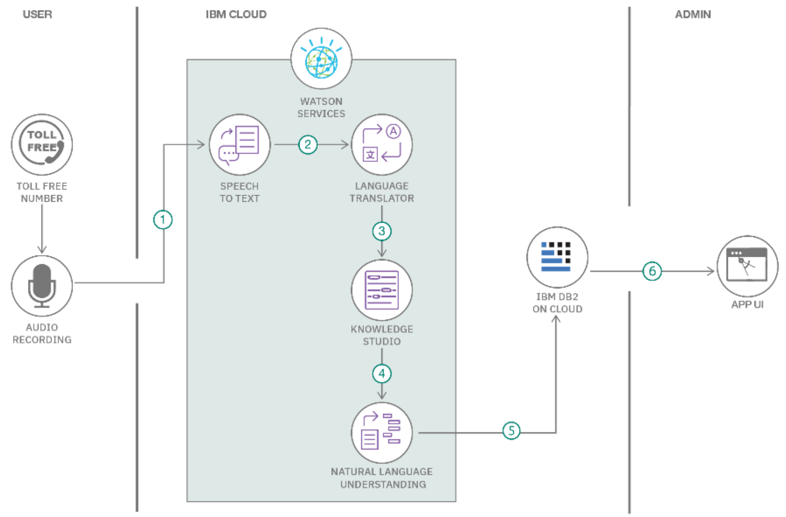
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	28 June 2025
Team ID	LTVIP2025TMID60548
Project Name	**Sustainable Smart City Assistant Using IBM Granite LLM**
Maximum Marks	4 Marks

Technical Architecture:

The solution uses a modular architecture combining AI services, ML models, vector search, and a dual-layer application stack. The frontend is built with **Streamlit** for a dynamic dashboard interface, while the backend uses **FastAPI** to handle routing, file processing, and LLM communication. **IBM Watsonx Granite LLM** is used for summarization, chat, eco tips, and report generation, while **Pinecone** powers semantic search. ML models using **scikit-learn** support KPI forecasting and anomaly detection. The system is containerized and scalable using cloud infrastructure.

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/



Guidelines:

Include all the processes (As an application logic / Technology Block)
Provide infrastructural demarcation (Local / Cloud)
Indicate external interfaces (third party API's etc.)
Indicate Data Storage components / services
Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web-based dashboard for admins and citizens	Streamlit (Python), HTML/CSS (Streamlit theming)
2.	Application Logic-1	Routing, validation, user inputs	FastAPI (Python), Pydantic
3.	Application Logic-2	AI interactions – summarization, chat, tips, reports	IBM Watsonx Granite LLM
4.	Application Logic-3	Semantic policy search & vector embeddings	Pinecone Vector DB, sentence-transformers
5.	Database	Storing feedback, KPIs, policy meta-data	SQLite / NoSQL (as needed)
6.	Cloud Database	Optional cloud persistence for feedback/KPI data	IBM Cloudant / Firebase Realtime DB (optional)
7.	File Storage	Storing .csv and .txt uploads	Local Filesystem or Cloud Block Storage
8.	External API-1	Fetching updated city metrics from public sources	IBM Weather API, etc.
9.	External API-2	Pinecone API for vectorsearch	Pinecone API
10.	Machine Learning Model	Forecasting KPIs, Anomaly Detection	Scikit-learn (Linear Regression, Statistical Check), Pandas
11.	Infrastructure (Server / Cloud)	Backend & frontend deployment	Localhost (dev) / Render / IBM Cloud / Docker

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Frameworks and packages used	Streamlit, FastAPI, Pydantic, Scikit-learn
2.	Security Implementations	API key encryption, role-based access	dotenv, OAuth2.0 (if extended), HTTPS, JWT
3.	Scalable Architecture	Modular, microservice-capable backend with separate frontend	FastAPI + Streamlit decoupled architecture
4.	Availability	Can be containerized & deployed to cloud with 99.9% uptime goal	Docker, IBM Cloud, Render, Load Balancing
5.	Performance	Fast response APIs, async FastAPI, light frontend, optimized vector queries	FastAPI async, LLM caching, Pinecone vector search

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d