**Project Title**

**Project Documentation**

**1. Introduction:**

* **Project Title**: Sustainable Smart City Assistant
* **Team Member 1**: Kayalvizhi. P
* **Team Member 2**: Gayathri. S
* **Team Member 3**: Nivitha Varshini. S
* **Team Member 4**: Yajjala Nishitha

**2. Project Overview:**

* **Purpose**:  
  The Sustainable Smart City Assistant is designed to help cities and their communities move towards a more sustainable, eco-aware, and connected future. With the use of artificial intelligence and real-time insights, the platform supports efficient usage of vital resources such as energy, water, and waste management. It also encourages eco-friendly habits among residents through customized tips and guidance. For administrators, it becomes a strategic decision-making partner by simplifying complex documents, offering predictive analysis, and generating sustainability reports. Ultimately, this tool strengthens the link between governance, technology, and citizens to create inclusive and resilient cities.

**2.2. Key Features:**

**Conversational Interface**

* *Key Point*: Interaction in natural language
* *Functionality*: Enables residents and officials to ask queries, obtain updates, and receive easy-to-understand responses.

**Policy Summarization**

* *Key Point*: Simplified access to regulations
* *Functionality*: Converts lengthy city policy documents into concise, clear, and actionable summaries.

**Resource Forecasting**

* *Key Point*: Smart predictions
* *Functionality*: Estimates future demand for water, power, and waste disposal using both historical and real-time inputs.

**Eco-Tip Generator**

* *Key Point*: Personalized eco-friendly suggestions
* *Functionality*: Provides daily recommendations that help users reduce environmental impact and adopt sustainable living.

**Citizen Feedback Loop**

* *Key Point*: Public participation
* *Functionality*: Collects and analyses feedback from citizens to guide service upgrades and sustainability strategies.

**KPI Forecasting**

* *Key Point*: Progress tracking for city goals
* *Functionality*: Helps officials monitor and forecast key performance indicators related to sustainability.

**Anomaly Detection**

* *Key Point*: Early warning of irregularities
* *Functionality*: Detects unusual or unexpected patterns in usage data and flags them for review.

**Multimodal Input Support**

* *Key Point*: Versatile data handling
* *Functionality*: Accepts multiple input formats like text, PDFs, and CSVs for analysis and predictions.

**Streamlet or Gradi UI**

* *Key Point*: Simple and intuitive dashboard
* *Functionality*: Provides an accessible interface for both administrators and citizens to interact with the assistant.

**3. Architecture:**

**Frontend (Stream lit):**  
The frontend is implemented using stream lit to create a lightweight, interactive web interface. It organizes various functions such as uploading files, exploring dashboards, chatting with the assistant, viewing policies, and generating reports. Navigation is streamlined using a sidebar, and the modular structure makes it easy to expand in the future.

**Backend (Fast API):**  
Fast API powers the backend, offering REST APIs for data processing, chat interactions, embedding, and generating reports. The backend is optimized for speed and supports Swagger for documentation and testing.

**LLM Integration (IBM Watson Granite):**  
The Watson Granite language model provides natural language understanding and generation for summaries, eco-advice, and reports.

**Vector Search (Pinecone):**  
Policy documents and data are embedded using Sentence Transformers and stored in Pinecone. Cosine similarity-based semantic search helps users find relevant information quickly.

**ML Modules (Forecasting & Anomaly Detection):**  
Time-series forecasting and anomaly detection are handled using Scikit-learn, while Pandas and Matplotlib are used for data processing and visualization.

**4. Setup Instructions:**

**Prerequisites:**

* Python 3.9 or newer
* pip and virtual environment tools
* API keys for IBM Watson and Pinecone
* Internet connection for accessing services

**Installation Steps:**

1. Clone the project repository
2. Install packages from requirements.txt
3. Create a .env file and configure API credentials
4. Launch the backend with Fast API
5. Run the frontend with stream lit
6. Upload files and begin using the assistant

**5. Folder Structure:**

* **app/** – Contains backend logic such as routes, models, and integrations
* **app/api/** – Submodules for chat, reports, feedback, and embeddings
* **Ui/** – stream lit code for UI components and layouts
* **smart\_dashboard.py** – Script to launch the main dashboard
* **granite\_llm.py** – Handles IBM Watson Granite LLM-based summarization and chat
* **document\_embedder.py** – Embeds documents into Pinecone
* **kpi\_file\_forecaster.py** – Predicts energy, water, and waste usage trends
* **anomaly\_file\_checker.py** – Identifies irregular values in datasets
* **report\_generator.py** – Produces sustainability reports using AI

**6. Running the Application:**

1. Start the backend server (Fast API)
2. Open the stream lit dashboard in a browser
3. Use the sidebar to navigate between modules
4. Upload CSVs or documents
5. Chat with the assistant and generate reports, summaries, or predictions

**7. API Documentation:**

Available APIs include:

* **POST /chat/ask** – Provides AI-generated responses to queries
* **POST /upload-doc** – Uploads and stores documents in Pinecone
* **GET /search-docs** – Returns semantically similar documents based on query
* **GET /get-eco-tips** – Provides eco-friendly tips on energy, water, and waste
* **POST /submit-feedback** – Saves citizen feedback for later analysis

**8. Authentication:**

For demo use, the system runs openly. For secure deployments, the following can be added:

* JWT or API Key authentication
* OAuth2 with IBM Cloud credentials
* Role-based access for admins, researchers, and citizens
* Session management and history tracking (future enhancement)

**9. User Interface:**

Key elements of the UI include:

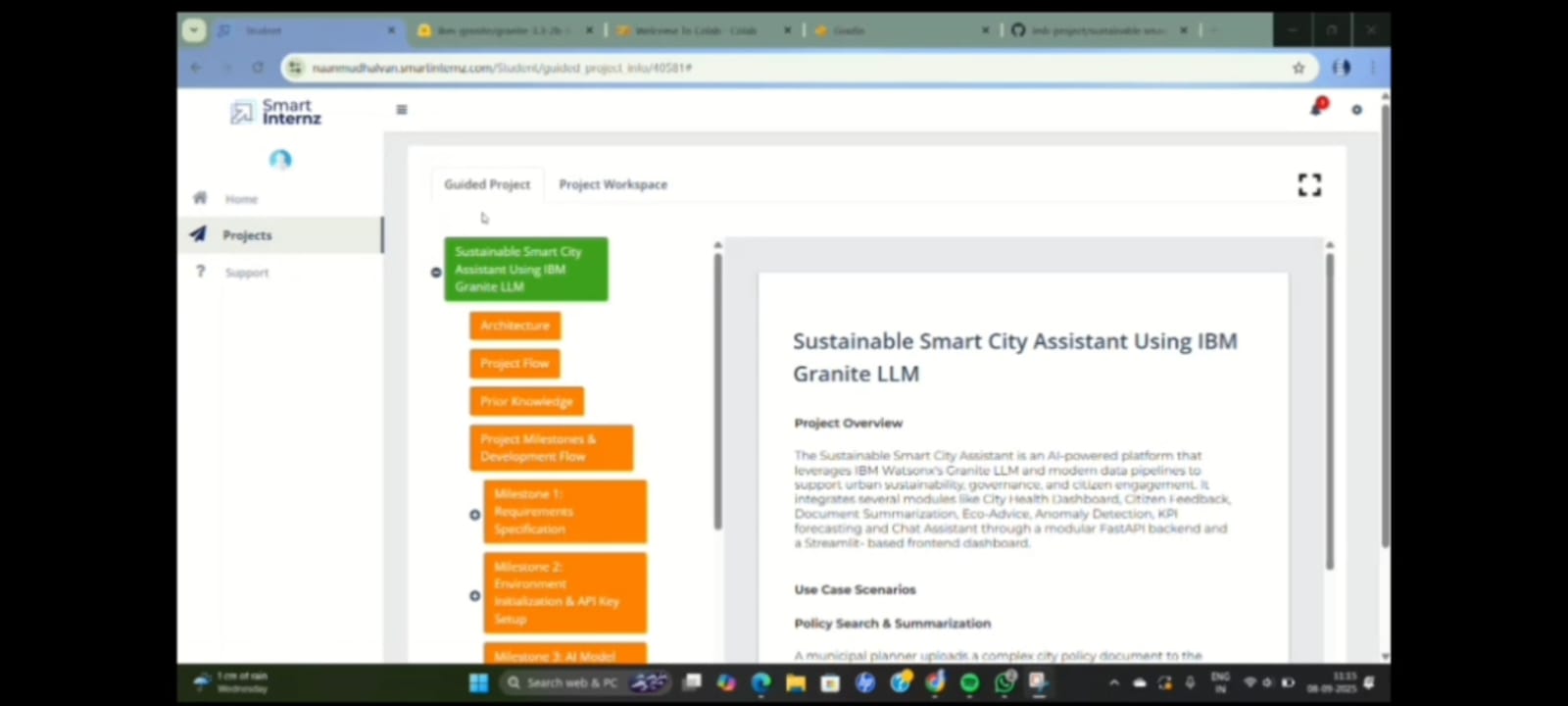
* Sidebar for navigation
* KPI visualizations and summary cards
* Tabs for chat, eco-tips, and forecasting
* Real-time form submissions
* Option to export reports in PDF format

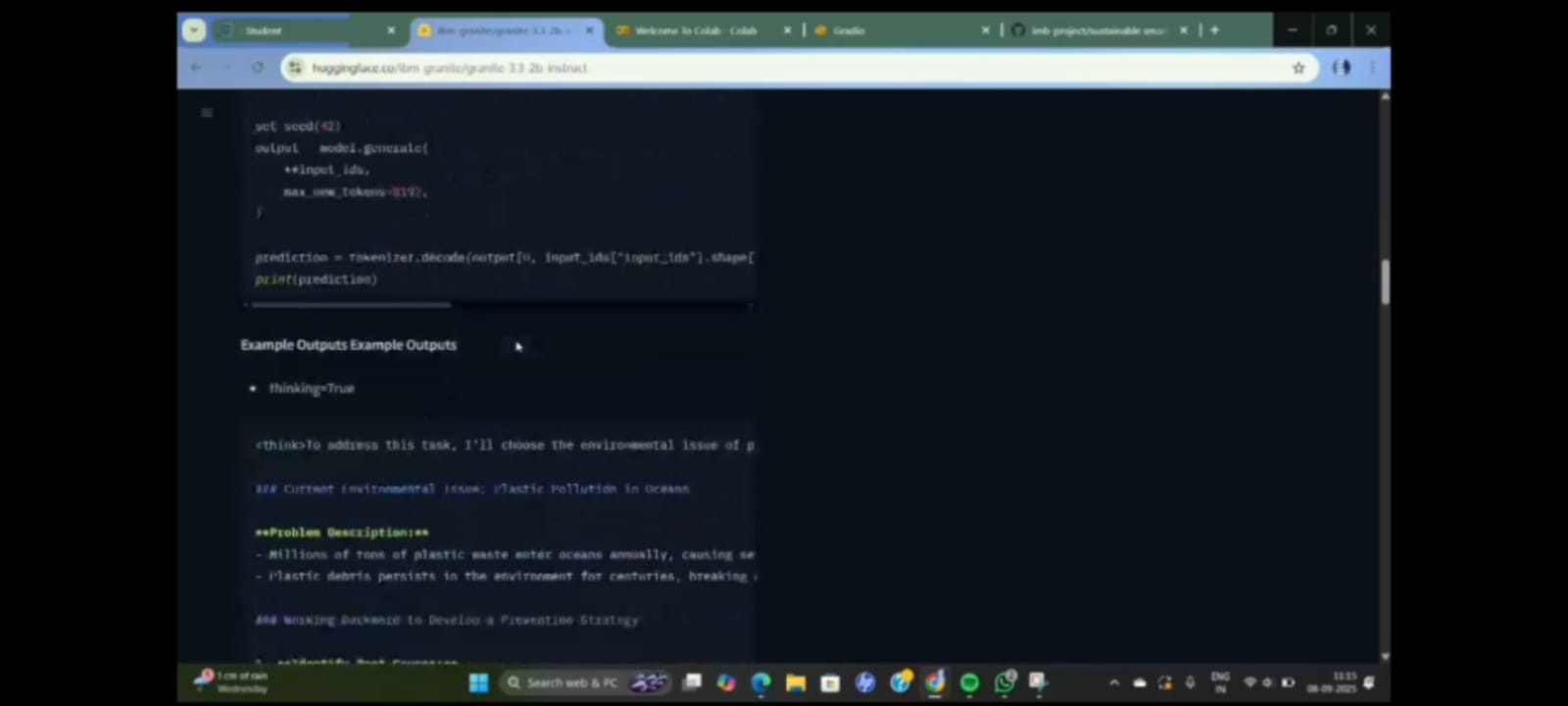
**10. Testing:**

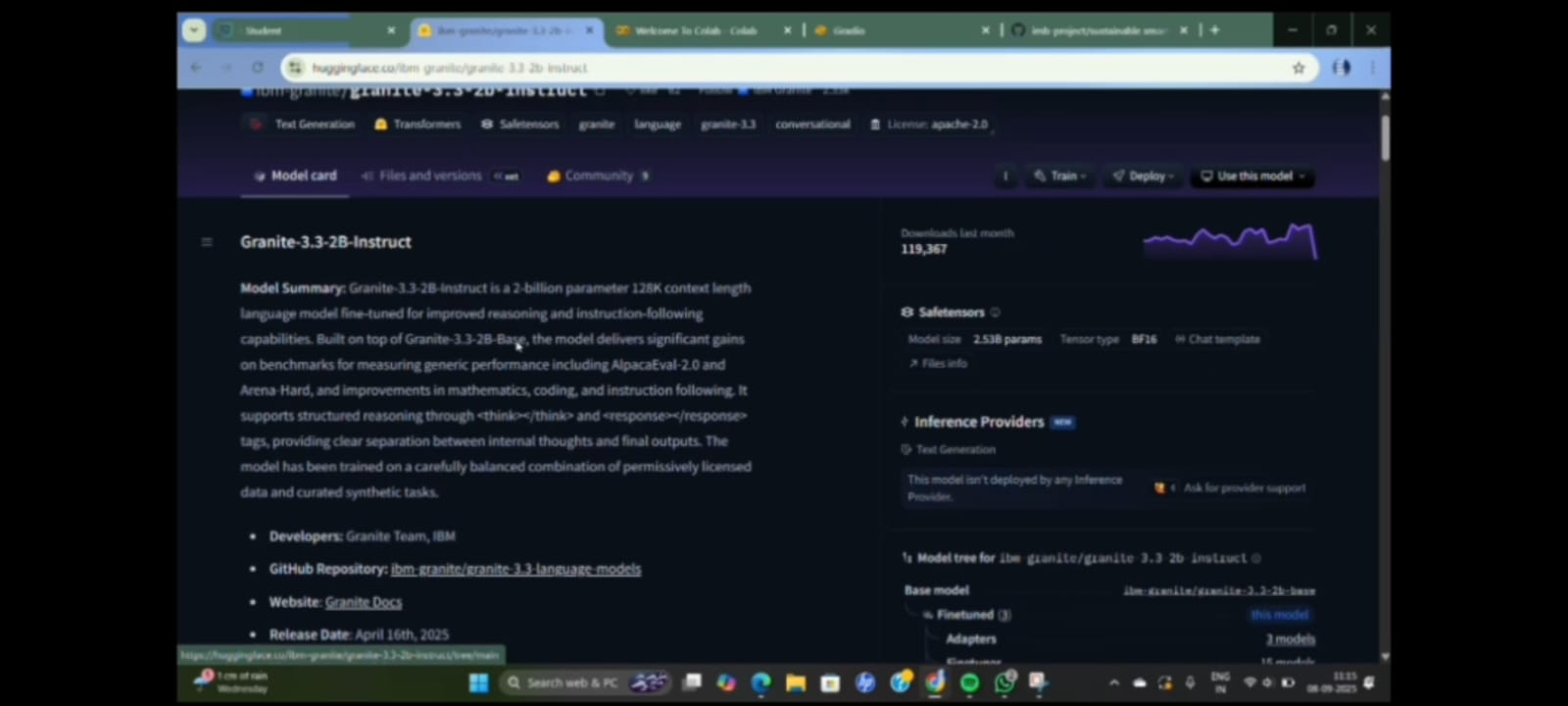
Testing covered different stages:

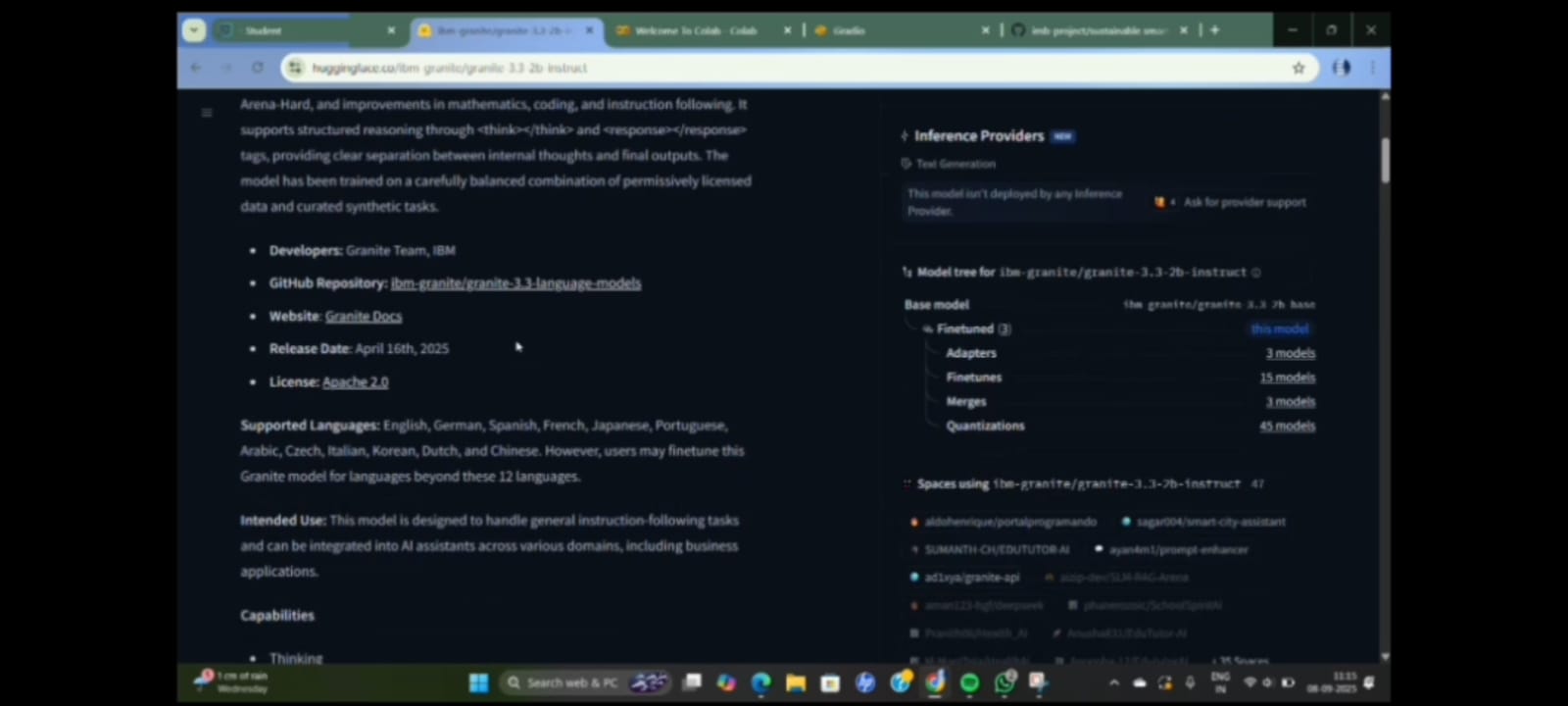
* **Unit Testing** – Verification of utility scripts and prompt functions
* **API Testing** – Swagger UI, Postman, and automated test scripts
* **Manual Testing** – File uploads, chat queries, and output checks
* **Edge Case Handling** – Large files, wrong API keys, malformed inputs

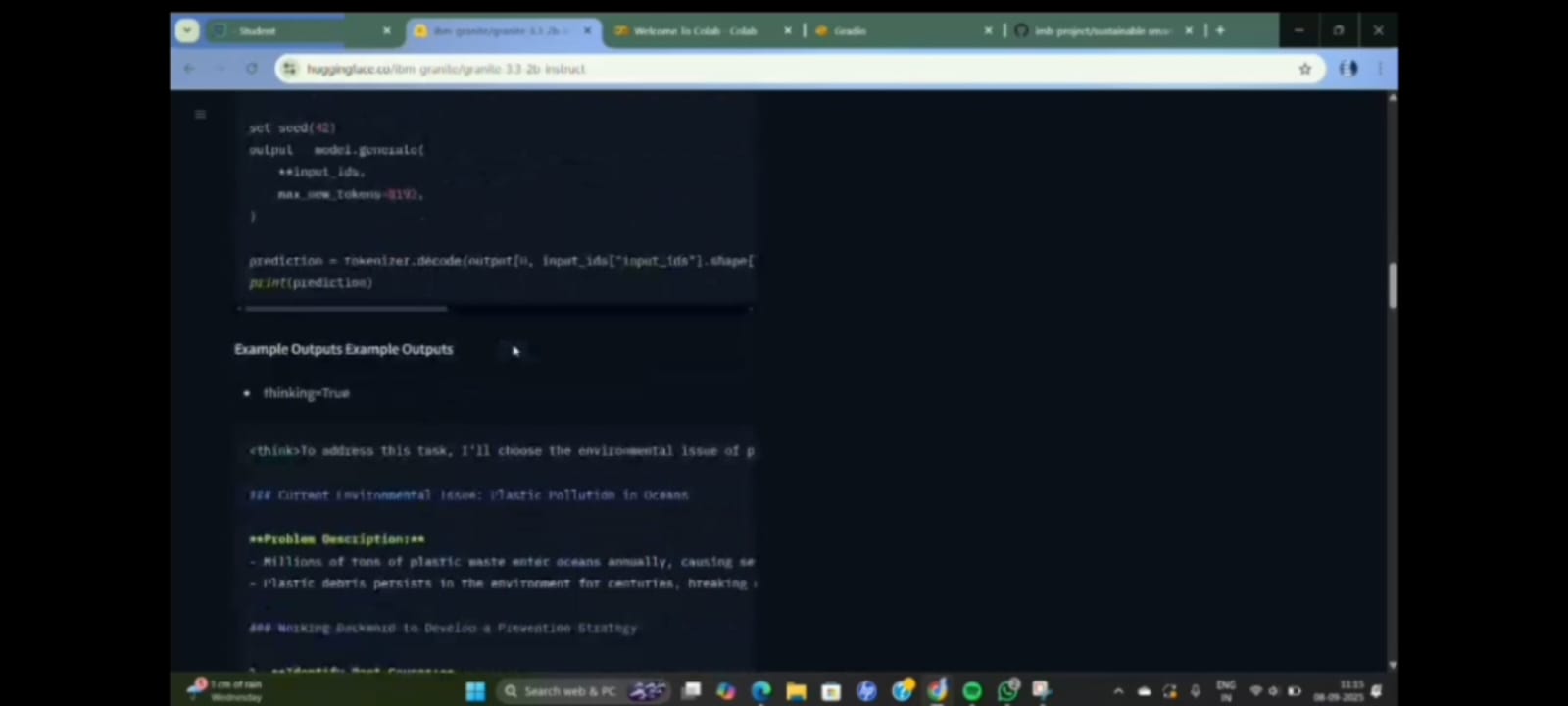
**11. Screenshots:**

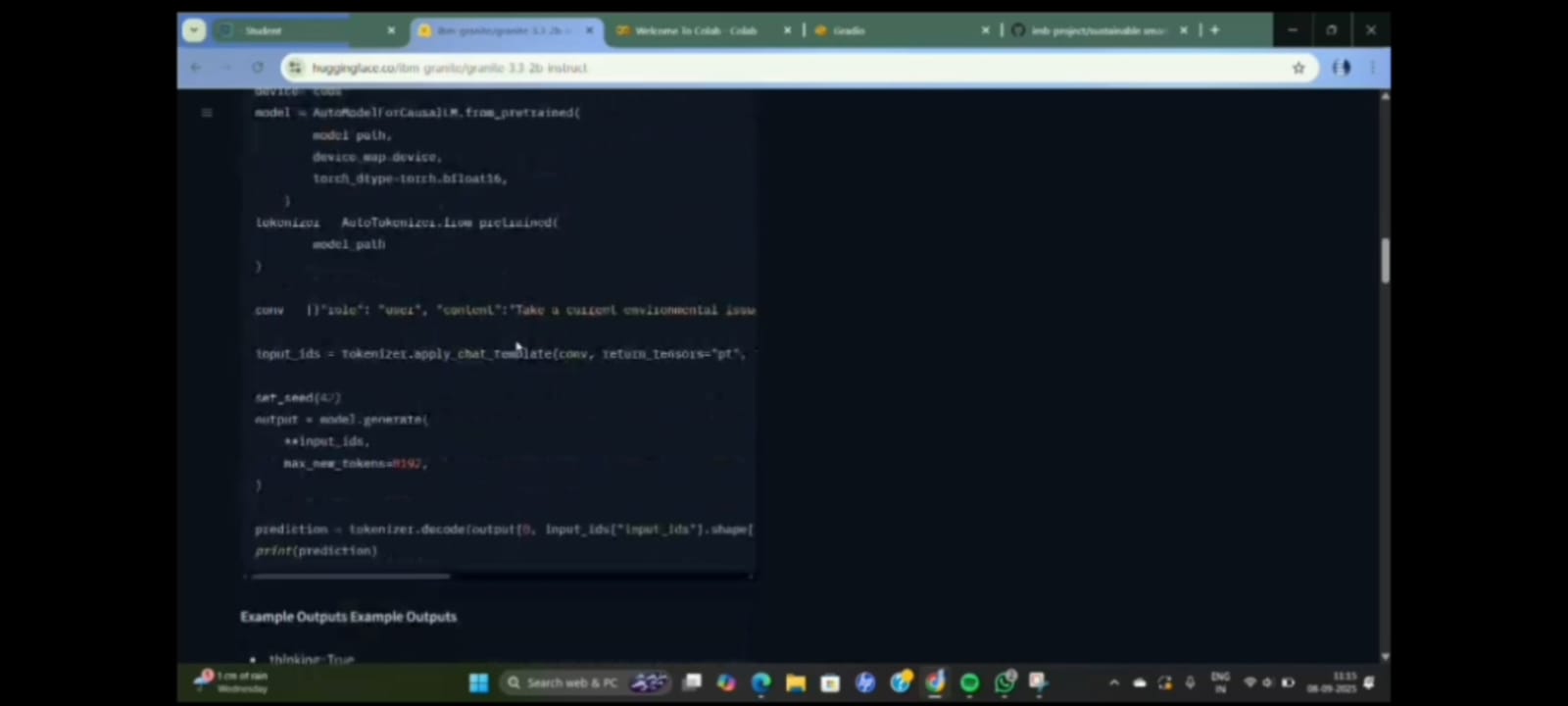


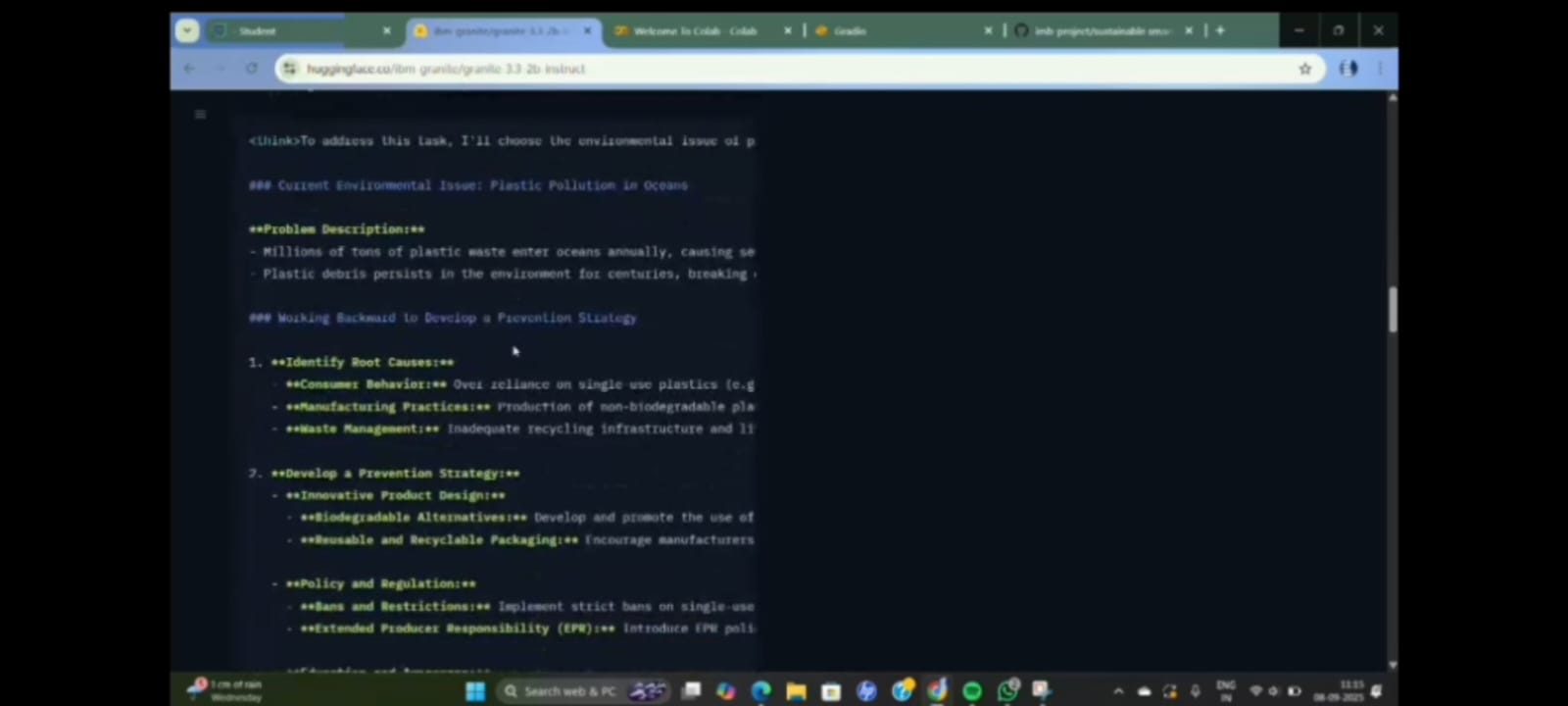










****

**12. Known Issues:**

**1. Response Delay with Large Datasets**

* When users upload very large CSV files or policy documents, the backend sometimes takes longer to process and generate outputs. This is mainly due to the time required for embedding and model inference.

**2.** **Limited Multilingual Support**

* Currently, the system mainly supports English. Queries in regional or non-English languages may not be understood accurately by the LLM, limiting accessibility for diverse citizen groups.

**3.** **Scalability Constraints**

* Current deployment is designed for pilot or small-scale use. Handling city-wide real-time data streams (e.g., IoT sensors across thousands of households) may require advanced infrastructure like distributed processing.

**4. Security and Privacy Risks**

* Since sensitive data (like citizen feedback or policy drafts) is uploaded, there is a risk of unauthorized access if strong authentication and encryption are not implemented.

**13. Future Enhancements:**

**1.** **Multilingual and Voice Support**

* Extend the assistant to support multiple regional languages and voice-based interaction, ensuring accessibility for a wider range of citizens.

**2.** **IoT Integration for Real-Time Data**

* Connect the system with IoT devices such as smart meters, water sensors, and waste tracking systems for continuous real-time monitoring.

**3.Mobile Application Development**

* Create mobile apps for Android and iOS platforms to allow citizens and officials to access the assistant anytime, anywhere.

**4.AI-Powered Policy Recommendations**

* Enhance the system to not only summarize but also provide suggestions and predictive insights for drafting and evaluating new policies.

**5.Advanced Anomaly Detection**

* Improve the anomaly detection module with advanced machine learning techniques to reduce false positives and provide more accurate alerts.