SOFTWARE REQUIREMENTS SPECIFICATION

for

Restaurant Management System

Version 1.0

1. **IMT2022003 Ritish Shrirao**
2. **IMT2022076 Mohit Naik**
3. **IMT2022086 Ananthakrishna K**
4. **IMT2022103 Anurag Ramaswamy**

November 7, 2024

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to define the requirements for a Command-Line Interface (CLI) application for managing the administrative tasks of a restaurant. This Restaurant Management System (RMS) will allow restaurant administrators to manage essential operational functions, including tracking bills, viewing sales and checking order statistics. The system is designed to streamline the administrative workload and provide a comprehensive overview of restaurant operations.

1.2 Document Conventions

- RMS - Restaurant Management System
- CLI - Command Line Interface

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Software Developers: For implementation and reference during development.
- Project Managers and Stakeholders: To understand the scope and functional requirements of the system.
- Quality Assurance Team: For testing and validation of functionalities.
- Restaurant Management Team: For understanding how the system will address their administrative needs.

Suggested reading order is sequential, as each section builds upon the previous ones. Sections 4 and 5 may be of particular interest to technical teams.

1.4 Project Scope

The RMS aims to automate and streamline various restaurant management processes by providing a command-line interface to perform routine administrative functions. This will help management monitor operations more efficiently, order statistics, handle billing and payment information, and access other business-critical data. The CLI will enhance productivity and facilitate data-driven decision-making for restaurant administrators.

2 Overall Description

2.1 Product Perspective

The RMS CLI is designed as an independent module that connects to a centralized database containing relevant restaurant information. It will be accessible from any compatible terminal, and it allows the restaurant admin to view and manage data related to billing, orders, and employee management. The system is modular, allowing for future updates or extensions of functionality as required.

2.2 Product Functions

Key functionalities of the RMS CLI include:

- Billing Management: View, edit and archive bill details for customer transactions.
- Employee Management: View employee performance statistics, assign roles, and manage schedules.
- Sales and Revenue Reports: Access daily, weekly, and monthly revenue reports.
- Menu Management: Update menu items, adjust prices, and track popularity of items.
- Customer Feedback Review: View and categorize feedback for quality control.

2.3 User Classes and Characteristics

- **Admin:**
  Admins have full access to manage the restaurant's operations, keeping everything running smoothly and efficiently. Here's what they're able to handle:
  - **Billing:** They can view, edit, and archive billing details to keep accurate transaction records.
  - **Employee Management:** Admins can monitor employee performance, assign roles, and create or adjust work schedules.
  - **Sales & Revenue Reports:** With access to daily, weekly, and monthly revenue reports, they stay on top of the restaurant's financial health.
  - **Menu Management:** Admins can update the menu, adjust prices, and keep an eye on which items are most popular with customers.

  - **Customer Feedback:** They can review customer feedback, helping to identify areas for improvement and ensure quality control.

- **Restaurant Employees** Employees focus on billing-related tasks, keeping transactions running smoothly:
  - **Billing:** Employees can create and view bills for customers, making sure the checkout process is efficient and accurate.

All users are expected to have a basic understanding of CLI operations and the administrative tasks involved in restaurant management.

2.4 Operating Environment

The RMS CLI will operate on:

- Platform: Any OS with a compatible terminal environment (Linux, macOS, or Windows).
- Database: SQL databases (MySQL, SQLite) for data storage
- Dependencies: Java and required libraries for CLI functionality and database connectivity.

2.5 Design and Implementation Constraints

The CLI-based restaurant management system adheres to the following constraints:

- **Platform Compatibility:** Lightweight and compatible across Windows, macOS, and Linux.
- **Role-Based Access Control:**
  - **Employees:** Limited to billing functions.
  - **Admins:** Full access to billing, employee management, menu updates, reports, and feedback.
- **Usability:** Simple commands, clear prompts, and error handling for a smooth user experience.
- **Maintenance:** Minimal requirements with straightforward updates and documentation for each function.

2.6 User Documentation

- A readme document

2.7 Assumptions and Dependencies

- **System Assumptions:**
  - The system assumes that users have basic familiarity with CLI commands.
  - The restaurant will use the software on a standalone system without network-based dependencies.
  - The database is assumed to be correctly set up and accessible on the local machine.
- **System Dependencies:**
  - The system depends on the presence of an operating system that supports command-line interfaces (Windows, macOS, or Linux).
  - Database must be installed and functional for data storage and retrieval.
  - Password hashing libraries or tools are required for secure authentication (if implemented).

3 External Interface Requirements

3.1 User Interfaces

- Simple command-line interface (CLI) for direct, text-based interaction.
- Role-specific commands:
  - **Employees:** Access to billing commands.
  - **Admins:** Access to all commands, including employee and menu management.
- Clear user prompts, feedback messages, and error handling to support ease of use.

3.2 Hardware Interfaces

- Compatible with basic hardware on Windows, macOS, and Linux.
- Requires only a keyboard for text-based input.

3.3 Software Interfaces

- Uses SQL for local data storage (billing, employee data, menu).
- Runs on Windows, macOS, and Linux command-line environments.

3.4 Communication Interfaces

- Standalone, with no internet or network dependency; all operations are local.

4 System Features

4.1 Description and Priority

All features are high priority, as they constitute the core functionalities required for effective restaurant management. Some features may be implemented in phases based on their complexity and immediate need.

4.2 Functional Requirements

- View and Manage Bills: Retrieve and generate bills for orders.
- Employee Statistics: View and manage employees.
- Revenue Analysis: Generate sales reports based on specified time periods.
- Customer Feedback Access: Store and categorize feedback for quick review.
- Menu Item Updates: Add, remove, or modify menu items.
- Access Control: Add or remove users, assign permissions, and track user activity logs.

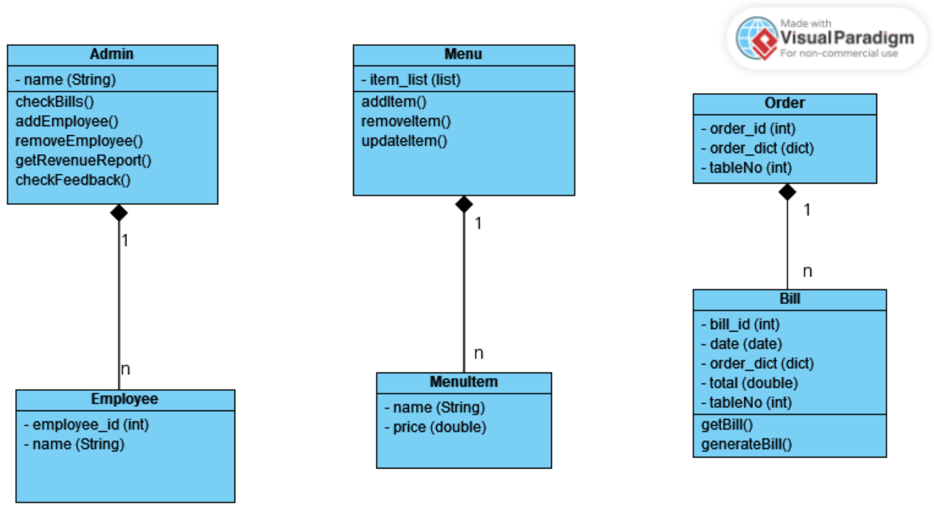

Figure 4.1: Class diagram

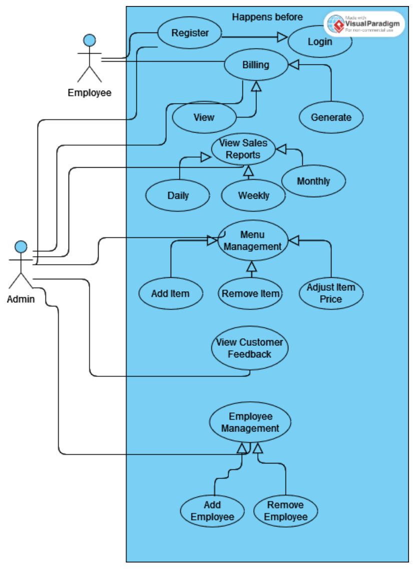

Figure 4.2: Usecase diagram

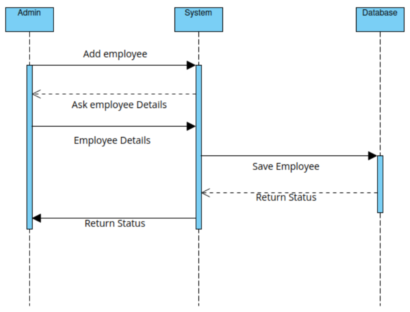

Figure 4.3: Sequence diagram for adding employee

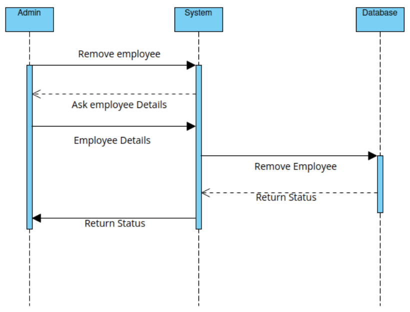

Figure 4.4: Sequence diagram for removing employee

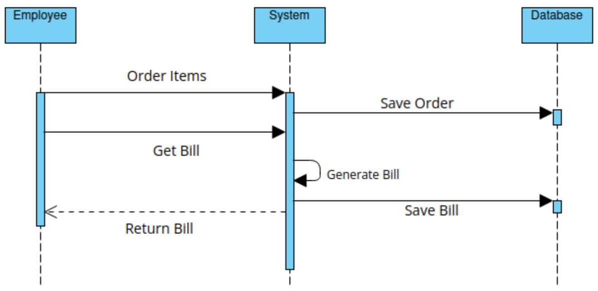

Figure 4.5: Sequence diagram for generating bill

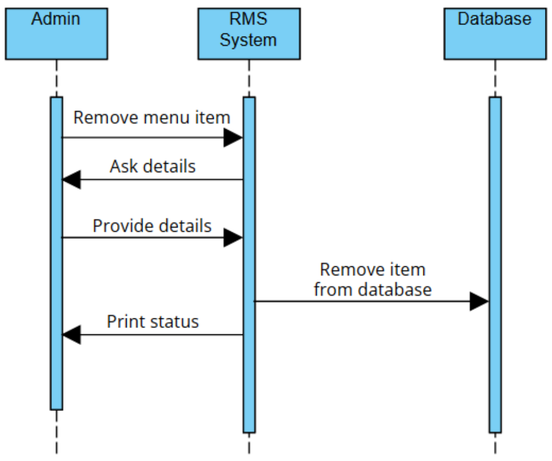

Figure 4.6: Sequence diagram for removing menu item

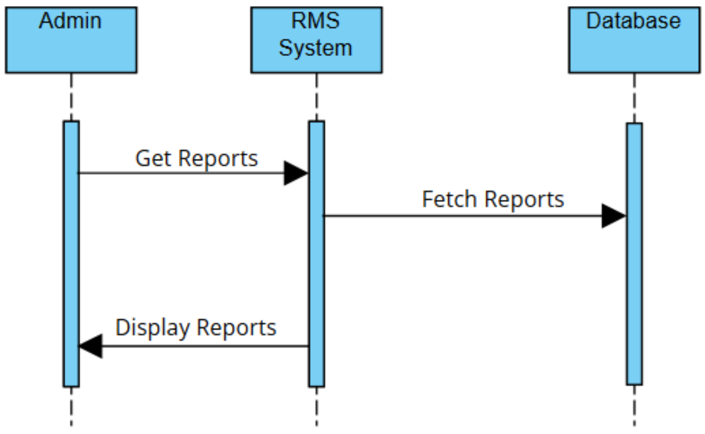

Figure 4.7: Sequence diagram for fetching reports

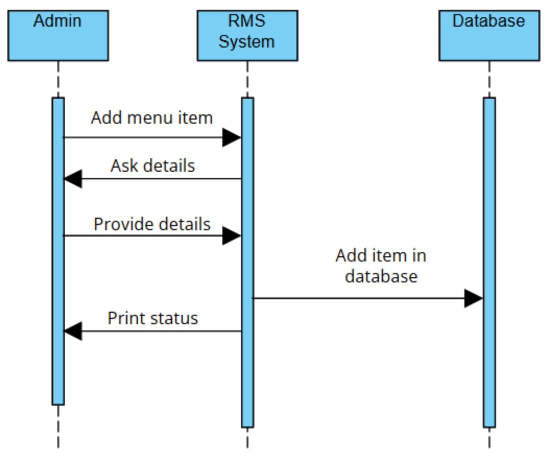

Figure 4.8: Sequence diagram for adding menu item

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- Response Time: System responses should be immediate (less than 2 seconds) for all commands.
- Scalability: The database should handle up to 10,000 records per entity (e.g., bills, orders, employees) without performance degradation.
- Data Consistency: All data updates (billing and orders) must be immediately reflected across the system.

5.2 Security Requirements

- User Authentication: System access must be password-protected, with different access levels for admins and supervisors.
- Data Encryption: Sensitive information (e.g., billing details) must be encrypted at rest.
- Audit Trail: Maintain logs of all actions performed by each user for security auditing.

5.3 Software Quality Attributes

- Reliability: The system should be robust and have a failure rate of less than 1% over extensive testing.
- Usability: The CLI should be intuitive, with clear prompts and command suggestions.
- Maintainability: The modular structure of the codebase should facilitate easy updates and bug fixes.
- Portability: The CLI must run smoothly across Windows, macOS, and Linux environments.

5.4 Business Rules

- Only administrators and authorized personnel should access the system.
- All financial data (bills, expenses) should be available for up to 6 months.
- User permissions must be structured to prevent unauthorized access to sensitive functions.

5.5 Other Requirements

- Backup and Recovery: The system should have support for backups with easy recovery procedures.
- Logging: The CLI should log all transactions and key actions for operational insights and security.
- Error Handling: System should provide clear error messages and suggest corrective actions.