Restaurant Management System

Requirements Specification

Version 2.0

April 3, 2025

**Table of Contents**

[**RESTORANT MANAGEMENT SYSTEM REQUIREMENTS SPECIFICATION 1**](#_heading=h.19c6y18)

[**VERSION 1.0 1**](#_heading=h.3tbugp1)

[**APRIL 19, 2021 1**](#_heading=h.28h4qwu)

[**1.**](#_heading=h.nmf14n) **EXECUTIVE SUMMARY 3**

[1.1](#_heading=h.37m2jsg) Project Overview 3

[1.2](#_heading=h.1mrcu09) Purpose and Scope of this Specification 3

[**2.**](#_heading=h.46r0co2) **PRODUCT/SERVICE DESCRIPTION 3**

[2.1](#_heading=h.2lwamvv) Product Context 3

[2.2](#_heading=h.111kx3o) User Characteristics 3

[2.3](#_heading=h.3l18frh) Assumptions 3

[2.4](#_heading=h.206ipza) Constraints 3

[2.5](#_heading=h.4k668n3) Dependencies 4

[**3.**](#_heading=h.2zbgiuw) **REQUIREMENTS 4**

[3.1](#_heading=h.1egqt2p) Functional Requirements 5

[3.2](#_heading=h.3ygebqi) Non-Functional Requirements 5

[*3.2.1*](#_heading=h.2dlolyb) *Product Requirements 5*

[**3.2.1.1**](#_heading=h.44sinio) **User Interface Requirements** 6

[**3.2.1.2**](#_heading=h.2jxsxqh) **Usability** 6

[**3.2.1.3**](#_heading=h.z337ya) **Efficiency** 6

[3.2.1.3.1](#_heading=h.3j2qqm3) Performance Requirements 6

[3.2.1.3.2](#_heading=h.1y810tw) Space Requirements 6

[**3.2.1.4**](#_heading=h.4i7ojhp) **Dependability** 6

[**3.2.1.5**](#_heading=h.2xcytpi) **Security** 7

[*3.2.2*](#_heading=h.sqyw64) *Organizational Requirements 7*

[**3.2.2.1**](#_heading=h.3whwml4) **Environmental Requirements** 7

[**3.2.2.2**](#_heading=h.2bn6wsx) **Operational Requirements** 7

[**3.2.2.3**](#_heading=h.qsh70q) **Development Requirements** 7

[*3.2.3*](#_heading=h.3cqmetx) *External Requirements 7*

[**3.2.3.1**](#_heading=h.1pxezwc) **Regulatory Requirements** 7

[**3.2.3.2**](#_heading=h.49x2ik5) **Ethical Requirements** 7

[**3.2.3.3**](#_heading=h.2p2csry) **Legislative Requirements** 7

[3.2.3.3.1](#_heading=h.147n2zr) Accounting Requirements 7

[3.2.3.3.2](#_heading=h.3o7alnk) Security Requirements 7

[3.3](#_heading=h.1rvwp1q) Domain Requirements 7

[**4.**](#_heading=h.4bvk7pj) **USER SCENARIOS/USE CASES4. SOFTWARE DESIGN / DIAGRAMS........................................................................23**

**4.1 REQUIREMENTS ANALYSIS .................................................................................................................. 23**

**4.1.1 User Scenarios....................................................................................................................... 23**

**4.1.1.1 User Scenarios List...................................................................................................................... 23**

**4.1.1.2 User Scenarios Extended............................................................................................................ 24**

**4.1.2 User Cases............................................................................................................................. 32**

**4.2 BEHAVIORAL DIAGRAMS..................................................................................................................... 45**

**4.2.1 Use Case Diagrams ............................................................................................................... 45**

**PMS Documentation**

**June 3, 2018 Page 3**

**4.2.2 Activity Diagrams.................................................................................................................. 50 7**

# Executive Summary

## Project Overview

The Restaurant Management System is a web-based application that allows customers to browse the menu, make reservations, and place orders; waiters to manage orders and billing; and managers to oversee restaurant operations, including inventory, menu items, and tables.

## Purpose and Scope of this Specification

This document outlines the requirements for the restaurant management system. It is intended for developers, testers, and stakeholders to align on the system’s goals.

In Scope:

-Role-based features for Customers, Waiters, and Managers

-Menu, ordering, and reservation features

-Payment handling and table management

Out of Scope:

-Kitchen display systems

-Loyalty/reward program

# Product/Service Description

## Product Context

This is a standalone application with a user-friendly interface. It may connect with external payment gateways for card payments.

## User Characteristics

Customers: General users with no technical expertise

Waiters: Basic computer skills

Managers: Intermediate technical skills for dashboard use

## Assumptions

-Users have internet access

-A browser-based interface is sufficient

-Admin roles are predefined in the system

## Constraints

-The app must work on standard modern browsers

-Must support up to 100 concurrent users

-Payment integration must be PCI compliant

## Dependencies

-Third-party payment API

-Database system (e.g., MySQL or PostgreSQL)

-Authentication services

# Requirements

* Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.
* Organize these requirements in a way that works best for your project. See Appendix DAppendix D, Organizing the Requirements for different ways to organize these requirements.
* Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)
* Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and System Interface/Integration, as well as these example priority definitions:

**Priority Definitions**

The following definitions are intended as a guideline to prioritize requirements.

* Priority 1 – The requirement is a “must have” as outlined by policy/law
* Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits
* Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

* A good requirement is:
* Correct
* Unambiguous (all statements have exactly one interpretation)
* Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)
* Consistent
* Ranked for importance and/or stability
* Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)
* Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)
* Does not specify any particular design
* Traceable (cross-reference with source documents and spawned documents).

## Functional Requirements

| **Req#** | **Requirement** | **Priority** |
| --- | --- | --- |
| FR-CUST-01 | Customers shall be able to register and log in. | 1 |
| FR-CUST-02 | Customers shall view the menu without logging in. | 1 |
| FR-CUST-03 | Customers shall place orders and make payments online. | 1 |
| FR-CUST-04 | Customers shall book table reservations. | 2 |
| FR-CUST-05 | Customers shall see a homepage with featured items and announcements. | 3 |
| FR-WAIT-01 | Waiters shall take orders on behalf of customers. | ~~1~~ |
| FR-WAIT-02 | Waiters shall view but not edit table information. | ~~2~~ |
| FR-WAIT-03 | Waiters shall generate bills and process payments. | ~~1~~ |
| FR-MGR-01 | Managers shall manage menu items (add/edit/delete). | ~~1~~ |
| FR-MGR-02 | Managers shall view and update inventory. | ~~2~~ |
| FR-MGR-03 | Managers shall view daily revenue reports. | ~~2~~ |
| FR-MGR-04 | Managers shall edit and delete table information. | ~~1~~ |

## Non-Functional Requirements

### Product Requirements

-User Interface Requirements:

Responsive UI for mobile and desktop

Clear navigation with role-based dashboards

-Usability:

Learnable within 10 minutes for new users

Tooltips and contextual help for forms and dashboards

-Efficiency:

95% of operations must complete within 1 second

Application supports 100 simultaneous users

-Space Requirements:

Max database size: 5 GB for the first year

Application codebase should not exceed 200 MB

-Dependability:

System availability: 99.5%

Uptime tracking and error logging enabled

-Security:

Login via secure protocol (HTTPS)

Role-based access control

Sensitive data encrypted at rest and in transit

#### **User Interface Requirements**

The application shall provide separate dashboards for Customers, Waiters, and Managers with clear visual distinctions.

The interface shall be responsive and function correctly on both desktop and mobile browsers.

Menu screens must include food categories, item names, prices, and images.

Reservation and order forms shall use dropdowns, calendars, and real-time input validation.

Error messages must be clearly displayed next to input fields with user-friendly explanations.

Managers shall have access to summary reports with tabular and chart-based data presentation.

Users should be able to navigate using a sidebar or top navigation bar, with breadcrumb trails on deeper pages.

#### **Usability**

The system shall be easy to use and require no more than 10 minutes to learn basic tasks.

Online documentation shall be provided for each user role and linked within the system.

Help icons should be available next to complex inputs to show task-specific tooltips.

The system shall offer a guided walkthrough on first login per role.

Common tasks like placing orders or managing tables should be no more than 3 clicks away from the dashboard.

#### **Efficiency**

##### Performance Requirements

The system shall support at least 100 concurrent users during peak hours.

95% of transactions (e.g., order submission, reservation confirmation) shall complete in less than 1 second.

The system shall be capable of handling at least 1,000 transactions per day.

Waiters and Managers should not experience more than 1-second delay for inventory or bill generation tasks.

##### Space Requirements

The application shall not exceed 200 MB for the deployed server-side code.

The database shall support up to 5 GB of data storage in the first operational year.

Image uploads for menu items shall be compressed to a maximum of 1 MB per image.

#### **Dependability**

**Availability**

The system shall be available 24/7, with downtime restricted to scheduled maintenance between 2–4 AM local time.

System uptime must be at least 99.5% monthly.

The system must be usable across all geographic areas with a stable internet connection.

Downtime must trigger a system-wide maintenance banner and notify the admin dashboard.

**Reliability**

Mean Time Between Failures (MTBF) must exceed 500 hours.

The system shall allow retry for failed transactions such as payment processing.

Transaction logs shall include time, user, and action for auditing and debugging.

**Monitoring**

All critical services must include logging for errors and warnings.

Admin shall receive email alerts for server crashes or failed login attempts.

**Maintenance**

The codebase shall be modular and documented to support future feature additions.

System logs must be retained for 6 months and accessible via the Manager’s dashboard.

**Integrity**

Order, payment, and reservation data must include checksums or versioning to prevent corruption or overwrites.

Role-based permissions will restrict access to CRUD operations on sensitive data.

#### **Security**

All communication shall be encrypted using HTTPS.

User passwords shall be hashed and stored securely.

The system shall implement role-based access control for Customers, Waiters, and Managers.

Activity logs shall be generated for login attempts, data edits, and admin actions.

External payments shall be processed via a PCI-DSS-compliant payment provider.

Data validation must be enforced both client-side and server-side to prevent injection or spoofing attacks.

### Organizational Requirements

Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc

#### **Environmental Requirements**

The system shall be developed and deployed on cloud-based environments compatible with Linux.

The client interface must run on all major modern browsers (Chrome, Firefox, Safari, Edge).

No specialized hardware shall be required—desktop or tablet with internet access is sufficient.

#### **Operational Requirements**

The system shall support real-time operations (order taking, billing) with a backend API that supports live data interactions.

Daily backups must occur automatically at 3:00 AM and be restorable within 1 hour.

Logs and reports must be downloadable from the Manager dashboard for audit or analysis.

#### **Development Requirements**

The system shall be built using standard web technologies: HTML, CSS, JavaScript (React for frontend), Node.js or Django for backend.

All code must follow industry standard documentation and use Git for version control.

Testing frameworks (e.g., Jest, Selenium) shall be used for automated testing.

### External Requirements

These are factors driven by external entities such as governments, regulators, or industry standards.

#### **Regulatory Requirements**

Payment processes must comply with PCI DSS standards.

The system must support user requests to export or delete their personal data, as mandated by GDPR.

#### **Ethical Requirements**

User data shall only be used for the intended restaurant functionality (e.g., order tracking, reservations).

The system shall not store or use customer data for marketing without explicit consent.

#### **Legislative Requirements**

All system logs, especially financial and administrative actions, must be recorded and retained for at least 6 months.

Sensitive fields such as card numbers, passwords, and personal IDs shall never be stored in plaintext.

##### Accounting Requirements

The system shall produce detailed, itemized receipts for each transaction, with tax information when applicable.

Daily revenue reports must include total sales, breakdown by payment method, and total taxes collected.

##### Security Requirements

Two-Factor Authentication (2FA) must be enabled for Manager logins.

Session expiration shall be enforced after 30 minutes of inactivity.

All endpoints must validate user permissions to prevent unauthorized access.

## Domain Requirements

Each food item belongs to a category (e.g., appetizer, main course, dessert).

A reservation is associated with a table, a customer, a date and time, and a party size.

Waiters serve specific tables; this association may be updated by the Manager.

Inventory must track quantities of raw ingredients used per dish.

Menu items may be marked as temporarily unavailable if inventory is insufficient.

# **User Scenarios/Use Cases**

**4.1.1.1 User Scenarios List**

| **Nr** | **Name** | **Description** |
| --- | --- | --- |
| **US\_01** | User logs in | Users (admin, waiter, customer) log in using username and password. |
| **US\_02** | Change password | Users (admin, waiter, customer) change their password. |
| **US\_03** | Add new customer | Admin adds a new customer to the system. |
| **US\_04** | Edit customer information | Admin or waiter edits existing customer details. |
| **US\_05** | Assign table to customer | Admin assigns a table to a customer who has arrived. |
| **US\_06** | Change table assignment | Admin or waiter changes the table assignment for a customer. |
| **US\_07** | View all customers | Admin views a list of all customers. |
| **US\_08** | Search for a customer | Admin or waiter searches for a specific customer. |
| **US\_09** | View orders by customer | Admin or waiter views a customer’s order history. |
| **US\_10** | Add a new menu item | Admin adds a new item to the restaurant menu. |
| **US\_11** | Update menu item | Admin updates the details of an existing menu item. |
| **US\_12** | Remove menu item | Admin removes a menu item from the system. |
| **US\_13** | Place order for customer | Waiter takes a new order from a customer. |
| **US\_14** | Edit order | Waiter or admin edits a customer's order. |
| **US\_15** | Apply discount to an order | Waiter applies a discount to a customer’s order. |
| **US\_16** | Cancel an order | Waiter or admin cancels a customer's order before delivery. |
| **US\_17** | Record customer feedback | Waiter records feedback from customers about their experience. |
| **US\_18** | Generate end-of-day report | Admin generates a report at the end of the day summarizing the operations. |
| **US\_19** | View daily revenue | Admin views the revenue generated on a specific day. |
| **US\_20** | View popular menu items | Admin views which menu items are the most ordered. |
| **US\_21** | Log out | Admin, waiter, or customer logs out from their account. |

**Table 2. User scenarios list**

**4.1.1.2 Extended User Scenarios**

**1. US\_01 – User logs in**  
a. User chooses their user type: admin, waiter, or customer.  
b. User is redirected to the login page.  
c. User enters their username and password.  
d. User checks the reCaptcha ‘I am not a robot’.  
e. User presses the login button.  
f. If data is correct, the user is redirected to their profile page.  
g. If data is not correct, an error message will be shown, and the user repeats the process from step b.

**2. US\_02 – Change password**  
a. User logs in following the steps in US\_01.  
b. User chooses the “Change Password” menu.  
c. User enters their old password and the new password (twice).  
d. User presses the “Save Changes” button.  
e. If the old password is correct and the new passwords match, a message “Password was changed successfully!” will be shown.  
f. If the old password is incorrect or the new passwords don’t match, the user will be alerted: “Password was not changed. Please try again.”

**3. US\_03 – Add new customer**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Add New Customer”.  
c. Admin fills in the required fields (name, contact, etc.).  
d. Admin double-checks the entered data.  
e. Admin clicks “Add Customer”.  
f. An alert appears: “Are you sure you want to continue?” YES/NO.  
g. If YES, the system validates the data and creates the account. A message “Customer added successfully!” will be shown.  
h. If validation fails, an error message will appear, and the admin will correct the issue.

**4. US\_04 – Edit customer information**  
a. Admin or waiter logs in following the steps in US\_01.  
b. Admin or waiter selects the customer to edit from the list.  
c. Admin or waiter edits the customer information (name, address, etc.).  
d. Admin or waiter double-checks the entered data.  
e. Admin or waiter clicks on “Save Changes”.  
f. The system validates the data and saves it if everything is correct. A message “Customer information updated successfully!” will be shown.  
g. If the data is incorrect, an error message will appear, prompting corrections.

**5. US\_05 – Delete customer**  
a. Admin logs in following the steps in US\_01.  
b. Admin selects the customer to delete from the list.  
c. Admin clicks on “Delete Customer”.  
d. A confirmation dialog appears: “Are you sure you want to delete this customer?” YES/NO.  
e. If YES, the system deletes the customer. A message “Customer deleted successfully!” will be shown.  
f. If NO, no changes occur, and the admin returns to the customer list.

**6. US\_06 – Add new waiter**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Add New Waiter”.  
c. Admin enters the waiter’s details (name, contact info).  
d. Admin clicks “Add Waiter”.  
e. A confirmation dialog appears: “Are you sure you want to add this waiter?” YES/NO.  
f. If YES, the system validates the data and adds the waiter. A message “Waiter added successfully!” will be shown.  
g. If validation fails, an error message will appear, and the admin will correct the issue.

**7. US\_07 – Edit waiter information**  
a. Admin logs in following the steps in US\_01.  
b. Admin selects the waiter to edit from the list.  
c. Admin edits the waiter’s information (name, shift details, etc.).  
d. Admin clicks “Save Changes”.  
e. The system validates the data and saves it. A message “Waiter information updated successfully!” will be shown.  
f. If there is an error in the data, an error message will appear.

**8. US\_08 – Delete waiter**  
a. Admin logs in following the steps in US\_01.  
b. Admin selects the waiter to delete from the list.  
c. Admin clicks on “Delete Waiter”.  
d. A confirmation dialog appears: “Are you sure you want to delete this waiter?” YES/NO.  
e. If YES, the system deletes the waiter. A message “Waiter deleted successfully!” will be shown.  
f. If NO, no changes occur, and the admin returns to the waiter list.

**9. US\_09 – Customer views menu**  
a. Customer logs in following the steps in US\_01.  
b. Customer clicks on “Menu” in the navigation bar.  
c. Customer views the list of available items, including categories, prices, and descriptions.  
d. Customer selects a category or item to view detailed information.  
e. If desired, the customer adds the item to their cart by clicking “Add to Cart”.

**10. US\_10 – Customer places an order**  
a. Customer logs in following the steps in US\_01.  
b. Customer browses the menu and adds items to their cart.  
c. Customer reviews the cart and presses “Place Order”.  
d. Customer chooses the payment method and enters payment details.  
e. Customer presses “Confirm Order”.  
f. The system processes the payment, sends an order confirmation, and notifies the waiter.

**11. US\_11 – Waiter views orders**  
a. Waiter logs in following the steps in US\_01.  
b. Waiter clicks on “Orders” in the navigation bar.  
c. Waiter sees a list of orders placed by customers, including their status (pending, completed).  
d. Waiter selects an order to view detailed information (items ordered, customer details).  
e. Waiter can mark the order as completed once served.

**12. US\_12 – Waiter marks an order as completed**  
a. Waiter logs in following the steps in US\_01.  
b. Waiter selects an order from the orders list.  
c. Waiter clicks “Mark as Completed”.  
d. The system updates the order status and notifies the customer of completion.

**13. US\_13 – Admin views restaurant performance**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Performance” in the navigation bar.  
c. Admin views reports on sales, orders, and customer satisfaction.  
d. Admin can filter reports by time period or specific items/services.

**14. US\_14 – Admin views employee performance**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Employee Performance” in the navigation bar.  
c. Admin selects the employee to review.  
d. Admin views the employee’s sales, feedback, and work hours.  
e. Admin can filter performance by time or other parameters.

**15. US\_15 – Admin views customer reviews**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Customer Reviews” in the navigation bar.  
c. Admin views a list of customer reviews with ratings and feedback.  
d. Admin can respond to reviews or mark them as addressed.

**16. US\_16 – Customer views order status**  
a. Customer logs in following the steps in US\_01.  
b. Customer clicks on “My Orders” in the navigation bar.  
c. Customer views the list of their past orders, including status updates (pending, completed, etc.).  
d. Customer can click on an order to view detailed information.

**17. US\_17 – Admin generates sales report**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Reports” and selects “Sales Report”.  
c. Admin specifies the time range for the report.  
d. Admin clicks “Generate Report”.  
e. The system processes the request and displays the sales data.

**18. US\_18 – Admin updates menu items**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Menu” and selects an item to update.  
c. Admin modifies the item details (price, description, availability).  
d. Admin clicks “Save Changes”.  
e. The system validates and updates the menu.

**19. US\_19 – Customer requests order modifications**  
a. Customer places an order following US\_10.  
b. Customer decides to modify the order (e.g., change items, add items).  
c. Customer contacts the waiter through the system chat or call option.  
d. Waiter processes the modification, updates the order, and notifies the customer of changes.

**20. US\_20 – Admin manages system settings**  
a. Admin logs in following the steps in US\_01.  
b. Admin clicks on “Settings” in the navigation bar.  
c. Admin can update general system settings (working hours, notification preferences, etc.).  
d. Admin saves changes and the system applies the updates.

**21. US\_21 – Waiter views customer history**  
a. Waiter logs in following the steps in US\_01.  
b. Waiter clicks on “Customer History”.  
c. Waiter selects a customer and views their order history.  
d. Waiter can see details of past orders, preferences, and feedback.

**4.1.2 User Cases**

## UC\_01 – User logs in

|  |  |
| --- | --- |
| **UC Name** | **UC\_01 - User logs in** |
| **Summary** | This use case describes the process where a user (admin, waiter, or customer) logs in using their username and password. |
| **Dependency** | None |
| **Actors** | Primary Actor: User (admin, waiter, customer) Secondary Actors: None |
| **Preconditions** | The user must have an existing account in the system. The user is on the login page. |
| **Description of the Main Sequence** | User chooses their user type (admin, waiter, or customer). User is redirected to the login page. User enters their username and password. User completes the reCaptcha verification. User presses the login button. If the credentials are correct, the user is redirected to their profile page. |
| **Description of the Alternative Sequence** | User enters incorrect credentials. The system shows an error message ("Invalid username or password"). User is prompted to retry the login process. |
| **Non-functional Requirements** | Performance: The login process should be completed within 2 seconds. Security: The system should ensure encrypted transmission of username and password using SSL/TLS. Availability: The login page must be available 99.9% of the time. |
| **Postconditions** | If the login is successful, the user is directed to their respective profile page. If login fails, the user remains on the login page and receives an error message. |

## UC\_02 – Change password

|  |  |
| --- | --- |
| **UC Name** | **UC\_02 - Change password** |
| **Summary** | This use case allows a user to change their account password. |
| **Dependency** | UC\_01 |
| **Actors** | Primary Actor: User (admin, waiter, customer) Secondary Actors: None |
| **Preconditions** | User is logged in. User is on the profile or settings page. |
| **Description of the Main Sequence** | User clicks "Change Password". User enters old password. User enters new password and confirms it. System validates input. Password is updated successfully. |
| **Description of the Alternative Sequence** | Old password is incorrect. System shows error. User retries. |
| **Non-functional Requirements** | Security: Password must be hashed. Usability: Password criteria are shown clearly. Responsiveness: Confirmation within 2 seconds. |
| **Postconditions** | Password is updated. User uses new password for future logins. |

## UC\_03 - Add new customer

|  |  |
| --- | --- |
| **UC Name** | **UC\_03 - Add new customer** |
| **Summary** | Admin adds a new customer to the system. |
| **Dependency** | None |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. Admin is on the customer management panel. |
| **Description of the Main Sequence** | Admin clicks "Add Customer". Admin enters required details. System validates and saves information. Confirmation is shown. |
| **Description of the Alternative Sequence** | Required fields are missing. Validation fails. Admin is prompted to correct data. |
| **Non-functional Requirements** | Data Integrity: All inputs must be validated. Responsiveness: Customer is added within 2 seconds. |
| **Postconditions** | New customer is added to the database. |

## UC\_04- Edit customer information

|  |  |
| --- | --- |
| **UC Name** | **UC\_04 - Edit customer information** |
| **Summary** | Admin or waiter edits existing customer details. |
| **Dependency** | UC\_03 |
| **Actors** | Primary Actor: Admin / Waiter Secondary Actors: None |
| **Preconditions** | User is logged in. Customer record exists. |
| **Description of the Main Sequence** | Admin/waiter searches for customer. Selects the customer from list. Edits required information. System validates and updates data. Success message shown. |
| **Description of the Alternative Sequence** | Customer not found. Validation fails. No changes saved. |
| **Non-functional Requirements** | Auditability: Changes should be logged. Validation: Fields must meet data rules. |
| **Postconditions** | Customer record is updated in the system. |

***UC\_05 - Assign table to customer***

|  |  |
| --- | --- |
| **UC Name** | **UC\_05 - Assign table to customer** |
| **Summary** | Admin assigns a table to a customer who has arrived. |
| **Dependency** | UC\_03 |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. Customer is registered and present in the system. |
| **Description of the Main Sequence** | Admin selects customer. Admin clicks "Assign Table". Admin selects available table. System updates table status. Confirmation is shown. |
| **Description of the Alternative Sequence** | No tables available. Admin notified to wait or choose later. |
| **Non-functional Requirements** | Real-time updates on table status. Responsiveness within 2 seconds. |
| **Postconditions** | Table is marked as occupied and linked to customer. |

***UC\_06 - Change table assignment***

|  |  |
| --- | --- |
| **UC Name** | **UC\_06 - Change table assignment** |
| **Summary** | Admin or waiter changes the table assignment for a customer. |
| **Dependency** | UC\_05 |
| **Actors** | Primary Actor: Admin / Waiter Secondary Actors: None |
| **Preconditions** | User is logged in. Customer already has a table assigned. |
| **Description of the Main Sequence** | User selects customer. Clicks “Change Table”. Selects new table. System updates the table record. Confirmation shown. |
| **Description of the Alternative Sequence** | New table unavailable. System shows warning. Admin selects another table. |
| **Non-functional Requirements** | Table update should reflect in real-time. System should prevent conflicts. |
| **Postconditions** | Customer is moved to new table. Old table becomes available. |

***UC\_07 – View all customers***

|  |  |
| --- | --- |
| **UC Name** | **UC\_07 - View all customers** |
| **Summary** | Admin views a list of all customers. |
| **Dependency** | UC\_03 |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. |
| **Description of the Main Sequence** | Admin clicks “View Customers”. System retrieves customer data. List is displayed with search and sort options. |
| **Description of the Alternative Sequence** | No customers found. System shows empty list or message. |
| **Non-functional Requirements** | Must support pagination for large data. Search must be responsive. |
| **Postconditions** | Admin has access to full customer list. |

***UC\_08 – Search for a customer***

|  |  |
| --- | --- |
| **UC Name** | **UC\_08 - Search for a customer** |
| **Summary** | Admin or waiter searches for a specific customer. |
| **Dependency** | UC\_07 |
| **Actors** | Primary Actor: Admin / Waiter Secondary Actors: None |
| **Preconditions** | User is logged in. Customers exist in the database. |
| **Description of the Main Sequence** | User enters search criteria. System filters customer list. Matching results shown instantly. |
| **Description of the Alternative Sequence** | No match found. System displays “No results”. |
| **Non-functional Requirements** | Search must be case-insensitive. Partial match support. Search within 1 second. |
| **Postconditions** | Matching customer record is displayed. |

***UC\_09 – View orders by customer***

|  |  |
| --- | --- |
| **UC Name** | **UC\_09 - View orders by customer** |
| **Summary** | Admin or waiter views a customer’s order history. |
| **Dependency** | UC\_03 |
| **Actors** | Primary Actor: Admin / Waiter Secondary Actors: None |
| **Preconditions** | User is logged in. Customer exists and has placed orders. |
| **Description of the Main Sequence** | User searches for a customer. Selects “View Orders”. System displays order history with date and status. |
| **Description of the Alternative Sequence** | No orders found. System shows message “No orders available”. |
| **Non-functional Requirements** | Search must be responsive. Results must load under 2 seconds. |
| **Postconditions** | Order list is shown to the user. |

## UC\_10 - Add a new menu item

|  |  |
| --- | --- |
| **UC Name** | **UC\_10 - Add a new menu item** |
| **Summary** | Admin adds a new item to the restaurant menu. |
| **Dependency** | None |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. |
| **Description of the Main Sequence** | Admin navigates to menu management. Clicks “Add New Item”. Fills in item name, category, price, and description. System validates input and saves item. Success message appears. |
| **Description of the Alternative Sequence** | Fields are incomplete or invalid. Admin corrects and retries. |
| **Non-functional Requirements** | Fields must be validated. Item added within 2 seconds. |
| **Postconditions** | New item is added to the menu and visible to customers. |

## UC\_11 - Update menu item

|  |  |
| --- | --- |
| **UC Name** | **UC\_11 - Update menu item** |
| **Summary** | Admin updates the details of an existing menu item. |
| **Dependency** | UC\_10 |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. Item exists in the menu. |
| **Description of the Main Sequence** | Admin searches for item. Clicks “Edit”. Updates desired fields. Saves changes. Confirmation appears. |
| **Description of the Alternative Sequence** | Item not found. System shows error. Admin retries. |
| **Non-functional Requirements** | All fields must support edit. System confirms change within 2 seconds. |
| **Postconditions** | Item details are updated. |

## UC\_12 - Remove menu item

|  |  |
| --- | --- |
| **UC Name** | **UC\_12 - Remove menu item** |
| **Summary** | Admin removes a menu item from the system. |
| **Dependency** | UC\_10 |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. Item exists. |
| **Description of the Main Sequence** | Admin searches for item. Clicks “Delete”. Confirms deletion. System removes item. Confirmation appears. |
| **Description of the Alternative Sequence** | Admin cancels deletion. No changes are made. |
| **Non-functional Requirements** | Deletion is confirmed instantly. Item removed from list and database. |
| **Postconditions** | Item is deleted and no longer visible in menu. |

## UC\_13 - Place order for customer

|  |  |
| --- | --- |
| **UC Name** | **UC\_13 - Place order for customer** |
| **Summary** | Waiter takes a new order from a customer. |
| **Dependency** | UC\_05 |
| **Actors** | Primary Actor: Waiter Secondary Actors: None |
| **Preconditions** | Waiter is logged in. Customer is seated at a table. |
| **Description of the Main Sequence** | Waiter opens active table list. Selects the table/customer. Adds menu items to the order. System stores order in progress. Order sent to kitchen. |
| **Description of the Alternative Sequence** | Menu item unavailable. Waiter informed to remove or replace item. |
| **Non-functional Requirements** | Orders must sync instantly with kitchen. System must auto-save drafts. |
| **Postconditions** | Order is placed and visible in kitchen display. |

## UC\_14 - Edit order

|  |  |
| --- | --- |
| **UC Name** | **UC\_14 - Edit order** |
| **Summary** | Waiter or admin edits a customer's order. |
| **Dependency** | UC\_13 |
| **Actors** | Primary Actor: Waiter / Admin Secondary Actors: None |
| **Preconditions** | Order has already been placed. User is logged in. |
| **Description of the Main Sequence** | User opens current order. Clicks "Edit". Adds/removes menu items. Confirms changes. System updates the order. |
| **Description of the Alternative Sequence** | Item not available. System prevents changes. User chooses alternative. |
| **Non-functional Requirements** | Order changes must reflect in real time. Kitchen must be notified immediately. |
| **Postconditions** | Order is updated and re-sent to kitchen. |

## UC\_15 - Apply discount to an order

|  |  |
| --- | --- |
| **UC Name** | **UC\_15 - Apply discount to an order** |
| **Summary** | Waiter applies a discount to a customer's order. |
| **Dependency** | UC\_13 |
| **Actors** | Primary Actor: Waiter Secondary Actors: Admin (approval if needed) |
| **Preconditions** | Order is placed and unpaid. Waiter is logged in. |
| **Description of the Main Sequence** | Waiter selects order. Clicks “Apply Discount”. Enters discount value. Confirms discount. System recalculates total. |
| **Description of the Alternative Sequence** | Discount exceeds limit. Admin approval requested. If denied, waiter tries again. |
| **Non-functional Requirements** | Discount logic must follow business rules. Discount must be reflected in final bill. |
| **Postconditions** | Order total is updated with discount applied. |

## UC\_16 - Cancel an order

|  |  |
| --- | --- |
| **UC Name** | **UC\_16 - Cancel an order** |
| **Summary** | Waiter or admin cancels a customer's order before delivery. |
| **Dependency** | UC\_13 |
| **Actors** | Primary Actor: Waiter / Admin Secondary Actors: None |
| **Preconditions** | Order must be pending or in process. User is logged in. |
| **Description of the Main Sequence** | User opens the order. Clicks “Cancel”. Confirms cancellation. System removes order from active list. Notification sent to kitchen. |
| **Description of the Alternative Sequence** | Order already prepared. System prevents cancellation. Refund procedure triggered. |
| **Non-functional Requirements** | System must log cancellation reason. Audit trail enabled. |
| **Postconditions** | Order is marked as cancelled and no longer processed. |

## UC\_17 - Record customer feedback

|  |  |
| --- | --- |
| **UC Name** | **UC\_17 - Record customer feedback** |
| **Summary** | Waiter records feedback from customers about their experience. |
| **Dependency** | UC\_13 |
| **Actors** | Primary Actor: Waiter Secondary Actors: None |
| **Preconditions** | Waiter is logged in. Customer has completed an order. |
| **Description of the Main Sequence** | Waiter accesses the customer record. Clicks “Add Feedback”. Inputs customer comments. Saves feedback. System stores it under customer's profile. |
| **Description of the Alternative Sequence** | Feedback is incomplete. Waiter receives warning. Waiter corrects and saves again. |
| **Non-functional Requirements** | Data should be timestamped. Feedback must be editable. |
| **Postconditions** | Feedback is linked to the customer and stored in system. |

## UC\_18 - Generate end-of-day report

|  |  |
| --- | --- |
| **UC Name** | **UC\_18 - Generate end-of-day report** |
| **Summary** | Admin generates a report summarizing the day’s operations. |
| **Dependency** | Multiple UC dependencies (orders, revenue, feedback, etc.) |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. System contains daily data. |
| **Description of the Main Sequence** | Admin opens Reports section. Selects “End-of-Day Report”. Chooses date. System gathers and summarizes data. Report is displayed/downloadable. |
| **Description of the Alternative Sequence** | No data for the date. System shows message “No report available”. |
| **Non-functional Requirements** | Report must generate within 5 seconds. Supports export to PDF. |
| **Postconditions** | Report is generated and accessible to admin. |

## UC\_19 - View daily revenue

|  |  |
| --- | --- |
| **UC Name** | **UC\_19 - View daily revenue** |
| **Summary** | Admin views the revenue generated on a specific day. |
| **Dependency** | UC\_18 |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. System contains revenue data. |
| **Description of the Main Sequence** | Admin goes to Reports. Selects “View Revenue”. Enters date. Revenue data is shown. |
| **Description of the Alternative Sequence** | No revenue recorded. System shows message. |
| **Non-functional Requirements** | Visual summary via chart. Accurate and real-time display. |
| **Postconditions** | Admin views financial summary for the selected day. |

## UC\_20 - View popular menu items

|  |  |
| --- | --- |
| **UC Name** | UC\_20 - View popular menu items |
| **Summary** | Admin views which menu items are the most ordered. |
| **Dependency** | UC\_13 |
| **Actors** | Primary Actor: Admin Secondary Actors: None |
| **Preconditions** | Admin is logged in. System tracks order frequency. |
| **Description of the Main Sequence** | Admin opens statistics. Selects “Popular Items”. System displays items ordered most. Data includes quantities and timespan. |
| **Description of the Alternative Sequence** | No data available. System shows empty chart. |
| **Non-functional Requirements** | Must show top items within 2 seconds. Data should auto-update daily. |
| **Postconditions** | Admin can analyze menu item performance. |

## UC\_21 - Log out

|  |  |
| --- | --- |
| **UC Name** | **UC\_21 - Log out** |
| **Summary** | Admin, waiter, or customer logs out from their account. |
| **Dependency** | UC\_01 |
| **Actors** | Primary Actor: User (admin, waiter, customer) Secondary Actors: None |
| **Preconditions** | User is logged in. |
| **Description of the Main Sequence** | User clicks on “Logout”. System terminates session. Redirects to home/login page. |
| **Description of the Alternative Sequence** | Session expires automatically. System logs user out. |
| **Non-functional Requirements** | Logout must clear session data immediately. Session timeout after inactivity. |
| **Postconditions** | User session is closed and access is revoked. |

APPENDIX

1. **Definitions, Acronyms, and Abbreviations**

RMS: Restaurant Management System

CRUD: Create, Read, Update, Delete

PCI DSS: Payment Card Industry Data Security Standard

2FA: Two-Factor Authentication

MTBF: Mean Time Between Failures

GDPR: General Data Protection Regulation

1. **References**

PCI DSS Guidelines: https://www.pcisecuritystandards.org

Usability Standards: http://www.usabilitynet.org

GDPR Compliance: https://gdpr.eu/

React Documentation: https://reactjs.org/docs/getting-started.html

Django Documentation: <https://docs.djangoproject.com/en/>

1. **Requirements Traceability Matrix**

The following matrix demonstrates how each business requirement (BR) connects to design and testing deliverables. Naming follows the pattern:

FunctionalArea-DocType-Number (e.g., ORD-UCD-01 = Order Use Case Diagram 01).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BizReqID** | **Area** | **ID** | **Deliverable Name** | **Status** |
| BR\_CUST\_01 | ORD | ORD-UCD-01 | Customer Login & Registration Use Case Diagram | Accepted |
| BR\_CUST\_02 | MENU | MENU-PF-01 | Menu Browsing Process Flow | Accepted |
| BR\_WAIT\_01 | ORD | ORD-UCD-02 | Waiter Order Management Use Case Diagram | In review |
| BR\_MGR\_01 | INV | INV-UCD-01 | Inventory and Menu Admin Use Case Diagram | Accepted |
| BR\_MGR\_02 | REP | REP-UI-01 | Revenue Report UI Mockup | ReadyForReview |
| BR\_ALL\_01 | SEC | SEC-TC-01 | Login & Role Access Test Case | ReadyForReview |

1. **Organizing the Requirements**

This section is provided to aid in understanding and preparing detailed requirements. Requirements can be extensive and should be organized in a meaningful and scalable way. The following organization techniques may be considered:

**By System Mode**

Although the Restaurant Management System does not operate in different system modes (e.g., training vs. emergency), modular flows like customer mode, waiter mode, and manager mode can be conceptually considered as modes for organizational clarity.

**By User Class**

This is the primary method used for this project. The system clearly defines three roles:

-Customer: Access to menu, ordering, payment, and reservations

-Waiter: Order-taking, billing, and viewing table

-Manager: Menu, inventory, table, and revenue management

Each role has distinct access and interface requirements.

**By Objects**

Key system objects include:

Menu Item

Reservation

Order

Table

Inventory Item

User (Customer, Waiter, Manager)

Each object supports various CRUD operations and interactions depending on user roles.

**By Feature**

Features such as "Make Reservation", "Process Order", and "Generate Bill" are defined and broken down into functional requirements and use cases for clarity.

**By Stimulus**

Certain system functions respond to direct user actions or external events:

A user clicking "Confirm Order" triggers inventory adjustment and order confirmation

A table becoming available after payment is processed

A manager updating a menu item prompts an update in the live menu feed

**By Response**

Functional areas are also structured around expected outcomes, like:

Viewing a reservation list

Receiving a payment confirmation

Generating a bill receipt

**By Functional Hierarchy**

Functions are grouped under logical modules such as:

User Authentication

Order Management

Reservation Handling

Menu Administration

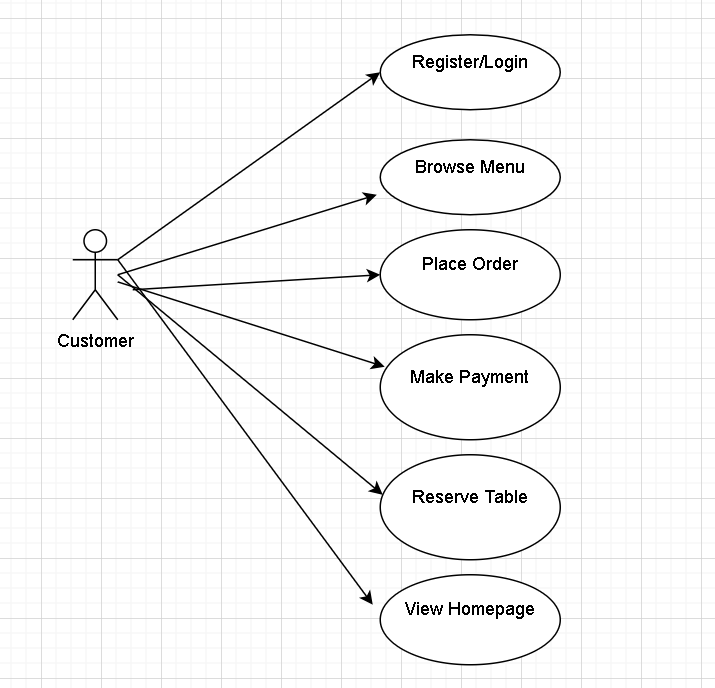
Inventory Control

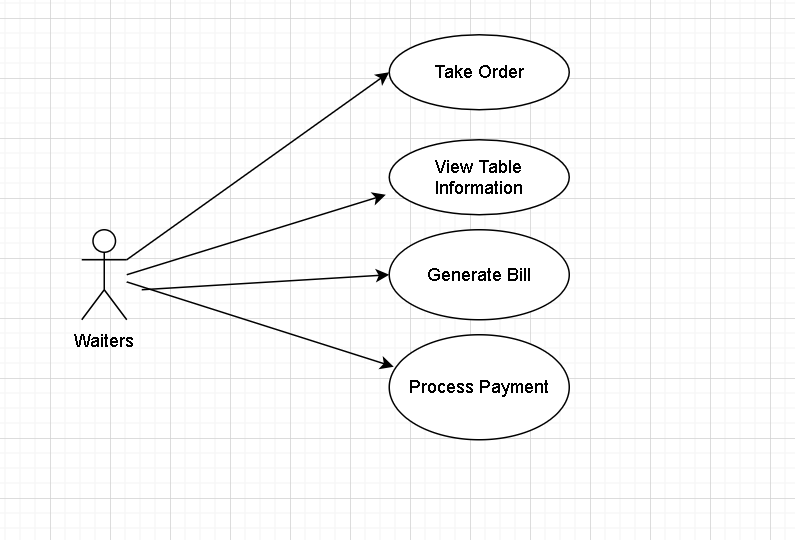
Reporting and Revenue Tracking

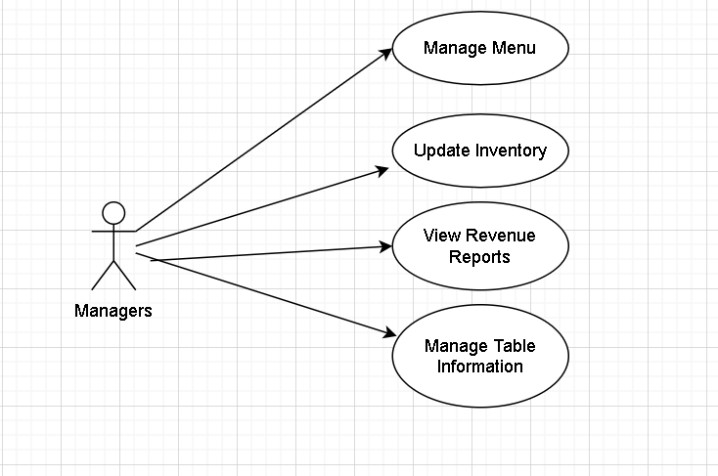
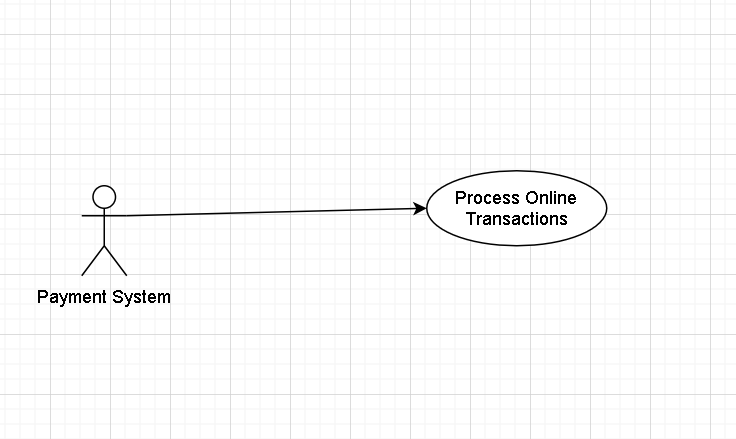
These groupings align with both user roles and system architecture.

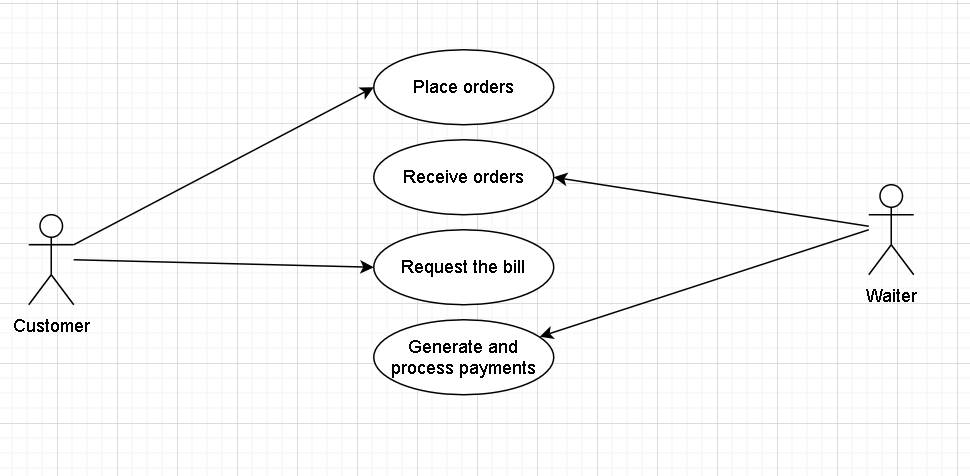
**Additional Comments**

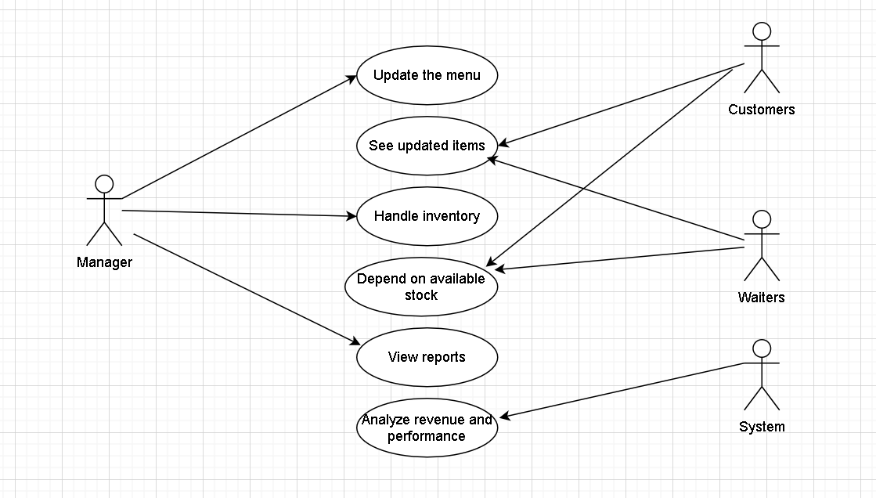
For this project, a combination of User Class, Feature, and Functional Hierarchy was used to organize the requirements effectively. This ensures clarity for developers, testers, and stakeholders, and supports scalability for future extensions (e.g., mobile app, kitchen display system, loyalty programs).





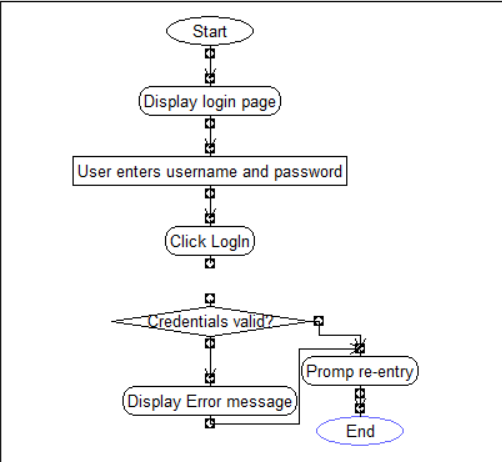
 



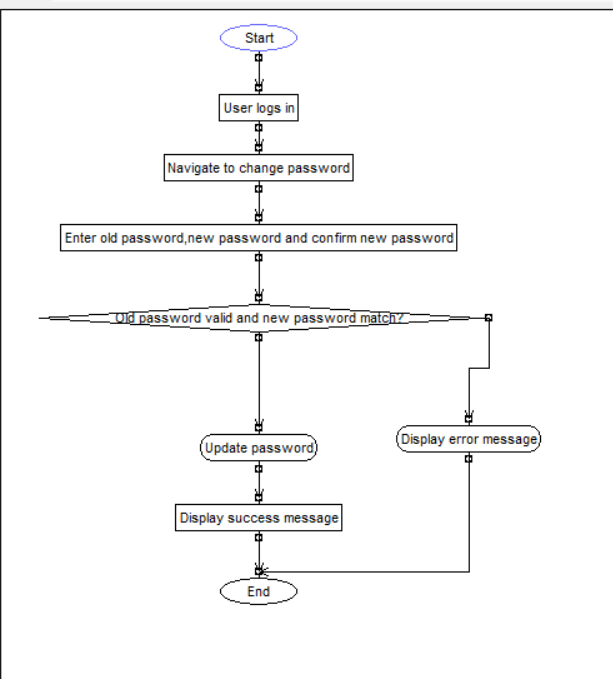


3.2.2 Activity Diagram

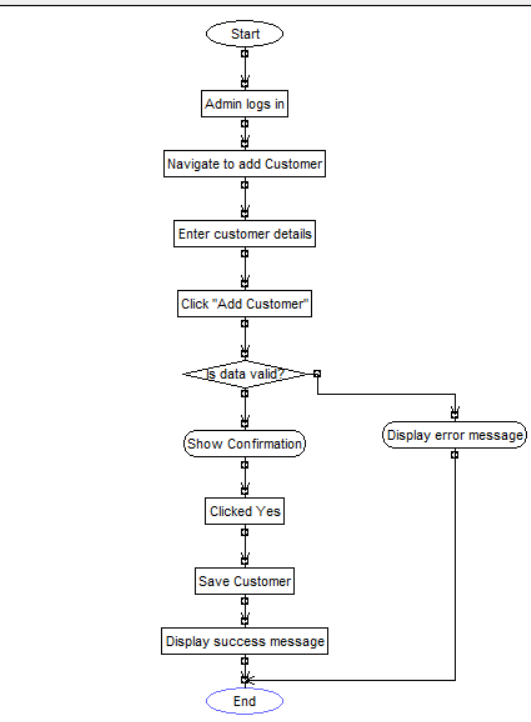
**UC\_1-User logs in**



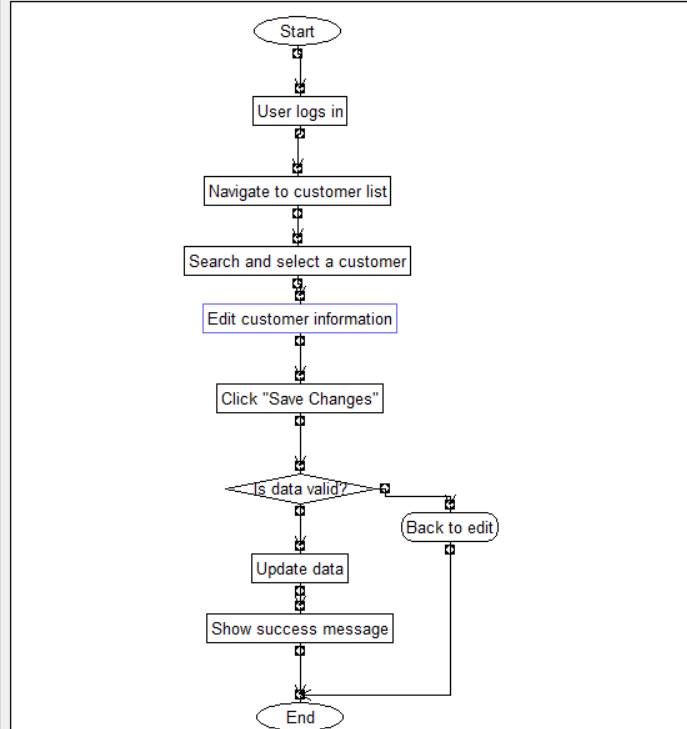
**UC\_2-Change password**



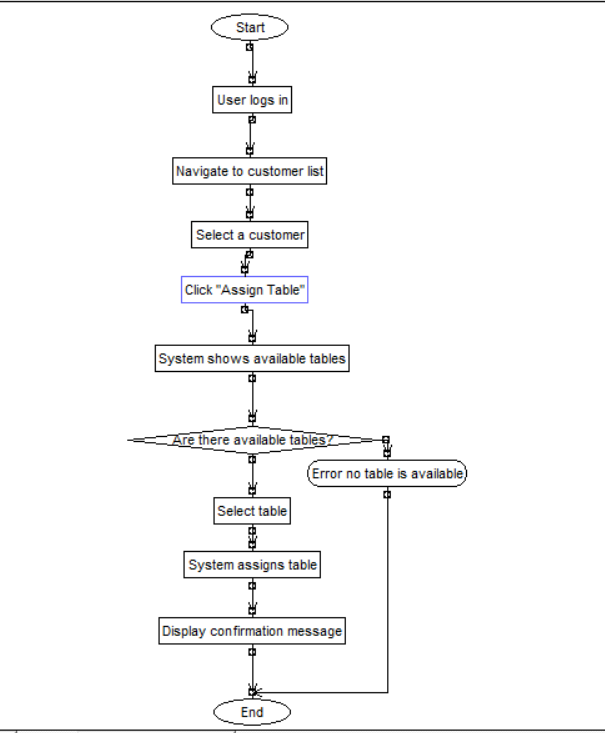
**UC\_3-Add new customer**

****

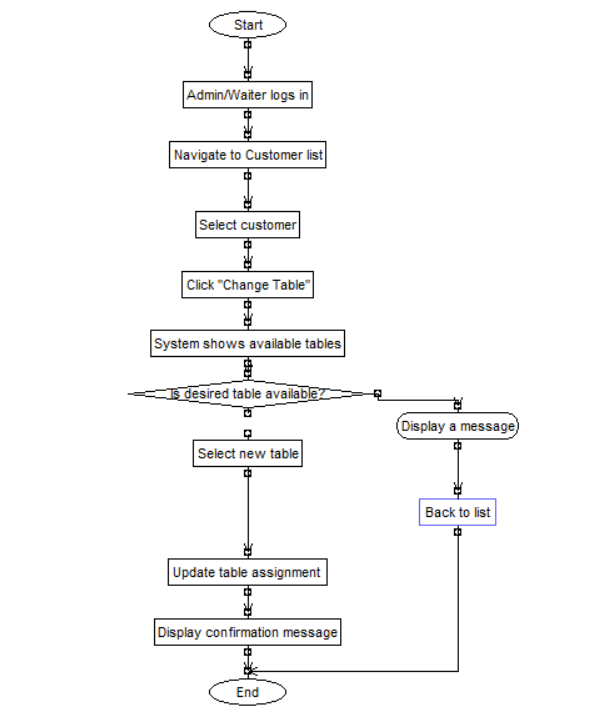
**UC\_4-Edit customer information**



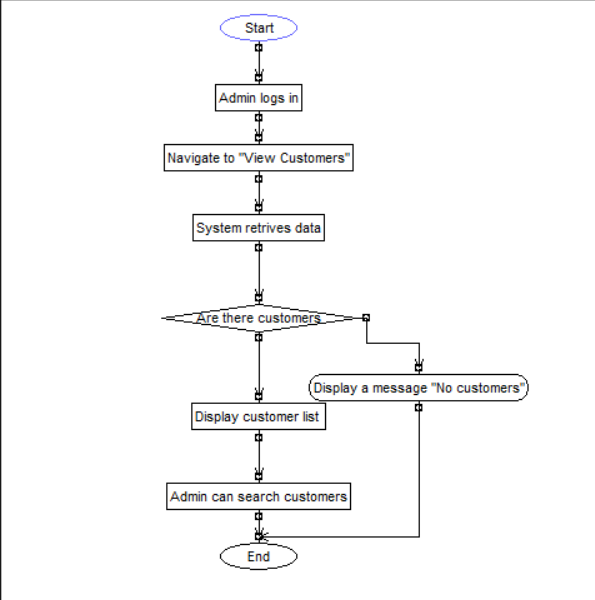
**UC\_5-Assign table to customer**



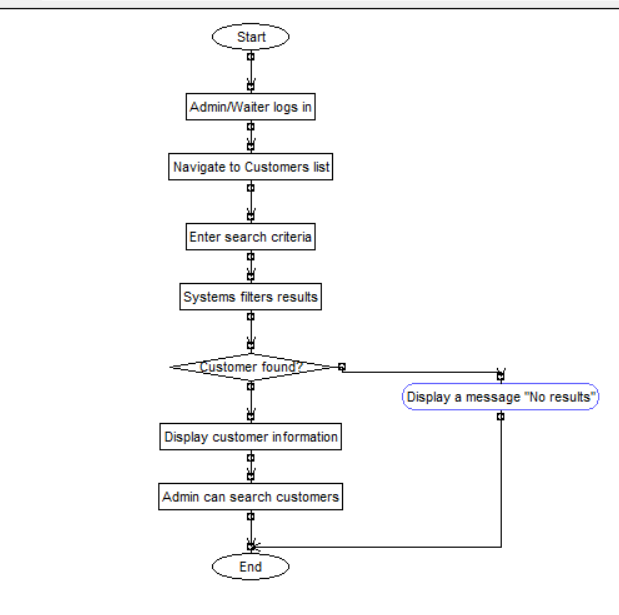
**UC\_6-Change table assignment**

****

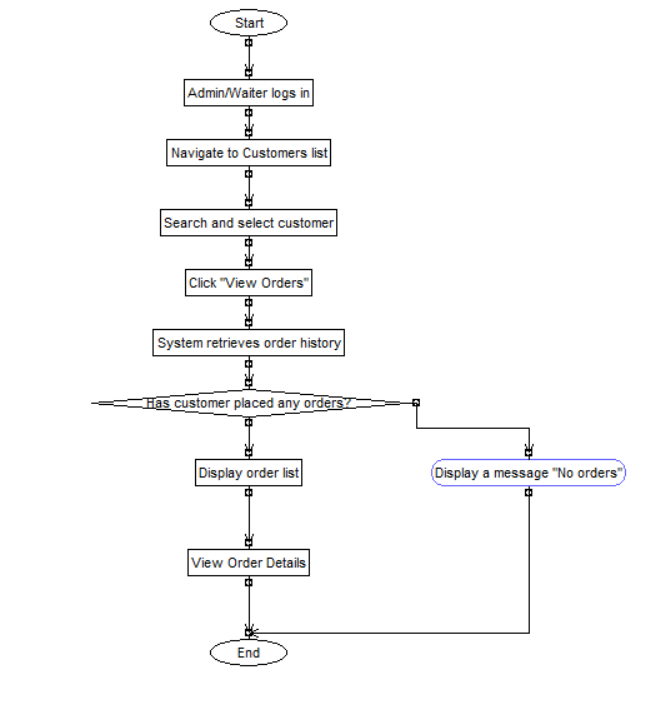
**UC\_7-View all customers**

****

**UC\_8-Search for a Customer**



**UC\_9-View orders by customer.**



**UC\_10-Add a new menu item.**

