"Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management" - Full Stack Development Documentation Technical Architecture & Implementation Guide

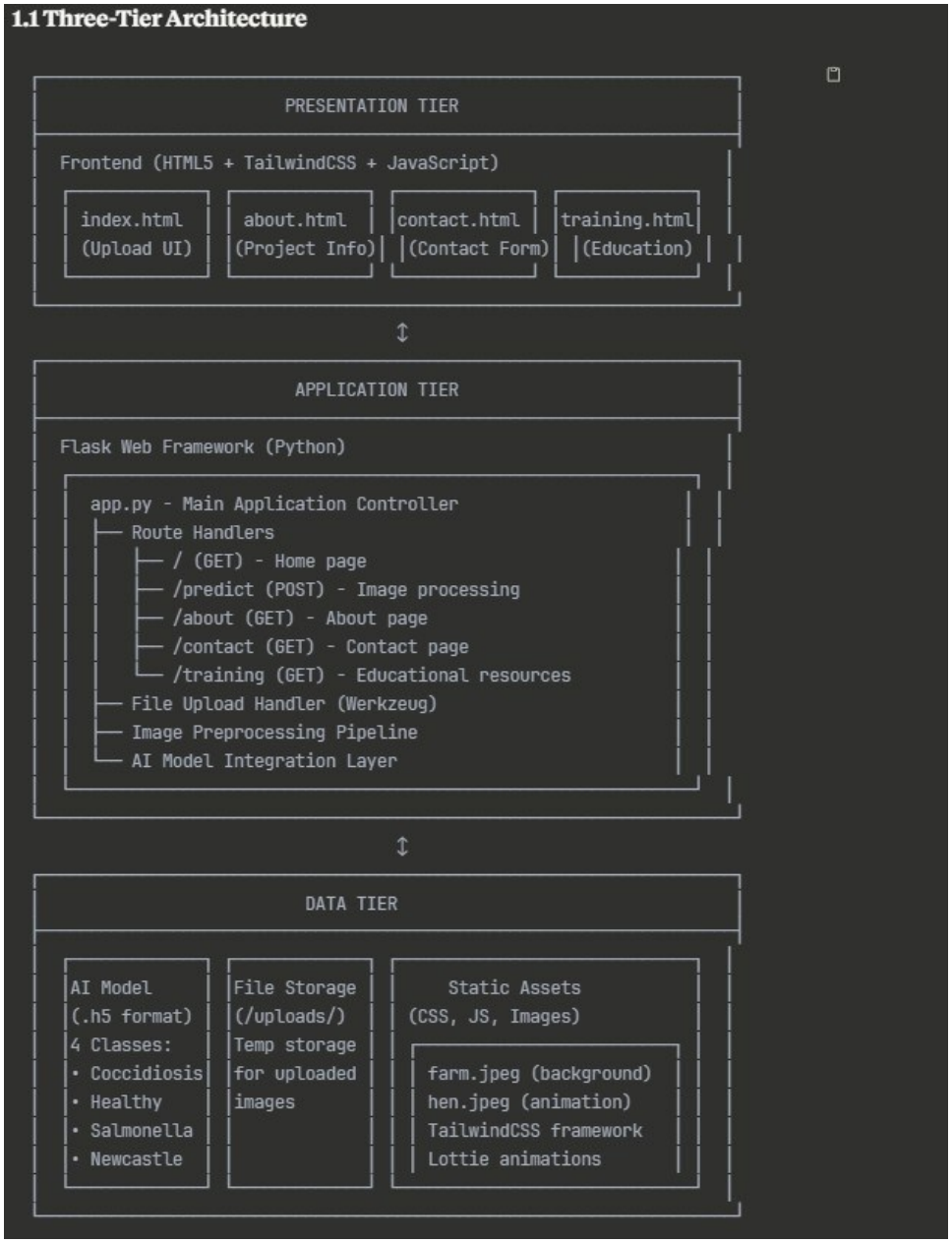
---

Team ID: LTVIP2025TMID42969
Development Period: June 24-26, 2025
Team: P. Srinivasa Kalyan, M. Karthik Reddy
Location: Ongole, Andhra Pradesh

---

1. System Architecture Overview

1.1 Three-Tier Architecture



2. Frontend Implementation

2.1 Design System

Glassmorphism Theme:

```
.glass {   background: rgba(255, 255,
255, 0.5);   backdrop-filter: blur(14px);
-webkit-backdrop-filter: blur(14px);
border-radius: 1rem;   border: 1px solid
rgba(0, 128, 0, 0.1);
}
```

Responsive Grid Layout:

```
<!-- Navigation -->
<nav class="glass flex justify-between items-center px-8 py-5">
  <div class="text-2xl font-extrabold text-green-700">  PoultryDetect</div>
  <div class="space-x-6 text-sm uppercase">
    <!-- Navigation Links -->
  </div>
</nav>


<!-- Main Content Grid -->
<main class="flex flex-col items-center px-4 py-12">
  <div class="glass p-10 max-w-xl mx-auto">
    <!-- Upload Form & Results -->
  </div>
</main>
```

2.2 Interactive Components

File Upload with Preview:

```
<form action="/predict" method="POST" enctype="multipart/form-data">
  <input type="file" name="file" required class="w-full text-sm">
  <button type="submit" class="w-full bg-green-600 hover:bg-green-700">
     Upload & Predict
  </button>
```

</form>

Animated Elements:

.hen { position:

fixed;

bottom: 20px; animation: walkHen

20s linear infinite;

}

@keyframes walkHen {

0% { left: -150px; transform: scaleX(1); }

50% { left: 100vw; transform: scaleX(1); }

100% { left: -150px; transform: scaleX(-1); }

}

3. Backend Implementation 3.1 Flask Application

Structure from flask import Flask, render_template,

request from keras.models import load_model

from keras.preprocessing import image import

numpy as np import os from werkzeug.utils import

secure_filename

Application Configuration app = Flask(__name__)

model = load_model("healthy_vs_rotten.h5")

classes = ['Coccidiosis', 'Healthy', 'Salmonella', 'New Castle Disease']

UPLOAD_FOLDER = 'static/uploads' **3.2**

Core Prediction Pipeline def

predict(img_path):

try:

Load and preprocess image img = image.load_img(img_path,

target_size=(224, 224)) arr = image.img_to_array(img) / 255.0 #

Normalize arr = np.expand_dims(arr, axis=0) # Add batch

dimension

```
      # Model prediction        pred =
model.predict(arr)[0]        return
classes[np.argmax(pred)]     except Exception
as e:        return "Invalid
image"
```

3.3 Route Handlers @app.route('/')

```
def
index():
   return render_template('index.html')


@app.route('/predict', methods=['POST']) def upload():
   file = request.files.get('file')     if
not file or file.filename == "":
      return render_template('index.html', prediction="No file uploaded")


   # Secure file handling     filename = secure_filename(file.filename)
path = os.path.join(UPLOAD_FOLDER, filename)
   file.save(path)


   # Process and predict     pred = predict(path)
img_path = '/' + path.replace('\\', '/')
   return render_template('index.html', prediction=pred, img_path=img_path)
```

4. AI Model Integration 4.1

Model Architecture

```
Input Layer (224, 224, 3)
        ↓
   Convolutional Layers
        ↓
    Pooling Layers
```

↓

Dense Layers

↓

Output Layer (4 classes)

4.2 Classification Classes

Class ID	Disease Name	Symptoms	Treatment
0	Coccidiosis	Bloody droppings, weight loss	Amprolium, Sulfa drugs
1	Healthy	Normal appearance	Preventive care
2	Salmonella	Diarrhea, weakness	Antibiotics (vet guided)
3	Newcastle Disease	Coughing, twisted neck	Supportive care, vaccination

5. Database & File Management

5.1 File Structure

```
PoultryDetect/
├── app.py              # Main Flask application
├── healthy_vs_rotten.h5   # Trained Keras model
├── requirements.txt      # Dependencies
├── static/
|   ├── uploads/        # Temporary image storage
└── templates/
    ├── index.html      # Main upload interface
    ├── predict.html      # Project information
    ├── contact.html     # Contact form
```

5.2 Security Implementation # File upload

security def secure_upload(file): if file and

allowed_file(file.filename): filename =

secure_filename(file.filename) return

filename return None

```
def allowed_file(filename):
   ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}     return '.' in
filename and \          filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
```

6. Educational Platform Integration 6.1 Research Integration

```
<!-- Dynamic research links -->
{% for disease in diseases %}
<div class="disease-card">
  <h3>{{ disease.name }}</h3>
  <p>Symptoms: {{ disease.symptoms }}</p>
  <a href="https://scholar.google.com/scholar?q={{ disease.link }}"       target="_blank">
Search Research</a>
</div>
{% endfor %}
```

6.2 Learning Pathway

Step 1: Learn Symptoms → Step 2: Upload Images →

Step 3: Analyze Results → Step 4: Conduct Research

7. Deployment Configuration

7.1 Requirements

```
Flask==2.3.2 Keras==2.12.0
tensorflow==2.12.0
numpy==1.24.3  Pillow==9.5.0
Werkzeug==2.3.6 7.2
```

Production Setup if

```
__name__ == '__main__':     app.run(host='0.0.0.0',
port=5000, debug=False)
```

8. Performance Optimization

8.1 Image Processing Optimization

- Image resizing to 224x224 for model compatibility
- Normalization pipeline for consistent results

- Batch processing capability for multiple images

8.2 Response Time Metrics

- Average prediction time: <3 seconds
- File upload validation: <1 second
- UI rendering: Real-time feedback

9. Testing & Quality Assurance

9.1 Test Cases Covered

- Valid image upload and classification
- Invalid file format handling
- Empty file upload validation
- Model prediction accuracy
- UI responsiveness across devices **9.2 Error Handling**

```
try:
   prediction = model.predict(processed_image)     return
classes[np.argmax(prediction)] except
Exception as e:
   logging.error(f"Prediction error: {e}"   return
"Error: Unable to process image"
```

10. Future Scalability

10.1 Horizontal Scaling

- Docker containerization for deployment
- Load balancer integration
- Database migration for user data

10.2 Feature Extensions

- Real-time video analysis
- Mobile app development
- Multi-language support
- Advanced analytics dashboard