Project Name:- The Restaurant Order Management System

Software Requirements Specification

Course Code:- INT 219 & INT 220
Course Name:- Front-End Web Developer & Sever site scripting

Student Names

Student Names:

Name	Registration No.
Palak Kumari	12323549
Archi Kumari	12302306
Tanish Garg	12306446
Aryan Kumar	12312571

Prepared for
Continuous Assessment 2
Spring 2025

Table of Contents

1. Introduction

This Software Requirements Specification (SRS) document provides a detailed and structured overview of the Restaurant Order Management System—a full-stack web application designed to enhance the efficiency of restaurant order handling processes. It serves as a blueprint for both the frontend and backend development teams, outlining the expectations, functional requirements, non-functional needs, and the operational flow of the application.

The system enables customers to seamlessly browse menus, customize and place food orders online, while restaurant staff can manage incoming orders, track statuses, and update order information in real-time. The goal is to minimize human error, reduce wait times, and create a digital bridge between customers and restaurant operations. This document aims to provide all technical stakeholders with a clear, unified view of how the system should behave and be implemented.

1.1 Purpose

The purpose of this document is to specify the requirements of the Restaurant Order Management System, which provides a user-friendly platform for customers to place food orders and a backend system for restaurant staff to manage those orders efficiently. It aims to guide the development process by detailing all the functionalities, constraints, and interactions necessary to implement the system successfully.

This SRS is intended for:

- Software Developers (Frontend and Backend)
- UI/UX Designers
- QA Testers
- Project Managers
- Restaurant Business Stakeholders

1.2 Scope

The Restaurant Order Management System is a web-based application that simplifies online ordering for restaurants. It allows:

- **Customers** to:
  - Browse the digital menu
  - Customize orders (quantity, special instructions)
  - Place orders and receive confirmation
  - Track real-time status updates
- **Restaurant Staff** to:
  - View and manage incoming orders
  - Change order status (e.g., "preparing", "ready", "delivered")
  - Manage menu items (add, update, remove dishes)

- Monitor customer details and order history

The **frontend** is designed using HTML, Tailwind CSS, and JavaScript to provide a responsive, intuitive UI. The **backend** includes database management, business logic, and RESTful APIs to process, store, and retrieve order data. The system supports secure user sessions, real-time updates, and scalable deployment.

1.3 Definitions, Acronyms, and Abbreviations

- **ROMS**: Restaurant Order Management System – The proposed system designed to handle customer food orders and streamline restaurant operations.
- **UI (User Interface)**: The visual part of the application through which users interact with the system.
- **UX (User Experience)**: The overall experience and satisfaction level of users when interacting with the system.
- **HTML (HyperText Markup Language)**: The standard markup language used to create the structure of web pages.
- **CSS (Cascading Style Sheets)**: A stylesheet language used to describe the presentation of HTML elements on screen.
- **SRS (Software Requirements Specification)**: This document that outlines the detailed functional and non-functional requirements of the system.
- **Backend** – Server-side application for processing logic, data storage, and APIs
- **API** – Application Programming Interface

1.4 References

- IEEE Standard 830-1998: Software Requirements Specification Practices
- Tailwind CSS Documentation: https://tailwindcss.com/docs
- HTML5 & JavaScript Standards: https://developer.mozilla.org
- Node.js Documentation *(or your chosen backend language)*: https://nodejs.org/en/docs/
- OWASP Security Best Practices: https://owasp.org
- GitHub Repository *(if any internal version control is used)*

1.5 Overview

This document is structured to serve as a comprehensive guide for the development of the **Restaurant Order Management System (ROMS)**, covering both frontend and backend components. It outlines the system's purpose, scope, and detailed functional and non-functional requirements that must be addressed by the development team.

Following this introductory section, the document includes:

- **Overall Description**: Explains the system's perspective, user classes, operating environment, and essential design considerations for both the client-side and server-side.

- **Specific Requirements**: Breaks down the core functionalities such as menu browsing, order placement, real-time status tracking, and order management, along with backend processing logic and database interactions.
- **Interface Requirements**: Details the layout and design elements of the user interface using Tailwind CSS, and also outlines API specifications, communication protocols, and data exchange formats.
- **Performance Requirements**: Defines expectations for responsiveness, real-time processing, scalability, load handling, and cross-device compatibility.

This SRS ensures that all stakeholders—including clients, frontend/backend developers, UI/UX designers, testers, and project managers—share a unified understanding of the system's features, constraints, and deliverables to enable seamless development and deployment.

2. General Description

This section provides a high-level overview of the Restaurant Order Management System (ROMS), covering the essential factors that influence the design, development, and functionality of the system. It aims to create a clear understanding of the environment in which the system will operate and the primary goals it is intended to achieve. While this section does not list specific technical requirements, it helps frame the context in which those requirements will be defined and implemented in later sections.

The ROMS is developed as a full-featured, responsive web application that digitizes and streamlines the order-taking and order-tracking processes in a restaurant setting. The system will be accessible to both customers and restaurant staff through intuitive user interfaces and supported by a robust backend infrastructure that handles data management, security, and business logic.

The system is intended to:

- Improve the speed and accuracy of food order processing.
- Enhance customer convenience by enabling online ordering.
- Support real-time communication between customers and restaurant staff.
- Maintain organized records of all orders and customer interactions.

By automating and simplifying manual tasks such as taking, processing, and managing orders, the ROMS is expected to reduce errors, save time, and improve overall customer satisfaction. The system will be designed to be scalable, allowing future enhancements such as integration with payment gateways, customer feedback systems, or third-party delivery platforms.

2.1 Product Perspective

The Restaurant Order Management System (ROMS) is a web-based application developed to enhance the efficiency of restaurant operations by digitizing the order-taking and management process. The system is designed as a **modular and scalable platform**, consisting of:

- A **frontend interface** built using HTML, Tailwind CSS, and JavaScript for seamless customer and staff interaction.
- A robust **backend** powered by technologies like Node.js, Express, or similar frameworks, handling data storage, user authentication, order processing, and real-time updates via APIs.

This system can be integrated into existing restaurant management workflows or deployed as a standalone product.

2.2 Product Functions

The ROMS provides the following core functionalities:

- **Customer Interface (Frontend):**
  - Browse the digital menu with food categories, descriptions, and prices.
  - Add/remove items from a cart.
  - Place online orders with customizations.
  - View order status updates in real-time.
- **Admin/Staff Interface (Frontend):**
  - View incoming orders in real-time.
  - Update order statuses (e.g., Preparing, Ready, Delivered).
  - Manage menu items (add/edit/remove).
  - View customer details and order history.
- **Backend System:**
  - Handle order requests and database transactions.
  - Store customer, order, and menu data securely.
  - Manage user roles and authentication (staff, admin, customer).
  - Provide APIs for real-time communication between frontend and database.

2.3 User Characteristics

The Restaurant Order Management System (ROMS) is intended for use by a variety of users, each with different levels of technical expertise. The system must be intuitive and user-friendly to accommodate all intended user groups:

- **Customers**
  - Technical Skill Level: Basic computer/mobile device usage.
  - Expectations: A responsive, easy-to-use interface for browsing the menu, placing orders, and viewing order statuses.
  - Access Method: Primarily through web browsers on mobile phones or desktops.
- **Restaurant Staff**
  - Technical Skill Level: Moderate familiarity with computer systems.
  - Expectations: Ability to view, manage, and update incoming customer orders, track order statuses, and manage menu items.
  - Access Method: Through a staff panel with secure login.
- **Administrators**

    - Technical Skill Level: Advanced.
    - Expectations: Full access to backend systems, including user management, order database, analytics, and performance monitoring.
    - Access Method: Admin dashboard with complete privileges.

All user interfaces should provide clear navigation, validation prompts, and visual cues to help users interact efficiently with the system.

2.4 General Constraints

The design and development of the Restaurant Order Management System (ROMS) are influenced by several general constraints. These constraints restrict the choices available to the developers and must be taken into consideration throughout the system's lifecycle. These include:

- **Platform Constraints**
    - The application must be accessible through modern web browsers (e.g., Chrome, Firefox, Safari, Edge) on both desktop and mobile devices.
    - The system must be deployable on standard web hosting services or cloud platforms such as Heroku, Vercel, or AWS.
- **Development Language and Tools**
    - The frontend must be developed using HTML, Tailwind CSS, and JavaScript to ensure responsiveness and compatibility.
    - The backend should use widely supported frameworks and technologies such as Node.js with Express.js and a database such as MongoDB or MySQL.
    - The development tools used should support cross-platform development and version control (e.g., Git).
- **Security Constraints**
    - The system must use HTTPS for all communications.
    - Authentication mechanisms (such as JWT or session-based login) must be implemented for staff and administrators.
    - User information must be stored securely with proper encryption for sensitive data.
- **Time and Budget Limitations**
    - The project must be developed and delivered within the predefined timeline and budget constraints.
    - Resource usage (such as server hosting or API calls) should be optimized to avoid exceeding free-tier limits on hosting platforms.
- **Database Constraints**
    - The database schema must support scalability to accommodate increasing orders and user data without significant restructuring.
    - Data consistency and integrity must be maintained during concurrent operations.
- **Legal and Regulatory Compliance**
    - The application must comply with applicable data protection and privacy regulations (e.g., GDPR, if deployed in Europe).

    - Customer data must be handled responsibly and in accordance with privacy policies.
- **Maintenance and Update Constraints**
    - The system should be designed in a modular way to allow for easy updates and feature additions in the future.
    - Downtime during updates should be minimal and should not affect the core functionality.

2.5 Assumptions and Dependencies

- Users have a stable internet connection.
- Staff and admin users have access to a computer or mobile device.
- Integration with a payment gateway is optional and may be added later.
- Third-party libraries and frameworks (e.g., Axios, Express.js) will be used for faster development.

3. Specific Requirements

This section defines the detailed functional and non-functional requirements of the Restaurant Order Management System (ROMS). These requirements describe how the system should behave and what it must accomplish to meet user and business expectations.

The ROMS will allow customers to:

- Browse a digital menu with categories and item descriptions.
- Place food orders and receive confirmation.
- Track the status of their orders in real-time.

The system will allow restaurant staff and administrators to:

- Receive, manage, and update customer orders.
- Track the order preparation and delivery status.
- Update the menu, manage user accounts, and view order history and reports.

3.1 External Interface Requirements

3.1.1 User Interfaces

The system shall include the following user-facing interfaces:

- **Customer Interface (Frontend):**
    - A responsive and visually appealing UI using Tailwind CSS.
    - Menu browsing functionality with categories, search, and filters.
    - An order cart page with editable item quantities.
    - A real-time order tracking interface.
    - A registration/login system for account creation and order history viewing.

- **Staff/Admin Interface:**
  - A dashboard to view incoming orders, update statuses (e.g., preparing, ready, delivered).
  - A panel to update menu items (add/edit/delete).
  - Interfaces for viewing analytics (total orders, peak hours, etc.).

3.1.2 Hardware Interfaces

- The system will not directly interact with any specialized hardware devices.
- It is assumed that all users (customers and staff) will access the application through standard hardware (desktops, laptops, tablets, or smartphones) with internet connectivity.
- For optimal experience, devices should support modern browsers with JavaScript enabled.

3.1.3 Software Interfaces

The system will interact with the following software components:

- **Frontend Technologies:**
  - HTML5, Tailwind CSS, JavaScript (for dynamic behavior).
- **Backend Technologies:**
  - Php for backend.
  - Database system (MySQL) for storing order, user, and menu data.

3.1.4 Functional Requirements

- The system shall allow users to register and log in using a secure form.
- The system shall allow customers to view the food menu categorized by dish types (e.g., Starters, Main Course).
- Users shall be able to add food items to a shopping cart.
- The system shall allow customers to review, modify, and confirm orders.
- The system shall display the status of ongoing orders (e.g., Preparing, Ready, Delivered).
- The system shall provide table booking features with date, time, and number of persons.
- Users shall receive real-time UI notifications on order updates.
- The system shall allow users to submit ratings and feedback after an order is completed.

3.1.5 Non-Functional Requirements

- The user interface shall be responsive and mobile-friendly.
- Page loading time shall not exceed 3 seconds on average.
- The UI shall follow accessibility standards (contrast ratio, keyboard navigation).
- All client-side validations (e.g., email format, required fields) shall be enforced.
- The system shall handle at least 100 concurrent users without performance degradation.

3.2 Backend Requirements

3.2.1 External Interfaces

- REST API with endpoints for all core functions
- MySQL/PostgreSQL database
- Payment gateway (e.g., Stripe/PayPal)

3.2.2 Functional Requirements

- The backend shall manage user authentication with hashed passwords.
- The backend shall perform CRUD operations for menu, orders, users, and feedback.
- The backend shall calculate order totals, apply taxes, and handle billing logic.
- Admin shall have access to a dashboard for managing orders, users, and items.
- Order statuses shall be updated in real-time and reflect on the frontend.
- API endpoints shall be created for all frontend data needs (menu, cart, user, etc.).

3.2.3 Non-Functional Requirements

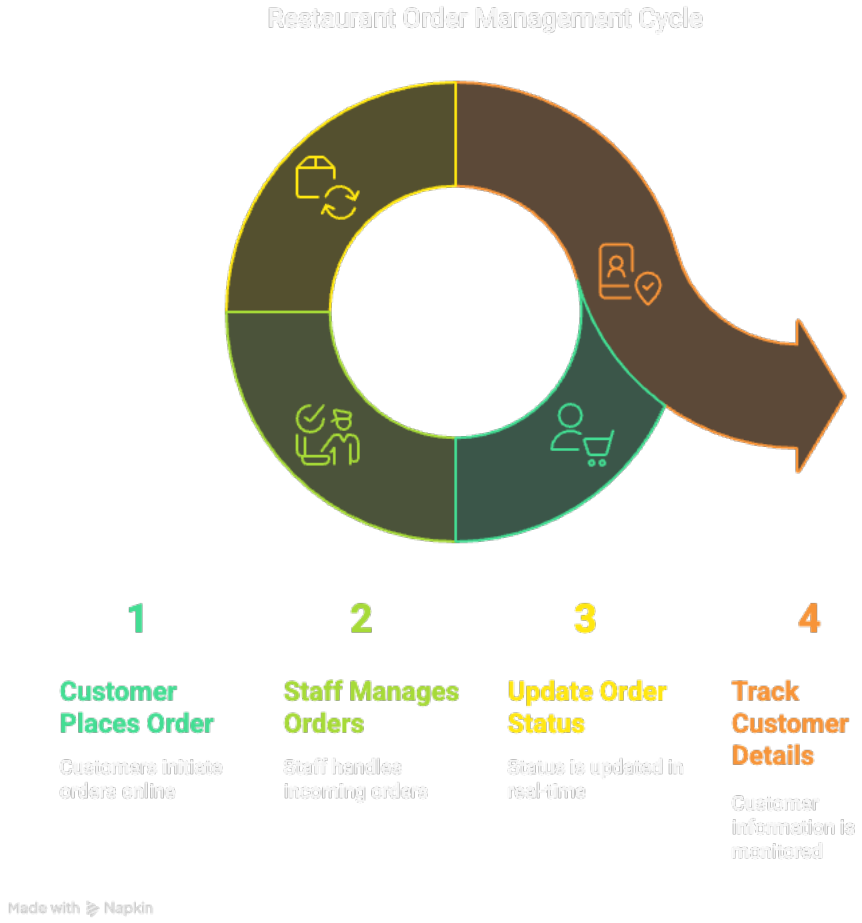
- The server shall be scalable to handle increasing load.
- All API endpoints shall respond within 2 seconds.
- Data privacy and integrity shall be maintained with JWT-based session tokens.
- Backend shall be protected against SQL Injection and XSS attacks.
- Daily backups of the database shall be taken.

3.2.4 Other Requirements

- System logs shall be maintained for admin actions and system errors.
- All food images shall be stored in a cloud or public file storage system.
- The project shall follow a modular structure for maintainability.
- Deployment shall be on a cloud platform (e.g., Vercel for frontend, Render/Heroku for backend).

4. Analysis Models

4.1 Data Flow Diagrams (DFD):-



5. GitHub Link :- https://github.com/archi1012/Akayresturant

A. Appendices

A.1 Appendix 1

The information in this paper is crucial for the development and quality assurance teams.

Every item in the appendix is regarded as a component of the system requirements.