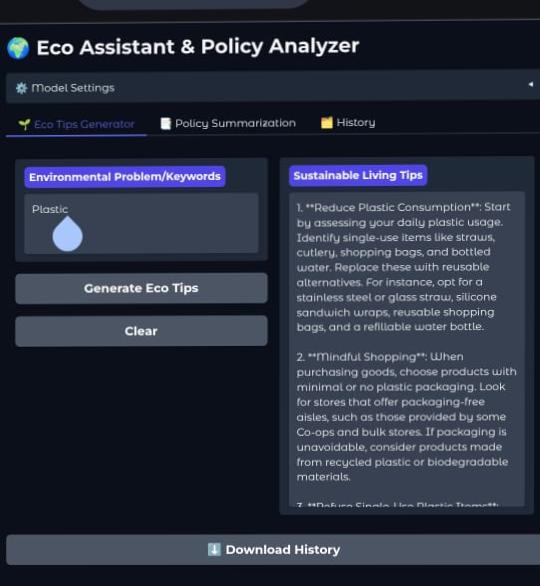
Project Documentation

Project Documentation: Sustainable Smart City Assistant using IBM Granite LLM  
  
**1. Introduction**  
Project Title: Sustainable Smart City Assistant using IBM Granite LLM  
  
Team Members:  
- Muktha Sri M  
- Mugilarasi D  
- Narmatha S  
- Nirmala R  
  
**2. Project Overview**  
Purpose: To build an AI-powered eco-assistant that generates sustainable living tips and summarizes policy documents for smart cities.  
  
How it Helps:  
- Citizens → Get personalized eco-tips (plastic reduction, energy saving, water conservation).  
- City Officials → Upload long policy PDFs, get concise summaries and key provisions for decision-making.  
  
Tech Used: IBM Granite LLM, Gradio UI, PyPDF2, Torch.  
  
**3**. **Features**  
🌱 Eco Tips Generator → Suggests practical actions for sustainable living.  
📑 Policy Summarization → Summarizes uploaded PDF/text documents into key points.  
🗂 History Tracking → Stores eco-tips & summaries, downloadable as text file.  
⚡ Conversational Interface → Natural language interaction via Gradio blocks.  
📊 Expandable → Can later add KPI forecasting, anomaly detection, feedback system.  
  
**4. Architecture**  
Frontend (Gradio):  
- Tabs for Eco Tips, Policy Summarization, and History.  
- File upload for PDFs, text input for policies.  
  
Backend (Torch + IBM Granite LLM):  
- Handles prompt → response generation.  
- Uses tokenizer + model from ibm-granite/granite-3.2-2b-instruct.  
  
Storage: In-memory history dictionary for eco & policy outputs.  
Extensibility: Can integrate Pinecone/DB for semantic search later.  
  
**5. Setup Instructions**  
Requirements:  
- Python 3.9+  
- Libraries: torch, transformers, gradio, PyPDF2  
  
Steps:  
1. Install dependencies → pip install torch transformers gradio pypdf2  
2. Run the script → python app.py  
3. Open the local Gradio link (or shareable link).  
 **6. Folder Structure**smart\_city\_assistant/  
│-- app.py (main file with Gradio UI & model integration)  
│-- history/ (stores downloaded summaries/tips)  
│-- requirements.txt  
  
**7. Running the Application**  
1. Start the script → launches Gradio Blocks UI.  
2. Select a tab:  
- Eco Tips → enter keywords, get suggestions.  
- Policy Summarization → upload PDF / paste text.  
- History → view/download all results.  
  
**8. API Documentation (Future Scope – if FastAPI added)**POST /generate-eco-tips  
POST /summarize-policy  
GET /download-history  
  
**9. Authentication**Current version: Open demo (no login).  
Future enhancements: JWT tokens, role-based access (citizen, official, admin).  
  
**10. User Interface**Built with Gradio Tabs + Accordions.  
Sliders for max length and temperature.   
Clear buttons to reset outputs.  
Download button for saving results.  
  
**11. Testing**✅ Eco Tips tested with multiple keywords (plastic, solar, water).  
✅ PDF Summarization tested with sample policy docs.  
✅ Edge cases: empty input, invalid PDF.  
Future: Unit testing & Postman API test cases (if backend APIs extended).

  
**12. Known Issues**Large PDFs → summarization may truncate due to token limits.  
History is session-based (resets when script restarts).  
No semantic search (yet).   
  
**13. Future Enhancements**✅ Add KPI forecasting (energy/water demand).  
✅ Add anomaly detection for city data.  
✅ Use Pinecone/Vector DB for semantic policy search.  
✅ Add user login + role-based dashboard.  
✅ Multi-language support (English + Tamil).