**Software Requirements Specification (SRS)**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to specify the requirements for the Food Ordering Web Application, which allows users to browse restaurant menus, add items to a cart, place orders, and allows restaurants to manage their menus and view orders.

**1.2 Scope**

This application is a full-stack web platform for food ordering, supporting:

* Buyer registration, authentication, and order placement
* Restaurant registration, authentication, menu management, and order management
* Admin features (optional/future)
* Responsive UI for both buyers and restaurants

**1.3 Definitions, Acronyms, and Abbreviations**

* **Buyer**: End-user who orders food
* **Restaurant**: Vendor who manages menu and receives orders
* **Menu Item**: Food item offered by a restaurant
* **Order**: A collection of menu items purchased by a buyer

**2. Overall Description**

**2.1 Product Perspective**

* The system is a web application built with React (frontend) and Node.js/Express (backend), using MySQL as the database.
* The frontend and backend communicate via RESTful APIs.

**2.2 Product Functions**

* Buyer registration and login (with OTP verification)
* Restaurant registration and login (with OTP verification)
* Menu management (add, remove, edit items)
* Cart management (add, remove items, view cart)
* Order placement and order history for buyers
* Order management for restaurants (view, dispatch, delete orders)
* Profile management for restaurants

**2.3 User Classes and Characteristics**

* **Buyers**: Can browse menus, add to cart, place orders, view order confirmation.
* **Restaurants**: Can manage their menu, view and dispatch orders, edit profile.
* **Admin**: (Optional/future) Can manage users and restaurants.

**2.4 Operating Environment**

* Web browsers (Chrome, Firefox, Edge, Safari)
* Node.js server environment
* MySQL database

**2.5 Design and Implementation Constraints**

* Must use React for frontend, Node.js/Express for backend, MySQL for database.
* RESTful API design.
* Responsive design for desktop and mobile.

**2.6 User Documentation**

* User guide for buyers and restaurants (to be provided separately).

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 Buyer Features**

* **Registration & Login**: Buyers can sign up and log in using email and password, with OTP verification.
* **Browse Menu**: Buyers can view all menu items from all restaurants.
* **Search Menu**: Buyers can search for menu items by name or restaurant.
* **Cart Management**: Buyers can add/remove items to/from the cart.
* **Place Order**: Buyers can place orders by providing name, phone, and address.
* **Order Confirmation**: Buyers see a confirmation page after placing an order.

**3.1.2 Restaurant Features**

* **Registration & Login**: Restaurants can sign up and log in using email and password, with OTP verification.
* **Menu Management**: Restaurants can add, edit, and remove menu items.
* **Order Management**: Restaurants can view incoming orders, mark them as dispatched, and remove them.
* **Profile Management**: Restaurants can edit their profile and image.

**3.1.3 Common Features**

* **Authentication**: Session management for buyers (localStorage) and restaurants (sessionStorage).
* **Responsive UI**: All pages are responsive and mobile-friendly.
* **Navigation**: Navbar adapts to user type and authentication state.

**3.2 Non-Functional Requirements**

* **Performance**: The system should handle at least 100 concurrent users.
* **Security**: Passwords are hashed; sensitive actions require authentication.
* **Reliability**: The system should recover gracefully from errors.
* **Usability**: The UI should be intuitive and easy to use.
* **Scalability**: The backend should be modular for future scaling.

**3.3 Database Requirements**

**3.3.1 Tables**

* **buyers**: id, name, email, password
* **restaurants**: id, name, email, password, description, image
* **menu**: id, restaurant\_name, item\_name, price, image
* **orders**: id, buyer\_name, address, contact, total, items (JSON), restaurantName, status, created\_at

**4. External Interface Requirements**

**4.1 User Interfaces**

* **Home Page**: Menu browsing, search, add to cart
* **Cart Page**: View, remove items, place order
* **Order Confirmation Page**: Order details and print option
* **Restaurant Dashboard**: Menu management, order management, profile editing

**4.2 Hardware Interfaces**

* Standard web browser and internet connection

**4.3 Software Interfaces**

* RESTful API endpoints (see backend routes)
* MySQL database

**5. System Features**

**5.1 Buyer Registration and Login**

* OTP-based email verification
* Secure password storage

**5.2 Restaurant Registration and Login**

* OTP-based email verification
* Secure password storage

**5.3 Menu Management**

* Add, edit, remove menu items
* Image upload via URL

**5.4 Cart and Order Management**

* Add/remove items to/from cart
* Place order with delivery details
* View order confirmation

**5.5 Restaurant Order Management**

* View all orders for the restaurant
* Mark orders as dispatched and remove them

**5.6 Profile Management**

* Edit restaurant name and image
* Delete restaurant account

**6. Other Requirements**

* **Error Handling**: All API errors are handled gracefully and shown to the user.
* **Session Management**: Buyers and restaurants are kept logged in using localStorage/sessionStorage.
* **Accessibility**: Basic accessibility standards are followed.

**7. Appendix**

* **Tech Stack**: React, Node.js, Express, MySQL, Bootstrap
* **Deployment**: Node.js server, MySQL database, static hosting for frontend (Vite build)
* **Future Enhancements**: Admin panel, order tracking, payment integration, reviews/ratings

**End of SRS**