Project Design Phase II
Technology Stack (Architecture & Stack)

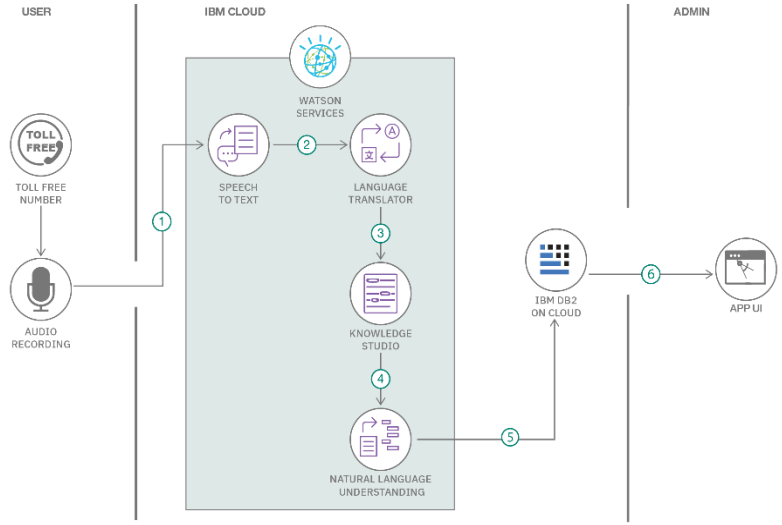
Date	27 June 2025
Team ID	LTVIP2025TMID36160
Project Name	Citizen AI – Intelligent Citizen Engagement Platform
Maximum Marks	4 Marks

Technical Architecture:

Citizen AI utilizes a modular, cloud-native architecture featuring a Flask backend that connects to IBM Watson and Granite APIs for real-time NLP and sentiment analysis. Data is managed via IBM Cloudant and Object Storage, with insights visualized through a dynamic admin dashboard.

The Deliverable shall include the architectural diagram as below, and the information as per Tables 1 and 2.

Example: Order processing during pandemics for offline mode



Guidelines:

The Citizen AI platform is a voice-driven solution where citizen calls via a toll-free number are transcribed and processed by a pipeline of IBM Watson services (Speech to Text, Language Translator, Knowledge Studio, NLU). Extracted insights are stored in IBM DB2 on Cloud, feeding a web-based administrative dashboard. This entire architecture is built on robust IBM Cloud services for end-to-end management.

S. No	Component	Description	Technology
1.	User Interface	The primary interactive portal enables citizens to engage with the AI assistant and empowers government officials with real-time insights via a dynamic dashboard	HTML5, CSS3, JavaScript (leveraging a lightweight approach, potentially server-rendered with Jinja2 templates via Flask, ensuring fast load times and minimal client-side dependencies)
2.	Application Logic - Flask Backend	Flask application logic handling routing and processing	Python (Flask framework) Python 3.9+, Flask 2.x, Gunicorn (WSGI HTTP Server), requests (for robust HTTP client), IBM_Watson SDK (for NLU), IBM Platform Services SDK (for watsonx.ai), psycopg2 (for PostgreSQL interaction).
3.	AI Model Integration - Conversational AI & Contextual Responses	Provides the platform's core conversational intelligence, enabling natural language understanding, context retention, and generative responses for accurate and personalized citizen support.	IBM Watson STT service IBM Granite Models (specifically granite-3.3-8b-instruct or granite-3.3-13b-chat via watsonx.ai API), utilizing an IBM API Key for authentication.
4.	AI Model Integration - Sentiment Analysis	Chat assistant using IBM Granite for contextual, personalized responses	IBM Watson Natural Language Understanding (NLU**)** service (accessed via IBM Watson SDK with an IBM API Key).
5.	Database	A robust, transactional database for storing structured data such as citizen interaction metadata, aggregated sentiment scores, user profiles (if implemented), and a knowledge base of government services.	IBM Cloud Databases for PostgreSQL (Managed Service on IBM Cloud).
6.	Cloud Database	Secure, highly available storage for large, unstructured data assets including complete conversation histories, raw citizen feedback submissions, and potentially media files (if future voice/image input is added)..	IBM Cloudant.
7.	File Storage	Stores logs or file-based citizen inputs	IBM Cloud Object Storage (COS)

8.	External API-1	External civic data such as weather, events, etc.	IBM Weather API
9.	External API-2	Could integrate government ID validation or public records	Aadhar API
10.	Machine Learning Model	This component encapsulates all the AI/ML models leveraged for the platform's intelligence.	IBM Granite LLM
11.	Infrastructure (Server / Cloud)	Application hosted on IBM Cloud (or local dev env during testing)	IBM Cloud Foundry or Kubernetes.

Table 2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask, dotenv, Chart.js / Plotly for dashboard	Flask (Python), Chart.js, dotenv.
2.	Security Implementations	API key protection, HTTPS endpoints, IAM, and environment variable usage.	IBM IAM, HTTPS, and SHA-256 for secure storage.
3.	Scalable Architecture	Microservice-style modular design – UI, API, and AI modules are decoupled.	Flask + IBM Cloud Functions + APIs
4.	Availability	IBM Cloud’s high availability, optional use of load balancers, and fallback APIs.	IBM Cloud Load Balancers, Redundancy.
5.	Performance	Optimized Flask routes, caching headers for static assets, and async API calls.	Flask optimization, IBM Cloud CDN