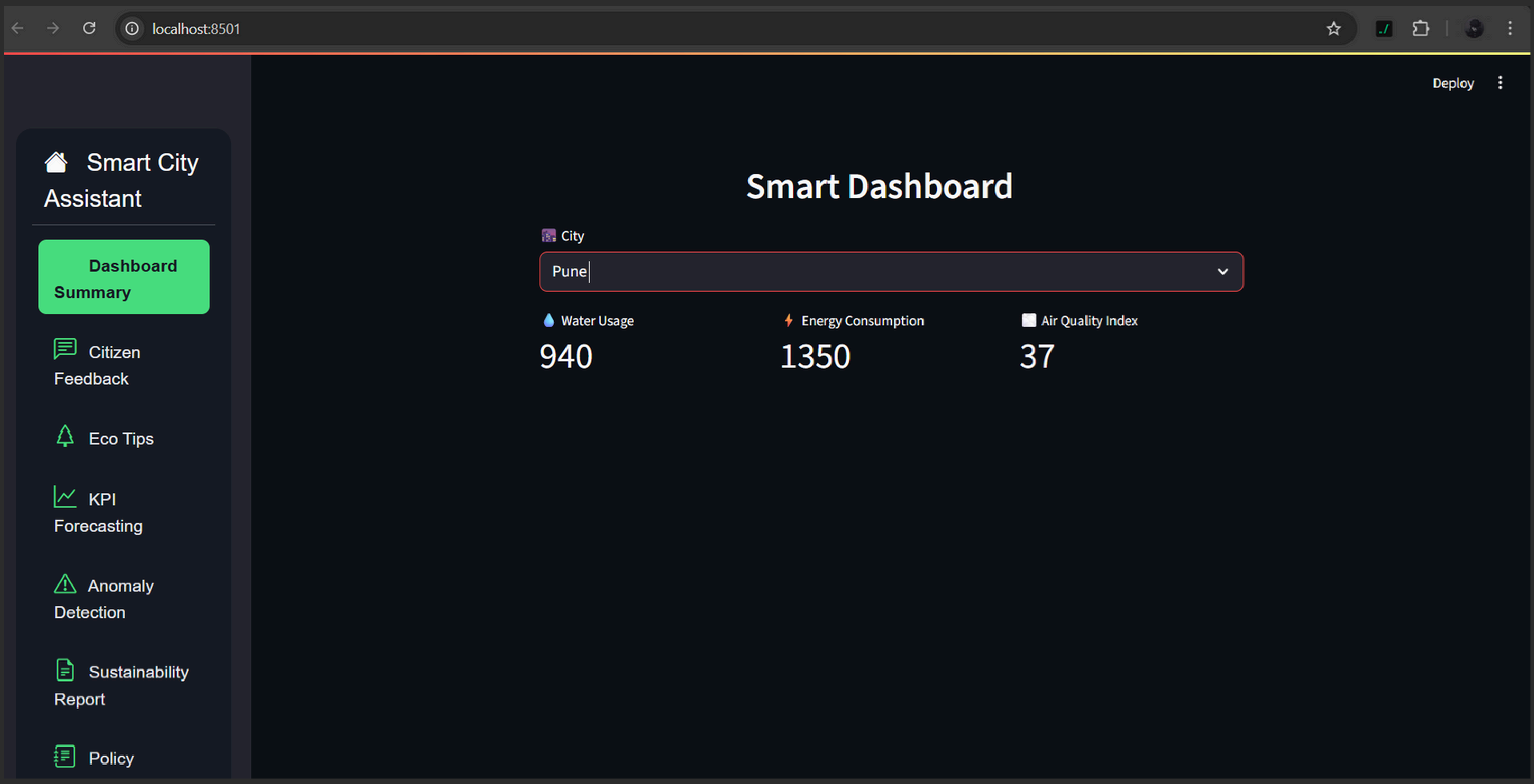
Smart City Assistant: Architecture and Module Breakdown

This document provides a comprehensive architecture overview for the Smart City Assistant project, detailing the interaction between the frontend (Streamlit) and backend (FastAPI) components. It outlines the functionality of each module/feature within the navigation, including data flow, AI integrations, and directory structure. The aim is to provide a clear understanding of the system's design and how each part contributes to the overall functionality.

˝I� ** Smart City Assistant Dashboard **



Smart Dashboard

City: Pune

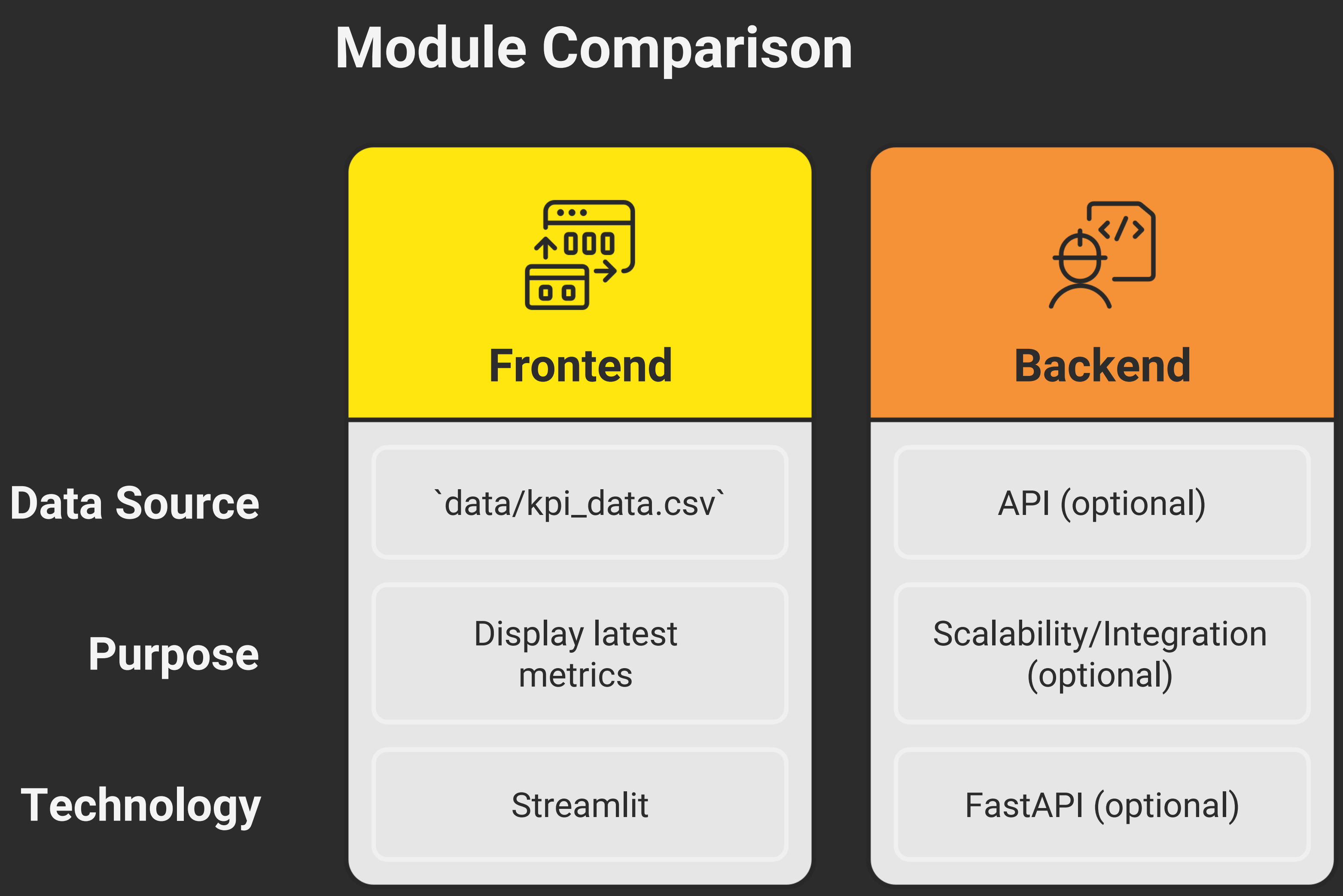
Water Usage	Energy Consumption	Air Quality Index
940	1350	37

```
graph TD
    subgraph Frontend (Streamlit)
        A[Sidebar Navigation]
        B[Dashboard Summary]
        C[Citizen Feedback]
        D[Eco Tips]
        E[KPI Forecasting]
        F[Anomaly Detection]
        G[Sustainability Report]
        H[Policy Summarizer]
        I[Chat Assistant]
    end

    subgraph Backend (FastAPI)
        J[API Routers]
        K[Services]
        L[Data Files/Storage]
        M[AI Integrations]
    end

    A --> B
    A --> C
    A --> D
    A --> E
    A --> F
    A --> G
    A --> H
    A --> I

    B -- reads --> L
    C -- POST/GET --> J
    D -- GET/POST --> J
    E -- reads --> L
    F -- reads --> L
    G -- POST --> M
    H -- POST --> M
    I -- POST --> J

    J -- business logic --> K
    K -- reads/writes --> L
    K -- calls --> M
    M -- LLM, Pinecone, Prophet, etc. --> M
```

ˇHι� ** Module/Feature Architecture Breakdown**

1. **Dashboard Summary**

• Frontend: This module reads KPI data from data/kpi_data.csv and displays the latest metrics for the selected city. The Streamlit interface presents this information in a user-friendly format, allowing for easy monitoring of key performance indicators.
• Backend: A backend is not strictly required for basic metrics display, as the frontend can directly read from the CSV file. However, if you need to serve the data via an API (e.g., for scalability or integration with other systems), a FastAPI endpoint can be implemented.



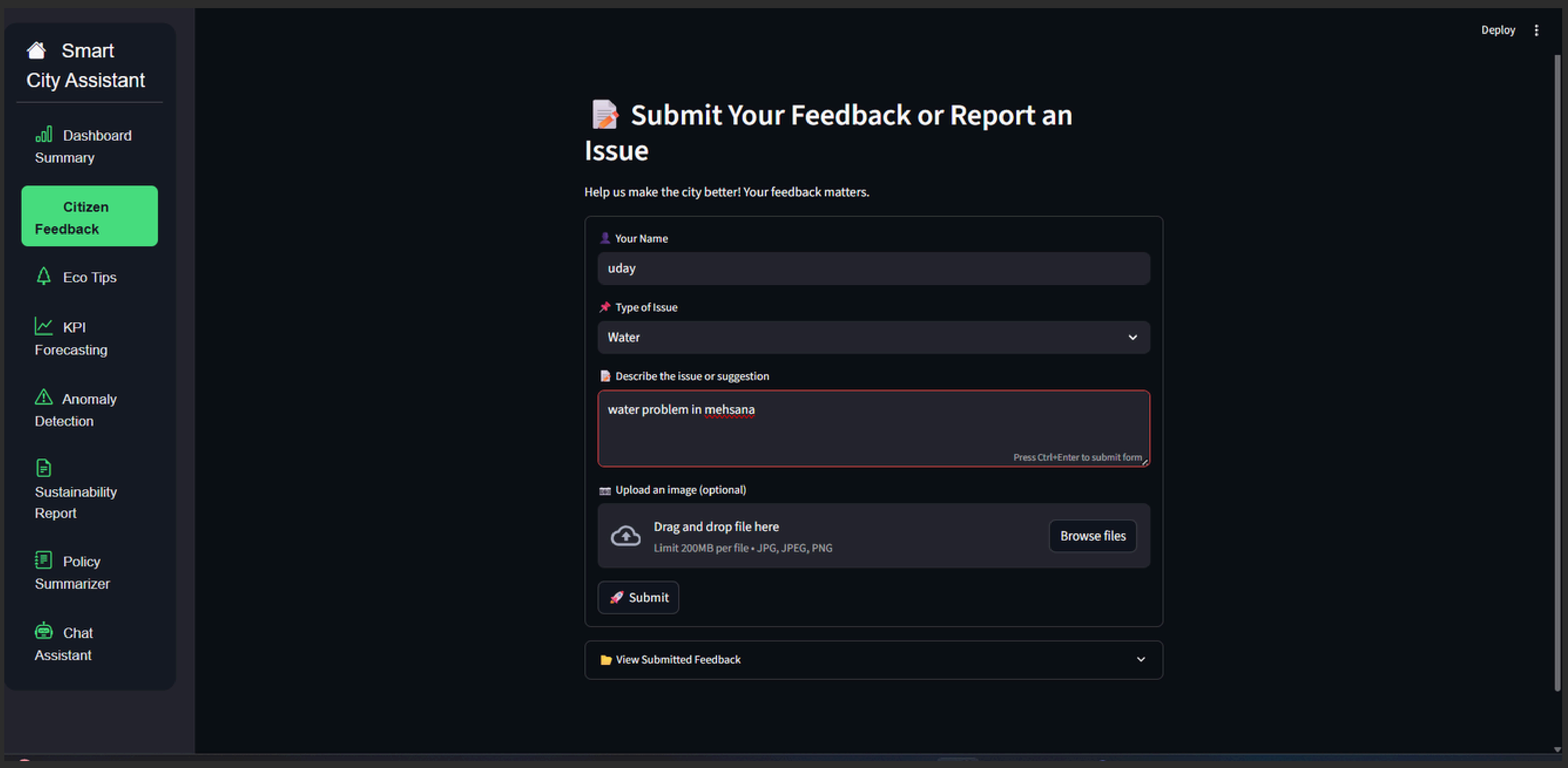
Module Comparison

	Frontend	Backend
Data Source	`data/kpi_data.csv`	API (optional)
Purpose	Display latest metrics	Scalability/Integration (optional)
Technology	Streamlit	FastAPI (optional)

2. **Citizen Feedback**

• Frontend: The frontend provides a form for citizens to submit feedback, including the option to upload images. The submitted data is saved to data/citizen_feedback.csv, and uploaded images are stored in the data/feedback_images/ directory.

• Backend: An optional FastAPI endpoint can be implemented to handle feedback submission and retrieval. This allows for more robust data management and can facilitate integration with other services. The API would handle POST requests for new feedback and GET requests for retrieving existing feedback.
• Data: The system uses a CSV file (data/citizen_feedback.csv) to store feedback data and a directory (data/feedback_images/) to store uploaded images.



Submit Your Feedback or Report an Issue

Help us make the city better! Your feedback matters.

Your Name: uday

Type of Issue: Water

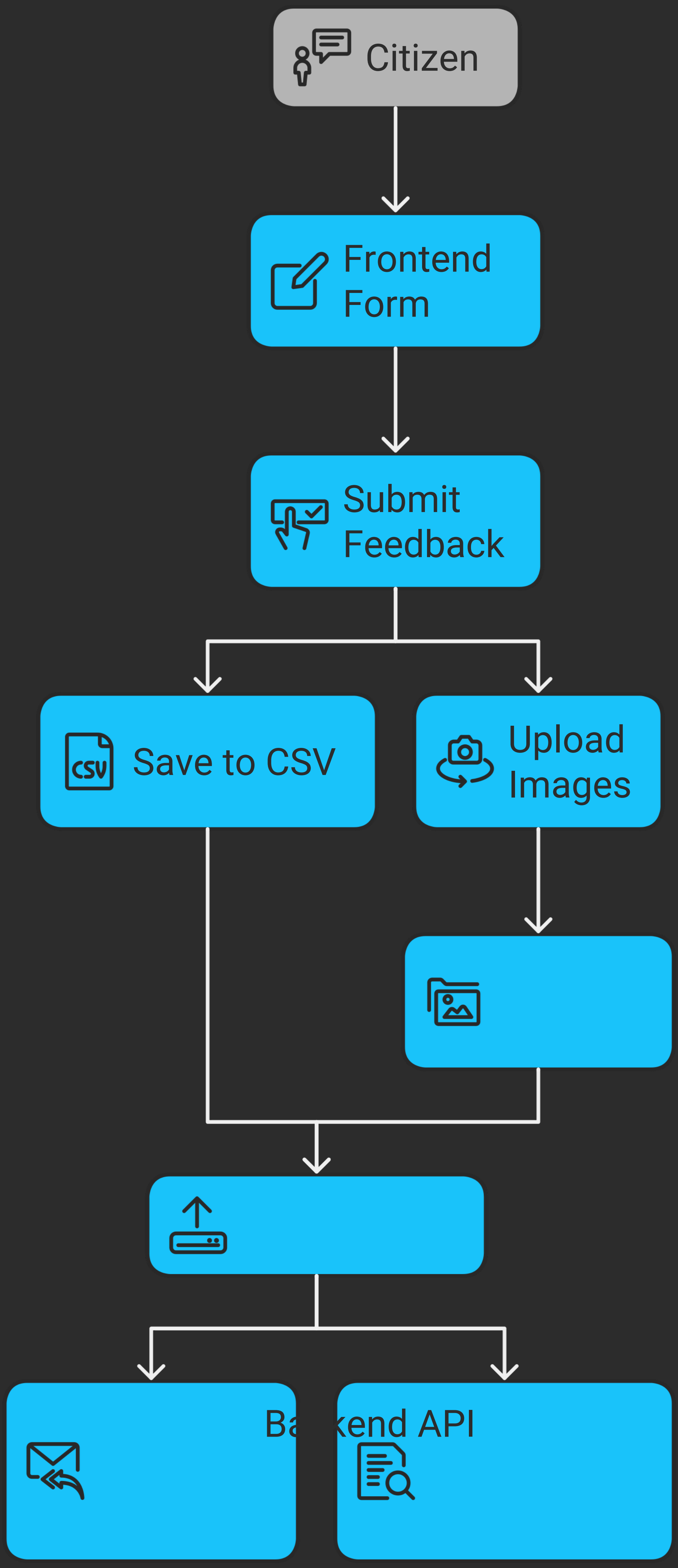
Describe the issue or suggestion: water problem in mehsana

Press Ctrl+Enter to submit form

Upload an image (optional)

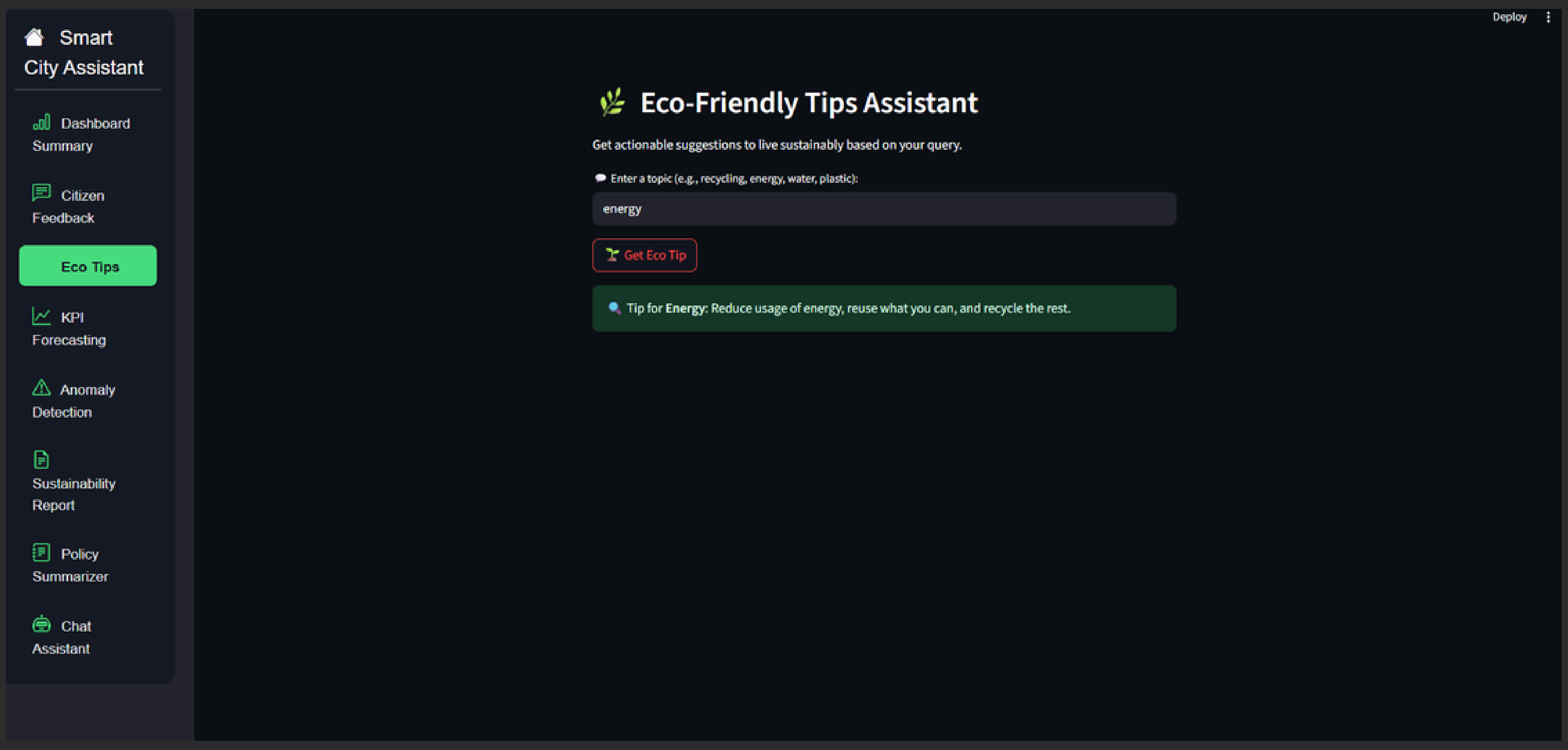
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Citizen Feedback System Flowchart



3. **Eco Tips**

- Frontend: This module includes a query box where users can search for eco tips. The tips are displayed either from a Pinecone vector database or a static JSON file.
- Backend: A FastAPI endpoint is used for semantic search using Pinecone or for serving static tips from a JSON file. This endpoint handles the logic for retrieving relevant eco tips based on user queries.
- AI: Pinecone is used for semantic search, allowing users to find relevant tips based on the meaning of their queries. Optionally, a Large Language Model (LLM) can be integrated to generate new eco tips.



Eco-Friendly Tips Assistant

Get actionable suggestions to live sustainably based on your query.

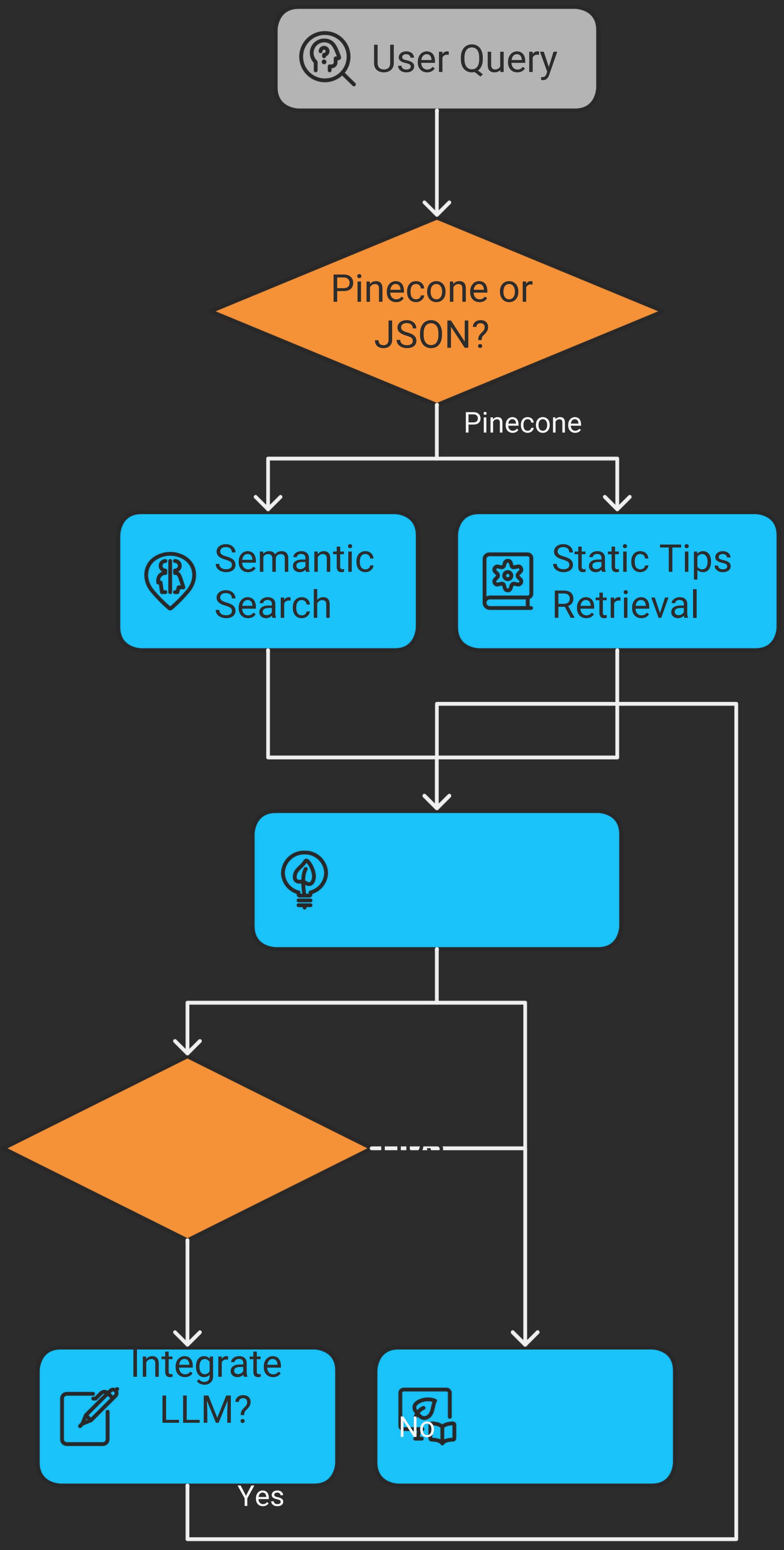
Enter a topic (e.g., recycling, energy, water, plastic):

energy

Get Eco Tip

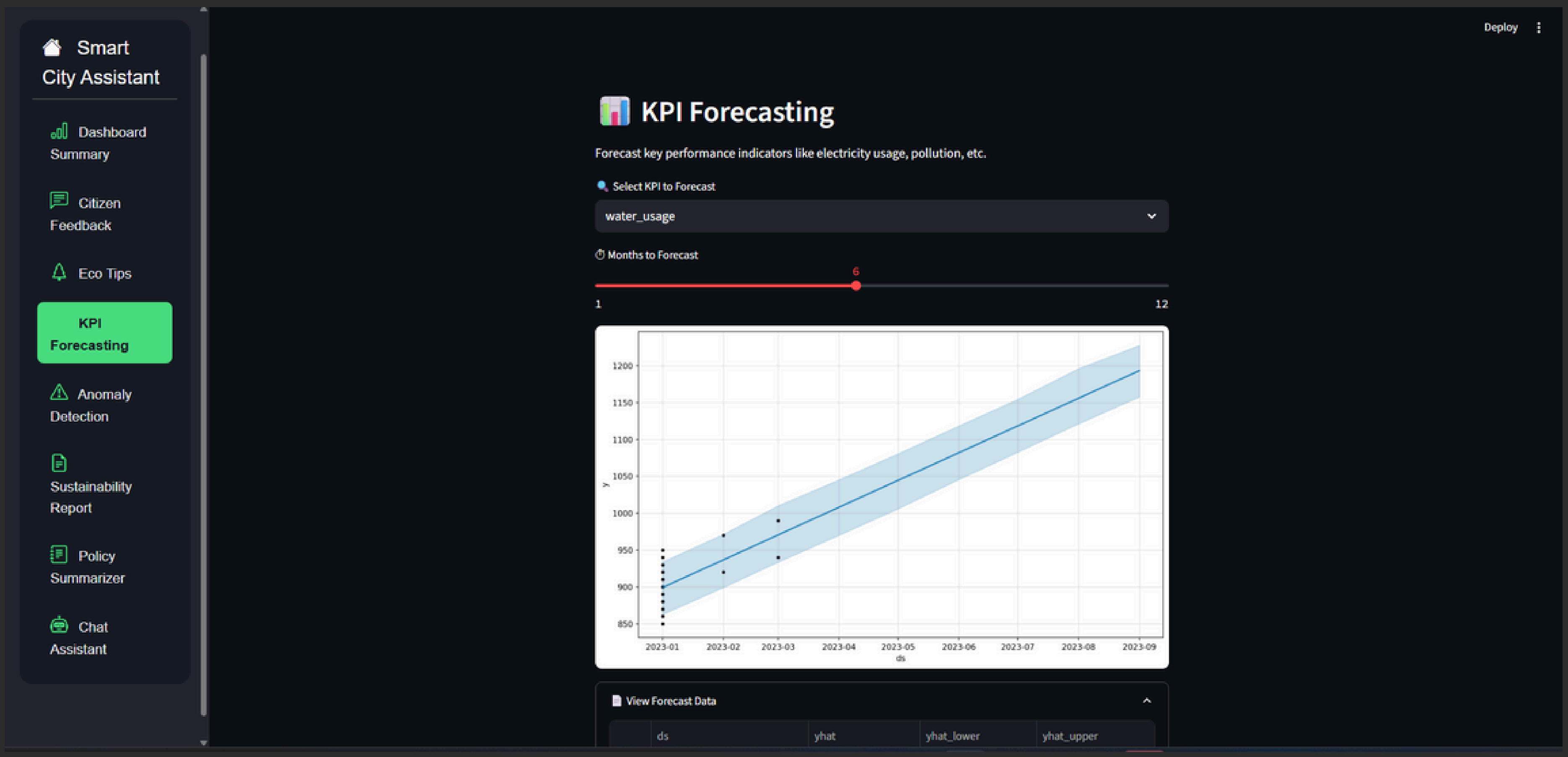
Tip for Energy: Reduce usage of energy, reuse what you can, and recycle the rest.

Eco Tips Retrieval and Generation System



4. **KPI Forecasting**

- Frontend: The frontend reads KPI data from data/kpi_data.csv and uses the Prophet library to forecast future KPI values. The forecasted data is then plotted and displayed to the user.
- Backend: A backend is not required unless you want to serve the forecasts via an API. If needed, a FastAPI endpoint can be created to generate and serve the forecasts.

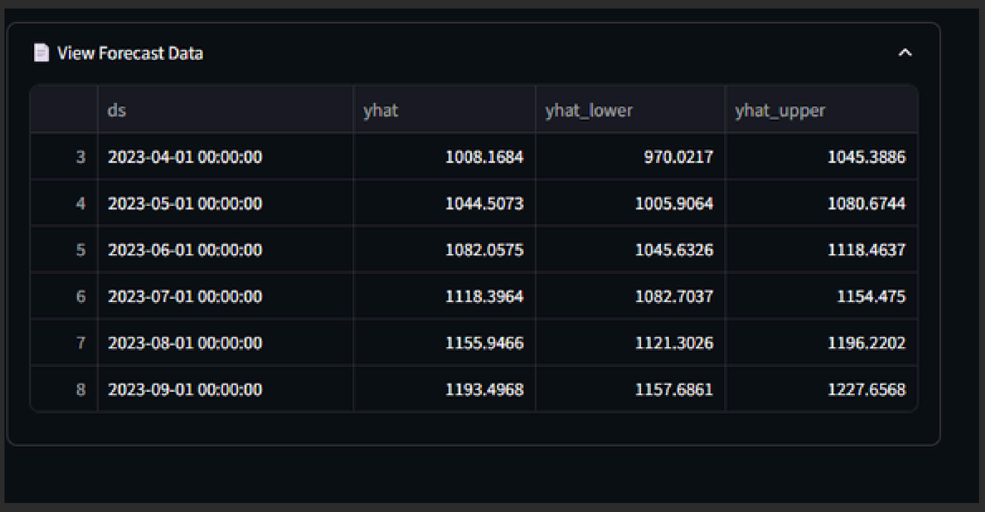


KPI Forecasting

Forecast key performance indicators like electricity usage, pollution, etc.

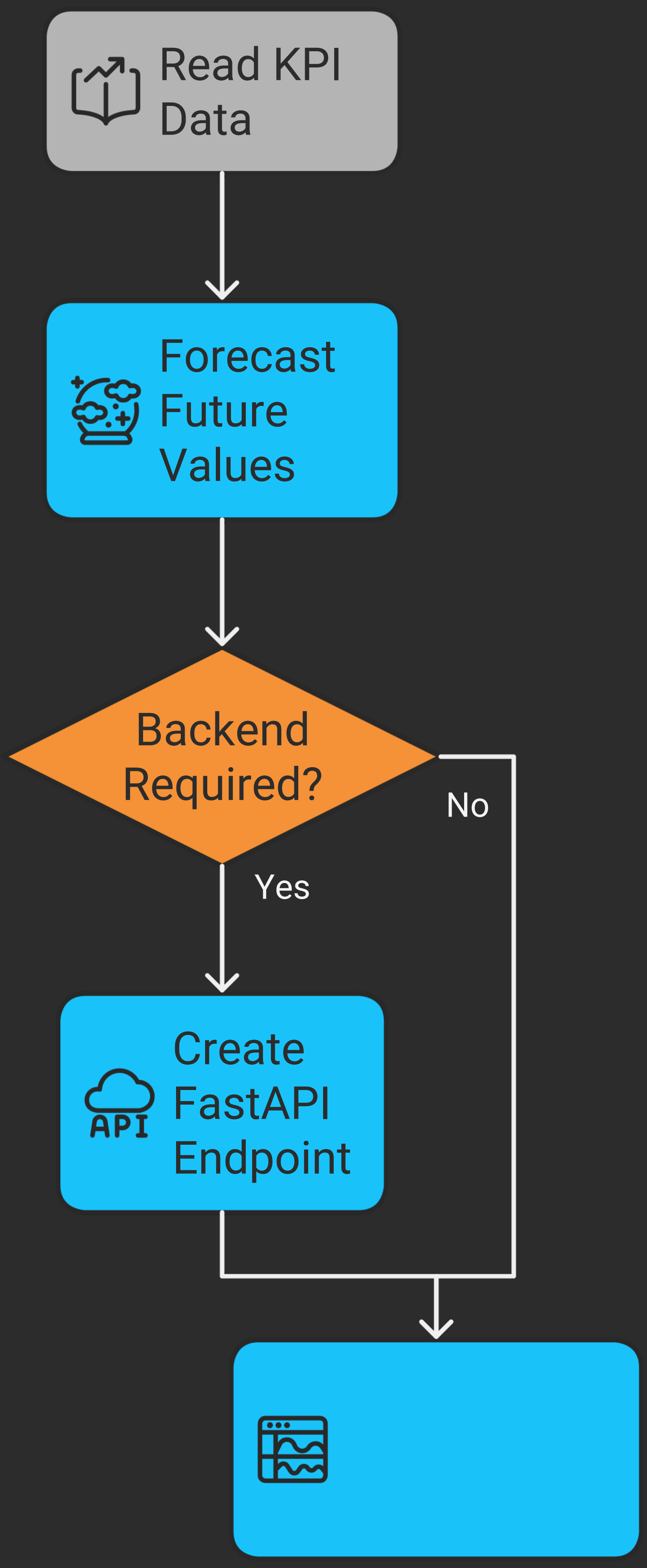
Select KPI to Forecast: water_usage

Months to Forecast: 6 (1–12)


View Forecast Data

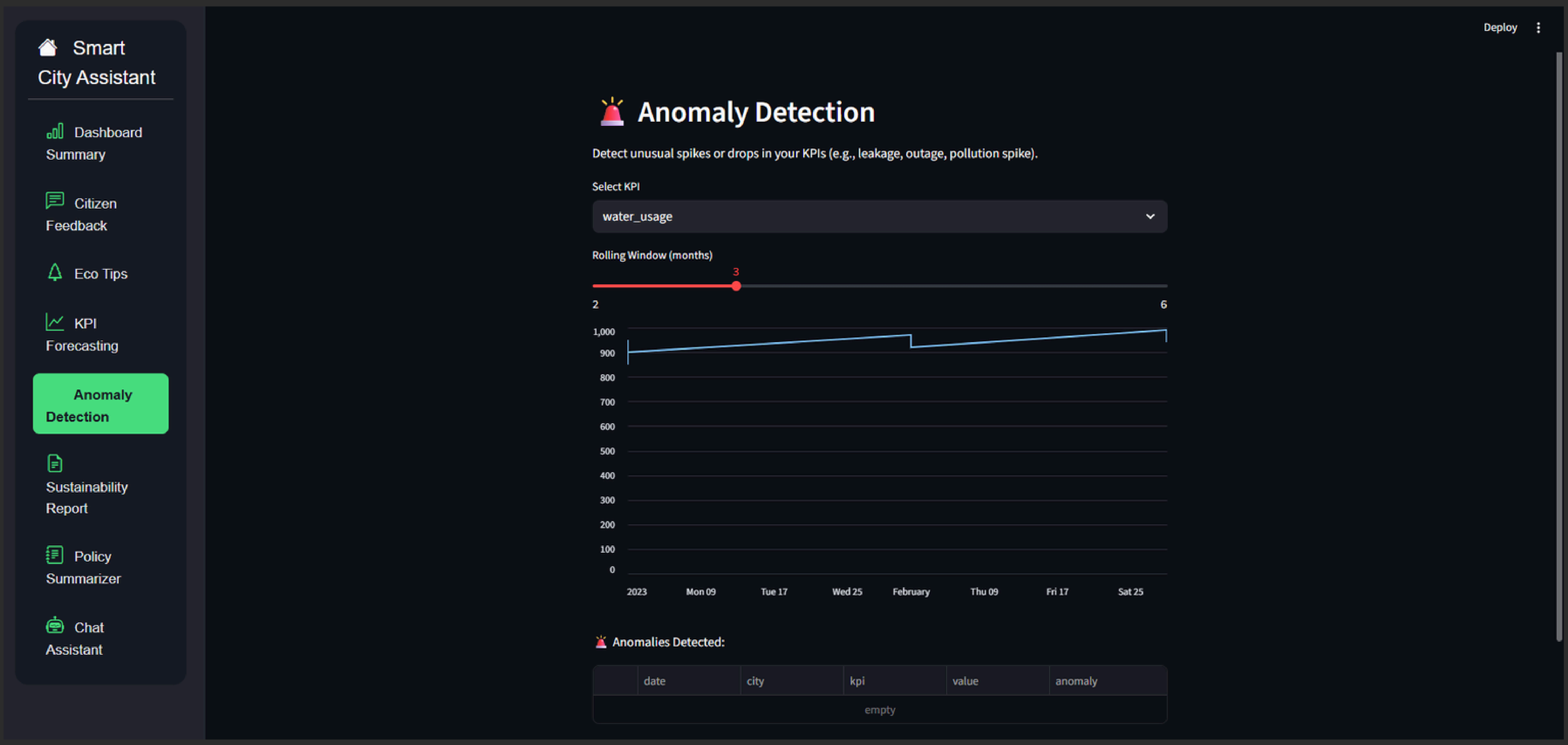
	ds	yhat	yhat_lower	yhat_upper
3	2023-04-01 00:00:00	1008.1684	970.0217	1045.3886
4	2023-05-01 00:00:00	1044.5073	1005.9064	1080.6744
5	2023-06-01 00:00:00	1082.0575	1045.6326	1118.4637
6	2023-07-01 00:00:00	1118.3964	1082.7037	1154.475
7	2023-08-01 00:00:00	1155.9466	1121.3026	1196.2202
8	2023-09-01 00:00:00	1193.4968	1157.6861	1227.6568

KPI Forecasting Process



5. **Anomaly Detection**

- Frontend: The frontend reads KPI data from data/kpi_data.csv and computes rolling statistics to identify anomalies. The detected anomalies are flagged and visualized in the user interface.
- Backend: A backend is not required unless you want to serve the anomaly detection results via an API. A FastAPI endpoint can be implemented to perform anomaly detection and serve the results.



🚨 Anomaly Detection

Detect unusual spikes or drops in your KPIs (e.g., leakage, outage, pollution spike).

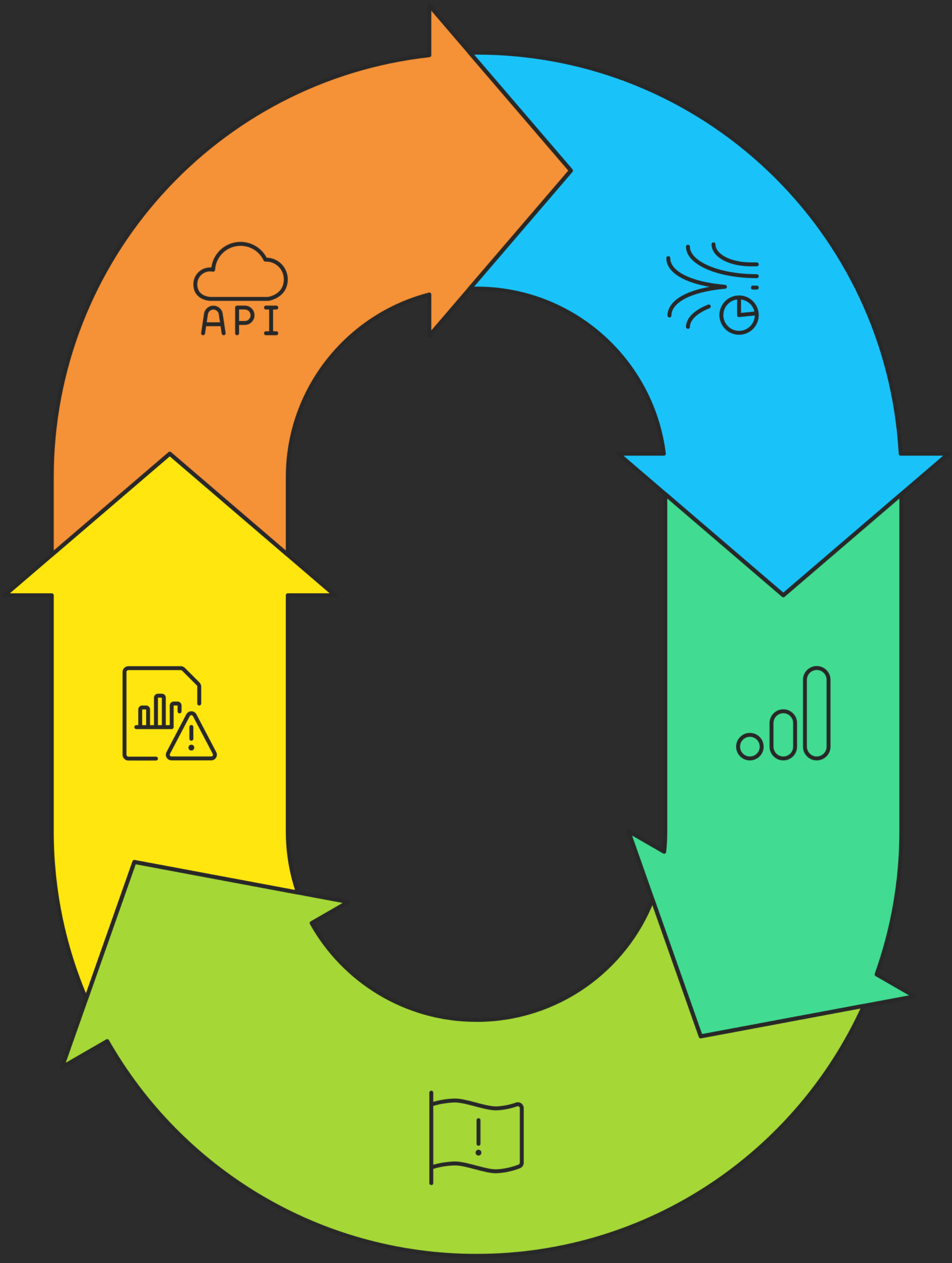
Select KPI: water_usage

Rolling Window (months): 3 (range 2–6)

🚨 Anomalies Detected:

	date	city	kpi	value	anomaly
empty					

Anomaly Detection Cycle



Serve Results via API

The backend serves anomaly detection results through an API.

Read KPI Data

The frontend reads KPI data from a CSV file.

Compute Rolling Statistics

The frontend computes rolling statistics to identify anomalies.

Visualize Anomalies

Anomalies are visualized in the user interface.

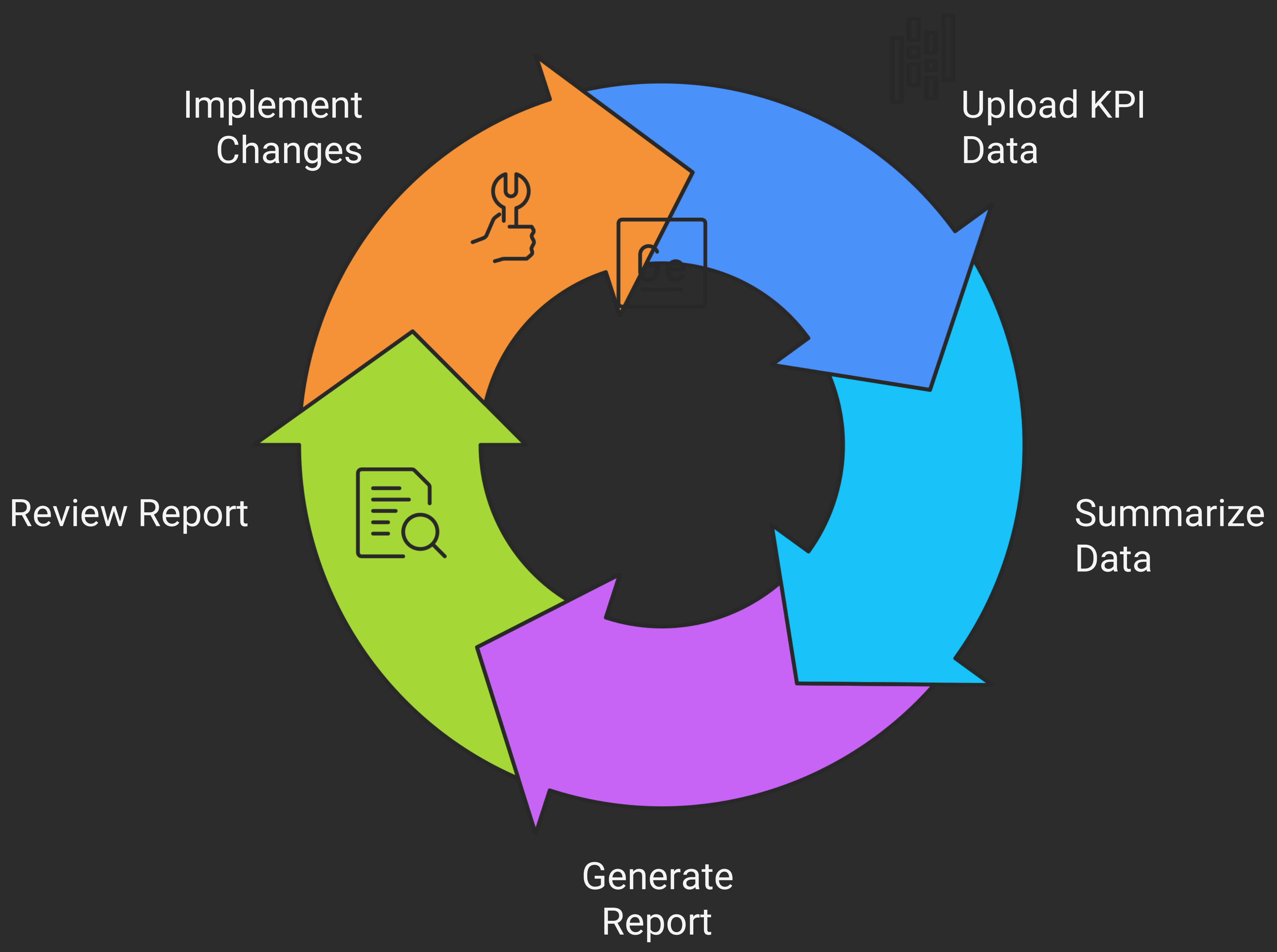
Flag Anomalies

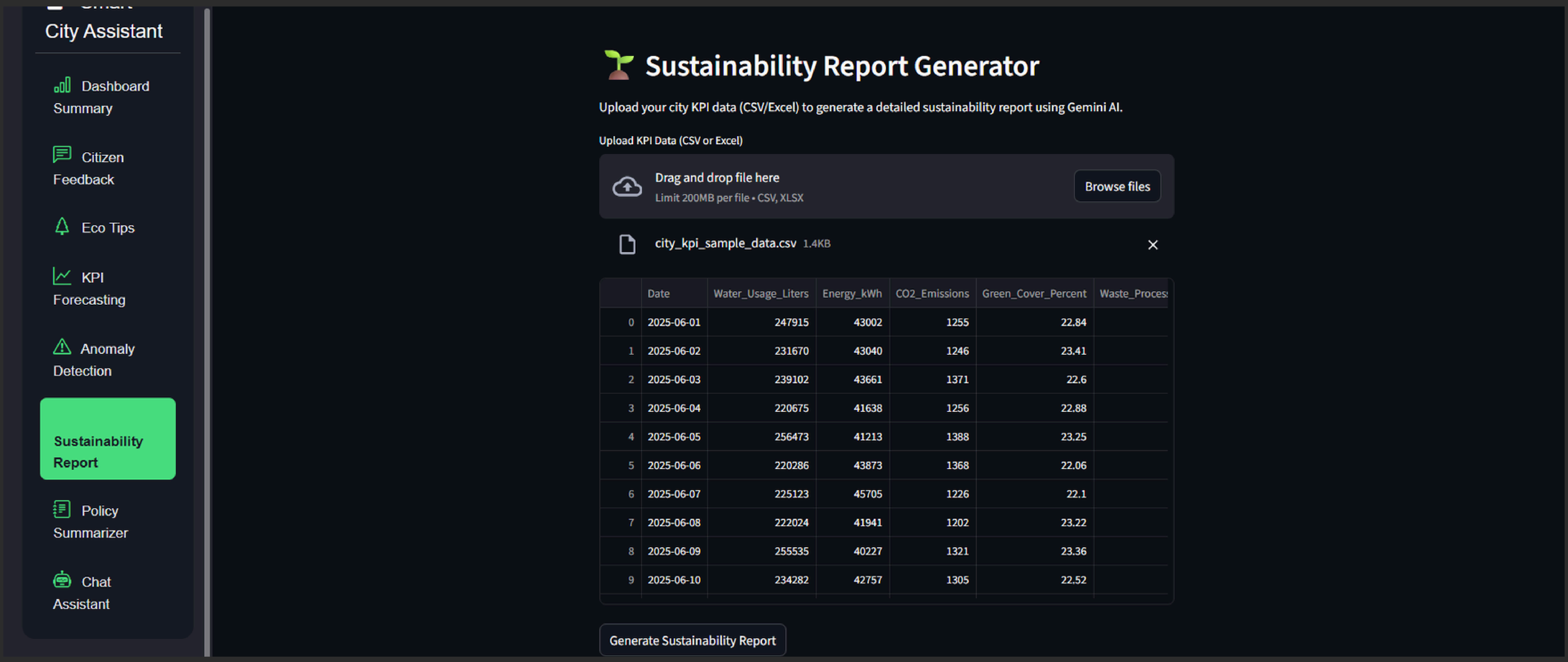
Anomalies are flagged for further analysis.

6. **Sustainability Report**

Sustainability Report Generation Cycle

- Frontend: This module allows users to upload KPI data, which is then summarized using pandas. The summarized data is sent to the Gemini API for report generation.
- Backend: An optional FastAPI endpoint can be implemented to generate reports via the Gemini API. This allows for more controlled access and management of the report generation process.





🌱 Sustainability Report Generator

Upload your city KPI data (CSV/Excel) to generate a detailed sustainability report using Gemini AI.

Upload KPI Data (CSV or Excel)

Drag and drop file here
Limit 200MB per file • CSV, XLSX

Browse files

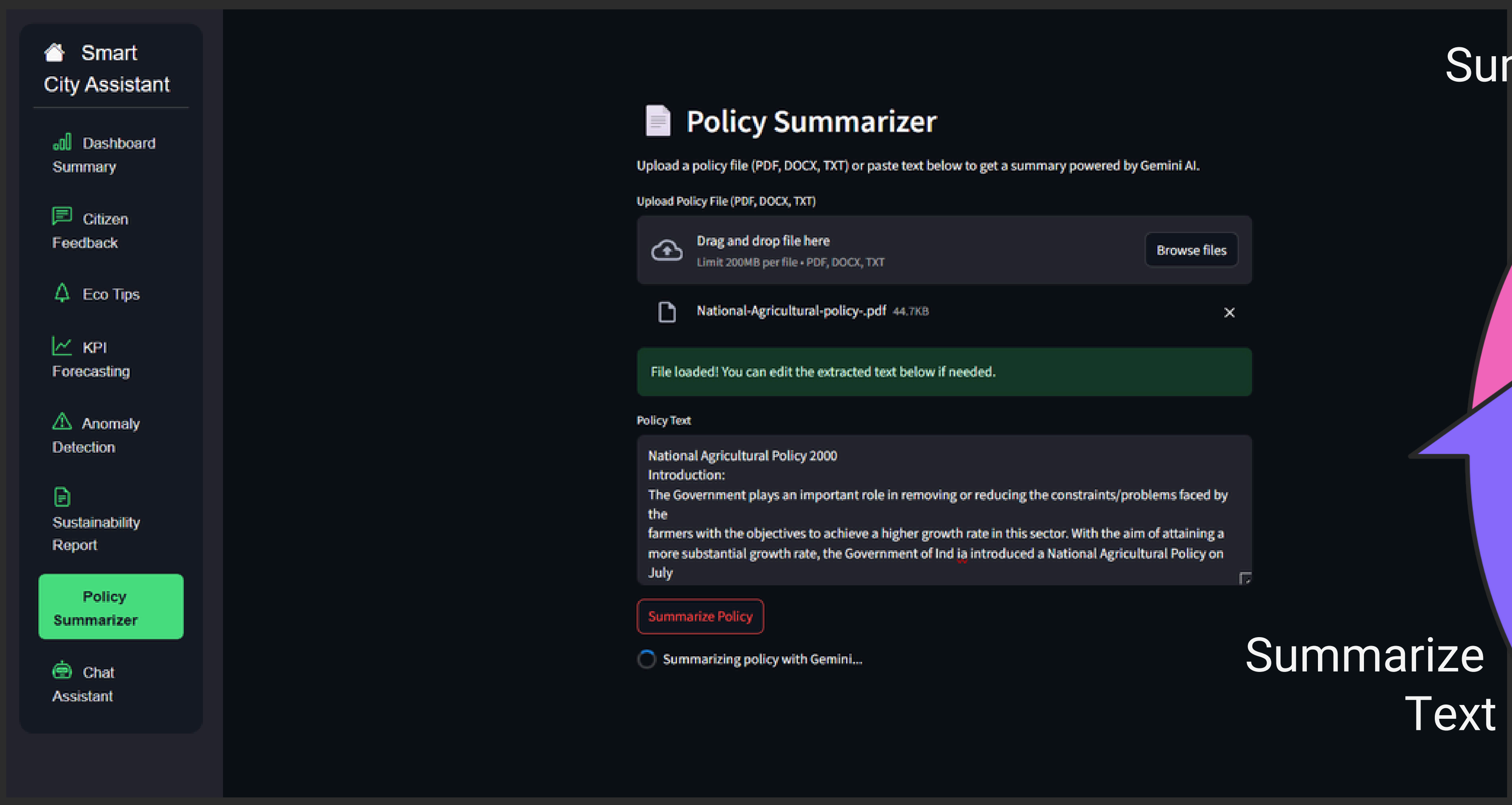
city_kpi_sample_data.csv 1.4KB

	Date	Water_Usage_Liters	Energy_kWh	CO2_Emissions	Green_Cover_Percent	Waste_Proces
0	2025-06-01	247915	43002	1255	22.84	
1	2025-06-02	231670	43040	1246	23.41	
2	2025-06-03	239102	43661	1371	22.6	
3	2025-06-04	220675	41638	1256	22.88	
4	2025-06-05	256473	41213	1388	23.25	
5	2025-06-06	220286	43873	1368	22.06	
6	2025-06-07	225123	45705	1226	22.1	
7	2025-06-08	222024	41941	1202	23.22	
8	2025-06-09	255535	40227	1321	23.36	
9	2025-06-10	234282	42757	1305	22.52	

Generate Sustainability Report

7. **Policy Summarizer**

• Frontend: Users can upload or paste policy text into this module. The text is extracted from the uploaded file (PDF, DOCX, or TXT) and sent to the Gemini API for summarization.
• Backend: An optional FastAPI endpoint can be implemented to handle the summarization process via the Gemini API. This provides a centralized point for managing policy summarization requests.



Policy Summarizer

Upload a policy file (PDF, DOCX, TXT) or paste text below to get a summary powered by Gemini AI.

Upload Policy File (PDF, DOCX, TXT)

Drag and drop file here
Limit 200MB per file • PDF, DOCX, TXT
Browse files

National-Agricultural-policy-.pdf 44.7KB

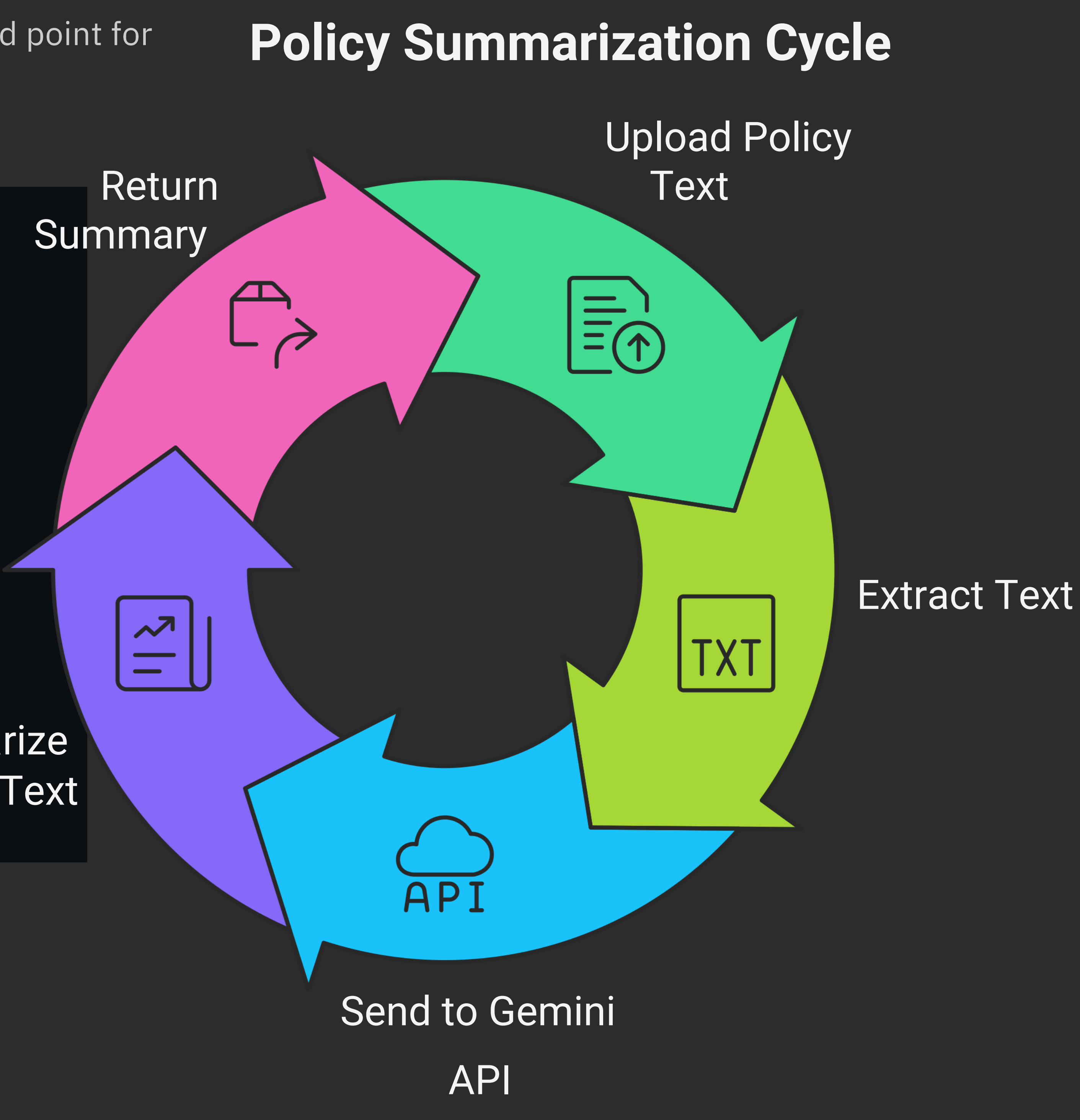
File loaded! You can edit the extracted text below if needed.

Policy Text

National Agricultural Policy 2000
Introduction:
The Government plays an important role in removing or reducing the constraints/problems faced by the
farmers with the objectives to achieve a higher growth rate in this sector. With the aim of attaining a more substantial growth rate, the Government of Ind ia introduced a National Agricultural Policy on July

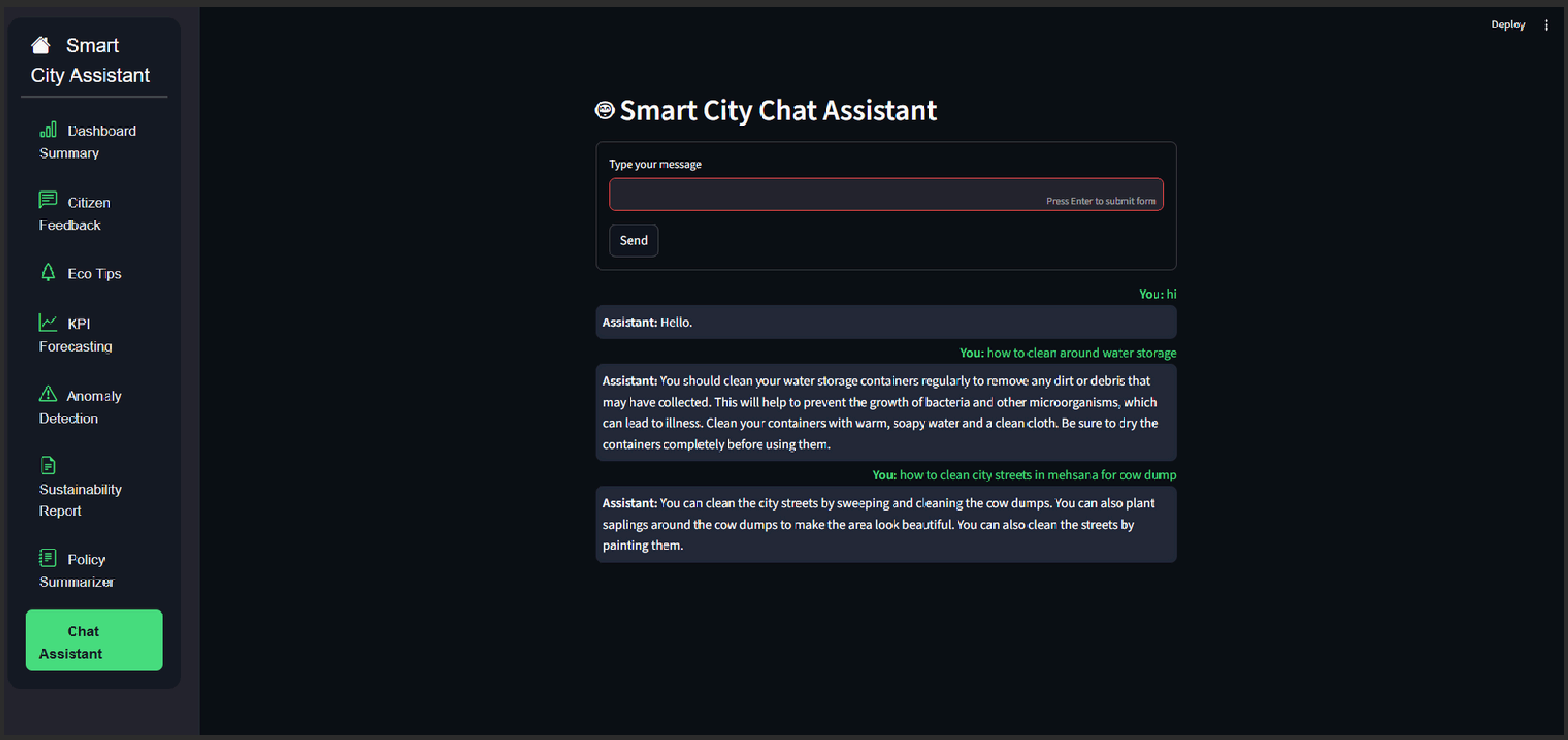
Summarize Policy

Summarizing policy with Gemini...



8. **Chat Assistant**

• Frontend: The frontend provides a chat user interface where users can input their queries. The user input and chat history are sent to the FastAPI backend.
• Backend: A FastAPI /chat endpoint handles the chat functionality. This endpoint calls an LLM (such as WatsonX or Gemini) to generate a response based on the user's input and chat history. The response is then returned to the frontend.



Smart City Chat Assistant

Type your message

Press Enter to submit form

Send

You: hi

Assistant: Hello.

You: how to clean around water storage

Assistant: You should clean your water storage containers regularly to remove any dirt or debris that may have collected. This will help to prevent the growth of bacteria and other microorganisms, which can lead to illness. Clean your containers with warm, soapy water and a clean cloth. Be sure to dry the containers completely before using them.

You: how to clean city streets in mehsana for cow dump

Assistant: You can clean the city streets by sweeping and cleaning the cow dumps. You can also plant saplings around the cow dumps to make the area look beautiful. You can also clean the streets by painting them.

�� ** Data Flow Summary**

• The Frontend primarily handles the user interface, file uploads, and local CSV reading for most analytics tasks.
• The Backend manages AI-powered features, the feedback API, chat functionality, and semantic search.

- AI/ML Services (Gemini, Pinecone, Prophet) are called from the backend or frontend as needed to perform specific tasks.

�� ** Directory Structure (Simplified)**

```
SmartCity Assistant/
 │
 ├── app/
 │  ├──api/      # FastAPI routers (chat, feedback, eco tips, etc.)
 │  ├──services/   # Business logic, LLM, Pinecone, etc.
 │  ├──core/     # Config, settings
 │  └──utils/     # Utilities
 │
 ├── frontend/
 │  ├──smart_dashboard.py #MainStreamlitapp
 │  └──...(otherUImodules)
 │
 ├── data/
 │  ├──kpi_data.csv
 │  ├──citizen_feedback.csv
 │  ├──eco_tips.json
 │  └──feedback_images/
 │
```