# Dark Horse 3PL Platform - Architecture Summary

**Document Version:** 1.0  
**Date:** September 18, 2025  
**Project:** Dark Horse Third-Party Logistics Platform  
**Company:** Dark Horse / Strugbits

## Executive Summary

Dark Horse is a comprehensive Third-Party Logistics (3PL) platform that seamlessly integrates with Salla e-commerce stores to provide end-to-end fulfillment services. The platform connects merchants with warehouses through a sophisticated microservices architecture built on Node.js, TypeScript, and modern cloud technologies.

### Key Business Value

* **Merchant Integration**: Seamless Salla store connectivity with automated product sync
* **Operational Efficiency**: Streamlined warehouse workflows (Pick → Pack → Ship)
* **Role-Based Access**: Hierarchical permission system for secure operations
* **Real-Time Tracking**: Live inventory and order status updates
* **Scalable Architecture**: Microservices design for rapid growth

## Business Flow Overview

### High-Level Process

🏪 Salla Store → 🏢 Dark Horse Platform → 🏭 Fulfillment Centers → 📦 Customer

1. **Merchant Onboarding**: Admin invites merchants (invitation-only)
2. **Store Connection**: OAuth integration with Salla
3. **Product Sync**: Automatic catalog synchronization
4. **Inventory Assignment**: Merchants assign stock to fulfillment centers
5. **Order Processing**: Automated order routing and fulfillment
6. **Real-Time Updates**: Inventory and status synchronization

### User Roles & Responsibilities

| Role | Access Level | Key Responsibilities |
| --- | --- | --- |
| **Admin** | Platform-wide | Create Directors, system settings, integrations |
| **Director** | Multi-warehouse | Inventory oversight, create Managers, fulfillment coordination |
| **Manager** | Single warehouse | Operations management, create Workers, local inventory |
| **Workers** | Task-specific | Receiving, Picking, Packing, Shipping operations |

## System Architecture

### Technology Stack

#### Backend Services

* **Runtime**: Node.js with TypeScript
* **Framework**: Express.js microservices
* **Database**: PostgreSQL with Prisma ORM
* **Cache**: Redis for session and data caching
* **Authentication**: JWT tokens with role-based access
* **Monorepo**: Nx workspace for unified development

#### Frontend Applications

* **Framework**: Next.js 15 with TypeScript
* **Styling**: Tailwind CSS + Shadcn/ui components
* **State Management**: Redux Toolkit
* **Deployment**: Separate repositories per portal

#### Infrastructure

* **Cloud**: AWS (EC2, RDS, ElastiCache, S3)
* **Containers**: Docker with Docker Compose
* **Load Balancing**: AWS Application Load Balancer
* **Monitoring**: CloudWatch + structured logging
* **CI/CD**: GitHub Actions with ECR

### Microservices Architecture

| Service | Port | Status | Responsibility |
| --- | --- | --- | --- |
| **API Gateway** | 8080 | ✅ Working | Request routing, authentication, rate limiting |
| **Auth Service** | 6001 | ✅ Working | User management, JWT tokens, invitations |
| **User Service** | 6002 | 🔄 Planned | Role management, permissions, hierarchy |
| **Salla Integration** | 6003 | 🔄 Planned | OAuth, product sync, order webhooks |
| **Inventory Service** | 6004 | 🔄 Planned | Stock tracking, adjustments, transfers |
| **Order Management** | 6005 | 🔄 Planned | Order processing, fulfillment routing |
| **Warehouse Service** | 6006 | 🔄 Planned | Operations workflow, task management |
| **Shipping Service** | 6007 | 🔄 Planned | Carrier integration, tracking |
| **Notification Service** | 6008 | 🔄 Planned | Email, SMS, push notifications |
| **Analytics Service** | 6009 | 🔄 Planned | Reporting, business intelligence |

## Salla Integration

### About Salla

Salla is a leading Middle Eastern e-commerce platform enabling merchants to create and manage online stores. Dark Horse integrates to provide seamless fulfillment services.

### Integration Features

* **OAuth 2.0 Authentication**: Secure store connection
* **Real-time Product Sync**: Automatic catalog updates
* **Order Webhooks**: Instant order processing
* **Inventory Synchronization**: Bi-directional stock updates
* **Payment Tracking**: COD/Prepaid order identification

### API Integration Points

| Integration | Purpose | Frequency |
| --- | --- | --- |
| **Product Catalog** | Sync merchant products | Real-time + Daily batch |
| **Order Webhooks** | Receive new orders | Real-time |
| **Inventory Updates** | Update stock levels | After each fulfillment |
| **Order Status** | Confirm shipments | Upon delivery |

## Frontend Portal Architecture

### Multi-Portal Design

The platform consists of three main frontend applications built with Next.js 15:

#### 1. Merchant Portal

* **Dashboard**: Revenue, orders, inventory overview
* **Product Management**: Salla sync, warehouse assignment
* **Order Tracking**: Real-time fulfillment status
* **Analytics**: Performance metrics and insights

#### 2. Admin Portal (Dark Horse Staff)

* **User Management**: Create and manage Directors
* **System Settings**: Platform configuration
* **Integration Monitoring**: Salla and platform connections
* **Global Analytics**: Cross-warehouse metrics

#### 3. Operations Portal (Warehouse Staff)

**Role-based interfaces**: - **Receivers**: Incoming inventory management - **Pickers**: Order picking tasks and workflows - **Packers**: Packing and preparation - **Shippers**: Shipping and tracking updates - **Managers**: Full warehouse oversight

## Security & Permissions

### Authentication Model

* **No Self-Registration**: Invitation-only access system
* **Hierarchical Invitations**: Admin → Director → Manager → Worker
* **JWT Security**: Access tokens (15 min) + Refresh tokens (7 days)
* **Role-Based Access Control**: Granular permissions per role

### Security Measures

* **API Rate Limiting**: Role-based request throttling
* **Data Encryption**: AES-256 for sensitive data
* **SQL Injection Protection**: Prisma ORM parameterized queries
* **HTTPS Enforcement**: SSL/TLS for all communications
* **Audit Logging**: Complete activity tracking

## Deployment Strategy

### AWS Cloud Architecture

* **Compute**: EC2 instances with Auto Scaling Groups
* **Database**: RDS PostgreSQL Multi-AZ
* **Cache**: ElastiCache Redis Cluster
* **Load Balancing**: Application Load Balancer
* **Storage**: S3 for files and backups
* **Monitoring**: CloudWatch metrics and logs

### Container Strategy

* **Docker Compose**: Multi-service orchestration
* **Zero-Downtime Deployment**: Rolling updates
* **Health Checks**: Automated service monitoring
* **Rollback Capability**: Quick version restoration

### Environment Configuration

| Environment | Infrastructure | Purpose |
| --- | --- | --- |
| **Development** | Local Docker | Developer workstation |
| **Staging** | Single EC2 (t3.large) | Testing and validation |
| **Production** | Multi-EC2 Auto Scaling | High availability operations |

## Data Flow Diagrams

### Order Fulfillment Flow

Customer → Salla Store → Dark Horse OMS → Warehouse → Shipping → Customer  
 ↓ ↓ ↓ ↓ ↓  
 Order Webhook Route to Pick/Pack Update  
Created Received Nearest /Ship Tracking  
 Warehouse

### Inventory Management Flow

Merchant → Product Selection → Warehouse Assignment → Receiving → Stock Updates  
 ↓ ↓ ↓ ↓ ↓  
 Salla Dark Horse Physical Accept/ Real-time  
 Sync Platform Delivery Reject Updates

## Performance & Monitoring

### Key Metrics

* **Response Times**: API endpoint performance (<200ms average)
* **Throughput**: Requests per second handling
* **Error Rates**: Service-level error tracking (<1% target)
* **Business KPIs**: Orders processed, inventory accuracy

### Monitoring Stack

* **Application Monitoring**: CloudWatch metrics and alarms
* **Log Management**: Structured JSON logging with centralized collection
* **Health Checks**: Automated service health verification
* **Alerting**: Real-time notifications for critical issues

## Development Workflow

### Nx Monorepo Benefits

* **Code Sharing**: Reusable libraries across services
* **Build Optimization**: Incremental builds and caching
* **Testing Integration**: Unified testing across services
* **Dependency Management**: Centralized package management

### Development Process

1. **Feature Planning**: Service design and API specification
2. **Implementation**: TypeScript development with Prisma
3. **Testing**: Unit, integration, and end-to-end testing
4. **Code Review**: Peer review and quality assurance
5. **Deployment**: Automated CI/CD pipeline

## Conclusion

The Dark Horse 3PL Platform represents a modern, scalable solution for e-commerce fulfillment operations. Key strengths include:

### Technical Excellence

* **Microservices Architecture**: Independent, scalable services
* **Modern Technology Stack**: Next.js 15, Node.js, TypeScript, AWS
* **Security-First Design**: Comprehensive security measures
* **Cloud-Native**: AWS-optimized for reliability and scale

### Business Value

* **Operational Efficiency**: Streamlined warehouse workflows
* **Merchant Experience**: Seamless Salla integration
* **Scalability**: Growth-ready architecture
* **Cost Optimization**: Efficient resource utilization

### Competitive Advantages

* **Deep Salla Integration**: Native Middle Eastern e-commerce support
* **Role-Based Security**: Hierarchical access control
* **Real-Time Operations**: Live inventory and order tracking
* **Rapid Development**: Nx monorepo efficiency

The platform foundation supports future enhancements while maintaining stability, security, and performance standards essential for 3PL operations.