Software Requirements Specification (SRS)

Restaurant Management Service

1. Introduction

1.1 Purpose

The purpose of the Restaurant Management Service (RMS) is to streamline restaurant operations through a full-stack web application. It allows customers to browse menus, place orders, view order history, and download invoices in PDF format. Chefs can manage orders in progress, while Admins oversee menu items, users, reports, and system settings. This document outlines the requirements and specifications for RMS.

1.2 Document Conventions

Abbreviations:
- RMS: Restaurant Management Service
- UI: User Interface
- RBAC: Role-Based Access Control

Standards:
- This document follows IEEE guidelines for SRS documentation.

1.3 Intended Audience and Reading Suggestions

The primary audience for this document includes:
- Developers: Focus on sections 3 and 4 for functional and interface requirements.
- Testers: Refer to Section 3 for system features to design test cases.
- Managers/Instructors: Review sections 1 and 2 for scope and project objectives.
- End Users: Admin, Chef, and Customer roles should read Section 3 for feature details.

1.4 Project Scope

The RMS is a standalone PHP-MySQL application deployed via XAMPP. It contains Home, Our Menu, About Us, Contact Us, Login/Logout, and dedicated dashboards for Admin, Chef, and Customer. Key objectives include providing role-based access, generating PDF order bills, and enabling Admins to monitor restaurant performance through reports.

1.5 References

- PHP 8.x and MySQL 8.x Documentation
- Bootstrap 5.x Documentation

- XAMPP Setup Guide
- FPDF/TCPDF Documentation for PDF invoice generation

2. Overall Description

2.1 Product Perspective

The RMS is a full-stack, standalone web app running locally on XAMPP. It integrates frontend (HTML, CSS, Bootstrap, JavaScript) with backend (PHP) and database (MySQL). The application follows a modular approach with role-based dashboards and centralized data management.

2.2 Product Features

- Home Page: Overview of restaurant services.
- Our Menu: Browse items by category with images and prices.
- Customer Dashboard: Place orders, view order history, download PDF bills.
- Chef Dashboard: Track incoming orders and update preparation status.
- Admin Dashboard: Manage users, menu, and generate reports.
- About Us and Contact Us pages.
- Login/Logout: Authentication with role-based redirection.
- Reports: Generate sales, revenue, and order trend reports.

2.3 User Classes and Characteristics

- Admin: Advanced users, responsible for managing system, menu, users, and reports.
- Chef: Intermediate users, handle kitchen orders and update statuses.
- Customer: Basic users, browse menu, place orders, view bills.
- Guest: General users, limited to viewing Home, About Us, and Menu.

2.4 Operating Environment

- Server: Apache (XAMPP) with PHP 8.x and MySQL.
- Client: Compatible with modern browsers (Chrome, Edge, Firefox).
- Libraries: Bootstrap 5 for UI; FPDF/TCPDF for PDF bill generation.

2.5 Design and Implementation Constraints

- Local deployment on XAMPP.
- MySQL schema normalized to 3NF.
- Secure session management and RBAC.
- Use of Bootstrap for responsive design.

2.6 User Documentation

- README with installation instructions.
- User guide with screenshots of dashboards.
- FAQ section for common issues.

2.7 Assumptions and Dependencies

- Assumes stable local environment.
- Single restaurant branch.
- Orders for dine-in or take-away only.

3. System Features

3.1 Authentication and RBAC

Description and Priority: Secure login/logout with role-based dashboards. Priority: High.

Stimulus/Response Sequences:
User submits login form → credentials validated → redirected to respective dashboard.

Functional Requirements:
- Passwords hashed with bcrypt/Argon2.
- Session-based role verification.
- Access restrictions per role.

3.2 Menu Management

Description and Priority: Admin manages menu items; customers view items with images and categories.

Stimulus/Response Sequences:
Admin creates/edits/deletes items → System updates menu → Customers see updated menu.

Functional Requirements:
- CRUD functionality for menu items.
- Image upload and storage.
- Display only active items to customers.

3.3 Customer Orders

Description and Priority: Customers place orders and download PDF invoices. Priority: High.

Stimulus/Response Sequences:
Customer selects items → System calculates totals → Order confirmed → PDF invoice generated.

Functional Requirements:
- Create order with line items.
- Track order statuses (Placed, Preparing, Ready, Completed).
- Generate invoice in PDF.

3.4 Chef Dashboard

Description and Priority: Chefs track orders and update their status. Priority: High.

Stimulus/Response Sequences:
Chef sees queued orders → Updates to Preparing/Ready → System logs change.

Functional Requirements:
- Display pending orders.
- Update statuses with timestamps.
- Provide sound/visual alerts for new orders.

3.5 Admin Dashboard and Reports

Description and Priority: Admins manage users, menu, and generate reports. Priority: High.

Stimulus/Response Sequences:
Admin logs in → Views KPIs (sales, revenue) → Exports reports.

Functional Requirements:
- CRUD for users and roles.
- Generate daily/weekly/monthly sales reports.
- Export reports to PDF/CSV.

3.6 Contact Us & Feedback

Description and Priority: Guests submit inquiries; Admins review. Priority: Medium.

Stimulus/Response Sequences:
User fills form → Message stored in DB → Admin views in dashboard.

Functional Requirements:
- Store message with timestamp.
- Notify Admin via dashboard.

4. External Interface Requirements

4.1 User Interfaces

- Responsive UI with Bootstrap 5.
- Dashboards with tables, forms, and navigation bar.

4.2 Hardware Interfaces

Runs on desktops/laptops; supports mobile browsers.

4.3 Software Interfaces

- PHP-MySQL integration using PDO/MySQLi.
- PDF generation using FPDF/TCPDF.
- Optional email notifications via PHPMailer.

4.4 Communication Interfaces

HTTP/HTTPS over Apache on localhost.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- Load time under 2 seconds on localhost.
- Support 20+ concurrent users.

5.2 Safety Requirements

- Fail gracefully with error messages.
- Database backups recommended.

5.3 Security Requirements

- Password hashing with bcrypt/Argon2.
- SQL injection prevention via prepared statements.
- CSRF tokens for forms.

5.4 Software Quality Attributes

- Usability through consistent UI.
- Maintainability with modular PHP.
- Reliability with transactional order processing.
- Portability across LAMP/WAMP.

6. Other Requirements

- Audit logs for admin actions.
- Order status history maintained.

Appendices

A. Glossary

- RBAC: Role-Based Access Control
- Invoice: PDF summary of order items
- KPI: Key Performance Indicator

B. Analysis Models

Actors: Admin, Chef, Customer, Guest.
Use Cases: Authentication, Manage Menu, Place Order, Update Order, Generate Reports.

C. Database Schema

users(id, name, email, password_hash, role, created_at)
customers(id, user_id, phone, address)
chefs(id, user_id, station)
menu_items(id, name, category, price, image, status)
orders(id, customer_id, date, status, total)
order_items(id, order_id, menu_item_id, qty, unit_price)
payments(id, order_id, method, amount, invoice_path)
messages(id, name, email, subject, message, created_at)