Software Development Plan

Real-World AR ChatGPT for Farmers

1. Executive Summary

Project Name: Real-World AR ChatGPT for Farmers
Project Type: WebAR Agricultural Assistant Application
Duration: 12 weeks (3 sprints of 4 weeks each)
Team Size: 6-8 developers
Target Platform: Mobile web browsers (iOS Safari, Android Chrome)

2. Project Objectives

Primary Goals

- Develop a WebAR application that assists farmers with real-time agricultural insights using NASA satellite data
- Implement RAG-based conversational AI for farming-specific queries
- Create gamified engagement through a Tamagotchi-style crop companion
- Deliver location-aware crop recommendations and irrigation guidance

Success Metrics

- Ground detection accuracy: >95% within 3 seconds
- API response time: <2.5 seconds p95
- Chat response accuracy: >90% for agriculture-related queries
- Cache hit rate: >70% for coordinate-based queries
- User engagement: Daily active usage >60% during growing season

3. Technical Architecture

3.1 Frontend Stack

- **WebAR Framework:** WebXR API with Three.js/A-Frame
- **UI Framework:** React 18 with TypeScript
- **State Management:** Redux Toolkit
- **Voice Processing:** Web Audio API with push-to-talk
- **Styling:** Tailwind CSS

3.2 Backend Stack

- **API Gateway:** Node.js with Express
- **Authentication:** JWT with bearer tokens
- **Data Processing:** Python services for NASA data fusion
- **RAG Pipeline:** LangChain with PostgreSQL + pgvector
- **Caching:** Redis with TTL management
- **Database:** PostgreSQL for persistent storage

3.3 External Integrations

- **NASA APIs:** SMAP, MODIS, Landsat/Sentinel
- **Weather Services:** OpenWeather API
- **Voice Services:** Web Speech API / Google Cloud Speech-to-Text
- **LLM Provider:** OpenAI GPT-4 or equivalent

4. Development Phases

Phase 1: Foundation (Weeks 1-4)

- WebAR ground plane detection implementation
- GPS location capture and permission handling
- Basic API infrastructure setup
- NASA data adapter development
- Database schema implementation

Phase 2: Core Features (Weeks 5-8)

- RAG pipeline implementation
- Voice input/output integration
- Crop recommendation engine
- Irrigation timing algorithm
- Real-time data fusion service

Phase 3: Enhancement & Polish (Weeks 9-12)

- Tamagotchi companion implementation
- Alert system and notifications

- Performance optimization
- Caching strategy refinement
- UI/UX polish and testing

5. Sprint Planning

Sprint 1 (Weeks 1-4)

Goal: Establish core AR functionality and data pipeline

User Stories:

1. Ground detection in WebAR (8 points)
2. Location permission and GPS capture (3 points)
3. NASA data fetch for coordinates (13 points)
4. Basic crop recommendation (8 points)
5. Irrigation timing calculation (5 points)

Deliverables:

- Working WebAR ground detection
- Location-aware data fetching
- Basic recommendation engine

Sprint 2 (Weeks 5-8)

Goal: Implement conversational AI and voice features

User Stories:

1. RAG content indexing (8 points)
2. Chat interface implementation (5 points)
3. Voice transcription integration (8 points)
4. Text-to-speech responses (5 points)
5. Citation system (3 points)

Deliverables:

- Functional chat with RAG
- Voice interaction capability

- Grounded responses with citations

Sprint 3 (Weeks 9-12)

Goal: Gamification and production readiness

User Stories:

1. Crop companion avatar system (8 points)
2. Alert rules engine (5 points)
3. Metrics dashboard (3 points)
4. Performance optimization (8 points)
5. Offline fallback (5 points)

Deliverables:

- Complete gamification features
- Production-ready application
- Monitoring and analytics

6. Resource Allocation

Team Structure

- **Project Lead:** 1 (20% allocation)
- **Frontend Developers:** 2 (WebAR specialist, UI developer)
- **Backend Developers:** 2 (API developer, Data engineer)
- **ML Engineer:** 1 (RAG and recommendation systems)
- **DevOps Engineer:** 1 (Infrastructure and monitoring)
- **QA Engineer:** 1 (Testing and validation)

Infrastructure Requirements

- **Development Environment:**
  - Local development servers
  - Staging environment on AWS/GCP
  - CI/CD pipeline with GitHub Actions
- **Production Environment:**
  - Load-balanced API servers

- Redis cluster for caching
- PostgreSQL with read replicas
- CDN for static assets

7. Risk Management

Technical Risks

Risk	Probability	Impact	Mitigation
WebAR compatibility issues	Medium	High	Test on multiple devices early, implement fallbacks
NASA API rate limits	High	Medium	Implement aggressive caching, coordinate bucketing
GPS accuracy in rural areas	Medium	Medium	Allow manual location entry, use area approximation
Voice recognition accuracy	Low	Low	Provide text input alternative
Data latency	Medium	High	Pre-warm cache, implement progressive loading

Mitigation Strategies

1. **Caching Strategy:** Implement multi-tier caching with Redis and CDN
2. **Graceful Degradation:** Fallback to cached or estimated data
3. **Progressive Enhancement:** Start with basic features, add complexity
4. **Error Boundaries:** Implement comprehensive error handling

8. Quality Assurance

Testing Strategy

- **Unit Testing:** Jest with 80% code coverage target
- **Integration Testing:** API endpoint testing with Supertest
- **E2E Testing:** Cypress for critical user flows
- **Performance Testing:** Lighthouse scores >85
- **AR Testing:** Manual testing on target devices

Acceptance Criteria Standards

- All user stories must have clear acceptance criteria
- Each feature requires testing on iOS and Android
- Performance benchmarks must be met before deployment
- Security scanning required for each release

9. Development Standards

Code Quality

- **Version Control:** Git with feature branch workflow
- **Code Review:** Mandatory PR reviews by 2 developers
- **Documentation:** JSDoc for all public APIs
- **Linting:** ESLint and Prettier enforcement
- **Commit Standards:** Conventional Commits specification

Security Requirements

- JWT token expiration: 24 hours
- HTTPS enforcement for all endpoints
- Input validation and sanitization
- Rate limiting per user/IP
- Secrets management via environment variables

10. Deployment Strategy

Environments

1. **Development:** Local development with Docker
2. **Staging:** Mirrors production, updated on merge to main
3. **Production:** Blue-green deployment with rollback capability

Release Process

1. Feature branch merged to develop
2. Automated tests run in CI
3. Deploy to staging for QA
4. Production deployment with monitoring
5. Post-deployment validation

11. Monitoring & Maintenance

Key Metrics

- API latency (p50, p95, p99)

- Error rates by endpoint
- Cache hit ratios
- Active user sessions
- NASA API usage

Maintenance Windows

- Weekly dependency updates
- Monthly security patches
- Quarterly performance reviews
- Annual architecture review

12. Success Criteria

MVP Completion Checklist

- WebAR ground detection functional on target browsers
- Location-based data retrieval operational
- Chat interface with RAG working
- Voice input/output implemented
- Basic gamification features active
- Performance targets met
- Documentation complete

Post-Launch Metrics

- User adoption rate: >100 users in first month
- User retention: >40% weekly active users
- System reliability: >99.5% uptime
- Response accuracy: >85% satisfaction rate

13. Communication Plan

Stakeholder Updates

- Weekly sprint progress reports
- Bi-weekly stakeholder demos
- Monthly steering committee reviews

Team Communication

- Daily standup meetings
- Weekly technical deep-dives
- Sprint retrospectives
- Slack for async communication

14. Budget Considerations

Development Costs

- Team salaries (12 weeks)
- Infrastructure costs (AWS/GCP)
- Third-party API costs (NASA data, LLM)
- Testing devices and tools

Ongoing Costs

- Server hosting
- API usage fees
- Maintenance and support
- Future enhancements

15. Timeline Summary

Week 1-2: Environment setup, WebAR prototype
Week 3-4: NASA data integration, basic API
Week 5-6: RAG implementation, chat interface
Week 7-8: Voice features, recommendation engine
Week 9-10: Gamification, performance optimization
Week 11: Integration testing, bug fixes
Week 12: Deployment preparation, documentation

Appendices

A. Technology Stack Details

- Complete list of dependencies
- Version requirements
- Compatibility matrix

B. API Documentation

- Endpoint specifications
- Request/response schemas
- Error codes

C. Database Schema

- Entity relationship diagrams
- Table definitions
- Index strategies