Internal Design Specification

Oh-Dish

Contents

User & Role Management Module

Version 1.0

Revision History

Date	Version	Description	Author
Feb 13, 2026	1.0	Initial Internal Design Specification	Sanyogita Herwathe

Contents

1. Introduction

The purpose of this document is to outline the architecture and technical design of the Oh-Dish Backoffice User & Role Management module.

This document serves as the technical working document for developers responsible for implementing authentication, user lifecycle management, and RBAC enforcement.

---

2. Requirements

The primary objective of the module is to:

- Enable secure authentication for merchant staff
- Enforce role-based access control (RBAC)
- Prevent unauthorized module access
- Maintain auditability of user and role operations
- Support scalable multi-merchant architecture

---

3. Architecture

3.1 High-Level View

The Oh-Dish User & Role Management module follows a **3-tier architecture**:

1. Presentation Layer
2. Business Logic Layer
3. Data Layer

Communication between browser and server uses HTTPS over secure REST APIs.

3.1.1 Presentation Layer

The Presentation Layer consists of:

- Login UI
- Dashboard UI
- Users management UI
- Roles management UI
- Permission Matrix UI

This layer:

- Renders UI components dynamically based on permissions
- Uses token-based authentication

- Handles client-side validations
- Communicates with backend APIs via REST endpoints

3.1.2 Business Logic Layer

The Business Logic Layer contains:

- Controllers (AuthenticationController, UserController, RoleController)
- Services (AuthService, RBACService, UserService)
- Middleware (AuthMiddleware, PermissionMiddleware)
- Exception Handling Components
- Routing configuration

Responsibilities:

- Validate input
- Hash and compare passwords
- Generate JWT tokens
- Fetch role permissions
- Enforce API-level authorization
- Log security events

3.1.3 Data Layer

The Data Layer is responsible for:

- Storing users, roles, permissions
- Maintaining referential integrity
- Enforcing unique constraints
- Logging audit events

Relational database (MySQL or PostgreSQL) is used.

3.2 Detailed View

Authentication Flow

User → Login Controller
Controller → Validate credentials
Controller → Compare hashed password
Controller → Check account status
Controller → Fetch Role
Controller → Fetch Permissions
Controller → Generate JWT
Controller → Return response

RBAC Enforcement Flow

1. User logs in
2. JWT issued
3. API request includes token
4. Middleware validates token
5. Permission middleware checks role_permissions
6. Allow or return 403

4. Frameworks

Component	Purpose
Web Framework (Laravel/Node)	Application structure
MySQL	Relational database
JWT Library	Token-based authentication
Bcrypt	Password hashing
REST API	Client-server communication

5. Architecture Structure in a Nutshell

5.1 Back-end

- Web framework
- REST controllers
- Service layer
- Middleware authorization
- Database integration
- Logging service

5.2 Front-end

- Dynamic UI rendering
- Permission-based module visibility
- Secure cookie storage
- Client-side validation

6. Database Structure

6.1 Data Definition Language (DDL)

Table: USERS

```
id INT PRIMARY KEY AUTO_INCREMENT,
username VARCHAR(50) UNIQUE NOT NULL,
email VARCHAR(100) UNIQUE NOT NULL,
password_hash VARCHAR(255) NOT NULL,
role_id INT NOT NULL,
status ENUM('Active','Suspended','Blocked','Deleted') DEFAULT 'Active',
failed_attempts INT DEFAULT 0,
last_login TIMESTAMP NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
FOREIGN KEY (role_id) REFERENCES roles(id)
```

Table: ROLES

```
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(50) UNIQUE NOT NULL,
description TEXT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

Table: PERMISSIONS

```
id INT PRIMARY KEY AUTO_INCREMENT,
module_name VARCHAR(50) NOT NULL,
action_type ENUM('View','Create','Update','Delete') NOT NULL
```

Table: ROLE_PERMISSIONS

```
role_id INT,
permission_id INT,
PRIMARY KEY (role_id, permission_id),
FOREIGN KEY (role_id) REFERENCES roles(id),
FOREIGN KEY (permission_id) REFERENCES permissions(id)
```

Table: AUDIT_LOGS
id INT PRIMARY KEY AUTO_INCREMENT,
user_id INT,
action VARCHAR(100),
timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
details TEXT

6.2 Entity Relationship Description

- One Role → Many Users
- One Role → Many Permissions
- One User → Many Audit Logs
- Many-to-Many between Roles and Permissions

---

7. User Interface

7.1 Login Window

Access Method

Accessible via:
https://ohdish.com/backoffice/login

Window Layout
Fields:

- Username (mandatory)
- Password (mandatory)
- Remember Me (optional)

Buttons:

- Sign In
- Forgot Password

7.2 Dashboard Page

Access Method

Accessible after successful authentication.

Behavior

- Dynamically renders modules based on permissions
- Hidden modules are not accessible via direct URL

7.3 Navigation Bar

Modules include:

- Dashboard
- Orders
- POS
- Inventory
- Reports
- Users
- Roles
- Settings
- Help

Modules visibility depends on role permissions.

---

8. Users Page

Access Method

Accessible only to roles with “View Users” permission.

Page Description

- Display paginated user grid
- Search capability
- Edit / Delete actions
- Add User button

---

9. Add/Edit User Page

Window Layout

- First Name
- Last Name
- Email
- Username
- Password
- Role Dropdown
- Status Dropdown

Behavior

- Save → Persist to database
- Save & Add Another → Clear form
- Cancel → Return to Users Page

---

10. Roles Page

Page Description

Displays:

- Role Name
- Access Count
- Edit/Delete options
- Add Role button

---

11. Permission Matrix Page

Window Layout

- Role Name
- Module list (Orders, POS, Users, Reports)
- Checkboxes for View/Create/Update/Delete

Behavior

- Save updates role_permissions table
- Changes apply immediately

Security Considerations

- Passwords hashed (bcrypt)
- JWT token expiration 15 minutes
- Account lock after 5 failed attempts
- HTTPS required
- API rate limiting enabled

Logging & Monitoring

System logs:

- Login attempts
- Failed logins
- User modifications
- Role changes
- Permission updates
- Unauthorized API attempts

Logs retained minimum 90 days.

POS Module

Version 1.0

Revision History

Date	Version	Description	Author
Feb 13, 2026	1.0	Initial POS IDD	Sanyogita Herwathe

Contents

1. Introduction

The POS (Point of Sale) module enables merchants to process in-store transactions including order placement, payment processing, receipt generation, and inventory deduction.

This document defines the internal technical design and architecture of the POS module.

2. Requirements

The POS module must:

- Allow staff to create orders
- Support cart management
- Apply taxes and discounts
- Process multiple payment types
- Deduct inventory
- Generate receipts
- Log transactions
- Enforce RBAC permissions

3. Architecture

3.1 High-Level View

POS follows 3-tier architecture:
Presentation Layer → Business Logic Layer → Data Layer
POS integrates with:

- Orders Module
- Inventory Module
- Payment Gateway
- Reporting Module

3.1.1 Presentation Layer

Includes:

- Product listing
- Category filters
- Cart panel

- Payment modal
- Receipt preview

Dynamic UI rendering based on:

- View POS permission
- Create Order permission

3.1.2 Business Logic Layer

Core services:

- Cart Service
- Pricing Service
- Tax Service
- Discount Service
- Payment Service
- Inventory Service

Responsibilities:

- Validate cart items
- Calculate subtotal
- Apply tax rules
- Apply promotional discounts
- Validate payment
- Deduct stock
- Persist transaction

3.1.3 Data Layer

Uses relational database for:

- Orders
- Order items
- Payments
- Inventory updates
- POS transaction logs

---

4. Architecture Structure in a Nutshell

Back-End

- REST Controllers
- Service Layer
- Transaction Management
- Inventory deduction atomicity
- Payment validation logic

Front-End

- Real-time cart updates
- Dynamic total calculation
- Error handling (insufficient stock)
- Loading states during payment

---

5. Database Structure

Table: POS_ORDERS

id INT PRIMARY KEY AUTO_INCREMENT,
order_number VARCHAR(50),
user_id INT,
total_amount DECIMAL(10,2),
tax_amount DECIMAL(10,2),
discount_amount DECIMAL(10,2),
status ENUM('Pending','Paid','Cancelled','Refunded'),
created_at TIMESTAMP

Table: POS_ORDER_ITEMS

id INT PRIMARY KEY AUTO_INCREMENT,
order_id INT,
product_id INT,
quantity INT,
unit_price DECIMAL(10,2),
total_price DECIMAL(10,2)

Table: PAYMENTS

id INT PRIMARY KEY AUTO_INCREMENT,
order_id INT,
payment_method ENUM('Cash','Card','Online'),

```
amount DECIMAL(10,2),
transaction_reference VARCHAR(100),
status ENUM('Success','Failed','Pending')
```

---

6. Business Logic Flow

POS Checkout Flow

1. User selects products
2. CartService validates availability
3. PricingService calculates totals
4. DiscountService applies promotions
5. TaxService calculates tax
6. PaymentService validates payment
7. InventoryService deducts stock
8. Order saved
9. Receipt generated

---

7. User Interface

POS Screen
Components:

- Product grid
- Category filter
- Cart summary
- Payment modal
- Receipt popup

Access Method:
Navigation → POS
Permission Required:

- View POS
- Create Order

---

8. Security & Logging

- Validate permission before checkout
- Prevent negative stock
- Log all transactions
- Log failed payment attempts
- Ensure atomic DB transactions (rollback on failure)

Oh-Dish Backoffice

Orders Module

Version 1.0

Revision History

Date	Version	Description	Author
Feb 13, 2026	1.0	Initial Orders IDD	Sanyogita Herwathe

---

1. Introduction

The Orders module manages all order records including POS, online, and manual orders.

It provides tracking, modification, cancellation, and reporting functionality.

---

2. Requirements

The Orders module must:

- Display order list
- Support filtering by date/status
- Allow viewing order details
- Allow order cancellation
- Support refund processing
- Integrate with POS
- Enforce role-based permissions

---

3. Architecture

Orders module interacts with:

- POS Module
- Inventory Module
- Reporting Module
- Payment Gateway

3.1 Presentation Layer

Includes:

- Orders grid
- Filters
- Order detail modal
- Status indicators

Permissions required:

- View Orders
- Update Orders
- Cancel Orders

3.2 Business Logic Layer

Core services:

- OrderQueryService
- OrderStatusService
- RefundService
- AuditService

Responsibilities:

- Retrieve orders with pagination
- Validate status transitions
- Process refunds
- Update inventory on cancellation
- Log status changes

---

4. Database Structure

Table: ORDERS

id INT PRIMARY KEY AUTO_INCREMENT,
order_number VARCHAR(50),
source ENUM('POS','Online','Manual'),
customer_id INT NULL,
total_amount DECIMAL(10,2),
status ENUM('Pending','Confirmed','Completed','Cancelled','Refunded'),
created_by INT,
created_at TIMESTAMP

Table: ORDER_HISTORY

id INT PRIMARY KEY AUTO_INCREMENT,
order_id INT,
previous_status VARCHAR(50),
new_status VARCHAR(50),
changed_by INT,
changed_at TIMESTAMP

---

5. Order Status Lifecycle

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

- Role permissions
- Payment state
- Refund policy rules

---

6. Business Logic Flow

Order Cancellation Flow

1. Validate permission
2. Check order status
3. Reverse inventory
4. Initiate refund
5. Update order status
6. Log action

---

7. User Interface

Orders Page
Components:

- Order list grid
- Date filter
- Status filter
- Search by order number
- View details modal

Access:
Navigation → Orders
Permissions:

- View Orders
- Update Orders
- Delete Orders (if allowed)

---

8. Security & Logging

- Restrict status change based on role
- Log all order updates
- Audit refund transactions
- Prevent unauthorized direct API calls