**Milestone 8: Sustainability Report Generation**

**🎯 Objective:**

Use **IBM Watsonx Granite LLM** to generate an **AI-powered city-wise sustainability report** based on uploaded KPI data.

**✅ Step 1: Backend Service — Generate Report Using LLM**

**📄 services/report\_generator.py**

python

CopyEdit

from services.granite\_llm import ask\_granite

def generate\_city\_report(kpi\_data: str, city\_name: str):

prompt = f"""

Generate a detailed sustainability report for the city of {city\_name}.

Use the following KPI data:

{kpi\_data}

Include the following:

- Overview of city performance

- Observed trends in water, air, energy

- Recommendations for improvement

- Short conclusion and score out of 10

"""

response = ask\_granite(prompt)

return response

**✅ Step 2: FastAPI Endpoint for Report Generation**

**📄 routers/report\_router.py**

python

CopyEdit

from fastapi import APIRouter

from pydantic import BaseModel

from services.report\_generator import generate\_city\_report

router = APIRouter()

class ReportRequest(BaseModel):

city: str

kpi\_data: str

@router.post("/generate-report")

def create\_sustainability\_report(req: ReportRequest):

report = generate\_city\_report(req.kpi\_data, req.city)

return {"report": report}

**✅ Step 3: Streamlit Frontend Component**

**📄 frontend/components/report\_generator.py**

python

CopyEdit

import streamlit as st

from utils.api\_utils import post\_request

def generate\_report\_ui():

st.subheader("📘 Generate Sustainability Report")

city = st.text\_input("Enter City Name")

kpi\_data = st.text\_area("Paste KPI data (e.g. water, air, energy usage over time):")

if st.button("Generate Report"):

if not city or not kpi\_data:

st.error("Please provide both city name and KPI data.")

return

result = post\_request("/generate-report", {"city": city, "kpi\_data": kpi\_data})

st.success("📝 AI-Generated Report:")

st.markdown(result["report"])

**✅ Optional (Enhancements)**

* **🔒 Save report as .txt or .pdf**
* **📊 Use data from dashboard directly**
* **🖼️ Add auto-score badges, charts, or sentiment bar**

**📌 Summary: Milestone 8 Deliverables**

| **Component** | **Description** | **Status** |
| --- | --- | --- |
| generate\_city\_report() | Calls Granite LLM with KPI + City | ✅ Done |
| /generate-report API | FastAPI POST endpoint | ✅ Done |
| Streamlit UI | City input + KPI text + output view | ✅ Done |
| Report style | Markdown format output | ✅ Done |

Would you like:

* The report saved/exported as .pdf?
* Automatic scoring with emoji or badge?

Let me know, and I’ll help you enhance it!

**You said:**

Activity 8.1: Prompt Engineering report\_generator.py uses a custom prompt to generate an AI-written report from KPI inputs Activity 8.2: Display/Download Render AI report on frontend Optionally provide markdown/PDF outpu

**ChatGPT said:**

Here's the complete implementation for:

**✅ Activity 8.1 & 8.2: Prompt Engineering + Display/Download**

**🎯 Goal:**

* **Generate an AI-written sustainability report** from KPI data using a **custom prompt**
* **Display** it nicely in Streamlit
* Provide **optional download** as Markdown or PDF

**✅ Activity 8.1: Prompt Engineering (LLM Integration)**

**📄 services/report\_generator.py**

python

CopyEdit

from services.granite\_llm import ask\_granite

def generate\_city\_report(kpi\_data: str, city\_name: str):

prompt = f"""

You are an expert in urban sustainability analytics. Your task is to write a detailed report for the city of {city\_name}, using the provided KPI data below.

KPI DATA:

{kpi\_data}

Please structure your report with the following sections:

1. Overview of the City's Sustainability Status

2. Observed Trends (e.g., air quality, energy consumption, water usage)

3. Anomalies or Concerns

4. Recommendations for Improvement

5. Final Evaluation and Sustainability Score (out of 10)

Keep the tone professional but readable. Format the output in clean Markdown.

"""

return ask\_granite(prompt)

**✅ Activity 8.2: Frontend Display + Download Option**

**📄 frontend/components/report\_generator.py**

python

CopyEdit

import streamlit as st

from utils.api\_utils import post\_request

from io import BytesIO

def generate\_report\_ui():

st.subheader("📘 Generate Sustainability Report")

city = st.text\_input("🌆 Enter City Name")

kpi\_data = st.text\_area("📊 Paste KPI data (e.g., air quality, energy usage):", height=200)

if st.button("Generate Report"):

if not city or not kpi\_data:

st.error("Please fill in both the city and KPI data.")

return

result = post\_request("/generate-report", {"city": city, "kpi\_data": kpi\_data})

report = result.get("report", "Error generating report.")

st.markdown("### 📝 AI-Generated Report:")

st.markdown(report, unsafe\_allow\_html=True)

# Download as Markdown

md\_bytes = BytesIO(report.encode("utf-8"))

st.download\_button(

label="⬇️ Download as Markdown",

data=md\_bytes,

file\_name=f"{city.lower().replace(' ', '\_')}\_sustainability\_report.md",

mime="text/markdown"

)

# Optional: PDF Export

# Requires fpdf or reportlab — let me know if you'd like to implement it.

**📦 Optional: Add PDF Export**

If you'd like a **PDF version**, we can use:

bash

CopyEdit

pip install fpdf

And implement export using:

python

CopyEdit

from fpdf import FPDF

def generate\_pdf(report\_text: str) → BytesIO

Let me know if you'd like this.