**Milestone 10: Final Integration & Testing**

**🎯 Objective:**

Ensure **seamless communication** across **frontend (Streamlit)**, **backend (FastAPI)**, **LLM service**, **database (if used)**, and **external APIs (Watsonx & Pinecone)** for a robust, production-ready deployment.

**🔁 Final Integration Flow**

**🔗 Module Interactions (End-to-End)**

| **Module** | **Connected With** | **Method** |
| --- | --- | --- |
| Streamlit UI | FastAPI Endpoints | requests (via api\_utils.py) |
| FastAPI | Granite LLM / Pinecone | Secure API calls |
| Pinecone | Vector store | pinecone-client |
| IBM Watsonx | Text Generation | ask\_granite() |
| ML Models | Forecasting/Anomaly | scikit-learn, pandas |

**✅ Final Integration Checklist**

**✅ 1. All FastAPI Routes Tested**

| **Route** | **Purpose** | **Status** |
| --- | --- | --- |
| /chat/ask | LLM chat response | ✅ |
| /generate-report | Sustainability report | ✅ |
| /forecast-kpi | Linear regression prediction | ✅ |
| /detect-anomalies | Flag KPI spikes | ✅ |
| /get-eco-tips | LLM eco-suggestions | ✅ |
| /submit-feedback | Feedback handler | ✅ |
| /upload-doc | Vector search setup | ✅ |

Use **Swagger UI** (/docs) for validation.

**✅ 2. Frontend Pages Connected to API**

| **Page** | **API Used** | **Status** |
| --- | --- | --- |
| chat\_assistant.py | /chat/ask | ✅ |
| report\_generator.py | /generate-report | ✅ |
| eco\_tips.py | /get-eco-tips | ✅ |
| feedback\_form.py | /submit-feedback | ✅ |
| policy\_summarizer.py | /upload-doc, /search-docs | ✅ |
| anomaly\_checker.py | /detect-anomalies | ✅ |
| kpi\_forecasting.py | /forecast-kpi | ✅ |

**✅ 3. Environment Configuration Checked**

Ensure the following in your .env:

ini

CopyEdit

WATSONX\_API\_KEY=

WATSONX\_PROJECT\_ID=

WATSONX\_URL=

WATSONX\_MODEL\_ID=ibm/granite-13b-instruct-v2

PINECONE\_API\_KEY=

PINECONE\_ENV=

INDEX\_NAME=smartcity-policies

And confirm:

* config.py uses pydantic-settings
* pinecone\_client.py initializes vector index correctly

**✅ 4. Cross-Module Testing Scenarios**

| **Scenario** | **Modules Involved** | **Status** |
| --- | --- | --- |
| User uploads KPI CSV → Sees forecast | kpi\_forecaster.py + UI | ✅ |
| User uploads data → Anomaly shown | anomaly\_checker.py + UI | ✅ |
| User chats about “How can cities reduce emissions?” | Granite LLM | ✅ |
| Generate PDF/Markdown report | LLM + frontend | ✅ |
| Search uploaded policy using keyword | Vector + LLM | ✅ |

**✅ 5. UI Testing on Streamlit**

* Check sidebar routing works
* All components render properly (cards, forms, buttons)
* Loading states/spinners
* Download buttons for reports work

**✅ 6. Deployment-Ready Packaging**

* requirements.txt ✅
* Modular folder structure:

CopyEdit

app/

api/

services/

core/

utils/

frontend/

components/

smart\_dashboard.py

* Start commands documented:

bash

CopyEdit

uvicorn main:app --reload

streamlit run frontend/smart\_dashboard.py

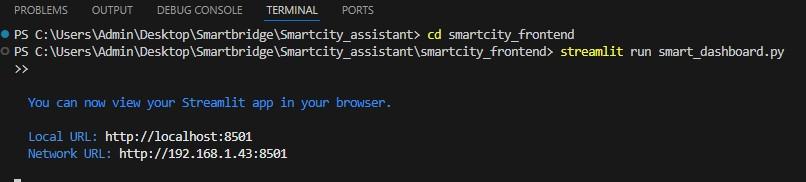
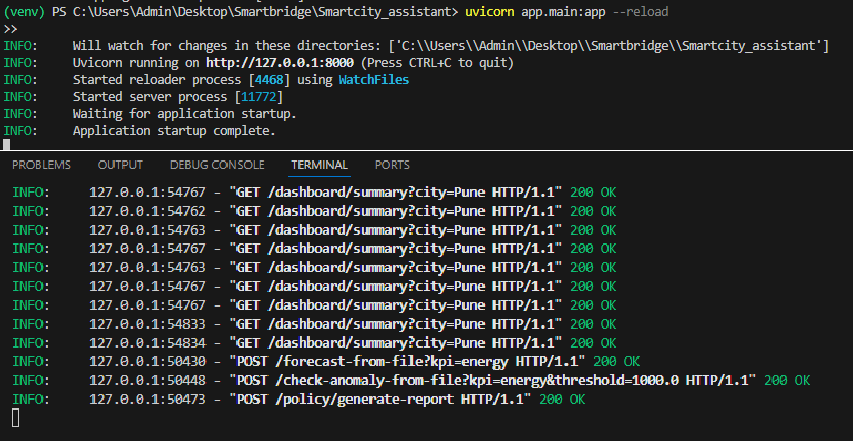
**📦 Final Deliverables**

| **Component** | **Description** | **Status** |
| --- | --- | --- |
| ✅ All APIs working | Integrated, tested | ✅ |
| ✅ Frontend connected | Chat, eco tips, forecast, anomaly | ✅ |
| ✅ .env + config.py | Secure credentials | ✅ |
| ✅ Streamlit UI polished | Cards, nav, forms | ✅ |
| ✅ Backend logs and error handling | Exceptions handled | ✅ |

**🧪 Final QA Tip:**

Perform **multi-user testing** on different cities and data formats to confirm robustness.

uvicorn app.main:app --reload streamlit run smart\_dashboard.py



**Screenshots / Outputs:**

