**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 24 June 2025 |
| Team ID | LTVIP2025TMID34708 |
| Project Name | Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management |
| Maximum Marks | 4 Marks |

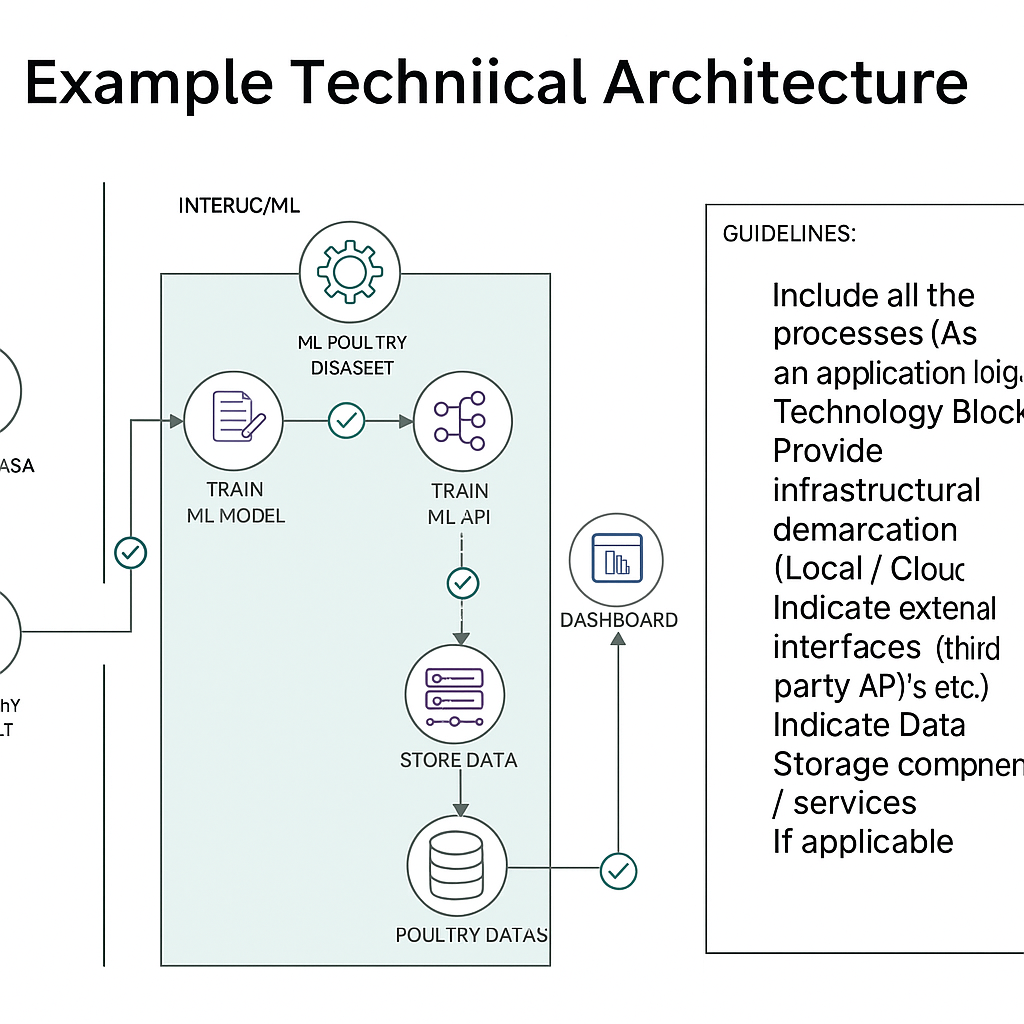
**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)



****

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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Mobile & web-based farmer and vet dashboard | HTML, CSS, JavaScript, React Native / React.js |
|  | Application Logic-1 | Data ingestion & cleaning logic for symptoms, environment, and sample data | Python (Pandas, NumPy, Requests) |
|  | Application Logic-2 | ML model training & transfer learning for disease classification | Python (TensorFlow, Keras, scikit-learn) |
|  | Application Logic-3 | API service for real-time disease prediction | Flask / FastAPI |
|  | Database | Store structured poultry health and environment data | MongoDB / PostgreSQL |
|  | Cloud Database | Scalable cloud database for data & analytics | AWS RDS / Google Cloud Firestore |
|  | File Storage | Store raw symptom images, CSV files, and model artifacts | AWS S3 / Google Cloud Storage |
|  | External API-1 | Get real-time weather and farm environmental data | OpenWeather API / Local IoT integrations |
|  | External API-2 | Get veterinary disease guidelines and reference data | WHO OIE APIs / FAO datasets |
|  | Machine Learning Model | Predict poultry diseases using transfer learning classification | Transfer Learning Models (MobileNetV2, ResNet50) |
|  | Infrastructure (Server / Cloud) | Cloud deployment for continuous access and scaling | Docker, Kubernetes, AWS / GCP / Azure |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Use of open-source ML and web frameworks for transparency and cost efficiency | TensorFlow, Keras, FastAPI, React, MongoDB |
|  | Security Implementations | API authentication, encrypted data transmission, role-based user access | HTTPS, JWT, OAuth2, IAM |
|  | Scalable Architecture | Microservice-based design with auto-scaling for increased user/data load | Kubernetes, Docker |
|  | Availability | Load-balanced, multi-region cloud deployment for high uptime | AWS ELB, GCP Load Balancer, Azure Front Door |
|  | Performance | Caching, optimized model inference, CDN for faster dashboard loading | Redis, Flask Async, Cloud CDN |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)