**Kyaari OMS (Order Management System) - Developer Documentation**

**1. Overview**

Kyaari is a plant and flower pot selling company that operates similar to quick commerce (like Blinkit). This OMS (Order Management System) is designed to manage orders, route them through internal workflows, and ensure vendors, accounts, and operations work seamlessly.

The OMS will centralize:

* **Order ingestion** (via Admin API + Postman collection provided to Client for data entry or manual Excel upload).
* **Vendor assignment & confirmation** (full/partial availability).
* **Invoice management** (Accounts + Vendor invoices in JSON format).
* **Dispatch tracking**.
* **Store operator verification**.
* **Ticketing for discrepancies**.
* **Payment processing**.
* **Admin oversight**.

**2. User Roles**

* **Admin:** Manages all workflows, assignments, and has full oversight.
* **Vendor:** Confirms order quantities, dispatches goods, uploads invoices.
* **Accounts Team:** Creates POs, receives invoices, manages payments.
* **Store Operator (Blinkit):** Verifies received goods, raises tickets if mismatched.
* **Ops Team:** Handles tickets/discrepancies and ensures resolution.

**3. Workflow (High-Level)**

1. **Order Intake**
   * Orders are created via API that we provide. Blinkit team will use Postman collection to submit orders.
   * Admin can also upload via Excel.
2. **Assignment**
   * Admin assigns orders (whole or partial) to vendors.
3. **Vendor Confirmation**
   * Vendor accepts, declines, or confirms partial fulfillment.
   * If partial, backorders are created for the remaining qty.
4. **Accounts PO/Invoice**
   * Accounts team generates PO/invoice in JSON and shares with vendors.
5. **Dispatch**
   * Vendor dispatches items with AWB + dispatch proof.
6. **Store Verification**
   * Operator verifies goods against invoice.
   * If mismatch → raise ticket.
7. **Ticket Resolution**
   * Ops team resolves discrepancies.
8. **Payment Release**
   * Accounts processes payments once orders are verified.

**4. Status Lifecycle**

* RECEIVED → ASSIGNED → VENDOR\_CONFIRMED\_FULL / PARTIAL / DECLINED → INVOICED → DISPATCHED → STORE\_RECEIVED → VERIFIED\_OK → PAYMENT\_PENDING → PAYMENT\_DONE
* In case of mismatch: STORE\_RECEIVED → VERIFIED\_MISMATCH → TICKET\_RAISED → RESOLVED → PAYMENT\_DONE

**5. Key Features**

* API-first design with Postman collection for client (Kyaari) to submit orders.
* Multi-vendor assignment (per item split).
* Vendor confirmation of partial availability.
* Invoice management in JSON (integration-ready).
* Discrepancy handling via ticketing system.
* Role-based dashboards.
* Comprehensive reporting & KPIs.
* Audit logs for compliance.

**6. Tech Stack**

**Backend:**

* **NestJS (TypeScript)** — modular, scalable, well-suited for APIs.
* **Flask (Python)** — optional fallback for lightweight services.

**Database:**

* PostgreSQL (transactional data).
* Redis (caching, session management).
* S3-compatible storage (invoices, attachments).

**Frontend:**

* React.js + Tailwind (for all dashboards).
* Role-based access (Admin, Vendor, Accounts, Operator).

**Messaging & Integration:**

* RabbitMQ / AWS SQS (async tasks, notifications).
* Postman collection for Blinkit (instead of webhook).

**Auth & Security:**

* JWT-based authentication.
* RBAC (Role-Based Access Control).
* Audit logging.

**Infra:**

* AWS EC2/ECS for hosting.
* CI/CD pipeline.
* Monitoring: Prometheus + Grafana.
* Error tracking: Sentry.

**7. Suggested Timeline (MVP First)**

**Phase 1 (2-3 weeks)**

* Order ingestion (Excel + custom API with Postman collection)
* Admin dashboard (basic CRUD, assignment)
* Vendor dashboard (confirmation workflow)

**Phase 2 (2-3 weeks)**

* Accounts dashboard (PO/Invoice JSON workflow)
* Vendor dispatch module
* Store operator portal (verification, ticket raising)

**Phase 3 (2-3 weeks)**

* Ticket management system
* Payment processing module
* Reporting & analytics (fill rate, payment aging, disputes)

**Phase 4 (1-2 weeks)**

* Advanced features (multi-vendor split, backorders)
* SLA automation (timeouts, alerts)
* Security & audit enhancements

**8. Important Points to Consider**

* **Client API use**: Kyaari will not push data; we provide API + Postman collection.
* **Partial fulfillment**: Ensure system supports partial orders & automatic backorders.
* **Invoice compliance**: JSON format must be integration-ready for client.
* **Discrepancy resolution**: Strong ticketing system needed.
* **Payments**: Link payment release strictly to store verification.
* **Scalability**: System should handle high-volume orders nationwide.
* **Idempotency**: Prevent duplicate orders during API ingestion.
* **Monitoring & Alerts**: SLA breaches (e.g., vendor not confirming in time) should auto-escalate.
* **Audit Trail**: Every action logged with user + timestamp.

**9. System Modules (Deep Dive)**

**Module 1: Sign-In & Main Screen**

* **Authentication:** JWT-based login with roles (Admin, Vendor, Accounts, Ops, Store Operator).
* **Login Types:**
  + Email + Password (all roles).
  + OTP (for store operators).
  + SSO (optional for enterprise clients).
* **Main Screen Post-Login:** Role-based dashboard redirect (e.g., Admin → Admin Dashboard, Vendor → Vendor Dashboard).
* **Forgot Password/Reset:** Secure token-based password reset.
* **Session Management:** Auto logout after inactivity.

**Module 2: Admin Dashboard**

* **Order Management:**
  + View all orders (status, vendor assignment, invoices, payments).
  + Import orders via Excel.
  + Assign vendors to orders (per item split allowed).
* **Vendor Management:**
  + Register vendors (with KYC, PAN, GST info).
  + Track vendor performance (fill-rate, SLA compliance).
* **Dispute Overview:**
  + Monitor tickets raised by store operators.
  + Escalate or reassign orders.
* **Reporting:**
  + Payment pending summary.
  + Order fulfillment KPIs.
* **Controls:**
  + Full CRUD for orders, vendors, users.
  + Access to audit logs.

**Module 3: Vendor Dashboard**

* **Assigned Orders List:**
  + Accept/decline/confirm orders.
  + Confirm available qty per item.
* **Dispatch Workflow:**
  + Mark dispatch.
  + Upload dispatch proof (PDF/image).
* **Invoice Management:**
  + Receive PO from accounts.
  + Upload invoice (PDF/JSON).
* **Performance:**
  + Track pending orders, SLA breaches.

**Module 4: Accounts Dashboard**

* **PO Creation:**
  + Generate purchase orders in JSON format (client-compliant).
  + Link PO to assigned vendor.
* **Invoice Processing:**
  + Receive vendor invoices.
  + Validate against PO (auto validation rules).
* **Payment Management:**
  + Queue invoices for payment.
  + Track payment status (pending, approved, released).
* **Reports:**
  + Vendor-wise payment aging.
  + Reconciliation reports.

**Module 5: Ops Dashboard**

* **Store Receipts:**
  + Verify received items vs invoice.
  + Mark as Verified\_OK or Mismatch.
* **Ticket Management:**
  + Raise tickets for mismatches.
  + Attach photos and comments.
  + Update status (Open, Under Review, Resolved).
* **Escalation:**
  + Escalate unresolved tickets to Admin/Accounts.
* **Reports:**
  + Ticket resolution times.
  + Dispute statistics.

**10. Deliverables for Developers**

* REST API (with Swagger/OpenAPI documentation + Postman collection).
* DB schema with ER diagrams.
* Role-based UI mockups.
* Excel import template.
* Test cases for all workflows.
* Deployment scripts + CI/CD pipeline setup.

**This document serves as the reference guide for the development team to align on architecture, workflows, and implementation priorities.**