dashboard.py. Other important files include the AI model handler, document embedding script, forecaster, anomaly checker, and report generator.

**6. Running the Application**

To start the application, first run the FastAPI server to activate all backend functions. Then, launch the Streamlit dashboard to access the web interface. You can move between pages using the sidebar. You’ll be able to upload documents, use the chatbot, view summaries, and generate reports, all in real-time using the backend APIs.

**7. API Documentation**

The project provides several APIs for communication between the frontend and backend. These include APIs for chatting with the assistant, uploading documents, searching policies, getting eco-tips, and submitting feedback. All these APIs are documented and can be tested using Swagger UI, which shows how to use each one easily.

**8. Authentication**

For demonstration, the current version of the project is open and does not require login. However, in future versions, secure access will be added. This includes token-based login, integration with IBM Cloud accounts, and roles like admin, citizen, and city official. There are also plans to track user sessions and history to improve the experience.

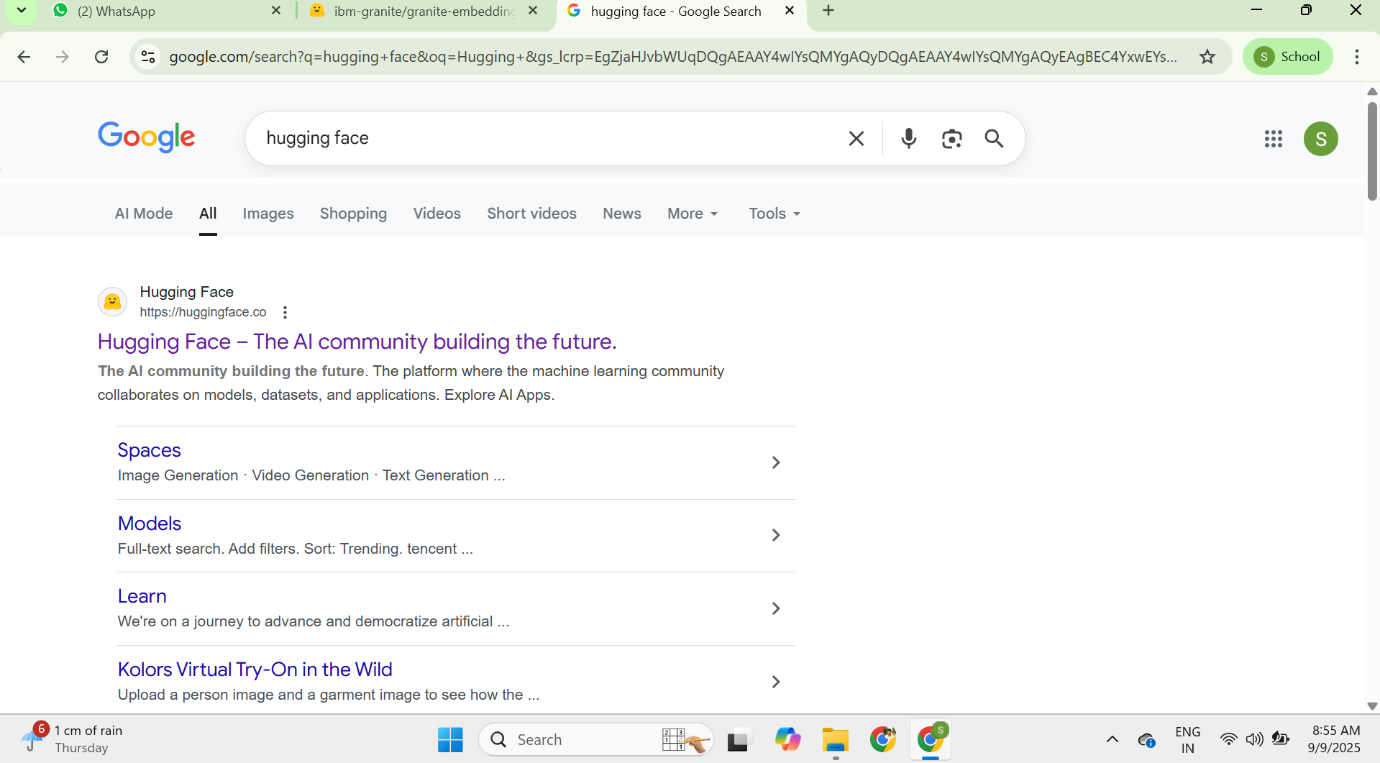
**9. User Interface**

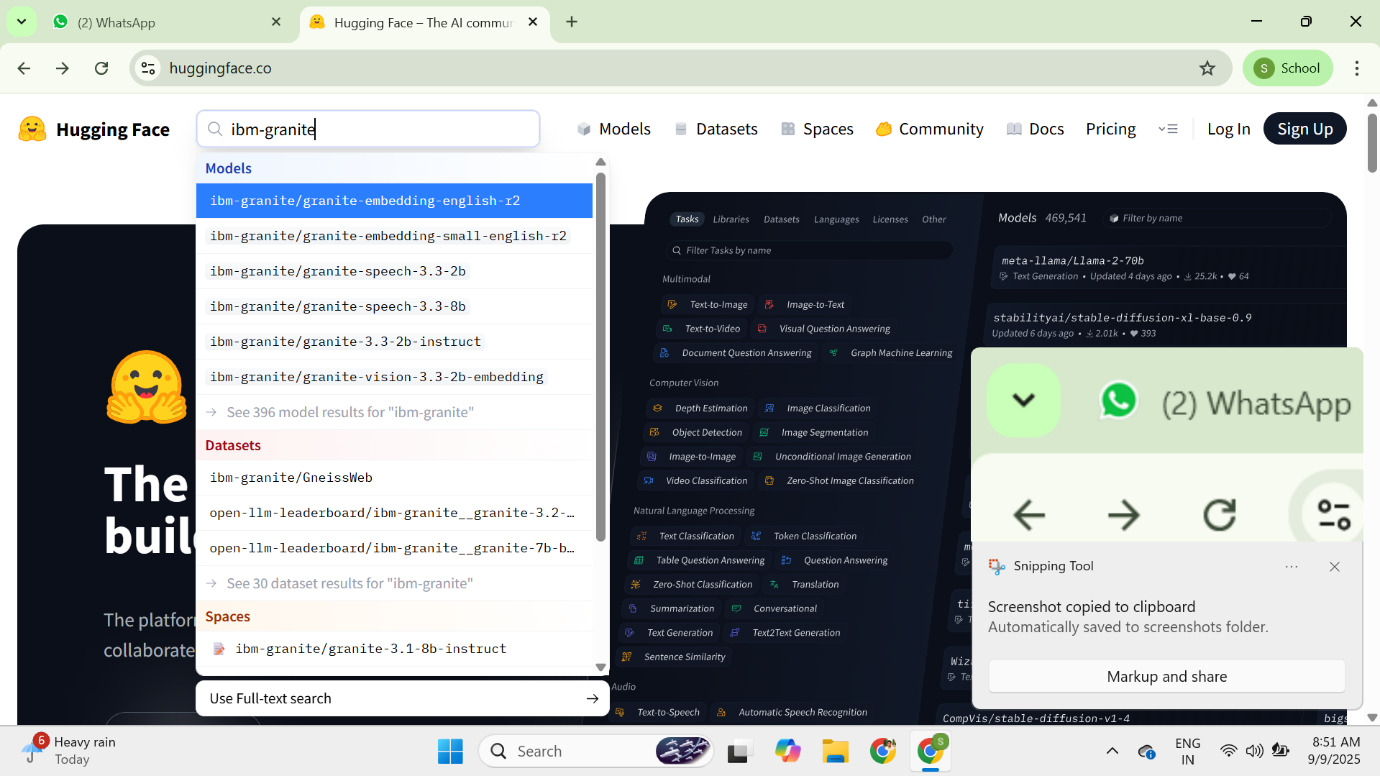
The user interface is simple and user-friendly. It includes a sidebar for navigation, visual cards for showing results, and tab layouts for different tools like eco-tips and forecasting. Users can fill out forms, upload files, and download reports easily. The design focuses on clarity, speed, and accessibility, especially for people who are not technical.

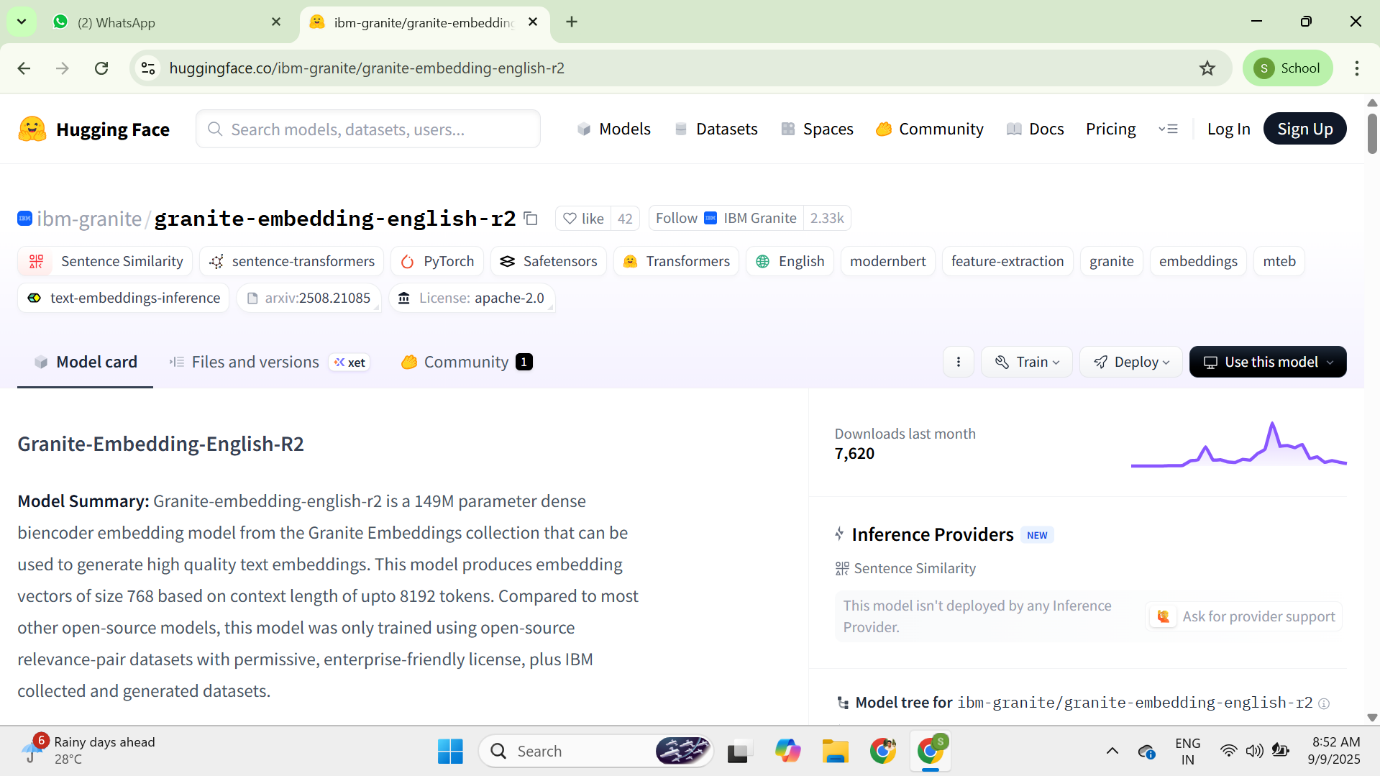
**10. Testing**

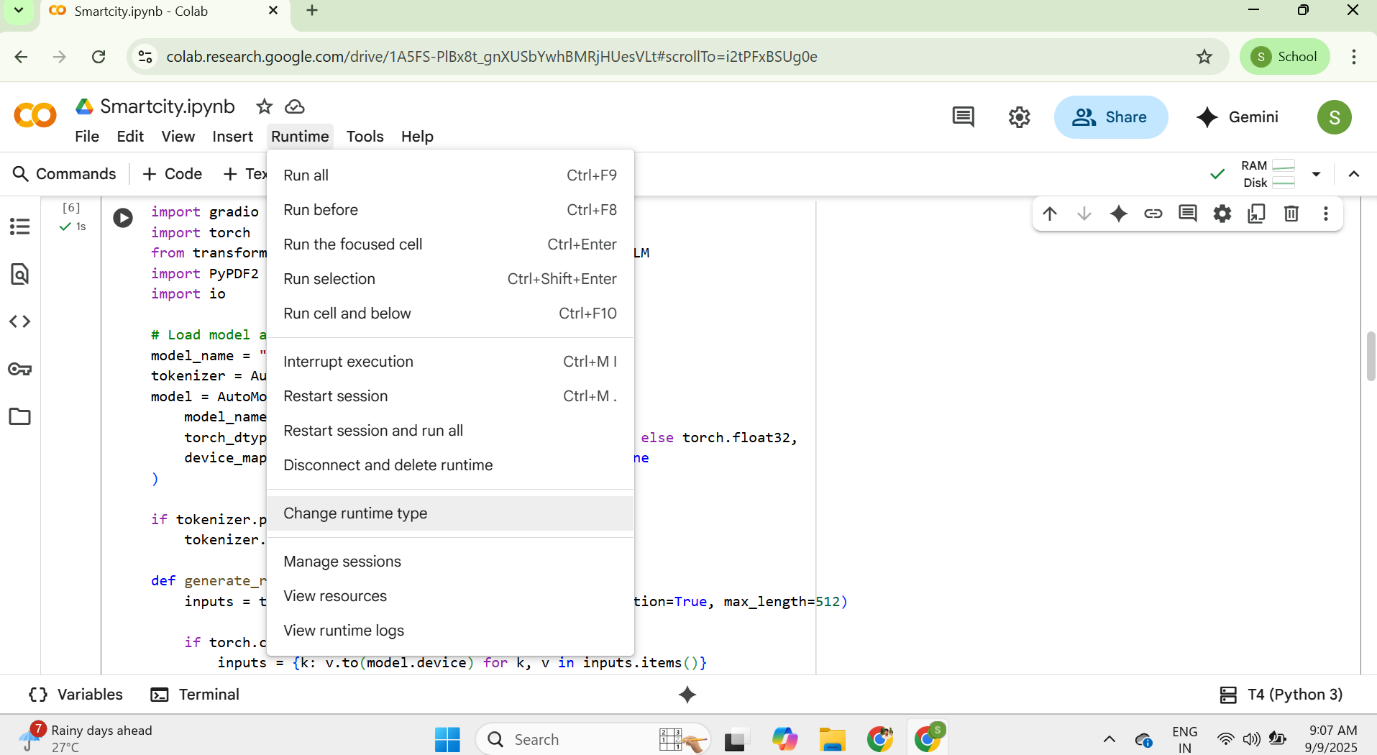
The application was tested in several ways. Unit testing was done for the prompt and AI response logic. API testing was done using Swagger UI and Postman. Manual testing was carried out by uploading files, interacting with the chatbot, and checking the outputs. It was also tested with incorrect inputs like broken files and missing data to make sure the system handles errors properly.

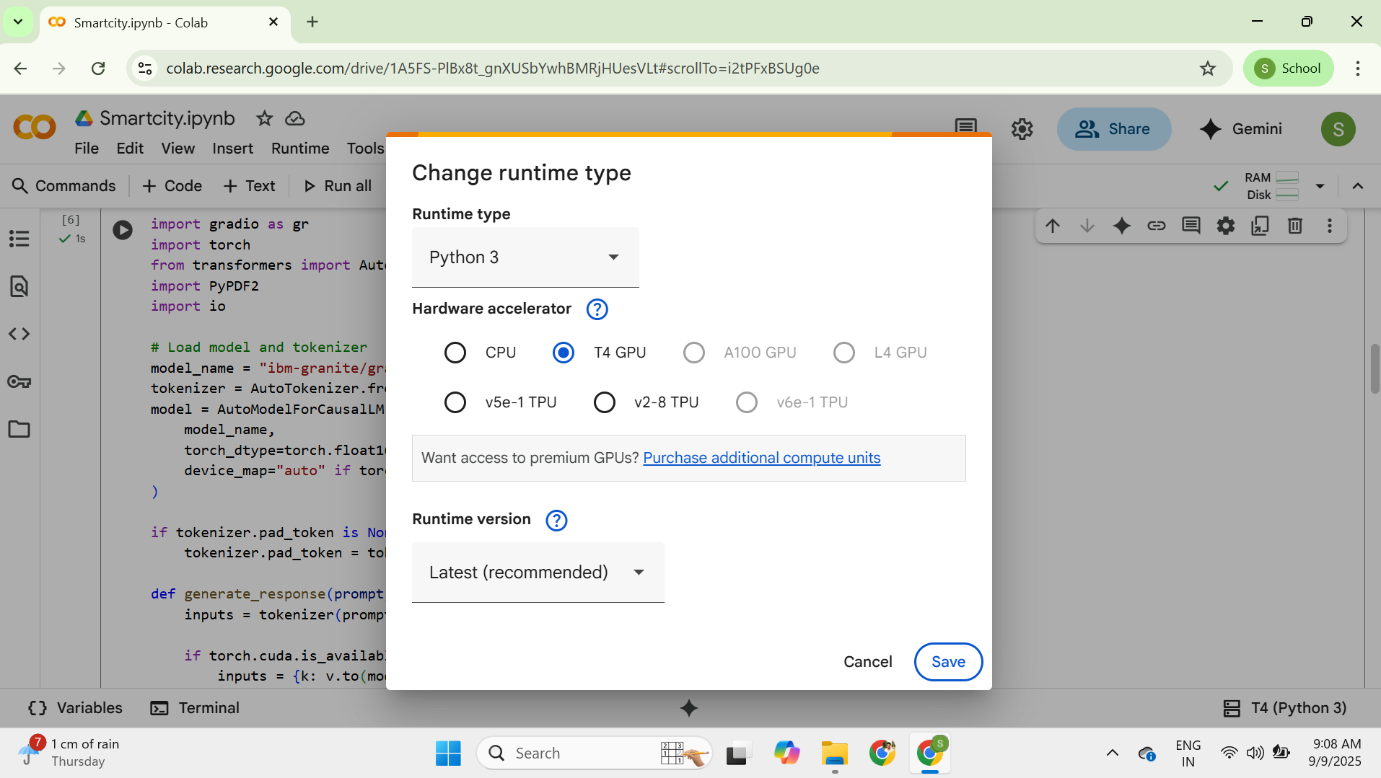
**11. Screenshots**

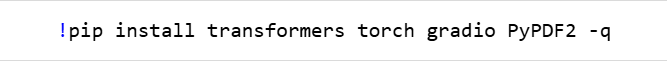
****

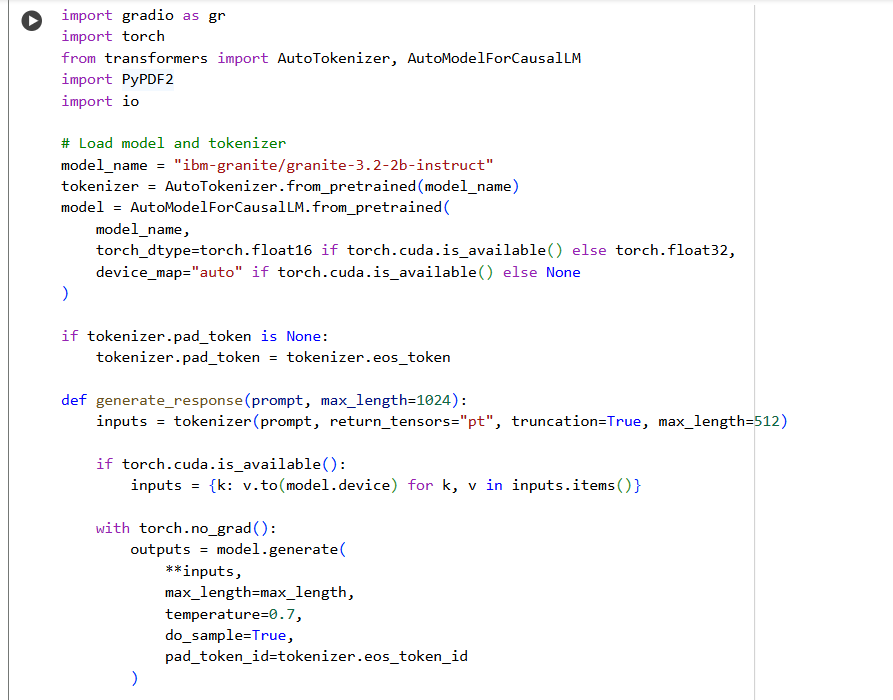
****

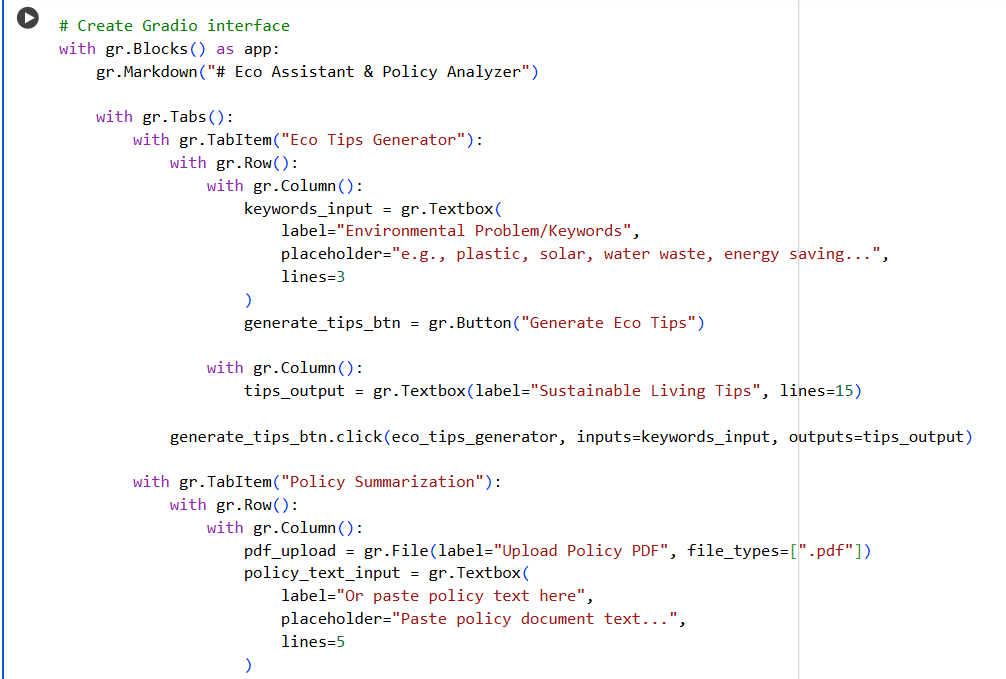
****

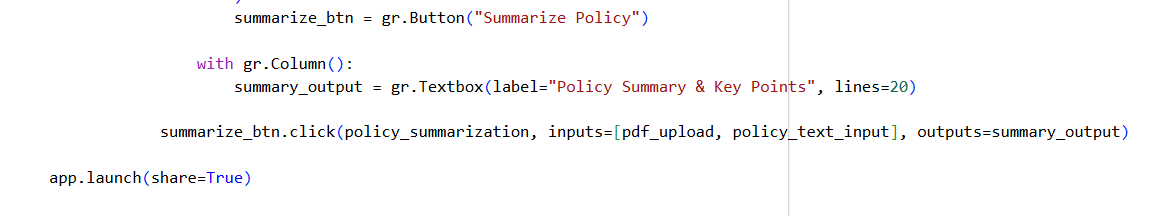
****

****

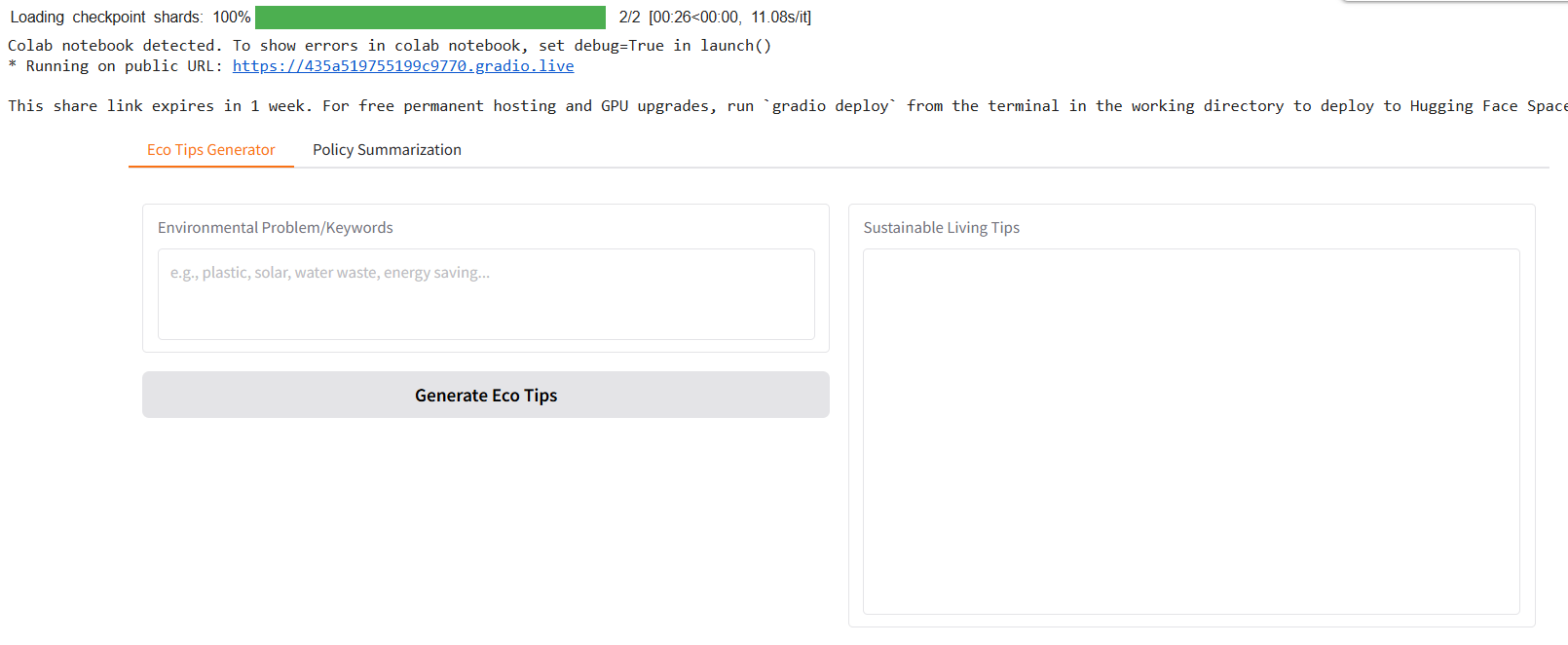
****

****

****

****

**OUTPUT:**

****

**12. Known Issues**

Currently, the summarization of very large documents may take time. Pinecone search may not work if the document hasn’t been uploaded correctly. The assistant doesn’t store user history yet, and the AI model has a limited context window, which may affect very long inputs. These issues will be addressed in future updates.

**13. Future Enhancements**

In the future, the project will include user login and session tracking, better visual dashboards for resource data, automatic report scheduling, voice support for accessibility, and the ability to support more languages. There are also plans to let multiple users work together on reports and city planning.