4. Project Planning Phase

Project Timeline

Phase 1: Data Collection & Analysis

- Collect Australian weather dataset
- Perform exploratory data analysis
- Identify patterns and correlations
- Handle missing values
- Feature selection

Phase 2: Model Development

- Data preprocessing pipeline
- Feature engineering
- Train multiple ML models
- Model evaluation and comparison
- Hyperparameter tuning
- Select best performing model

Phase 3: Web Application Development

- Flask application setup
- Frontend design and development
- Backend API development
- Model integration
- Form validation implementation

Phase 4: Testing & Refinement

- Unit testing
- Integration testing
- User acceptance testing
- Bug fixes
- Performance optimization

Phase 5: Documentation & Deployment

- Code documentation
- User manual creation
- README preparation
- Deployment setup
- Final review

Resource Requirements

Human Resources

- **Data Scientist**: Model development and evaluation
- **Backend Developer**: Flask application development
- **Frontend Developer**: UI/UX design and implementation
- **QA Engineer**: Testing and quality assurance

Technical Resources

- **Development Environment**: Python 3.x, Jupyter Notebook
- **Libraries**: Flask, Scikit-learn, Pandas, NumPy, XGBoost
- **Tools**: Git, VS Code, Anaconda
- **Hosting**: Local server / Cloud platform (AWS, Heroku, Azure)

Data Resources

- Australian Weather Dataset (WeatherAUS.csv)
- Historical meteorological data
- Training and testing datasets

Task Breakdown

1. Data Preparation Tasks

- [ ] Download and load dataset
- [ ] Explore data structure and statistics
- [ ] Visualize data distributions
- [ ] Handle missing values
- [ ] Encode categorical variables

- [ ] Split data into train/test sets

2. Model Development Tasks

- [ ] Implement preprocessing pipeline
- [ ] Train Random Forest model
- [ ] Train Decision Tree model
- [ ] Train KNN model
- [ ] Train XGBoost model
- [ ] Evaluate model performance
- [ ] Generate confusion matrices
- [ ] Create ROC curves
- [ ] Select best model
- [ ] Serialize model artifacts

3. Backend Development Tasks

- [ ] Set up Flask project structure
- [ ] Create route handlers
- [ ] Implement data validation
- [ ] Integrate ML model
- [ ] Implement preprocessing pipeline
- [ ] Error handling implementation
- [ ] Testing backend functionality

4. Frontend Development Tasks

- [ ] Design input form layout
- [ ] Implement responsive grid
- [ ] Create result pages (chance.html, noChance.html)
- [ ] Add CSS styling
- [ ] Implement animations (raindrops)
- [ ] Form validation (client-side)
- [ ] Cross-browser testing

5. Integration Tasks

- [ ] Connect frontend with backend
- [ ] Test end-to-end flow
- [ ] Handle edge cases
- [ ] Optimize performance

- [ ] Security testing

6. Documentation Tasks

- [ ] Write code comments
- [ ] Create README.md
- [ ] Document API endpoints
- [ ] Create user guide
- [ ] Prepare project documentation

7. Deployment Tasks

- [ ] Set up hosting environment
- [ ] Configure server
- [ ] Deploy application
- [ ] Test deployed version
- [ ] Monitor performance

Risk Management

Risk 1: Model Accuracy

- **Risk**: Model accuracy below acceptable threshold
- **Impact**: High
- **Probability**: Medium
- **Mitigation**:
  - Try multiple algorithms
  - Perform hyperparameter tuning
  - Use ensemble methods
  - Collect more training data

Risk 2: Data Quality

- **Risk**: Insufficient or poor quality data
- **Impact**: High
- **Probability**: Low
- **Mitigation**:
  - Validate data sources
  - Implement robust data cleaning
  - Handle missing values appropriately

Risk 3: Performance Issues

- **Risk**: Slow prediction response time
- **Impact**: Medium
- **Probability**: Low
- **Mitigation**:
  - Optimize preprocessing pipeline
  - Use efficient data structures
  - Implement caching
  - Load models at startup

Risk 4: Browser Compatibility

- **Risk**: UI issues on different browsers
- **Impact**: Medium
- **Probability**: Medium
- **Mitigation**:
  - Use standard HTML/CSS
  - Test on multiple browsers
  - Implement fallbacks

Risk 5: Deployment Challenges

- **Risk**: Issues during deployment
- **Impact**: Medium
- **Probability**: Medium
- **Mitigation**:
  - Test in staging environment
  - Document deployment steps
  - Have rollback plan

Quality Assurance Plan

Testing Strategy

1. **Unit Testing**: Test individual functions
2. **Integration Testing**: Test component interactions
3. **System Testing**: Test complete system
4. **User Acceptance Testing**: Validate with end users

Test Cases

- Valid input scenarios
- Invalid input scenarios
- Boundary value testing
- Error handling testing
- Performance testing
- Cross-browser testing

Success Metrics

- Model accuracy > 85%
- Response time < 3 seconds
- Zero critical bugs
- User satisfaction > 80%
- Code coverage > 70%

Budget Estimation

Development Costs

- Data Scientist: 8 weeks
- Backend Developer: 6 weeks
- Frontend Developer: 4 weeks
- QA Engineer: 2 weeks

Infrastructure Costs

- Cloud hosting: Monthly subscription
- Domain name: Annual fee
- SSL certificate: Annual fee

Tools & Software

- Development tools: Free (VS Code, Git)
- Python libraries: Free (Open source)
- Dataset: Free (Public dataset)

Communication Plan

Stakeholder Updates

- Weekly progress reports
- Bi-weekly demos
- Monthly steering committee meetings

Team Communication

- Daily stand-ups
- Weekly team meetings
- Slack/Teams for instant messaging
- Git for code collaboration

Deliverables

1. Trained ML model with >85% accuracy
2. Functional web application
3. Source code repository
4. Technical documentation
5. User manual
6. Deployment guide
7. Test reports
8. Project presentation