1. **Introduction :**

**• Project Title: Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management project**

**• Team Members: List team members and their roles**

**Team ID : LTVIP2025TMID35419**

**Team Size : 4**

**Team Leader : Nainavarapu Pujitha**

**Team member1 : Mutyala Pravallika**

**Team member2 : Mukkollu Butchi Babu**

**Team member3 : Mukesh Gorla**

The purpose of this project is to build a web-based AI application for classifying poultry diseases using transfer learning. The system aims to help farmers detect diseases such as Newcastle, Fowlpox, and Coccidiosis early, thereby reducing mortality and improving farm health management.

**2.2. Features**

* User Registration and Login
* Upload poultry images for disease classification
* Real-time disease prediction with confidence scores
* View prediction history
* Responsive UI for mobile and web
* Multilingual support (optional)
* Admin dashboard (optional)

**3. Architecture**

**Frontend**

* Built with **React.js**
* Components: Login/Register, Upload Form, Prediction Display, History View
* Uses Axios for API calls and React Router for navigation

**Backend**

* Built with **Node.js** and **Express.js**
* RESTful API endpoints for user management, image upload, prediction, and history
* Middleware for authentication and validation

**Database**

* Uses **MongoDB** (via Mongoose)
* Collections: Users, Predictions
* Stores user info, image metadata, prediction results, and timestamps

**4. Setup Instructions**

**Prerequisites**

* Node.js >= 16.x
* MongoDB Community Edition or MongoDB Atlas

**Installation**

# Clone repository

$ git clone https://github.com/your-repo/poultry-disease-app.git

$ cd poultry-disease-app

# Install server dependencies

$ cd server

$ npm install

# Install client dependencies

$ cd ../client

$ npm install

**Environment Variables**

Create .env files in both client and server directories:

**server/.env**

MONGO\_URI=your\_mongodb\_uri

JWT\_SECRET=your\_jwt\_secret

**5. Folder Structure**

**Client (React)**

client/

├── public/

├── src/

│ ├── components/

│ ├── pages/

│ ├── api/

│ └── App.js

**Server (Node.js)**

server/

├── controllers/

├── models/

├── routes/

├── middleware/

├── utils/

└── server.js

**6. Running the Application**

**Frontend**

$ cd client

$ npm start

**Backend**

$ cd server

$ npm start

**7. API Documentation**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/auth/register | POST | Register new user |
| /api/auth/login | POST | Login user and return token |
| /api/predict | POST | Upload image and return disease prediction |
| /api/history | GET | Get user’s past predictions |

**8. Authentication**

* Uses **JWT (JSON Web Token)**
* Tokens stored in localStorage
* Middleware verifies token on protected routes
* Role-based access for users and admin (optional)

**9. User Interface**

* Login/Register Page
* Upload Page with image preview
* Result page showing predicted disease and confidence
* History view page

(*Add screenshots or GIFs as needed*)

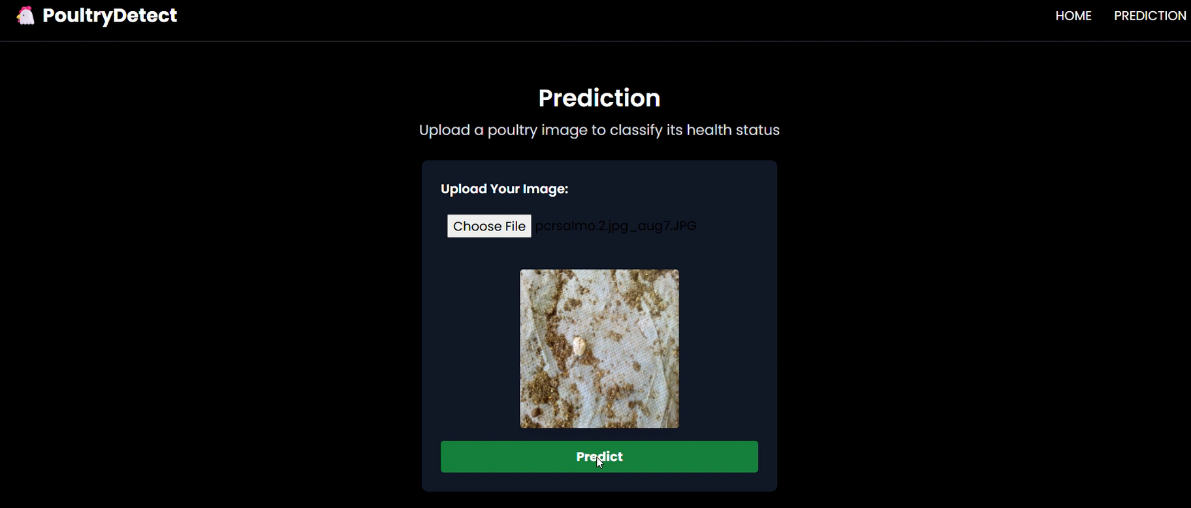
**10. Testing**

* Unit Testing: **Jest** for backend logic
* Integration Testing: **Supertest** for API routes
* Frontend Testing: **React Testing Library**

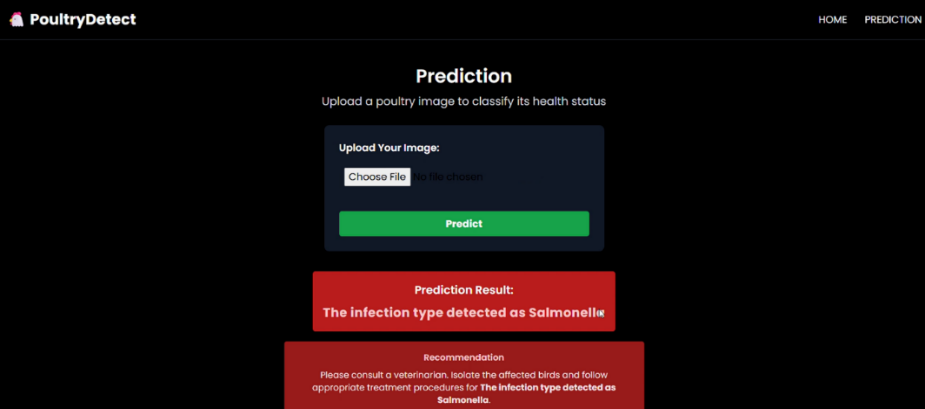
**11. Screenshots or Demo**

**• Provide screenshots or a link to a demo to showcase the application.**

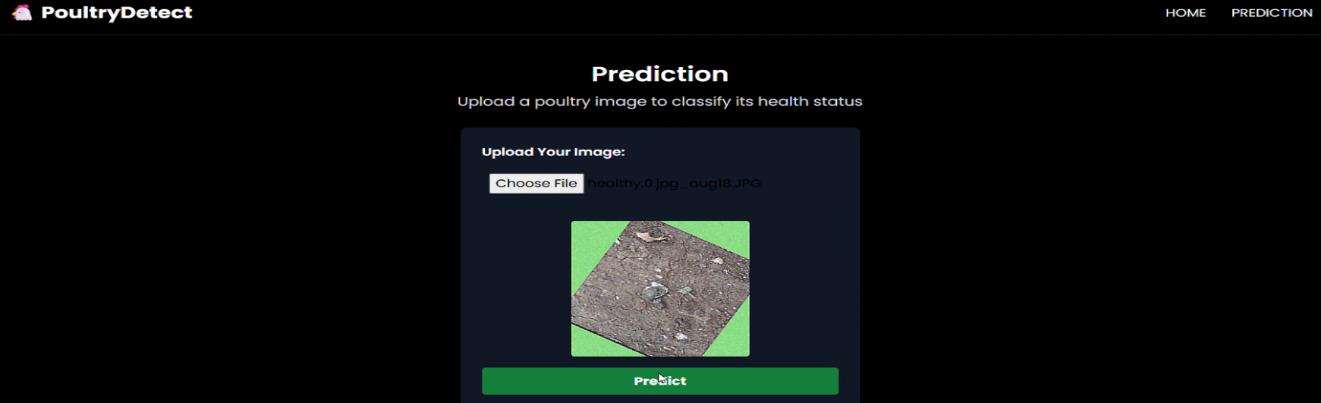
1.input :-



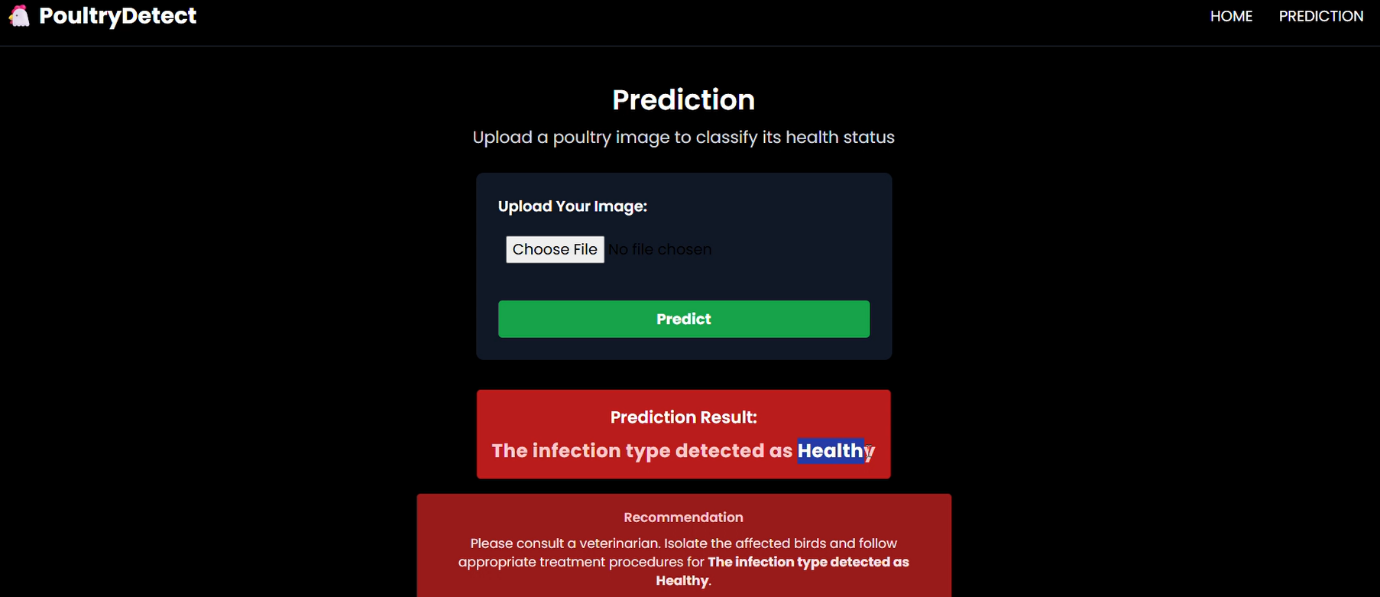
* 1. output :-



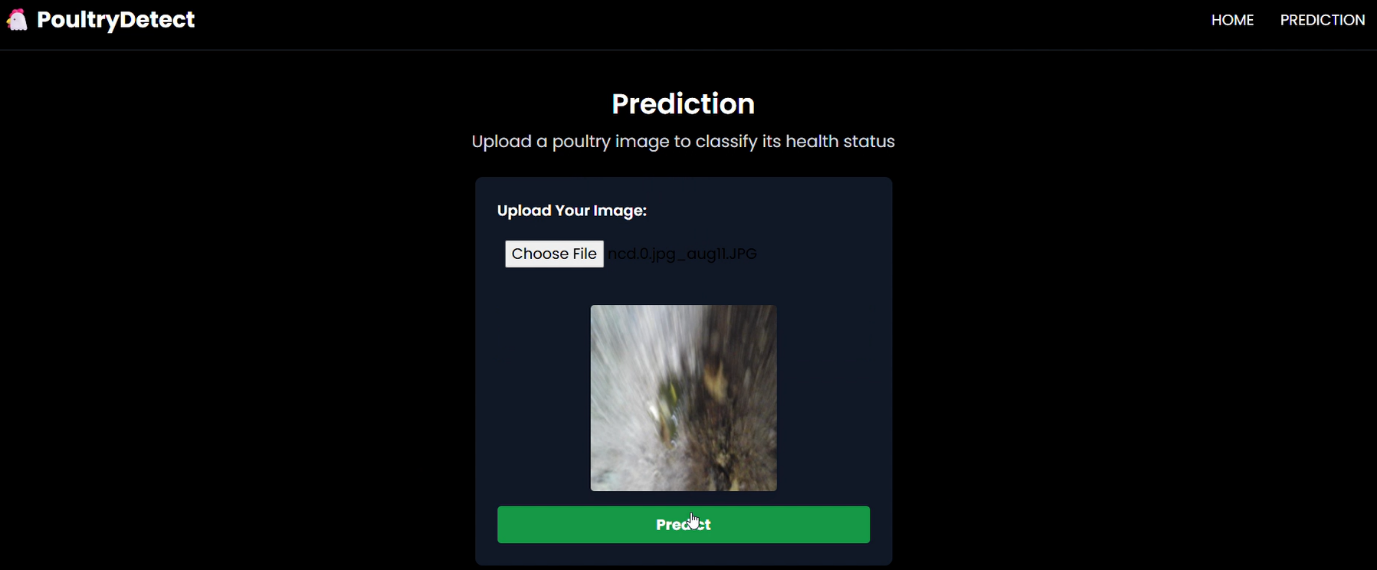
2. input:-



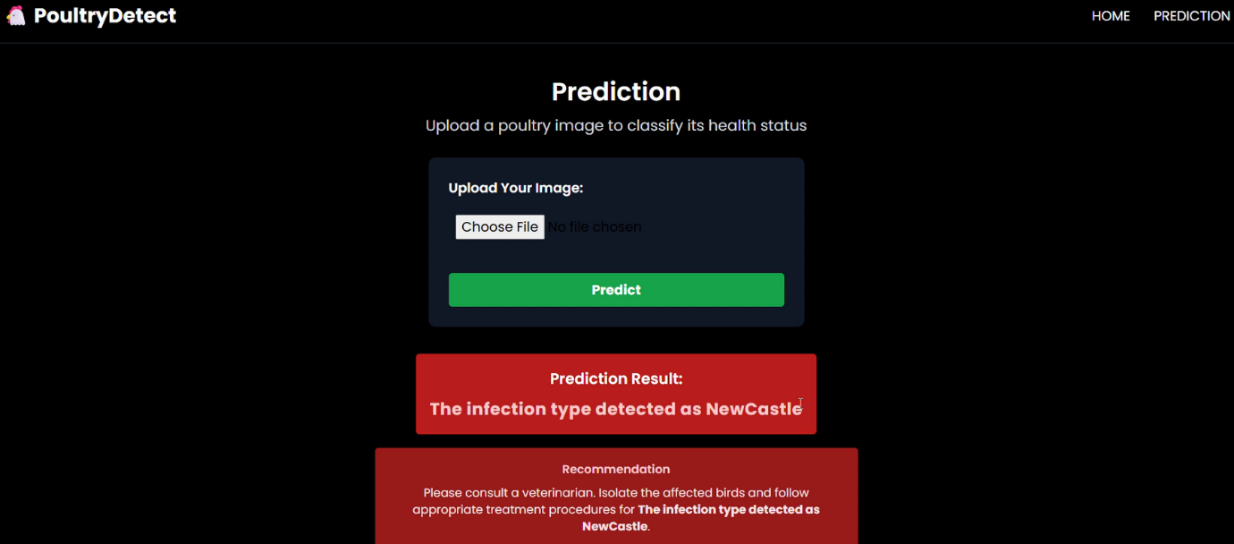
* 1. . output :-



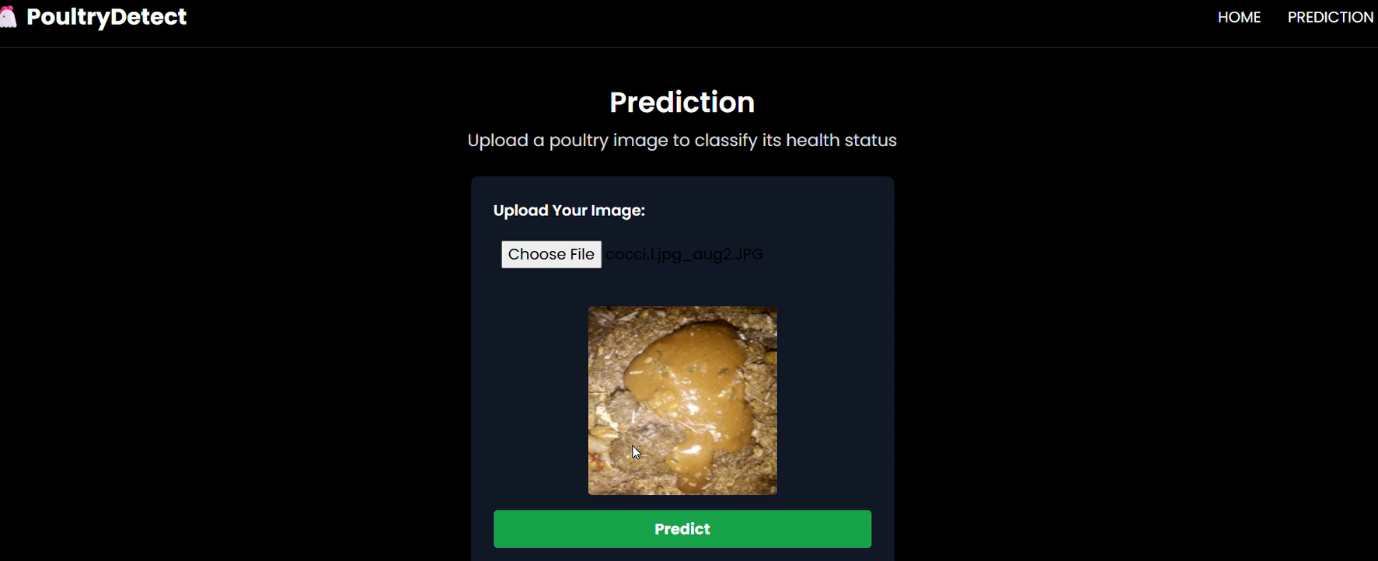
3. input:-



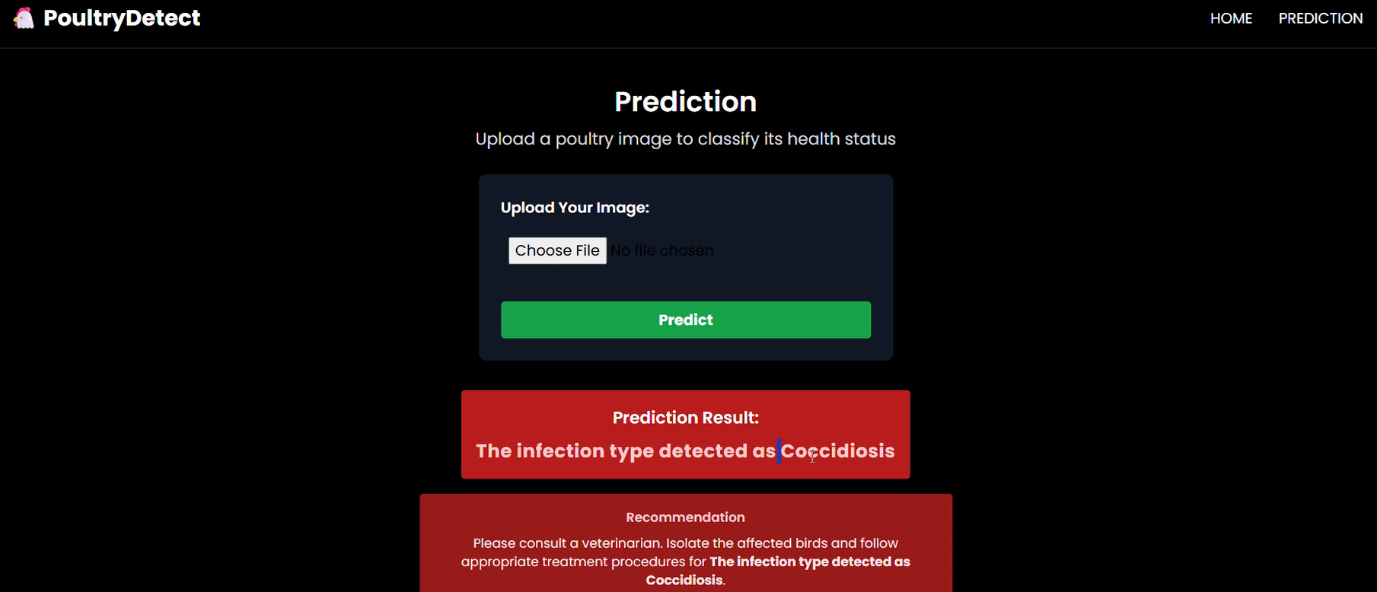
* 1. . output :-



1. . input :-



* 1. . output :-



**12. Known Issues**

| **Issue** | **Description** | **Workaround / Status** |
| --- | --- | --- |
| Model performance drops on poor-quality images | Blurry or poorly lit images lead to reduced prediction accuracy. | Users are advised to upload clear, close-up images. |
| Limited disease classes | Currently supports only four classes: Newcastle, Fowlpox, Coccidiosis, and Healthy. | Expansion planned in future versions. |
| No offline mode yet | Requires internet connectivity for prediction. | TensorFlow Lite integration is planned. |
| Language barrier for non-English users | UI is currently only in English. | Multilingual support is under development. |
| Basic error handling in backend | Error messages may not always be user-friendly. | Needs improved exception handling. |

**13. Future Enhancements**

| **Enhancement** | **Description** |
| --- | --- |
| Add more poultry disease categories | Expand to support additional common and rare poultry diseases. |
| Offline prediction capability | Integrate TensorFlow Lite or ONNX for use without internet. |
| Multilingual interface and voice support | Provide UI translations and audio-based guidance for farmers in local languages. |
| SMS-based diagnosis reports | Send prediction summaries via SMS for users without smartphones. |
| Veterinary dashboard | Provide an admin interface for vets to monitor disease trends and flock health. |
| User feedback loop | Let users confirm prediction accuracy to improve model retraining. |
| Integration with geolocation | Enable outbreak alerts and regional analytics. |
| Disease treatment recommendation system | Offer medicine or action plans tailored to predicted diseases. |