Software Requirements Specification (SRS) - Single Restaurant Food Delivery App

1. Introduction

1.1 Purpose of Document

This document specifies the functional and non-functional requirements for developing a food ordering and delivery system for a single restaurant. It aims to ensure that all stakeholders (developers, product owner) have a clear understanding of the system's capabilities and constraints

1.2 Scope of Product

The food delivery application will allow customers to browse the menu, place orders, and track their order status. The system will also provide an Admin Dashboard for the restaurant to manage its menu, orders, and customers.

Main Features:

- Display menu categories and food items
- Place a food order online
- Track live order status (Pending → Preparing → Out for Delivery → Delivered)
- customer registration
- Restaurant admin can manage menu, orders, and offers

Limitations:

- System supports one restaurant only
- No support for delivery driver tracking
- No integration with payroll or HR systems
- Limited external integrations

1.3 Definitions, Acronyms, Abbreviations

- **FDS**: Food Delivery System
- **API**: Application Programming Interface
- **Admin**: Restaurant manager who manages the system
- **Staff**: Restaurant staff who manage orders related stuff (view incoming orders , manage order status , orders issue ,etc...)
- **Delivery staff**: view assigned orders and changes order status to delivered
- **User / Customer**: The person placing food orders
- **Order Status**: The state of an order (Pending, Preparing, Out for Delivery, Delivered, Canceled)
- **MVP**: Minimum Viable Product
- **AOP**: Aspect-Oriented Programming.

1.4 References

IEEE/ANSI 830-1993 Requirements Specification Standard

2. General Perspective

2.1 Product Perspective

The Food Delivery System is a centralized online ordering system that allows a restaurant to operate its delivery service digitally.

The application will include:

- A customer interface (mobile or web)
- A restaurant admin dashboard
- A backend API built with Spring Boot
- A database to store menu items, orders, and user data

The platform streamlines food ordering, improves customer experience, and reduces manual restaurant workload.

2.2 Product Functions

The platform must support:

Menu management: Add / Edit / Delete food items, manage categories and set item availability

Order Management: Customers place new orders, staff manages order statuses and views ongoing and previous orders

User Registration & Authentication: Login, register, save addresses**,** view past orders

Payment Handling: Cash on delivery

Admin Dashboard: Provide a panel to monitor orders and manage the menu.

2.3 User Characteristics

Customers:

- Expected to have basic mobile/website skills
- Need a simple UI to browse menu and place orders

Admin:

- Moderate technical skills
- Must manage menu items, prices

Staff:

- Manage order status
- Can view assigned orders

2.4 General Constraints

Technical Constraints:

- Backend must be built using Spring Boot
- APIs must follow REST architecture
- Database is required
- The system must support role-based access (Admin vs User)

Security Constraints:

- Users' personal data must be encrypted
- Authentication tokens must be securely stored

Organizational Constraints

- Restaurant is responsible for updating its menu
- Admin must manually update order price or add items
- Only admin can POST/PUT/DELETE /menu

2.5 Assumptions and Dependencies

Assumptions:

- Customers have stable internet connection
- Restaurant staff can use the dashboard
- Users can access the system from mobile or desktop

Dependencies:

- Reliance on the successful integration of user authentication and authorization logic.
- Dependence on a functioning database instance.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 Menu Management (API Endpoints: /menu)

FR-1.1 (Admin): The Admin must be able to add new products (including name, category, price, image, description, and availability status) via POST /menu.

FR-1.2 (Admin): The Admin must be able to modify (e.g., change price, availability) and delete existing products via PUT /menu/{id} and DELETE /menu/{id} respectively.

FR-1.3 (Customer/Public): The system must allow fetching the complete menu and item details via GET /menu and GET /menu/{id}.

3.1.2 Order Processing (API Endpoints: /orders)

FR-2.1 (Customer/Public): The system must allow a user to create a new order (listing items, quantities, and customer data) via POST /orders.

FR-2.2: The system must accurately calculate the final price of the order, including item prices and quantity.

FR-2.3 (Staff): Staff must be able to update the order status (Pending, Preparing, Out for delivery, Delivered, Canceled) via PUT /orders/{id}/status.

FR-2.4 (Customer/Public): The system must allow a user to retrieve their specific order details and current status via GET /orders/{id}.

3.1.3 User Registration and Roles (API Endpoints: /register, /login)

FR-3.1: The system must include user registration and authentication logic via POST /register and POST /login.

FR-3.2: Admin must assign users to distinct roles (e.g., Client, Staff).

FR-3.3: The system must enforce API authorization, ensuring only the Admin role can access management endpoints (e.g., POST/PUT/DELETE /menu).

3.1.4 Payment System

FR-4.1: The system must support Cash on Delivery for payment.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

NFR-1.1: The system must provide quick response times for all critical user actions (e.g., fetching the menu, placing an order).

3.2.2 Security Requirements

NFR-2.1: All sensitive data, including customer's personal information (Name, Address, Phone) and order details, must be encrypted both in transit and at rest.

NFR-2.2: The system must enforce authorization checks based on user roles for all API access.

3.2.3 Usability Requirements

NFR-3.1: The design (UI/UX) must be simple and user-friendly for placing orders.

NFR-3.2: The system must be accessible from various devices (computer, phone, tablet).

3.2.4 Reliability and Availability Requirements

NFR-4.1: The system must maintain an uptime of 99.9%.

3.3 Interface Requirements

3.3.1 User Interface Requirements:

- Fully responsive UI
- Customer Home Page must show menu categories
- Product page must show: (Image , Price , Description)
- Admin Dashboard must include: (Menu management panel , Orders management table , Status update buttons)

3.4 Logical Database Requirements

The database must store data for all core entities: Categories, Products, Orders, Order Items, Users, and Delivery Addresses.

The database must support ACID properties (Atomicity, Consistency, Isolation, Durability) to ensure data integrity for all transactions (especially ordering).

4. Appendixes

4.1 Sample Input Format

- Customer name
- Phone number
- Address
- Item id
- Quantity

4.2 Sample Output Format

- Order ID
- List of items
- Total price
- Order status
- Estimated delivery time

4.3 Problem Description

Restaurants struggle with:

- Receiving orders through phone calls
- Miscommunication about order details
- Unclear timing for order preparation and delivery
- Difficulty tracking orders and revenue manually

The Food Delivery System solves these problems with a digital ordering workflow.

4.4 Stakeholder

- Restaurant manager
- Restaurant Staff
- Customers
- Development Team

5. Index