Functional Specification: OpenBoxes to ShipStation Integration

# 1. Overview / Purpose

• Project Name: OpenBoxes–ShipStation Sync

• Date: Mar 7, 2023

• Objective:
 Seamlessly integrate OpenBoxes (WMS) with ShipStation (TMS) to automate order processing, reduce manual data entry, and improve overall order tracking.

• Stakeholders:
 Mickey (Project Manager): Oversees the project timelines and coordination.
 Minnie (Business Analyst): Gathers and documents business requirements.
 Donald (Technical Lead): Ensures technical feasibility and system integration.

# 2. Functional Requirements

* • User Login:
* • Description: Provide a secure login for warehouse staff and shipping managers to access the integration app.
* • Rationale: Ensure that only authorized personnel can access sensitive order data.
* • Order Synchronization:
* • Description: Automatically sync order data between OpenBoxes and ShipStation in near real-time when the orders are shipped via parcel services. Also sync tracking numbers on shipments.
* • Rationale: Eliminate manual data entry, reduce errors, and streamline order processing.
* • Notification System:
* • Description: Generate real-time alerts for order discrepancies or sync failures.
* • Rationale: Enable prompt action to address issues and maintain data accuracy.
* • Reporting:
* • Description: Provide dashboards that display real-time order statuses and historical trends.
* • Rationale: Help managers track performance and identify areas for improvement.

# 3. User Stories or Scenarios

* • User Story 1:
* As a warehouse manager, I want to view updated order statuses in OpenBoxes when orders have shipped in ShipStation so that I can manage inventory efficiently. I should also see the shipment tracking numbers.
* • User Story 2:
* As a shipping manager, I need automatic order updates so that I can schedule shipments without delay.
* • User Story 3:
* As an IT support specialist, I want to receive immediate notifications about data sync failures so that I can resolve issues before they affect operations.
* • User Story 4:
* As a warehouse order packer I need to be able to ship orders in ShipStation and have them automatically removed from the 'To Ship' queue, because the order has shipped.

# 4. Business Rules

• Only orders from the Tampa warehouse should be synced.

• Orders must have a valid shipping address and carrier preference selected.

• Orders with backordered items should be excluded from syncing.

• If an order is edited after sync, the app will not re-sync it automatically.

# 5. Field Definitions

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Source System | Required | Notes |
| Order ID | OpenBoxes | Yes | Unique identifier for the order |
| Customer Name | OpenBoxes | Yes | Full shipping name |
| Shipping Method | OpenBoxes | Yes | e.g., UPS Ground |
| Tracking Number | ShipStation | Yes | Saved back to OpenBoxes |
| Shipment Label URL | ShipStation | No | Optional storage in audit log |

# 6. External System Interactions (API, triggers)

• OpenBoxes API will be polled every 15 minutes for orders marked 'Ready to Ship'.

• ShipStation API is used to create new shipments and retrieve tracking numbers.

• A webhook will notify the system of new tracking info to update OpenBoxes.

# 7. Screenshots or Wireframes (if UI is involved)

• Login Screen: User enters secure credentials to access the app.

• Dashboard: Displays total synced orders, recent sync errors, and last sync timestamp.

• [Attach wireframe or UI sketch if available.]

# 8. Assumptions

• API credentials for both systems have already been provisioned.

• SKUs and customer data are consistent between both systems.

• Network latency between services is under 2 seconds on average.

# 9. Edge Cases and Error Handling

• Orders missing shipping address will be skipped and logged.

• Failed API calls will be retried once. If unsuccessful, they are logged and an email alert is sent.

• Duplicates will be avoided using Order ID checks against synced history.