Software Requirements Specification

for

Restaurant Management System

Version 2.0 approved

Prepared by

Hala Mohammed Maher
Nada Khalil Mohammed
Hend Ahmed Haroun
Shahd Hussien
Samar El-Amir
Safia Hassan

Faculty of Computer Science
Hurghada University

11th December 2025

Table of Contents

Table of Contents ..... **ii**
1. Introduction **1**
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 Development Model 2
1.6 References 2
2. Overall Description **2**
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 3
2.4 Operating Environment 3
2.5 Design and Implementation Constraints 3
2.6 User Documentation 4
2.7 Assumptions and Dependencies 4
3. External Interface Requirements **4**
3.1 User Interfaces 4
3.1.1 UI Screen List 4
3.2 Hardware Interfaces 5
3.3 Software Interfaces 5
3.4 Communications Interfaces 5
4. System Features **5**
4.1 Online Ordering 5
4.2 Order Management Dashboard 6
4.3 Menu Recommendation 7
4.4 Basic Reporting 7
5. Other Nonfunctional Requirements **8**
5.1 Performance Requirements 8
5.2 Safety Requirements 8
5.3 Security Requirements 8
5.4 Software Quality Attributes 8
5.5 Business Rules 9
5.6 Constraints & Limitations 9
5.7 External & Organizational Requirements 9
6. Other Requirements **9**
6.1 Database Requirements 9
6.2 Development and Deployment Requirements 9
6.3 Legal and Complaince Requirements 9
6.4 Reuse & Future Extension 10
Appendix A: Glossary **10**
Appendix B: Analysis Models **10**
B.1 Use Case Description 11
B.2 ERD 11
Appendix C: To Be Determined List **12**
Appendix D: Project Risks **12**

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) identifies and defines the requirements for the Restaurant Management System, Release 2.0. This document covers the complete software system, including the customer-facing online ordering website and the staff-facing management dashboard. It includes all functional and non-functional requirements of the system.

1.2 Document Conventions

This document follows the IEEE 830-1998 standard for SRS structure.
we have used the Times New Roman font style with 18 pts bold for main headings, 14 pts bold for subheadings and 12 pts for body text. Priority levels are High (H), Medium (M) and Low (L).

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Developers – to implement the system according to specified requirements.
- Testers – to design test cases and validate functionality.
- Project Managers – to track progress and milestones.
- Restaurant Owners & Staff – to understand system workflow.
- Clients & Stakeholders – to review scope and approve deliverables.

It is recommended to read this document sequentially, starting with Sections 1 and 2 for an overview, then proceeding to Section 3 for detailed requirements. User interface descriptions and system features are elaborated in later sections.

1.4 Product Scope

The Restaurant Management System is a web-based software platform designed to allow customers to order food and enable restaurant staff to manage those orders. Its purpose is to digitize restaurant ordering and management processes, thereby improving order accuracy and enhancing customer experience and reducing operational workload. The software supports the strategic goals of increasing customer satisfaction and enabling scalable, efficient restaurant operations in a digital marketplace.

1.5 Development Model

For this project, we have selected the **Incremental Development model**. This model involves constructing the software in repeated cycles (increments) where each increment adds a portion of the total functionality. We chose this model for several reasons: First, it aligns with our academic timeline. Second, it facilitates continuous feedback from our professor and peers. Third, it reduces project risk by integrating and testing components regularly, rather than leaving all integration for the end. Given our team size and the modular nature of our system features, the incremental approach is the most pragmatic and manageable choice.

1.6 References

- GeeksforGeeks, *Software Requirement Specification (SRS) Format*. Available: https://www.geeksforgeeks.org/software-engineering/software-requirement-specification-srs-format/
- Scikit-learn Developers, *Scikit-learn Documentation*, 2023. Available: https://scikit-learn.org/stable/documentation.html
- Elmasri, R., & Navathe, S. B., Database Systems, 7th Edition, Pearson, 2016.

2. Overall Description

2.1 Product Perspective

The Restaurant Management System (RMS) is a new, self-contained web application developed as a student project for a Software Engineering course. It is not part of a larger system nor a replacement for an existing system. The RMS consists of two main components:

1. Customer Website – for online food ordering.
2. Staff Dashboard – for order management and reporting.

2.2 Product Functions

The major functions of the RMS include:

- Customer Functions:
  - Browse restaurant menu with images and descriptions.
  - Add items to a cart and customize orders.
  - Place orders online with delivery information.
  - View order status.
- Staff Functions:
  - View incoming orders.
  - Update order status.

- System Functions:
    - Provide menu recommendations using simple machine learning.
    - Store and retrieve data from a local MySQL database.

2.3 User Classes and Characteristics

- **Customers:**
    - General public with basic web browsing skills.
    - Primary goal: order food.
    - No technical expertise required.
- **Restaurant Staff (Kitchen/Manager):**
    - Employees who manage orders and update statuses.
    - May have varying levels of computer familiarity.
    - Access restricted via login credentials.
- **System Administrators (Developers):**
    - Student team members who maintain and deploy the system.
    - Technical expertise in web development and database management.

2.4 Operating Environment

- Frontend Environment : HTML, CSS, JavaScript.
- Compatible with Chrome web browser.
- Responsive design for desktop.
- Backend Environment: Python (Flask)
- Database Environment: MySQL

2.5 Design and Implementation Constraints

Technology Stack Constraints:

- Must use HTML, CSS, and JavaScript for frontend development
- Must use Python for backend logic
- Must use MySQL for database management

Development Constraints:

- Project must be completed within academic semester timeline
- Team of six developers with varying skill levels

Design Constraints:

- Must be fully responsive
- Must comply with basic web accessibility standards

Integration Constraints:

- Database schema must support both customer and staff data models

2.6 User Documentation

For Customers: A clean 'FAQ section' on the website explaining how to:

- Browse the menu and view item details.
- Add items to the cart and place an order.
- A short 'How to Order' guide with illustrations.

For Restaurant Staff: A 'Staff Quick-Start Guide' PDF covering:

- How to log into the dashboard.
- How to view new orders and update their status.

2.7 Assumptions and Dependencies

Assumptions:

- Customers have basic web browsing knowledge
- Restaurant has existing menu items with descriptions and images
- Restaurant operates during standard business hours

Dependencies:

- MySQL Database Server: System cannot function without database
- Python Environment: Backend requires Python with necessary libraries
- Web Browsers: System designed for current browser versions
- Team Expertise: Project success depends on team's ability to learn and implement required technologies

3. External Interface Requirements

3.1 User Interfaces

The user interface will be implemented using website interface with basic pages: Home, Menu, and Staff Dashboard. The interface will be user friendly so it can be used easily by customers and staff. Each page will contain buttons, text inputs, and tables to display information.

3.1.1 UI Screen List

Customer Facing Screens:

1. **Landing Page (Home):** Hero image, call-to-action, link to menu.
2. **Menu Page:** Categorized grid/listing of menu items with images, names, prices, and "Add to Cart" buttons.

3. **Shopping Cart Page:** List of selected items, quantities, prices, total, and a "Proceed to Checkout" button.
4. **Checkout Page:** Form for delivery details (name, address, phone) and final order review with a "Place Order" button.
5. **Order Confirmation Page:** Displays order number, summary, and estimated time.

Staff Facing Screens:

6. **Staff Login Page:** Form for username and password.
7. **Dashboard Main Page:** Overview showing counts of new/pending/completed orders.
8. **Order Management Page:** Detailed list of orders with filters and status update dropdowns.
9. **Reports Page:** Interface to select date range and view sales/popular items report, with an export option.

3.2 Hardware Interfaces

The system will run on standard laptops or PCs used by the development team. No special hardware is required; only a keyboard, mouse, and screen are needed.

3.3 Software Interfaces

- MySQL will be used as the local database.
- HTML, CSS, JavaScript for the frontend website.
- Python (Flask) for the backend logic.
- The system will be tested on Chrome web browser.

3.4 Communications Interfaces

- The website will connect to the local MySQL database using SQL queries.
- No external APIs or internet-based services will be used.

4. System Features

4.1 Online ordering

4.1.1 Description and Priority

This feature enables customers to browse the restaurant menu, select items, customize orders, add to cart, and place orders online through a simple website.

Priority: High

4.1.2 Stimulus/Response Sequences

Step	Actor Action	System Response
1	Customer visits site	Shows homepage with menu link
2	Clicks “Menu”	Displays categorized menu with images & prices
3	Clicks “Add to Cart”	Adds item to cart, updates cart icon
4	Clicks “Checkout”	Shows delivery form & order summary
5	Submits order	Saves order, shows confirmation, clears cart

4.1.3 Functional Requirements

REQ-1: The system shall display all menu items with name, image, description, and price.

REQ-2: The system shall allow customers to adjust item quantities in the cart.

REQ-3: The system shall calculate and display order total price.

REQ-4: The system shall show “Item unavailable” if selected item is out of stock.

4.2 Order Management Dashboard

4.2.1 Description and Priority

This feature provides restaurant staff with a secure web dashboard to view incoming orders, update statuses, and manage order flow.

Priority: High

4.2.2 Stimulus/Response Sequences

Step	Actor Action	System Response
1	Staff logs in	Authenticates and loads dashboard
2	Views “New Orders”	Lists pending orders with details
3	Selects an order	Shows full order info (items, notes, customer)
4	Changes status (e.g., to “Ready”)	Updates database and reflects change in real time
5	Filters by date/status	Displays filtered order list

4.2.3 Functional Requirements

REQ-5: The system shall require login with username/password for staff access.

REQ-6: The system shall allow staff to update order status via dropdown.

REQ-7: The system shall prevent editing of completed/cancelled orders.

REQ-8: The system shall show error if database update fails.

4.3 Menu Recommendations

4.3.1 Description and Priority
This feature suggests menu items to customers based on order history, using a simple machine learning model for personalization.
Priority: Medium

4.3.2 Stimulus/Response Sequences

Step	Actor Action	System Response
1	Customer views menu	Loads past order data
2	System analyzes data	Generates 3–5 recommended items
3	Displays “Recommended” section	Shows items with “Try This” label
4	Customer clicks recommendation	Navigates to item page

4.3.3 Functional Requirements

REQ-9: The system shall collect anonymous order history for recommendations.

REQ-10: The system shall use a Scikit-learn model (or rule-based fallback) for suggestions.

REQ-11: The system shall display recommendations only if sufficient data exists.

REQ-12: The system shall allow staff to disable recommendations via dashboard.

REQ-13: The system shall default to popular items if ML model fails.

4.4 Basic Reporting

4.4.1 Description and Priority
This feature allows managers to view simple sales reports, such as daily totals and popular items, for business insight.
Priority**:** Low

4.4.2 Stimulus/Response Sequences

Step	Actor Action	System Response
1	Manager goes to Reports	Shows report options (sales, top items, dates)
2	Selects report type & date range	Queries database and generates report
3	Clicks “Export”	Downloads report as CSV

4.4.3 Functional Requirements

REQ-14: The system shall restrict reports to manager-level users.

REQ-15: The system shall list top 5 ordered items for selected period.

REQ-16: The system shall allow CSV export of report data.

REQ-17: The system shall show “No data” if no orders in selected range.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system should respond quickly when many users use it at the same time. Menu items and orders should load without delay. Order total and offers should be calculated. sThe database should handle many orders at the same time.

5.2 Safety Requirements

- The system shall not allow any user (customer or staff) to access or modify another user's personal data without proper authorization.
- Staff shall only be able to update an order's status forward (e.g., Pending -> Preparing -> Ready), not revert it, to maintain audit integrity.

5.3 Security Requirements

- User passwords should be stored securely.
- Staff passwords should be stored securely.

5.4 Software Quality Attributes

- The system should be reliable and work without frequent errors.

- The system should be easy to maintain and update.
- The system should support future expansion.

5.5 Business Rules

- Each order belongs to one user.
- Each order must have one payment.
- Only admin users can manage menu items and offers.
- Only users with the "Manager" role can access the "Basic Reporting" feature.

5.6 Constraints & Limitations

- The initial release is a **local deployment only**; it is not hosted on a public web server.
- Payment processing is **simulated**; no real financial transactions are handled.
- The recommendation engine is based on simple aggregate data; it is not a real-time, personalized AI system.

5.7 External & Organizational Requirements

- The system shall be developed by a team of 6 students using the prescribed tech stack (HTML/CSS/JS, Python, MySQL).
- All documentation and code shall be submitted by the final project deadline

6. Other Requirements

6.1 Database Requirements

- The system shall use MySQL as the relational database.
- Database tables shall include: Users, MenuItems, Orders, OrderItems, Staff.
- Data backups are not required for the student version.

6.2 Development & Deployment Requirements

- The system shall be developed and tested locally (no hosting required).
- Code shall be version-controlled using Git.
- Documentation shall include a basic README file with setup instructions.

6.3 Legal & Compliance Requirements

- User data shall not be shared externally.
- The system is for academic use only; no real payments or sensitive data are processed.

6.4 Reuse & Future Extension

- The code shall be modular to allow future features (e.g., payment integration, SMS notifications).
- The frontend shall be built with reusable components for easy maintenance.

Appendix A: Glossary

Term	Definition
RMS	Restaurant Management System
Frontend	User interface built with HTML, CSS, JavaScript
Backend	Server-side logic built with Python (Flask)
ML	Machine Learning (Scikit-learn) used for recommendations
MySQL	Open-source relational database used for data storage
Dashboard	Web interface for restaurant staff to manage orders
Order Status	State of an order: Pending, Preparing, Ready, Delivered, Cancelled
Cart	Temporary storage for selected menu items before checkout

Appendix B: Analysis Models

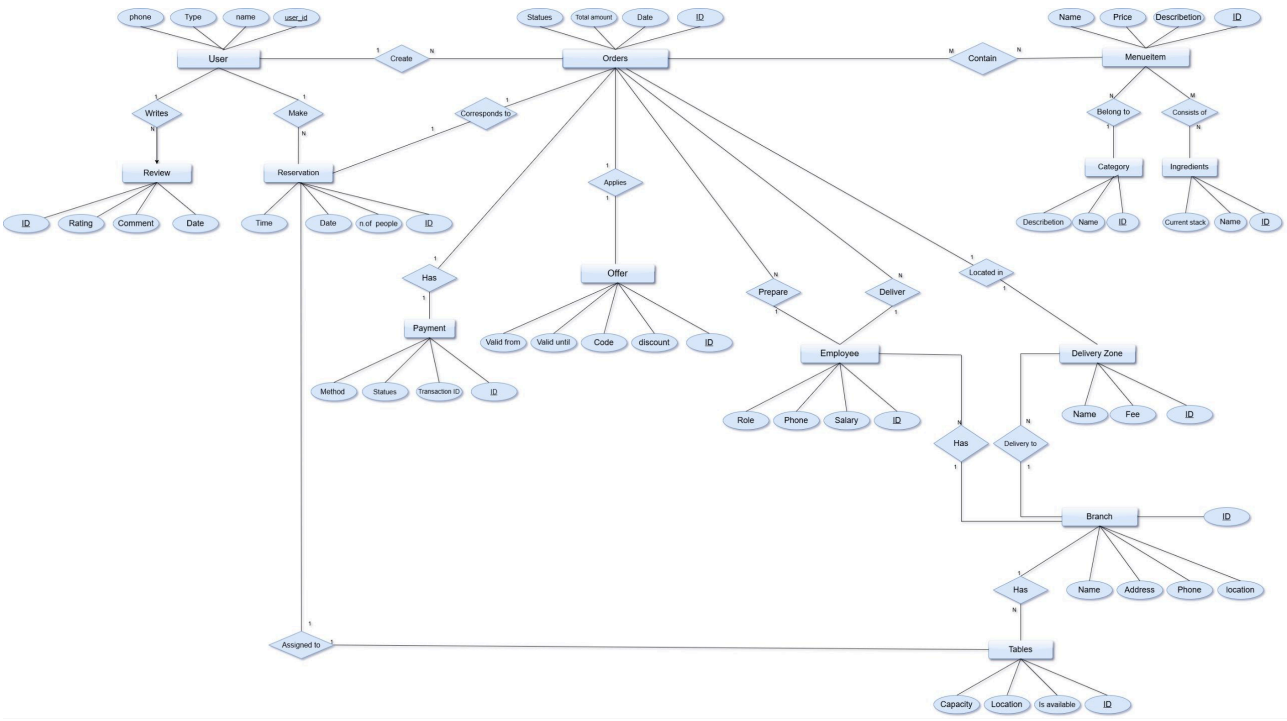
B.1 Use Case Description

- **UC-1: Place Order**
  - **Actor:** Customer

- **Description:** Customer browses the menu, selects items, and completes an order for delivery.

- **UC-2: Update Order Status**
  - **Actor:** Kitchen Staff
  - **Description:** Staff views new orders and changes their status (e.g., to "Preparing" or "Ready").
- **UC-3: Generate Sales Report**
  - **Actor:** Manager
  - **Description:** Manager selects a date range and views a summary of sales and popular items.
- **UC-4: Manage Menu (Future)**
  - **Actor:** Manager
  - **Description:** Manager adds, removes, or edits items in the restaurant's digital menu.

B.2 ERD



Appendix C: To Be Determined List

TBD-1: Exact colour scheme and logo for the website.

TBD-2: Sample dataset for ML recommendations.

TBD-3: Final list of menu items and categories.

TBD-4: Decision on whether to implement user profiles for customers.

TBD-5: Whether to include order cancellation feature.

Appendix D: Project Risks

1. **Technical Risk - Machine Learning Integration:** Team has limited experience with Scikit-learn. The recommendation feature may be simplistic or buggy.
    - **Mitigation:** Start with a very simple rule-based system (e.g., "most ordered items") as a fallback. Allocate time for team learning.
2. **Schedule Risk - Overambitious Scope:** Attempting to implement all features at once could lead to missing the deadline.
    - **Mitigation:** Adhere strictly to the Incremental model. Define a "Minimum Viable Product" (MVP) core for the first deliverable.
3. **Technical Risk - Database Design Flaws:** A poorly designed schema could make the application slow or hard to extend.
    - **Mitigation:** Review the ERD with the professor/TA early. Use normalization principles.
4. **Resource Risk - Skill Variation:** Team members have different proficiency levels in web development.
    - **Mitigation:** Pair programming, peer reviews, and sharing of learning resources. Clear task allocation based on skill growth goals.
5. **Operational Risk - Last-Minute Integration Issues:** The frontend and backend, developed somewhat separately, may not integrate smoothly at the end.
    - **Mitigation:** Define clear API/interaction contracts early. Integrate small components continuously, not all at once at the deadline.