Oh-Dish

Full System FRD

Contents

DOCUMENT APPROVAL

Name	Role	Signature	Date
Product Owner	**Business Approval**	___	___
Business Analyst	**Requirements Approval**	___	___
QA Lead	**Validation Approval**	___	___

1. Document Purpose

This document defines:

- Complete UI structure
- Module breakdown
- Cross-module interaction mapping
- Database entities
- API integration model
- RBAC architecture
- Transaction flows
- Order lifecycle

This FRD serves as the foundation for:

- Sprint planning
- Backlog creation
- QA validation
- Automation design

2. System Architecture Overview

2.1 Logical Architecture Layers

Layer 1 – Presentation Layer
Layer 2 – Application Layer
Layer 3 – RBAC Authorization Engine
Layer 4 – Database Layer
Layer 5 – External Services (Email, Media, Payment Gateway)

---

3. MODULE 1 – AUTHENTICATION

APIs

POST /api/auth/login
POST /api/auth/logout
POST /api/auth/forgot-password

Database Entities

- users
- roles
- role_permissions

---

4. MODULE 2 – USER MANAGEMENT

Cross-Module Impact

- Role assignment affects POS & Orders access
- Status affects login ability

---

5. MODULE 3 – ROLE MANAGEMENT (RBAC Core)

Permission Types:

- View
- Create
- Update
- Delete

Functional Areas:

- Dashboard
- Orders
- POS
- Inventory
- Reports
- Users
- Roles
- Payment Gateway

RBAC enforced at:

- UI level
- API level

---

6. MODULE 4 – POS MODULE

6.1 Page Mockup: POS Screen

UI Components

- Product grid
- Category filter
- Cart panel
- Subtotal
- Tax
- Discount
- Total
- Checkout button

6.2 Backend APIs

GET /api/products
POST /api/pos/orders
POST /api/payments/process

6.3 Database Entities

Table: pos_orders

- id
- order_number
- user_id
- total_amount
- tax_amount
- discount_amount
- status
- created_at

Table: pos_order_items

- order_id
- product_id
- quantity
- unit_price
- total_price

Table: payments

- order_id
- payment_method
- amount
- transaction_reference
- status

6.4 POS Business Rules

- Stock must be validated before checkout
- Negative stock not allowed
- Payment must succeed before order status = Paid
- Transaction rollback if payment fails

6.5 POS Integration Flow

Product Selection → Cart → Tax Calculation → Discount → Payment → Inventory Deduction → Receipt Generation → Order Logged

7. MODULE 5 – ORDERS MODULE

7.1 Page Mockup: Orders List

Grid Fields:

- Order Number
- Source (POS / Online)
- Status
- Total
- Created By
- Date
- Actions

7.2 Backend APIs

GET /api/orders?page=&limit=
GET /api/orders/{id}
PUT /api/orders/{id}/status
POST /api/orders/{id}/refund

7.3 Database Entities

Table: orders

- id
- order_number
- source
- customer_id
- total_amount
- status
- created_by
- created_at

Table: order_history

- order_id
- previous_status
- new_status
- changed_by
- timestamp

7.4 Order Status Lifecycle

Pending → Confirmed → Completed
Pending → Cancelled
Completed → Refunded
Status transitions validated by:

- Role permission
- Payment state
- Refund policy

---

8. Cross-Module Interaction

Trigger	Affected Module
Role Change	POS & Orders access
Payment Success	Orders module
Refund	Inventory update
User Status Change	System-wide access

---

9. RBAC Enforcement Flow

1. User logs in
2. Role ID retrieved
3. Permissions fetched
4. UI dynamically renders allowed modules
5. Middleware validates every API request
6. Unauthorized → HTTP 403

---

10. Error Handling

Scenario	HTTP Code
Invalid Login	401
Unauthorized Access	403

Scenario	HTTP Code
Validation Error	422
Payment Failure	402
Server Error	500

---

11. Non-Functional Requirements

Security

- Password hashing
- JWT authentication
- Role-based middleware
- Rate limiting
- HTTPS enforced

Performance

- POS checkout < 3 seconds
- Order list load < 2 seconds
- Pagination required

Audit

- Log all transactions
- Log role updates
- Log order status changes
- Log refunds

---

12. Interaction Flow (System-Level)

Login → Dashboard → POS → Order Created → Payment Processed → Inventory Deducted → Order Stored → Reports Updated

Admin → Role Update → Permission Matrix Updated → All Users Affected Immediately