Software Requirements Document

Plant Operations Management System (POMS)

Company: American SpiralWeld Pipe Company, LLC
Prepared for: Casey, Plant Manager
Version: 1.0
Date: June 19, 2025

1. Project Overview

The Plant Operations Management System (POMS) is a web-based application designed to optimize and digitize the operational workflows of the American SpiralWeld Pipe factory. It aims to provide real-time visibility into production, inventory, maintenance, QA, and logistics, thereby increasing efficiency and reducing manual errors.

2. System Architecture Overview

Layer	Technology Stack Suggestion
Frontend	React.js + Tailwind CSS
Backend	Node.js + Express or ASP.NET Core
Database	PostgreSQL or Microsoft SQL Server
Hosting	AWS / Azure
AI Integration	Python microservice via API (optional)
Crypto Integration	Blockchain-based material tracking (optional)

3. Frontend Requirements

3.1 General

- Responsive design (desktop, tablet, mobile)
- Role-based navigation menus
- Dark mode (optional)

3.2 Modules

Dashboard

- Real-time metrics (production, downtime, inventory, QA issues)
- Charts and KPIs (output per shift, line efficiency)
- Alerts and notifications

Work Order Management

- Create/edit/delete work orders
- Assign operators and machines
- Status tracking: *Pending → In Progress → Completed*

Inventory Management

- View current raw material and finished goods stock
- Material usage logs
- Reorder alerts

QA & Inspection

- Digital inspection forms
- Upload weld photos/certificates
- Tag non-conformance issues

Maintenance

- Maintenance calendar
- Log repair tickets and resolutions
- Maintenance history per machine

Logistics

- Outbound shipment scheduling
- Pipe bundle tracking
- Generate packing list / BOL

User Management

- Add/remove users
- Set roles: Admin, Operator, QA, Maintenance, Logistics
- Permission-based access

---

4. Backend Requirements

4.1 API

- RESTful API endpoints
- JWT-based authentication
- Role-based authorization
- Rate limiting & logging

4.2 Business Logic

Work Order Engine

- Auto-assign based on capacity
- Status transitions
- Historical tracking

Inventory Engine

- FIFO material consumption
- Auto-decrement on usage
- Threshold notifications

QA System

- Auto-flag repeated defects
- Batch-wise QA results

Maintenance Module

- Trigger preventive maintenance based on hours or usage
- Escalate unresolved tickets

---

5. Database Requirements

5.1 Core Tables

- Users
- Roles & Permissions
- Work Orders
- Machines
- Materials
- Inventory Logs
- QA Reports
- Maintenance Logs
- Shipments
- Documents

5.2 Relationships

- One-to-many: Work Order → QA Reports
- Many-to-many: Users ↔ Roles
- One-to-many: Machines → Maintenance Logs

5.3 Data Integrity

- Foreign key constraints
- Cascading deletes where appropriate
- Timestamps on all records

---

6. AI Integration (Optional/Future Phase)

Use Cases

- **Predictive Maintenance**: Use machine data to predict failures
- **Defect Detection**: Image-based weld inspection using CV
- **Production Forecasting**: Time-series analysis of output

Architecture

- Python-based AI services (Flask or FastAPI)
- REST API integration with main app
- Model training on historical data

---

7. Crypto / Blockchain Integration (Optional/Future Phase)

Use Cases

- **Material Traceability**: Immutable tracking of coil-to-pipe transformation
- **Digital Certificates**: Smart contracts for QA documents and test results

Suggested Tech

- Hyperledger Fabric or Ethereum (private network)
- Smart contracts for pipe lifecycle events
- QR code linking to blockchain record

---

8. Testing & QA Requirements

- Unit tests (Jest, Mocha, xUnit)
- Integration tests
- End-to-end tests (Cypress or Playwright)
- Load & stress testing
- Manual QA scripts for production floor validation

---

9. Security Requirements

- HTTPS-only
- JWT or OAuth2 Authentication
- Role-based Access Control (RBAC)
- Audit logs (user activity, data changes)
- Secure file uploads (virus scan, size limits)

---

10. Analytics & Reporting

- Built-in dashboard reporting
- Export to PDF/Excel
- Admin-level usage reports
- Optional: Integration with Power BI, Grafana, or Tableau

---

11. Deployment & DevOps

- CI/CD pipeline (GitHub Actions, Jenkins, or Azure DevOps)
- Dockerized microservices
- Environment config: dev → staging → prod
- Daily backups
- Monitoring via Prometheus & Grafana

---

12. Project Milestones (Example)

Phase	Timeline	Deliverables
Discovery & UI/UX	3 weeks	Wireframes, tech stack decision
MVP Development	6-8 weeks	Dashboard, Work Orders, Inventory
QA & Feedback	2 weeks	Closed beta testing
Phase 2	Later	AI + Blockchain modules