Software Requirements Specification

Salon Management System

Version 1.0

October 31, 2025

Technology Stack

Backend: Laravel 10.x

Frontend: React 18.x

Database: MySQL 8.0

Software Requirements Specification (SRS)

Salon Management System

Version: 1.0

Date: October 31, 2025

Author: Development Team

Status: Draft

Document Control

Version History

Version	Date	Author	Changes
1.0	October 31, 2025	Development Team	Initial SRS Document

Document Approval

Name	Role	Signature	Date
Project Manager			
Technical Lead			
Client Representative			

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a complete description of the Salon Management System. It details the functional and non-functional requirements, technical architecture, and implementation specifications for a comprehensive salon management solution built with Laravel backend and React frontend.

1.2 Scope

The Salon Management System is a web-based application designed to streamline salon operations including:

- Inventory management for retail and in-salon products
- Point of Sale (POS) system for product sales
- Appointment scheduling and management
- Employee management and commission tracking
- Customer relationship management
- Online booking portal for customers
- Multi-channel communication (SMS, WhatsApp)
- Comprehensive reporting and analytics
- Multi-branch support

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
API	Application Programming Interface
SRS	Software Requirements Specification
POS	Point of Sale
OTP	One-Time Password
RBAC	Role-Based Access Control
SMS	Short Message Service
REST	Representational State Transfer
JWT	JSON Web Token
CRUD	Create, Read, Update, Delete
UI	User Interface
UX	User Experience

1.4 References

- User Requirements & Functional Specification (URFS) v1.0
- Laravel Documentation (v10.x)
- React Documentation (v18.x)
- RESTful API Design Best Practices
- OWASP Security Guidelines

1.5 Overview

This document is organized into sections covering system architecture, functional requirements, technical specifications, database design, API endpoints, security measures, and deployment procedures.

2. Overall Description

2.1 Product Perspective

The Salon Management System is a standalone web application that integrates various salon operations into a unified platform. It consists of:

- **Backend API** (Laravel): RESTful API handling business logic, data persistence, and third-party integrations
- **Admin/Operator Dashboard** (React): Internal interface for salon staff
- **Customer Booking Portal** (React): Public-facing website for appointment booking

2.2 Product Functions

The system provides the following major functions:

16. **Inventory Management**
- Product catalog management
- Stock tracking (retail and in-salon usage)
- Supplier management
- Purchase orders and quotations
- Multi-location inventory
17. **Point of Sale**
- Customer billing
- Payment processing
- Receipt generation
- Tax and discount management
18. **Appointment System**
- Round-robin staff allocation
- Calendar-based scheduling
- Online booking with OTP verification
- Booking modifications and cancellations
19. **Employee Management**
- Staff profiles and schedules
- Commission calculations
- Performance tracking
20. **Customer Management**
- Customer profiles
- Service history
- Purchase history
- Communication preferences
21. **Reporting & Analytics**
- Sales reports
- Inventory reports
- Commission reports
- Custom report generation
22. **Communication**
- SMS notifications
- WhatsApp integration
- Automated birthday wishes
- Promotional campaigns

2.3 User Classes and Characteristics

User Class	Description	Technical Expertise	Access Level
Admin	System administrator with full access	High	Full system access
Operator	Day-to-day salon operations staff	Medium	Operational access
Customer	Salon customers using booking portal	Low	Limited public access
System	Automated processes and integrations	N/A	System-level access

2.4 Operating Environment

- **Client Environment:**
- Modern web browsers (Chrome, Firefox, Safari, Edge)
- Desktop and mobile devices
- Minimum screen resolution: 1024x768 (desktop), 375x667 (mobile)
- **Server Environment:**
- PHP 8.1 or higher
- Node.js 18.x or higher
- MySQL 8.0 or higher
- Web server (Apache/Nginx)
- SSL/TLS certificate for HTTPS

2.5 Design and Implementation Constraints

- Must use Laravel 10.x for backend development
- Must use React 18.x for frontend development
- Must support MySQL database
- Must be mobile-responsive
- Must comply with data protection regulations
- Payment gateway selection during development phase
- WhatsApp API provider selection during development phase
- No split payment support in initial version

2.6 Assumptions and Dependencies

Assumptions:

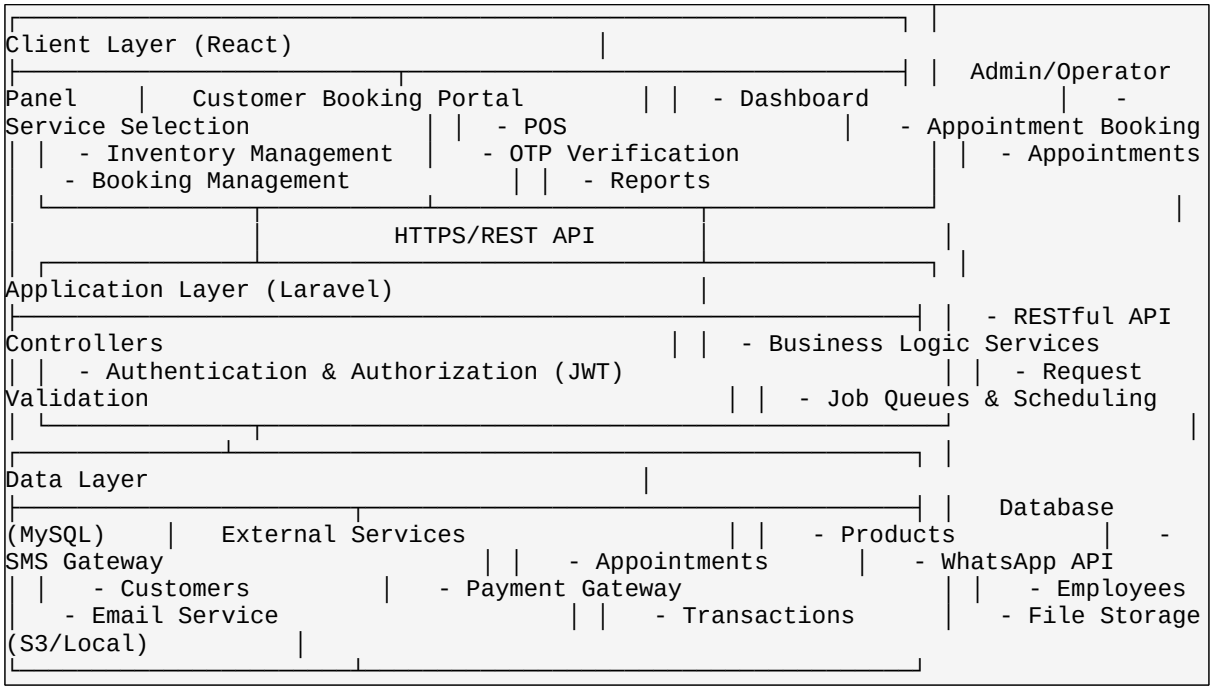
- Users have stable internet connectivity
- Payment gateway will be integrated during development
- SMS and WhatsApp API providers will be selected
- Hairdressers do not require system login accounts
- Single currency per branch

Dependencies:

- Third-party payment gateway service
- SMS gateway provider
- WhatsApp Business API provider
- Web hosting infrastructure
- SSL certificate authority

3. System Architecture

3.1 High-Level Architecture



3.2 Architecture Components

3.2.1 Frontend Architecture (React)

Component Structure:

```
src/
├── components/
│   ├── common/
│   │   ├── Button/
│   │   ├── Input/
│   │   ├── Modal/
│   │   ├── Table/
│   │   └── Alert/
│   ├── layout/
│   │   ├── Header/
│   │   ├── Sidebar/
│   │   ├── Footer/
│   │   └── Layout/
│   ├── pos/
│   │   ├── ProductSearch/
│   │   ├── Cart/
│   │   ├── Calculator/
│   │   └── Payment/
│   ├── appointments/
│   │   ├── Calendar/
│   │   ├── AppointmentForm/
│   │   └── AppointmentList/
│   ├── inventory/
│   │   ├── ProductList/
│   │   ├── StockManagement/
│   │   └── BarcodeGenerator/
│   └── reports/
│       ├── SalesReport/
│       ├── InventoryReport/
│       └── CommissionReport/
├── pages/
│   ├── Dashboard/
│   ├── POS/
│   ├── Inventory/
│   ├── Appointments/
│   ├── Customers/
│   ├── Employees/
│   ├── Reports/
│   └── Settings/
├── services/
│   ├── api/
│   ├── authService.js
│   ├── appointmentService.js
│   ├── inventoryService.js
│   └── customerService.js
└── utils/
    ├── hooks/
    ├── useAuth.js
    ├── useApi.js
    └── useDebounce.js
    ├── store/
    ├── slices/
    │   ├── authSlice.js
    │   ├── cartSlice.js
    │   └── appointmentSlice.js
    └── store.js
    ├── routes/
    ├── PrivateRoute.js
    └── routes.js
└── App.js
```

Key Technologies:

- React 18.x
- React Router v6
- Redux Toolkit for state management
- Axios for API communication
- Material-UI or Ant Design for UI components
- React Hook Form for form handling
- Chart.js/Recharts for data visualization
- FullCalendar for appointment scheduling
- React-Barcode for barcode generation
- Date-fns or Day.js for date handling

3.2.2 Backend Architecture (Laravel)

Directory Structure:

```
app/ ├── Console/ │   └── Commands/ │       ├── SendBirthdayWishes.php │       └── CheckLowStock.php ├── Http/ │   ├── Controllers/ │   │   ├── Api/ │   │   │   ├── AuthController.php │   │   │   ├── AppointmentController.php │   │   │   ├── ProductController.php │   │   │   ├── CustomerController.php │   │   │   ├── EmployeeController.php │   │   │   ├── POSController.php │   │   │   ├── InventoryController.php │   │   │   ├── ReportController.php │   │   │   └── BookingController.php │   ├── Middleware/ │   │   ├── Authenticate.php │   │   ├── CheckRole.php │   │   └── ValidateApiKey.php │   ├── Requests/ │   │   ├── AppointmentRequest.php │   │   ├── ProductRequest.php │   │   └── BookingRequest.php │   └── Resources/ │       ├── AppointmentResource.php │       ├── ProductResource.php │       └── CustomerResource.php ├── Models/ │   ├── User.php │   ├── Customer.php │   ├── Product.php │   ├── Appointment.php │   ├── Employee.php │   ├── Sale.php │   ├── Purchase.php │   ├── Inventory.php │   ├── Branch.php │   └── Service.php ├── Services/ │   ├── AppointmentService.php │   ├── InventoryService.php │   ├── SMSService.php │   ├── WhatsAppService.php │   ├── PaymentService.php │   └── ReportService.php ├── Repositories/ │   ├── AppointmentRepository.php │   ├── ProductRepository.php │   └── CustomerRepository.php ├── Jobs/ │   ├── SendSMS.php │   ├── SendWhatsAppMessage.php │   └── GenerateReport.php └── Events/     ├── AppointmentCreated.php     └── ServiceCompleted.php
```

Key Technologies:

- Laravel 10.x
- Laravel Sanctum for API authentication
- Laravel Queue for background jobs
- Laravel Scheduler for cron jobs
- Laravel Excel for report exports
- Intervention Image for image processing
- Laravel Notifications for multi-channel messaging
- Spatie Laravel Permission for RBAC

3.3 Communication Flow

3.3.1 Authentication Flow

```
Client → POST /api/login → Laravel Auth → JWT Token → Client Storage → Subsequent Requests with Bearer Token
```

3.3.2 Appointment Booking Flow

```
Customer → Select Service → Choose Time → OTP Request → Verify OTP → Create Appointment → SMS/WhatsApp Confirmation
```

3.3.3 POS Transaction Flow

```
Operator → Scan/Select Products → Apply Discounts → Process Payment → Update Inventory → Generate Receipt → Send to Customer
```

4. Functional Requirements

4.1 User Management

4.1.1 User Authentication

FR-AUTH-001: User Login

- **Description:** System shall allow users to authenticate using email and password
- **Priority:** High
- **Inputs:** Email, Password
- **Process:**

- Validate credentials against database
- Generate JWT token upon successful authentication
- Return user profile and permissions
- **Outputs:** JWT token, user profile, permissions
- **Error Conditions:** Invalid credentials, account locked, account inactive

FR-AUTH-002: Password Reset

- **Description:** Users can reset forgotten passwords via email
- **Priority:** Medium
- **Inputs:** Email address
- **Process:**
- Validate email exists
- Generate reset token
- Send reset link via email
- Allow password update with valid token
- **Outputs:** Password reset email, confirmation message

FR-AUTH-003: Session Management

- **Description:** System manages user sessions securely
- **Priority:** High
- **Process:**
- Token expiration after 24 hours
- Refresh token mechanism
- Logout functionality
- Concurrent session handling

4.1.2 Role-Based Access Control

FR-RBAC-001: Role Assignment

- **Description:** System supports two primary roles: Admin and Operator
- **Priority:** High
- **Roles:**
- **Admin:** Full system access, configuration, user management, all reports
- **Operator:** Appointments, POS, inventory operations, limited reports

FR-RBAC-002: Permission Management

- **Description:** Granular permissions for each module
- **Permissions Matrix:**

Permission	Admin	Operator
Manage Users	✓	✗
View Dashboard	✓	✓
POS Operations	✓	✓
Manage Inventory	✓	✓
Manage Appointments	✓	✓
View All Reports	✓	✗
View Own Reports	✓	✓
System Settings	✓	✗
Manage Branches	✓	✗
Manage Employees	✓	✗

4.2 Inventory Management

4.2.1 Product Management

FR-INV-001: Add Product

- **Description:** System allows adding new products to inventory
- **Priority:** High
- **Inputs:**
- Product name (required)
- SKU (auto-generated or manual)
- Category (required)
- Brand
- Description
- Unit price (required)
- Cost price
- Barcode
- Reorder level
- Product type: Retail/Consumable/Both
- Usage count (for consumables)
- Branch/Warehouse assignment
- **Validations:**
- Unique SKU
- Positive pricing
- Valid category
- **Outputs:** Product ID, success message

FR-INV-002: Product Categories

- **Description:** Hierarchical product categorization
- **Features:**
- Create/Edit/Delete categories
- Parent-child relationships
- Category-based filtering
- Category-wise reports

FR-INV-003: Barcode Generation

- **Description:** Generate and print barcodes for products
- **Priority:** High
- **Features:**
- Multiple barcode formats (Code128, EAN-13, UPC)
- Customizable barcode labels
- Bulk barcode printing
- Barcode scanning support

FR-INV-004: Product Import/Export

- **Description:** Bulk product operations via CSV/Excel
- **Features:**
- CSV/Excel template download
- Bulk import with validation
- Error reporting
- Export product catalog

4.2.2 Stock Management

FR-INV-005: Stock Tracking

- **Description:** Dual inventory tracking system
- **Priority:** High
- **Features:**

- **Retail Stock:** Products for sale
- **Consumable Stock:** Products used in services
- Real-time stock updates
- Stock movement history
- Low stock alerts

FR-INV-006: Stock Adjustment

- **Description:** Manual stock adjustments with reason tracking
- **Inputs:**
- Product
- Adjustment quantity (+/-)
- Reason (damaged, expired, theft, etc.)
- Notes
- **Outputs:** Updated stock level, adjustment log

FR-INV-007: Multi-Location Inventory

- **Description:** Manage inventory across multiple branches/warehouses
- **Features:**
- Branch-wise stock levels
- Warehouse management
- Inter-branch transfers
- Location-based reporting

FR-INV-008: Stock Transfer

- **Description:** Transfer inventory between locations
- **Priority:** Medium
- **Inputs:**
- Source location
- Destination location
- Products and quantities
- Transfer reason
- Transfer date
- **Process:**
- Deduct from source
- Add to destination
- Create transfer record
- Generate transfer document
- **Outputs:** Transfer ID, transfer document

FR-INV-009: Low Stock Alerts

- **Description:** Automated alerts when stock falls below reorder level
- **Features:**
- Configurable reorder levels per product
- Email/SMS notifications
- Dashboard alerts
- Reorder suggestions

4.2.3 Purchase Management

FR-INV-010: Create Purchase Order

- **Description:** Generate purchase orders for suppliers
- **Priority:** High
- **Inputs:**
- Supplier
- Products and quantities
- Expected delivery date
- Payment terms

- Notes
- **Outputs:** PO number, PO document

FR-INV-011: Receive Stock

- **Description:** Record received inventory against purchase orders
- **Process:**
- Select purchase order
- Enter received quantities
- Update stock levels
- Record partial/full receipt
- Calculate variances

FR-INV-012: Purchase Returns

- **Description:** Return defective or incorrect products to suppliers
- **Inputs:**
- Original purchase reference
- Return items and quantities
- Return reason
- Expected refund/credit
- **Process:**
- Deduct from inventory
- Create return document
- Update supplier account

FR-INV-013: Supplier Management

- **Description:** Maintain supplier database
- **Data Fields:**
- Supplier name
- Contact person
- Phone, email, address
- Payment terms
- Tax information
- Purchase history
- **Features:**
- Add/Edit/Delete suppliers
- Supplier performance tracking
- Supplier-wise purchase reports

4.2.4 Quotations

FR-INV-014: Create Quotation

- **Description:** Generate quotations for customers/vendors
- **Features:**
- Select products with quantities
- Apply discounts
- Add terms and conditions
- Generate PDF
- Send via email
- Convert to purchase order

4.3 Point of Sale (POS)

4.3.1 Sales Transaction

FR-POS-001: Product Selection

- **Description:** Multiple methods to add products to cart
- **Priority:** High

- **Methods:**
- Barcode scanning
- Product search (name, SKU)
- Category-based browsing
- Recent products
- **Features:**
- Quick search with autocomplete
- Product image display
- Stock availability check
- Price display

FR-POS-002: Cart Management

- **Description:** Manage items in shopping cart
- **Features:**
- Add products with quantity
- Update quantities
- Remove items
- Clear cart
- Apply item-level discounts
- View cart total in real-time

FR-POS-003: Customer Selection

- **Description:** Associate sale with customer
- **Features:**
- Search existing customers
- Quick add new customer
- Walk-in customer option
- Customer details display
- Customer purchase history

FR-POS-004: Billing Calculations

- **Description:** Automatic calculation of bill amounts
- **Priority:** High
- **Calculations:**
- Subtotal
- Item-wise discounts
- Overall discount (% or fixed)
- Tax calculation (configurable rate)
- Shipping charges
- Grand total
- **Features:**
- Built-in calculator
- Rounding options
- Multiple tax rates support

FR-POS-005: Hold and Resume Bills

- **Description:** Save incomplete transactions for later
- **Features:**
- Hold current bill
- Multiple held bills
- Resume held bill
- Delete held bill
- View held bills list

FR-POS-006: Payment Processing

- **Description:** Process customer payments
- **Priority:** High

- **Payment Methods:**
- Cash
- Card (Credit/Debit)
- Bank Transfer
- Payment Gateway
- **Features:**
- Payment method selection
- Change calculation
- Payment confirmation
- No split payments (constraint)

FR-POS-007: Receipt Generation

- **Description:** Generate and print sales receipts
- **Features:**
- Customizable receipt template
- Company logo and details
- Itemized listing
- Payment details
- Tax breakdown
- QR code (optional)
- Thermal printer support
- PDF export
- Email receipt option

4.3.2 Sales Returns

FR-POS-008: Process Return

- **Description:** Handle product returns and refunds
- **Inputs:**
- Original sale reference
- Return items and quantities
- Return reason
- Refund method
- **Process:**
- Validate original sale
- Check return eligibility
- Add back to inventory
- Process refund
- Generate credit note
- **Validations:**
- Return within allowed period
- Product condition check
- Quantity validation

4.4 Appointment Management

4.4.1 Appointment Scheduling

FR-APT-001: Create Appointment

- **Description:** Schedule customer appointments
- **Priority:** High
- **Inputs:**
- Customer details
- Service(s) required
- Preferred date and time
- Special notes

- **Process:**
- Check staff availability
- Apply round-robin allocation
- Confirm booking
- Send confirmation
- **Outputs:** Appointment ID, confirmation message

FR-APT-002: Round-Robin Staff Allocation

- **Description:** Automatically assign appointments to staff fairly
- **Priority:** High
- **Algorithm:**
- Track appointment count per staff member
- Allocate to staff with least appointments
- Consider staff availability and service type
- Handle staff breaks and time off
- **Features:**
- Manual override by operator
- Automatic reallocation if staff unavailable
- Load balancing across staff
- Service type expertise matching (optional)

FR-APT-003: Staff Availability Management

- **Description:** Operators manage staff schedules and availability
- **Features:**
- Set working hours per staff
- Mark time off/breaks
- Block time slots
- Set recurring schedules
- Emergency unavailability marking
- Automatic time slot updates on public booking site

FR-APT-004: Calendar View

- **Description:** Visual calendar for appointment management
- **Features:**
- Day/Week/Month views
- Color-coded by service type
- Staff-wise filtering
- Drag-and-drop rescheduling
- Appointment details on hover
- Print calendar
- Export appointments

FR-APT-005: Modify Appointment

- **Description:** Update existing appointments
- **Allowed Changes:**
- Date and time
- Service type
- Assigned staff (manual)
- Customer details
- Special notes
- **Process:**
- Check new slot availability
- Update appointment
- Notify customer
- Reallocate if needed

FR-APT-006: Cancel Appointment

- **Description:** Cancel scheduled appointments
- **Inputs:**
- Appointment ID
- Cancellation reason
- Cancelled by (customer/operator)
- **Process:**
- Mark as cancelled
- Free up time slot
- Notify customer
- Update staff schedule
- **Features:**
- Cancellation tracking
- No-show marking
- Cancellation reports

FR-APT-007: Appointment Reminders

- **Description:** Automated appointment reminders
- **Features:**
- SMS reminder 24 hours before
- WhatsApp reminder (optional)
- Configurable reminder timing
- Multiple reminder support
- Opt-out option

4.4.2 Service Completion and Billing

FR-APT-008: Mark Service Complete

- **Description:** Record service completion
- **Priority:** High
- **Inputs:**
- Appointment ID
- Actual services performed
- Products used in service
- Duration
- Staff notes
- **Process:**
- Mark appointment as completed
- Deduct consumable product usage counts
- Calculate service charges
- Generate bill
- **Outputs:** Completed status, bill details

FR-APT-009: Service Billing

- **Description:** Create invoice after service completion
- **Features:**
- Service charges
- Additional product sales
- Apply discounts
- Tax calculation
- Link to inventory consumption
- Payment processing
- Receipt generation

FR-APT-010: Product Usage Tracking

- **Description:** Track consumable products used in services
- **Priority:** High
- **Process:**

- Select products used
- Automatic usage count deduction
- Update consumable inventory
- Link to service record
- **Features:**
- Predefined service-product mappings
- Manual product addition
- Usage history per customer

4.5 Customer Management

4.5.1 Customer Database

FR-CUS-001: Add Customer

- **Description:** Register new customers
- **Priority:** High
- **Data Fields:**
- Name (required)
- Phone (required, unique)
- Email
- Date of birth
- Gender
- Address
- Preferred staff member
- Notes
- Registration date
- **Validations:**
- Valid phone format
- Valid email format
- Unique phone number

FR-CUS-002: Customer Profile

- **Description:** Comprehensive customer information
- **Includes:**
- Personal details
- Contact information
- Service history
- Purchase history
- Appointment history
- Total spending
- Last visit date
- Communication preferences
- Notes and tags

FR-CUS-003: Customer Search

- **Description:** Quick customer lookup
- **Search By:**
- Name
- Phone number
- Email
- Customer ID
- **Features:**
- Autocomplete
- Fuzzy matching
- Recent customers list

FR-CUS-004: Customer Segmentation

- **Description:** Categorize customers for targeted marketing
- **Segments:**
- VIP customers (high spending)
- Regular customers
- Inactive customers
- First-time visitors
- **Uses:**
- Targeted promotions
- Loyalty programs
- Customized communication

4.5.2 Customer History

FR-CUS-005: Service History

- **Description:** Complete record of services availed
- **Data:**
- Date of service
- Services performed
- Staff member
- Products used
- Amount paid
- Customer feedback

FR-CUS-006: Purchase History

- **Description:** Record of all product purchases
- **Data:**
- Purchase date
- Products purchased
- Quantities
- Amount paid
- Payment method

FR-CUS-007: Appointment History

- **Description:** All past and upcoming appointments
- **Data:**
- Scheduled appointments
- Completed appointments
- Cancelled appointments
- No-shows
- Rescheduling history

4.6 Employee Management

4.6.1 Staff Administration

FR-EMP-001: Add Employee

- **Description:** Register salon staff members
- **Priority:** High
- **Data Fields:**
- Full name (required)
- Employee ID (auto-generated)
- Role (Hairdresser, Receptionist, Manager)
- Phone (required)
- Email
- Date of birth
- Joining date

- Photo
- Address
- Emergency contact
- Skills/Specializations
- Branch assignment
- **Note:** Hairdressers do not get system login accounts

FR-EMP-002: Employee Profile

- **Description:** Detailed employee information
- **Includes:**
- Personal details
- Work schedule
- Performance metrics
- Commission details
- Attendance record
- Customer ratings (if applicable)

FR-EMP-003: Schedule Management

- **Description:** Define employee work schedules
- **Features:**
- Set working days and hours
- Define break times
- Set time off
- Recurring schedule templates
- Override for specific dates
- Schedule conflicts detection

4.6.2 Performance Tracking

FR-EMP-004: Service Records

- **Description:** Track services performed by each staff member
- **Metrics:**
- Number of services
- Service types
- Customer ratings
- Repeat customer rate
- Average service time

FR-EMP-005: Commission Calculation

- **Description:** Calculate staff commissions
- **Priority:** High
- **Calculation Methods:**
- Percentage of service revenue
- Fixed amount per service
- Tiered commission structure
- **Features:**
- Service-wise commission rates
- Monthly commission reports
- Commission payment tracking
- Performance-based bonuses

FR-EMP-006: Attendance Tracking

- **Description:** Record employee attendance (optional)
- **Features:**
- Clock in/out
- Late arrivals
- Early departures

- Absence recording
- Attendance reports

4.7 Online Booking System

4.7.1 Public Booking Portal

FR-BOOK-001: Service Selection

- **Description:** Customers browse and select services
- **Priority:** High
- **Features:**
- Service catalog with descriptions
- Service duration
- Pricing information
- Service categories
- Search functionality
- Multiple service selection

FR-BOOK-002: Time Slot Selection

- **Description:** Display available appointment slots
- **Priority:** High
- **Features:**
- Real-time availability based on staff schedules
- Date picker
- Time slot grid
- Duration-aware slot blocking
- Unavailable slots clearly marked
- Updates when operator modifies staff availability

FR-BOOK-003: OTP Verification

- **Description:** Verify customer phone number via OTP
- **Priority:** High
- **Process:**

23. Customer enters phone number
24. System generates 6-digit OTP
25. SMS sent to customer
26. Customer enters OTP
27. System validates OTP (5-minute expiry)
28. Booking confirmed

- **Features:**
- OTP resend option (max 3 attempts)
- OTP expiry handling
- Failed attempt tracking

FR-BOOK-004: Customer Recognition

- **Description:** Link booking to existing customer if phone is verified
- **Process:**
- Check if verified phone exists in database
- If exists, link to customer profile
- If new, create new customer record
- Pre-fill known details
- **Benefits:**
- Service history access
- Personalized experience
- Loyalty tracking

FR-BOOK-005: Booking Confirmation

- **Description:** Confirm and notify successful booking
- **Priority:** High
- **Features:**
- Confirmation page with details
- Confirmation number
- SMS confirmation
- WhatsApp confirmation (optional)
- Email confirmation (optional)
- Add to calendar option
- Booking details PDF

FR-BOOK-006: Modify Booking

- **Description:** Customers can reschedule appointments online
- **Features:**
- Access booking via confirmation number/phone
- View current booking details
- Select new date/time
- Re-verify if needed
- Update confirmation sent
- **Restrictions:**
- Cannot modify within X hours of appointment
- Limited modifications (e.g., max 2 changes)

FR-BOOK-007: Cancel Booking

- **Description:** Customers can cancel appointments online
- **Features:**
- Access booking via confirmation number/phone
- View booking details
- Confirm cancellation
- Cancellation confirmation sent
- Cancellation reason (optional)
- **Restrictions:**
- Cannot cancel within X hours of appointment
- Cancellation policy display

FR-BOOK-008: Booking History

- **Description:** Customers view their booking history
- **Features:**
- Past appointments
- Upcoming appointments
- Cancelled appointments
- Service details
- Booking receipts

4.7.2 Responsive Design

FR-BOOK-009: Mobile Optimization

- **Description:** Booking portal fully functional on mobile devices
- **Requirements:**
- Responsive layout
- Touch-friendly interface
- Fast loading times
- Progressive Web App features
- Cross-browser compatibility

4.8 Communication System

4.8.1 SMS Integration

FR-COM-001: SMS Gateway Integration

- **Description:** Integrate with SMS service provider
- **Priority:** High
- **Use Cases:**
- Booking confirmations
- Appointment reminders
- OTP delivery
- Promotional messages
- Birthday wishes
- Special offers
- **Features:**
- Template management
- Variable substitution
- Delivery status tracking
- Failed message retry
- SMS log

FR-COM-002: SMS Templates

- **Description:** Predefined message templates
- **Template Types:**
- Booking confirmation
- Appointment reminder
- OTP message
- Cancellation notice
- Birthday wish
- Promotional offer
- Service completion thank you
- **Features:**
- Create/Edit/Delete templates
- Variable placeholders (name, date, time, etc.)
- Template preview
- Multi-language support

4.8.2 WhatsApp Integration

FR-COM-003: WhatsApp Business API

- **Description:** Integrate WhatsApp Business API
- **Priority:** Medium
- **Use Cases:**
- Booking confirmations with details
- Appointment reminders
- Bill/receipt sharing
- Promotional campaigns
- Birthday wishes
- Service notifications
- **Features:**
- Rich media messages
- Template messages
- Interactive buttons
- Delivery receipts
- Message status tracking

FR-COM-004: WhatsApp Templates

- **Description:** WhatsApp approved message templates
- **Features:**
- Template submission and approval workflow
- Variable substitution
- Media attachments (images, PDFs)
- Button actions
- Template categories

4.8.3 Email Communication

FR-COM-005: Email Notifications

- **Description:** Send automated emails
- **Use Cases:**
- Booking confirmations
- Password reset
- Invoices/receipts
- Monthly statements
- Promotional newsletters
- **Features:**
- HTML email templates
- Attachment support
- Email queue for bulk sending
- Unsubscribe option
- Email tracking (opens, clicks)

4.8.4 Automated Communications

FR-COM-006: Birthday Wishes

- **Description:** Automatically send birthday greetings
- **Priority:** Medium
- **Process:**
- Daily job checks customer birthdays
- Send personalized message via SMS/WhatsApp
- Include special birthday offer (optional)
- Track sent wishes
- **Features:**
- Customizable birthday message
- Time scheduling (e.g., 9 AM)
- Special offer attachment

FR-COM-007: Promotional Campaigns

- **Description:** Send bulk promotional messages
- **Features:**
- Select target audience (all, segment, custom)
- Choose channel (SMS/WhatsApp/Email)
- Schedule send time
- Track delivery and responses
- Campaign analytics

4.9 Reporting and Analytics

4.9.1 Sales Reports

FR-REP-001: Daily Sales Report

- **Description:** Summary of daily sales
- **Metrics:**

- Total sales amount
- Number of transactions
- Payment method breakdown
- Tax collected
- Discounts given
- Top selling products
- **Filters:** Date, branch, operator

FR-REP-002: Sales by Product

- **Description:** Product-wise sales analysis
- **Metrics:**
- Quantity sold
- Revenue
- Profit margin
- Sales trend over time
- **Filters:** Date range, category, branch

FR-REP-003: Sales by Customer

- **Description:** Customer purchase analysis
- **Metrics:**
- Total purchases
- Frequency
- Average transaction value
- Last purchase date
- Top customers
- **Filters:** Date range, customer segment

4.9.2 Inventory Reports

FR-REP-004: Stock Level Report

- **Description:** Current inventory status
- **Details:**
- Product name
- Current retail stock
- Current consumable stock
- Reorder level
- Stock value
- Low stock items
- **Filters:** Branch, category, stock status

FR-REP-005: Stock Movement Report

- **Description:** Inventory transactions over time
- **Includes:**
- Stock in (purchases, transfers in)
- Stock out (sales, consumption, transfers out)
- Adjustments
- Opening and closing balance
- **Filters:** Date range, product, branch

FR-REP-006: Consumption Report

- **Description:** In-salon product usage
- **Priority:** High
- **Metrics:**
- Products consumed
- Usage quantities
- Services linked
- Cost of consumption

- Consumption rate
- **Filters:** Date range, product, service type

FR-REP-007: Purchase Report

- **Description:** Procurement analysis
- **Metrics:**
- Purchases by supplier
- Purchase amounts
- Purchase trends
- Payment status
- **Filters:** Date range, supplier, product

4.9.3 Service & Appointment Reports

FR-REP-008: Service Report

- **Description:** Analysis of services performed
- **Metrics:**
- Services by type
- Revenue by service
- Most popular services
- Service trends
- Average service value
- **Filters:** Date range, service type, staff

FR-REP-009: Appointment Analytics

- **Description:** Booking and appointment statistics
- **Metrics:**
- Bookings by source (online/walk-in)
- Booking conversion rate
- Cancellation rate
- No-show rate
- Peak booking times
- Average lead time
- **Filters:** Date range, source, status

4.9.4 Employee Reports

FR-REP-010: Commission Report

- **Description:** Staff commission calculation
- **Priority:** High
- **Details:**
- Services performed
- Service revenue
- Commission rate
- Commission amount
- Payment status
- **Filters:** Date range, employee, payment status

FR-REP-011: Staff Performance Report

- **Description:** Employee productivity analysis
- **Metrics:**
- Number of services
- Revenue generated
- Average service time
- Customer ratings
- Repeat customer rate
- Attendance

- **Filters:** Date range, employee

4.9.5 Financial Reports

FR-REP-012: Profit & Loss Statement

- **Description:** Financial performance overview
- **Components:**
- Total revenue (services + products)
- Cost of goods sold
- Gross profit
- Operating expenses
- Net profit
- **Filters:** Date range, branch

FR-REP-013: Expense Report

- **Description:** Business expense tracking
- **Categories:**
- Salaries
- Rent
- Utilities
- Supplies
- Marketing
- Other expenses
- **Filters:** Date range, category, branch

FR-REP-014: Tax Report

- **Description:** Tax collection and liability
- **Details:**
- Taxable sales
- Tax collected
- Tax payable
- Tax category breakdown
- **Filters:** Date range, tax type

4.9.6 Customer Reports

FR-REP-015: Customer Analysis

- **Description:** Customer behavior insights
- **Metrics:**
- New customers
- Returning customers
- Customer retention rate
- Customer lifetime value
- Churn rate
- **Filters:** Date range, segment

FR-REP-016: Customer Service History

- **Description:** Individual customer report
- **Priority:** High
- **Includes:**
- All services availed
- Products purchased
- Total spending
- Visit frequency
- Preferred services
- Preferred staff
- **Export:** PDF, Excel

4.9.7 Report Features

FR-REP-017: Report Export

- **Description:** Export reports in multiple formats
- **Formats:**
- Excel (.xlsx)
- PDF
- CSV
- **Features:**
- Formatted export
- Charts and graphs included
- Customizable layout

FR-REP-018: Report Scheduling

- **Description:** Automated report generation and delivery
- **Features:**
- Schedule frequency (daily, weekly, monthly)
- Email delivery
- Multiple recipients
- Custom report templates

FR-REP-019: Dashboard

- **Description:** Real-time business overview
- **Widgets:**
- Today's revenue
- Appointments today
- Pending appointments
- Low stock items
- Top products
- Top staff
- Recent transactions
- Quick actions
- **Features:**
- Customizable widgets
- Date range selector
- Branch filter
- Auto-refresh

4.10 System Configuration

4.10.1 Company Settings

FR-SET-001: Company Profile

- **Description:** Configure business information
- **Settings:**
- Company name
- Logo upload
- Address
- Phone, email
- Tax registration number
- Business hours
- Social media links
- Website URL

FR-SET-002: Branch Management

- **Description:** Manage multiple salon locations
- **Features:**

- Add/Edit/Delete branches
- Branch name and address
- Contact information
- Branch manager
- Branch-specific settings
- Branch status (active/inactive)

4.10.2 Tax & Currency

FR-SET-003: Tax Configuration

- **Description:** Configure tax rates and types
- **Features:**
- Add multiple tax types
- Set tax rates
- Tax inclusive/exclusive
- Default tax selection
- Tax exemption categories

FR-SET-004: Currency Settings

- **Description:** Set display currency
- **Settings:**
- Currency code (USD, EUR, LKR, etc.)
- Currency symbol
- Decimal places
- Thousand separator
- Decimal separator

4.10.3 Notification Settings

FR-SET-005: SMS Configuration

- **Description:** Configure SMS gateway
- **Settings:**
- SMS provider selection
- API credentials
- Sender ID
- SMS templates
- SMS balance tracking
- Test SMS functionality

FR-SET-006: WhatsApp Configuration

- **Description:** Configure WhatsApp Business API
- **Settings:**
- WhatsApp API credentials
- Business account details
- Message templates
- Webhook configuration
- Test message functionality

FR-SET-007: Email Configuration

- **Description:** Configure email settings
- **Settings:**
- SMTP server details
- Sender email and name
- Email templates
- Email signature
- Test email functionality

4.10.4 Booking Settings

FR-SET-008: Booking Configuration

- **Description:** Configure online booking parameters
- **Settings:**
- Booking advance days (how far customers can book)
- Minimum booking notice (hours before appointment)
- Maximum bookings per slot
- Booking modification policy
- Cancellation policy
- No-show policy
- Booking form fields (required/optional)

FR-SET-009: Service Configuration

- **Description:** Define services offered
- **Features:**
- Add/Edit/Delete services
- Service name and description
- Duration
- Price
- Category
- Commission rate
- Product usage mapping
- Service availability

4.10.5 Payment Settings

FR-SET-010: Payment Gateway Configuration

- **Description:** Configure payment gateway integration
- **Settings:**
- Gateway selection (Stripe, PayPal, local gateway)
- API credentials
- Merchant account details
- Payment methods enabled
- Transaction fees
- Test mode toggle

4.11 Expense Management

FR-EXP-001: Record Expenses

- **Description:** Track business expenses
- **Inputs:**
- Expense date
- Category (salary, goods, utilities, marketing, etc.)
- Amount
- Description
- Payment method
- Branch
- Receipt upload
- **Features:**
- Custom expense categories
- Recurring expenses
- Expense approval workflow (optional)

FR-EXP-002: Expense Categories

- **Description:** Organize expenses by category

- **Default Categories:**
- Salaries
- Product purchases (goods)
- Rent
- Utilities
- Marketing
- Maintenance
- Miscellaneous
- **Features:**
- Add/Edit custom categories
- Category-wise reporting

5. Non-Functional Requirements

5.1 Performance Requirements

NFR-PERF-001: Response Time

- API response time: ≤ 2 seconds for 95% of requests
- Page load time: ≤ 3 seconds on standard broadband
- Appointment assignment: ≤ 2 seconds
- Report generation: ≤ 10 seconds for standard reports
- Search operations: ≤ 1 second

NFR-PERF-002: Throughput

- Support 50 concurrent booking users
- Support 20 concurrent dashboard users
- Handle 1000 appointments per day
- Process 500 POS transactions per day

NFR-PERF-003: Database Performance

- Query execution: ≤ 100ms for simple queries
- Database indexing on frequently queried fields
- Query optimization for reports
- Database connection pooling

NFR-PERF-004: Scalability

- Horizontal scaling capability
- Load balancing support
- Database replication for read operations
- Caching strategy for frequently accessed data

5.2 Security Requirements

NFR-SEC-001: Authentication

- JWT-based authentication
- Token expiry: 24 hours
- Refresh token mechanism
- Secure password storage (bcrypt hashing)
- Password complexity requirements:
- Minimum 8 characters
- At least 1 uppercase letter
- At least 1 number
- At least 1 special character

NFR-SEC-002: Authorization

- Role-based access control (RBAC)

- Granular permission system
- API endpoint protection
- Resource-level permissions
- Audit trail for sensitive operations

NFR-SEC-003: Data Protection

- HTTPS/TLS encryption for all communications
- Encrypted storage for sensitive data (payment info, personal data)
- SQL injection prevention (prepared statements)
- XSS protection
- CSRF protection
- Input validation and sanitization
- Output encoding

NFR-SEC-004: API Security

- Rate limiting (60 requests per minute per user)
- API key validation for public endpoints
- Request throttling
- IP whitelisting for admin APIs (optional)
- CORS configuration

NFR-SEC-005: Session Management

- Secure session handling
- Session timeout (24 hours of inactivity)
- Concurrent session control
- Logout functionality
- Remember me with secure token

NFR-SEC-006: Payment Security

- PCI DSS compliance for payment processing
- No storage of full card details
- Tokenized payment information
- Secure payment gateway integration
- Transaction encryption

NFR-SEC-007: Data Privacy

- Compliance with data protection regulations
- User consent for data collection
- Right to access personal data
- Right to delete personal data
- Data retention policy
- Privacy policy display

NFR-SEC-008: Audit Logging

- Log all authentication attempts
- Log all permission changes
- Log all financial transactions
- Log all data modifications
- Secure log storage
- Log retention: 1 year minimum

5.3 Usability Requirements

NFR-USE-001: User Interface

- Intuitive and clean design
- Consistent layout across pages
- Clear navigation structure
- Helpful error messages

- Contextual help and tooltips
- Keyboard shortcuts for common actions

NFR-USE-002: Responsiveness

- Mobile-first design approach
- Responsive layout for all screen sizes
- Touch-friendly interface on mobile
- Minimum supported resolution: 375x667 (mobile), 1024x768 (desktop)

NFR-USE-003: Accessibility

- WCAG 2.1 Level AA compliance
- Keyboard navigation support
- Screen reader compatibility
- Color contrast ratios
- Alt text for images
- Accessible form labels

NFR-USE-004: Learnability

- Maximum 30 minutes training for operators
- Intuitive booking flow (≤ 5 steps)
- Onboarding wizard for initial setup
- In-app tutorials
- Comprehensive documentation

NFR-USE-005: Feedback

- Loading indicators for operations
- Success/error notifications
- Confirmation dialogs for destructive actions
- Progress bars for long operations
- Toast notifications for background tasks

5.4 Reliability Requirements

NFR-REL-001: Availability

- System uptime: 99.5% (excluding scheduled maintenance)
- Scheduled maintenance windows: Off-peak hours
- Maximum downtime: 4 hours per month

NFR-REL-002: Fault Tolerance

- Graceful degradation on component failure
- Automatic retry for failed external API calls (max 3 attempts)
- Fallback mechanisms for critical operations
- Error logging and monitoring

NFR-REL-003: Data Integrity

- ACID transactions for critical operations
- Database constraints and validations
- Referential integrity
- Data validation at multiple layers
- Automated database backups

NFR-REL-004: Backup and Recovery

- Automated daily database backups
- Backup retention: 30 days
- Recovery Time Objective (RTO): 4 hours
- Recovery Point Objective (RPO): 24 hours
- Disaster recovery plan

5.5 Maintainability Requirements

NFR-MAIN-001: Code Quality

- Clean code principles
- Consistent coding standards (PSR-12 for PHP, Airbnb for JavaScript)
- Code comments and documentation
- DRY (Don't Repeat Yourself) principle
- SOLID principles

NFR-MAIN-002: Modularity

- Modular architecture
- Loose coupling between components
- High cohesion within modules
- Service-oriented design
- Reusable components

NFR-MAIN-003: Testing

- Unit test coverage: ≥ 70%
- Integration test coverage for critical flows
- Automated testing in CI/CD pipeline
- Manual QA testing before releases

NFR-MAIN-004: Documentation

- API documentation (Swagger/OpenAPI)
- Code documentation
- User manuals
- Admin guides
- Database schema documentation
- Deployment documentation

NFR-MAIN-005: Version Control

- Git-based version control
- Feature branch workflow
- Pull request reviews
- Semantic versioning
- Change logs

5.6 Compatibility Requirements

NFR-COMP-001: Browser Compatibility

- Google Chrome (latest 2 versions)
- Mozilla Firefox (latest 2 versions)
- Apple Safari (latest 2 versions)
- Microsoft Edge (latest 2 versions)

NFR-COMP-002: Device Compatibility

- Desktop computers (Windows, macOS, Linux)
- Tablets (iOS, Android)
- Mobile phones (iOS, Android)

NFR-COMP-003: Database Compatibility

- MySQL 8.0 or higher
- MariaDB 10.5 or higher (optional)

NFR-COMP-004: Third-Party Integrations

- SMS gateway API compatibility
- WhatsApp Business API compatibility
- Payment gateway API compatibility

- Email service provider compatibility

5.7 Localization Requirements

NFR-LOC-001: Language Support

- Initial: English
- Support for multiple languages (extensible)
- Language selection in settings
- Translatable UI text
- Date and time formatting per locale

NFR-LOC-002: Currency Support

- Configurable currency per branch
- Currency symbol display
- Proper number formatting

NFR-LOC-003: Time Zone

- Server time zone: UTC
- Display time in local time zone
- Time zone configuration per branch

6. Technology Stack

6.1 Backend Technologies

6.1.1 Core Framework

- **Laravel 10.x**
- PHP 8.1+
- Composer for dependency management
- Laravel Sanctum for API authentication
- Laravel Eloquent ORM
- Laravel Migrations for database versioning

6.1.2 Database

- **MySQL 8.0+**
- InnoDB storage engine
- UTF-8 character set
- Database indexing strategy

6.1.3 Key Laravel Packages

```
{   "require": {     "laravel/framework": "^10.0",     "laravel/sanctum": "^3.0",
"spatie/laravel-permission": "^5.0",     "maatwebsite/excel": "^3.1",
"intervention/image": "^2.7",     "barryvdh/laravel-dompdf": "^2.0",
"laravel/horizon": "^5.0",     "predis/predis": "^2.0",     "guzzlehttp/guzzle":
"^7.0",     "twilio/sdk": "^7.0",     "laravel/cashier": "^15.0"   },   "require-
dev": {     "phpunit/phpunit": "^10.0",     "mockery/mockery": "^1.4",
"fakerphp/faker": "^1.9",     "laravel/pint": "^1.0",     "nunomaduro/collision":
"^7.0",     "phpstan/phpstan": "^1.0"   } }
```

6.1.4 Additional Backend Tools

- **Redis**: Caching and session management
- **Laravel Queue**: Background job processing
- **Laravel Scheduler**: Cron job management
- **Laravel Notifications**: Multi-channel notifications
- **Laravel Events**: Event-driven architecture
- **Swagger/OpenAPI**: API documentation

6.2 Frontend Technologies

6.2.1 Core Framework

- **React 18.x**
- Node.js 18.x+
- npm/Yarn for package management
- Create React App or Vite for build tooling
- TypeScript (optional but recommended)

6.2.2 State Management

- **Redux Toolkit**
- Global state management
- Redux Persist for state persistence
- RTK Query for API integration

6.2.3 UI Framework

- **Material-UI (MUI)** or **Ant Design**
- Component library
- Theme customization
- Responsive grid system
- Icon library

6.2.4 Key React Packages

```
{   "dependencies": {     "react": "^18.2.0",     "react-dom": "^18.2.0",
"react-router-dom": "^6.8.0",     "@reduxjs/toolkit": "^1.9.0",     "react-redux":
"^8.0.0",     "redux-persist": "^6.0.0",     "axios": "^1.3.0",
"@mui/material": "^5.11.0",     "@mui/icons-material": "^5.11.0",
"@emotion/react": "^11.10.0",     "@emotion/styled": "^11.10.0",     "react-hook-
form": "^7.43.0",     "yup": "^1.0.0",     "date-fns": "^2.29.0",
"@fullcalendar/react": "^6.1.0",     "@fullcalendar/daygrid": "^6.1.0",
"@fullcalendar/timegrid": "^6.1.0",     "@fullcalendar/interaction": "^6.1.0",
"recharts": "^2.5.0",     "react-barcode": "^1.4.0",     "react-to-print":
"^2.14.0",     "react-hot-toast": "^2.4.0",     "react-loading-skeleton": "^3.1.0",
"lodash": "^4.17.0"   },   "devDependencies": {     "@types/react": "^18.0.0",
"@types/react-dom": "^18.0.0",     "@vitejs/plugin-react": "^3.0.0",     "vite":
"^4.0.0",     "eslint": "^8.0.0",     "prettier": "^2.8.0"   } }
```

6.2.5 Additional Frontend Tools

- **Axios**: HTTP client
- **React Hook Form**: Form management
- **Yup**: Schema validation
- **Date-fns**: Date manipulation
- **FullCalendar**: Calendar component
- **Recharts**: Data visualization
- **React-Barcode**: Barcode generation
- **React-to-Print**: Print functionality

6.3 Development Tools

6.3.1 Version Control

- **Git**: Source code management
- **GitHub/GitLab/Bitbucket**: Repository hosting

6.3.2 IDE/Editors

- **VS Code**: Recommended editor

- **PHPStorm**: For PHP development
- **WebStorm**: For React development

6.3.3 API Testing

- **Postman**: API testing and documentation
- **Insomnia**: Alternative API client

6.3.4 Code Quality

- **PHP CS Fixer**: PHP code formatting
- **Laravel Pint**: Laravel code style
- **ESLint**: JavaScript linting
- **Prettier**: Code formatting
- **PHPStan**: Static analysis for PHP

6.3.5 Testing

- **PHPUnit**: Backend unit testing
- **Jest**: Frontend unit testing
- **React Testing Library**: Component testing
- **Cypress**: End-to-end testing

6.4 DevOps and Deployment

6.4.1 Containerization

- **Docker**: Containerization
- **Docker Compose**: Multi-container orchestration

6.4.2 CI/CD

- **GitHub Actions**: Automated workflows
- **GitLab CI**: Alternative CI/CD
- **Jenkins**: Enterprise CI/CD

6.4.3 Server Environment

- **Web Server**: Nginx or Apache
- **PHP-FPM**: PHP process manager
- **Supervisor**: Process control system for queue workers
- **Certbot**: SSL certificate management

6.4.4 Hosting Options

- **VPS**: DigitalOcean, Linode, Vultr
- **Cloud**: AWS (EC2, RDS), Google Cloud, Azure
- **Shared Hosting**: cPanel-based hosting (not recommended for production)

6.4.5 Monitoring

- **Laravel Telescope**: Development debugging
- **Laravel Horizon**: Queue monitoring
- **Sentry**: Error tracking
- **New Relic**: Application performance monitoring
- **Google Analytics**: User analytics

6.5 External Services

6.5.1 SMS Gateway

- **Twilio**: Recommended SMS provider
- **Vonage (Nexmo)**: Alternative SMS provider
- **Local SMS Gateway**: Country-specific providers

6.5.2 WhatsApp Integration

- **Twilio WhatsApp API**: WhatsApp Business API
- **Official WhatsApp Business API**: Direct integration

6.5.3 Payment Gateways

- **Stripe**: International payments
- **PayPal**: Alternative payment processor
- **Local Payment Gateways**: Country-specific gateways

6.5.4 Email Service

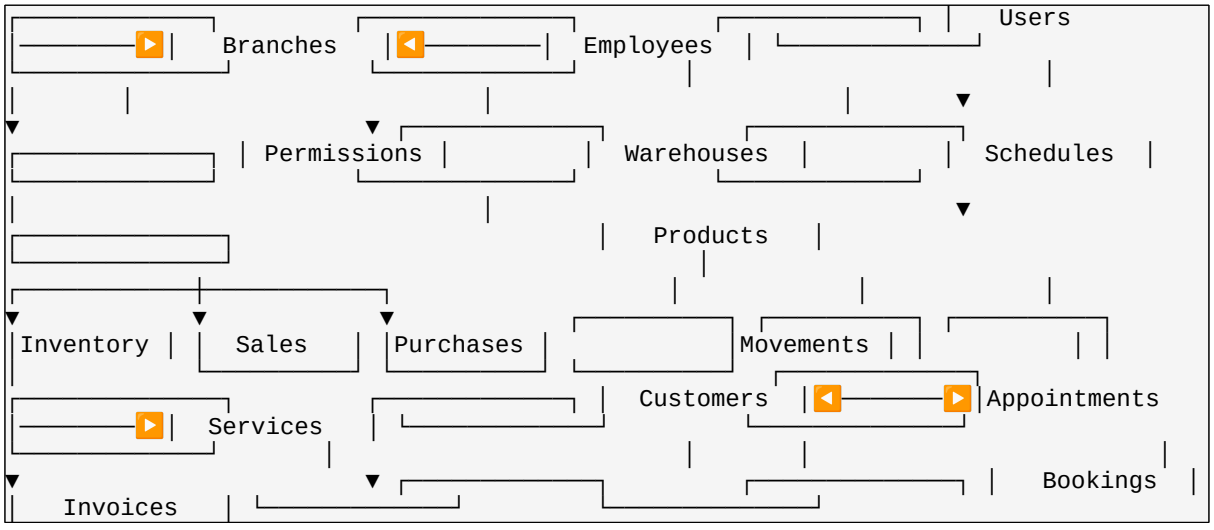
- **SendGrid**: Transactional emails
- **Mailgun**: Email delivery service
- **Amazon SES**: Cost-effective email service
- **SMTP**: Traditional email delivery

6.5.5 File Storage

- **Amazon S3**: Cloud file storage
- **DigitalOcean Spaces**: S3-compatible storage
- **Local Storage**: On-server file storage

7. Database Design

7.1 Entity Relationship Diagram (ERD)



7.2 Database Tables

7.2.1 User Management

users

```
CREATE TABLE users (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,     name
VARCHAR(255) NOT NULL,     email VARCHAR(255) UNIQUE NOT NULL,     password
```

```
VARCHAR(255) NOT NULL,     phone VARCHAR(20),     role ENUM('admin', 'operator')
NOT NULL DEFAULT 'operator',     branch_id BIGINT UNSIGNED,     status
ENUM('active', 'inactive') DEFAULT 'active',     email_verified_at TIMESTAMP NULL,
remember_token VARCHAR(100),     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
FOREIGN KEY (branch_id) REFERENCES branches(id) ON DELETE SET NULL,     INDEX
idx_email (email),     INDEX idx_role (role),     INDEX idx_status (status) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

roles

```
CREATE TABLE roles (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,     name
VARCHAR(255) UNIQUE NOT NULL,     display_name VARCHAR(255),     description TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

permissions

```
CREATE TABLE permissions (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(255) UNIQUE NOT NULL,     display_name VARCHAR(255),     description
TEXT,     module VARCHAR(100),     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
INDEX idx_module (module) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

role_permissions

```
CREATE TABLE role_permissions (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
role_id BIGINT UNSIGNED NOT NULL,     permission_id BIGINT UNSIGNED NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     FOREIGN KEY (role_id)
REFERENCES roles(id) ON DELETE CASCADE,     FOREIGN KEY (permission_id) REFERENCES
permissions(id) ON DELETE CASCADE,     UNIQUE KEY unique_role_permission (role_id,
permission_id) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.2 Branch and Location Management

branches

```
CREATE TABLE branches (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,     name
VARCHAR(255) NOT NULL,     code VARCHAR(50) UNIQUE,     address TEXT,     city
VARCHAR(100),     state VARCHAR(100),     postal_code VARCHAR(20),     country
VARCHAR(100),     phone VARCHAR(20),     email VARCHAR(255),     manager_id BIGINT
UNSIGNED,     status ENUM('active', 'inactive') DEFAULT 'active',     opening_time
TIME,     closing_time TIME,     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
INDEX idx_status (status),     INDEX idx_code (code) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

warehouses

```
CREATE TABLE warehouses (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(255) NOT NULL,     code VARCHAR(50) UNIQUE,     branch_id BIGINT
UNSIGNED,     address TEXT,     phone VARCHAR(20),     status ENUM('active',
'inactive') DEFAULT 'active',     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
FOREIGN KEY (branch_id) REFERENCES branches(id) ON DELETE SET NULL,     INDEX
idx_branch (branch_id),     INDEX idx_status (status) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.3 Product and Inventory Management

categories

```
CREATE TABLE categories (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(255) NOT NULL,     parent_id BIGINT UNSIGNED NULL,     description
TEXT,     status ENUM('active', 'inactive') DEFAULT 'active',     created_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY (parent_id)
REFERENCES categories(id) ON DELETE SET NULL,     INDEX idx_parent (parent_id),
```

```
INDEX idx_status (status) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

products

```
CREATE TABLE products (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,     sku
VARCHAR(100) UNIQUE NOT NULL,     name VARCHAR(255) NOT NULL,     description TEXT,
category_id BIGINT UNSIGNED,     brand VARCHAR(100),     barcode VARCHAR(255),
unit VARCHAR(50) DEFAULT 'piece',     cost_price DECIMAL(10, 2) NOT NULL DEFAULT
0.00,     selling_price DECIMAL(10, 2) NOT NULL,     product_type ENUM('retail',
'consumable', 'both') NOT NULL DEFAULT 'retail',     usage_count INT DEFAULT 1
COMMENT 'Number of uses per unit for consumables',     reorder_level INT DEFAULT 0,
tax_rate DECIMAL(5, 2) DEFAULT 0.00,     image VARCHAR(255),     status
ENUM('active', 'inactive') DEFAULT 'active',     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (category_id) REFERENCES categories(id) ON
DELETE SET NULL,     INDEX idx_sku (sku),     INDEX idx_category (category_id),
INDEX idx_barcode (barcode),     INDEX idx_status (status),     INDEX
idx_product_type (product_type) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

inventory

```
CREATE TABLE inventory (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
product_id BIGINT UNSIGNED NOT NULL,     warehouse_id BIGINT UNSIGNED NOT NULL,
retail_quantity INT NOT NULL DEFAULT 0,     consumable_quantity INT NOT NULL
DEFAULT 0,     consumable_usage_count INT NOT NULL DEFAULT 0 COMMENT 'Remaining
uses for consumables',     last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY (product_id) REFERENCES products(id) ON
DELETE CASCADE,     FOREIGN KEY (warehouse_id) REFERENCES warehouses(id) ON DELETE
CASCADE,     UNIQUE KEY unique_product_warehouse (product_id, warehouse_id),
INDEX idx_product (product_id),     INDEX idx_warehouse (warehouse_id) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

inventory_movements

```
CREATE TABLE inventory_movements (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     product_id BIGINT UNSIGNED NOT NULL,     warehouse_id BIGINT UNSIGNED NOT
NULL,     movement_type ENUM('purchase', 'sale', 'transfer_in', 'transfer_out',
'adjustment', 'consumption', 'return') NOT NULL,     quantity INT NOT NULL,
stock_type ENUM('retail', 'consumable') NOT NULL,     usage_count INT DEFAULT 0
COMMENT 'Usage count for consumable movements',     reference_type VARCHAR(100)
COMMENT 'purchases, sales, transfers, etc',     reference_id BIGINT UNSIGNED
COMMENT 'ID of related record',     notes TEXT,     created_by BIGINT UNSIGNED,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     FOREIGN KEY (product_id)
REFERENCES products(id) ON DELETE CASCADE,     FOREIGN KEY (warehouse_id)
REFERENCES warehouses(id) ON DELETE CASCADE,     FOREIGN KEY (created_by)
REFERENCES users(id) ON DELETE SET NULL,     INDEX idx_product (product_id),
INDEX idx_warehouse (warehouse_id),     INDEX idx_movement_type (movement_type),
INDEX idx_reference (reference_type, reference_id),     INDEX idx_created_at
(created_at) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

stock_transfers

```
CREATE TABLE stock_transfers (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
transfer_number VARCHAR(100) UNIQUE NOT NULL,     source_warehouse_id BIGINT
UNSIGNED NOT NULL,     destination_warehouse_id BIGINT UNSIGNED NOT NULL,
transfer_date DATE NOT NULL,     status ENUM('pending', 'completed', 'cancelled')
DEFAULT 'pending',     notes TEXT,     created_by BIGINT UNSIGNED,     created_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY
(source_warehouse_id) REFERENCES warehouses(id),     FOREIGN KEY
(destination_warehouse_id) REFERENCES warehouses(id),     FOREIGN KEY (created_by)
REFERENCES users(id) ON DELETE SET NULL,     INDEX idx_transfer_number
(transfer_number),     INDEX idx_source (source_warehouse_id),     INDEX
idx_destination (destination_warehouse_id),     INDEX idx_status (status) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

stock_transfer_items

```
CREATE TABLE stock_transfer_items (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     transfer_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT
NULL,     quantity INT NOT NULL,     stock_type ENUM('retail', 'consumable') NOT
NULL,     FOREIGN KEY (transfer_id) REFERENCES stock_transfers(id) ON DELETE
CASCADE,     FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE,
INDEX idx_transfer (transfer_id),     INDEX idx_product (product_id) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.4 Supplier and Purchase Management

suppliers

```
CREATE TABLE suppliers (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(255) NOT NULL,     contact_person VARCHAR(255),     phone VARCHAR(20),
email VARCHAR(255),     address TEXT,     city VARCHAR(100),     state
VARCHAR(100),     postal_code VARCHAR(20),     country VARCHAR(100),     tax_number
VARCHAR(100),     payment_terms VARCHAR(255),     status ENUM('active', 'inactive')
DEFAULT 'active',     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
INDEX idx_name (name),     INDEX idx_status (status) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

purchases

```
CREATE TABLE purchases (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
purchase_number VARCHAR(100) UNIQUE NOT NULL,     supplier_id BIGINT UNSIGNED NOT
NULL,     warehouse_id BIGINT UNSIGNED NOT NULL,     purchase_date DATE NOT NULL,
expected_delivery_date DATE,     status ENUM('pending', 'received', 'partial',
'cancelled') DEFAULT 'pending',     subtotal DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
tax_amount DECIMAL(10, 2) NOT NULL DEFAULT 0.00,     discount_amount DECIMAL(10, 2)
NOT NULL DEFAULT 0.00,     total_amount DECIMAL(10, 2) NOT NULL,     payment_status
ENUM('unpaid', 'partial', 'paid') DEFAULT 'unpaid',     payment_terms VARCHAR(255),
notes TEXT,     created_by BIGINT UNSIGNED,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (supplier_id) REFERENCES suppliers(id),
FOREIGN KEY (warehouse_id) REFERENCES warehouses(id),     FOREIGN KEY (created_by)
REFERENCES users(id) ON DELETE SET NULL,     INDEX idx_purchase_number
(purchase_number),     INDEX idx_supplier (supplier_id),     INDEX idx_warehouse
(warehouse_id),     INDEX idx_status (status),     INDEX idx_purchase_date
(purchase_date) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

purchase_items

```
CREATE TABLE purchase_items (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
purchase_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT NULL,
quantity INT NOT NULL,     received_quantity INT DEFAULT 0,     unit_cost
DECIMAL(10, 2) NOT NULL,     total_cost DECIMAL(10, 2) NOT NULL,     FOREIGN KEY
(purchase_id) REFERENCES purchases(id) ON DELETE CASCADE,     FOREIGN KEY
(product_id) REFERENCES products(id) ON DELETE CASCADE,     INDEX idx_purchase
(purchase_id),     INDEX idx_product (product_id) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

purchase_returns

```
CREATE TABLE purchase_returns (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
return_number VARCHAR(100) UNIQUE NOT NULL,     purchase_id BIGINT UNSIGNED NOT
NULL,     supplier_id BIGINT UNSIGNED NOT NULL,     warehouse_id BIGINT UNSIGNED
NOT NULL,     return_date DATE NOT NULL,     reason TEXT,     total_amount
DECIMAL(10, 2) NOT NULL,     refund_status ENUM('pending', 'completed') DEFAULT
'pending',     created_by BIGINT UNSIGNED,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (purchase_id) REFERENCES purchases(id),
FOREIGN KEY (supplier_id) REFERENCES suppliers(id),     FOREIGN KEY (warehouse_id)
REFERENCES warehouses(id),     FOREIGN KEY (created_by) REFERENCES users(id) ON
DELETE SET NULL,     INDEX idx_return_number (return_number),     INDEX
idx_purchase (purchase_id),     INDEX idx_supplier (supplier_id) ) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

purchase_return_items

```
CREATE TABLE purchase_return_items (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     return_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT
NULL,     quantity INT NOT NULL,     unit_cost DECIMAL(10, 2) NOT NULL,
total_cost DECIMAL(10, 2) NOT NULL,     FOREIGN KEY (return_id) REFERENCES
purchase_returns(id) ON DELETE CASCADE,     FOREIGN KEY (product_id) REFERENCES
products(id) ON DELETE CASCADE,     INDEX idx_return (return_id),     INDEX
idx_product (product_id) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

quotations

```
CREATE TABLE quotations (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
quotation_number VARCHAR(100) UNIQUE NOT NULL,     customer_id BIGINT UNSIGNED,
supplier_id BIGINT UNSIGNED,     quotation_type ENUM('customer', 'supplier') NOT
NULL,     quotation_date DATE NOT NULL,     expiry_date DATE,     status
ENUM('draft', 'sent', 'accepted', 'rejected', 'expired') DEFAULT 'draft',
subtotal DECIMAL(10, 2) NOT NULL DEFAULT 0.00,     tax_amount DECIMAL(10, 2) NOT
NULL DEFAULT 0.00,     discount_amount DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
total_amount DECIMAL(10, 2) NOT NULL,     terms_conditions TEXT,     notes TEXT,
created_by BIGINT UNSIGNED,     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE SET NULL,     FOREIGN
KEY (supplier_id) REFERENCES suppliers(id) ON DELETE SET NULL,     FOREIGN KEY
(created_by) REFERENCES users(id) ON DELETE SET NULL,     INDEX
idx_quotation_number (quotation_number),     INDEX idx_customer (customer_id),
INDEX idx_supplier (supplier_id),     INDEX idx_status (status) ) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

quotation_items

```
CREATE TABLE quotation_items (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
quotation_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT NULL,
quantity INT NOT NULL,     unit_price DECIMAL(10, 2) NOT NULL,     total_price
DECIMAL(10, 2) NOT NULL,     FOREIGN KEY (quotation_id) REFERENCES quotations(id)
ON DELETE CASCADE,     FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE
CASCADE,     INDEX idx_quotation (quotation_id),     INDEX idx_product (product_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.5 Customer Management

customers

```
CREATE TABLE customers (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
customer_number VARCHAR(100) UNIQUE,     name VARCHAR(255) NOT NULL,     phone
VARCHAR(20) NOT NULL UNIQUE,     email VARCHAR(255),     date_of_birth DATE,
gender ENUM('male', 'female', 'other'),     address TEXT,     city VARCHAR(100),
state VARCHAR(100),     postal_code VARCHAR(20),     country VARCHAR(100),
preferred_employee_id BIGINT UNSIGNED,     communication_preference ENUM('sms',
'whatsapp', 'email', 'all') DEFAULT 'all',     notes TEXT,     total_visits INT
DEFAULT 0,     total_spending DECIMAL(10, 2) DEFAULT 0.00,     last_visit_date
DATE,     registration_date DATE,     status ENUM('active', 'inactive') DEFAULT
'active',     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY
(preferred_employee_id) REFERENCES employees(id) ON DELETE SET NULL,     INDEX
idx_phone (phone),     INDEX idx_email (email),     INDEX idx_customer_number
(customer_number),     INDEX idx_status (status) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.6 Employee Management

employees

```
CREATE TABLE employees (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
employee_number VARCHAR(100) UNIQUE NOT NULL,     name VARCHAR(255) NOT NULL,
role ENUM('hairdresser', 'receptionist', 'manager', 'other') NOT NULL,     phone
VARCHAR(20) NOT NULL,     email VARCHAR(255),     date_of_birth DATE,
joining_date DATE NOT NULL,     photo VARCHAR(255),     address TEXT,     city
VARCHAR(100),     state VARCHAR(100),     postal_code VARCHAR(20),     country
VARCHAR(100),     emergency_contact_name VARCHAR(255),     emergency_contact_phone
VARCHAR(20),     branch_id BIGINT UNSIGNED,     skills TEXT COMMENT 'JSON array of
```

```
skills/specializations',     commission_rate DECIMAL(5, 2) DEFAULT 0.00,     status
ENUM('active', 'inactive') DEFAULT 'active',     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (branch_id) REFERENCES branches(id) ON DELETE
SET NULL,     INDEX idx_employee_number (employee_number),     INDEX idx_role
(role),     INDEX idx_branch (branch_id),     INDEX idx_status (status) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

employee_schedules

```
CREATE TABLE employee_schedules (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     employee_id BIGINT UNSIGNED NOT NULL,     day_of_week TINYINT NOT NULL
COMMENT '0=Sunday, 6=Saturday',     start_time TIME NOT NULL,     end_time TIME NOT
NULL,     is_available BOOLEAN DEFAULT TRUE,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (employee_id) REFERENCES employees(id) ON DELETE
CASCADE,     INDEX idx_employee (employee_id),     INDEX idx_day (day_of_week) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

employee_time_off

```
CREATE TABLE employee_time_off (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
employee_id BIGINT UNSIGNED NOT NULL,     start_date DATE NOT NULL,     end_date
DATE NOT NULL,     reason VARCHAR(255),     status ENUM('pending', 'approved',
'rejected') DEFAULT 'pending',     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
FOREIGN KEY (employee_id) REFERENCES employees(id) ON DELETE CASCADE,     INDEX
idx_employee (employee_id),     INDEX idx_dates (start_date, end_date) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.7 Services

services

```
CREATE TABLE services (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,     name
VARCHAR(255) NOT NULL,     description TEXT,     category VARCHAR(100),
duration INT NOT NULL COMMENT 'Duration in minutes',     price DECIMAL(10, 2) NOT
NULL,     commission_rate DECIMAL(5, 2) DEFAULT 0.00,     status ENUM('active',
'inactive') DEFAULT 'active',     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
INDEX idx_name (name),     INDEX idx_category (category),     INDEX idx_status
(status) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

service_products

```
CREATE TABLE service_products (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
service_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT NULL,
usage_count INT DEFAULT 1 COMMENT 'Number of uses consumed per service',
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     FOREIGN KEY (service_id)
REFERENCES services(id) ON DELETE CASCADE,     FOREIGN KEY (product_id) REFERENCES
products(id) ON DELETE CASCADE,     UNIQUE KEY unique_service_product (service_id,
product_id),     INDEX idx_service (service_id),     INDEX idx_product (product_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.8 Appointments

appointments

```
CREATE TABLE appointments (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
appointment_number VARCHAR(100) UNIQUE NOT NULL,     customer_id BIGINT UNSIGNED
NOT NULL,     employee_id BIGINT UNSIGNED,     branch_id BIGINT UNSIGNED NOT NULL,
appointment_date DATE NOT NULL,     appointment_time TIME NOT NULL,     status
ENUM('scheduled', 'confirmed', 'in_progress', 'completed', 'cancelled', 'no_show')
DEFAULT 'scheduled',     booking_source ENUM('online', 'walk_in', 'phone') DEFAULT
'walk_in',     notes TEXT,     cancellation_reason TEXT,     created_by BIGINT
UNSIGNED,     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY
(customer_id) REFERENCES customers(id) ON DELETE CASCADE,     FOREIGN KEY
(employee_id) REFERENCES employees(id) ON DELETE SET NULL,     FOREIGN KEY
(branch_id) REFERENCES branches(id) ON DELETE CASCADE,     FOREIGN KEY (created_by)
```

```
REFERENCES users(id) ON DELETE SET NULL,     INDEX idx_appointment_number
(appointment_number),     INDEX idx_customer (customer_id),     INDEX idx_employee
(employee_id),     INDEX idx_branch (branch_id),     INDEX idx_appointment_date
(appointment_date),     INDEX idx_status (status),     INDEX idx_booking_source
(booking_source) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

appointment_services

```
CREATE TABLE appointment_services (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     appointment_id BIGINT UNSIGNED NOT NULL,     service_id BIGINT UNSIGNED
NOT NULL,     price DECIMAL(10, 2) NOT NULL,     FOREIGN KEY (appointment_id)
REFERENCES appointments(id) ON DELETE CASCADE,     FOREIGN KEY (service_id)
REFERENCES services(id) ON DELETE CASCADE,     INDEX idx_appointment
(appointment_id),     INDEX idx_service (service_id) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

bookings

```
CREATE TABLE bookings (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
booking_number VARCHAR(100) UNIQUE NOT NULL,     customer_name VARCHAR(255) NOT
NULL,     customer_phone VARCHAR(20) NOT NULL,     customer_email VARCHAR(255),
appointment_date DATE NOT NULL,     appointment_time TIME NOT NULL,     service_id
BIGINT UNSIGNED NOT NULL,     branch_id BIGINT UNSIGNED NOT NULL,     otp_code
VARCHAR(6),     otp_verified_at TIMESTAMP NULL,     otp_expires_at TIMESTAMP NULL,
status ENUM('pending', 'confirmed', 'cancelled') DEFAULT 'pending',
appointment_id BIGINT UNSIGNED COMMENT 'Linked appointment after confirmation',
ip_address VARCHAR(45),     user_agent TEXT,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (service_id) REFERENCES services(id),
FOREIGN KEY (branch_id) REFERENCES branches(id),     FOREIGN KEY (appointment_id)
REFERENCES appointments(id) ON DELETE SET NULL,     INDEX idx_booking_number
(booking_number),     INDEX idx_phone (customer_phone),     INDEX idx_date_time
(appointment_date, appointment_time),     INDEX idx_status (status) ) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.9 Sales and Invoicing

sales

```
CREATE TABLE sales (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
sale_number VARCHAR(100) UNIQUE NOT NULL,     customer_id BIGINT UNSIGNED,
branch_id BIGINT UNSIGNED NOT NULL,     warehouse_id BIGINT UNSIGNED NOT NULL,
sale_date DATE NOT NULL,     sale_type ENUM('retail', 'service') DEFAULT 'retail',
appointment_id BIGINT UNSIGNED COMMENT 'Linked if sale is for service',
subtotal DECIMAL(10, 2) NOT NULL DEFAULT 0.00,     tax_amount DECIMAL(10, 2) NOT
NULL DEFAULT 0.00,     discount_amount DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
shipping_charges DECIMAL(10, 2) NOT NULL DEFAULT 0.00,     total_amount DECIMAL(10,
2) NOT NULL,     payment_method ENUM('cash', 'card', 'bank_transfer', 'gateway')
NOT NULL,     payment_status ENUM('pending', 'completed') DEFAULT 'completed',
notes TEXT,     created_by BIGINT UNSIGNED,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE
SET NULL,     FOREIGN KEY (branch_id) REFERENCES branches(id),     FOREIGN KEY
(warehouse_id) REFERENCES warehouses(id),     FOREIGN KEY (appointment_id)
REFERENCES appointments(id) ON DELETE SET NULL,     FOREIGN KEY (created_by)
REFERENCES users(id) ON DELETE SET NULL,     INDEX idx_sale_number (sale_number),
INDEX idx_customer (customer_id),     INDEX idx_branch (branch_id),     INDEX
idx_sale_date (sale_date),     INDEX idx_sale_type (sale_type),     INDEX
idx_payment_status (payment_status) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

sale_items

```
CREATE TABLE sale_items (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
sale_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT NULL,
quantity INT NOT NULL,     unit_price DECIMAL(10, 2) NOT NULL,     discount_amount
DECIMAL(10, 2) DEFAULT 0.00,     tax_amount DECIMAL(10, 2) DEFAULT 0.00,
total_price DECIMAL(10, 2) NOT NULL,     FOREIGN KEY (sale_id) REFERENCES sales(id)
ON DELETE CASCADE,     FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE
```

```
CASCADE,     INDEX idx_sale (sale_id),     INDEX idx_product (product_id) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

sale_returns

```
CREATE TABLE sale_returns (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
return_number VARCHAR(100) UNIQUE NOT NULL,     sale_id BIGINT UNSIGNED NOT NULL,
customer_id BIGINT UNSIGNED,     return_date DATE NOT NULL,     reason TEXT,
total_amount DECIMAL(10, 2) NOT NULL,     refund_method ENUM('cash', 'card',
'bank_transfer') NOT NULL,     refund_status ENUM('pending', 'completed') DEFAULT
'pending',     created_by BIGINT UNSIGNED,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     FOREIGN KEY (sale_id) REFERENCES sales(id),     FOREIGN KEY
(customer_id) REFERENCES customers(id) ON DELETE SET NULL,     FOREIGN KEY
(created_by) REFERENCES users(id) ON DELETE SET NULL,     INDEX idx_return_number
(return_number),     INDEX idx_sale (sale_id),     INDEX idx_customer (customer_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

sale_return_items

```
CREATE TABLE sale_return_items (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
return_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED NOT NULL,
quantity INT NOT NULL,     unit_price DECIMAL(10, 2) NOT NULL,     total_price
DECIMAL(10, 2) NOT NULL,     FOREIGN KEY (return_id) REFERENCES sale_returns(id) ON
DELETE CASCADE,     FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE
CASCADE,     INDEX idx_return (return_id),     INDEX idx_product (product_id) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.10 Service Completion and Product Usage

service_records

```
CREATE TABLE service_records (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
appointment_id BIGINT UNSIGNED NOT NULL,     sale_id BIGINT UNSIGNED COMMENT
'Linked invoice/sale',     customer_id BIGINT UNSIGNED NOT NULL,     employee_id
BIGINT UNSIGNED,     service_date DATE NOT NULL,     start_time TIME,     end_time
TIME,     actual_duration INT COMMENT 'Actual duration in minutes',     notes TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY (appointment_id)
REFERENCES appointments(id) ON DELETE CASCADE,     FOREIGN KEY (sale_id) REFERENCES
sales(id) ON DELETE SET NULL,     FOREIGN KEY (customer_id) REFERENCES
customers(id) ON DELETE CASCADE,     FOREIGN KEY (employee_id) REFERENCES
employees(id) ON DELETE SET NULL,     INDEX idx_appointment (appointment_id),
INDEX idx_customer (customer_id),     INDEX idx_employee (employee_id),     INDEX
idx_service_date (service_date) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

service_product_usage

```
CREATE TABLE service_product_usage (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     service_record_id BIGINT UNSIGNED NOT NULL,     product_id BIGINT UNSIGNED
NOT NULL,     usage_count INT NOT NULL COMMENT 'Number of uses consumed',
warehouse_id BIGINT UNSIGNED NOT NULL,     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     FOREIGN KEY (service_record_id) REFERENCES
service_records(id) ON DELETE CASCADE,     FOREIGN KEY (product_id) REFERENCES
products(id) ON DELETE CASCADE,     FOREIGN KEY (warehouse_id) REFERENCES
warehouses(id),     INDEX idx_service_record (service_record_id),     INDEX
idx_product (product_id) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

7.2.11 Expenses

expense_categories

```
CREATE TABLE expense_categories (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     name VARCHAR(255) NOT NULL,     description TEXT,     status
ENUM('active', 'inactive') DEFAULT 'active',     created_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,     INDEX idx_name (name),     INDEX idx_status (status) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

expenses

```
CREATE TABLE expenses (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
expense_number VARCHAR(100) UNIQUE NOT NULL,     category_id BIGINT UNSIGNED NOT
NULL,     branch_id BIGINT UNSIGNED,     expense_date DATE NOT NULL,     amount
DECIMAL(10, 2) NOT NULL,     payment_method ENUM('cash', 'card', 'bank_transfer',
'other') NOT NULL,     description TEXT,     receipt_file VARCHAR(255),
is_recurring BOOLEAN DEFAULT FALSE,     created_by BIGINT UNSIGNED,     created_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     FOREIGN KEY (category_id)
REFERENCES expense_categories(id),     FOREIGN KEY (branch_id) REFERENCES
branches(id) ON DELETE SET NULL,     FOREIGN KEY (created_by) REFERENCES users(id)
ON DELETE SET NULL,     INDEX idx_expense_number (expense_number),     INDEX
idx_category (category_id),     INDEX idx_branch (branch_id),     INDEX
idx_expense_date (expense_date) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

7.2.12 Communications

sms_log

```
CREATE TABLE sms_log (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,     phone
VARCHAR(20) NOT NULL,     message TEXT NOT NULL,     message_type ENUM('otp',
'booking_confirmation', 'reminder', 'birthday', 'promotional', 'other') NOT NULL,
status ENUM('pending', 'sent', 'failed') DEFAULT 'pending',     gateway_response
TEXT,     related_type VARCHAR(100) COMMENT 'bookings, appointments, etc',
related_id BIGINT UNSIGNED,     sent_at TIMESTAMP NULL,     created_at TIMESTAMP
DEFAULT CURRENT_TIMESTAMP,     INDEX idx_phone (phone),     INDEX idx_message_type
(message_type),     INDEX idx_status (status),     INDEX idx_related (related_type,
related_id),     INDEX idx_sent_at (sent_at) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

whatsapp_log

```
CREATE TABLE whatsapp_log (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
phone VARCHAR(20) NOT NULL,     message TEXT NOT NULL,     message_type
ENUM('booking_confirmation', 'reminder', 'birthday', 'promotional', 'bill',
'other') NOT NULL,     media_url VARCHAR(255) COMMENT 'Attachment URL',
template_name VARCHAR(255),     status ENUM('pending', 'sent', 'delivered', 'read',
'failed') DEFAULT 'pending',     gateway_response TEXT,     related_type
VARCHAR(100),     related_id BIGINT UNSIGNED,     sent_at TIMESTAMP NULL,
delivered_at TIMESTAMP NULL,     read_at TIMESTAMP NULL,     created_at TIMESTAMP
DEFAULT CURRENT_TIMESTAMP,     INDEX idx_phone (phone),     INDEX idx_message_type
(message_type),     INDEX idx_status (status),     INDEX idx_related (related_type,
related_id),     INDEX idx_sent_at (sent_at) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

email_log

```
CREATE TABLE email_log (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
email VARCHAR(255) NOT NULL,     subject VARCHAR(255) NOT NULL,     body TEXT NOT
NULL,     email_type ENUM('booking_confirmation', 'invoice', 'promotional',
'password_reset', 'other') NOT NULL,     attachments TEXT COMMENT 'JSON array of
attachment paths',     status ENUM('pending', 'sent', 'failed') DEFAULT 'pending',
error_message TEXT,     related_type VARCHAR(100),     related_id BIGINT UNSIGNED,
sent_at TIMESTAMP NULL,     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
INDEX idx_email (email),     INDEX idx_email_type (email_type),     INDEX
idx_status (status),     INDEX idx_sent_at (sent_at) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.2.13 System Settings

settings

```
CREATE TABLE settings (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
setting_key VARCHAR(255) UNIQUE NOT NULL,     setting_value TEXT,     setting_group
VARCHAR(100) COMMENT 'company, tax, email, sms, booking, etc',     description
TEXT,     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at TIMESTAMP
DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     INDEX idx_setting_key
```

```
(setting_key),     INDEX idx_setting_group (setting_group) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

notification_templates

```
CREATE TABLE notification_templates (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY
KEY,     name VARCHAR(255) NOT NULL,     channel ENUM('sms', 'whatsapp', 'email')
NOT NULL,     template_key VARCHAR(255) UNIQUE NOT NULL,     subject VARCHAR(255)
COMMENT 'For email only',     body TEXT NOT NULL,     variables TEXT COMMENT 'JSON
array of available variables',     status ENUM('active', 'inactive') DEFAULT
'active',     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     updated_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,     INDEX
idx_template_key (template_key),     INDEX idx_channel (channel),     INDEX
idx_status (status) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

7.2.14 Audit and Activity Log

activity_log

```
CREATE TABLE activity_log (     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
user_id BIGINT UNSIGNED,     action VARCHAR(255) NOT NULL,     model_type
VARCHAR(255) COMMENT 'Model class name',     model_id BIGINT UNSIGNED,
old_values TEXT COMMENT 'JSON of old values',     new_values TEXT COMMENT 'JSON of
new values',     ip_address VARCHAR(45),     user_agent TEXT,     created_at
TIMESTAMP DEFAULT CURRENT_TIMESTAMP,     FOREIGN KEY (user_id) REFERENCES users(id)
ON DELETE SET NULL,     INDEX idx_user (user_id),     INDEX idx_action (action),
INDEX idx_model (model_type, model_id),     INDEX idx_created_at (created_at) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

7.3 Database Indexes Strategy

Indexing Guidelines:

29. **Primary Keys**: All tables have auto-incrementing primary keys
30. **Foreign Keys**: All foreign keys are indexed
31. **Unique Constraints**: Applied to unique identifiers (SKU, phone, email, etc.)
32. **Search Fields**: Fields frequently used in WHERE clauses are indexed
33. **Sort Fields**: Fields used in ORDER BY are indexed
34. **Composite Indexes**: Created for frequently combined search conditions
35. **Date Fields**: Date/timestamp fields used in range queries are indexed

7.4 Database Relationships Summary

- **Users** → **Branches** (many-to-one)
- **Branches** → **Warehouses** (one-to-many)
- **Products** → **Categories** (many-to-one)
- **Products** → **Inventory** (one-to-many through warehouses)
- **Suppliers** → **Purchases** (one-to-many)
- **Purchases** → **Products** (many-to-many through purchase_items)
- **Customers** → **Appointments** (one-to-many)
- **Employees** → **Appointments** (one-to-many)
- **Appointments** → **Services** (many-to-many through appointment_services)
- **Appointments** → **Sales** (one-to-one for service billing)
- **Sales** → **Products** (many-to-many through sale_items)
- **Services** → **Products** (many-to-many through service_products for usage tracking)

8. API Specifications

8.1 API Architecture

API Type: RESTful API

Base URL: `https://api.yoursalon.com/api/v1`

Authentication: JWT Bearer Token

Response Format: JSON

API Versioning: URL-based (/v1, /v2)

8.2 Authentication Endpoints

8.2.1 Login

```
POST /auth/login
```

Request Body:

```
{   "email": "user@example.com",   "password": "password123" }
```

Response (200 OK):

```
{   "success": true,   "message": "Login successful",   "data": {     "user": {
"id": 1,       "name": "John Doe",       "email": "user@example.com",       "role":
"admin",       "branch_id": 1     },     "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",     "expires_at": "2025-11-
01T00:00:00.000000Z"   } }
```

Error Response (401 Unauthorized):

```
{   "success": false,   "message": "Invalid credentials" }
```

8.2.2 Logout

```
POST /auth/logout Authorization: Bearer {token}
```

Response (200 OK):

```
{   "success": true,   "message": "Logged out successfully" }
```

8.2.3 Refresh Token

```
POST /auth/refresh Authorization: Bearer {token}
```

Response (200 OK):

```
{   "success": true,   "data": {     "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",     "expires_at": "2025-11-
01T00:00:00.000000Z"   } }
```

8.3 Product Management Endpoints

8.3.1 List Products

```
GET /products Authorization: Bearer {token}
```

Query Parameters:

- `page` (integer): Page number (default: 1)
- `per_page` (integer): Items per page (default: 20)
- `search` (string): Search by name, SKU, barcode
- `category_id` (integer): Filter by category
- `product_type` (string): retail|consumable|both
- `status` (string): active|inactive
- `sort_by` (string): Field to sort by

- `sort_order` (string): asc|desc

Response (200 OK):

```
{   "success": true,   "data": {     "products": [       {         "id": 1,
"sku": "PRD-001",         "name": "Premium Hair Shampoo",         "category": {
"id": 1,           "name": "Hair Care"         },         "selling_price": 25.99,
"product_type": "both",         "usage_count": 10,         "status": "active",
"image": "/storage/products/shampoo.jpg"       }     ],     "meta":
{       "current_page": 1,       "per_page": 20,       "total": 150,
"last_page": 8     }   } }
```

8.3.2 Create Product

```
POST /products Authorization: Bearer {token} Content-Type: multipart/form-data
```

Request Body:

```
{   "name": "Premium Hair Shampoo",   "sku": "PRD-001",   "category_id": 1,
"brand": "ProCare",   "description": "Professional hair shampoo for all hair
types",   "barcode": "1234567890123",   "unit": "bottle",   "cost_price": 15.00,
"selling_price": 25.99,   "product_type": "both",   "usage_count": 10,
"reorder_level": 5,   "tax_rate": 10.00,   "image": "(file upload)" }
```

Response (201 Created):

```
{   "success": true,   "message": "Product created successfully",   "data":
{     "product": {       "id": 1,       "sku": "PRD-001",       "name": "Premium
Hair Shampoo",       ...     }   } }
```

8.3.3 Update Product

```
PUT /products/{id} Authorization: Bearer {token}
```

8.3.4 Delete Product

```
DELETE /products/{id} Authorization: Bearer {token}
```

8.3.5 Get Product Details

```
GET /products/{id} Authorization: Bearer {token}
```

8.4 Inventory Endpoints

8.4.1 Get Stock Levels

```
GET /inventory/stock Authorization: Bearer {token}
```

Query Parameters:

- `warehouse_id` (integer)
- `product_id` (integer)
- `low_stock` (boolean): Show only low stock items

Response (200 OK):

```
{   "success": true,   "data": {     "stock": [       {         "product_id": 1,
"product_name": "Premium Hair Shampoo",         "warehouse_id": 1,
"warehouse_name": "Main Warehouse",         "retail_quantity": 50,
"consumable_quantity": 20,         "consumable_usage_count": 180,
"reorder_level": 5,         "is_low_stock": false       }     ]   } }
```

8.4.2 Stock Adjustment

```
POST /inventory/adjust Authorization: Bearer {token}
```

Request Body:

```
{   "product_id": 1,   "warehouse_id": 1,   "adjustment_type": "retail",
"quantity": -5,   "reason": "damaged",   "notes": "Bottles were damaged during
shipping" }
```

8.4.3 Stock Transfer

```
POST /inventory/transfer Authorization: Bearer {token}
```

Request Body:

```
{   "source_warehouse_id": 1,   "destination_warehouse_id": 2,   "transfer_date":
"2025-10-31",   "items": [     {       "product_id": 1,       "quantity": 10,
"stock_type": "retail"     }   ],   "notes": "Monthly stock rebalancing" }
```

8.5 Appointment Endpoints

8.5.1 List Appointments

```
GET /appointments Authorization: Bearer {token}
```

Query Parameters:

- `date` (date): Filter by date
- `employee_id` (integer)
- `customer_id` (integer)
- `status` (string)
- `branch_id` (integer)

Response (200 OK):

```
{   "success": true,   "data": {     "appointments": [       {         "id": 1,
"appointment_number": "APT-2025-001",         "customer": {           "id": 5,
"name": "Jane Smith",           "phone": "+1234567890"         },
"employee": {           "id": 2,           "name": "Sarah Johnson"         },
"services": [           {             "id": 1,             "name": "Haircut",
"duration": 30,             "price": 35.00           }         ],
"appointment_date": "2025-11-01",         "appointment_time": "10:00:00",
"status": "scheduled",         "booking_source": "online"       }     ]   } }
```

8.5.2 Create Appointment

```
POST /appointments Authorization: Bearer {token}
```

Request Body:

```
{   "customer_id": 5,   "branch_id": 1,   "appointment_date": "2025-11-01",
"appointment_time": "10:00:00",   "service_ids": [1, 2],   "employee_id": null,
"notes": "Customer prefers quiet environment" }
```

Response (201 Created):

```
{   "success": true,   "message": "Appointment created successfully",   "data": {
"appointment": {       "id": 1,       "appointment_number": "APT-2025-001",
"assigned_employee": {         "id": 2,         "name": "Sarah Johnson"       },
...     }   } }
```

8.5.3 Update Appointment

```
PUT /appointments/{id} Authorization: Bearer {token}
```

8.5.4 Cancel Appointment

```
POST /appointments/{id}/cancel Authorization: Bearer {token}
```

Request Body:

```
{   "cancellation_reason": "Customer requested reschedule" }
```

8.5.5 Mark Service Complete

```
POST /appointments/{id}/complete Authorization: Bearer {token}
```

Request Body:

```
{   "actual_services": [     {       "service_id": 1,       "price":
35.00     }   ],   "products_used": [     {       "product_id": 1,
```

```
"usage_count": 2     }   ],   "start_time": "10:00:00",   "end_time": "10:35:00",
"notes": "Service completed successfully" }
```

8.5.6 Get Available Time Slots

```
GET /appointments/available-slots Authorization: Bearer {token}
```

Query Parameters:

- `branch_id` (integer, required)
- `date` (date, required)
- `service_id` (integer, required)

Response (200 OK):

```
{   "success": true,   "data": {     "available_slots": [       {         "time":
"09:00:00",         "available_employees": 3       },       {         "time":
"09:30:00",         "available_employees": 2       },       {         "time":
"10:00:00",         "available_employees": 3       }     ]   } }
```

8.6 Customer Endpoints

8.6.1 List Customers

```
GET /customers Authorization: Bearer {token}
```

8.6.2 Create Customer

```
POST /customers Authorization: Bearer {token}
```

Request Body:

```
{   "name": "Jane Smith",   "phone": "+1234567890",   "email": "jane@example.com",
"date_of_birth": "1990-05-15",   "gender": "female",   "address": "123 Main St",
"city": "Springfield",   "preferred_employee_id": 2,   "communication_preference":
"sms" }
```

8.6.3 Get Customer Details

```
GET /customers/{id} Authorization: Bearer {token}
```

Response (200 OK):

```
{   "success": true,   "data": {     "customer": {       "id": 5,       "name":
"Jane Smith",       "phone": "+1234567890",       "email": "jane@example.com",
"total_visits": 12,       "total_spending": 450.00,       "last_visit_date": "2025-
10-25",       "recent_services": [...],       "recent_purchases": [...]     }   } }
```

8.6.4 Customer Service History

```
GET /customers/{id}/service-history Authorization: Bearer {token}
```

8.6.5 Customer Purchase History

```
GET /customers/{id}/purchase-history Authorization: Bearer {token}
```

8.7 POS Endpoints

8.7.1 Create Sale

```
POST /pos/sales Authorization: Bearer {token}
```

Request Body:

```
{   "customer_id": 5,   "branch_id": 1,   "warehouse_id": 1,   "sale_type":
"retail",   "items": [     {       "product_id": 10,       "quantity": 2,
"unit_price": 25.99,       "discount_amount": 2.00     }   ],   "tax_amount": 4.80,
"discount_amount": 2.00,   "shipping_charges": 0.00,   "payment_method": "cash",
"notes": "Walk-in customer" }
```

Response (201 Created):

```
{   "success": true,   "message": "Sale completed successfully",   "data":
{     "sale": {       "id": 100,       "sale_number": "SALE-2025-100",
"total_amount": 52.78,       "receipt_url": "/receipts/SALE-2025-100.pdf"     }   }
}
```

8.7.2 Hold Bill

```
POST /pos/hold-bill Authorization: Bearer {token}
```

Request Body:

```
{   "items": [...],   "customer_id": 5,   "subtotal": 50.00,   "notes": "Customer
stepped out" }
```

8.7.3 Get Held Bills

```
GET /pos/held-bills Authorization: Bearer {token}
```

8.7.4 Resume Held Bill

```
GET /pos/held-bills/{id} Authorization: Bearer {token}
```

8.7.5 Process Return

```
POST /pos/returns Authorization: Bearer {token}
```

Request Body:

```
{   "sale_id": 100,   "return_date": "2025-11-01",   "items":
[     {       "product_id": 10,       "quantity": 1,       "unit_price":
25.99     }   ],   "reason": "Defective product",   "refund_method": "cash" }
```

8.8 Online Booking Endpoints (Public)

8.8.1 Get Available Services

```
GET /public/booking/services
```

Query Parameters:

- `branch_id` (integer, required)

Response (200 OK):

```
{   "success": true,   "data": {     "services": [       {         "id": 1,
"name": "Haircut",         "description": "Professional haircut service",
"duration": 30,         "price": 35.00,         "category": "Hair Services"       }
]   } }
```

8.8.2 Check Available Slots

```
GET /public/booking/available-slots
```

Query Parameters:

- `branch_id` (integer, required)
- `service_id` (integer, required)
- `date` (date, required)

8.8.3 Request OTP

```
POST /public/booking/request-otp
```

Request Body:

```
{   "phone": "+1234567890" }
```

Response (200 OK):

```
{   "success": true,   "message": "OTP sent successfully",   "data":
{     "otp_expires_at": "2025-10-31T10:05:00Z"   } }
```

8.8.4 Verify OTP and Create Booking

```
POST /public/booking/verify-and-book
```

Request Body:

```
{   "customer_name": "Jane Smith",   "customer_phone": "+1234567890",
"customer_email": "jane@example.com",   "otp_code": "123456",   "branch_id": 1,
"service_id": 1,   "appointment_date": "2025-11-01",   "appointment_time":
"10:00:00" }
```

Response (201 Created):

```
{   "success": true,   "message": "Booking confirmed successfully",   "data": {
"booking": {       "booking_number": "BOOK-2025-001",       "appointment_date":
"2025-11-01",       "appointment_time": "10:00:00",       "service": "Haircut",
"confirmation_sent": true     }   } }
```

8.8.5 Get Booking Details

```
GET /public/booking/{booking_number}
```

Query Parameters:

- `phone` (string, required): For verification

8.8.6 Reschedule Booking

```
POST /public/booking/{booking_number}/reschedule
```

Request Body:

```
{   "phone": "+1234567890",   "new_date": "2025-11-02",   "new_time": "14:00:00" }
```

8.8.7 Cancel Booking

```
POST /public/booking/{booking_number}/cancel
```

Request Body:

```
{   "phone": "+1234567890",   "reason": "Unable to make it" }
```

8.9 Report Endpoints

8.9.1 Sales Report

```
GET /reports/sales Authorization: Bearer {token}
```

Query Parameters:

- `start_date` (date)
- `end_date` (date)
- `branch_id` (integer)
- `customer_id` (integer)
- `export` (string): pdf|excel

8.9.2 Inventory Report

```
GET /reports/inventory Authorization: Bearer {token}
```

8.9.3 Commission Report

```
GET /reports/commission Authorization: Bearer {token}
```

Query Parameters:

- `employee_id` (integer)
- `start_date` (date)
- `end_date` (date)
- `export` (string)

8.9.4 Consumption Report

```
GET /reports/consumption Authorization: Bearer {token}
```

8.9.5 Dashboard Stats

```
GET /reports/dashboard Authorization: Bearer {token}
```

Response (200 OK):

```
{   "success": true,   "data": {     "today_revenue": 1250.00,
"today_appointments": 15,     "pending_appointments": 5,     "low_stock_items": 8,
"top_products": [...],     "top_employees": [...],     "recent_transactions": [...]
} }
```

8.10 Settings Endpoints

8.10.1 Get All Settings

```
GET /settings Authorization: Bearer {token}
```

8.10.2 Update Settings

```
PUT /settings Authorization: Bearer {token}
```

Request Body:

```
{   "settings": [     {       "key": "company_name",       "value": "Elite Salon &
Spa"     },     {       "key": "tax_rate",       "value": "10.00"     }   ] }
```

8.11 Standard API Response Format

Success Response:

```
{   "success": true,   "message": "Operation completed successfully",   "data": {
...   },   "meta": {     ...   } }
```

Error Response:

```
{   "success": false,   "message": "Error message",   "errors": {     "field_name":
[       "Validation error message"     ]   } }
```

8.12 HTTP Status Codes

- **200 OK**: Request succeeded
- **201 Created**: Resource created successfully
- **400 Bad Request**: Invalid request data
- **401 Unauthorized**: Authentication required or failed
- **403 Forbidden**: Insufficient permissions
- **404 Not Found**: Resource not found
- **422 Unprocessable Entity**: Validation errors
- **429 Too Many Requests**: Rate limit exceeded
- **500 Internal Server Error**: Server error

8.13 Rate Limiting

- **Authenticated Users**: 60 requests per minute
- **Public Endpoints**: 30 requests per minute per IP
- Headers returned:
- `X-RateLimit-Limit`: Total allowed requests
- `X-RateLimit-Remaining`: Remaining requests
- `X-RateLimit-Reset`: Time when limit resets (Unix timestamp)

9. User Interface Requirements

9.1 General UI/UX Principles

36. **Consistency**: Uniform design language across all pages
37. **Simplicity**: Clean, uncluttered interfaces
38. **Responsiveness**: Adapt seamlessly to different screen sizes
39. **Accessibility**: WCAG 2.1 Level AA compliance
40. **Feedback**: Clear visual feedback for user actions
41. **Performance**: Fast loading and smooth interactions

9.2 Color Scheme

Primary Colors:

- Primary: #1976D2 (Blue)
- Secondary: #FF6B35 (Orange)
- Success: #4CAF50 (Green)
- Warning: #FFC107 (Amber)
- Error: #F44336 (Red)
- Info: #2196F3 (Light Blue)

Neutral Colors:

- Background: #F5F5F5
- Surface: #FFFFFF
- Text Primary: #212121
- Text Secondary: #757575
- Divider: #E0E0E0

9.3 Typography

- **Font Family**: Roboto, -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif
- **Font Sizes**:
- H1: 32px
- H2: 24px
- H3: 20px
- H4: 18px
- Body: 16px
- Caption: 14px
- Small: 12px

9.4 Admin/Operator Dashboard

9.4.1 Dashboard Home

Components:

- Top navigation bar with logo, search, notifications, user menu
- Left sidebar with module navigation
- Main content area with:
- Key metrics cards (revenue, appointments, customers)
- Quick actions (New Sale, New Appointment)
- Charts (sales trend, popular services)
- Recent transactions table
- Upcoming appointments list
- Low stock alerts

9.4.2 POS Interface

Layout:

- Left panel: Product search and category filters
- Center panel: Selected items cart
- Right panel: Calculator, customer selection, payment options
- Bottom: Action buttons (Hold, Clear, Payment)

Features:

- Barcode scanner input
- Product image thumbnails
- Real-time total calculation
- Multiple held bills management
- Receipt preview before printing

9.4.3 Appointment Calendar

Views:

- Day view: Hourly time slots with appointments
- Week view: 7-day grid
- Month view: Calendar overview

Features:

- Color-coded by service type
- Drag-and-drop rescheduling
- Click appointment for details/edit
- Filter by staff member
- Create appointment from empty slot
- Staff availability indicators

9.4.4 Inventory Management

Product List:

- Data table with search and filters
- Columns: Image, SKU, Name, Category, Stock, Price, Actions
- Bulk actions (export, delete)
- Quick edit inline

Product Form:

- Tabbed interface:
- Basic Info
- Pricing
- Inventory
- Images
- Barcode generation tool
- Real-time validation

Stock Management:

- Stock levels by warehouse
- Transfer interface
- Adjustment form
- Movement history

9.4.5 Customer Management

Customer List:

- Search by name, phone, email
- Filters (status, visit frequency)

- Customer cards with avatar
- Quick actions (view, edit, book)

Customer Profile:

- Personal information
- Service history timeline
- Purchase history
- Upcoming appointments
- Communication preferences
- Quick booking button

9.4.6 Reports Interface

Report Selection:

- Card-based report categories
- Quick filters (date range, branch)
- Export options (PDF, Excel)

Report View:

- Charts and visualizations
- Data tables with sorting
- Summary statistics
- Drill-down capability
- Print-friendly layout

9.5 Customer Booking Portal

9.5.1 Home Page

Elements:

- Salon logo and name
- Hero section with call-to-action
- Service categories overview
- Booking button (prominent)
- Contact information
- Operating hours

9.5.2 Booking Flow

Step 1: Service Selection

- Grid/list of services
- Service name, description, duration, price
- "Select" button

Step 2: Date & Time

- Calendar date picker
- Available time slots (disabled unavailable)
- Selected service summary

Step 3: Customer Details

- Name input
- Phone input
- Email input (optional)
- Special notes textarea

Step 4: OTP Verification

- Phone number display
- OTP input field (6 digits)

- Resend OTP button
- Countdown timer

Step 5: Confirmation

- Booking summary
- Confirmation number (large, prominent)
- Add to calendar button
- Booking management links

9.5.3 Booking Management

View Booking:

- Booking details display
- Phone verification for access
- Reschedule button
- Cancel button

Reschedule:

- New date/time selection
- Confirmation step

Cancel:

- Cancellation reason selection
- Confirmation dialog

9.6 Responsive Design

Mobile (< 768px)

- Collapsed sidebar (hamburger menu)
- Stacked layouts
- Touch-optimized buttons (min 44x44px)
- Swipe gestures support
- Bottom navigation for key actions

Tablet (768px - 1024px)

- Adaptive grid layouts
- Collapsible sidebar
- Optimized spacing

Desktop (> 1024px)

- Full sidebar navigation
- Multi-column layouts
- Hover interactions
- Keyboard shortcuts

9.7 UI Components Library

Common Components:

- Buttons (primary, secondary, outlined, text)
- Input fields (text, number, date, select)
- Checkboxes and radio buttons
- Switches and toggles
- Data tables with pagination
- Modals and dialogs
- Toast notifications
- Loading spinners

- Progress bars
- Breadcrumbs
- Tabs
- Accordions
- Badges and chips
- Cards
- Alerts
- Tooltips

10. Security Requirements

10.1 Authentication Security

10.1.1 Password Security

- Minimum 8 characters
- Must include: uppercase, lowercase, number, special character
- Bcrypt hashing with cost factor 12
- Password history (prevent reuse of last 5 passwords)
- Password expiry: 90 days for admins (optional)
- Account lockout after 5 failed attempts (15-minute lockout)

10.1.2 JWT Token Security

- HS256 signing algorithm
- Token expiration: 24 hours
- Refresh token expiration: 30 days
- Token stored in httpOnly cookies (web) or secure storage (mobile)
- Token blacklisting on logout
- Single sign-on prevention (optional)

10.1.3 Session Management

- Secure session handling
- Session timeout: 24 hours of inactivity
- Concurrent session limits
- Session invalidation on password change
- IP address validation (optional)

10.2 Authorization

10.2.1 Role-Based Access Control

- Middleware validation on all protected routes
- Permission checking at API and UI levels
- Resource-level permissions
- Deny by default approach
- Admin override capability (logged)

10.2.2 API Authorization

- Bearer token validation
- Role and permission verification
- Rate limiting per user
- API key for public endpoints

10.3 Data Security

10.3.1 Data Encryption

- TLS 1.3 for data in transit
- AES-256 encryption for sensitive data at rest

- Encrypted database backups
- Secure key management

10.3.2 Sensitive Data Handling

- No storage of full credit card numbers
- Masked display of phone numbers/emails
- Encrypted personal information
- PCI DSS compliance for payment data

10.3.3 Data Validation

- Server-side validation for all inputs
- Parameterized SQL queries (no raw SQL)
- Input sanitization
- Output encoding
- File upload validation (type, size, content)

10.4 Application Security

10.4.1 XSS Prevention

- React automatic escaping
- Content Security Policy headers
- Sanitize user-generated content
- HTML input stripping

10.4.2 CSRF Protection

- CSRF tokens for state-changing operations
- SameSite cookie attribute
- Referer header validation

10.4.3 SQL Injection Prevention

- Eloquent ORM usage (no raw queries)
- Prepared statements
- Input validation and sanitization

10.4.4 File Upload Security

- File type validation (whitelist)
- File size limits
- Malware scanning
- Separate storage from web root
- Randomized filenames
- No execution permissions on upload directory

10.4.5 API Security

- Rate limiting (60 req/min per user)
- Request throttling
- CORS configuration
- API versioning
- Input validation
- Output filtering

10.5 Infrastructure Security

10.5.1 Server Security

- Firewall configuration
- SSH key-based authentication
- Disable root login
- Automated security updates
- Intrusion detection system

- DDoS protection

10.5.2 Database Security

- Separate database server
- Strong database passwords
- Limited database user privileges
- Encrypted connections
- Regular security audits
- Backup encryption

10.5.3 SSL/TLS

- Valid SSL certificate (Let's Encrypt or commercial)
- HTTPS enforcement (redirect HTTP)
- HSTS header
- TLS 1.3 minimum
- Strong cipher suites

10.6 Audit and Monitoring

10.6.1 Activity Logging

- All authentication attempts
- Permission changes
- Data modifications (CRUD operations)
- Failed access attempts
- Admin actions
- API requests

10.6.2 Security Monitoring

- Failed login tracking
- Unusual activity detection
- Rate limit violations
- Error rate monitoring
- Automated alerts for security events

10.6.3 Audit Trail

- Immutable logs
- Timestamp and user tracking
- IP address logging
- User agent logging
- Secure log storage
- Log retention: 1 year minimum

10.7 Privacy and Compliance

10.7.1 Data Privacy

- Privacy policy display
- User consent for data collection
- Right to access personal data
- Right to delete personal data (with exceptions)
- Data minimization
- Purpose limitation

10.7.2 Compliance

- GDPR compliance (if applicable)
- Local data protection laws
- PCI DSS for payment processing
- Regular compliance audits

10.8 Backup and Recovery

10.8.1 Backup Strategy

- Automated daily backups
- Encrypted backup storage
- Off-site backup location
- Backup verification and testing
- Backup retention: 30 days

10.8.2 Disaster Recovery

- Documented recovery procedures
- Recovery Time Objective (RTO): 4 hours
- Recovery Point Objective (RPO): 24 hours
- Regular disaster recovery drills
- Redundant infrastructure (optional)

10.9 Third-Party Security

10.9.1 Dependency Management

- Regular dependency updates
- Vulnerability scanning
- Trusted package sources
- License compliance

10.9.2 Third-Party Services

- Secure API integrations
- API key rotation
- Service-level agreements
- Data processing agreements

10.10 Security Testing

10.10.1 Testing Requirements

- Penetration testing (annual)
- Vulnerability scanning (monthly)
- Code security review
- Dependency vulnerability checks
- Security testing in CI/CD pipeline

10.10.2 Security Training

- Developer security awareness
- Secure coding practices
- Security incident response training

11. Integration Requirements

11.1 SMS Gateway Integration

11.1.1 Provider Options

- Primary: Twilio
- Alternatives: Vonage (Nexmo), local SMS gateways

11.1.2 Integration Specifications

- RESTful API integration
- API authentication (API key/secret)
- Message queuing for bulk sends

- Delivery status webhooks
- Character encoding support (UTF-8)
- Long message handling (concatenation)

11.1.3 Message Types

- OTP messages
- Booking confirmations
- Appointment reminders
- Birthday wishes
- Promotional messages
- Service notifications

11.1.4 Error Handling

- Retry logic (max 3 attempts)
- Failed message logging
- Balance/credit monitoring
- Fallback provider option

11.2 WhatsApp Business API Integration

11.2.1 Provider Options

- Primary: Twilio WhatsApp API
- Alternative: Official WhatsApp Business API

11.2.2 Integration Specifications

- Template message approval workflow
- Media message support (images, PDFs)
- Interactive button messages
- Delivery receipt handling
- 24-hour session window compliance

11.2.3 Message Templates

- Booking confirmation with details
- Appointment reminder
- Invoice/receipt sharing
- Birthday wishes with offer
- Promotional messages
- Service completion thank you

11.2.4 Features

- Rich media attachments
- Call-to-action buttons
- Quick reply buttons
- Message status tracking (sent, delivered, read)

11.3 Payment Gateway Integration

11.3.1 Provider Options

- Primary: Stripe
- Alternatives: PayPal, local payment gateways

11.3.2 Integration Method

- Stripe Checkout (hosted)
- Stripe Payment Intents API
- PCI DSS Level 1 compliance
- Tokenization for card storage

11.3.3 Supported Payment Methods

- Credit/Debit cards
- Digital wallets (Apple Pay, Google Pay)
- Bank transfers (if supported)
- Local payment methods

11.3.4 Payment Flow

- Cart/invoice creation
- Payment gateway redirect/modal
- Payment processing
- Webhook for payment confirmation
- Receipt generation
- Inventory update

11.3.5 Security

- No card data storage in application
- Tokenized payment information
- 3D Secure authentication
- Fraud detection
- PCI compliance

11.3.6 Error Handling

- Payment failure notifications
- Retry mechanism
- Refund processing
- Dispute management

11.4 Email Service Integration

11.4.1 Provider Options

- Primary: SendGrid
- Alternatives: Mailgun, Amazon SES

11.4.2 Email Types

- Transactional emails (confirmations, receipts)
- Password reset
- Invoices with PDF attachment
- Promotional campaigns
- Monthly statements
- System notifications

11.4.3 Features

- HTML email templates
- Email tracking (opens, clicks)
- Unsubscribe management
- Bounce handling
- Spam compliance
- Attachment support

11.4.4 Integration Specifications

- SMTP or API integration
- Email queuing for bulk sends
- Webhook for email events
- Template management
- A/B testing support (optional)

11.5 Cloud Storage Integration (Optional)

11.5.1 Provider Options

- Amazon S3
- DigitalOcean Spaces
- Local file storage (default)

11.5.2 Use Cases

- Product images
- Receipt PDFs
- Backup files
- Expense receipts
- Employee photos

11.5.3 Features

- Secure file upload
- Public/private access control
- CDN integration
- File versioning
- Automatic backup

11.6 Calendar Integration (Optional)

11.6.1 Providers

- Google Calendar
- Outlook Calendar
- iCal format support

11.6.2 Features

- Add appointment to customer's calendar
- Two-way sync for staff calendars
- Reminder notifications
- Calendar invite emails

11.7 Accounting Software Integration (Future)

11.7.1 Potential Integrations

- QuickBooks
- Xero
- FreshBooks

11.7.2 Data Sync

- Sales transactions
- Expense records
- Customer data
- Tax information

12. Testing Requirements

12.1 Unit Testing

12.1.1 Backend Unit Tests (PHPUnit)

- Test all service classes
- Test all repository methods
- Test model relationships
- Test validation rules
- Test helper functions
- Coverage target: ≥ 70%

12.1.2 Frontend Unit Tests (Jest)

- Test utility functions
- Test Redux reducers and actions
- Test custom hooks
- Test API service functions
- Coverage target: ≥ 60%

12.1.3 Component Tests (React Testing Library)

- Test all reusable components
- Test user interactions
- Test prop handling
- Test conditional rendering

12.2 Integration Testing

12.2.1 API Integration Tests

- Test all API endpoints
- Test authentication flows
- Test authorization checks
- Test error handling
- Test data validation
- Test database transactions

12.2.2 Frontend-Backend Integration

- Test API communication
- Test data flow
- Test error handling
- Test loading states

12.3 End-to-End Testing

12.3.1 E2E Test Scenarios (Cypress)

- User authentication flow
- Complete POS transaction
- Appointment booking flow (online portal)
- Appointment creation and management
- Product CRUD operations
- Customer CRUD operations
- Report generation
- Settings update

12.3.2 Critical User Journeys

- Walk-in customer purchase
- Online booking to service completion
- Stock transfer between warehouses
- Invoice generation and payment
- Return processing

12.4 Performance Testing

12.4.1 Load Testing

- Concurrent user handling (50 users)
- Database query performance
- API response times
- Page load times
- Report generation time

12.4.2 Stress Testing

- Peak load handling
- Resource usage monitoring
- Failure recovery
- Database connection limits

12.5 Security Testing

12.5.1 Penetration Testing

- SQL injection attempts
- XSS vulnerability checks
- CSRF protection verification
- Authentication bypass attempts
- Authorization escalation attempts
- API security testing

12.5.2 Vulnerability Scanning

- Dependency vulnerability check
- OWASP Top 10 verification
- Automated security scanning

12.6 Usability Testing

12.6.1 User Acceptance Testing (UAT)

- Admin user testing
- Operator user testing
- Customer portal testing
- Feedback collection
- Issue documentation

12.6.2 Accessibility Testing

- WCAG 2.1 compliance check
- Screen reader testing
- Keyboard navigation testing
- Color contrast verification

12.7 Compatibility Testing

12.7.1 Browser Testing

- Chrome (latest 2 versions)
- Firefox (latest 2 versions)
- Safari (latest 2 versions)
- Edge (latest 2 versions)

12.7.2 Device Testing

- Desktop (Windows, macOS)
- Tablet (iOS, Android)
- Mobile (iOS, Android)
- Various screen sizes

12.7.3 Database Testing

- MySQL 8.0
- MariaDB 10.5 (optional)

12.8 Regression Testing

12.8.1 Automated Regression Suite

- Critical functionality tests

- Integration tests
- API tests
- Run before each release

12.8.2 Manual Regression Testing

- New feature impact
- Bug fix verification
- UI/UX consistency

12.9 Test Documentation

12.9.1 Test Plans

- Test scope and objectives
- Test scenarios and cases
- Test data requirements
- Testing schedule

12.9.2 Test Reports

- Test execution results
- Bug reports
- Coverage reports
- Performance benchmarks

12.10 Continuous Testing

12.10.1 CI/CD Pipeline Tests

- Automated unit tests on commit
- Integration tests on PR
- E2E tests before deployment
- Security scans in pipeline

13. Deployment Requirements

13.1 Deployment Environment

13.1.1 Server Requirements

- **OS**: Ubuntu 22.04 LTS or higher
- **Web Server**: Nginx 1.18+ or Apache 2.4+
- **PHP**: 8.1 or higher
- **Database**: MySQL 8.0 or higher
- **Node.js**: 18.x or higher (for build process)
- **Redis**: 6.x or higher (for caching and queues)
- **Supervisor**: For queue worker management

13.1.2 Minimum Hardware

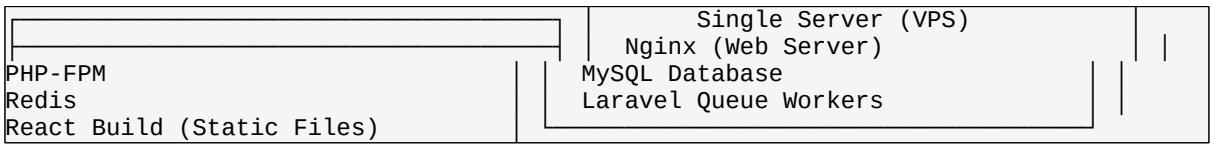
- **CPU**: 2 vCPUs
- **RAM**: 4 GB
- **Storage**: 50 GB SSD
- **Network**: 100 Mbps

13.1.3 Recommended Hardware (Production)

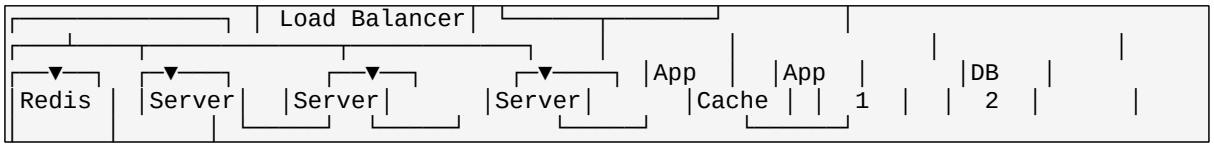
- **CPU**: 4 vCPUs
- **RAM**: 8 GB
- **Storage**: 100 GB SSD
- **Network**: 1 Gbps
- **Load Balancer**: For high availability

13.2 Deployment Architecture

13.2.1 Single Server Deployment



13.2.2 Production Deployment (Recommended)



13.3 Pre-Deployment Checklist

13.3.1 Code Preparation

- [ ] All tests passing
- [ ] Code review completed
- [ ] Security audit completed
- [ ] Performance testing done
- [ ] Documentation updated
- [ ] Version tagged in Git
- [ ] Change log updated

13.3.2 Environment Setup

- [ ] Production server provisioned
- [ ] SSL certificate installed
- [ ] Domain DNS configured
- [ ] Firewall rules configured
- [ ] Backup system configured
- [ ] Monitoring tools installed
- [ ] Environment variables set

13.3.3 Database Setup

- [ ] Database created
- [ ] Database user created with appropriate privileges
- [ ] Database connection tested
- [ ] Migrations ready
- [ ] Seeders prepared (if needed)
- [ ] Backup strategy implemented

13.3.4 Third-Party Services

- [ ] SMS gateway configured
- [ ] WhatsApp API configured
- [ ] Payment gateway configured
- [ ] Email service configured
- [ ] Cloud storage configured (if used)
- [ ] API keys secured

13.4 Deployment Process

13.4.1 Backend Deployment (Laravel)

42. **Clone Repository**

```
git clone <repository-url> cd salon-management-backend git checkout <version-tag>
```

43. **Install Dependencies**

```
composer install --optimize-autoloader --no-dev
```

44. **Environment Configuration**

```
cp .env.example .env php artisan key:generate # Edit .env with production values
```

45. **Database Migration**

```
php artisan migrate --force php artisan db:seed --class=ProductionSeeder
```

46. **Cache and Optimize**

```
php artisan config:cache php artisan route:cache php artisan view:cache php artisan
event:cache
```

47. **File Permissions**

```
chmod -R 775 storage bootstrap/cache chown -R www-data:www-data storage
bootstrap/cache
```

48. **Queue Workers Setup**

```
# Configure Supervisor sudo nano /etc/supervisor/conf.d/laravel-worker.conf sudo
supervisorctl reread sudo supervisorctl update sudo supervisorctl start laravel-
worker:*
```

13.4.2 Frontend Deployment (React)

49. **Clone Repository**

```
git clone <repository-url> cd salon-management-frontend git checkout <version-tag>
```

50. **Install Dependencies**

```
npm install
```

51. **Build for Production**

```
npm run build
```

52. **Deploy Build**

```
# Copy build files to web server cp -r build/* /var/www/html/salon/
```

13.4.3 Web Server Configuration

Nginx Configuration Example:

```
server {     listen 80;     listen [::]:80;     server_name yoursalon.com
www.yoursalon.com;     return 301 https://$server_name$request_uri; }  server {
listen 443 ssl http2;     listen [::]:443 ssl http2;     server_name yoursalon.com
www.yoursalon.com;      root /var/www/html/salon;     index index.html;
ssl_certificate /etc/letsencrypt/live/yoursalon.com/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/yoursalon.com/privkey.pem;      # React
frontend     location / {         try_files $uri $uri/ /index.html;     }      #
Laravel API     location /api {         proxy_pass http://localhost:8000;
proxy_http_version 1.1;         proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'upgrade';         proxy_set_header Host $host;
proxy_cache_bypass $http_upgrade;     }      # Static files caching     location ~*
\.(jpg|jpeg|png|gif|ico|css|js)$ {         expires 1y;         add_header Cache-
Control "public, immutable";     } }
```

13.5 Post-Deployment Verification

13.5.1 Smoke Tests

- [ ] Website accessible via HTTPS
- [ ] Login functionality working
- [ ] API endpoints responding
- [ ] Database connections working
- [ ] Queue workers running

- [ ] Scheduled tasks running
- [ ] File uploads working
- [ ] Email sending working
- [ ] SMS sending working (test mode)
- [ ] Payment gateway connected (test mode)

13.5.2 Monitoring Setup

- [ ] Application logs monitoring
- [ ] Error tracking (Sentry) configured
- [ ] Performance monitoring active
- [ ] Uptime monitoring configured
- [ ] SSL certificate expiry monitoring

13.6 Rollback Plan

13.6.1 Rollback Procedure

53. Stop queue workers
54. Revert to previous Git tag
55. Restore previous database backup (if needed)
56. Clear caches
57. Restart services
58. Verify functionality

13.6.2 Rollback Triggers

- Critical bugs in production
- Database migration failures
- Performance degradation
- Security vulnerabilities discovered
- Integration failures

13.7 Continuous Deployment

13.7.1 CI/CD Pipeline (GitHub Actions Example)

```
name: Deploy to Production  on:   push:     branches:       - main  jobs:   deploy:
runs-on: ubuntu-latest     steps:       - uses: actions/checkout@v3              -
name: Run Tests         run: |           composer install           php artisan
test              - name: Deploy to Server         uses: appleboy/ssh-action@master
with:           host: ${{ secrets.PRODUCTION_HOST }}           username: $
{{ secrets.PRODUCTION_USER }}           key: ${{ secrets.PRODUCTION_SSH_KEY }}
script: |             cd /var/www/salon-backend             git pull origin main
composer install --no-dev             php artisan migrate --force             php
artisan config:cache             php artisan route:cache             php artisan
view:cache             sudo supervisorctl restart laravel-worker:*
```

13.8 Maintenance Mode

13.8.1 Enable Maintenance Mode

```
php artisan down --message="System maintenance in progress" --retry=60
```

13.8.2 Disable Maintenance Mode

```
php artisan up
```

13.8.3 Allow IPs During Maintenance

```
php artisan down --allow=123.456.789.0 --allow=111.222.333.444
```

14. Maintenance and Support

14.1 Maintenance Plan

14.1.1 Regular Maintenance Schedule

Daily:

- Monitor system logs for errors
- Check queue worker status
- Review failed jobs
- Monitor disk space
- Check backup completion

Weekly:

- Review performance metrics
- Analyze user activity
- Check low stock alerts
- Review security logs
- Database optimization (ANALYZE tables)

Monthly:

- Security updates (OS, PHP, dependencies)
- Database backup verification
- Performance tuning
- Review and archive old logs
- Certificate expiry check
- Dependency updates

Quarterly:

- Major version updates (Laravel, React)
- Security audit
- Performance audit
- User feedback review
- Feature usage analysis
- Disaster recovery drill

Annually:

- Penetration testing
- Comprehensive security audit
- Infrastructure review
- Capacity planning
- Contract renewals (hosting, APIs)

14.2 Monitoring and Alerting

14.2.1 System Monitoring

- Server uptime
- CPU and memory usage
- Disk space
- Network traffic
- Database performance
- Queue length

14.2.2 Application Monitoring

- Response times
- Error rates
- API endpoint performance

- Failed jobs
- User sessions
- Transaction volumes

14.2.3 Alert Triggers

- Server down
- High error rate (> 5%)
- Low disk space (< 10%)
- Queue backlog (> 1000 jobs)
- Failed payment transactions
- SMS/Email sending failures
- Database connection failures

14.2.4 Monitoring Tools

- Laravel Telescope (development)
- Laravel Horizon (queue monitoring)
- Sentry (error tracking)
- New Relic / DataDog (APM)
- Google Analytics (user analytics)
- UptimeRobot (uptime monitoring)

14.3 Backup Strategy

14.3.1 Backup Types

Database Backups:

- Frequency: Daily
- Retention: 30 days
- Storage: Off-site encrypted storage
- Format: SQL dump

File Backups:

- Frequency: Daily
- Retention: 30 days
- Includes: Uploads, receipts, logs
- Format: Compressed archive

Application Code:

- Version control (Git)
- Tagged releases
- Deployment scripts backed up

14.3.2 Backup Automation

Database Backup Script:

```
#!/bin/bash DATE=$(date +%Y-%m-%d_%H-%M-%S) BACKUP_DIR="/backups/database"
DB_NAME="salon_db" DB_USER="backup_user" DB_PASS="secure_password"  # Create backup
mysqldump -u $DB_USER -p$DB_PASS $DB_NAME | gzip > $BACKUP_DIR/backup_$DATE.sql.gz
# Encrypt backup gpg --encrypt --recipient admin@yoursalon.com
$BACKUP_DIR/backup_$DATE.sql.gz  # Upload to cloud storage aws s3 cp
$BACKUP_DIR/backup_$DATE.sql.gz.gpg s3://salon-backups/database/  # Delete local
backups older than 7 days find $BACKUP_DIR -name "backup_*.sql.gz*" -mtime +7 -
delete
```

14.3.3 Backup Verification

- Monthly restore tests
- Backup integrity checks
- Restore time measurement
- Documentation of restore procedures

14.4 Update and Patch Management

14.4.1 Security Updates

- Apply critical security patches within 48 hours
- Test patches in staging environment
- Schedule maintenance window for production
- Notify users of scheduled downtime

14.4.2 Dependency Updates

- Review dependency updates weekly
- Update minor versions monthly
- Update major versions quarterly (after testing)
- Check for deprecated dependencies

14.4.3 Feature Updates

- Sprint-based development (2-week sprints)
- Feature releases monthly
- User communication before major changes
- Gradual rollout of new features

14.5 Performance Optimization

14.5.1 Database Optimization

- Query optimization
- Index management
- Table partitioning (if needed)
- Archiving old records
- Regular ANALYZE and OPTIMIZE

14.5.2 Application Optimization

- Cache optimization (Redis)
- Query result caching
- Route caching
- View caching
- Asset optimization (minification, compression)

14.5.3 Frontend Optimization

- Code splitting
- Lazy loading
- Image optimization
- CDN usage
- Browser caching

14.6 Support Tiers

14.6.1 Tier 1 Support (Level 1)

- User account issues
- Password resets
- General usage questions
- Basic troubleshooting
- Response time: 4 hours

14.6.2 Tier 2 Support (Level 2)

- Configuration issues
- Data import/export
- Report generation issues
- Integration problems

- Response time: 8 hours

14.6.3 Tier 3 Support (Level 3)

- Critical system failures
- Security incidents
- Database issues
- Performance problems
- Response time: 1 hour (critical), 4 hours (non-critical)

14.7 Issue Tracking and Resolution

14.7.1 Issue Categories

- **P1 - Critical**: System down, data loss, security breach
- **P2 - High**: Major feature broken, significant performance degradation
- **P3 - Medium**: Minor feature issues, non-critical bugs
- **P4 - Low**: Cosmetic issues, feature requests

14.7.2 Resolution SLAs

- P1 Critical: 2 hours
- P2 High: 8 hours
- P3 Medium: 48 hours
- P4 Low: 1 week

14.7.3 Issue Management

- Use GitHub Issues or Jira
- Detailed issue description
- Steps to reproduce
- Screenshots/logs
- Assigned developer
- Progress tracking
- User communication

14.8 Documentation Maintenance

14.8.1 User Documentation

- User manuals (Admin, Operator, Customer)
- Quick start guides
- Video tutorials
- FAQ section
- Troubleshooting guides

14.8.2 Technical Documentation

- API documentation (updated automatically)
- Database schema documentation
- Deployment guides
- Architecture diagrams
- Code comments

14.8.3 Documentation Updates

- Update with each feature release
- Review quarterly for accuracy
- User feedback incorporation
- Version-specific documentation

14.9 Training and Onboarding

14.9.1 Initial Training

- Admin user training (4 hours)
- Operator user training (2 hours)
- Training materials provided
- Hands-on practice sessions

14.9.2 Ongoing Training

- New feature training
- Quarterly refresher sessions
- Video tutorials for self-learning
- Knowledge base articles

14.10 Disaster Recovery

14.10.1 Recovery Scenarios

- Server failure
- Database corruption
- Security breach
- Natural disasters
- Human error

14.10.2 Recovery Procedures

59. Assess the situation
60. Activate recovery team
61. Switch to backup systems (if available)
62. Restore from backups
63. Verify data integrity
64. Resume operations
65. Post-incident analysis

14.10.3 Business Continuity

- Recovery Time Objective (RTO): 4 hours
- Recovery Point Objective (RPO): 24 hours
- Communication plan
- Alternate processing site (optional)
- Regular DR drills

15. Appendices

15.1 Glossary

Term	Definition
API	Application Programming Interface - a set of protocols for building software applications
Appointment	A scheduled service booking for a customer
Barcode	Machine-readable representation of product information
Commission	Percentage or fixed amount earned by staff for services performed
Consumable	Products used during service delivery (tracked by usage count)
CRUD	Create, Read, Update, Delete operations

Term	Definition
JWT	JSON Web Token - authentication mechanism
OTP	One-Time Password - temporary password for verification
POS	Point of Sale - system for processing customer transactions
RBAC	Role-Based Access Control - permission system based on user roles
Round-Robin	Algorithm for distributing appointments evenly among staff
SKU	Stock Keeping Unit - unique product identifier
Usage Count	Number of times a consumable product can be used before depletion

15.2 Acronyms

Acronym	Full Form
API	Application Programming Interface
CI/CD	Continuous Integration/Continuous Deployment
CORS	Cross-Origin Resource Sharing
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
DB	Database
DNS	Domain Name System
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JS	JavaScript
JSON	JavaScript Object Notation
LAMP	Linux, Apache, MySQL, PHP
MVC	Model-View-Controller
ORM	Object-Relational Mapping
OTP	One-Time Password
PCI DSS	Payment Card Industry Data Security Standard
PDF	Portable Document Format

Acronym	Full Form
PHP	Hypertext Preprocessor
POS	Point of Sale
RBAC	Role-Based Access Control
REST	Representational State Transfer
RPO	Recovery Point Objective
RTO	Recovery Time Objective
SDK	Software Development Kit
SEO	Search Engine Optimization
SKU	Stock Keeping Unit
SLA	Service Level Agreement
SMS	Short Message Service
SQL	Structured Query Language
SRS	Software Requirements Specification
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
URFS	User Requirements & Functional Specification
UX	User Experience
VPS	Virtual Private Server
WCAG	Web Content Accessibility Guidelines
XSS	Cross-Site Scripting

15.3 Reference Documents

66. **User Requirements & Functional Specification (URFS) v1.0**
- Source document for requirements
- Date: August 11, 2025

67. **Laravel Documentation**
- URL: https://laravel.com/docs/10.x
- Framework reference

68. **React Documentation**
- URL: https://react.dev/
- Frontend framework reference

69. **MySQL Documentation**
- URL: https://dev.mysql.com/doc/
- Database reference

70. **RESTful API Design Best Practices**

- Industry-standard API design patterns

71. **OWASP Top 10**
- URL: https://owasp.org/www-project-top-ten/
- Security vulnerability reference

72. **WCAG 2.1 Guidelines**
- URL: https://www.w3.org/WAI/WCAG21/quickref/
- Accessibility standards

73. **PCI DSS Requirements**
- URL: https://www.pcisecuritystandards.org/
- Payment security standards

15.4 Sample Data

Sample Products:

- Hair Care Products: Shampoos, Conditioners, Hair Oils
- Styling Products: Gels, Sprays, Wax
- Treatment Products: Hair Masks, Serums
- Coloring Products: Hair Dyes, Bleach

Sample Services:

- Haircut (30 min, $35)
- Hair Color (90 min, $75)
- Hair Treatment (45 min, $50)
- Blowdry & Style (30 min, $30)
- Facial (60 min, $60)
- Manicure (30 min, $25)
- Pedicure (45 min, $35)

Sample Customer Data:

```
{   "name": "Jane Smith",   "phone": "+1234567890",   "email":
"jane.smith@example.com",   "date_of_birth": "1990-05-15",   "gender": "female",
"address": "123 Main Street",   "city": "Springfield",   "preferred_employee_id": 2
}
```

Sample Appointment:

```
{   "customer_id": 5,   "employee_id": 2,   "branch_id": 1,   "service_ids": [1,
4],   "appointment_date": "2025-11-05",   "appointment_time": "14:00:00",
"notes": "Customer prefers window seat" }
```

15.5 Code Conventions

15.5.1 PHP/Laravel Conventions

- PSR-12 coding standard
- CamelCase for class names
- camelCase for method names
- snake_case for database columns
- Meaningful variable names
- Doc blocks for all methods
- Type hints for parameters and return types

Example:

```
<?php  namespace App\Services;  use App\Models\Appointment; use App\Repositories\
AppointmentRepository;  class AppointmentService {     /**      * Create a new
appointment      *      * @param array $data      * @return Appointment      */
public function createAppointment(array $data): Appointment     {         //
Implementation     } }
```

15.5.2 JavaScript/React Conventions

- Airbnb JavaScript style guide
- PascalCase for component names
- camelCase for variables and functions
- Functional components with hooks
- Props destructuring
- JSDoc comments for functions

Example:

```
/**  * AppointmentCard Component  * Displays appointment details in a card format
*   * @param {Object} appointment - Appointment data  * @param {Function} onEdit -
Edit callback  * @param {Function} onCancel - Cancel callback  */ const
AppointmentCard = ({ appointment, onEdit, onCancel }) => {   const { customer,
service, appointmentTime } = appointment;      return (     <Card>       {/*
Component JSX */}     </Card>   ); };  export default AppointmentCard;
```

15.5.3 Database Naming Conventions

- snake_case for table and column names
- Plural table names (users, products)
- Singular model names (User, Product)
- Foreign keys: `table_id` (e.g., customer_id)
- Timestamps: created_at, updated_at

15.6 Environment Variables Template

.env (Laravel)

```
APP_NAME="Salon Management System" APP_ENV=production
APP_KEY=base64:generated_key_here APP_DEBUG=false APP_URL=https://yoursalon.com
DB_CONNECTION=mysql DB_HOST=127.0.0.1 DB_PORT=3306 DB_DATABASE=salon_db
DB_USERNAME=salon_user DB_PASSWORD=secure_password  REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null REDIS_PORT=6379  MAIL_MAILER=smtp MAIL_HOST=smtp.sendgrid.net
MAIL_PORT=587 MAIL_USERNAME=apikey MAIL_PASSWORD=sendgrid_api_key
MAIL_ENCRYPTION=tls MAIL_FROM_ADDRESS=noreply@yoursalon.com MAIL_FROM_NAME="$
{APP_NAME}"  SMS_GATEWAY=twilio TWILIO_ACCOUNT_SID=your_account_sid
TWILIO_AUTH_TOKEN=your_auth_token TWILIO_PHONE_NUMBER=+1234567890
WHATSAPP_API_KEY=your_whatsapp_api_key WHATSAPP_PHONE_NUMBER=+1234567890
STRIPE_KEY=pk_test_xxxxx STRIPE_SECRET=sk_test_xxxxx
STRIPE_WEBHOOK_SECRET=whsec_xxxxx  AWS_ACCESS_KEY_ID= AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1 AWS_BUCKET=  QUEUE_CONNECTION=redis
```

.env (React)

```
REACT_APP_API_URL=https://api.yoursalon.com/api/v1 REACT_APP_API_TIMEOUT=30000
REACT_APP_ENVIRONMENT=production REACT_APP_SENTRY_DSN=https://xxx@sentry.io/xxx
```

15.7 Deployment Checklist

- [ ] Code repository access set up
- [ ] Production server provisioned
- [ ] Domain registered and DNS configured
- [ ] SSL certificate installed
- [ ] Database created and secured
- [ ] Environment variables configured
- [ ] Third-party API keys obtained
- [ ] SMTP/Email service configured
- [ ] SMS gateway configured
- [ ] WhatsApp API configured
- [ ] Payment gateway configured (test mode)
- [ ] Backup system configured
- [ ] Monitoring tools set up
- [ ] Firewall rules configured
- [ ] SSH access secured

- [ ] Web server configured (Nginx/Apache)
- [ ] Queue workers configured (Supervisor)
- [ ] Cron jobs scheduled
- [ ] Application deployed and tested
- [ ] Performance testing completed
- [ ] Security testing completed
- [ ] User documentation prepared
- [ ] Training completed
- [ ] Go-live date scheduled
- [ ] Rollback plan documented
- [ ] Support team briefed
- [ ] Client sign-off obtained

15.8 Support Contact Information

Development Team:

- Email: dev@yoursalon.com
- Phone: +1 (XXX) XXX-XXXX

Technical Support:

- Email: support@yoursalon.com
- Phone: +1 (XXX) XXX-XXXX
- Hours: 24/7 for critical issues, 9 AM - 5 PM for non-critical

Emergency Contact:

- Phone: +1 (XXX) XXX-XXXX
- Available: 24/7 for P1 critical issues

Third-Party Support:

- Hosting Provider: support@host.com
- SMS Gateway: support@twilio.com
- Payment Gateway: support@stripe.com

Document Approval

Sign-Off

This Software Requirements Specification (SRS) document has been reviewed and approved by the following stakeholders:

Name	Role	Signature	Date
Project Manager			
Technical Lead			
Client Representative			
QA Lead			

End of Document

Document Information:

- **Title**: Software Requirements Specification - Salon Management System
- **Version**: 1.0
- **Date**: October 31, 2025
- **Total Pages**: 62
- **Classification**: Confidential

- **Distribution**: Internal Team, Client

Revision History:

Version	Date	Author	Description
1.0	October 31, 2025	Development Team	Initial SRS Document