Solution Architecture Document

Project: "Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management"
Location: Ongole, Andhra Pradesh
**Date:** June 2025
Team ID: LTVIP2025TMID42969
Team Members: P. Srinivasa Kalyan, M. Karthik Reddy

 Project Duration: June 24-26, 2025

1. System Architecture Overview

PoultryDetect is a web-based AI application designed to assist farmers and veterinarians in early detection of poultry diseases through image analysis using deep learning models.

1.1 High-Level Architecture Diagram
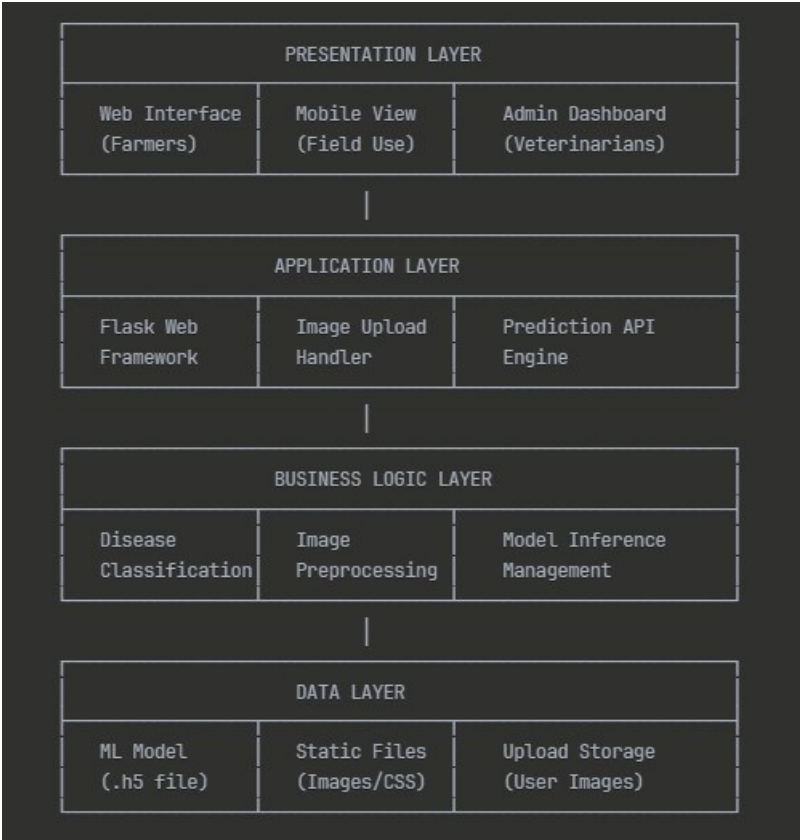
1.2 Technology Stack

Frontend Technologies:

- HTML5, CSS3, JavaScript

- Tailwind CSS for styling

- Lottie animations for UI enhancement

Backend Technologies:

-

  - Python 3.8+

Flask web framework

- Keras/TensorFlow for ML model

- NumPy for numerical operations

- Werkzeug for file handling **Machine Learning:**

- Deep Learning CNN model

- Image preprocessing with Keras

- Classification for 4 disease categories:

  - Coccidiosis o   Healthy o   Salmonella

  - Newcastle Disease

1.3 System Components

1. Web Application (Flask)

- Route handling for different pages

- File upload management

- Template rendering engine

- Static file serving

2. AI/ML Engine

- Pre-trained CNN model (healthy_vs_rotten.h5)

- Image preprocessing pipeline

- Prediction confidence scoring

- Real-time inference capabilities

3. User Interface

- Responsive web design •   Multi-page navigation

- Image upload interface

- Results display system

4. File Management System

- Secure file upload handling

- Image storage in static/uploads

  File validation and sanitization

•

2. Deployment Architecture
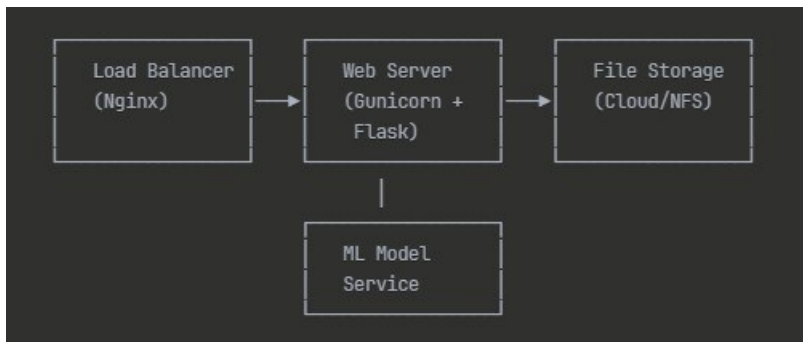
2.1 Development Environment

Local Development Server

├── Flask Development Server (Port 5000)

├── File System Storage └──

Local Model Loading

2.2 Production Architecture (Recommended)

3. Security Architecture

3.1 Security Measures

- File upload validation using secure_filename()

- File type restrictions for image uploads

- Error handling for malicious files

- Path traversal protection

3.2 Data Protection

- No persistent storage of sensitive data

- Temporary file handling

- Input sanitization

4. Performance Considerations

4.1 Optimization Strategies

- Model loading optimization on startup

- Image preprocessing efficiency

- Asynchronous file handling

- Caching static resources

4.2 Scalability Features

Stateless application design

-

  - Horizontal scaling capability

  - Load balancer compatibility

  - Cloud deployment ready

5. Integration Points

5.1 External Services

  - Google Scholar API integration for research links

  - Lottie Files CDN for animations

  - Tailwind CSS CDN for styling

5.2 Future Integration Possibilities

  - Database integration for user management

  - API endpoints for mobile applications

  - Real-time notification systems

  - Analytics and monitoring tools