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

Date	28 June 2025
Team ID	LTVIP2025TMID60871
Project Name	Transfer learning based classification of poultry diseases for enhanced health management
Maximum Marks	

Technical Architecture:

Table-1: Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	Mobile/Web interface for farmers, vets, and admins	HTML, CSS, JavaScript, ReactJS / Flutter / Android
2.	Application Logic-1	Handles data collection, symptom input, report generation	Python / Node.js
3.	Application Logic-2	Voice-to-text input for illiterate farmers	IBM Watson STT / Google STT
4.	Application Logic-3	Chatbot for common health tips and disease FAQs	IBM Watson Assistant / Dialogflow
5.	Database	Storage of poultry health records, user profiles, diagnosis results	MySQL / PostgreSQL / MongoDB
6.	Cloud Database	Cloud-hosted database for syncing mobile data and backup	IBM Cloudant / Firebase Realtime DB / AWS RDS
7.	File Storage	Poultry images, diagnostic reports, logs	IBM Cloud Object Storage / AWS S3 / Local FS
8.	External API-1	Weather data for predicting climate-related disease outbreaks	IBM Weather API / OpenWeather API
9.	External API-2	Authentication/verification of farmer credentials	Aadhaar API / Government Livestock API

10.	Machine Learning Model	Classify poultry diseases using image input (transfer learning model)	ResNet / MobileNet / TensorFlow / PyTorch
11.	Infrastructure (Server/Cloud)	Deployment on scalable infrastructure	IBM Cloud, Kubernetes, Docker, Local Ubuntu Server

---

Table-2: Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Frameworks used for front-end, back-end and ML model development	ReactJS, Flask, TensorFlow, PyTorch, Scikit-learn
2.	Security Implementations	Secure login, encrypted communication, access control, IAM roles	SHA-256, JWT, HTTPS, OWASP Guidelines, IAM Policies
3.	Scalable Architecture	Microservices-based structure with loosely coupled components and containerization	Kubernetes, Docker, REST APIs, Message Queues (Kafka)
4.	Availability	Cloud deployment across multiple availability zones, auto-scaling, load balancing	IBM Cloud Multi-Zone, NGINX Load Balancer, Failover
5.	Performance	Efficient caching for repeat queries, ML inference optimization, CDN for static assets	Redis, TensorRT (for model), Cloudflare CDN

Would you like a visual architecture diagram or a PowerPoint/Word document version of this content?