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

Date	31 January 3035
Team ID	LTVIP2025TMID29459
Project Name	**Citizen AI – Intelligent Citizen Engagement Platform**
Maximum Marks	4 Marks

🏗 Technical Architecture: CitizenAI

The following architecture illustrates how different components of the CitizenAI system interact to provide features like user authentication, AI-based chat responses, and sentiment analysis using IBM Granite Models.

▯ Use Case: AI-powered citizen feedback and assistance platform
🕹 Mode: Web-based frontend with Flask backend and transformer-based model inference

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 Used
1	User Interface	Web UI for user interaction	HTML, CSS, JavaScript, Jinja2 (Flask templating)
2	Application Logic-1	Web application logic & routing	Python (Flask)
3	Application Logic-2	AI model handling user queries	IBM Granite 3.3 2B Instruct
4	Application Logic-3	Sentiment analysis pipeline	HuggingFace Transformers (pipeline)
5	Database	Stores session, chat history, user inputs (if persistent DB needed)	SQLite (in-memory or local DB)
6	Cloud Database	Optional cloud storage for concerns and logs	IBM Cloudant / Firebase
7	File Storage	Static files (HTML, CSS, scripts, logs)	Local Filesystem
8	External API-1	Identity or location API (optional)	IPinfo / GeoIP (optional)
9	External API-2	Aadhar/SSO integration for auth (optional future use)	UIDAI Aadhar API (optional)
10	Machine Learning Model	Generates responses, performs NLP tasks	IBM Granite 3.3 2B Instruct
11	Infrastructure	Deployed locally or on cloud	Flask server locally / IBM Cloud / Replit etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology / Approach
1	Open-Source Frameworks	Backend and model use open-source stack	Flask, Transformers, HuggingFace
2	Security Implementations	Password authentication, session control, HTTPS (if deployed online)	SHA-256 (for future), Flask session security
3	Scalable Architecture	Scalable via Flask + API endpoints, potential for containerization	3-tier Architecture, Docker (optional)
4	Availability	Local or cloud deployment ensures availability; backups supported	IBM Cloud, Cloudant DB, Replit backup options
5	Performance	Lightweight backend, optimized token limits for model responses	Flask async, transformer caching (if enabled)

📚 References

- C4 Model
- IBM Cloud Architecture
- IBM AI Patterns
- AWS Architecture Center
- How to draw useful technical architecture diagrams