

Project Planning Document

Project: PoultryDetect - AI-Powered Poultry Disease Detection System

Location: Ongole, Andhra Pradesh

Date: June 2025

Team ID: LTVIP2025TMID42969

Team Members: M. Karthik Reddy, P. Srinivasa Kalyan

Project Duration: June 24-26, 2025 (3 Days)

1. Project Overview

1.1 Project Objectives

- Develop an AI-powered web application for poultry disease detection
- Create an intuitive interface for farmers and veterinarians
- Implement machine learning model for accurate disease classification
- Provide educational resources for poultry disease management

1.2 Project Scope

In Scope:

- Web-based application development
- AI model integration for disease detection
- User interface design and implementation
- Basic file upload and processing functionality
- Educational content for disease awareness

Out of Scope:

- Mobile application development
- Database integration
- User authentication system
- Real-time monitoring capabilities
- Advanced analytics dashboard

2. Project Timeline & Milestones

2.1 Day-wise Project Schedule

Day 1 (June 24, 2025) - Foundation & Setup

- 09:00 - 10:00 | Project kickoff and requirement analysis
- 10:00 - 12:00 | Environment setup and dependency installation
- 12:00 - 13:00 | Lunch Break
- 13:00 - 15:00 | Flask application structure development
- 15:00 - 16:00 | ML model integration planning
- 16:00 - 17:00 | Basic routing and template setup
- 17:00 - 18:00 | Day 1 review and next day planning

Day 2 (June 25, 2025) - Core Development

- 09:00 - 10:00 | ML model integration and testing
- 10:00 - 12:00 | Frontend UI development with Tailwind CSS
- 12:00 - 13:00 | Lunch Break
- 13:00 - 15:00 | Image upload functionality implementation
- 15:00 - 16:00 | Prediction engine development
- 16:00 - 17:00 | Error handling and validation
- 17:00 - 18:00 | Day 2 testing and bug fixes

Day 3 (June 26, 2025) - Finalization & Documentation

- 09:00 - 10:00 | UI/UX enhancements and animations
- 10:00 - 12:00 | Educational content integration
- 12:00 - 13:00 | Lunch Break
- 13:00 - 15:00 | Final testing and optimization
- 15:00 - 16:00 | Documentation preparation
- 16:00 - 17:00 | Project presentation preparation
- 17:00 - 18:00 | Final review and deployment

2.2 Key Milestones

| Milestone | Target Date | Status | Deliverable |
|----------------------|-------------------|------------|--------------------------------|
| Environment Setup | June 24, 11:00 AM | ✔ Complete | Development environment ready |
| Flask App Structure | June 24, 5:00 PM | ✔ Complete | Basic app with routing |
| ML Model Integration | June 25, 10:00 AM | ✔ Complete | Working prediction engine |
| Frontend Development | June 25, 4:00 PM | ✔ Complete | Complete UI with styling |
| Testing & Bug Fixes | June 25, 6:00 PM | ✔ Complete | Stable application |
| Final Documentation | June 26, 5:00 PM | ✔ Complete | Complete project documentation |

3. Resource Allocation

3.1 Team Responsibilities

M. Karthik Reddy:

- Backend development (Flask application)
- ML model integration and optimization
- File handling and security implementation
- API endpoint development

P. Srinivasa Kalyan:

- Frontend development (HTML, CSS, JavaScript)
- UI/UX design implementation
- Educational content creation
- Testing and quality assurance

3.2 Technology Resources

- Development Tools: VS Code, Python 3.8+
- Frameworks: Flask, Keras/TensorFlow
- Frontend: HTML5, CSS3, Tailwind CSS
- Version Control: Git
- Testing: Manual testing protocols

3.3 Hardware Requirements

- Development Machines: 2 laptops with 8GB+ RAM
- Storage: 50GB available space for models and data
- Network: Stable internet for CDN resources

4. Risk Management

4.1 Identified Risks & Mitigation

| Risk | Probability | Impact | Mitigation Strategy |
|-----------------------------|-------------|--------|--------------------------------|
| Model loading issues | Medium | High | Test model compatibility early |
| File upload vulnerabilities | Low | High | Implement secure file handling |
| UI/UX complexity | Medium | Medium | Use proven CSS framework |
| Time constraints | High | Medium | Prioritize core features |
| Technical dependencies | Low | Medium | Have backup libraries ready |

4.2 Contingency Plans

- **Model Issues:** Fallback to simpler prediction logic
- **Time Overrun:** Reduce scope to essential features

- **Technical Failures:** Pair programming for critical components

5. Quality Assurance

5.1 Testing Strategy

- **Unit Testing:** Individual component validation
- **Integration Testing:** End-to-end workflow testing
- **User Acceptance Testing:** Simulate real user scenarios
- **Performance Testing:** File upload and prediction speed

5.2 Success Criteria

- ☒ Successful image upload and processing
- ☒ Accurate disease prediction display
- ☒ Responsive UI across devices
- ☒ Error handling for edge cases
- ☒ Educational content accessibility

6. Communication Plan

6.1 Daily Standups

- **Time:** 9:00 AM daily
- **Duration:** 15 minutes
- **Format:** Progress update, blockers, day's plan

6.2 Review Meetings

- **End of Day 1:** Architecture and foundation review
- **End of Day 2:** Feature completeness assessment
- **End of Day 3:** Final presentation preparation

7. Deliverables Checklist

7.1 Technical Deliverables

- ☒ Working Flask web application
- ☒ Integrated ML model for disease detection
- ☒ Responsive user interface
- ☒ File upload and processing system
- ☒ Educational content pages

7.2 Documentation Deliverables

- ☒ Solution Architecture Document
- ☒ Project Planning Document
- ☒ Requirements Analysis
- ☒ Customer Journey Map
- ☒ Data Flow Diagrams
- ☒ User Stories
- ☒ Technology Stack Documentation
- ☒ Final Report
- ☒ FSD Documentation

8. Project Closure

8.1 Final Review Criteria

- All planned features implemented and tested
- Documentation completed and reviewed
- Code quality meets standards
- Application ready for demonstration

8.2 Lessons Learned

- Early ML model testing prevents integration issues
- Tailwind CSS significantly speeds up UI development
- Pair programming effective for complex components
- Time-boxed development maintains focus