**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 19 June 2025 |
| Team ID | LTVIP2025TMID31478 |
| Project Name | Sustainable Smart city assistant using IBM granite LLM |
| Maximum Marks | 4 Marks |

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | How users interact with Sustainable Smart city | HTML, CSS, JavaScript / Angular Js / React Js etc. |
|  | Application Logic-1 | Handles user authentication, services requests, grievance submissions. | Python / java |
|  | Application Logic-2 | Converts user speech input into text | IBM Watson Speech to Text |
|  | Application Logic-3 | Chatbot logic to guide citizens and answer | IBM Watson Assistant |
|  | Database | Stores user profiles, submitted issues, service history | MySQL / MongoDB (NoSQL) |
|  | Cloud Database | Scalable storage of structured/unstructured citizen data | IBM DB2 / IBM Cloudant |
|  | File Storage | Stores uploaded documents, photos of civic issues, reports | IBM Cloud Object Storage |
|  | External API-1 | For accessing weather updates related to civic planning | IBM Weather API |
|  | External API-2 | Identity verification | Aadhar UIDAI API |
|  | Machine Learning Model | Predictive model to prioritize issues or detect urgency from complaint text | NLP-based Sentiment Analysis / Classification ML Model |
|  | Infrastructure (Server / Cloud) | Hosting and deployment of the application | IBM Cloud Foundry / Kubernetes |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | List of open-source frameworks used in Citizen AI for front-end, back-end, and AI logic. | React.js, Flask |
|  | Security Implementations | Measures taken to secure user data and control access. | SHA-256 encryption, JWT tokens for authentication, Role-based Access |
|  | Scalable Architecture | The architecture allows scaling based on user load and demand. | 3-Tier Architecture (UI – Logic – Database), Microservices, Docker, Kubernetes |
|  | Availability | Ensures the application is accessible with minimal downtime | Load Balancers, Multi-region Cloud Deployment (AWS/GCP), Auto-scaling Servers |
|  | Performance | Enhances system performance to handle multiple requests and deliver quick responses | Redis Cache, CDN (Cloudflare), Asynchronous APIs, Optimized Database Queries |