# Module Specifications – Implementation

## Purpose and scope

This document defines the **detailed implementation specifications** for each core module in the PineCone Pro ERP/IMS solution. These module specifications translate the high‑level requirements, acceptance criteria and architecture decisions into implementable features and workflows. Each module description covers its purpose, functional capabilities, data entities, process flows, API and integration considerations, user roles and permissions, error handling, performance targets and non‑functional considerations. The objective is to provide development teams and stakeholders with a clear and consistent blueprint for building and testing the system.

## Conventions

* **CRUD notation:** Create, Read, Update and Delete operations.
* **Status codes:** Use standard HTTP response codes for APIs (e.g. 200 OK, 201 Created, 400 Bad Request, 404 Not Found, 500 Internal Server Error).
* **Time stamps:** All records must include CreatedAt, UpdatedAt and, where relevant, StatusChangedAt fields. Timestamps use ISO 8601 format and the system’s default timezone (America/New\_York).
* **Identifiers:** Use UUIDs for primary keys and correlation IDs. Reference numbers visible to users (e.g. order number, RMA number) can be sequential but must map back to the UUID.

## 1. Product Information Management (PIM)

### 1.1 Overview

The PIM module is the **central hub** for storing and governing all product‑related data across the organisation. It provides a unified repository for over 12 000 SKUs, including finished goods, kits/bundles, variants and hazardous products. PIM supports multi‑channel publishing (web store, Amazon, POS, B2B portal) and ensures that product data is consistent, complete and up to date. According to industry guidance, PIM consolidates product information from different departments and distributes it to multiple sales channels【303118104719862†L129-L164】. It helps manage complex product data, supports conversion optimisation and automates product creation processes【303118104719862†L129-L164】.

### 1.2 Functional capabilities

1. **Product catalogue management**
   * Maintain hierarchical categories (department → category → subcategory) and attributes for each product (e.g. dimensions, weight, unit of measure, hazardous classification).
   * Support variants (size, colour, packaging) via parent–child relationships.
   * Allow creation of **kits/bundles** that reference component products and calculate kit inventory availability from component stock levels.
   * Provide **rich descriptions**, images and digital assets for marketing channels. Include support for multiple languages.
2. **Data governance and validation**
   * Enforce **required attributes** for each product type (SKU, name, description, unit conversions, hazardous handling instructions). Required fields ensure completeness and accuracy when products are created or updated.
   * Validate units of measure and conversions (e.g. case → each) at the time of data entry to prevent inconsistent packaging definitions.
   * Maintain approval workflows: new products must be reviewed and approved by the Operations Manager and the E‑Commerce Manager before publishing to sales channels.
   * Provide audit trails capturing who created, updated or approved each record.
3. **Channel syndication**
   * Support scheduled and ad‑hoc **export of product data** to external channels (web store, Amazon, B2B portal). Use the events and API layer to publish ProductCreated, ProductUpdated and ProductDeleted events to downstream systems.
   * Allow mapping of internal attribute names to channel‑specific fields. Channel connectors transform and transmit data using each platform’s API.
   * Provide status tracking of each export job (success, error) and error remediation capabilities.
4. **Hazardous product management**
   * Identify products with hazardous classifications (DOT/IATA classes) and flag them accordingly.
   * Store required regulatory data such as **UN/NA numbers**, hazard classes, packaging instructions and hazard statements.
   * Prevent publication to channels that do not support hazmat items (e.g. certain carriers or Amazon categories).

### 1.3 Data entities

* **Product** – primary entity containing master attributes (ID, SKU, name, description, category, manufacturer, brand, status, dimensions, weight, hazardous flag, productType). Contains ValidFrom and ValidTo fields for versioning.
* **ProductAttribute** – key–value table for extensible attributes (e.g. colour, size, material). Supports multi‑language translations.
* **KitComponent** – mapping table linking kit parents to component products and their quantities.
* **ChannelMapping** – stores mapping definitions between internal product fields and external channel fields.
* **ProductRevision** – captures historical versions of product records for auditing and rollback.

### 1.4 Process flows

1. **Create/Update product**
   1. User (Purchasing Lead or Product Administrator) enters product details via UI or API.
   2. System validates required fields and unit conversions. If validations fail, returns 400 with error details.
   3. Upon submission, record status is set to *Draft*. Audit fields CreatedBy and CreatedAt are captured.
   4. Operations Manager and E‑Commerce Manager receive a **ProductApprovalTask**. They review and approve; once both approve, status changes to *Active*.
   5. System publishes a ProductCreated or ProductUpdated event to the event bus.
   6. Channel connectors detect the event and push product data to external systems.
2. **Publish to channel**
   1. Scheduled job or user triggers an export.
   2. System gathers active products, transforms fields via ChannelMapping and calls external channel APIs.
   3. On success, sets LastExportedAt timestamp; on failure, logs error and notifies administrators.

### 1.5 API and integration

* **Endpoints** (see API Specifications for full definitions): /api/v1/products for CRUD operations; /api/v1/products/{id}/publish to trigger channel export; events ProductCreated, ProductUpdated.
* **External integrations:** Web store, Amazon, B2B portal (via connectors) receive product data. Shipping & hazmat module consumes hazardous flags to enforce restrictions.

### 1.6 Roles and permissions

* **Purchasing Lead / Product Administrator:** Create/update products and kits but cannot approve.
* **Operations Manager:** Approve products and manage hazardous classifications.
* **E‑Commerce Manager:** Approve products for channel publication and manage digital assets.
* **Warehouse Associate:** Read-only access for reference on packaging and handling.
* **Auditor:** Read-only access to product history and audit logs.

### 1.7 Error handling

* **Validation errors** (missing mandatory fields, invalid unit conversions) return 400 with detailed error messages.
* **Duplicate SKU** returns 409 Conflict.
* **Channel export failure** logs error with correlation ID and sends a notification. Retries are attempted according to exponential backoff.

### 1.8 Performance and scalability

* The system must support **thousands of concurrent product updates** with minimal latency. Publishing to channels should handle batch sizes of up to 10 000 products.
* Use asynchronous processing for exports via background workers and message queue to avoid blocking UI transactions.
* Indexes on SKU, status and channel mapping fields ensure fast queries.

### 1.9 Non‑functional considerations

* **Security:** Enforce role-based access control. Sensitive fields (e.g. cost price) should be masked except for authorised roles.
* **Regulatory compliance:** Follow hazardous materials regulations; restrict editing of hazmat data to authorised personnel. Keep historical records for regulatory audits.
* **Audit:** Every change to product data must be logged with user, timestamp and before/after values.

## 2. Inventory & Warehouse Management

### 2.1 Overview

This module governs the storage, tracking and control of physical inventory across all warehouses, cross‑docks and 3PL facilities. It must maintain **real‑time stock levels** and ensure that physical counts match system records. The system supports bin/zone management, RF‑enabled picking and cycle counts to achieve ≥ 98 % inventory accuracy【121583754693540†L300-L333】. Cycle counting is a method of periodic inventory auditing that verifies physical counts against recorded balances without halting operations【121583754693540†L300-L333】. Proper cycle counting improves data integrity and uses scheduled counts for high‑value items【121583754693540†L334-L346】. Inventory management also monitors reorder points, manages lot/expiry and serial numbers, and integrates with purchasing and order fulfilment workflows.

### 2.2 Functional capabilities

1. **Multi‑warehouse & location management**
   * Define physical warehouses, distribution centers and cross‑docks. Each facility has zones, aisles and bins. Include location attributes (temperature control, hazardous storage).
   * Support **3PL integration** for remote warehouses; maintain separate location codes but unify inventory view across internal and 3PL sites.
   * Transfer stock between locations with appropriate paperwork and inventory adjustments.
2. **Inventory transactions**
   * Track all stock movements: receipts, put‑aways, picks, transfers, adjustments, cycle counts, returns and disposals.
   * Each transaction records TransactionType, Quantity, LotNumber (if applicable), SerialNumber (if applicable), FromLocation, ToLocation, CreatedBy and timestamps.
3. **Cycle counting and accuracy**
   * Configure **ABC cycle counting**: high‑value or fast‑moving items counted more frequently. Generate cycle count tasks automatically.
   * Provide RF scanning support for counting. After counts, variance is calculated and an adjustment transaction is created if needed.
   * Report **inventory accuracy** as the ratio of correct counts to total counts【121583754693540†L343-L383】.
4. **Lot, expiry and serial management**
   * Capture lot numbers and expiration dates at receiving. Require this information for regulated products as per primary requirements.
   * Assign serial numbers to high‑value tools; link serial numbers to warranty or service records.
   * Implement **first‑expiry‑first‑out** (FEFO) picking for items with expiry dates.
5. **Reorder point and safety stock**
   * Calculate reorder points based on historical demand, lead times and safety stock policies. When on‑hand falls below the reorder point, generate purchase recommendations.
   * Provide dashboards showing days‑of‑supply, safety stock levels and pending POs.
6. **Inventory valuation**
   * Maintain cost layers (FIFO, LIFO or weighted average). Costing method is configurable per product.
   * Update cost of goods sold (COGS) when inventory is issued to orders or scrapped.

### 2.3 Data entities

* **Warehouse** – defines physical facility; includes address, timezone, capacity and flags for 3PL.
* **Location** – child of warehouse; includes zone, aisle, bin code, type (picking, bulk, returns, hazmat), capacity.
* **InventoryItem** – snapshot of quantity on hand by Product, Location, Lot, Serial (optional) and status (available, allocated, quarantined). Contains reorder point and safety stock fields.
* **InventoryTransaction** – logs every inventory movement; references product, lot/serial, quantity, source/destination, user, date and reason.
* **CycleCountTask** and **CycleCountResult** – tasks for scheduled counts and recorded results.

### 2.4 Process flows

1. **Receiving and put‑away**
   1. Create or import **ASN** (advanced shipping notice) from vendor or 3PL.
   2. Warehouse associate scans incoming cartons, captures lot/serial information and verifies against expected quantities.
   3. System creates a Receipt transaction and updates InventoryItem records.
   4. Generate put‑away tasks to move stock from inbound staging to storage bins.
2. **Picking and shipping**
   1. When an order is allocated, a **pick list** is generated. Items are selected using FEFO or FIFO rules.
   2. Warehouse associate picks items using RF scanners; system validates picks by scanning barcodes.
   3. Upon completion, system records Pick transactions and updates InventoryItem balances; picks are associated with Shipment records in the orders module.
   4. If lot or serial numbers are required, they are captured during picking.
3. **Cycle counting**
   1. Scheduler generates daily/weekly cycle count tasks by item class.
   2. Associate counts items in designated bins and enters counts via RF device.
   3. System compares counted quantity to system quantity; variances create Adjustment transactions and update inventory accuracy metrics【121583754693540†L343-L383】.

### 2.5 API and integration

* **Endpoints:** /api/v1/inventory for querying available, allocated and on‑hand quantities; /api/v1/inventory/transactions to record movements; /api/v1/cycle-counts to manage tasks.
* **External integrations:** 3PL systems exchange stock data via asynchronous events (e.g. InventoryUpdated, CycleCountCompleted); purchasing module triggers reorders when reorder points are breached.

### 2.6 Roles and permissions

* **Warehouse Associate:** Create inventory transactions (receipts, picks, transfers) and cycle count results.
* **Operations Manager:** Manage location definitions, approve adjustments and review inventory accuracy reports.
* **Purchasing Lead:** View inventory levels to plan procurement.
* **Auditor:** Read-only access to inventory transactions.

### 2.7 Error handling

* **Over‑pick/under‑pick**: If pick quantity exceeds available stock, system raises error; under‑picks prompt re-allocation or backorder.
* **Mismatch during receiving**: If received quantity differs from ASN, system flags discrepancy and requires supervisor approval.
* **Negative inventory prevention**: Transactions cannot cause negative on‑hand; system rejects transaction and logs event.

### 2.8 Performance and scalability

* **Real‑time updates:** Inventory changes must propagate within < 1 minute to ensure accurate availability across all modules.
* **High volume transactions:** System should handle thousands of RF scans per hour. Use batch processing for cycle count tasks.
* **Indexing:** Index InventoryItem on ProductId, WarehouseId, LocationId and LotNumber for fast lookups.

### 2.9 Non‑functional considerations

* **Reliability:** Use event sourcing or journalling to reconstruct inventory state from transactions in case of failures.
* **Security:** Limit write access to inventory transactions. Serial numbers and lot details must be traceable for recalls.
* **Compliance:** Track hazardous storage locations; ensure proper segregation of hazmat goods and maintain necessary documentation.

## 3. Orders & Return Merchandise Authorization (RMA)

### 3.1 Overview

The Orders & RMA module manages the entire lifecycle of customer orders (B2B and B2C), from entry through fulfilment, invoicing and returns. It must centralize orders from multiple channels (web store, Amazon, POS, B2B portal) and ensure that fulfilment meets the organisation’s service‑level targets. The module also handles the reverse logistics process—returns management—to streamline authorisation, inspection, disposition and refund processing. Efficient returns management improves customer satisfaction and reduces costs【477846722703772†L147-L163】, while ensuring that returned items are reintegrated into inventory where possible【477846722703772†L238-L288】.

### 3.2 Functional capabilities

1. **Order capture and validation**
   * Accept orders via API or UI, capturing customer details, shipping addresses, items, quantities, and chosen shipping methods.
   * Validate inventory availability through the Inventory module; allocate stock immediately or place items on backorder if insufficient.
   * Calculate taxes and shipping costs using Tax & Shipping modules.
   * Perform **fraud screening** using integrated payment processor services.
2. **Order workflows**
   * **New → Allocated → Picked → Shipped → Invoiced → Closed**. Orders can also be **Cancelled** or **Returned**.
   * Support partial shipments and partial invoices.
   * Provide actions for Customer Service Representatives (CSR) to edit orders before shipment (e.g. change quantities, shipping address) within defined time limits.
   * Automatically generate **shipment records** with tracking numbers and integrate with the Shipping module.
3. **Payment processing**
   * Integrate with payment gateways to authorise and capture payments. Securely store tokens; do not retain full card data.
   * Handle pre‑authorisations at order placement and capture upon shipping. Support refunds and partial refunds.
4. **Returns and RMA management**
   * Implement an **RMA issuance workflow**: customer requests return, CSR validates reason and issues RMA number with expiration date.
   * Track return status: *Authorised*, *Received*, *Inspected*, *Completed*.
   * Record inspection results and determine disposition (restock, refurbish, scrap) as per requirements【91221083353783†L32-L33】.
   * Generate refund or store credit depending on disposition; update financial records and inventory.
   * Provide reverse logistics labels via Shipping module; track return shipments and update status accordingly.

### 3.3 Data entities

* **Order** – header with customer info, status, total amounts, payment status, timestamps.
* **OrderLine** – product, quantity ordered, quantity allocated, unit price, tax and discount details.
* **Payment** – stores payment authorisations, captures and refunds; includes gateway transaction IDs, status and amount.
* **Shipment** – records pick and pack details, carrier, service level, tracking numbers and dates.
* **RMA** – header with customer, order reference, reason, status, issue date and expiration date.
* **RmaLine** – returned items, quantity, condition, disposition and refund amount.

### 3.4 Process flows

1. **Order placement**
   1. Order is submitted via web store or API. System performs data validation and calculates totals.
   2. Inventory module reserves stock; if not available, sets backorder flag.
   3. Payment is pre‑authorised. Order status becomes *Allocated* and events OrderPlaced and OrderAllocated are published.
2. **Picking and shipping**
   1. Warehouse module creates pick lists; after items are picked, shipments are created and shipping labels generated.
   2. Order status moves to *Picked*; after shipping, tracking numbers are recorded and status changes to *Shipped*. OrderShipped event is published.
   3. Payment is captured; invoice is issued.
3. **Returns process**
   1. Customer requests a return through the portal or CSR. CSR issues an RMA number and instructs the customer to send items back.
   2. When returned goods arrive, they are inspected. Inspection results determine disposition: restock, refurbish or scrap【91221083353783†L32-L33】.
   3. Inventory and financial adjustments are made; refund or credit is processed.
   4. RmaCompleted event is published.

### 3.5 API and integration

* **Endpoints:** /api/v1/orders for order CRUD; /api/v1/orders/{id}/cancel; /api/v1/orders/{id}/ship; /api/v1/returns for RMA operations.
* **External integrations:** Payment gateways (e.g. Stripe, Authorize.Net), shipping carriers via the Shipping module, tax calculation via Tax module, financial posting via Accounting Sync.

### 3.6 Roles and permissions

* **Customer Service Representative (CSR):** Create orders, edit orders until shipped, issue RMAs, process returns.
* **Warehouse Associate:** Pick, pack and ship orders; cannot cancel or modify orders.
* **Operations Manager:** Override allocations and approve cancellations and refunds.
* **Accountant:** View financial impacts and issue refunds.

### 3.7 Error handling

* **Payment failure:** If payment pre‑authorisation fails, order is not created; return error and ask user to retry.
* **Allocation failure:** If inventory cannot be reserved, system places items on backorder and notifies Purchasing.
* **Invalid RMA:** If return is attempted without a valid RMA or after RMA expiration, system rejects return.

### 3.8 Performance and scalability

* **Concurrent orders:** Support at least 200 concurrent users and order placements, meeting the SLA defined in non‑functional requirements.
* **Order volume:** System must handle spikes (e.g. seasonal sales) of thousands of orders per hour; order processing tasks should run asynchronously where possible.

### 3.9 Non‑functional considerations

* **Security:** All payment interactions must comply with PCI DSS. Sensitive data (cards, personal info) should be tokenised.
* **Audit:** Record all status changes and user actions; include OrderHistory and RmaHistory tables.
* **Customer experience:** Provide order tracking and status notifications to customers via email/SMS.

## 4. Purchasing & Vendor Management

### 4.1 Overview

This module automates the procurement process—from requisition through purchase order issuance, vendor communication and receipt of goods. A robust purchasing system helps reduce manual data entry, track spending and maintain supplier relationships. According to procurement software guidance, purchase order systems automate requisition approval and PO creation, support users in submitting requests and comparing supplier quotes, and provide flexible approval routing based on purchase amount or category【302420343612716†L112-L190】【302420343612716†L215-L228】. This module also maintains vendor scorecards, handles minimum order quantities and lead times and integrates with inventory and accounts payable.

### 4.2 Functional capabilities

1. **Requisition and approval workflow**
   * Internal users create requisitions for products or services, specifying product, quantity, required date and justification.
   * The system routes requisitions for approval based on purchasing policies (amount thresholds, department budgets). Approvers receive tasks and can approve, reject or request changes.
   * Once approved, requisitions convert automatically into purchase orders.
2. **Request for quote (RFQ) and supplier negotiation**
   * Buyers can send RFQs to one or more suppliers for items lacking contracted pricing.
   * Suppliers respond with price, lead time and terms. Buyers compare responses and select the best option.
   * System logs responses and selected supplier; declines other quotes.
3. **Purchase order management**
   * Generate POs from requisitions or manually. POs include line items, unit costs, quantities, expected delivery dates and shipping terms.
   * Support drop‑ship orders directly to customers.
   * Provide PO status tracking (Sent, Acknowledged, In Transit, Partially Received, Closed).
   * Include approval workflows for POs exceeding thresholds or requiring additional sign‑off.
4. **Vendor management and scorecards**
   * Maintain **Vendor** records (contact info, payment terms, lead times, ratings).
   * Track vendor performance through on‑time delivery, quality, fill rate and cost metrics; compute scorecards.
   * Support vendor onboarding with required documentation (W‑9, hazmat handling agreements).
5. **Receiving and matching**
   * Integrate with Inventory module to match received quantities with PO lines. Support partial receipts and backorders.
   * Provide three‑way matching for accounts payable: PO, receiving and vendor invoice.
6. **Spend analysis and forecasting**
   * Analyse purchasing spend by category, supplier and time period. Provide dashboards and alerts for spend overages.
   * Support forecasting of demand and reorder points to plan procurement activities.

### 4.3 Data entities

* **Requisition** – header containing requester, department, reason, status. Lines contain requested product/service, quantity, required date and notes.
* **PurchaseOrder** – header with supplier, shipping terms, payment terms, status, total amounts; lines with product, unit cost, quantity, delivery date and cost centre.
* **RFQ** – request for quotes sent to suppliers; includes items, required quantities and due date.
* **RFQResponse** – responses from suppliers; includes quoted price, lead time, currency and validity date.
* **Vendor** – record of supplier with address, contact, payment terms, MOQs and lead times; includes performance metrics fields.

### 4.4 Process flows

1. **Requisition to PO**
   1. User submits requisition; system validates budget and ensures requested items exist in product master.
   2. Approval workflow routes to department manager and purchasing lead; after approval, a PO is generated.
   3. PO is sent to vendor via EDI or email; vendor acknowledges receipt.
2. **RFQ and selection**
   1. For new items or when price changes, buyer creates an RFQ; system sends to selected vendors.
   2. Vendors submit responses; buyer compares and selects supplier.
   3. Selected quote is converted to PO; non‑selected quotes are archived.
3. **Receiving and invoicing**
   1. Upon shipment arrival, warehouse records receipt. Received quantity and quality are matched to PO lines.
   2. Accounts payable matches vendor invoice to PO and receipt; any discrepancy triggers exception workflow.

### 4.5 API and integration

* **Endpoints:** /api/v1/purchase-orders, /api/v1/requisitions, /api/v1/vendors, /api/v1/rfqs.
* **External integrations:** EDI with large vendors for PO transmission and acknowledgement; email or vendor portals for small suppliers; accounts payable systems for invoice matching; 3PL for drop‑ship orders.

### 4.6 Roles and permissions

* **Requester:** Create requisitions; cannot submit PO.
* **Department Manager:** Approve requisitions and RFQs.
* **Purchasing Lead:** Manage vendors, issue POs, send RFQs and evaluate responses.
* **Operations Manager:** Approve POs above threshold; override supplier selection.
* **Accountant:** Perform three‑way match and approve vendor invoices.

### 4.7 Error handling

* **Budget exceed:** If requested amount exceeds department budget, system rejects requisition and notifies requester.
* **Over‑received goods:** If received quantity exceeds PO quantity, system flags a discrepancy and requires approval.
* **Vendor non‑response:** If RFQs receive no response by due date, system escalates to Purchasing Lead.

### 4.8 Performance and scalability

* Must handle hundreds of active POs and requisitions simultaneously. Use async workflows for RFQ and PO communications.
* Provide near‑real‑time updates of vendor performance metrics to support decision making.

### 4.9 Non‑functional considerations

* **Compliance:** Enforce segregation of duties (requesters cannot approve their own requisitions). Retain audit logs of approvals and changes.
* **Vendor agreements:** For hazardous materials, require valid hazmat agreements and automatically verify before issuing POs【79366541072420†L13-L27】.
* **Security:** Limit vendor payment data access to authorised roles; encrypt sensitive information.

## 5. Shipping & Hazmat

### 5.1 Overview

This module manages the integration with shipping carriers and ensures that outbound and return shipments comply with hazardous materials (hazmat) regulations. It provides functionality for rate shopping, label generation, tracking, address validation and compliance documentation. According to the shipping integration overview by Cleveroad, a shipping API acts as a bridge between carriers, online stores and other systems, automating tasks such as **rate calculation, label generation and tracking updates**【520123862111848†L340-L344】. Shipping APIs allow businesses to compare shipping rates across carriers, reduce costs, provide real‑time tracking, validate addresses and even generate airway bills and labels【520123862111848†L373-L404】. They also automate order creation, show estimated delivery dates and send tracking updates to customers【520123862111848†L420-L447】. For hazardous materials, carriers and shippers must comply with federal regulations (DOT and IATA) and use carrier‑approved shipping solutions【79366541072420†L13-L27】.

### 5.2 Functional capabilities

1. **Carrier rate shopping**
   * Integrate with multiple carriers (UPS, FedEx, USPS, DHL, local carriers). Use a multi‑carrier API to request real‑time rates based on package dimensions, weight, origin, destination and service levels.
   * Compare rates and transit times; apply business rules (preferred carrier, lowest cost, fastest service) to select the best option for each shipment.
   * Provide shipping cost estimates during checkout.
2. **Label generation and documentation**
   * Automatically generate shipping labels (PDF, ZPL) containing sender, recipient, weight, dimensions and tracking information. This aligns with shipping APIs that automatically create orders and AWB IDs and generate labels【520123862111848†L420-L432】.
   * Support printing of **hazardous materials documentation** (shippers’ declaration, material safety data sheets) and compliance labels for regulated products. Only authorised personnel can print hazmat documents.
   * Store label and documentation files for reference in the Shipment entity.
3. **Real‑time tracking and notifications**
   * Subscribe to carrier webhooks or poll for status updates; update shipment status in real time and publish ShipmentUpdated events.
   * Provide tracking links for customers and send email/SMS notifications when shipments are created, in transit, or delivered.
   * Include support for non‑delivery reporting (NDR) and manage re‑delivery or return to sender【520123862111848†L463-L475】.
4. **Address validation and EDD (Estimated Delivery Date)**
   * Validate destination addresses to avoid misdeliveries and reduce customer complaints【520123862111848†L404-L411】.
   * Calculate and display estimated delivery dates based on historical carrier performance and destination data【520123862111848†L434-L447】.
5. **Hazardous materials compliance**
   * Flag shipments containing hazmat products. Before generating labels, verify that carriers support the hazardous classification.
   * Generate proper hazmat documentation (e.g. UN number, hazard class, packaging group) and include required markings and labels on packages.
   * Use carrier‑approved shipping software, as required by UPS and other carriers, and ensure shipments comply with **federal Hazardous Materials Regulations (HMR)** and **IATA Dangerous Goods Regulations**【79366541072420†L13-L27】【79366541072420†L73-L86】.
   * Maintain signed hazmat agreements for each carrier and verify expiration dates.
6. **Returns and RMA shipping**
   * Generate prepaid return labels; track return shipments and update return status in the Orders module.
   * Manage pickup scheduling for 3PL returns when necessary.

### 5.3 Data entities

* **Carrier** – defines shipping providers; includes services, API credentials, hazmat capabilities and transit time profiles.
* **Shipment** – links to Order, contains carrier, service level, package details (weight, dimensions), hazmat flag, tracking number, label file and status.
* **ShipmentRate** – temporary entity capturing rate quotes from carriers.
* **HazmatDocument** – file record storing hazardous goods declarations and supporting data (UN number, hazard class, packaging group).

### 5.4 Process flows

1. **Rate shopping and label purchase**
   1. When order is ready for shipping, system requests rate quotes from carriers via multi‑carrier API.
   2. System applies business rules to select carrier and service; user may override.
   3. System purchases label from chosen carrier; receives tracking number and label file.
   4. Shipment record is created with tracking information; label is sent to printer.
   5. ShipmentCreated event is published.
2. **Hazmat shipment**
   1. If any line on the order is flagged as hazardous, system verifies hazmat agreement and obtains required packaging instructions and documentation.
   2. Hazmat documents are generated and printed along with the shipping label.
   3. Shipments are booked only with carriers that handle the specific hazard class【79366541072420†L51-L56】【79366541072420†L73-L86】.
3. **Tracking and notifications**
   1. Carrier sends status updates; system updates shipment status and notifies customers.
   2. If NDR occurs, system alerts CSR to take corrective action.

### 5.5 API and integration

* **Endpoints:** /api/v1/shipments for creating and querying shipments; /api/v1/shipments/{id}/rates to request rates; /api/v1/shipments/{id}/labels to purchase labels.
* **External integrations:** Multi‑carrier shipping API providers (e.g. ShipEngine, EasyPost), carriers’ proprietary APIs, hazmat regulation services, event bus for publishing ShipmentCreated, ShipmentUpdated, ShipmentDelivered events.

### 5.6 Roles and permissions

* **Warehouse Associate:** Request rates, print labels and schedule pickups.
* **Operations Manager:** Configure carrier accounts, set business rules for rate selection, approve hazmat shipments.
* **CSR:** View tracking status and respond to customer enquiries.
* **Auditor:** Access shipment history and hazmat documentation.

### 5.7 Error handling

* **Carrier API failure:** If rate or label purchase fails, system retries; after multiple failures, escalates to Operations Manager and allows manual processing.
* **Address validation error:** Return 422 Unprocessable Entity with validation errors; user must correct address.
* **Hazmat non‑compliance:** If hazmat agreement is missing or product class is unsupported, system blocks shipment and logs error.

### 5.8 Performance and scalability

* **Concurrent shipments:** Support hundreds of shipments per hour; use asynchronous requests for rate shopping to avoid blocking user sessions.
* **Real‑time tracking:** Poll or subscribe to carrier updates with minimal latency to ensure timely notifications.

### 5.9 Non‑functional considerations

* **Compliance:** Maintain up‑to‑date hazmat training and documentation; implement controls to ensure packaging and labelling compliance.
* **Security:** API credentials for carriers must be encrypted. Access to hazmat information is restricted.
* **Audit:** Log all label purchases, rate selections and hazmat document generations.

## 6. Tax & Reporting

### 6.1 Overview

This module calculates sales taxes for each order based on jurisdiction (county, state, country) and generates tax reporting data. It also provides financial and operational reporting across modules. External tax APIs, such as those provided by TaxJar, deliver reliable tax calculations; TaxJar reports an average response time below 20 ms and 99.99 % uptime【118742738972405†L9-L20】. Accurate tax calculation ensures compliance across North Carolina, Virginia, South Carolina and future jurisdictions【91221083353783†L36-L38】. The reporting component builds dashboards and analytical reports from transactional data, supporting decision‑making and audit needs.

### 6.2 Functional capabilities

1. **Sales tax calculation**
   * Determine tax rates based on ship‑to address, ship‑from address and product taxability. Use a third‑party tax API (e.g. TaxJar) to calculate county and state taxes with high accuracy【118742738972405†L9-L20】.
   * Support exemptions (resale certificates, tax‑exempt customers) and record exemption certificates.
   * Cache frequently used tax rates to minimise API calls and latency.
2. **Tax filing and reporting**
   * Generate monthly, quarterly and annual tax reports by jurisdiction, summarising taxable sales, non‑taxable sales, collected tax and returns.
   * Provide **audit trails** showing how taxes were calculated for each transaction (input addresses, tax codes, rates and API responses).
   * Export data in formats compatible with state filing systems.
3. **Financial and operational reporting**
   * Create dashboards and standard reports (sales by product, margin analysis, inventory turns, vendor performance, order fulfilment times). Use the data model and event stream to produce aggregated metrics.
   * Provide ad‑hoc reporting capabilities with filters and drill‑down; restrict data visibility by role.
   * Integrate with Business Intelligence tools via data warehouse or API.

### 6.3 Data entities

* **TaxTransaction** – captures tax details for each sale: transaction ID, jurisdiction codes, taxable amount, tax amount, exemption code, API response ID and timestamp.
* **TaxRateCache** – stores tax rates by zip code, state and county with effective dates.
* **ReportDefinition** – metadata for saved reports (name, description, SQL/dataset, schedule).
* **ReportExecution** – stores each report run with parameters, user, execution time and output location.

### 6.4 Process flows

1. **Tax calculation during order placement**
   1. Order module calls Tax API with ship‑to/ship‑from addresses and line items.
   2. Tax module receives response; records TaxTransaction and returns tax amount to Orders module.
   3. If API call fails, system uses cached rates and logs fallback.
2. **Filing preparation**
   1. Scheduler triggers report generation at end of tax period.
   2. System aggregates TaxTransaction data by jurisdiction and generates filings.
   3. Report is reviewed by accountant and submitted to tax authority.

### 6.5 API and integration

* **Endpoints:** /api/v1/tax/calculate to compute taxes; /api/v1/tax/reports to obtain tax reports; /api/v1/reports for general reports.
* **External integrations:** Tax calculation API (TaxJar or similar) for real‑time tax rates; Business Intelligence tools for advanced analytics.

### 6.6 Roles and permissions

* **Accountant:** Manage tax settings, run tax filings and access financial reports.
* **Operations Manager:** View operational dashboards but not raw tax details.
* **Auditor:** Access tax audit trails and generate compliance reports.

### 6.7 Error handling

* **API failure:** If tax API returns error, system uses cached rates and marks transaction for review.
* **Invalid address:** Return error requiring user to provide valid address; cannot compute tax without correct jurisdiction.
* **Reporting failure:** If report generation fails due to data error, system logs and notifies report owner.

### 6.8 Performance and scalability

* **Low latency:** Tax calculations should return within 100 ms on average; rely on caching and high‑performance API providers【118742738972405†L9-L20】.
* **High concurrency:** Support concurrent tax calculations for hundreds of orders during peak periods.
* **Reporting:** Off‑load heavy reports to asynchronous processes; provide status updates and allow retrieval when complete.

### 6.9 Non‑functional considerations

* **Compliance:** Keep transaction data and audit trails for regulatory retention periods. Follow Sarbanes‑Oxley (SOX) and relevant tax authority guidelines.
* **Security:** Restrict access to tax data; encrypt API keys and sensitive fields.
* **Audit:** Provide reproducible calculation details for each transaction; include event logs and API response data.

## 7. Accounting Sync

### 7.1 Overview

The Accounting Sync module integrates ERP operational data with external accounting systems (e.g. QuickBooks, NetSuite, general ledger). It ensures that all financial transactions resulting from purchases, sales, inventory movements and returns are correctly posted to the general ledger and sub‑ledgers. An ERP accounting system integrates financial management across HR, supply chain logistics, manufacturing, sales and CRM, providing real‑time data visibility【870928011918532†L247-L253】. Modern accounting software modules include general ledger, accounts payable, accounts receivable, payroll, expense tracking, bank reconciliation and financial reporting【870928011918532†L390-L417】. The Accounting Sync module aims to synchronise these functions with the ERP’s operational data.

### 7.2 Functional capabilities

1. **General ledger posting**
   * Map operational transactions to accounting journals. For example, when goods are received, debit inventory and credit accounts payable; when orders are shipped and invoiced, debit cost of goods sold and credit inventory.
   * Support different accounting methods (accrual vs. cash) and multi‑currency transactions.
   * Provide **GL mappings** configuration to assign ledger accounts by product category, location or transaction type.
2. **Accounts payable (AP) integration**
   * Synchronise vendor invoices from purchasing module to AP ledger. Include PO reference, receipt data, due date, discounts and approval status.
   * Support partial receipts and partial invoice matching.
   * Publish payment approvals and payment status updates back to ERP.
3. **Accounts receivable (AR) integration**
   * Synchronise customer invoices and credit memos. Include tax and shipping charges, payment terms and due dates.
   * Update customer balances and record payment receipts via the payment gateway.
4. **Bank reconciliation and cash management**
   * Import bank transactions; match them with recorded payments and receipts.
   * Create journal entries for bank fees and interest; reconcile cash accounts.
5. **Payroll and expense integration**
   * Optionally, import payroll journal entries from HR/payroll systems and allocate labour expenses to cost centres.
   * Import corporate card expenses and employee reimbursements into the GL and expense tracking.
6. **Data synchronization and reconciliation**
   * Provide two‑way synchronisation: push ERP transactions to accounting software and retrieve ledger balances for reconciliation. Handle data mapping differences (e.g. chart of accounts, customers vs. vendors definitions).
   * Implement batch and incremental sync; allow manual sync triggers.
   * Provide dashboards showing sync status (success, pending, failed) and reconciliation discrepancies.

### 7.3 Data entities

* **JournalEntry** – header with batch number, date, description and status; lines with account, debit/credit amounts, currency and reference ID.
* **LedgerAccountMapping** – links ERP transaction types to accounting ledger accounts.
* **SyncJob** – logs each sync run with source system, target system, start/end times, status and error details.
* **SyncError** – records individual failed records with reason and resolution flag.

### 7.4 Process flows

1. **Transaction posting**
   1. When an operational transaction occurs (e.g. PO receipt, order shipment), the ERP triggers creation of corresponding journal entries via mapping rules.
   2. Journal entries are stored in ERP and queued for synchronisation.
   3. Sync job processes queued entries, transforms them to the accounting system’s schema and calls the accounting API. If successful, marks the journal as posted.
   4. Accounting system returns a response with posted entry ID or error; errors are logged and require review.
2. **Reconciliation**
   1. At regular intervals, the sync job requests account balances from the accounting system.
   2. System compares ERP sub‑ledger totals with accounting system balances; any discrepancies generate reconciliation tasks.
   3. Users investigate and correct mapping or transaction errors.

### 7.5 API and integration

* **Endpoints:** /api/v1/accounting/sync to trigger manual sync; /api/v1/accounting/journals to query journal entries and status.
* **External integrations:** QuickBooks Online, NetSuite or other GL systems via their APIs (OAuth2 authentication). Support for message queues or middleware to decouple ERP from accounting system.

### 7.6 Roles and permissions

* **Accountant:** Configure ledger mappings, review and approve journal entries, initiate sync, resolve sync errors.
* **Operations Manager:** View sync dashboards and reconciliation reports; cannot modify entries.

### 7.7 Error handling

* **Mapping error:** If no mapping is defined for a transaction type, system logs error and notifies accountant; transaction remains unposted until mapping is added.
* **Sync failure:** If external API is unavailable or returns error, system retries with backoff; after repeated failures, marks sync as failed.
* **Reconciliation discrepancy:** System flags mismatch; user must resolve before closing period.

### 7.8 Performance and scalability

* Sync jobs must process all transactions within 30 minutes of the end of day. Use incremental sync to avoid processing the entire dataset daily.
* Support thousands of journal lines per day; batching and pagination are required for API calls.

### 7.9 Non‑functional considerations

* **Data integrity:** Ensure that no transactions are lost or duplicated. Use idempotent operations and unique reference IDs.
* **Security:** Secure connection to accounting systems via TLS; store API credentials securely. Restrict access to sync configuration.
* **Compliance:** Follow GAAP/IFRS standards and audit requirements; provide comprehensive audit trail of posted entries.

## 8. Cross‑cutting considerations

1. **Event‑driven architecture:** Each module publishes and consumes domain events via the event bus (Service Bus). Events include ProductCreated, InventoryUpdated, OrderPlaced, ShipmentCreated, TaxCalculated, JournalPosted and RmaCompleted. Messages follow the standardized schemas defined in the Event Schema document. Modules subscribe only to events relevant to their bounded context.
2. **Authentication and authorization:** All APIs require JWT tokens issued by the identity provider. Role‑based access controls are enforced at both the service and data layer (see RBAC matrix). Sensitive operations (e.g. refund, hazmat shipping, journal posting) require additional multi‑factor approval.
3. **Error logging and monitoring:** Centralized logging captures errors and performance metrics. Alerts trigger when error rates exceed defined thresholds or service level objectives are breached.
4. **Testing and validation:** Implement unit tests, integration tests and end‑to‑end tests for each module. Use synthetic transactions in staging to validate workflows. Performance testing should confirm that SLAs (e.g. order processing ≤ 2 seconds) are met.
5. **Documentation:** Keep API specifications, event schemas, data models and ADRs up to date. Provide user guides for different roles. Document all regulatory compliance procedures (hazmat handling, tax filing) for audit readiness.