Konibui E-commerce Platform

Product Requirements Document

Version 1.0

Date: July 20, 2025

---

1. Executive Summary

1.1 Product Vision

Konibui is a specialized e-commerce platform designed for the Pokémon Trading Card Game (TCG) community. The platform combines the efficiency of a modern web application with the nostalgic charm of a Japanese convenience store (konbini), creating an intuitive marketplace for TCG products including single cards, booster packs, booster boxes, and supplies.

1.2 Strategic Objectives

- Create a premium, specialized marketplace for Pokémon TCG products
- Implement a local-first, server-rendered architecture for optimal performance and data control
- Establish a scalable foundation ready for AI-driven automation and future enhancements
- Deliver a unique user experience that differentiates from generic e-commerce platforms

1.3 Success Metrics

- User engagement: Average session duration >3 minutes
- Conversion rate: >2.5% from visitor to customer
- Technical performance: Page load times <2 seconds
- Inventory accuracy: 99.5% real-time stock synchronization

---

2. Product Overview

2.1 Core Value Proposition

Konibui transforms the traditional TCG shopping experience by providing:

- **Specialized Catalog Management**: Advanced product modeling for TCG-specific variants (condition-based pricing)
- **Real-time Inventory**: Atomic stock management preventing overselling
- **Buyback System**: Integrated platform for users to sell cards back to the store
- **Future-Ready Architecture**: MCP integration for AI-driven workflows

2.2 Target Market Segments

Segment	Demographics	Primary Needs	Features Priority
TCG Players	Teens/Adults, competitive players	Specific singles, supplies	Advanced search, condition filtering
Collectors	All ages, high-value purchases	Rare cards, sealed products	Authentication, detailed conditions
Gift Buyers	Parents, friends	Popular sets, starter products	Simplified browsing, recommendations
Resellers	Adult entrepreneurs	Bulk purchases, market data	Buyback system, analytics

---

3. Technical Architecture Requirements

3.1 Development Environment

Requirement: Local-first development environment with Docker containerization

Implementation:

- Multi-container Docker architecture (app, db, webserver, db-admin)
- Single `docker-compose up` command for complete environment setup
- Persistent data volumes for development consistency
- Environment variable management through `.env` files

Acceptance Criteria:

- New developers can set up complete environment in <5 minutes
- Environment consistency across all development machines
- Zero manual configuration steps required

3.2 Backend Technology Stack

Framework: Laravel 12 with PHP 8.2+

Core Requirements:

- Strict adherence to Laravel best practices and MVC architecture
- Service layer pattern for business logic encapsulation
- Form Request classes for all input validation
- Policy-based authorization system

Performance Optimizations:

- Laravel 12 asynchronous caching implementation
- Route and configuration caching for production
- Octane-ready architecture for future scaling

3.3 Database Architecture

Database: MySQL 8.x with advanced feature utilization

Advanced Database Features:

- **Database triggers** for atomic inventory management
- **Stored procedures** for complex transaction processing
- **MySQL roles** for defense-in-depth security
- **Automated archiving** for performance optimization

Data Integrity Requirements:

- All inventory updates must be atomic with order creation
- Zero tolerance for overselling scenarios
- Audit trail for all financial transactions

3.4 Frontend Technology Stack

Primary: Livewire 3 with server-side rendering **Styling**: TailwindCSS with TailwindUI and Flowbite components **Client-side interactions**: Alpine.js for ephemeral UI state

Performance Requirements:

- Computed properties for all derived data
- Deferred model binding for form inputs
- Lazy loading for non-critical components
- Strategic Alpine.js usage to minimize server round-trips

---

4. Functional Requirements

4.1 User Management System

4.1.1 Role-Based Access Control

Roles:

- **Customer**: Browse, purchase, view own orders, submit buyback requests
- **Employee**: Manage orders, process buybacks, view inventory
- **Admin**: Full system access including user management and analytics

Authentication:

- Laravel Breeze implementation with Livewire stack
- Email verification required
- Password reset functionality
- Session management with CSRF protection

4.1.2 User Registration Flow

1. User provides name, email, password
2. System sends email verification
3. User confirms email to activate account
4. Default "Customer" role assignment
5. Welcome email with platform introduction

4.2 Product Catalog System

4.2.1 TCG-Specific Data Model

Card Hierarchy:

```
None
Cards (Base Information)
├── Name, Set, Rarity, Collector Number
└── Products (Sellable Variants)
    ├── Condition (NM, LP, MP, HP, DMG)
    ├── Price per condition
    ├── SKU
    └── Individual inventory tracking
```

4.2.2 Product Categories

- **Single Cards**: Individual TCG cards with condition variants
- **Booster Packs**: Sealed card packs
- **Booster Boxes**: Cases containing multiple booster packs
- **TCG Supplies**: Sleeves, deck boxes, playmats, accessories

4.2.3 Inventory Management

Real-time Stock Tracking:

- Database triggers automatically decrement stock on order creation
- Stock validation prevents overselling
- Admin notifications for low stock items
- Automatic inventory reconciliation

Stock Display Logic:

- "In Stock" for quantities > 0
- "Out of Stock" for zero quantity
- "Low Stock" warning for quantities < 5

4.3 Shopping and Checkout System

4.3.1 Shopping Cart

Implementation Requirements:

- Database-backed cart for registered users
- Session-based cart for guests (with migration on login)
- CartService class for centralized logic
- Real-time price updates
- Quantity validation against current stock

4.3.2 Checkout Process

Multi-step Flow:

1. **Cart Review**: Final item verification and quantity adjustment
2. **Shipping Information**: Address collection and validation
3. **Payment Method**: Initial support for Cash on Delivery (COD)
4. **Order Confirmation**: Final review before submission
5. **Order Processing**: Database transaction with inventory update

Payment Integration:

- Phase 1: Cash on Delivery only
- Phase 2: PayPal integration (architecture ready)
- Extensible PaymentService for future gateway additions

4.4 Order Management System

4.4.1 Order Processing Workflow

Order States: Pending → Processing → Shipped → Delivered → Completed **Cancellation**: Available until "Shipped" status **Returns**: Available within 30 days of delivery

4.4.2 Order Data Structure

- Order header with user, payment method, total
- Order items with snapshot pricing and quantities
- Shipping address and contact information
- Transaction logs for audit trail

4.5 Buyback System

4.5.1 Submission Process

User Workflow:

1. User uploads images of cards for sale
2. Provides card details (name, set, condition, quantity)
3. Submits for evaluation
4. Receives offer from staff
5. Accepts/rejects offer
6. Payment processing for accepted offers

4.5.2 Administrative Workflow

Staff Process:

1. Review submitted images and details
2. Evaluate condition and market value
3. Generate offer with expiration date
4. Process payment upon acceptance
5. Update inventory with acquired cards

4.5.3 Status Management

Buyback States: Submitted → In Review → Offer Made → Accepted/Rejected → Payment Sent → Completed

Implementation: PHP 8.1+ Enums for type-safe status management

4.6 Search and Discovery

4.6.1 Full-Text Search

Technology: Laravel Scout with TNTSearch driver for local-first approach

Search Capabilities:

- Card name search with fuzzy matching
- Set name and rarity filtering
- Advanced filters for price range, condition, availability
- Auto-complete suggestions
- Search result ranking by relevance

4.6.2 Browsing Features

- Category-based navigation
- Filter combinations (Set + Rarity + Condition)
- Sort options (Price, Name, Set, Rarity)
- Pagination with configurable page sizes

---

5. User Experience Requirements

5.1 Design Theme: Modern Konbini

Visual Identity:

- Clean, bright aesthetic mimicking Japanese convenience stores
- White/light gray backgrounds highlighting product colors
- Pokémon-inspired color palette (Red, Black, White, Grey)
- Rounded, friendly typography (Nunito for headings, Inter for body)

5.2 Interactive Elements

Product Cards:

- Prominent product images with hover effects
- Clear pricing and stock status
- Rarity indicators with custom icons
- One-click "Add to Cart" with visual feedback

Navigation:

- Simple category-based structure
- Persistent shopping cart indicator
- Mobile-first responsive design
- Breadcrumb navigation for deep browsing

5.3 Responsive Design Requirements

- Mobile-first approach with TailwindCSS utilities
- Breakpoints: Mobile (320px+), Tablet (768px+), Desktop (1024px+)
- Touch-friendly interface elements
- Fast loading on mobile networks

---

6. Technical Integration Requirements

6.1 Model Context Protocol (MCP) Integration

Purpose: Transform application into programmable platform for AI agents

Implementation:

- Laravel MCP server exposing business logic as tools/resources
- Authenticated access with existing authorization policies
- STDIO transport for development, production-ready transports available

Exposed Capabilities:

- Order status queries
- Product search and retrieval
- Inventory management
- Sales analytics
- Administrative task automation

6.2 Future AI Workflows

Customer Support:

- Automated order status inquiries
- Product recommendations based on purchase history
- FAQ responses with context awareness

Administrative Automation:

- Sales report generation
- Inventory analysis and reorder suggestions
- Fraud detection and prevention

---

7. Security Requirements

7.1 Data Protection

Input Validation:

- Server-side validation for all user inputs
- Laravel Form Requests for structured validation
- XSS prevention through Blade template escaping
- CSRF protection for all state-changing operations

File Upload Security:

- Strict MIME type and size validation
- Secure storage in non-public directories
- Authorized access through controller endpoints
- Virus scanning for uploaded images

7.2 Authentication Security

Password Policy:

- Minimum 8 characters with complexity requirements
- Bcrypt hashing with automatic salt generation
- Rate limiting for login attempts
- Session timeout for inactive users

Database Security:

- MySQL role-based access control
- Parameterized queries preventing SQL injection
- Regular security updates for all dependencies
- Encrypted storage for sensitive data

---

8. Performance Requirements

8.1 Response Time Targets

- **Page Load Time**: <2 seconds for first paint
- **Database Queries**: <100ms average response time
- **Search Results**: <500ms for full-text search
- **Image Loading**: Progressive loading with placeholders

8.2 Scalability Considerations

Current Architecture:

- Single server deployment with Docker
- MySQL with optimized queries and indexing
- File-based caching with Redis readiness

Future Scaling Path:

- Laravel Octane for request performance
- Redis cache cluster
- CDN integration for static assets
- Database read replicas

8.3 Monitoring Requirements

- Application performance monitoring
- Database query analysis
- Error tracking and alerting
- User behavior analytics

---

9. Implementation Timeline

Phase 1: Foundation (Weeks 1-4)

- Docker development environment setup
- Laravel 12 application scaffolding
- Database schema implementation with advanced features
- Basic user authentication and authorization

Phase 2: Core E-commerce (Weeks 5-8)

- Product catalog with TCG-specific modeling
- Shopping cart and checkout flow

- Order management system
- Payment integration (COD)

Phase 3: Advanced Features (Weeks 9-12)

- Buyback system implementation
- Full-text search with Scout/TNTSearch
- Admin panel and reporting
- Image handling and optimization

Phase 4: Polish and Integration (Weeks 13-16)

- UI/UX refinement and responsive design
- MCP server implementation
- Performance optimization
- Security audit and testing

---

10. Testing and Quality Assurance

10.1 Testing Strategy

Unit Testing:

- Laravel's built-in PHPUnit framework
- Model, service, and policy testing
- Database transaction testing
- 80%+ code coverage target

Integration Testing:

- End-to-end user workflows
- Payment processing validation
- Inventory synchronization testing
- Email and notification systems

10.2 Performance Testing

- Load testing for concurrent users
- Database performance under load
- Image upload and processing speed
- Search performance with large catalogs

10.3 Security Testing

- Vulnerability scanning
- Penetration testing
- Authentication bypass attempts
- SQL injection and XSS testing

---

11. Deployment and Operations

11.1 Production Environment

Infrastructure:

- Docker-based deployment for consistency
- NGINX reverse proxy with SSL termination
- MySQL with regular backups
- File storage with backup strategy

Environment Configuration:

- Environment-specific `.env` files
- Configuration caching for performance
- Log aggregation and monitoring
- Automated deployment pipeline

11.2 Backup and Recovery

- Daily database backups with 30-day retention
- File storage backup with versioning
- Disaster recovery procedures
- Regular restore testing

---

12. Success Criteria and KPIs

12.1 Technical Metrics

- **Uptime**: 99.5% availability target
- **Performance**: <2s page load times
- **Error Rate**: <0.1% server errors

- **Security**: Zero successful attacks

12.2 Business Metrics

- **Conversion Rate**: >2.5% visitor to customer
- **Average Order Value**: Track and optimize
- **Customer Retention**: >60% repeat customers
- **Buyback Participation**: >20% of customers use buyback

12.3 User Experience Metrics

- **Page Load Speed**: Core Web Vitals compliance
- **Mobile Experience**: >90% mobile usability score
- **Search Success Rate**: >95% searches return relevant results
- **Cart Abandonment**: <70% abandonment rate

Conclusion

This Product Requirements Document establishes a comprehensive blueprint for building Konibui as a specialized, high-performance e-commerce platform for the Pokémon TCG community. The technical architecture balances modern development practices with forward-thinking AI integration, ensuring both immediate functionality and long-term scalability.

The local-first development approach, combined with advanced database features and server-rendered UI, creates a robust foundation for a platform that can grow with its user base while maintaining exceptional performance and security standards.