Software Requirements Specification (SRS)

Online Food Ordering System

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the requirements for an Online Food Ordering System (OFOS). It describes the system's intended features, behaviors, constraints, and interfaces for stakeholders including customers, restaurant owners, delivery staff, administrators, and the development team.

1.2 Scope

The Online Food Ordering System enables customers to browse restaurants, view menus, place and pay for orders online, and track order status in real time. Restaurants can manage menus, handle incoming orders, and update order statuses. Administrators can manage users, restaurants, and system-wide configurations.

The system will provide:

- User registration and authentication for customers, restaurant owners, and admins.
- Restaurant listing and search with filters (cuisine, rating, delivery time).
- Menu browsing with item details (price, description, availability).
- Shopping cart and order placement.
- Online payment integration (simulated or with external payment gateway).
- Real-time order status tracking (Placed, Accepted, Preparing, Out for Delivery, Delivered, Cancelled).
- Basic analytics for restaurants and administrators (e.g., total orders, revenue, popular items).

The system will be implemented as a web application and, for this project, built using Appsmith as the front-end application layer integrated with a database or APIs.

1.3 Definitions & Acronyms

- **OFOS**: Online Food Ordering System.
- **Customer**: End-user who browses restaurants, places orders, and makes payments.
- **Restaurant Owner**: User who manages a specific restaurant's menu and orders.
- **Admin**: System administrator with privileges to manage users, restaurants, and configurations.
- **Order**: A confirmed request by a customer for one or more menu items from a restaurant.
- **Cart**: Temporary collection of items selected by a customer before checkout.
- **NFR**: Non-Functional Requirement.
- **UI**: User Interface.
- **API**: Application Programming Interface.

2. Overall Description

2.1 Product Perspective

The OFOS is a standalone web application that interacts with:

- A relational database (e.g., PostgreSQL/MySQL) or REST APIs that store/manage users, restaurants, menus, and orders.
- Optionally, a third-party payment gateway for processing online payments.

In Appsmith, the system will be composed of multiple pages, each connected to data sources through queries and APIs. The application will sit between users (via browser) and the backend data services.

2.2 Product Functions

At a high level, the system will provide the following functions:

- **User Management**: Register, log in, log out, and manage basic profile information.
- **Restaurant Management**: View list of restaurants; admins approve/add restaurants; restaurant owners manage their own restaurant info.
- **Menu Management**: Restaurant owners manage menu categories and items (create, read, update, delete).
- **Ordering**: Customers select items, manage a cart, place orders, and receive confirmation.
- **Payment Handling**: Customers pay for orders online (or mark as Cash on Delivery), with storage of payment status.
- **Order Tracking**: Customers see current order status; restaurant owners and delivery staff update statuses.
- **Reporting & Analytics**: Basic dashboards for restaurant owners and admins (e.g., daily orders, revenue, popular dishes).

2.3 User Characteristics

- **Customers**:
    - Basic computer/smartphone literacy.
    - Expect intuitive navigation and minimal steps to place an order.
- **Restaurant Owners/Staff**:
    - Familiar with basic web applications.
    - Need efficient views for new orders and quick status updates.
- **Admins**:
    - Technical or semi-technical background.
    - Comfortable with management dashboards and configuration panels.

2.4 Constraints

- The application UI will be implemented using Appsmith, which imposes:
    - Available widget types and layout constraints.
    - Query execution limits and security rules defined by Appsmith.
- The system must run in a standard web browser (latest Chrome/Edge/Firefox).
- Backend (database/APIs) must be reachable from Appsmith over HTTPS.
- Performance constraints:

- Typical user load: small-to-medium scale (e.g., up to a few hundred concurrent users for this project scope).
- Security constraints:
    - Authentication must be handled securely (e.g., JWT or session-based via backend; or Appsmith’s built-in auth if applicable).
    - Sensitive data (passwords, payment tokens) must never be stored in plain text.

---

3. Specific Requirements

3.1 Functional Requirements

FR1 – User Registration & Login

- The system shall allow users (customers, restaurant owners, admins) to register with email and password.
- The system shall allow registered users to log in and log out.
- The system shall restrict access to certain pages based on user role.

FR2 – Restaurant Listing & Search

- The system shall display a list of active restaurants to customers.
- The system shall allow users to search restaurants by name, cuisine type, or location.
- The system shall allow sorting/filtering by rating and estimated delivery time (if available in data source).

FR3 – Menu Browsing

- The system shall display the menu for a selected restaurant, organized by categories (e.g., Starters, Main Course, Desserts).
- The system shall display key details for each menu item: name, description, price, and availability status.

FR4 – Shopping Cart Management

- The system shall allow customers to add one or more menu items to a cart.
- The system shall allow customers to modify quantities and remove items from the cart.
- The system shall calculate and display order subtotal, taxes/fees (if applicable), and total amount in real time.

FR5 – Order Placement

- The system shall allow customers to place an order from the cart by providing delivery address and contact information.
- The system shall validate that the cart is not empty before placing the order.
- The system shall create an order record with status set to "Placed" and an associated timestamp.

FR6 – Payment Processing

- The system shall support at least two payment options: Online Payment and Cash on Delivery (COD).
- For Online Payment, the system shall integrate with a payment API or simulate payment confirmation for the prototype.

- The system shall store payment status (e.g., Pending, Paid, Failed) associated with each order.

FR7 – Order Status Management & Tracking

- The system shall allow restaurant owners or staff to update order status through predefined states: Placed, Accepted, Preparing, Out for Delivery, Delivered, Cancelled.
- The system shall allow customers to view the current status of their orders from an "My Orders" page.
- The system shall refresh or poll order status periodically (or on user action) for near real-time updates.

FR8 – Restaurant & Menu Administration

- The system shall allow restaurant owners to create, update, and deactivate menu items.
- The system shall allow restaurant owners to update restaurant details (name, address, contact, opening hours, cuisines, delivery settings).
- The system shall allow admins to activate/deactivate restaurants.

FR9 – Reporting & Analytics (Optional but recommended)

- The system shall provide a dashboard for restaurant owners showing key metrics (e.g., number of orders per day, total revenue per day, top-selling items).
- The system shall provide an admin overview dashboard (e.g., total active restaurants, total orders across platform, total revenue).

3.2 Non-Functional Requirements (NFRs)

NFR1 – Usability

- The UI shall be intuitive, with clear labels and minimal steps to place an order.
- The system shall use consistent visual styling (colors, fonts, buttons) across all pages.

NFR2 – Performance

- The system shall load primary user pages (restaurant list and menu pages) in under 3 seconds on a typical broadband connection under nominal load.
- Queries and API calls should be optimized to avoid unnecessary round trips.

NFR3 – Reliability & Availability

- The system shall handle transient backend errors gracefully, displaying user-friendly error messages.
- Core actions (view restaurants, place orders, update status) shall be resilient to brief network interruptions (e.g., retry or allow manual refresh).

NFR4 – Security

- The system shall ensure that only authenticated users can access protected pages (e.g., order history, restaurant management, admin dashboards).
- The system shall enforce role-based access control (RBAC) to restrict admin and restaurant-owner-only functions.
- All communication with backend APIs must use HTTPS.

---

4. Assumptions & Dependencies

4.1 Assumptions

- Users have access to a modern web browser and stable internet connection.
- Restaurants provide accurate and up-to-date menu and pricing information.
- Payment gateway (if used) is available and correctly configured.
- Delivery operations (drivers, logistics) are managed outside the scope of this system; the system only tracks status updates.
- Email verification, SMS notifications, and advanced marketing features are out of scope for this initial version.

4.2 Dependencies

- Availability and reliability of the chosen database system (e.g., PostgreSQL/MySQL).
- Availability of any third-party payment gateway service (if integrated).
- Availability and performance of Appsmith cloud/self-hosted instance where the application is deployed.
- Proper configuration of authentication/authorization mechanisms used by Appsmith and/or backend APIs.
