# Requirement Analysis – Structuring the Smart City AI Assistant

## 🧭 Customer Journey Map

Understanding the user’s experience from start to finish is critical. The **Customer Journey Map** highlights how a user interacts with the system and how their needs evolve at each touchpoint.

| **Step** | **User Action** | **User Experience** |
| --- | --- | --- |
| 1 | Uploads a policy PDF | 😕 Overwhelmed |
| 2 | Views summarized content | 🙂 Relieved |
| 3 | Uploads CSV file for forecasting | 😯 Curious |
| 4 | Sees anomaly flag or prediction | ⚠️ Alerted |
| 5 | Downloads AI-generated report | 😊 Informed & Empowered |

🔍 **Insight:** Clear, timely, and readable feedback at each step helps users stay engaged and make informed decisions.

## ⚙️ Solution Requirements

Based on the user experience and system goals, the project required a blend of natural language processing, machine learning, and interactive interfaces.

### ✅ Functional Requirements

* Accept text, PDF, and CSV inputs from users
* Summarize large policy documents using an LLM
* Forecast trends using regression on CSV input
* Detect anomalies in data based on threshold logic
* Generate PDF reports from text or derived insights
* Collect user feedback with session-based memory

### 🧪 Non-Functional Requirements

* Fast response time (under 10 seconds per interaction)
* Session-based operation (no login or database)
* Lightweight UI usable in a browser (Gradio)
* Compatible with Google Colab for free GPU access

## 🔄 Data Flow Diagram

A simplified Data Flow Diagram (DFD) helps visualize how data moves through the system:

User Inputs (Text / PDF / CSV)  
 ↓  
 Preprocessing Modules  
 ↓  
LLM or ML Inference Engine  
 ↓  
 Output Rendering (Text / Graph / Table)  
 ↓  
 Gradio UI & Optional PDF Export

**Flow Description:**

* PDF and text inputs are routed to the LLM for summarization or tip generation
* CSV files trigger forecasting and anomaly detection modules
* Outputs are passed through renderers to return readable text, tables, or downloadable reports

## 🧰 Technology Stack

This project leverages open-source tools and modern ML libraries to ensure high performance with minimal infrastructure.

| **Layer** | **Technology Used** |
| --- | --- |
| Frontend | Gradio UI |
| Backend | Python (FastAPI-style functions) |
| AI Models | IBM Granite |
| ML Libraries | scikit-learn (forecasting), pandas |
| PDF Tools | PyMuPDF (reading), FPDF (writing) |
| Hosting | Google Colab |

🔧 **Design Decision:** Google Colab was chosen to allow GPU access without the need for backend servers or complex deployment, keeping the tool accessible and free.

📌 *This phase ensured that the technical backbone of the Smart City AI Assistant aligned perfectly with the user needs discovered during the ideation stage.*