PLANNING LOGIC

Team ID: LTVIP2025TMID42969

Location: Ongole, Andhra Pradesh

Date: June 2025

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

Project Planning Methodology

Planning Framework: Agile-Waterfall Hybrid

```
Research Phase → Design Phase → Development Phase → Testing Phase → Deployment Phase
   (2 weeks)     (1 week)       (4 weeks)          (1 week)       (1 week)
```

Planning Logic Framework

1. Goal-Oriented Planning

```
PRIMARY GOAL: Develop AI-powered poultry disease detection system
    ↓
SUB-GOALS:
├── Technical Excellence (Model Accuracy >85%)
├── User Experience (Usability Score >4/5)
├── Performance (Response Time <3s)
└── Accessibility (Mobile-friendly, Local language)
```

2. Risk-Based Planning

```
IDENTIFIED RISKS → MITIGATION STRATEGIES → CONTINGENCY PLANS
    ↓                    ↓                      ↓
High Impact Risks    Preventive Actions    Backup Plans
```

Risk Matrix:

Risk	Probability	Impact	Priority	Mitigation
Model accuracy below target	Medium	High	1	Extended training, data augmentation
Poor user adoption	Medium	High	2	User testing, feedback incorporation
Technical performance issues	Low	Medium	3	Load testing, optimization
Timeline delays	Medium	Medium	4	Buffer time, parallel development

3. Resource-Constraint Planning

```
AVAILABLE RESOURCES:
├── Time: 9 weeks total
├── Team: 2 developers
├── Budget: Educational project (minimal)
├── Technology: Open-source tools
└── Infrastructure: Cloud free tiers
```

Resource Allocation Logic:

- 40% Development time
- 25% Research & Planning
- 20% Testing & Validation
- 10% Documentation
- 5% Deployment & Setup

Work Breakdown Structure (WBS)

Level 1: Major Phases

```
1. PROJECT INITIATION (Week 1-2)
2. SYSTEM DESIGN (Week 3)
3. DEVELOPMENT (Week 4-7)
4. TESTING & VALIDATION (Week 8)
5. DEPLOYMENT & DOCUMENTATION (Week 9)
```

Level 2: Phase Breakdown

1. PROJECT INITIATION
   ├── 1.1 Problem Definition
   ├── 1.2 Requirements Gathering
   ├── 1.3 Technology Research
   └── 1.4 Project Planning

2. SYSTEM DESIGN
   ├── 2.1 Architecture Design
   ├── 2.2 UI/UX Design
   ├── 2.3 Database Design
   └── 2.4 API Design

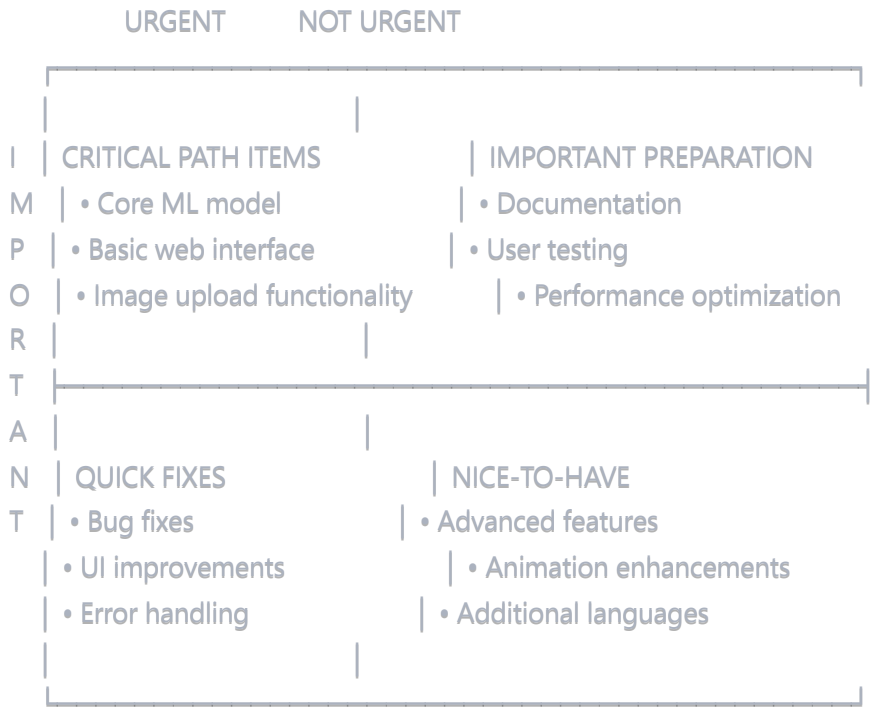
3. DEVELOPMENT
   ├── 3.1 Model Development
   ├── 3.2 Backend Development
   ├── 3.3 Frontend Development
   └── 3.4 Integration

4. TESTING & VALIDATION
   ├── 4.1 Unit Testing
   ├── 4.2 Integration Testing
   ├── 4.3 User Acceptance Testing
   └── 4.4 Performance Testing

5. DEPLOYMENT
   ├── 5.1 Production Setup
   ├── 5.2 Documentation
   ├── 5.3 User Training
   └── 5.4 Project Handover

Task Prioritization Logic

Priority Matrix (Eisenhower Method)



Task Dependencies Logic

```
SEQUENTIAL DEPENDENCIES:
Research → Model Training → Web Development → Testing → Deployment

PARALLEL OPPORTUNITIES:
├── Frontend Development || Backend Development
├── Documentation || Testing Preparation
└── UI Design || Model Optimization
```

Decision-Making Framework

Technical Decisions

```
DECISION CRITERIA:
1. Technical Feasibility (Can we implement it?)
2. Resource Availability (Do we have skills/time?)
3. User Impact (Does it solve the problem?)
4. Maintenance Burden (Can we support it?)
5. Scalability Potential (Will it grow?)
```

Example Decision Process:

Question: Choose between React frontend vs HTML templates
Evaluation:
├── Feasibility: Both feasible
├── Resources: HTML templates faster to implement
├── User Impact: Similar user experience
├── Maintenance: HTML simpler for team
└── Scalability: React better for future, but HTML sufficient now
Decision: HTML templates (resource-optimized choice)

Scope Management Logic

MUST HAVE (Core Features):
├── Image upload and classification
├── Disease information display
├── Mobile-responsive design
└── Basic error handling

SHOULD HAVE (Important):
├── Educational content
├── Research links
├── Performance optimization
└── Security measures

COULD HAVE (Nice-to-have):
├── Animation effects
├── Advanced styling
├── Additional diseases
└── Offline functionality

WON'T HAVE (Future scope):
├── Real-time monitoring
├── Multi-language support
├── Mobile application
└── Database integration

Timeline Logic

Critical Path Analysis

CRITICAL PATH (42 days):
Requirements (3) → Research (5) → Model Training (7) →
Web Development (14) → Integration (3) → Testing (5) →
Deployment (3) → Documentation (2)

PARALLEL TRACKS:
Track A: ML Development (Research → Training → Testing)
Track B: Web Development (Design → Frontend → Backend)
Track C: Documentation (Planning → Writing → Review)

Buffer Time Strategy

BUFFER ALLOCATION:
├── Technical Risks: 15% additional time for complex tasks
├── Learning Curve: 20% extra for new technologies
├── Integration Issues: 10% buffer for system integration
└── Testing & Fixes: 25% time for testing and bug fixes

Quality Assurance Logic

Definition of Done (DoD)

FOR EACH FEATURE:
├── ✓ Code completed and reviewed
├── ✓ Unit tests written and passing
├── ✓ Integration testing completed
├── ✓ User acceptance criteria met
├── ✓ Documentation updated
├── ✓ Performance benchmarks met
└── ✓ Security checks passed

Review and Feedback Loops

FEEDBACK CYCLES:
Daily: Internal team sync (15 min)
Weekly: Progress review and planning (1 hour)
Bi-weekly: Stakeholder demo and feedback (30 min)
Phase-end: Comprehensive review and retrospective (2 hours)

Communication Plan Logic

Information Flow

INTERNAL COMMUNICATION:
Team Members ↔ Daily standups, Slack/WhatsApp
↔ Code reviews, GitHub
↔ Design discussions, shared docs

EXTERNAL COMMUNICATION:
Team → Mentors: Weekly progress reports
Team → Users: Testing feedback sessions
Team → Stakeholders: Milestone demonstrations

Documentation Strategy

LIVING DOCUMENTS (Updated continuously):
├── Project README
├── API documentation
├── User manual
└── Deployment guide

MILESTONE DOCUMENTS (Version-controlled):
├── Requirements specification
├── Architecture design
├── Test plans
└── Final project report

Continuous Improvement Logic

Retrospective Framework

WHAT WORKED WELL?
- Effective collaboration methods
- Successful technical decisions
- Productive work patterns
- Useful tools and processes

WHAT COULD BE IMPROVED?
- Communication gaps
- Technical challenges
- Resource utilization
- Time management

ACTION ITEMS FOR NEXT ITERATION:
- Process improvements
- Tool changes
- Skill development
- Risk mitigation

Document prepared by Team LTVIP2025TMID42969