**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 31 January 3035 |
| Team ID | LTVIP2025TMID59571 |
| 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:**

**TECHNICAL STACK**

**Component Description Technology Used**

1User Interface How user interacts with the application (uploading images, viewing results) HTML, CSS, JavaScript (Vanilla or Bootstrap)

2Application Logic-1 Logic for uploading poultry images and interacting with backend Python (Flask Framework)

3Application Logic-2 Logic for disease classification using ML model PyTorch, ResNet18 Transfer Learning

4Application Logic-3 Logic for serving predictions via API and routing Flask REST API

5 Database Storage of user credentials, prediction history MySQL

6Cloud Database Optional cloud deployment for DB AWS RDS / Google Cloud SQL / Render.com DB

7File Storage Local storage of trained model files (model.h5/.pt) or temporary uploads Local Filesystem or AWS S3 (if cloud)

8External API-1 (Optional) Integration of weather API to track temperature effect on poultry health OpenWeather API (optional)

9External API-2 (Optional) User verification using Aadhaar or OTP service Aadhaar API / Twilio API (if applicable)

10Machine Learning Model Classifi

**Application Characteristics**

**S.No Characteristics Description Technology Used**

1 Open-Source Frameworks Flask for backend routing, PyTorch for model training, Bootstrap for frontend Flask, PyTorch, Bootstrap

2 Security Implementations User password hashing, secure API endpoints, validation SHA-256, Flask-JWT/Auth, HTTPS

3 Scalable Architecture The app follows a modular design (3-tier) that can be scaled using cloud services Flask API + Frontend + DB (3-Tier)

4 Availability Deployable to 24/7 cloud platforms (e.g., Render, Heroku, or AWS) with minimal downtime Load balancing via Render/Heroku

5Performance Designed to handle multiple requests using lightweight models, could use Redis for caching Flask + Gunicorn, Optional Redis

**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)