**SUSTAINABLE SMART CITY ASSISTANT**

***PROJECT DOCUMENTATION***

**Team Members**

* Subhashree J
* Sindhu M
* Venisha G
* Shailaja P

**1. Introduction**

The Sustainable Smart City Assistant is an AI-powered platform designed to support eco-friendly urban development. It leverages IBM Granite LLM via Hugging Face to generate eco-tips, analyze city policies, forecast resource usage, and provide citizens with actionable insights for greener living.

**2. Project Overview**

**Purpose**

The purpose of this project is to empower cities and their residents to make informed decisions about sustainability. The system provides an interactive web interface where citizens, students, and policymakers can access AI-powered tools for environmental consciousness and policy understanding.

**Key Features**

* **Eco Tips Generator** -- Provides personalized tips on waste management, renewable energy, smart transportation, and water conservation
* **Policy Summarization** -- Converts lengthy government policies into digestible summaries (supports PDF uploads and text input)
* **Interactive Q&A Chatbot** -- Conversational AI for sustainability-related queries with chat history
* **Carbon Footprint Estimator** -- Calculates daily CO₂ emissions based on transportation, electricity, and diet
* **Renewable Energy Advisor** -- Location and budget-specific recommendations for renewable energy solutions
* **Analytics Dashboard** -- Visual usage statistics and system performance metrics
* **Export Center** -- Download all generated content as ZIP files for offline use

**3. System Architecture**

The system is built with a **Gradio-based frontend** for user interaction, **Python backend** for logic and AI integration, and **IBM Granite LLM** (via Hugging Face) for natural language processing. The architecture is designed to run efficiently on Google Colab or local systems with GPU support.

**Components:**

* Frontend: Gradio web interface with tabbed navigation
* AI Engine: IBM Granite 3.2-2B Instruct model via Transformers
* Document Processing: PyPDF2 for policy document analysis
* Data Visualization: Matplotlib for usage analytics
* File Management: Built-in export and download functionality

**4. Prerequisites**

* Python 3.9 or above
* Hugging Face account with access token
* IBM Granite LLM access via Hugging Face
* Gradio for interactive web applications
* Google Colab or system with GPU support (recommended)
* Required Python libraries: transformers, torch, gradio, PyPDF2, matplotlib

**Hugging Face Integration**

**1. Create a Hugging Face account & token**

1. Sign up / sign in at Hugging Face and create an access token (Settings → Access Tokens). Give it appropriate scope (e.g., "read"). Keep this token secret.

**2. Recommended secure methods in Google Colab**

Method A --- Use getpass (recommended for interactive Colab):

from getpass import getpass

hf\_token = getpass('Paste your Hugging Face token here: ')

from huggingface\_hub import login

login(hf\_token)

Method B --- Use environment variable (avoid hardcoding tokens in notebooks):

import os

os.environ['HF\_TOKEN'] = 'hf\_xxx' # NOT recommended to hardcode

from huggingface\_hub import login

login(os.environ['HF\_TOKEN'])

Method C --- Use huggingface-cli in shell:

pip install huggingface\_hub

huggingface-cli login

# then paste the token when prompted

**3. Using the token with Transformers (example)**

If you need to access private models, pass the token when loading model weights or rely on the login session:

from transformers import AutoTokenizer, AutoModelForCausalLM

MODEL\_NAME = 'ibm-granite/granite-3.2-2b-instruct'

# If you've already called login(hf\_token) earlier, the session will be used.

# Alternatively, some libraries accept use\_auth\_token parameter:

tokenizer = AutoTokenizer.from\_pretrained(MODEL\_NAME)

model = AutoModelForCausalLM.from\_pretrained(MODEL\_NAME, use\_auth\_token=True)

**4. Best practices & security**

* Never commit your token into public repositories.
* Prefer using getpass or secret managers when sharing notebooks.
* Rotate tokens if you believe they've been exposed.
* Remove tokens from runtime memory before sharing outputs or screenshots.

**5. Installation & Setup**

**Step-by-Step Setup**

1. **Install Required Libraries:**

pip install transformers torch gradio PyPDF2 matplotlib huggingface\_hub

1. **Authenticate with Hugging Face:**

from huggingface\_hub import login

from getpass import getpass

hf\_token = getpass('Enter your HF token: ')

login(hf\_token)

3. **Load the Model:**

from transformers import AutoTokenizer, AutoModelForCausalLM

MODEL\_NAME = "ibm-granite/granite-3.2-2b-instruct"

tokenizer = AutoTokenizer.from\_pretrained(MODEL\_NAME)

model = AutoModelForCausalLM.from\_pretrained(MODEL\_NAME, device\_map="auto")

1. **Launch the Application:**

app.launch(share=True, debug=False)

**6. Application Structure**

**Core Functions**

* **generate\_response()** -- Handles AI text generation with temperature control
* **extract\_text\_from\_pdf()** -- PDF document processing utility
* **eco\_tips\_generator()** -- Creates personalized sustainability recommendations
* **policy\_summarization()** -- Summarizes policy documents from PDF or text
* **sustainability\_chat()** -- Manages conversational AI with history
* **carbon\_footprint\_estimator()** -- Calculates environmental impact metrics
* **renewable\_energy\_advisor()** -- Provides energy solution recommendations

**User Interface Tabs**

1. **Eco Tips Generator** -- Input keywords, receive actionable sustainability tips
2. **Policy Summarization** -- Upload PDF or paste text for policy analysis
3. **Q&A Chatbot** -- Interactive sustainability consultation
4. **Carbon Footprint Estimator** -- Personal environmental impact calculator
5. **Renewable Energy Advisor** -- Customized energy recommendations
6. **Analytics Dashboard** -- System usage visualization
7. **Export Center** -- Download generated content

**7. Running the Application**

**Local Execution**

1. Ensure all dependencies are installed
2. Set up Hugging Face authentication
3. Run the Python script
4. Access the Gradio interface via the provided URL

**Google Colab Execution**

1. Upload the notebook to Google Colab
2. Install libraries using !pip install
3. Authenticate with Hugging Face using getpass
4. Execute all cells
5. Use the public shareable link for access

**8. Usage Analytics**

The system includes built-in usage tracking that monitors:

* Number of eco-tips generated
* Policy documents processed
* Chat interactions completed
* Carbon footprint calculations
* Renewable energy consultations

Analytics are visualized in the dashboard and can be exported for analysis.

**9. File Management**

**Automated File Generation**

* **eco\_tips.txt** -- Generated sustainability recommendations
* **policy\_summary.txt** -- Processed policy summaries
* **dashboard.png** -- Usage analytics visualization
* **ZIP exports** -- Timestamped archives of all generated content

**Export Functionality**

Users can download all generated content as organized ZIP files for offline reference and sharing.

**10. Testing & Validation**

**Functional Testing**

* AI response quality and relevance
* PDF processing accuracy
* Carbon footprint calculation validation
* User interface responsiveness

**Performance Testing**

* Model loading time optimization
* Memory usage monitoring
* Concurrent user handling (Gradio limitations)

**Edge Case Handling**

* Large PDF file processing
* Malformed input handling
* Network connectivity issues
* Token authentication failures

**11. Known Limitations**

* **Model Size:** Requires significant GPU memory for optimal performance
* **Response Time:** Complex queries may take 10-30 seconds to process
* **PDF Processing:** Very large documents may cause memory issues
* **Concurrent Users:** Limited by Gradio's sharing capabilities
* **Language Support:** Currently optimized for English language inputs

**12. Future Enhancements**

**Planned Improvements**

* Integration with real-time city data APIs
* Advanced forecasting models for resource usage
* Multi-language support for global deployment
* Enhanced security features for production use
* Mobile-responsive interface optimization

**Scalability Considerations**

* Migration to FastAPI backend for better performance
* Database integration for persistent data storage
* Kubernetes deployment for high availability
* Load balancing for concurrent user support

**13. Security Considerations**

**Current Security Measures**

* Secure model loading from Hugging Face public repository
* Input validation for user queries
* Safe file handling for PDF uploads
* No external API dependencies for core functionality

**Production Security Requirements**

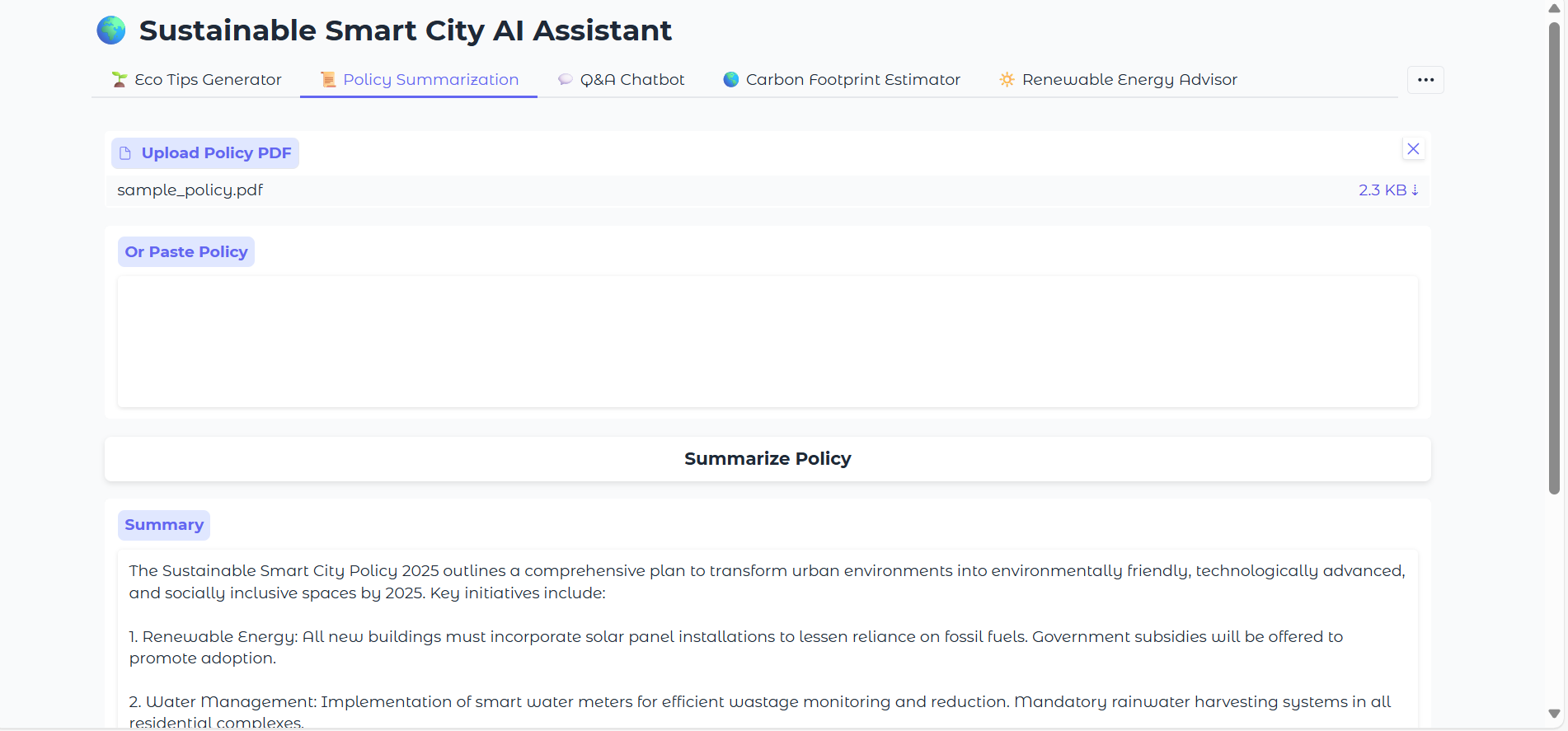
* HTTPS encryption for all communications
* User authentication and authorization
* Rate limiting to prevent abuse
* Regular security audits and updates

**14.OUTPUT OF THE APPLICATION:**

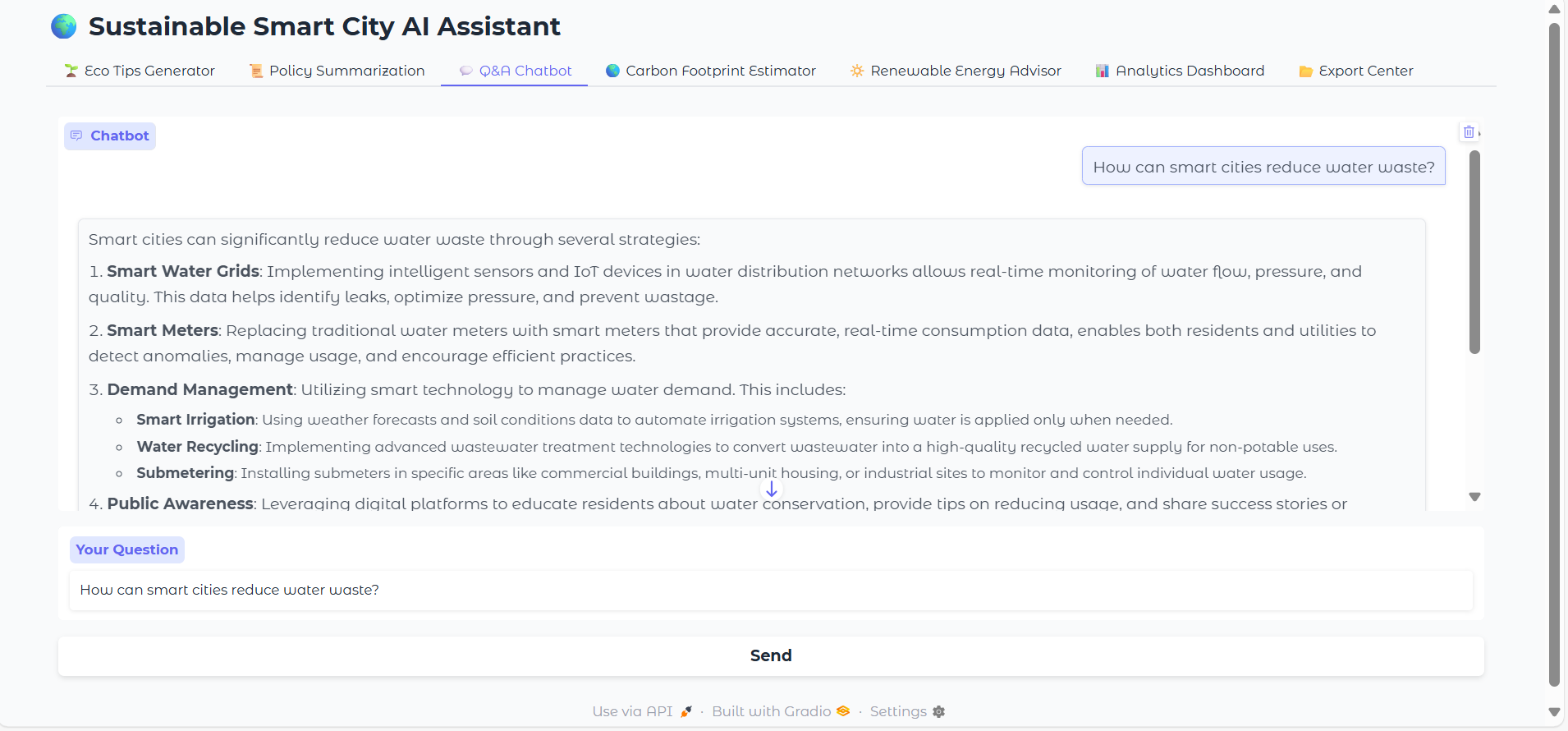
**Eco Tips Generator**

****

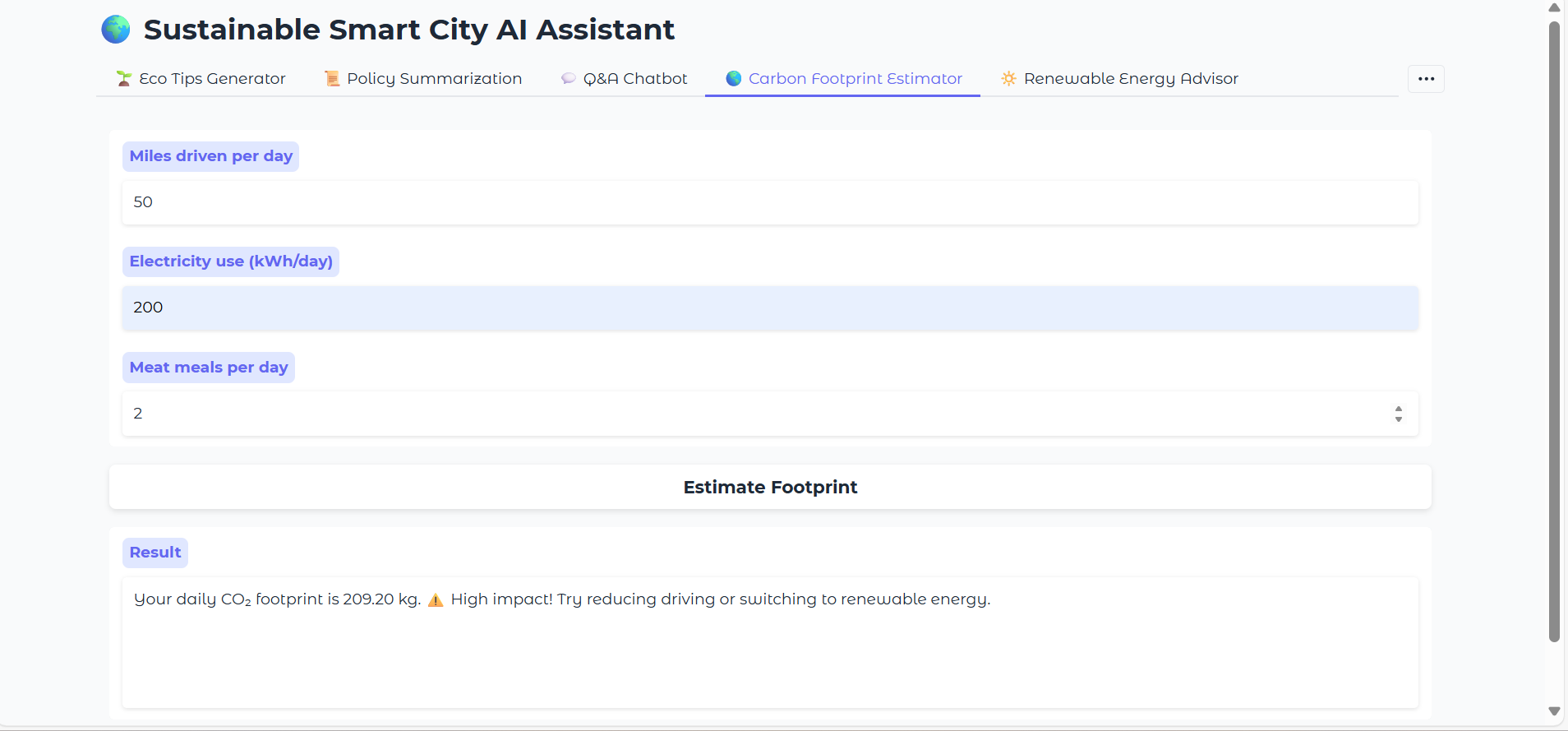
**Policy Summarization**

****

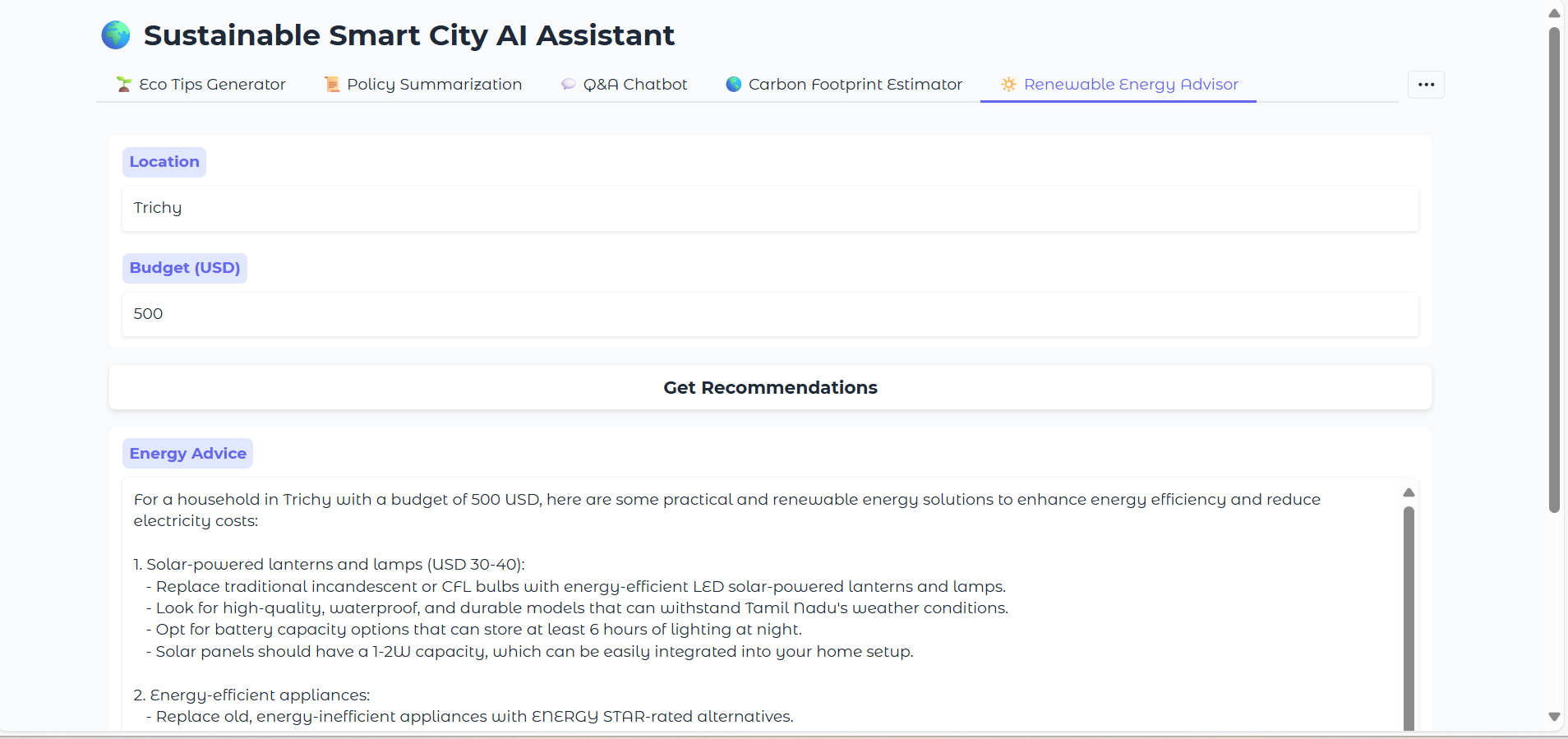
**Q&A Chatbot**

****

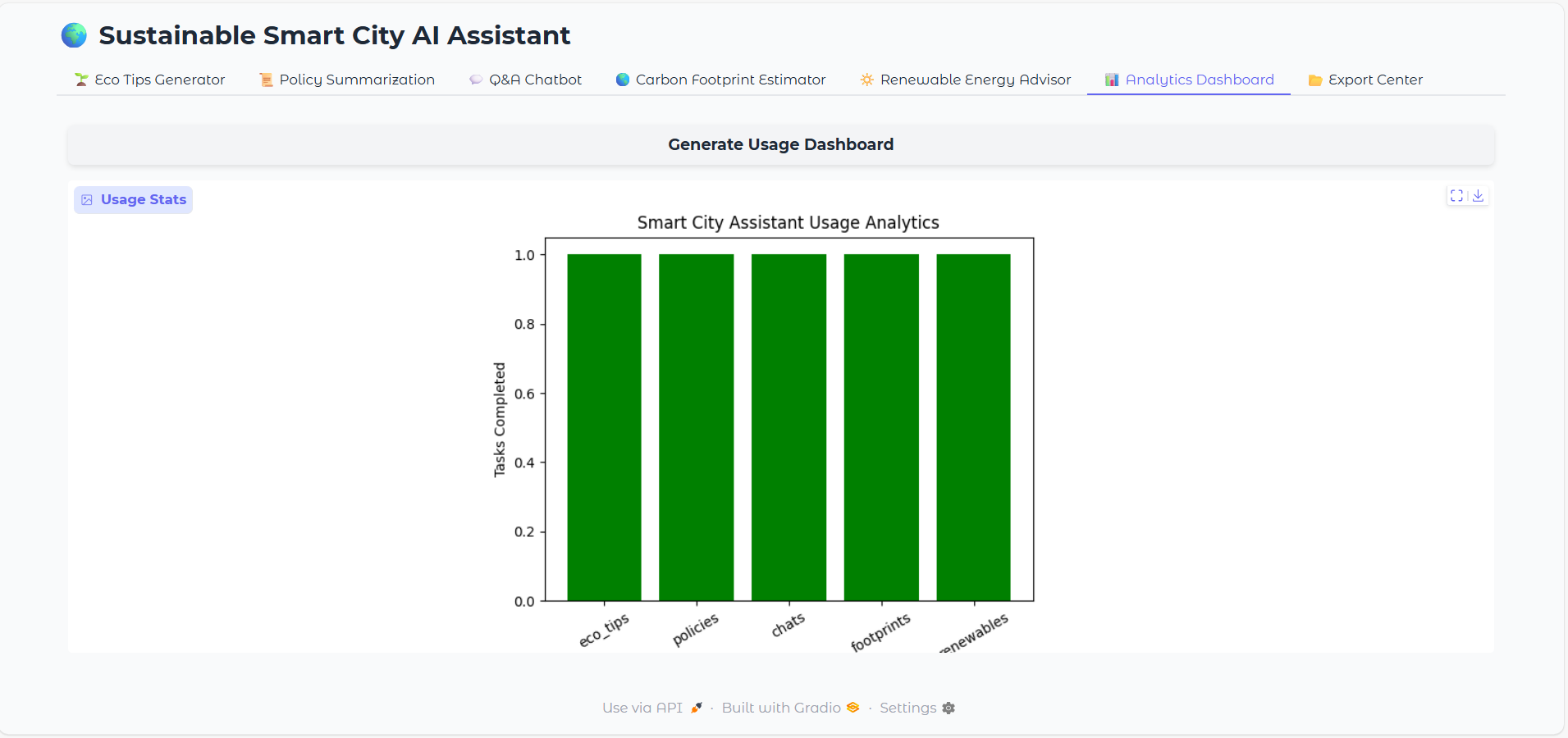
**Carbon Footprint Estimator**

****

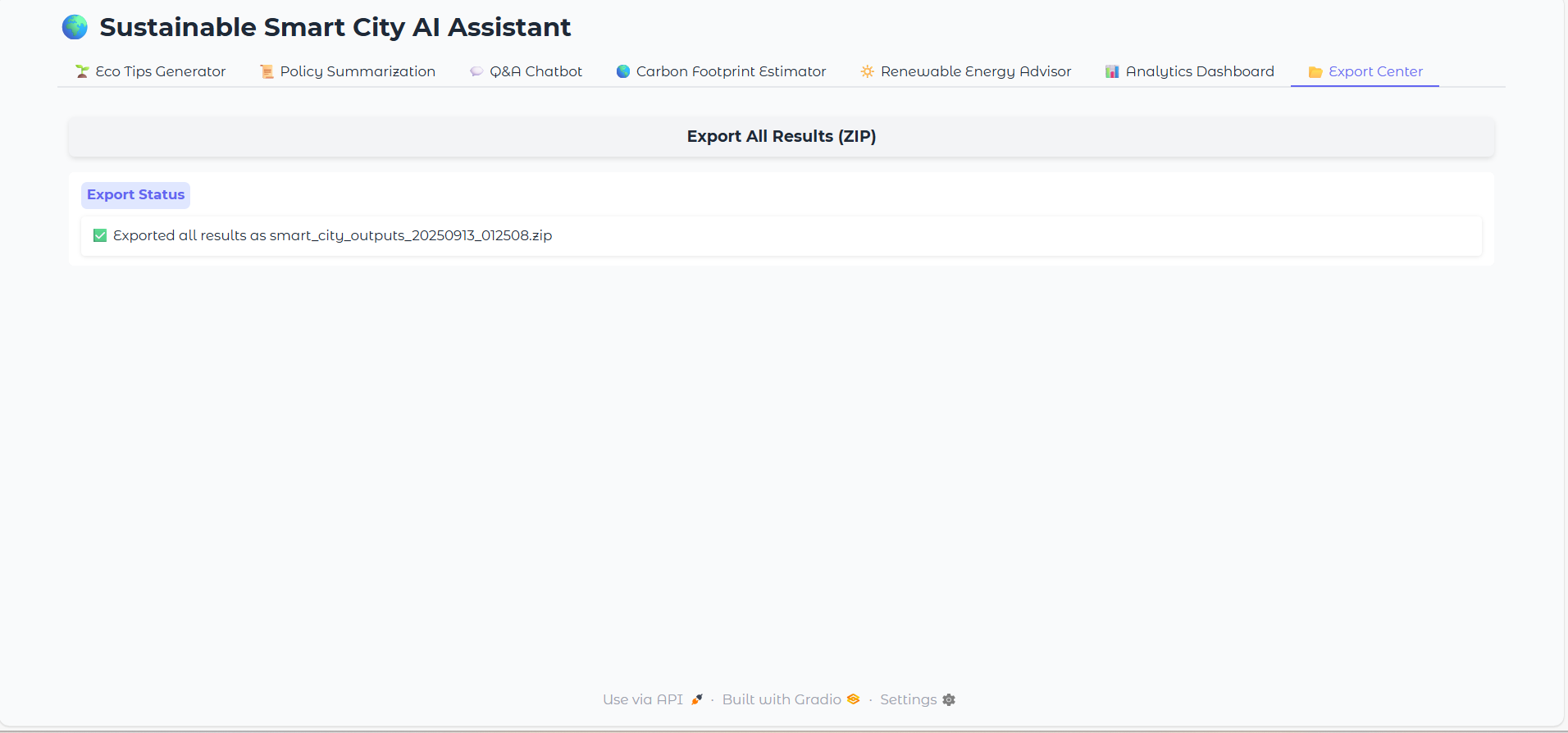
**Renewable Energy Advisor**

****

**Analytics Dashboard**

****

**Export Center**

****

**15. Conclusion**

The Sustainable Smart City Assistant represents a practical implementation of AI-powered environmental consultation. Built with modern tools and frameworks, it provides immediate value to users while maintaining simplicity and accessibility. The modular design allows for future enhancements and scalability as needs evolve.