SmartPOS: A Web-Based Restaurant Management System

Submitted By
Monir Ullah
Student Id: 212-35-750
Department of Software Engineering
Daffodil International University

Supervised By
Dr. A. H. M. Saifullah Sadi
Professor
Faculty of Science and Information Technology
Department of Software Engineering
Daffodil International University

This project report has been submitted in fulfilment of the requirements for the degree a **Bachelor of Science in Software Engineering**

APPROVAL

This thesis titled on **"SmartPOS: A Web-Based Restaurant Management System"**, submitted by **Monir Ullah (ID: 212-35-750)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Fazla Elahe

Chairman

Dr. Md. Fazla Elahe
Assistant Professor & Associate Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Marzia

Internal Examiner 1

Dr. Marzia Ahmed
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

SM 13/09/2025

Internal Examiner 2

Dr. Shabnom Mustary
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

13/09/25

External Examiner

Mohammod Abul Kashem
Professor
Department of Computer Science and Engineering
Dhaka University of Engineering & Technology, Gazipur.

DECLARATION

I want to state that this project was carried out under the supervision of Dr. A. H. M. Saifullah Sadi, Professor in the Department of Software Engineering at Daffodil International University. I also confirm that this work, in whole or in part, has not been submitted for any other academic qualification.

Submitted by:

Date: 15/9/25

Monir Ullah

Monir Ullah

Student Id: 212-35-750

Department of Software Engineering

Daffodil International University

Certified by:

Date:

15/9/2025

Dr. A. H. M. Saifullah Sadi

Professor

Faculty of Science and Information Technology

Department of Software Engineering

Daffodil International University

ACKNOWLEDGEMENT

I am genuinely thankful to Almighty Allah for giving me the strength and opportunity to complete this project. I want to express my deep appreciation to my supervisor, **Dr. A. H. M. Saifullah Sadi**, Professor in the Department of Software Engineering at Daffodil International University. His constant support, expert guidance, and encouragement have played a key role in shaping the outcome of this work. I am also grateful to everyone who shared their knowledge and offered help during various stages of this project. Their assistance and thoughtful feedback have significantly contributed to its successful completion. Lastly, I would like to extend my gratitude to all the faculty members of the Department of Software Engineering for their continuous motivation and support throughout my academic journey.

Table of Contents

Table of Table

Table of Figure

Chapter 1 : Introduction

This chapter introduces the SmartPOS project, presenting the rationale for developing a web-based restaurant management system. It identifies the limitations of traditional, paper-based operations and emphasizes the growing demand for digital solutions that enhance efficiency, accuracy, and customer satisfaction.

1.1. Background work

A Restaurant Management System (RMS) is essential for streamlining restaurant operations such as order processing, kitchen coordination, billing, and reporting. Many systems already exist in the market, but each comes with certain limitations related to cost, scalability, customization, and real-time processing.

Review of Existing Systems:

1. **Toast POS**

   Strengths: Cloud-based, real-time updates, kitchen display system (KDS), advanced analytics, and delivery integrations.

   Weaknesses: High subscription costs, hardware dependency, limited offline functionality.

2. **Square for Restaurants**

   Strengths: Budget-friendly, easy to use, mobile support, free basic plan.

   Weaknesses: Limited inventory features, no real-time kitchen updates, restricted customization.

3. **Lightspeed Restaurant**

   Strengths: Advanced analytics, multi-location support, inventory reordering, loyalty programs.

   Weaknesses: Expensive, complex setup, not suitable for small restaurants.

4. **TouchBistro**

   Strengths: iPad-based, offline mode, flexible billing, staff scheduling.

   Weaknesses: Limited reporting, hardware restrictions (iPad-only), lacks delivery integrations.

5. **Clover POS**

Strengths: Flexible hardware, loyalty programs, employee scheduling, remote monitoring.

Weaknesses: High setup costs, long-term contracts, limited advanced customization.

How SmartPOS Improves on Existing Systems

- Real-Time Order Updates: Uses WebSockets for instant communication between staff.
- Role-Based Access: Separate dashboards for Admins, Waiters, Kitchen Staff, and Cashiers.
- Scalability: Works efficiently for small cafés, medium restaurants, and large franchises.
- Cost-Effectiveness: Designed to be affordable for SMEs.
- Payment Integration: Supports cash, cards, and digital wallets.
- Inventory Management: Automated low-stock alerts.
- User-Friendly Interface: Simple, responsive design built with React.js and Tailwind CSS .

1.2. Problem Statements

Running a restaurant efficiently is challenging, especially when many establishments still rely on manual processes such as paper-based order taking, handwritten billing, and outdated communication methods. These traditional practices often lead to:

1. **Errors in order-taking –** Waiters may record orders incorrectly, resulting in customer dissatisfaction.
2. **Delays and miscommunication –** Kitchen staff may receive unclear or late instructions, slowing down food preparation.
3. **Customer frustration –** Longer waiting times negatively impact the dining experience and reduce customer retention.
4. **Reduced efficiency and profits –** Inefficiencies in operations directly affect restaurant revenue and overall growth.

1.3. Project Objective

The main goal of this project is to design and develop a web-based Restaurant Management System (SmartPOS) that automates order processing, kitchen coordination, payment handling, and inventory management to enhance restaurant efficiency and customer satisfaction.

The objectives are:

I. To analyze the problems in traditional restaurant management systems.
II. To design a simple and easy-to-use system for managing orders.
III. To implement different user roles for Admins, Waiters, Kitchen Staff, and
IV. Cashiers.
V. To provide sales reports and data insights for business decisions.

1.4. Project Planning and Initiation

In today's restaurant industry, where speed and customer satisfaction are paramount, relying on outdated paper-based systems can hinder operations and lead to frequent errors. These inefficiencies not only compromise service quality but also affect overall profitability. To address these challenges, the SmartPOS system was developed—a web-based Restaurant Management System (RMS) designed to simplify daily tasks through automation and real-time coordination.

SmartPOS serves as a comprehensive platform, enabling restaurant staff, managers, and owners to manage orders efficiently, monitor tables, process billing, and generate reports—all from a single system. With features such as digital menus, real-time order tracking, and kitchen alerts, it significantly minimizes the risk of manual errors and accelerates service delivery. Additionally, the system offers tools for inventory monitoring, sales analytics, and performance tracking, empowering owners to make informed decisions based on precise data.

What sets SmartPOS apart is its adaptability. Whether operating a quaint café or a bustling quick-service restaurant (QSR), the system can scale according to the specific needs of the business. It also supports various payment options, streamlining transactions for customer convenience.

By moving away from traditional methods, SmartPOS effectively bridges the communication gap between floor staff and the kitchen, reduces delays, and fosters a more organized workflow within restaurant operations. Its user-friendly interface ensures that staff can quickly become proficient with the system with minimal training.

Table 1-1: Define the purpose of the POS system

Objective	Description
Enhance Efficiency	Automate order management, billing, and payment processes to reduce manual errors and save time.
Elevate Customer Experience	Provide real-time updates on order status and table availability for a smooth and satisfying dining experience.
Simplify Management	Empower owners and managers with tools for sales monitoring, inventory control, and generating detailed business reports.
Facilitate Scalability	Design the system to meet the operational needs of restaurants of varying sizes, from small cafés to large dining establishments.

Phase 1: Preliminary Analysis & Project Scope Definition

a. Define the purpose of the POS system.

The primary objective of the Restaurant POS (Point of Sale) System is to provide a comprehensive, web-based solution that automates and streamlines essential restaurant operations. The system is designed to:

In summary, the POS system will serve as a central hub, integrating functions such as order processing, table tracking, menu updates, billing, and reporting into a unified, user-friendly platform.

b. Identify Key Stakeholders

Table 1-2: Identify Key Stakeholders

Stakeholder	Role	Key Needs
Admin (Owners/ Managers)	Oversee restaurant operations	a. Real-time sales & revenue tracking b. Menu and pricing management c. Business analytics and report generation

		d. Staff performance and table occupancy monitoring
Staff (Waiters, Chefs, Cashiers)	Daily system users managing orders, tables, and payments	a. Easy order entry and updates b. Table status tracking c. Quick and accurate payment processing d. Real-time order notifications for kitchen staff
Developers & Maintainers	Build, deploy, and support the POS system	a. Clear, well-defined requirements b. Scalable architecture for future growth c. Reliable hosting and deployment infrastructure

c. Outline the Scope of the Project:

The scope of this project clearly delineates the essential functionalities and boundaries of the Restaurant POS system, ensuring that development efforts are focused and aligned with the intended objectives.

- **Order Management:** The system will enable users to create, modify, and cancel customer orders as necessary. Orders can be assigned to specific tables, with their statuses tracked in real-time—from pending to in progress, and ultimately marked as completed—facilitating smooth order flow and enhanced coordination.

- **Table Management:** Users will have the ability to assign orders to tables and monitor the status of each one, indicating whether it is occupied, vacant, or reserved. The system will also provide live updates on table availability, aiding in efficient seating management and helping to avoid confusion during peak hours.

- **Menu Management:** Administrators will be able to add, edit, or remove menu items and categorize them into sections such as starters, main courses, or desserts. Prices and availability of menu items can be updated instantly, simplifying the management of changes during service hours.

- **Billing and Payment Processing:** The POS system will generate customer bills and accommodate various payment methods, including cash, credit/debit cards, and mobile wallets such as Bkash. Upon processing, the system will issue payment confirmations and provide either digital or printed receipts, ensuring a seamless checkout experience.

- **Real-Time Updates:** Staff will receive immediate notifications for new or updated orders, particularly in the kitchen. Table and order statuses will refresh in real-time, keeping both service and kitchen staff synchronized and informed throughout the dining experience.

- **Reporting and Analytics:** The system will produce regular sales reports—daily, weekly, and monthly—and monitor inventory levels, with alerts for low-stock items. It will also offer insights into staff performance and table turnover rates, thereby supporting more informed operational decision-making.

d. Target Audience:

Table 1-3: Target Audience

Customer Segment	Description
Small to Medium-Sized Restaurants	Require an affordable, efficient POS system to manage daily operations effectively.
Quick-Service Restaurants (QSRs)	High-volume, fast-paced environments that need rapid order processing and minimal downtime.
Fine Dining Restaurants	Need advanced features like detailed table management, multi-course order tracking, and a smooth guest experience.

e. Deliverables:

Table 1-4: Deliverables

Deliverable	Description
Web-Based POS System	A responsive, browser-accessible platform offering core POS functionalities.
User Interfaces	a. **Admin Panel:** For menu, sales, and report management b. **Staff Interface:** For managing orders, tables, and payments c. **Kitchen Display System:** For displaying incoming orders to kitchen staff
Payment Integration	Secure payment support through Bkash API, as well as cash and card transactions.
Real-Time Functionality	Live updates for orders, table status, and inventory using WebSocket or similar technologies.
Reporting Tools	Built-in analytics and insights for business performance tracking.
Documentation	Comprehensive user manuals and technical guides for setup, use, and maintenance.

Phase 2: Market Feasibility Analysis

The Market Feasibility Analysis examines the demand for the POS system, identifies competitors, and evaluates the market potential. This phase is critical in ensuring that the system aligns with the needs of the target market and possesses a competitive advantage.

1. Market Demand Analysis

Table 1-5: Market Demand Analysis

Aspect	Details
Target Market	Small to medium-sized restaurants, quick-service restaurants (QSRs), and fine dining establishments seeking affordable, efficient, and feature-rich POS solutions.
Market Trends	Shift toward digital payments; growing adoption of cloud-based POS with real-time capabilities; rising demand for user-friendly, customizable interfaces.

Customer Pain Point	Manual order errors; lack of real-time status updates; insufficient inventory management and reporting tools.

2. Competitor Analysis

Table 1-6: Competitor Analysis

Category	Details
Direct Competitors	**a. Square POS:** Payment processing, inventory management, reporting; can be cost-prohibitive for very small venues. **b. Toast POS:** Tailored to restaurants with table management, menu customization & seamless payments; better for larger operations. **c. Lightspeed POS:** Advanced analytics & inventory features; may be overly complex for small businesses.
Indirect Competitors	a. Traditional cash registers & manual order-taking systems. b. Basic POS solutions offering limited functionality.
Competitive Advantage	a. Affordability: Priced competitively for small & medium restaurants. b. Real-Time Updates: Live order/table status via WebSockets. c. Localized Payments: Built-in Bkash API integration for your local market.

3. Market Potential

Table 1-7: Market Potential

Category	Details
Geographic Reach	a. Initially targeting urban areas with a high concentration of small to medium-sized restaurants.

	b. Potential to expand to rural areas as digital payment adoption grows.
Revenue Model	a. Subscription-based pricing for restaurants. b. One-time setup fee for customization and integration.
Marketing Strategy	a. Partner with local restaurant associations to promote events. b. Offer free trials to attract early adopters. c. Use digital marketing (social media, Google Ads) to reach target customers.

Phase 3: Technical Feasibility Analysis

The Technical Feasibility Analysis evaluates whether the proposed system can be developed using the available technology and resources. It also identifies potential technical challenges and solutions.

a. Technology Stack

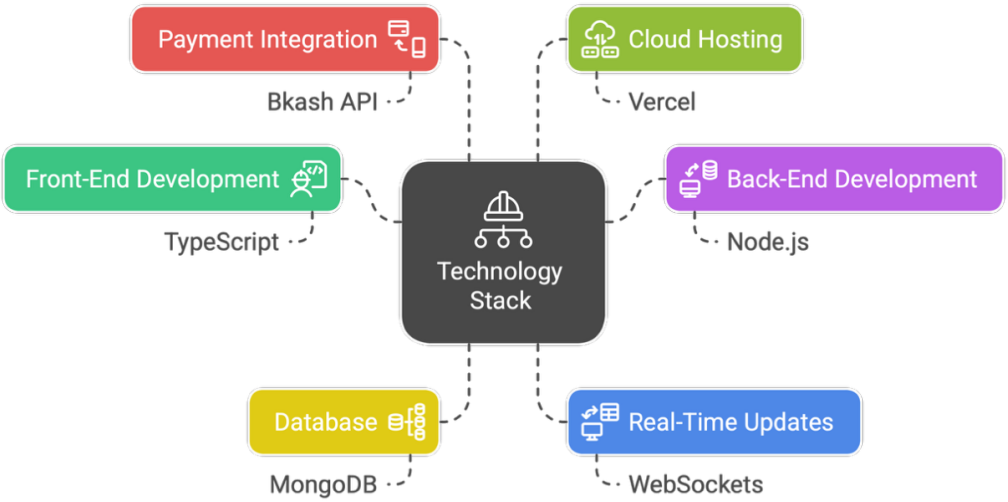


Figure 1—1 : Technology Stack

b. Resource Availability

Development Team:

- Skilled in React.js, Node.js, MongoDB, and WebSockets.
- Experience in integrating third-party APIs (e.g., Bkash).

Tools and Software:

- **IDE:** Visual Studio Code.

- **Version Control:** Git and GitHub.
- **Project Management:** Trello or Jira.

Phase 4: Financial Feasibility Analysis

The Financial Feasibility Analysis assesses the costs associated with developing, deploying, and maintaining the POS system. It also assesses the potential return on investment (ROI).

1. Development Costs

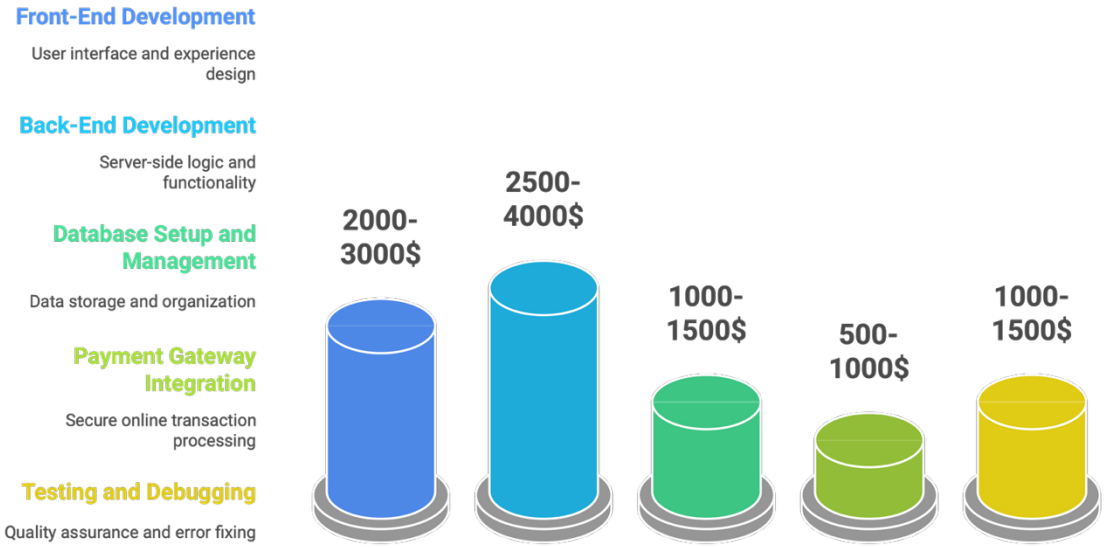


Figure 1—2 : Development Costs

2. Operational Costs

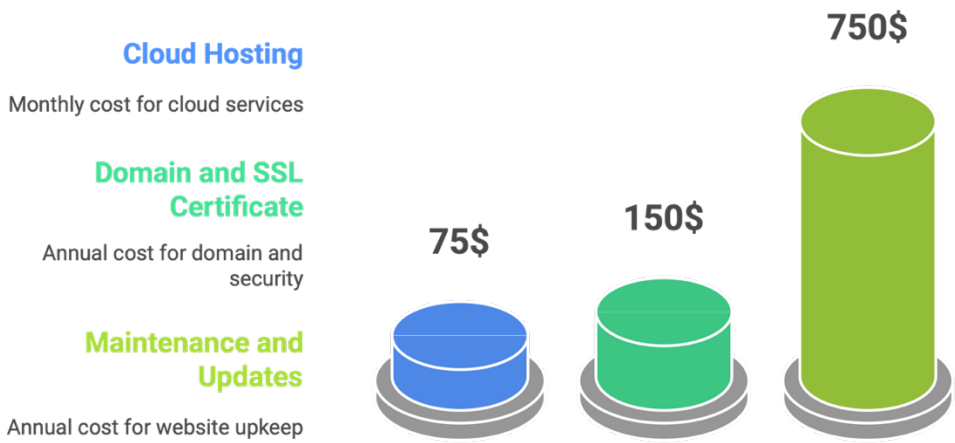


Figure 1—3 : Operational Costs

3. Revenue Model

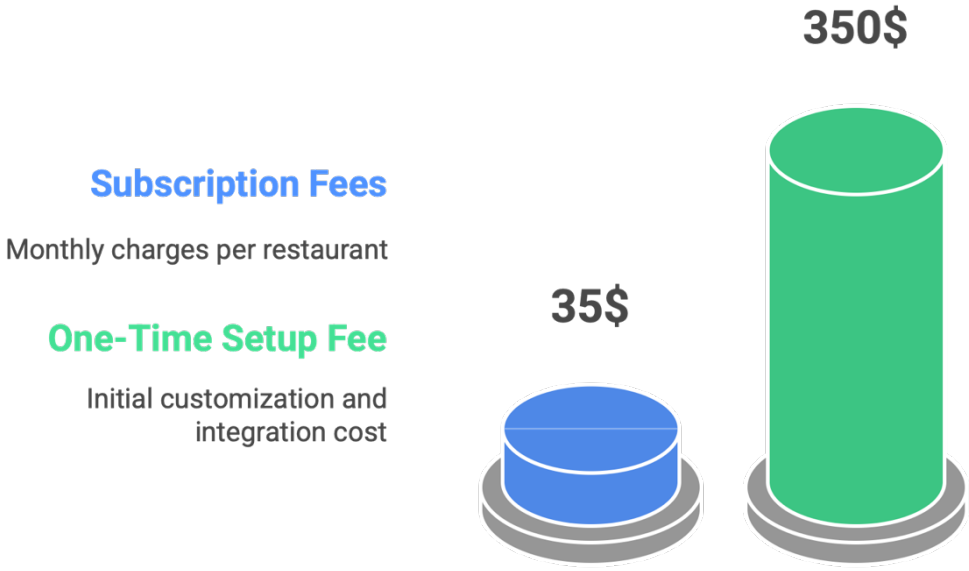


Figure 1—4 : Revenue Model

4. **ROI Analysis**
   - **Break-Even Point:** Achieved within 6-12 months, depending on the number of subscribers.
   - **Profit Margin:** 30% - 40% after covering operational costs.

1.5. Methodology

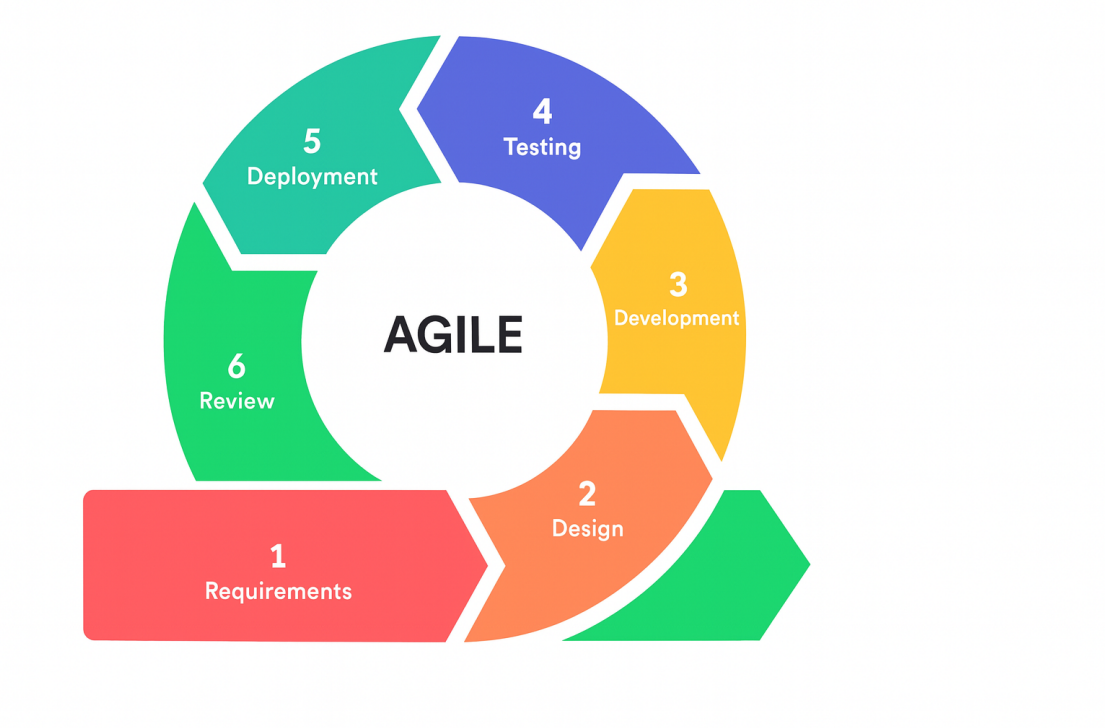


Figure 1—5 : Methodology

1.6. Target User Profile and Tentative Elicitation Process

1.6.1. Target User Profile

The **Restaurant POS System** is designed to cater to a diverse group of users, each with specific roles and responsibilities. Understanding the needs and expectations of these users is critical to developing a system that meets their requirements.

a) Restaurant Owners/Managers

Table 1-8: Restaurant Operations for Owners/Managers

Role	Oversee the overall operations of the restaurant.

Responsibilities	a. Manage menu items and pricing b. Track sales and revenue c. Generate reports for business analysis d. Monitor staff performance and table occupancy
Needs	a. A centralized dashboard to view real-time sales and inventory data b. Easy-to-use tools for menu management and pricing updates c. Comprehensive reporting and analytics for decision-making d. Secure access control to protect sensitive data

b) Staff (Waiters, Chefs, Cashiers)

Table 1-9: Restaurant Operations for Staff

Role	**Interact with the system daily to manage orders, tables, and payments.**
Responsibilities	a. Take and update customer orders b. Track table status (occupied, vacant, reserved) c. Process payments and generate bills d. Communicate order details to kitchen
Needs	a. A simple, intuitive interface for order management b. Real-time updates on table and order status c. Quick and accurate payment processing d. Notifications for new orders

c) Developers and Maintainers

Table 1-10: Developers and Maintainers

Role	**Build, deploy, and maintain the POS system.**

Responsibilities	a. Develop and test the system b. Deploy the system to the cloud c. Provide ongoing maintenance and updates
Needs	a. Clear and detailed requirements for development b. Scalable architecture to accommodate future updates c. Reliable hosting and deployment solutions

1.6.2. Tentative Elicitation Process

The elicitation process involves gathering requirements from stakeholders to ensure the system meets their needs. Below is a tentative plan for the elicitation process:

a) Stakeholder Interviews

Table 1-11: Stakeholder Interviews

Stakeholder	Method	Focus Areas	Deliverable
Owners / Managers	One-on-one interviews	Menu & pricing , Sales & reporting	Functional requirements list High-level use cases
Front-of-House Staff	Guided workflow walk-throughs (on the floor)	Order entry & updates Table status management	Task-flow diagrams UI “wireframe” sketches
Kitchen Staff	Shadowing & contextual inquiry	Order receipt & ticketing Notification needs	Message-flow diagrams Notification spec
System Developer	Requirements workshop + Q&A sessions	Deployment constraints Scalability & security	Non-functional requirements (SLA, uptime, backups)

b) Surveys and Questionnaires

Table 1-12: Surveys and Questionnaires

Aspect	Details
Objective	Collect feedback from a broader group of stakeholders, including restaurant staff and customers.
Activities	a. Design surveys with a mix of multiple-choice and open-ended questions b. Distribute surveys to restaurant staff and customers c. Analyze responses to identify common themes and requirements
Outcome	Actionable insights into user preferences, pain points, and feature requests to inform both functional and non-functional requirements.

c) Observation and Shadowing

Table 1-13: Observation and Shadowing

Aspect	Details
Objective	Observe restaurant operations to identify inefficiencies and areas for improvement.
Activities	a. Spend time in restaurants observing how staff take orders, manage tables, and process payments . b. Note any bottlenecks or challenges in the current workflow.
Outcome	A deeper understanding of the operational challenges and opportunities for automation, informing design decisions for smoother workflows.

d) Competitor Analysis

Table 1-14: Competitor Analysis

Aspect	Details
Objective	Analyze existing POS systems to identify best practices and gaps.
Activities	a. Research popular POS systems like Square, Toast, and Lightspeed. b. Identify features that are well-received by users. c. Note any missing features or areas for improvement.
Outcome	A list of features and functionalities to include in the system.

e) Workshops and Brainstorming Sessions

Table 1-15: Workshops and Brainstorming Sessions

Step	Action	Participants	Techniques/Tools	Output
1	Plan & Schedule Workshops	Project lead, Owners/Managers, FOH, Dev team	Calendar invites, brief agenda doc	Confirmed dates & agendas
2	Kick-off & Context Setting	All stakeholders	Presentation deck, whiteboard	Shared understanding of goals
3	Brainstorm Features	Mixed-group teams (owners, staff, devs)	Post-it notes, affinity mapping	Raw feature ideas
4	Group & Cluster Ideas	Facilitator + small teams	Dot-voting, digital boards (e.g. Miro)	Themed feature clusters
5	Assess Feasibility & Impact	Dev leads + Owners/Managers	Impact/Effort matrix	Scored requirements
6	Prioritize & Finalize Requirements	Full stakeholder group	Consensus voting	Final prioritized requirements list
7	Document & Distribute	Project lead	Shared doc (Confluence/Drive)	Published requirements spec & backlog

f) Prototyping and Feedback

Table 1-16: Prototyping and Feedback

Phase	Prototype Fidelity	Description	Stakeholders Involved	Deliverable
Design Drafting	Low (Wireframes)	Sketch core screens for order management, billing, and	UX Designer, Product Owner	Set of hand-drawn or digital wireframes

		dashboard layouts.		
Initial Review	Low	Walk stakeholders through wireframes to collect first impressions and "big picture" feedback.	Owners, Managers, FOH Staff	Consolidated feedback notes
Interactive Mock	Mid (Clickable Prototype)	Build a simple clickable prototype (e.g., using Figma or InVision) to simulate basic workflows.	Dev Lead, QA, Kitchen Staff	Clickable prototype URL/file
Usability Testing	Mid	Ask a small group of end-users to perform key tasks (take an order, process a payment) and observe.	Front-of-House Staff, Customers	Usability report highlighting friction points
Iteration	Low → Mid	Update prototypes to address major issues—refine layouts, labels, and interaction flows.	UX Designer, Dev Team	Revised wireframes & prototype
Final Validation	Mid	Present the updated clickable prototype for final sign-off on workflows and visual hierarchy.	All Stakeholders	Approved prototype ready for UI mock-ups

Conclusion

The Target User Profile and Tentative Elicitation Process provides a clear understanding of the system's users and their needs. The elicitation process ensures that the requirements are gathered systematically and validated through stakeholder feedback. This phase lays the foundation for the design and development of the POS system.

1.7. System Requirements

1.7.1. Development Environmrnt Requirements

Table 1-17: Development Environmrnt Requirements

Category	Requirements
Hardware	Any modern computer
Operating System	Windows, MacOS, & Linux
Software	1. Node js (Version 18.x) only 2. Code Editor (Visual Studio Code recommended) 3. GitHub 4. MongoBD Compass 5. Web Browser
Accounts	1. GitHub 2. Vercel 3. MongoBD

1.7.2. User Environmrnt Requirements

Table 1-18: User Environmrnt Requirements

Category	Requirements
Hardware	Any modern computer
Operating System	Windows, MacOS, & Linux
Software	Web Browser
Network	Stable broadband or mobile data connection

1.8. Project Scheduling

1.8.1. Gantt Chart

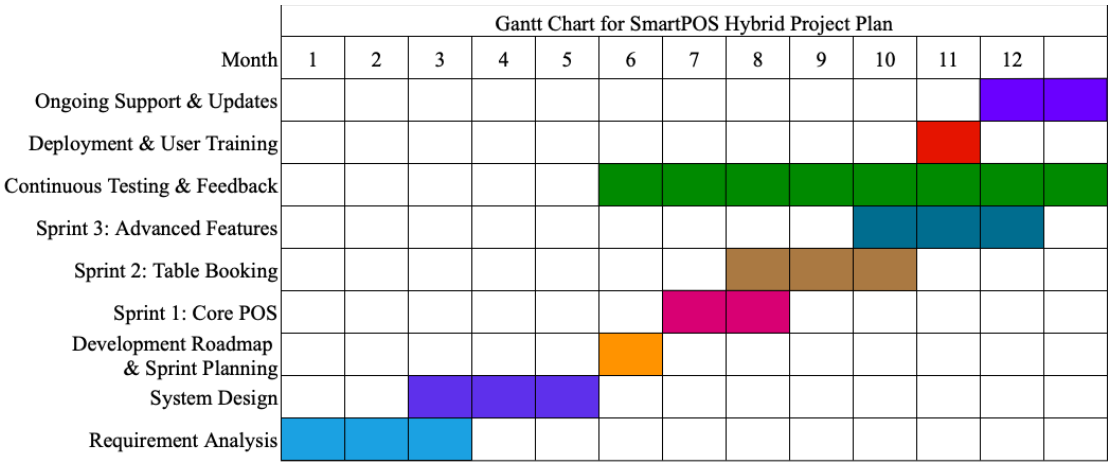


Figure 1—6 : Gantt Chart

1.8.2. Risk Management

Table 1-19: Risk Management

Risk	Likelihood	Impact	Mitigation Strategy
Delays in payment gateway integration	Medium	High	Allocate extra time for integration and testing.
Staff shortages during development	Low	Medium	Cross-train team members to handle multiple roles.
Performance issues during peak hours	High	High	Conduct stress testing and optimize for scalability.
Scope creep (unplanned feature requests)	Medium	High	Define clear project scope and stick to it. Use change management processes for any new requests.
Delays in testing due to bugs	High	High	Allocate buffer time in the testing phase and prioritize critical bug fixes.

In conclusion, establishes the foundation of the project by defining its objectives, outlining the scope, and identifying key stakeholders. It demonstrates how SmartPOS aims to streamline restaurant operations through automation, scalability, and real-time functionality.

Chapter 2 : Design and Implementation

This chapter describes the design and implementation strategy of SmartPOS. It discusses functional and non-functional requirements, system architecture, and UML-based modeling, ensuring that each feature is systematically structured to meet operational needs.

2.1. Functional Requirements

Functional requirements define the specific behaviors and feature the system must have to meet the needs of its users.

Case Description - 01 : Registration & Login

Table 2-1: Registration & Login

FR 001	Registration & Login
Description	Users should be able to register with basic information (Username , password & role), which the system must verify for validity. After registration, users can securely log in using their username and password to access personalized features.
Stakeholder	Admin, Staff

Case Description - 02 : Create Category

Table 2-2: Create Category

FR 002	Create Category
Description	Authorized users should be able to create new food categories by providing a category name and description. These categories help organize menu items efficiently.
Stakeholder	Admin

Case Description - 03 : Category List

Table 2-3: Category List

FR 003	Category List
Description	Displays all existing food categories. Authorized users can view details, edit category information, or delete categories from the list.
Stakeholder	Admin

Case Description - 04 : Food List

Table 2-4: Food List

FR 004	Food List
Description	Shows all food items grouped by category. Authorized users can view details, edit food information (like name or price), delete items, and search for specific food items.
Stakeholder	Admin, Staff

Case Description - 05 : Create Food

Table 2-5: Create Food

FR 005	Create Food
Description	Authorized users can add new food items by entering the name, price, category, description, and image URL. This allows for easy expansion of the restaurant's digital menu.
Stakeholder	Admin

Case Description - 06 : Create Table

Table 2-6: Create Table

FR 006	Create Table
Description	Authorized users can create new tables by specifying the table number, seat capacity, and availability status (like Available or Occupied).
Stakeholder	Admin

Case Description - 07 : Table List

Table 2-7: Table List

FR 007	Table List
Description	Displays all existing tables with their details. Authorized users can view, edit, or update table information such as seat capacity, availability, and status.
Stakeholder	Admin, Staff

Case Description - 08 : Create Order

Table 2-8: Create Order

FR 008	Create Order
Description	Authorized users can create new orders by selecting a customer name, assigning a table, choosing the order type (like Dine In, Takeaway), and adding food items with quantities.

Stakeholder	Admin, Staff

Case Description - 09 : Order List

Table 2-9: Order List

FR 009	Order List
Description	Displays all placed orders along with their details such as items, table number, status, time, and payment info. Users can view, search, and delete orders as needed.
Stakeholder	Admin, Staff

Case Description - 10 : Order Queue

Table 2-10: Order Queue

FR 010	Order Queue
Description	Shows all active or pending orders in real-time for kitchen or processing staff. Users can update order statuses and search for specific orders by customer name.
Stakeholder	Admin, Staff

Case Description - 11 : Income Report

Table 2-11: Income Report

FR 011	Income Report
Description	Generates income reports based on completed (paid) orders within a selected date range. Users can view financial summaries like monthly earnings for analysis.
Stakeholder	Admin

Case Description - 12 : Chat with AI Assistant

Table 2-12: Chat with AI Assistant

FR 012	Chat with AI Assistant
Description	The AI assistant provides real-time updates on ongoing sales, calculates total earnings, tracks the number of waiting customers, and offers quick insights to support business decisions.
Stakeholder	Admin, Staff

2.2. Non-Functional Requirements

Case Description - 01 : Performance

Table 2-13: Performance

NFR 01	Performance
Description	The system must support up to 100 users at once and respond to actions within 0.20 to 0.40 seconds for smooth operation.
Stakeholder	Admin , Staff & Customers (indirectly)

Case Description - 02 : Security

Table 2-14: Security

NFR 02	Security
Description	The system must use SSL encryption, role-based access control, and secure storage for sensitive data like payment info.
Stakeholder	Admin , Staff & Customers (indirectly)

Case Description - 03 : Usability

Table 2-15: Usability

NFR 03	Usability
Description	The system should have a user-friendly interface with tooltips and guides to help new users navigate easily.
Stakeholder	Admin , Staff & Customers (indirectly)

Case Description - 04 : Scalability

Table 2-16: Scalability

NFR 04	Scalability
Description	The system must support multiple branches and allow easy addition of new features through a modular design.
Stakeholder	Admin , Staff and system maintainers.

Case Description - 05 : Reliability

Table 2-17: Reliability

NFR 05	Reliability
Description	The system should ensure 99.9% uptime and automatically recover from failures to maintain continuous operation.

Stakeholder	Admin , Staff and developers.

Case Description - 06 : Compatibility

Table 2-18: Compatibility

NFR 06	Compatibility
Description	The system must work across major browsers (Chrome, Firefox, Safari) and be accessible on both desktop and mobile devices.
Stakeholder	Admin , Staff and developers.

2.3. Object-Oriented System Design Using UML

A. Use Case Diagram

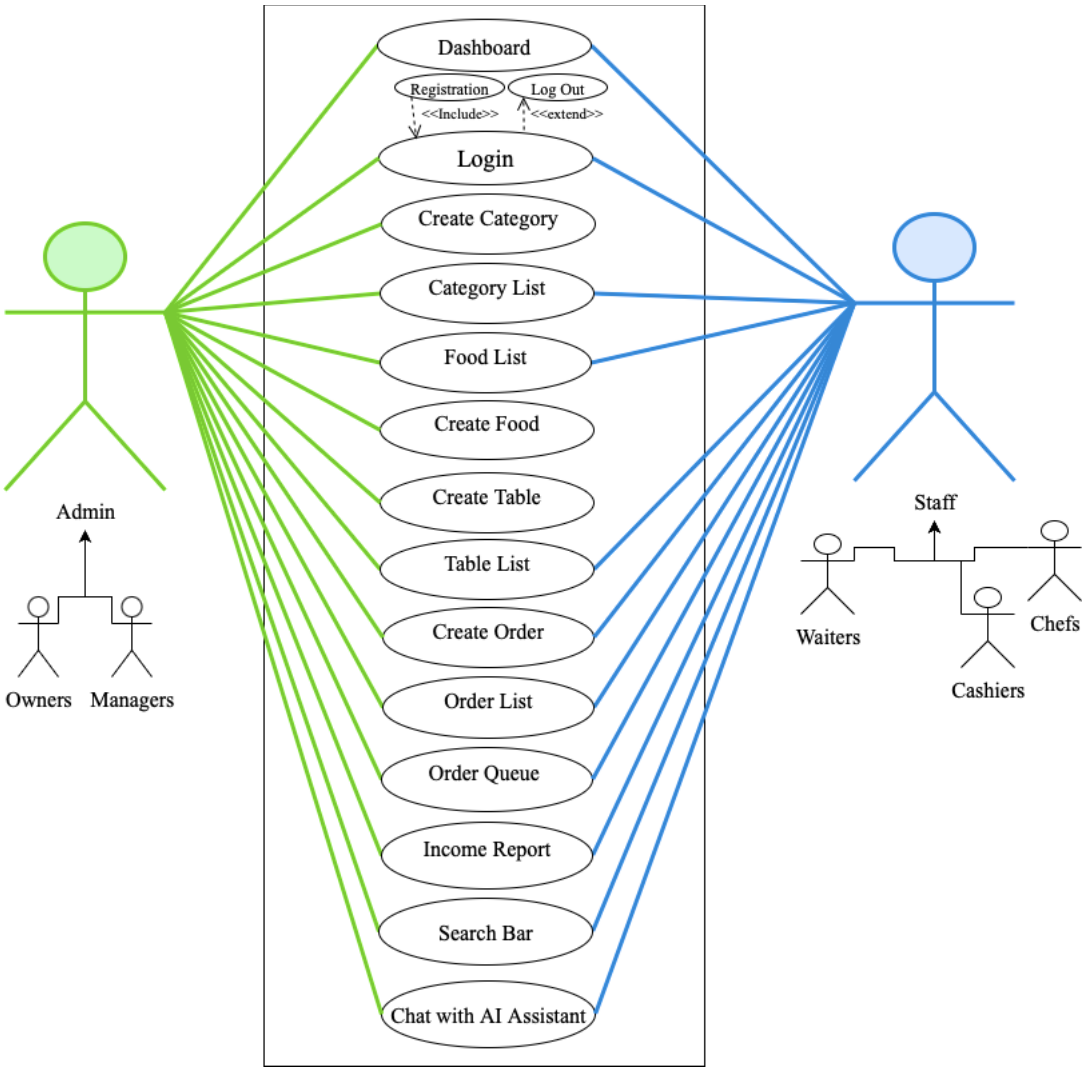


Figure 2—1: Use Case Diagram

B. Case Description

- **Registration & Login**

Table 2-19: Case Description for Registration & Login

<table>
<tr><td>Use Case</td><td>Registration & Login</td></tr>
<tr><td>Goal</td><td>Allow users to register and log in securely to access the system based on their role.</td></tr>
<tr><td>Precondition</td><td>The user is not logged in; registration page or login page is accessible.</td></tr>
<tr><td>Success End Condition</td><td>The user is registered or logged in successfully and redirected to their dashboard.</td></tr>
<tr><td>Failed End Condition</td><td>Registration or login fails due to invalid data or incorrect credentials.</td></tr>
<tr><td>Primary Actor</td><td>Admins, Staff</td></tr>
<tr><td>Trigger</td><td>The user navigates to the registration or login page.</td></tr>
<tr><td>Description</td><td><table><tr><td>Step</td><td>Action</td></tr><tr><td>1</td><td>User opens the system and clicks “Register” or “Login”.</td></tr><tr><td>2</td><td>User fills in required fields (registration: Username, password, Choose Role).</td></tr><tr><td>3</td><td>System validates inputs and either registers/logs in the user or shows an error.</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>Step</td><td>Action</td></tr><tr><td>2.a</td><td>User enters invalid or missing information.</td></tr><tr><td>2.a.1</td><td>System displays appropriate error messages and missing information re-entry.</td></tr><tr><td>3.a</td><td>User enters invalid ‘Username’ / ‘Password’</td></tr><tr><td>3.a.1</td><td>System displays appropriate error messages and enter corrected ‘Username’ / ‘Password’.</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Form should be simple and clear, with validation and tooltips.
• Performance: Response time < 2 seconds.</td></tr>
</table>

	• Security: Passwords stored securely, login over HTTPS, input validation to prevent SQL/NoSQL injection. • Compatibility: Works on all modern browsers and mobile devices.

- **Create Category**

Table 2-20: Case Description for Create Category

Use Case	Create Category
Goal	To allow authorized users to add new food categories for organizing menu items.
Precondition	The user must be logged in with admin privileges.
Success End Condition	A new food category is successfully created and added to the system.
Failed End Condition	The category is not created due to missing data, duplication, or system error.
Primary Actor	Admins
Trigger	User navigates to the "Create Category" section and submits category details.
Description	<table><tr><td>Step</td><td>Action</td></tr><tr><td>1</td><td>Admin logs in and clicks on the "Create Category" button.</td></tr><tr><td>2</td><td>Enters the category name and a brief description.</td></tr><tr><td>3</td><td>Clicks the "Submit" button to add the category.</td></tr><tr><td>4</td><td>System validates input and confirms successful creation.</td></tr></table>
Alternative Flows	<table><tr><td>Step</td><td>Action</td></tr><tr><td>2.a</td><td>Admin enters a category name that already exists.</td></tr><tr><td>2.a.1</td><td>System shows an error: "Category already exists." and prompts correction.</td></tr><tr><td>3.a</td><td>Required fields are left blank.</td></tr><tr><td>3.a.1</td><td>System displays validation error messages for missing fields.</td></tr></table>

Quality Requirements	• Usability: Clear labels and instructions for each input field. • Performance: Category creation completes within 2 seconds. • Security: Only authorized users (admin/manager) can access this feature. • Reliability: System prevents duplicate entries and handles input validation errors gracefully.

- **Category List**

Table 2-21: Case Description for Category List

Use Case	Category List
Goal	To allow authorized users to view, edit, or delete existing food categories.
Precondition	The user must be logged in with proper access
Success End Condition	The user successfully views the list of categories and performs desired actions.
Failed End Condition	The list fails to load, or edit/delete operations are unsuccessful due to errors.
Primary Actor	Admins, Staff
Trigger	User selects the "Category List" option from the dashboard / sidebar.
Description	

Step	Action
1	User navigates to the "Category List" page.
2	System displays all existing food categories with edit and deletes options.
3	User chooses to either view details, edit, or delete a specific category.
4	System performs the selected action and updates the list accordingly.

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>2.a</td><td>System fails to load category list.</td></tr>
<tr><td>2.a.1</td><td>Displays an error message like “Unable to load categories. Please try again.”</td></tr>
<tr><td>3.a</td><td>User tries to delete a category linked to existing menu items.</td></tr>
<tr><td>3.a.1</td><td>System prevents deletion and shows a warning: “Category in use, cannot be deleted.”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Clear layout with action buttons (edit, delete, details).
• Performance: List loads within 2 seconds even with many categories.
• Security: Only authorized roles can edit or delete categories.
• Reliability: Prevents deletion of in-use categories and ensures real-time updates.</td></tr>
</table>

- **Food List**

Table 2-22: Case Description for Food List

<table>
<tr><td>Use Case</td><td>Food List</td></tr>
<tr><td>Goal</td><td>To allow authorized users to view, search, edit, or delete food items organized by category.</td></tr>
<tr><td>Precondition</td><td>User is logged in with appropriate access (admin or Staff). At least one food item exists.</td></tr>
<tr><td>Success End Condition</td><td>The user successfully views and manages the food list.</td></tr>
<tr><td>Failed End Condition</td><td>The list fails to load, or edit/delete operations fail due to errors or invalid input.</td></tr>
<tr><td>Primary Actor</td><td>Admins, Staff</td></tr>
<tr><td>Trigger</td><td>User selects the “Food List” option from the menu / sidebar.</td></tr>
<tr><td>Description</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>1</td><td>User navigates to the “Food List” page.</td></tr>
<tr><td>2</td><td>System displays all food items grouped by category.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td>3</td><td>User can search, edit, or delete a food item.</td></tr>
<tr><td>4</td><td>System processes the action and updates the list accordingly.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>2.a</td><td>System fails to load food list.</td></tr>
<tr><td>2.a.1</td><td>Displays an error: “Unable to load food items.”</td></tr>
<tr><td>3.a</td><td>User tries to delete a food item that’s part of an active order.</td></tr>
<tr><td>3.a.1</td><td>System blocks deletion and shows warning: “Item in use, cannot delete.”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Easy navigation, search bar, and clearly labeled action buttons.
• Performance: List and search results should load within 2 seconds.
• Security: Only authorized users can edit or delete food items.
• Reliability: Prevents deletion of in-use items and ensures up-to-date display.</td></tr>
</table>

- **Create Food**

Table 2-23: Case Description for Create Food

Use Case	Create Food
Goal	To allow authorized users to add new food items with relevant details to the system menu.
Precondition	User is logged in with admin access. At least one food category already exists.
Success End Condition	A new food item is successfully created and appears in the food list.
Failed End Condition	Food item creation fails due to validation errors or missing information.
Primary Actor	Admins
Trigger	User clicks the “Create Food” button from the menu section.

<table>
<tr><td>Description</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>1</td><td>User navigates to the “Create Food” page.</td></tr>
<tr><td>2</td><td>Enters food name, price, selects a category, adds a description, and an image URL.</td></tr>
<tr><td>3</td><td>Clicks “Create Food Item” to add the food item.</td></tr>
<tr><td>4</td><td>System validates and saves the item, showing a success message.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>2.a</td><td>Required fields (e.g., name or price) are left empty.</td></tr>
<tr><td>2.a.1</td><td>System shows error: “Please fill all required fields.”</td></tr>
<tr><td>3.a</td><td>Duplicate food item name is used.</td></tr>
<tr><td>3.a.1</td><td>System alerts: “Food item already exists.”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Clean form layout with dropdowns for category selection and helpful error messages.
• Performance: Food creation should complete within 2 seconds.
• Security: Only authorized roles can access this function.
• Reliability: Validates all fields and ensures the item is properly saved and displayed.</td></tr>
</table>

- **Create Table**

Table 2-24: Case Description for Create Table

Use Case	Create Table
Goal	To allow authorized users to create new tables with seat capacity and availability status.
Precondition	User is logged in with admin or manager privileges.
Success End Condition	A new table is successfully created and listed in the system.
Failed End Condition	Table creation fails due to missing/invalid inputs or duplication.
Primary Actor	Admins

<table>
<tr><td>Trigger</td><td>User clicks the “Create Table” button in the menu section.</td></tr>
<tr><td>Description</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>1</td><td>User navigates to the “Create Table” page.</td></tr>
<tr><td>2</td><td>Enters table number and seat capacity, then selects availability and status.</td></tr>
<tr><td>3</td><td>Clicks “Submit” to create the table.</td></tr>
<tr><td>4</td><td>System validates and saves the table, showing a success message.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>2.a</td><td>User leaves a required field blank or enters invalid data.</td></tr>
<tr><td>2.a.1</td><td>System shows a validation error prompting correction.</td></tr>
<tr><td>3.a</td><td>User enters a table number that already exists.</td></tr>
<tr><td>3.a.1</td><td>System displays an error: “Table number already exists.”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>
Usability: Easy-to-use form with dropdowns for status and availability.
Performance: Table creation should complete within 2 seconds.
Security: Access restricted to admin/manager roles.
Reliability: System validates data and prevents duplication or errors.
</td></tr>
</table>

- **Table List**

Table 2-25: Case Description for Table List

Use Case	Table List
Goal	To allow authorized users to view, edit, or delete tables with details like status and seat capacity.
Precondition	User is logged in with admin or manager privileges. Tables must already exist in the system.

<table>
<tr><td>Success End Condition</td><td>Tables are displayed, and the user can successfully view, edit, or delete them.</td></tr>
<tr><td>Failed End Condition</td><td>Table list fails to load, or an operation (edit/delete) is unsuccessful due to an error.</td></tr>
<tr><td>Primary Actor</td><td>Admins, Staff</td></tr>
<tr><td>Trigger</td><td>User selects the “Table List” option button in the menu section.</td></tr>
<tr><td>Description</td><td><table><tr><td>Step</td><td>Action</td></tr><tr><td>1</td><td>User navigates to the “Table List” page.</td></tr><tr><td>2</td><td>System displays all tables with information like number, seats, and availability.</td></tr><tr><td>3</td><td>User selects a table to view, edit, or delete.</td></tr><tr><td>4</td><td>System processes the update or deletion and refreshes the list.</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>Step</td><td>Action</td></tr><tr><td>2.a</td><td>No tables exist or system fails to load list.</td></tr><tr><td>2.a.1</td><td>System shows a message: “No tables found” or “Error loading tables.”</td></tr><tr><td>3.a</td><td>User attempts to delete a table currently assigned to an order.</td></tr><tr><td>3.a.1</td><td>System blocks deletion and shows warning: “Table in use, cannot be deleted.”</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Clear layout, action buttons (edit/delete), filters.
• Performance: List should load in under 2 seconds.
• Security: Only authorized users can edit or delete tables.
• Reliability: Prevents changes that would affect active orders.</td></tr>
</table>

- **Create Order**

Table 2-26: Case Description for Create Order

Use Case	Create Order

<table>
<tr><td>Goal</td><td>To allow staff to create and assign new customer orders to a table with selected food items.</td></tr>
<tr><td>Precondition</td><td>User is logged in with appropriate access (e.g., waiter, manager). Tables and food items are already available in the system.</td></tr>
<tr><td>Success End Condition</td><td>The order is successfully created, saved, and sent to the kitchen queue.</td></tr>
<tr><td>Failed End Condition</td><td>Order creation fails due to missing fields or system errors.</td></tr>
<tr><td>Primary Actor</td><td>Staff</td></tr>
<tr><td>Trigger</td><td>User clicks the “Create Order” option from the menu section.</td></tr>
<tr><td>Description</td><td>
<table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>1</td><td>User navigates to the “Create Order” page.</td></tr>
<tr><td>2</td><td>Enters customer name, selects table number and order type.</td></tr>
<tr><td>3</td><td>Adds food items with desired quantities.</td></tr>
<tr><td>4</td><td>Submits the order.</td></tr>
<tr><td>5</td><td>System validates, saves the order, and updates the order list/queue.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>2.a</td><td>Table is already occupied or unavailable.</td></tr>
<tr><td>2.a.1</td><td>System shows a warning: “Table not available.”</td></tr>
<tr><td>3.a</td><td>Food item is out of stock.</td></tr>
<tr><td>3.a.1</td><td>System alerts: “Selected item is unavailable.”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>

Usability: Smooth item selection, dropdowns for tables/order type, quantity input.
Performance: Order should be saved and reflected in real-time within 2 seconds.
Security: Only logged-in staff can create orders.
Reliability: Prevents ordering from unavailable tables or out-of-stock items.

</td></tr>
</table>

- **Order List**

Table 2-27: Case Description for Order List

<table>
<tr><td>Use Case</td><td>Order List</td></tr>
<tr><td>Goal</td><td>To allow staff or managers to view, search, and manage all placed orders.</td></tr>
<tr><td>Precondition</td><td>User is logged in with appropriate access. At least one order must exist in the system.</td></tr>
<tr><td>Success End Condition</td><td>The user views the order list and optionally edits, searches, or deletes an order.</td></tr>
<tr><td>Failed End Condition</td><td>The list fails to load or an action like delete/edit fails due to system or permission errors.</td></tr>
<tr><td>Primary Actor</td><td>Staff</td></tr>
<tr><td>Trigger</td><td>User selects the “Order List” from the menu section.</td></tr>
<tr><td>Description</td><td><table><tr><td>Step</td><td>Action</td></tr><tr><td>1</td><td>User navigates to the “Order List” page.</td></tr><tr><td>2</td><td>System displays all orders with filters/search options.</td></tr><tr><td>3</td><td>User views detail or performs actions like delete or update.</td></tr><tr><td>4</td><td>System updates the list accordingly</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>Step</td><td>Action</td></tr><tr><td>2.a</td><td>No orders exist or system fails to load list.</td></tr><tr><td>2.a.1</td><td>System displays “No orders found” or error message.</td></tr><tr><td>3.a</td><td>User tries to delete an order already completed or paid.</td></tr><tr><td>3.a.1</td><td>System shows warning: “Completed orders cannot be deleted.”</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Search bar, filter options, and clearly labeled action buttons.
• Performance: Orders should load within 2 seconds.
• Security: Only authorized users can modify or delete orders.</td></tr>
</table>

	• Reliability: System ensures data integrity and prevents deletion of finalized orders.

- **Order Queue**

Table 2-28: Case Description for Order Queue

Use Case	Order Queue
Goal	To allow kitchen staff and managers to view and manage active or pending orders in real-time.
Precondition	User is logged in with appropriate access (e.g., kitchen staff, admin). At least one order is in pending, preparing, or in-progress status.
Success End Condition	Active orders are displayed and updated in real time, with status changes reflected immediately.
Failed End Condition	Order queue fails to load or status changes are not applied due to errors.
Primary Actor	Staff
Trigger	User clicks on "Order Queue" from the navigation menu.
Description	<table><tr><td>Step</td><td>Action</td></tr><tr><td>1</td><td>User opens the "Order Queue" page.</td></tr><tr><td>2</td><td>System shows live orders with current statuses.</td></tr><tr><td>3</td><td>User updates the status of an order as it progresses.</td></tr><tr><td>4</td><td>System reflects the status change in real time for all users.</td></tr></table>
Alternative Flows	<table><tr><td>Step</td><td>Action</td></tr><tr><td>2.a</td><td>No active orders in the queue.</td></tr><tr><td>2.a.1</td><td>System displays: "No active orders."</td></tr><tr><td>3.a</td><td>Status update fails due to network or system error.</td></tr></table>

	3.a.1 System displays error and keeps previous status unchanged.
Quality Requirements	• Usability: Color-coded statuses, easy status-change dropdown/buttons. • Performance: Real-time updates within 1–2 seconds via WebSockets or polling. • Security: Only kitchen staff, admin, or authorized users can access or change status. • Reliability: Ensures order status integrity and prevents unauthorized updates.

- **Income Report**

Table 2-29: Case Description for Income Report

Use Case	Income Report
Goal	To allow authorized users to generate and view income reports based on completed orders over a selected date range.
Precondition	User is logged in with admin or manager access. There must be completed (paid) orders in the system.
Success End Condition	Income report is generated and displayed correctly with totals and filters.
Failed End Condition	Report generation fails due to lack of data, system error, or invalid date input.
Primary Actor	Staff
Trigger	User navigates to the “Income Report” section and selects a date range.
Description	Step / Action 1 User accesses the “Income Report” page. 2 Selects a date range (e.g., daily, weekly, monthly). 3 Clicks “Generate” or similar action. 4 System calculates and displays the income report.

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>2.a</td><td>No completed orders in the selected date range.</td></tr>
<tr><td>2.a.1</td><td>System displays: “No income data available for this period.”</td></tr>
<tr><td>3.a</td><td>User enters an invalid or future date range.</td></tr>
<tr><td>3.a.1</td><td>System alerts: “Invalid date selection.”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>• Usability: Easy-to-use date picker, clear totals and breakdowns.
• Performance: Report should generate in under 2 seconds.
• Security: Only admin or manager roles can access financial reports.
• Reliability: Accurate data retrieval and consistent formatting.</td></tr>
</table>

- **Search Bar**

Table 2-30: Case Description for Search Bar

<table>
<tr><td>Use Case</td><td>Search Bar</td></tr>
<tr><td>Goal</td><td>To allow users to quickly find specific items (e.g., food, categories, orders, or customers) using keywords.</td></tr>
<tr><td>Precondition</td><td>User is logged in and is on a page that includes a searchable list (e.g., Food List, Order List).</td></tr>
<tr><td>Success End Condition</td><td>Matching results are displayed based on the user’s input.</td></tr>
<tr><td>Failed End Condition</td><td>No results found or search function fails to execute due to system error.</td></tr>
<tr><td>Primary Actor</td><td>Amin, Staff</td></tr>
<tr><td>Trigger</td><td>User types a keyword in the search bar and initiates the search.</td></tr>
<tr><td>Description</td><td><table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>1</td><td>User enters text into the search bar on a list page (e.g., Food List or Order List).</td></tr>
</table></td></tr>
</table>

	2 | The system filters and displays results that match the input.
Alternative Flows	Step | Action 2.a | No matching results found. 2.a.1 | System shows: "No results found." 2.b | Search bar left empty and submitted. 2.b.1 | System returns the full, unfiltered list.
Quality Requirements	• Usability: Search should support partial matches and be case-insensitive. • Performance: Results should appear within 1 second for small to medium datasets. • Reliability: Works consistently across modules with searchable content. • Compatibility: Functions on both desktop and mobile views.

- **Chat with AI Assistant**

Table 2-31: Case Description for Chat with AI Assistant

Use Case	Chat with AI Assistant
Goal	To allow users to interact with an AI assistant for real-time business insights, including ongoing sales count, total earnings, and the number of waiting customers.
Precondition	User is logged in with authorized access to sales and customer data.
Success End Condition	AI assistant accurately provides the requested information and suggestions in real time.
Failed End Condition	The system fails to retrieve data or provides inaccurate/outdated information due to connectivity or processing errors.
Primary Actor	Admin, Staff
Trigger	User initiates a chat session with the AI assistant from the dashboard or relevant page.

<table>
<tr><td>Description</td><td>
<table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>1</td><td>User opens the AI assistant chat interface.</td></tr>
<tr><td>2</td><td>User types a query (e.g., “How many sales are ongoing?”).</td></tr>
<tr><td>3</td><td>The AI assistant processes the query and retrieves relevant data from the system.</td></tr>
<tr><td>4</td><td>System displays the results in the chat along with additional insights if available.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>Step</td><td>Action</td></tr>
<tr><td>3.a</td><td>AI assistant fails to retrieve data.</td></tr>
<tr><td>3.a.1</td><td>System displays: “Unable to fetch data at the moment. Please try again later.”</td></tr>
<tr><td>3.b</td><td>User asks a query outside the AI assistant’s scope.</td></tr>
<tr><td>3.b.1</td><td>System replies with: “I’m not able to help with that, but you can try [related feature].”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>

Usability: Simple, intuitive chat interface with suggested queries and quick action buttons.
Performance: Responses generated within 2 seconds.
Reliability: Always retrieves the most recent and accurate data.
Security: Ensures that only authorized users can access sensitive business metrics.
Compatibility: Works seamlessly on desktop and mobile platforms.

</td></tr>
</table>

C. Activity Diagram

- **Registration**

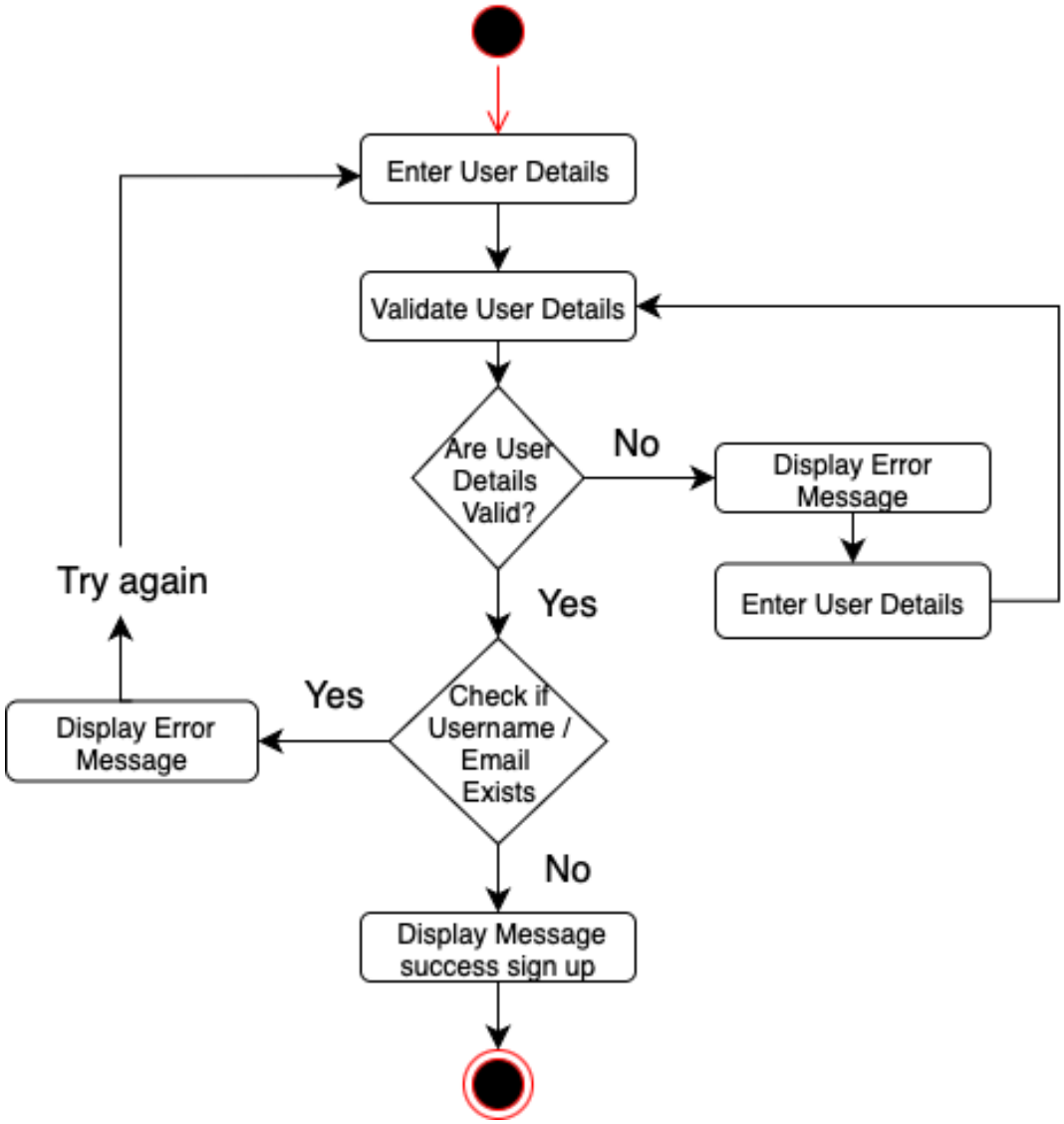


Figure 2—2 : Registration

- **Login**

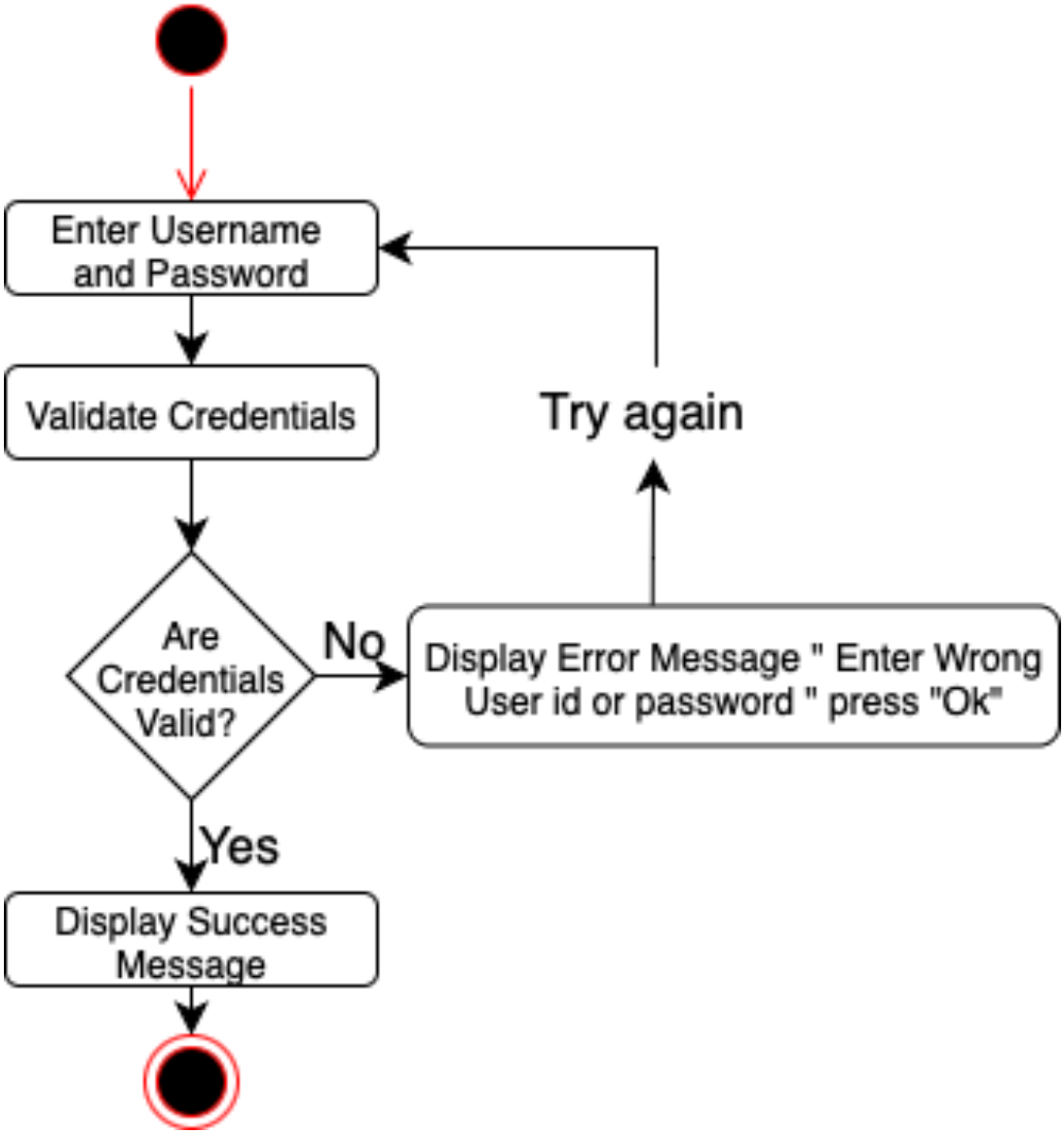


Figure 2—3 : Login

- **Create Category**

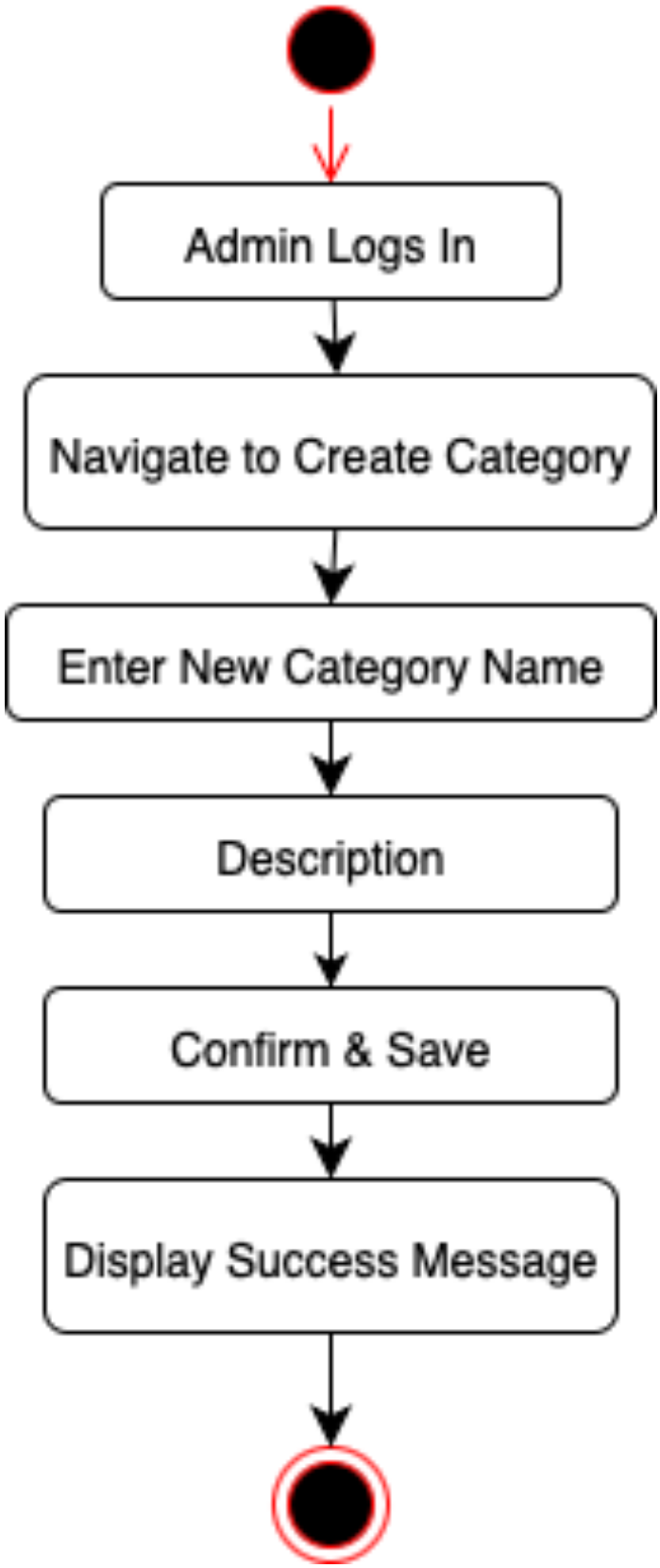


Figure 2—4 : Create Category

- **Category List**

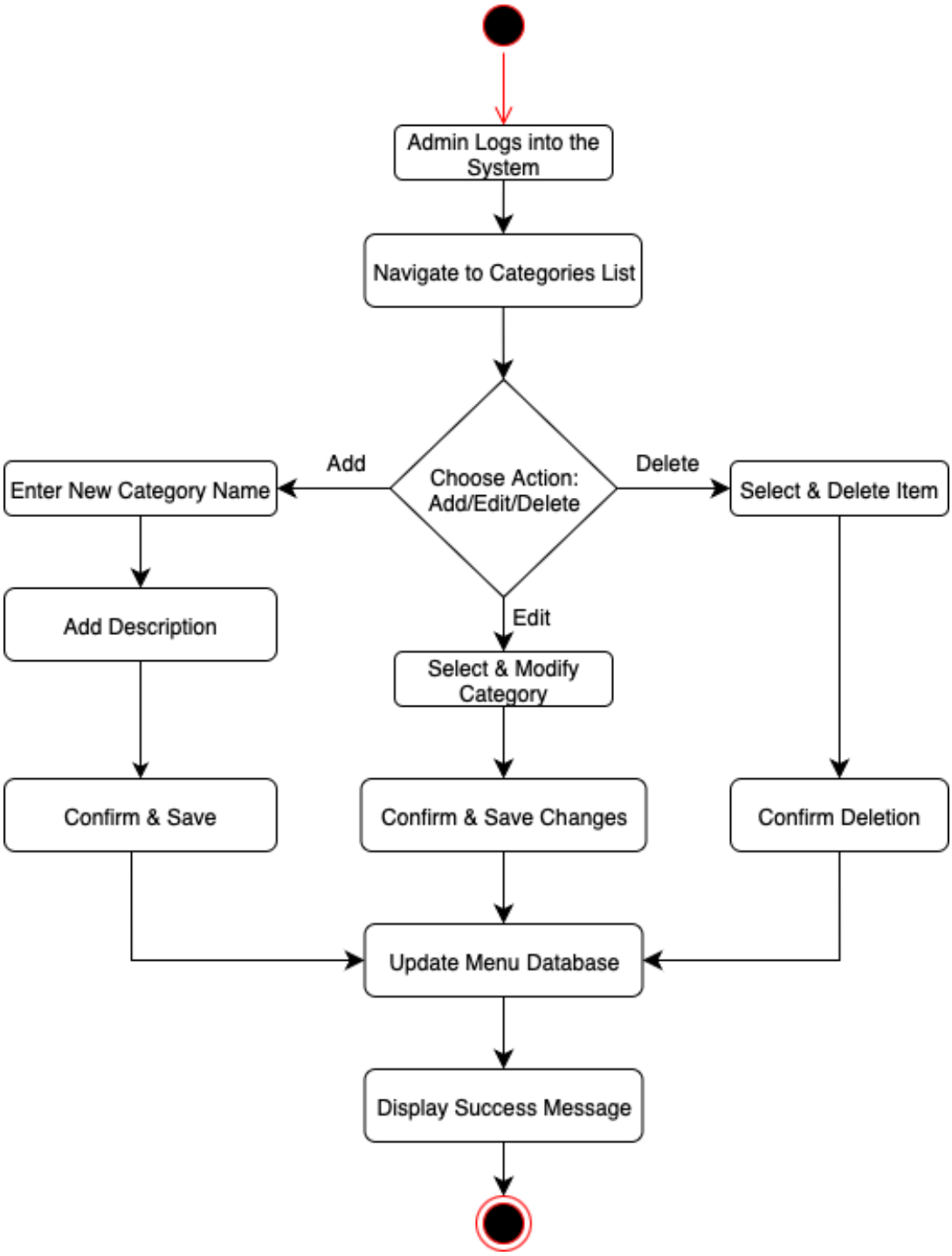


Figure 2—5 : Category List

- **Food List**

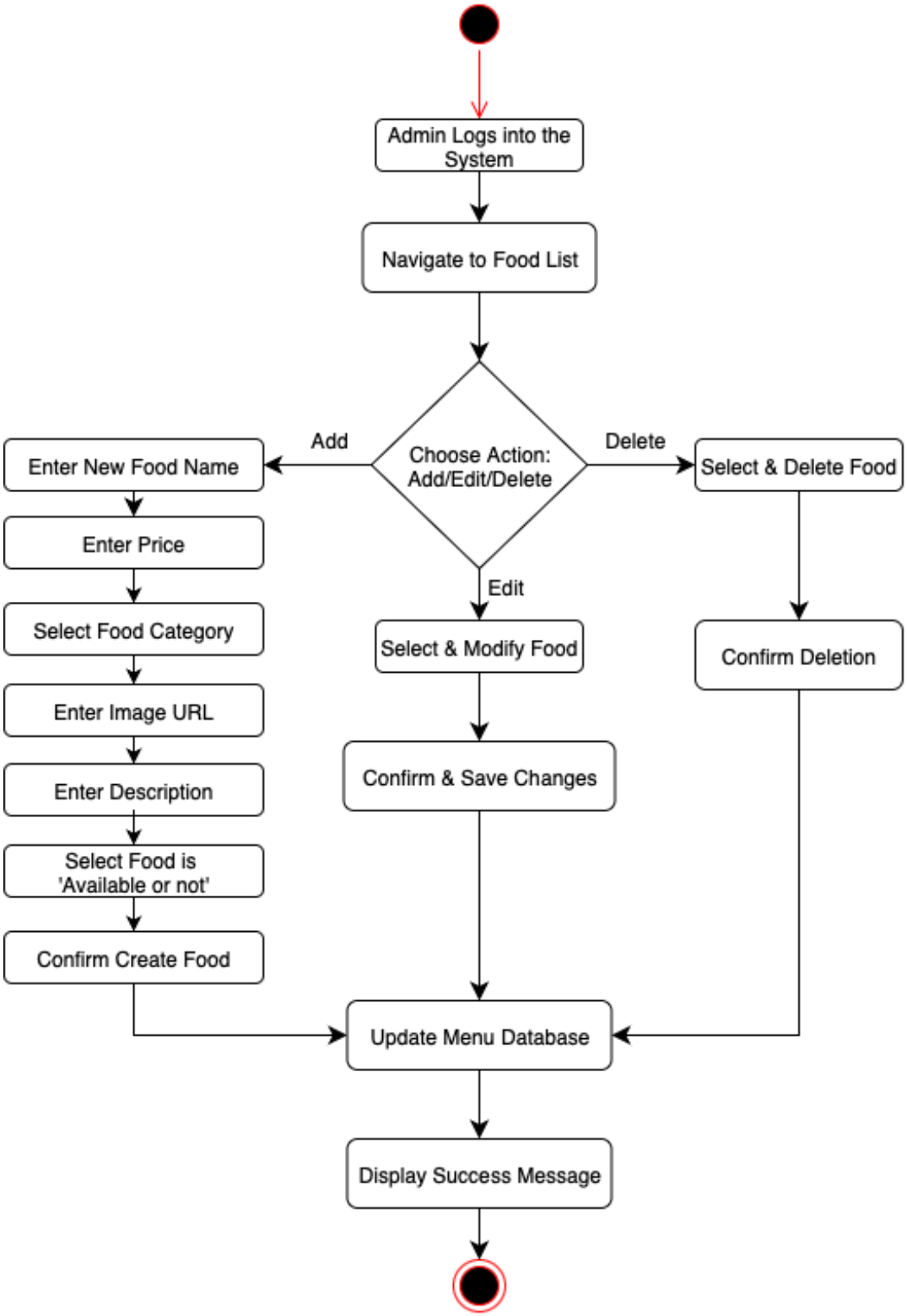


Figure 2—6 : Food List

- **Create Food**

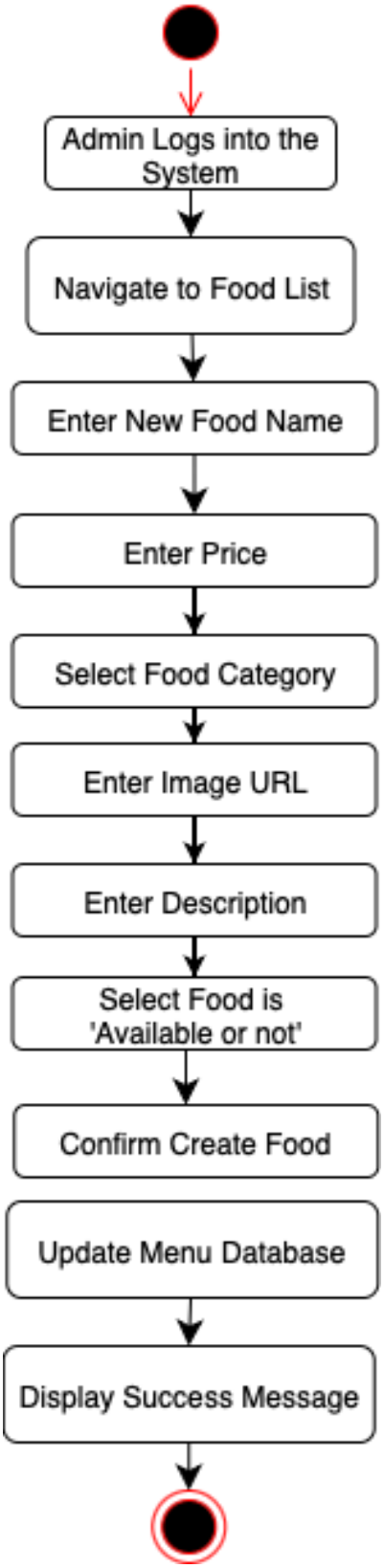


Figure 2—7 : Create Food

- **Create Table**

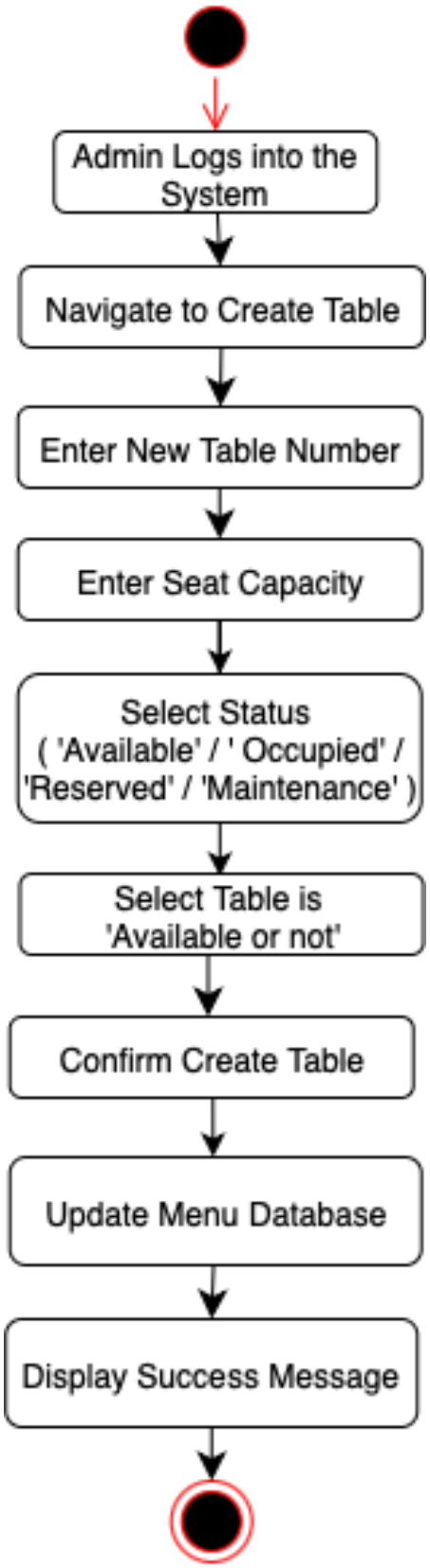


Figure 2—8 : Create Table

- **Table List**

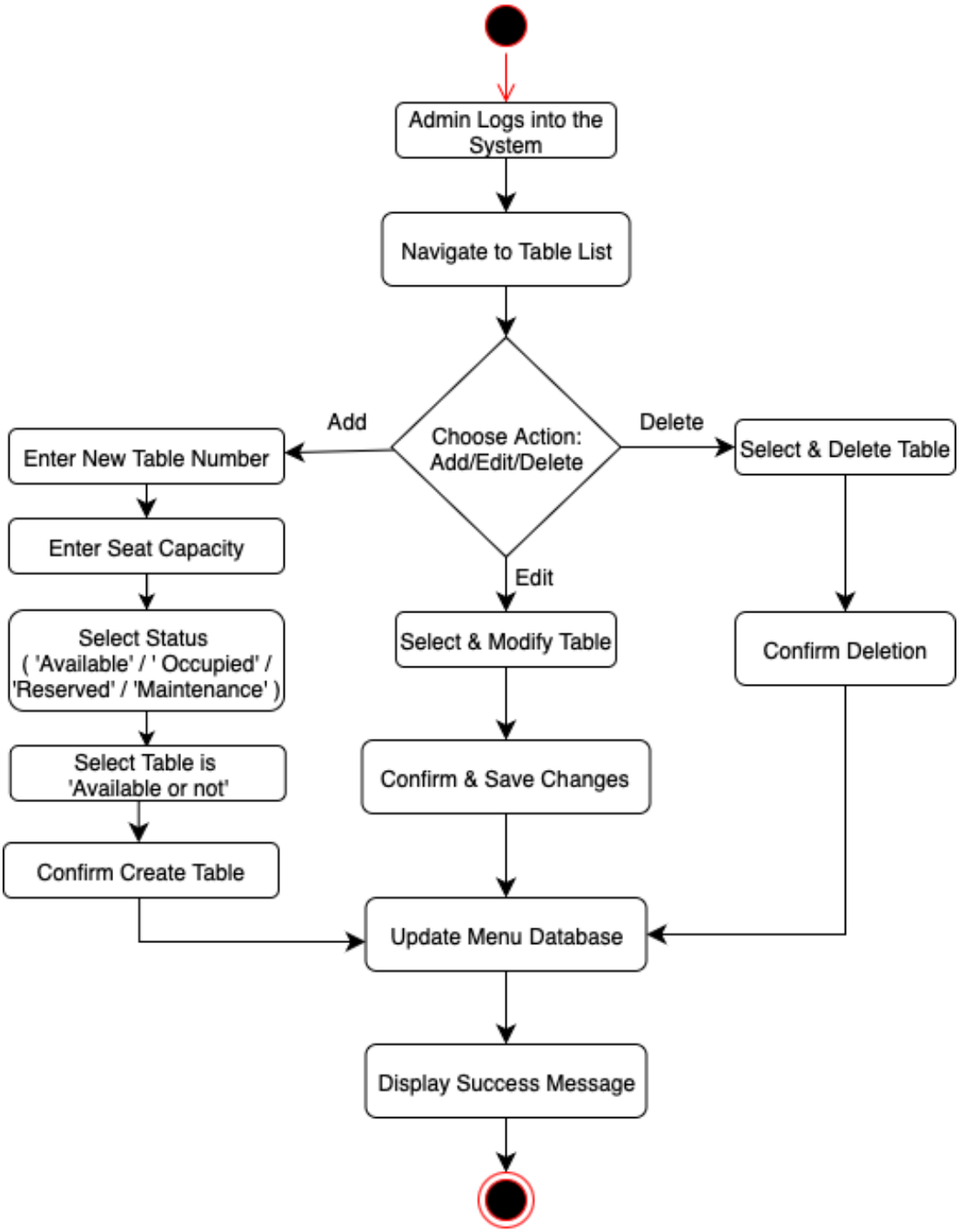


Figure 2—9 : Table List

- **Create Order**

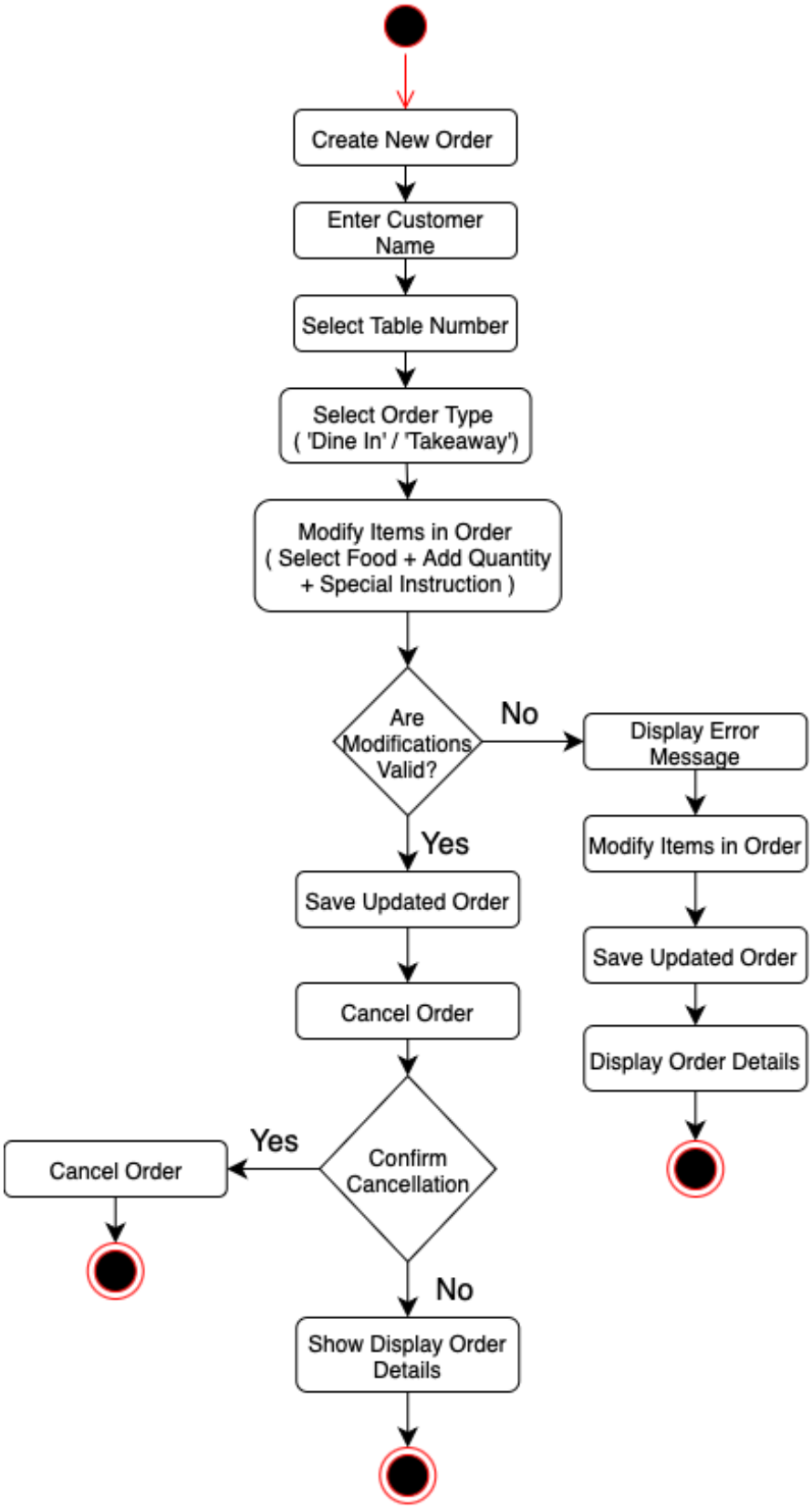


Figure 2—10 : Create Order

- **Order List**

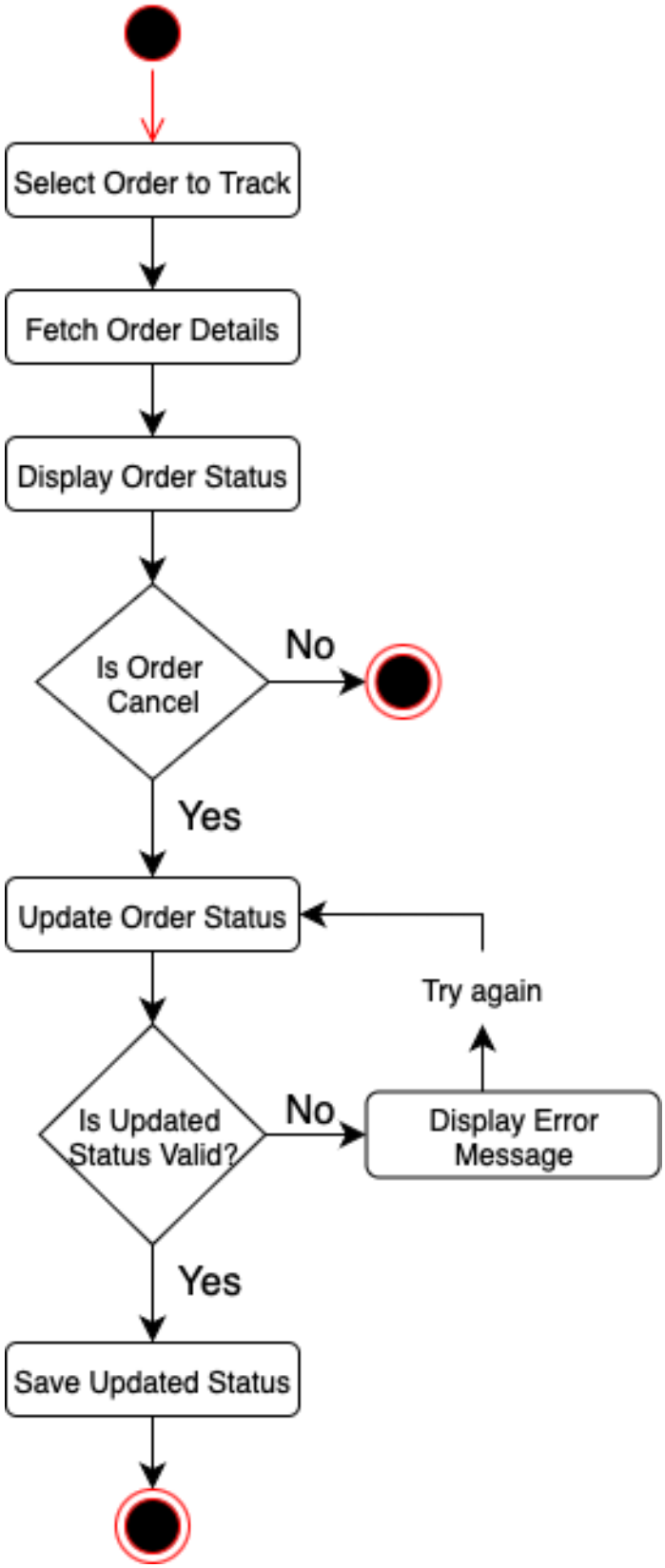


Figure 2—11 : Order List

- **Order Queue**

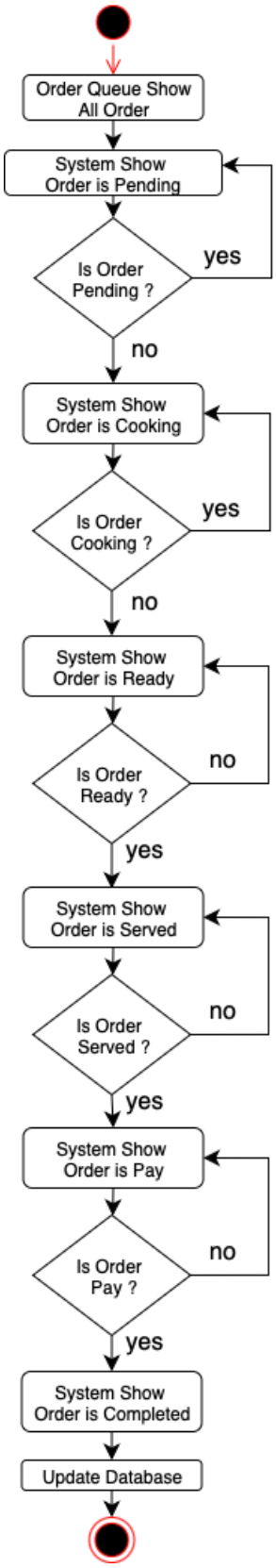


Figure 2—12 : Order Queue

- **Income Report**

Figure 2—13 : Income Report

- **Search Bar**

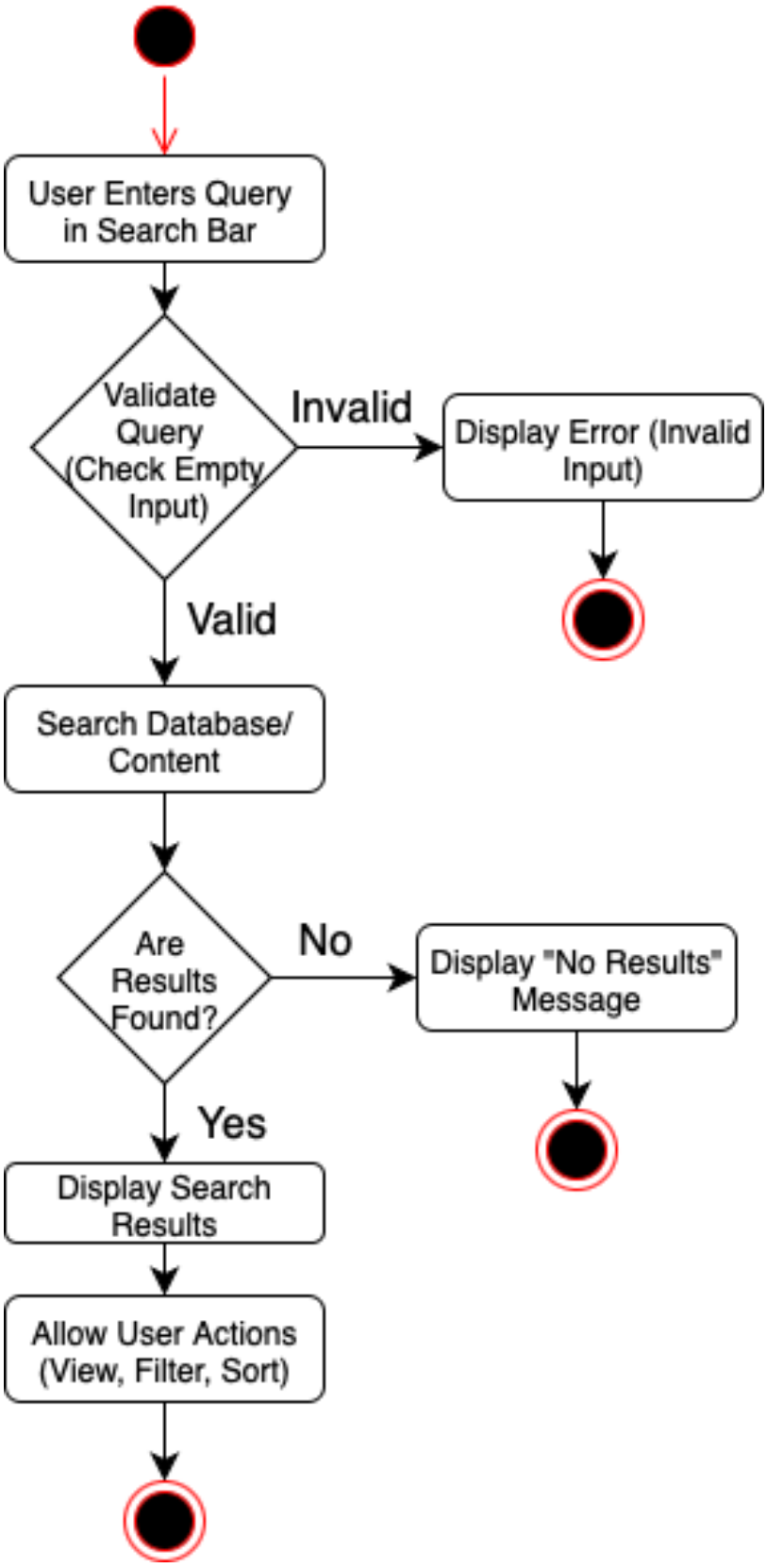


Figure 2—14 : Search Bar

- **Chat with AI Assistant**

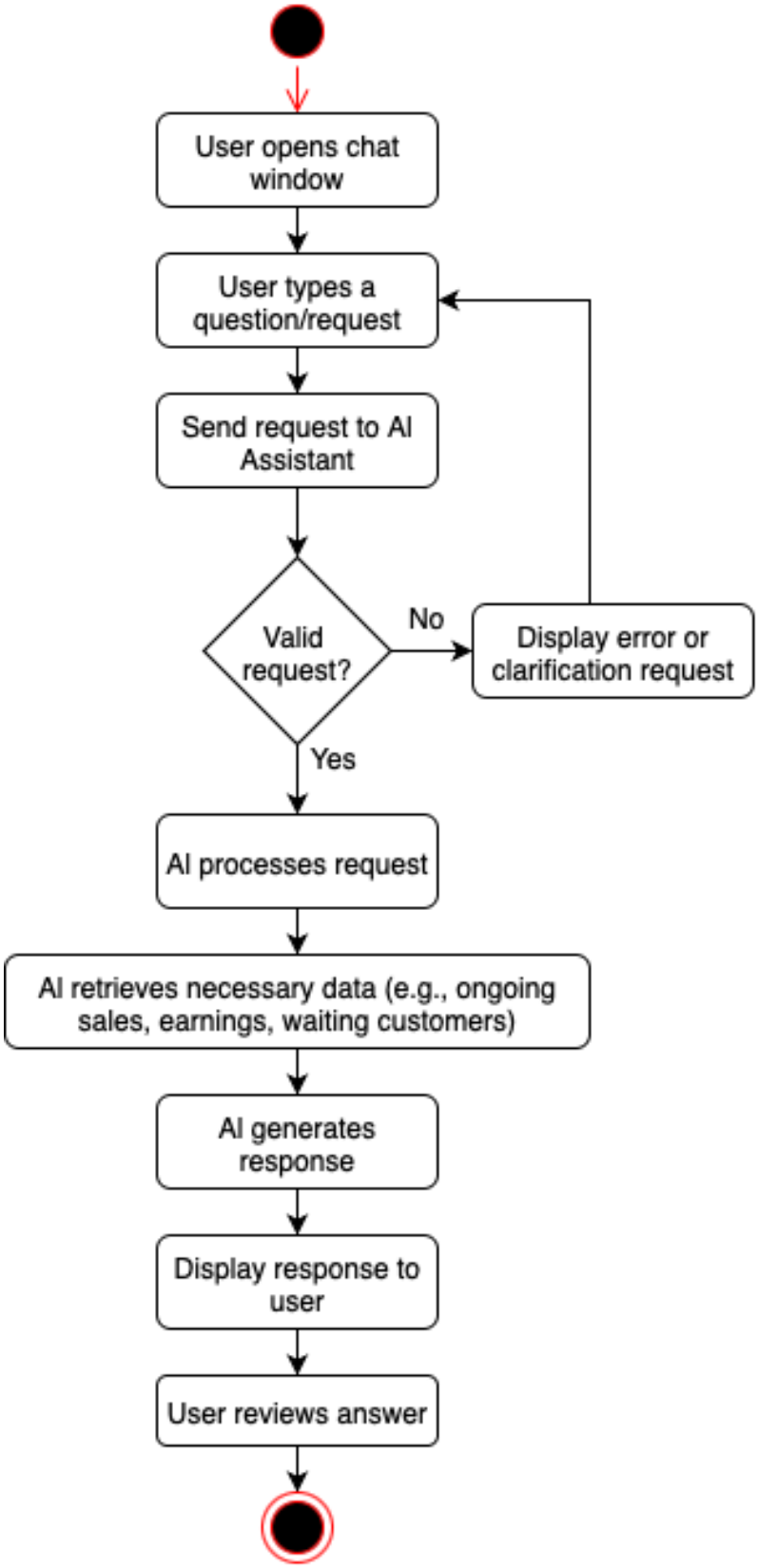


Figure 2—15 : Chat with AI Assistant

D. Sequence Diagram

- **Registration**

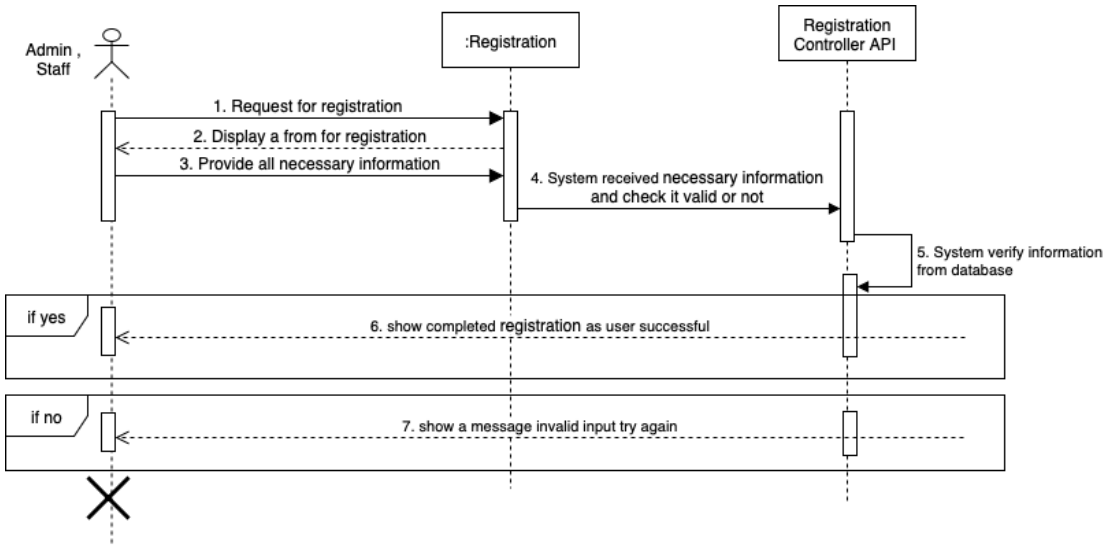


Figure 2—16 : Registration

- **Login**

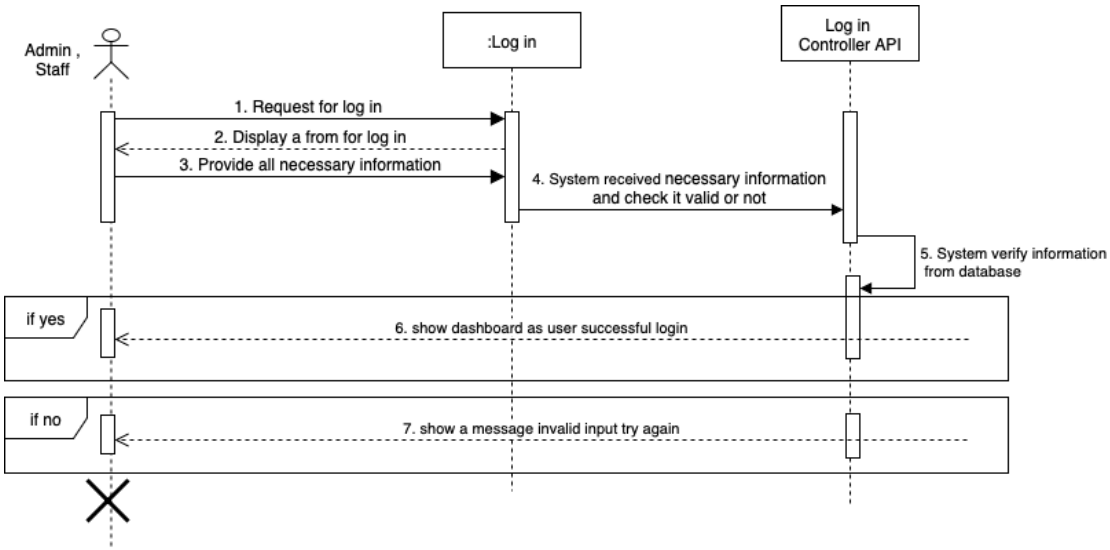


Figure 2—17 : Login

- **Create Category**

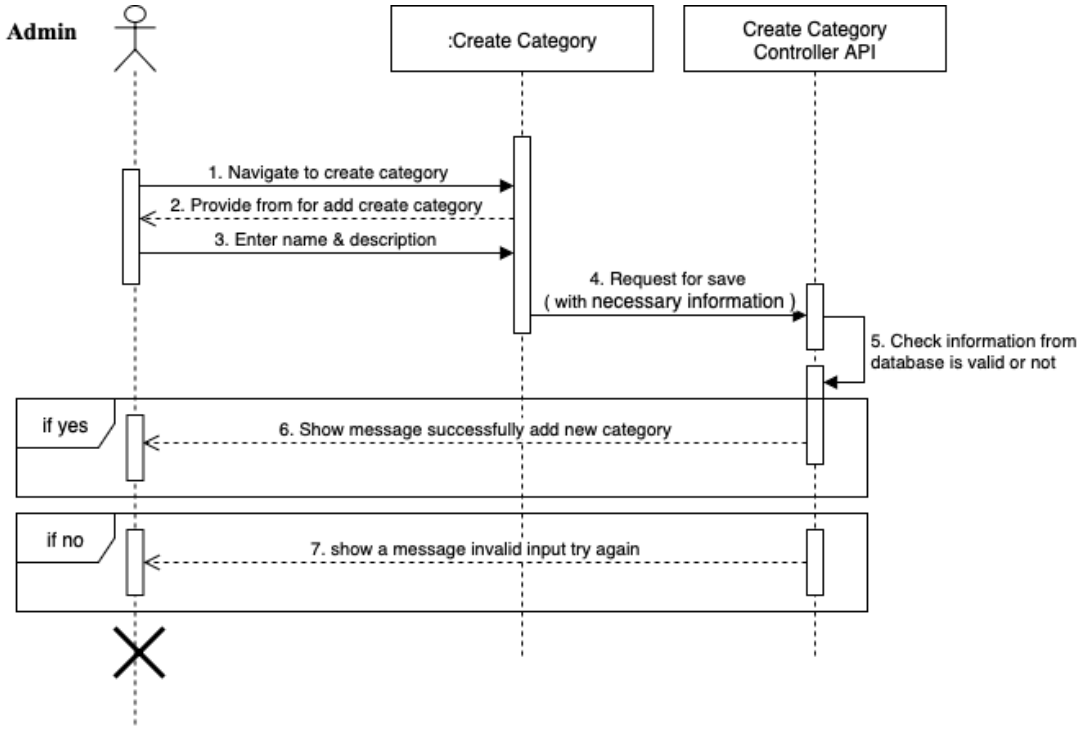


Figure 2—18 : Create Category

- **Category List**

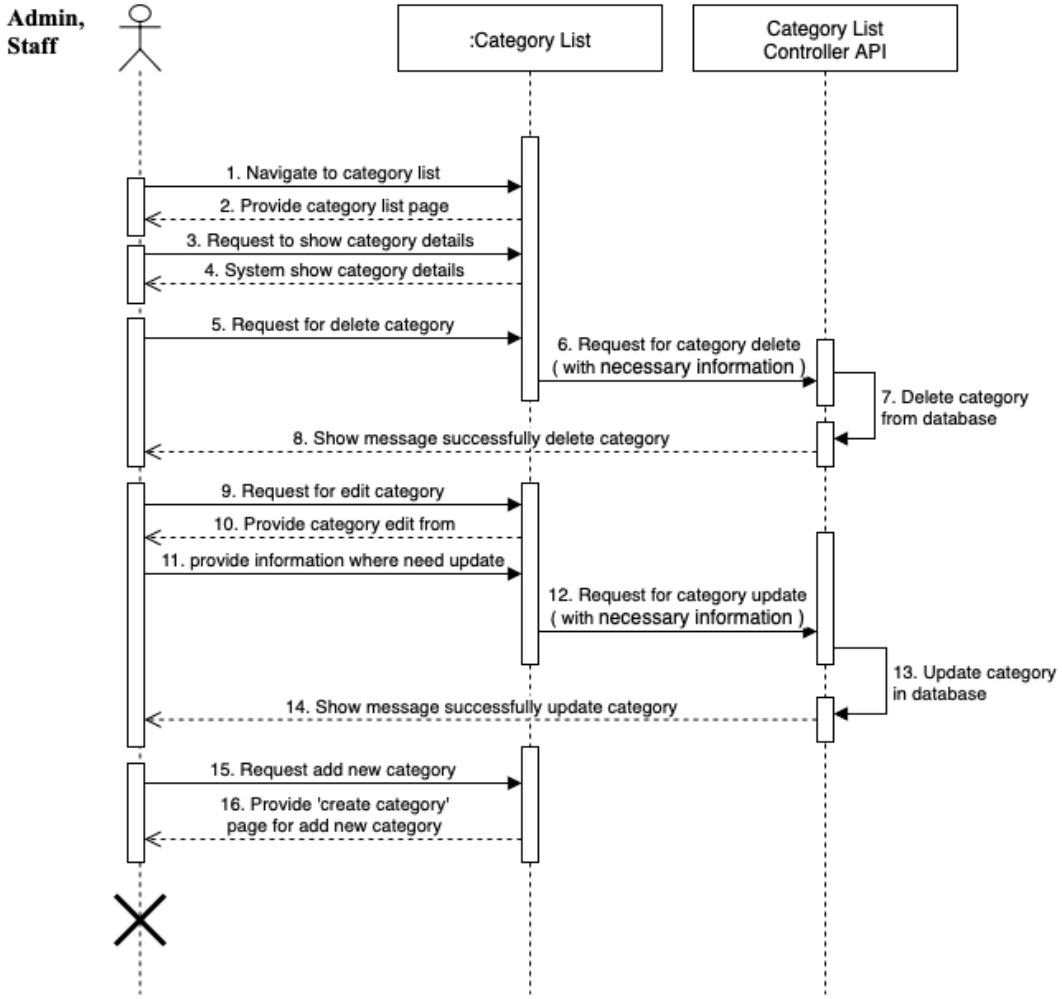


Figure 2—19 : Category List

- **Food List**

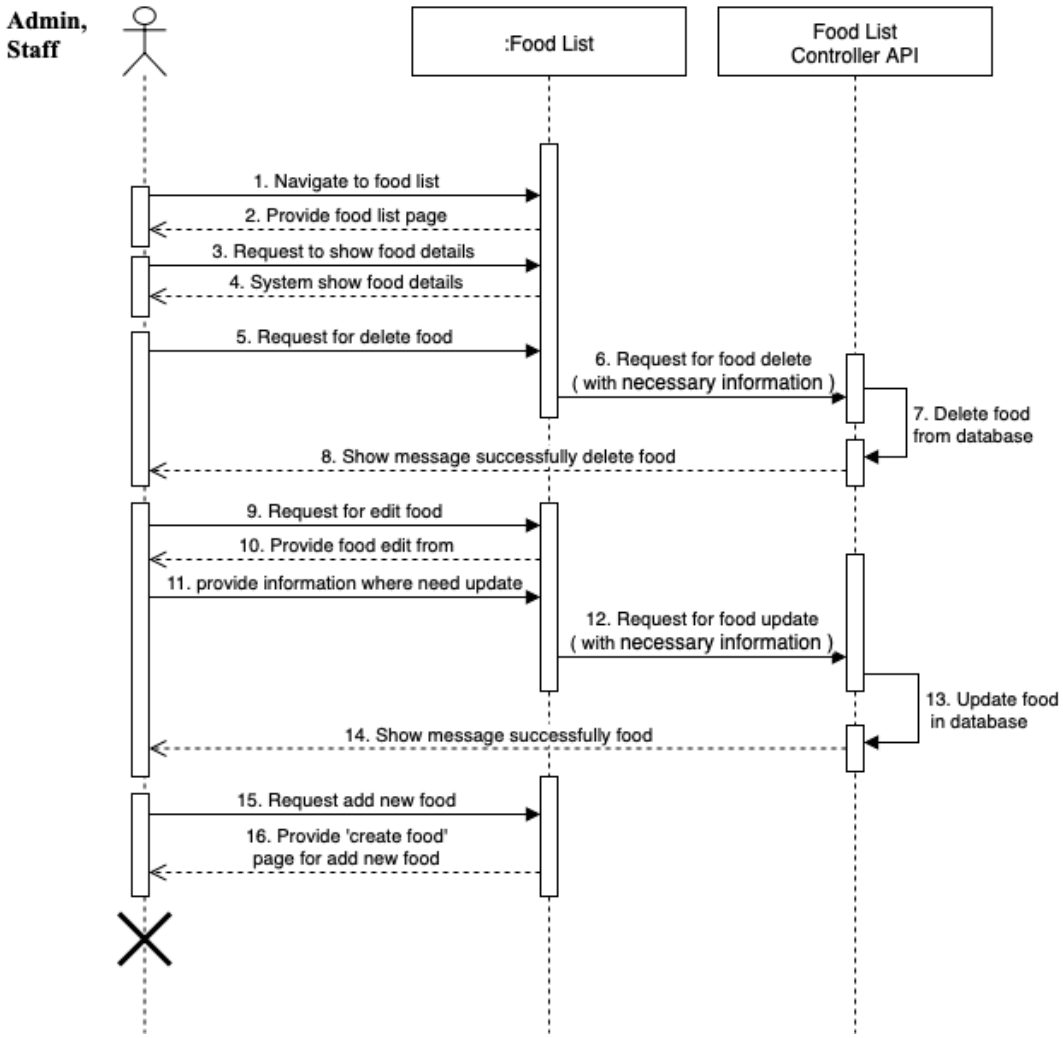


Figure 2—20 : Food List

- **Create Food**

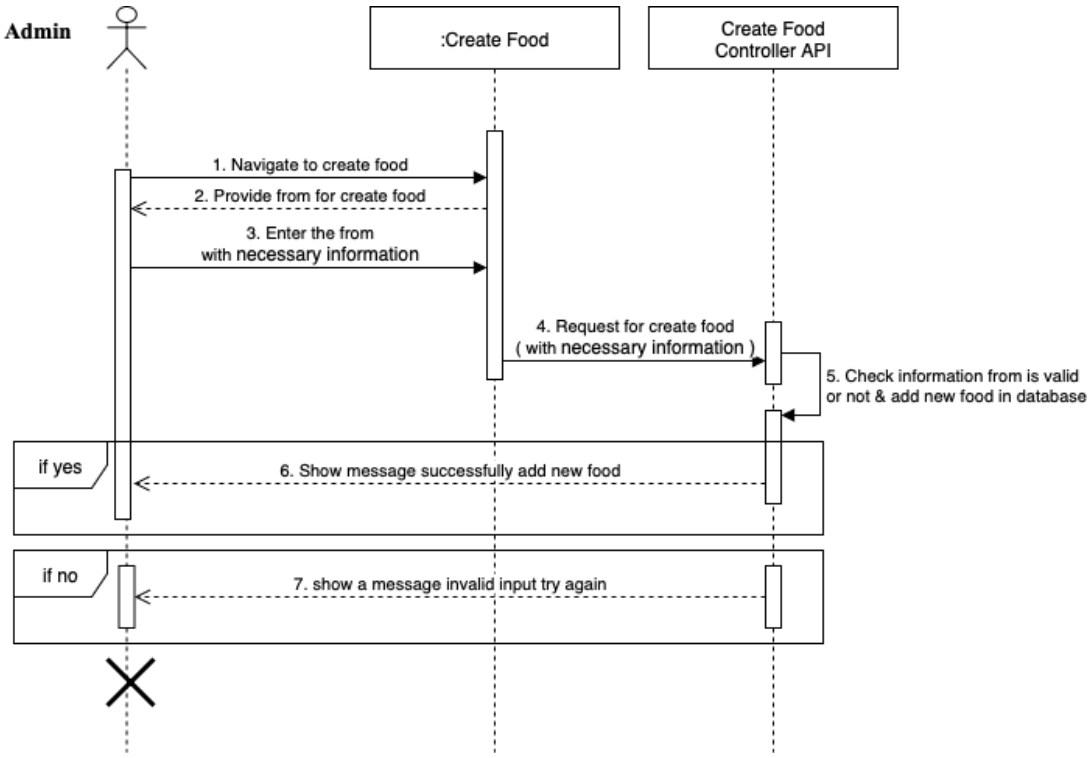


Figure 2—21 : Create Food

- **Create Table**

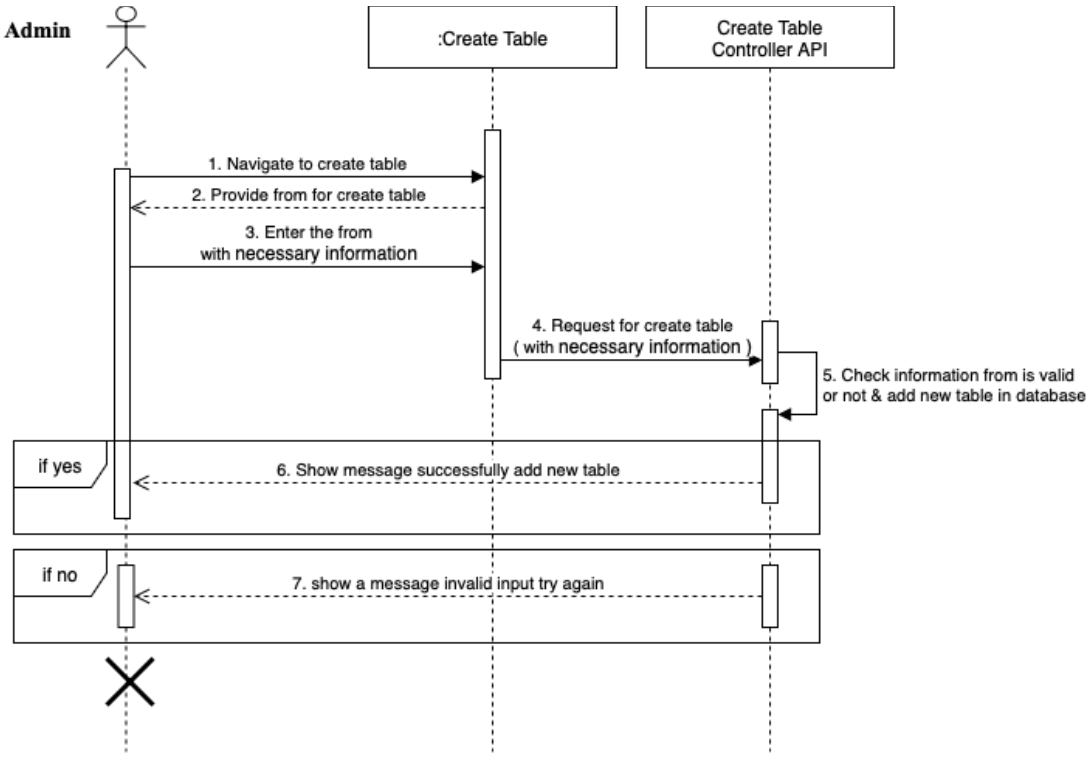


Figure 2—22 : Create Table

- **Table List**

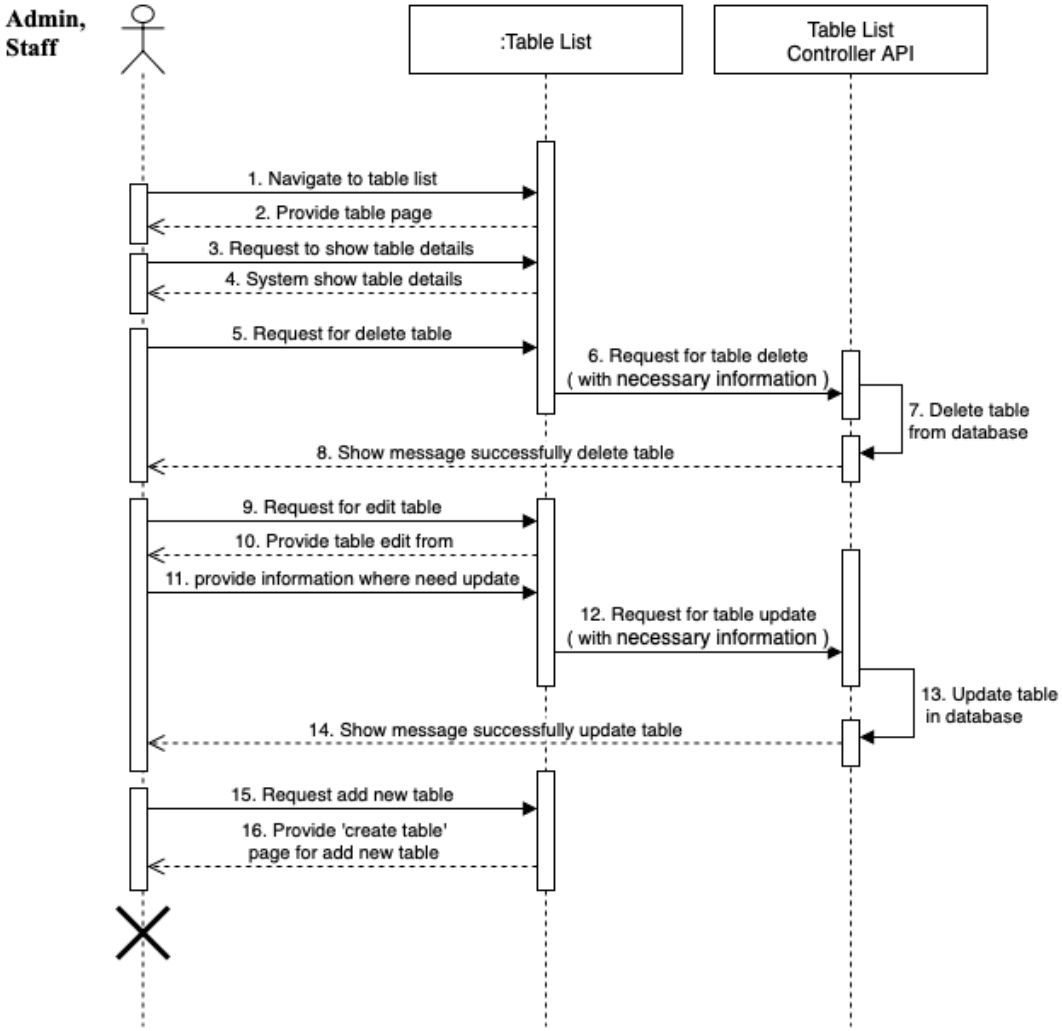


Figure 2—23 : Table List

- **Create Order**

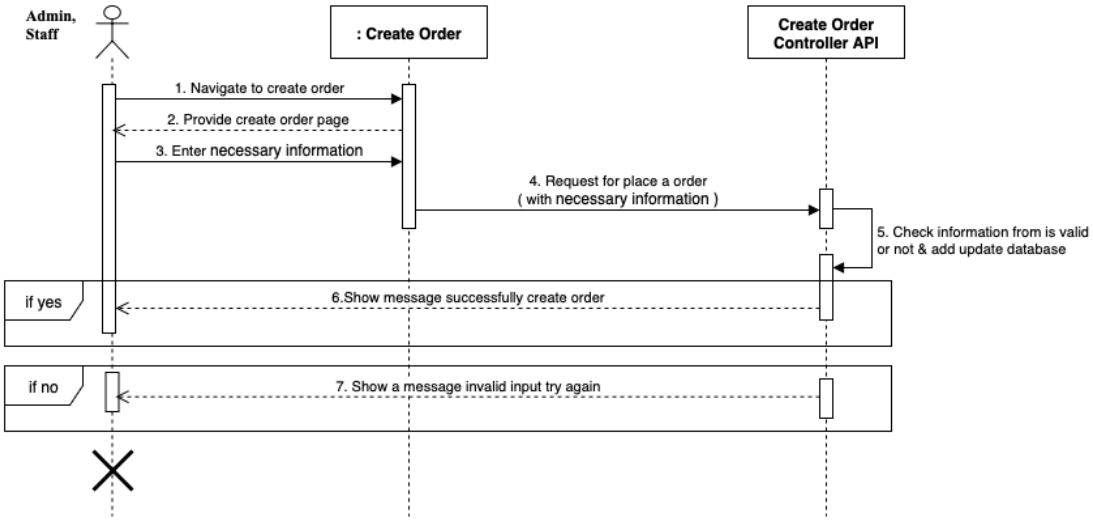


Figure 2—24 : Create Order

- **Order List**

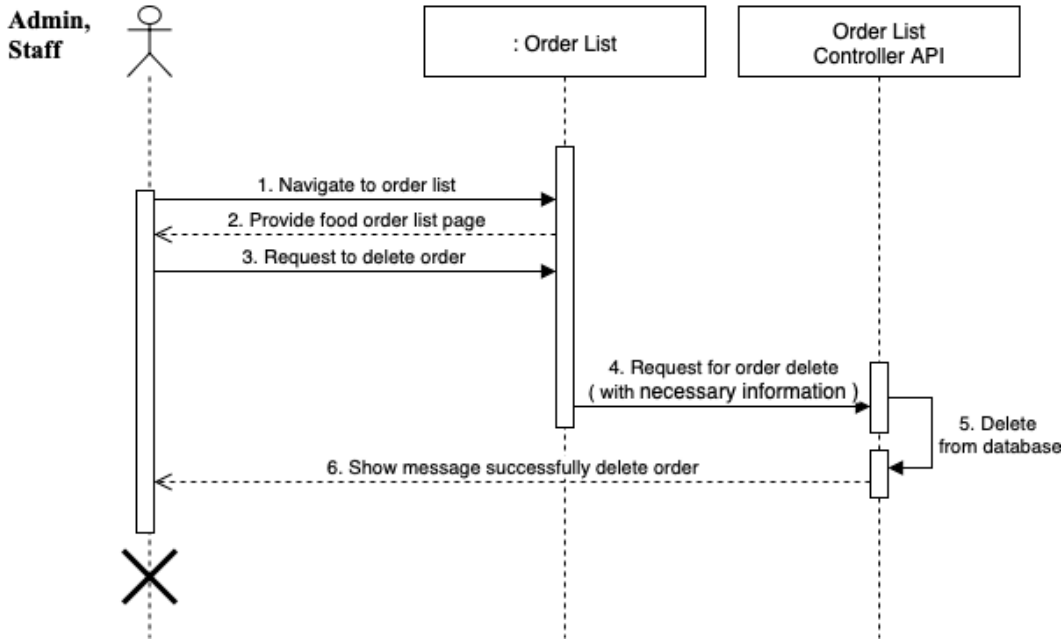


Figure 2—25 : Order List

- **Order Queue**

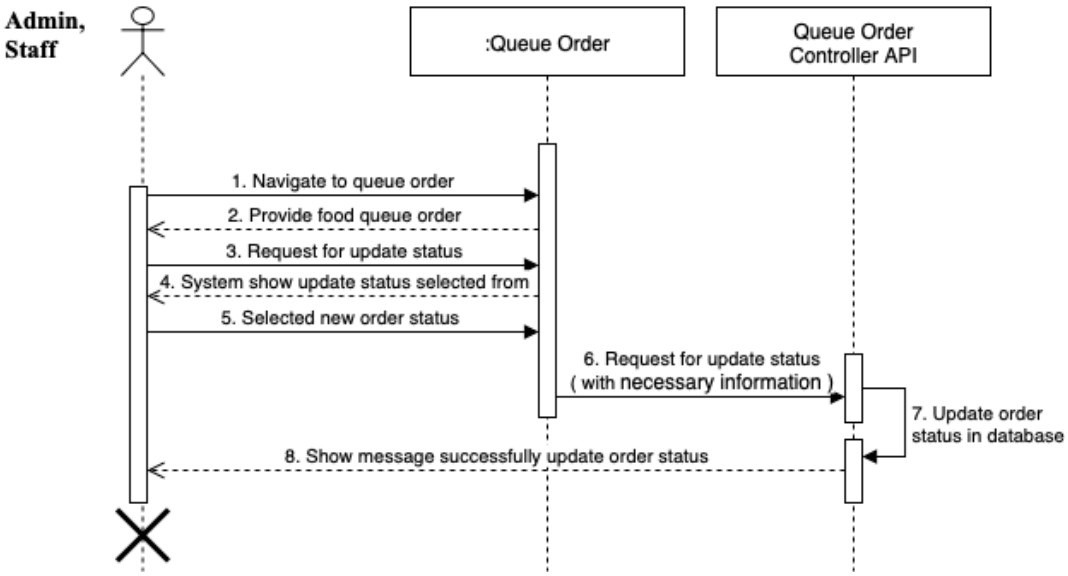


Figure 2—26 : Order Queue

- **Income Report**

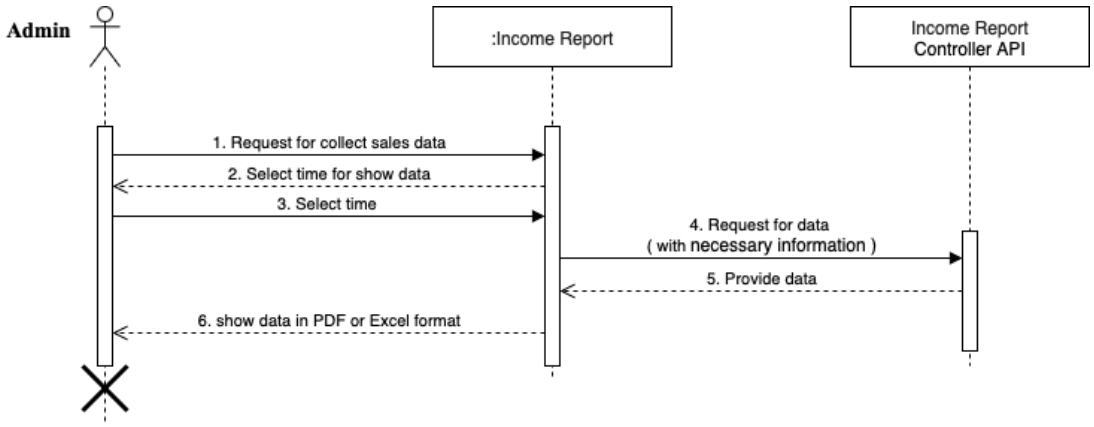


Figure 2—27 : Income Report

- **Search Bar**

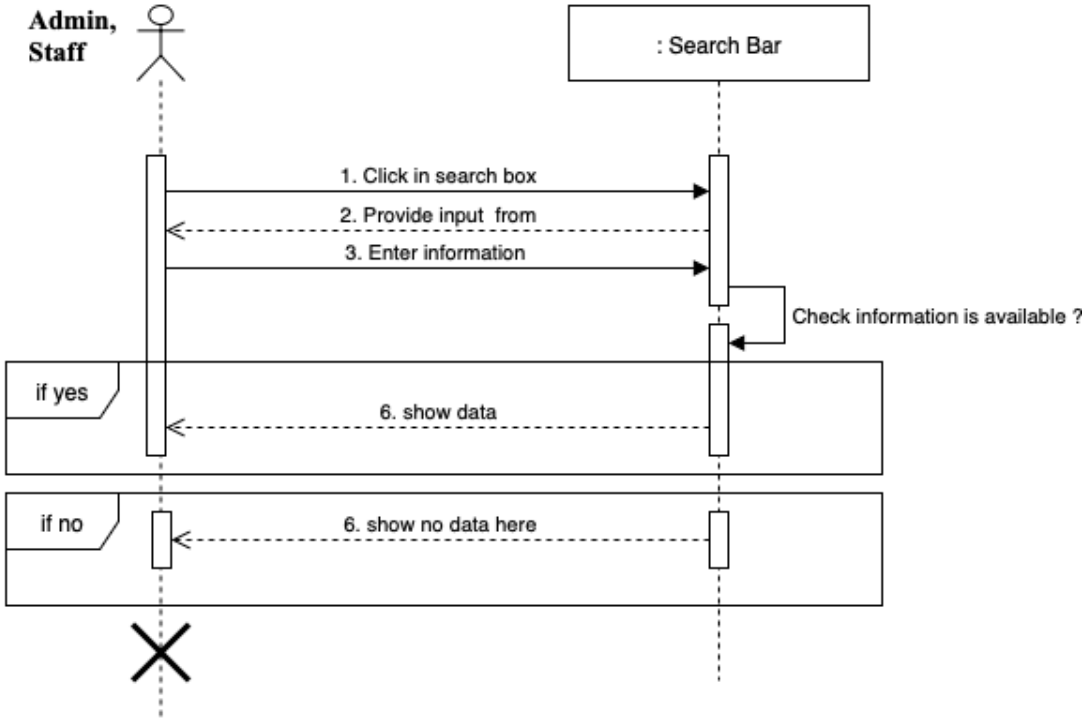


Figure 2—28 : Search Bar

- **Chat with AI Assistant**

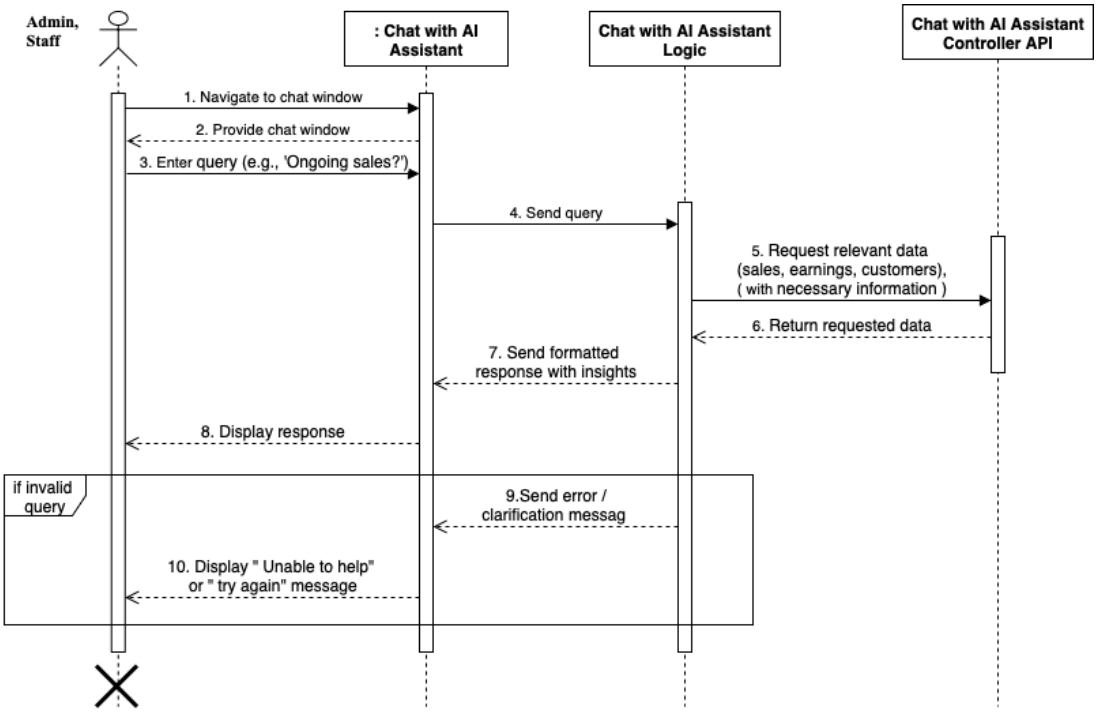


Figure 2—29 : Chat with AI Assistant

E. ER Diagram

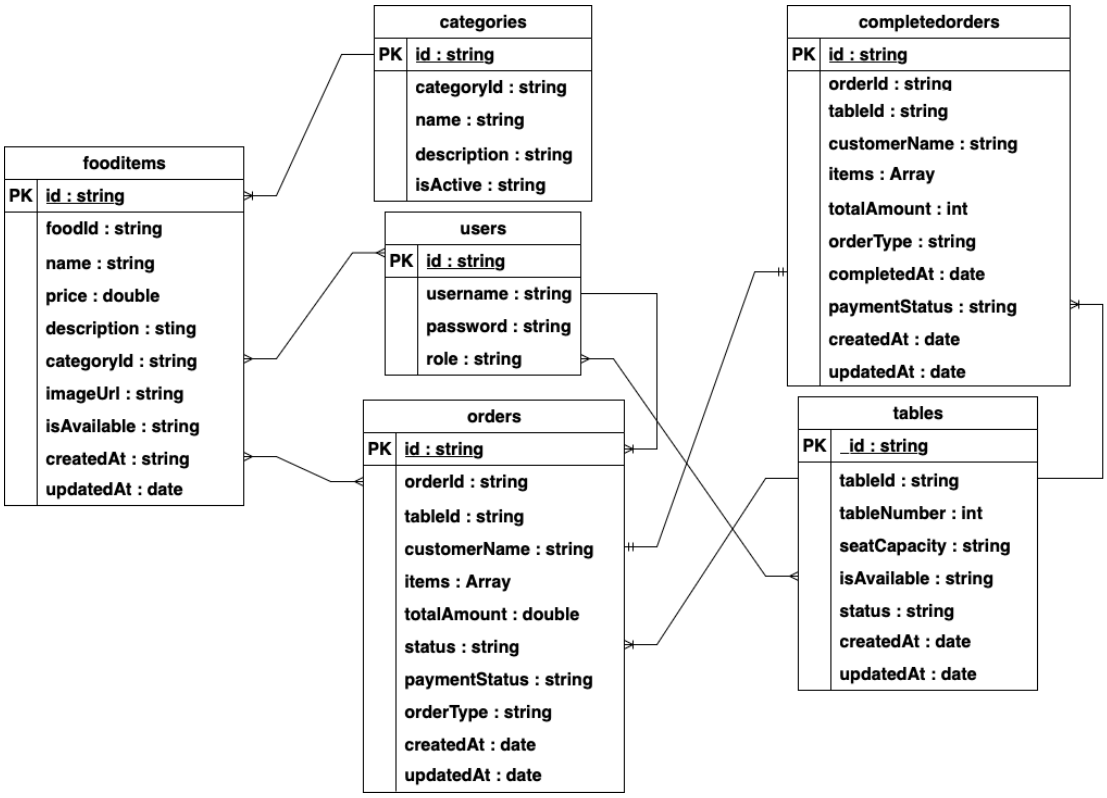


Figure 2—30 : ER Diagram

Here provides a comprehensive design framework for SmartPOS. It illustrates how requirements are translated into technical solutions, ensuring usability, performance, and security while supporting the long-term scalability of the system.

Chapter 3 : Software Testing

This chapter outlines the testing procedures conducted to validate the reliability and performance of SmartPOS. It includes unit, integration, and system testing approaches, supported by detailed test cases for critical functionalities.

3.1. Testing Features.

A. Registration
B. Login
C. Create Category
D. Category List
E. Food List
F. Create Food
G. Create Table
H. Table List
I. Create Order
J. Order List
K. Order Queue
L. Income Report
M. Chat with AI Assistant

3.2. Testing Strategies.

a. **Unit Testing:** Validated the correct rendering and behavior of individual React components in isolation to ensure functionality and stability.
b. **Integration Testing:** Tested the interaction between interconnected components, such as the Problem List and IDE Modal, to confirm seamless data flow and user experience.
c. **Manual End-to-End (E2E) Testing:** Executed comprehensive user workflow scenarios for both Student and Faculty roles to ensure full-system functionality from login to task completion.

3.3. System Testing.

- **Test Case A: Registration**

Table 3-1: Test Case for Registration

Test Case :A			Test Case Name: Registration			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description		If a user is using the system for the first time, they must register before accessing any features.				
Pre-Condition		• The user is not already registered in the system. • The registration page is accessible and properly loaded.				
Test Id	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Status
T01	Username Field	1.Open The Site. 2.Click The Registration Button. 3.Click Username Field 4.Fill Up Username Field	monirullah2	Successfully Fill Up	Successfully Fill Up	Pass
T02	Password Field	1.Open The Site. 2.Click The Registration Button. 3.Click Password Field 4.Fill Up Password Field	password123	Successfully Fill Up	Successfully Fill Up	Pass
T03	Role Field	1.Open The Site. 2.Click The Registration Button. 3.Click Role Field 4.Fill Up Role Field	Select Manager	Successfully Select	Successfully Select	Pass
T04	Submit Button	.Open The Site. 2.Click The Registration Button. 3.Click Submit Button 4.Fill Up Submit Button	Click	Successfully Registered	Successfully Registered	Pass
T05	Username Field	1.Open The Site. 2.Click The Registration Button. 3.Click Username Field 4.Fill Up Username Field	monirullah2	Fail Fill Up	Fail Fill Up	Pass
T06	Go Login Page	1.Open The Site. 2.Click The Registration Button. 3.Click 'Login Page'	Click	Show "User Login" Page	Show "User Login" Page	Pass

- **Test Case B: Login**

Table 3-2: Test Case Login

Test Case :B			Test Case Name: Login			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description		Verify that a registered user can log in with valid credentials.				
Pre-Condition		• User must be already registered in the system. • The login page must be accessible and properly loaded.				
Test Id	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Status
T07	Username Field	1.Open The Site. 2.Show Login Page. 3.Click Username Field 4.Fill Up Username Field	monirullah2	Successfully Fill Up	Successfully Fill Up	Pass
T08	Password Field	1.Open The Site. 2.Show Login Page. 3.Click Password Field 4.Fill Up Password Field	password123	Successfully Fill Up	Successfully Fill Up	Pass
T09	Login	1.Open The Site. 2.Click The Login Button. 3.Click Submit Button	Click	Successfully Login	Successfully Login	Pass
T10	Username Field	1.Open The Site. 2.Show Login Page. 3.Click Username Field 4.Fill Up Username Field	monir	Fail Fill Up	Fail Fill Up	Pass
T11	Password Field	1.Open The Site. 2.Show Login Page. 3.Click Password Field 4.Fill Up Password Field	123	Fail Fill Up	Fail Fill Up	Pass
T12	Login	1.Open The Site. 2.Click The Login Button. 3.Click Submit Button	Click	Fail Fill Up	Fail Fill Up	Pass

- **Test Case C: Create Category**

Table 3-3: Test Case Create Category

Test Case :C		Test Case Name: Create Category				
System: SmartPOS: A Web-Based Restaurant Management System		Test Priority:				
Created By: Monir Ullah		Date of Creation : 01/07/2025				
Performed By: Monir Ullah		Date of Review: 04/07/2025				
Description		Verify that an admin or authorized user can create a new food category successfully.				
Pre-Condition		• User is logged in with appropriate permissions (e.g., owner / manager). • The "Create Category" page or modal is accessible.				
Test Id	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Status
T13	Category Name Field	1.Login As an Authorized User. 2.Click 'Create Category' Button. 3.Fill Up Category Name Field	Desserts	Successfully Fill Up	Successfully Fill Up	Pass
T14	Category Description Field	1.Login As an Authorized User. 2.Click 'Create Category' Button. 3.Fill Up Category Name Field	Sweet delicacies served at the end of meals, including cakes, pastries, ice creams, puddings, and traditional sweet treats. Perfect for satisfying your sweet cravings and providing a delightful end to any dining experience.	Successfully Fill Up	Successfully Fill Up	Pass
T15	Create Category	1.Login As an Authorized User. 2.Click 'Create Category' Button. 3. Submit By Click 'Create Category' Button.	Click	Successfully Create Category	Successfully Create Category	Pass

- **Test Case D: Category List**

Table 3-4: Test Case for Category List

Test Case :D			Test Case Name: Category List			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description	Verify that the system displays a list of all existing food categories.					
Pre-Condition	• User is logged in. • The "Categories" page or section is accessible.					
Test Id	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Status
T16	Category Field Edit	1.Login As an Authorized User. 2.Click 'Category List' Button. 3.Click 'Edit' Button. 4.Fill Up Update Category Field	Change 'Appetizer 85' to "Appetizer 90'	Successfully Fill Up	Successfully Fill Up	Pass
T17	Category Details	1.Login As an Authorized User. 2.Click 'Category List' Button. 3.Click 'Details' Button.	Click	Show Details	Show Details	Pass
T18	Category Delete	1.Login As an Authorized User. 2.Click 'Category List' Button. 3.Click 'Delete' Button.	Click	Successfully Delete Category	Successfully Delete Category	Pass

- **Test Case E: Food List**

Table 3-5: Test Case for Food List

Test Case :E			Test Case Name: Food List			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description	Verify that the system displays all food items under their respective categories.					
Pre-Condition	• User is logged in. • At least one food item is added in the system. • The "Food List" page or section is accessible.					
Test Id	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Status
T19	Food List Edit	1.Login As an Authorized User. 2.Click 'Food List' Button. 3.Click 'Edit' Button. 4.Fill Up Update Food Field	Change Price '380' to "390'	Successfully Fill Up	Successfully Fill Up	Pass
T20	Food Details	1.Login As an Authorized User. 2.Click 'Food List' Button. 3.Click 'Details' Button.	Click	Show Details	Show Details	Pass
T21	Food Delete	1.Login As an Authorized User. 2.Click 'Food List' Button. 3.Click 'Delete' Button.	Click	Successfully Delete Category	Successfully Delete Category	Pass
T22	Food Search	1.Login As an Authorized User. 2.Click 'Food Search' Button. 3.Enter Food Name. 4. Click Enter	Updated Chicken Biryani	Successfully Find Food	Successfully Find Food	Pass

- **Test Case F: Create Food**

Table 3-6: Test Case for Create Food

Test Case :F			Test Case Name: Create Food			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description		Verify that an authorized user can successfully add a new food item to the system.				
Pre-Condition		• User is logged in with appropriate permissions (e.g., owner / manager). • At least one food category exists. • The "Create Food" form/page is accessible.				
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T23	Create Food	1.Login As an Authorized User. 2.Click 'Create Food' Button. 3.Click 'Food Name Field' 4.Fill Up 'Food Name' Field	Chicken Biryni	Successfully Fill Up	Successfully Fill Up	Pass
T24	Food Price	1.Login As an Authorized User. 2.Click 'Create Food' Button. 3.Click 'Food Price Field' 4.Fill Up 'Food Price' Field	350.00	Successfully Fill Up	Successfully Fill Up	Pass
T25	Select Food Category	1.Login As an Authorized User. 2.Click 'Create Food' Button. 3.Click 'Category Field' 4.Select Food Category	Click 'Main Course'	Successfully Select Category	Successfully Select Category	Pass
T26	Add Food Description	1.Login As an Authorized User. 2.Click 'Create Food' Button.	Premium basmati rice cooked with tender chicken and	Successfully Find Food	Successfully Find Food	Pass

		3.Click 'Description Field' 4.Fill Up 'Description' Field	aromatic spices, served with raita.			
T27	Add Image URL	1.Login As an Authorized User. 2.Click 'Create Food' Button. 3.Click 'Image URL' 4.Fill Up 'Image URL' Field	Food Link	Successfully Create Food	Successfully Create Food	Pass

- **Test Case G: Create Table**

Table 3-7: Test Case for Create Table

Test Case :G			Test Case Name: Create Table			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description		Verify that an authorized user can create a new table for customer seating or order assignment.				
Pre-Condition		• User is logged in with appropriate permissions (e.g., owner / manager). • The "Create Table" page or interface is accessible.				
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T28	Create Table	1.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Table Number' Field 4.Fill Up 'Table Number' Field	8	Successfully Fill Up	Successfully Fill Up	Pass
T29	Seat Capacity	11.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Seat Capacity' Field 4.Fill Up 'Seat Capacity' Field	12	Successfully Fill Up	Successfully Fill Up	Pass
T30	Is Available	1.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Is Available' Field	Click 'Yes'	Successfully Select 'Yes'	Successfully Select 'Yes'	Pass
T31	Status Select	1.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Status Field' 4. Select 'Available' Field	Click 'Available'	Successfully Select 'Status'	Successfully Select 'Status'	Pass

- **Test Case H: Table List**

Table 3-8: Test Case for Create List

Test Case :H			Test Case Name: Create List			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description		Verify that a user can create a new list, such as a menu list or custom order list.				
Pre-Condition		• User is logged in. • The 'Create List' interface is accessible.				
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T32	Table List Edit	1.Login As an Authorized User. 2.Click 'Table List' Button. 3.Click 'Edit' Button 4.Fill Up Update Field	8	Successfully Fill Up	Successfully Fill Up	Pass
T33	Seat Capacity	1.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Seat Capacity' Field 4.Fill Up 'Seat Capacity' Field	12	Successfully Fill Up	Successfully Fill Up	Pass
T34	Is Available	1.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Is Available' Field	Click 'Yes'	Successfully Select 'Yes'	Successfully Select 'Yes'	Pass
T35	Status Select	1.Login As an Authorized User. 2.Click 'Create Table' Button. 3.Click 'Status Field' 4. Select 'Available' Field	Click 'Available'	Successfully Select 'Status'	Successfully Select 'Status'	Pass

- **Test Case I: Create Order**

Table 3-9: Test Case for Create Order

Test Case :I			Test Case Name: Create Order			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description		Verify that a user can successfully create a new order by selecting food items and assigning them to a customer or table.				
Pre-Condition		• User is logged in. • Food items and tables are already created. • The "Create Order" page or POS interface is accessible.				
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T36	Customer Name	1.Login As an Authorized User. 2.Click 'Create Order' Button. 3.Click 'Customer Name' Field 4.Fill Up 'Customer Name' Field	Shamim Ahamed	Successfully Fill Up	Successfully Fill Up	Pass
T37	Table Number	1.Login As an Authorized User. 2.Click 'Create Order' Button. 3.Click 'Table Number' Field 4. Select Table	Click 'Table 10'	Successfully Select Table	Successfully Select Table	Pass
T38	Order Type	1.Login As an Authorized User. 2.Click 'Create Order' Button. 3.Click 'Order Type' Field 4. Select Order Type	Click 'Dine In'	Successfully Select 'Order Type'	Successfully Select 'Order Type'	Pass
T39	Add Food Item	1.Login As an Authorized User. 2.Click 'Create Order' Button. 3.Click 'Add Food Item' Field 4.Select Food Items	Click 'Chicken Biryani' & 3-Quantity'	Successfully Select 'Food Item;	Successfully Select 'Food Item'	Pass

- **Test Case J: Order List**

Table 3-10: Test Case for Order List

Test Case :J		Test Case Name: Order List				
System: SmartPOS: A Web-Based Restaurant Management System		Test Priority:				
Created By: Monir Ullah		Date of Creation : 01/07/2025				
Performed By: Monir Ullah		Date of Review: 04/07/2025				
Description		Verify that the system displays a list of all placed orders with their details (e.g., items, status, table, time, payment, etc.).				
Pre-Condition		• User is logged in. • At least one order has been created in the system. • The "Order List" page or section is accessible.				
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T40	Show Order List	1.Login As an Authorized User. 2.Click 'Order List ' Button.	Click	Successfully Show Order List	Successfully Show Order List	Pass
T41	Delete Order	1.Login As an Authorized User. 2.Click 'Order List' Button. 3.Click 'Delete' Button	Click	Successfully Delete Order	Successfully Delete Order	Pass
T42	Search Food Items	1.Login As an Authorized User. 2.Click 'Order List ' Button. 3.Click 'Search' Field 4. Enter Customer name	Shamim Ahamed	Successfully Show Customer Order	Successfully Show Customer Order	Pass

- **Test Case K: Order Queue**

Table 3-11: Test Case for Order Queue

Test Case :K			Test Case Name: Order Queue			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description	Verify that the system displays active or pending orders in a queue, showing real-time status updates for kitchen or processing staff.					
Pre-Condition	• User is logged in. • User is logged in (e.g., kitchen staff or admin). • At least one order is placed and is in pending, preparing, or in-progress status. • The "Order Queue" view or dashboard is accessible.					
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T43	Show Order Queue	1.Login As an Authorized User. 2.Click 'Order Queue' Button.	Click	Successfully Show Order Queue	Successfully Show Order Queue	Pass
T44	Update Status	1.Login As an Authorized User. 2.Click 'Order Queue' Button. 3. Update Order Status	Click Change 'Pending' to 'Cooking'	Successfully Shoe Change Order Status	Successfully Shoe Change Order Status	Pass
T45	Search Food Items	1.Login As an Authorized User. 2.Click 'Order Queue' Button. 3.Click 'Search' Field 4. Enter Customer name	Monir Ullah	Successfully Show Customer Order	Successfully Show Customer Order	Pass

- **Test Case L: Income Report**

Table 3-12: Test Case for Income Report

Test Case :L			Test Case Name: Income Report			
System: SmartPOS: A Web-Based Restaurant Management System			Test Priority:			
Created By: Monir Ullah			Date of Creation : 01/07/2025			
Performed By: Monir Ullah			Date of Review: 04/07/2025			
Description	Verify that the system generates an accurate income report based on completed orders within a selected date range.					
Pre-Condition	• User is logged in with admin or manager role. • Completed (paid) orders exist in the system. • The "Income Report" page or module is accessible.					
Test Id	Test Steps	Test Data	Expected Results	Expected Results	Actual Results	Status
T46	Show Income Report	1.Login As an Authorized User. 2.Click 'Income Report' Button.	Click	Successfully Show Income Report	Successfully Show Income Report	Pass
T47	Select Income Report Range	1.Login As an Authorized User. 2.Click 'Income Report' Button. 3. Select Range	Click 'Monthly'	Successfully Show monthly Income Report	Successfully Show monthly Income Report	Pass

In conclusion, Chapter 3 confirms that SmartPOS has undergone rigorous testing to ensure accuracy and stability. The results demonstrate that the system performs as intended across diverse scenarios, reinforcing its suitability for real-world deployment.

Chapter 4 : Deployment and Maintenance

This chapter presents the deployment strategy for SmartPOS, detailing the use of continuous integration and delivery pipelines. It also outlines the maintenance plan, focusing on monitoring, error handling, and iterative improvements.

4.1. Software Release Life Cycle

Development Approach: The project followed an Agile, iterative methodology, allowing for rapid development, feedback incorporation, and continuous improvement across stages.

4.2. Project Lifecycle:

a. **Alpha Stage:** Set up the core backend infrastructure using Firebase Authentication and Fire-store Security Rules. Developed essential UI components for initial prototyping.
b. **Beta Stage:** Built the complete student problem-solving workflow and faculty dashboard. Shared this version with a small group of peers for early usability testing and feedback.
c. **Release Stage:** Resolved bugs identified during beta testing. Finalized and prepared the application for production deployment.

Deployment: The project is deployed on Vercel, with the GitHub repository's main branch connected via Continuous Deployment (CD).

Every git pushes to main triggers an automatic live build and deployment.

4.3. Maintenance & Monitoring:

Actively monitor Vercel and Firebase dashboards for:

a. Runtime errors
b. Performance issues
c. Usage spikes

Future updates are managed through feature branches and pull requests, ensuring code quality and application stability.

To conclude, Chapter 4 emphasizes the significance of structured deployment and proactive maintenance. By leveraging modern deployment platforms and agile methodologies, SmartPOS ensures long-term stability, adaptability, and sustainable performance.

Chapter 5 : User Manual

This user manual is intended for local environment usage with cloud or public deployment. It provides step-by-step instructions for installing, configuring, and operating the SmartPOS Web-Based Restaurant Management System on a local server using tools such as Visual Studio Code, Node.js, and MongoDB.

This guide is specifically designed for developers, testers, or users working in a development or testing environment.

5.1. Environment Setup

a. Install Visual Studio Code.

1. Go to the Visual Studio Code website: https://code.visualstudio.com/download
2. Click the appropriate download button for your operating system (Windows, macOS, or Linux) and install follow website instructions.

b. Install MongoDB Community Edition.

1. Go to: https://www.mongodb.com/try/download/community
2. Choose: Version: Latest
3. Click Download, then run the .msi installer.
4. In the wizard: Choose Complete setup.
5. Click Install, then Finish.

5.2. Pull Project from GitHub .

[notes: In VS Code (in split windows or two tabs)]

c. For Font-end steps:

1. Open VS Code.
2. Click View → Terminal (or press Ctrl + ~).

3. Clone the repository using this command:

bash
https://github.com/monir-ullah/SmartPOS-Restaurant-Management-System-Fontend

d. After cloning, open the folder:

1. Go to file → Open folder.
2. Select the cloning project folder name 'SmartPOS-Restaurant-Management-System-Fontend'.

e. For Back-end steps:

1. Open VS Code.
2. Click View → Terminal (or press Ctrl + ~).
3. Clone the repository using this command:

bash
https://github.com/monir-ullah/SmartPOS-Restaurant-Management-System-Backend

f. After cloning, open the folder:

1. Go to file → Open folder.
2. Select the cloning project folder name 'SmartPOS-Restaurant-Management-System-Backend.

5.3. Setup Front-end Back-end (API)

1. Open terminal in SmartPOS-Restaurant-Management-System-Fontend folder & SmartPOS-Restaurant-Management-System-Backend.
2. Install dependencies:

bash
npm install

3. Set up **.env** file (create if not exists) Example:

.env
PORT=5000 DATABASE_URL=mongodb+srv://mullah725:dY6I88s48xcdhHt9@cluster0.3ssla.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0 DEV_ENVIRONMENT=development Salt_Rounds=10 SECRET=f7a8b9c2d4e6g5h3i1j0k2l4m6n8p9q7r5s3t1u0v2w4x6y8z9

4. Run both Front-end and Back-end server

bash
npm run dev

5. Front-end not running? Follow this fix:

If the **frontend app does not run** or shows API connection errors, follow these steps:

6. Open this file in VS Code :

path
SmartPOS-Restaurant-Management-System-Fontend/src/redux/features/baseApi.ts

7. Find the following line:

baseApi.ts
baseUrl: "http://localhost:5000/api/v1/",

8. Change it to

baseApi.ts
baseUrl: "http://localhost:8000/api/v1/",

9. Save the file (Ctrl + S)

10. Restart the front-end:

```
bash
npm start
```

5.4. Connect to MongoDB Atlas (Remote)

1. Open MongoDB Compass.
2. Use the URL:

```
url
mongodb+srv://mullah725:dY6I88s48xcdhHt9@cluster0.3ssla.mongodb.net/?
```

3. Click **connect** → show the database.

5.5. SmartPOS Interface

- **User Login**

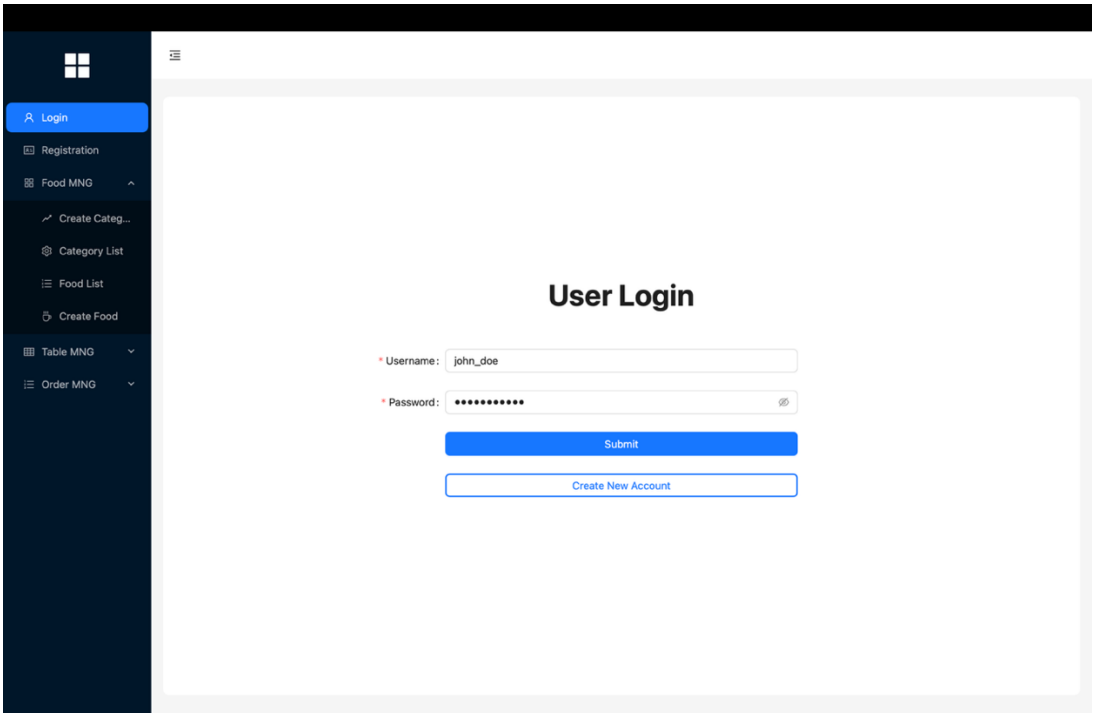


Figure 5—1 : User Login

- **User Registration**

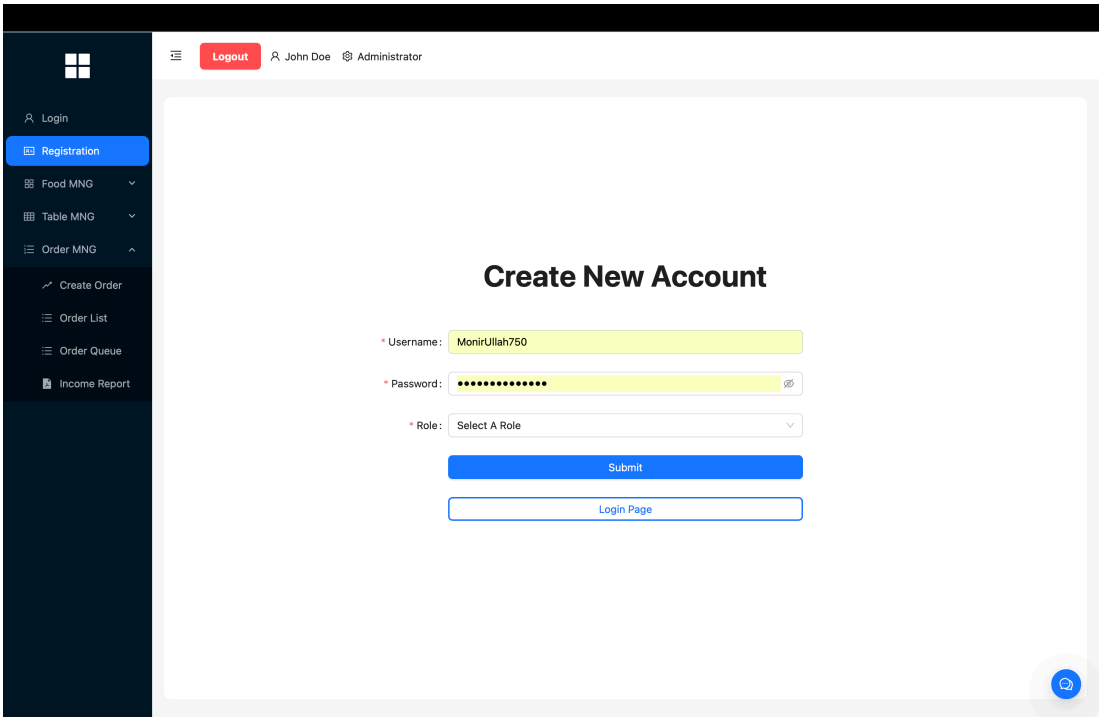


Figure 5—2 : User Registration

- **Dashboard**

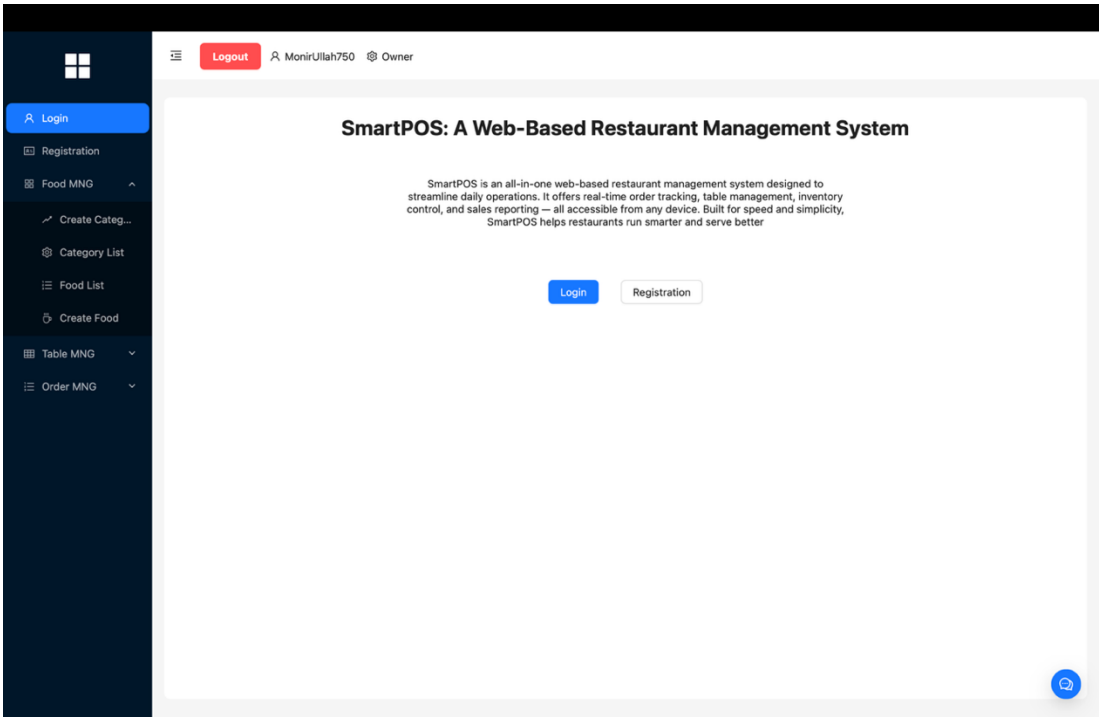


Figure 5—3 : Dashboard

- **Create Category**

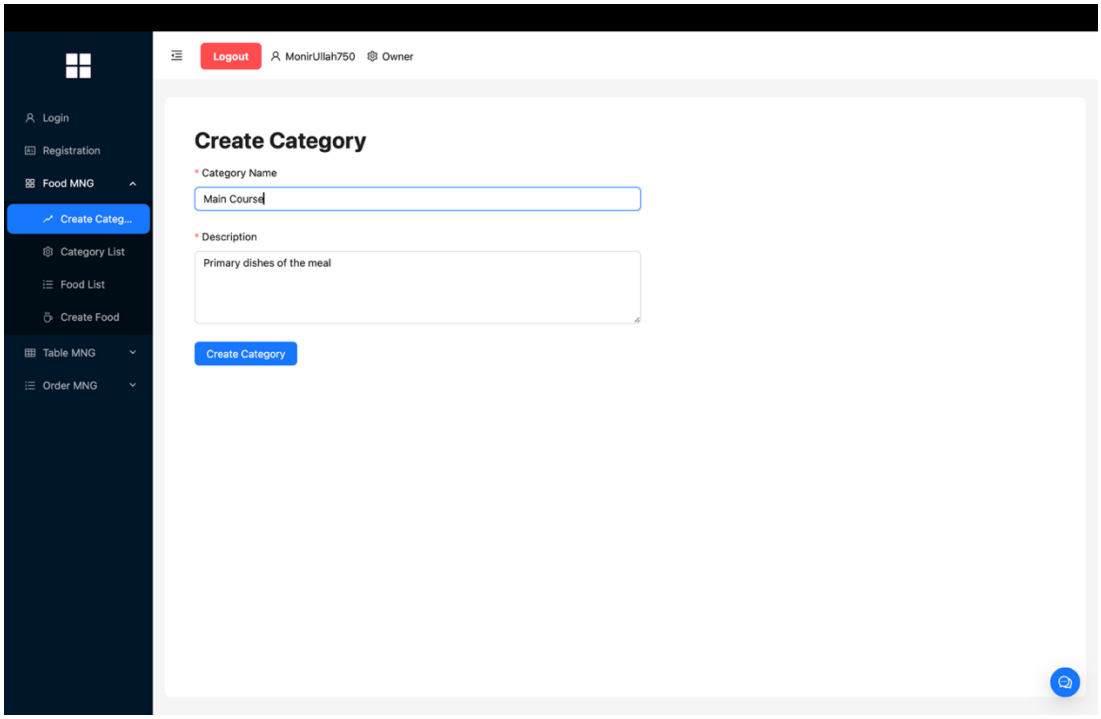


Figure 5—4 : Create Category

- **Categories List**

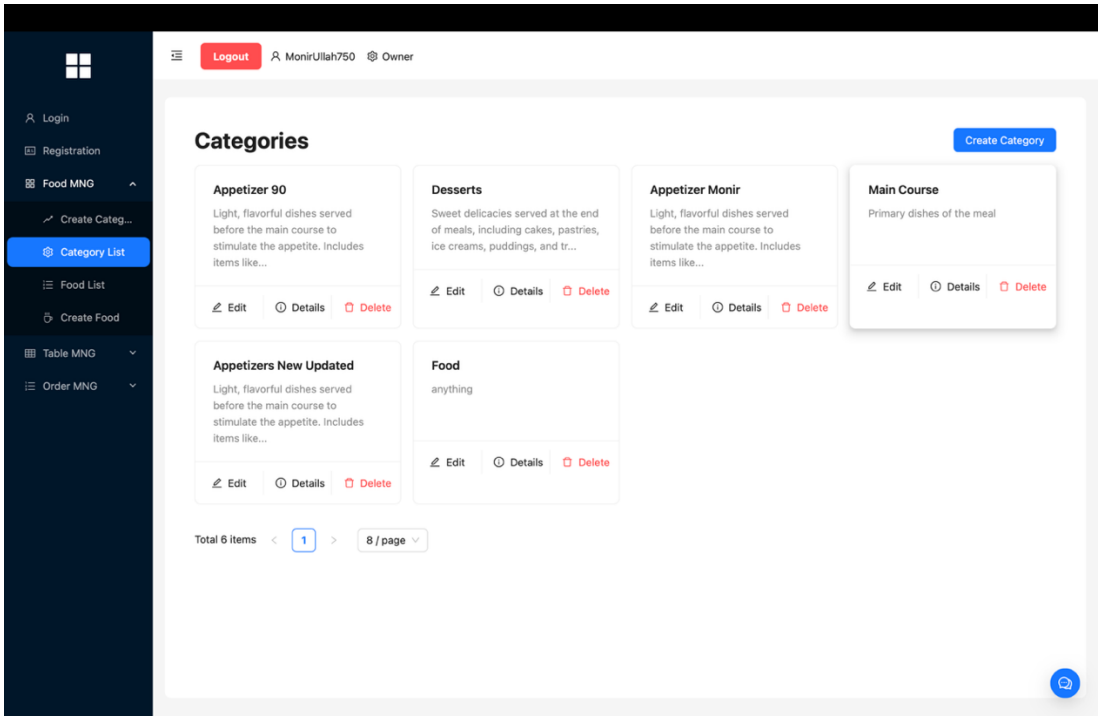


Figure 5—5 : Categories List

- **Create Food**

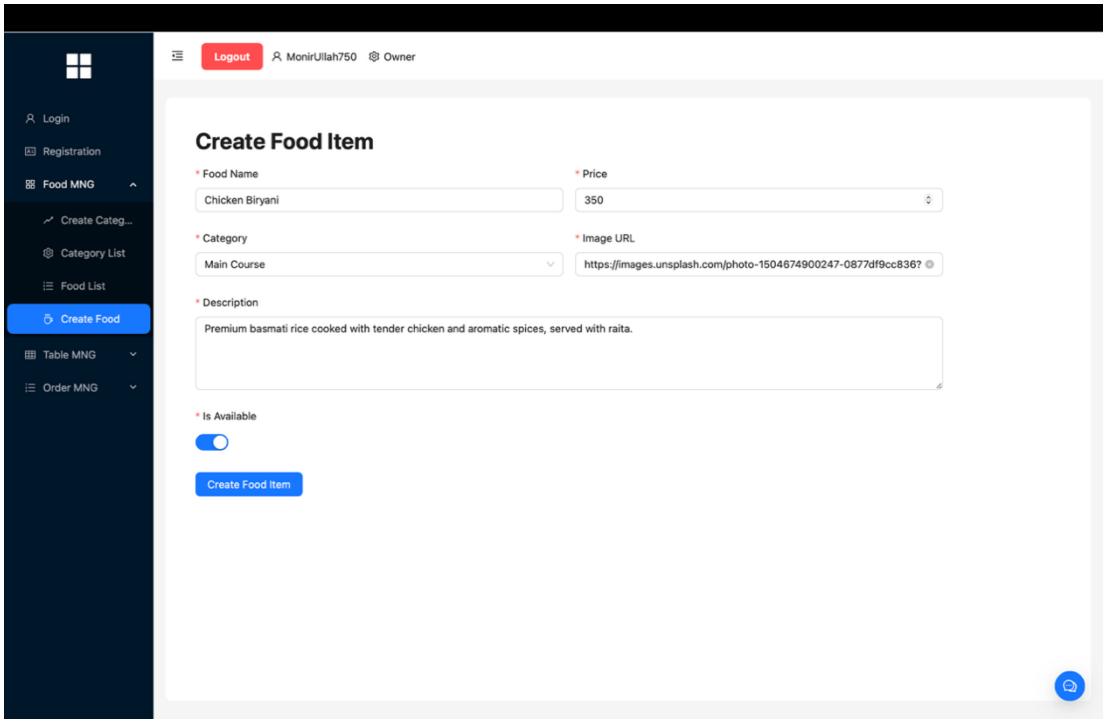


Figure 5—6 : Create Food

- **Food List**

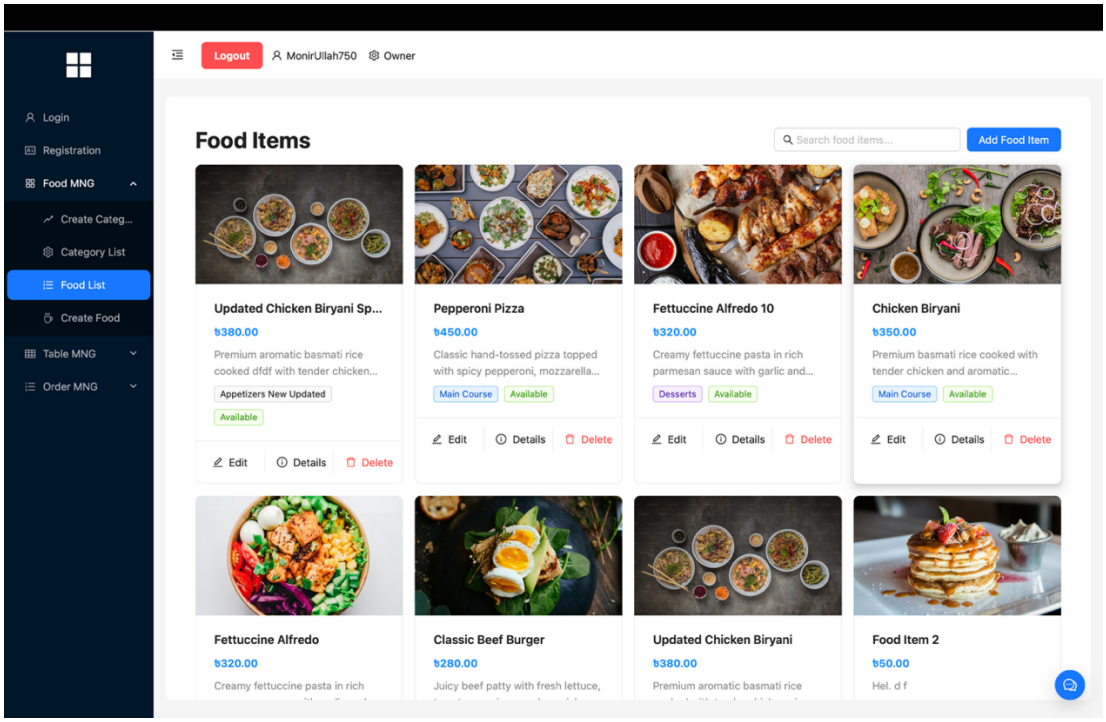


Figure 5—7 : Food List

- **Create Table**

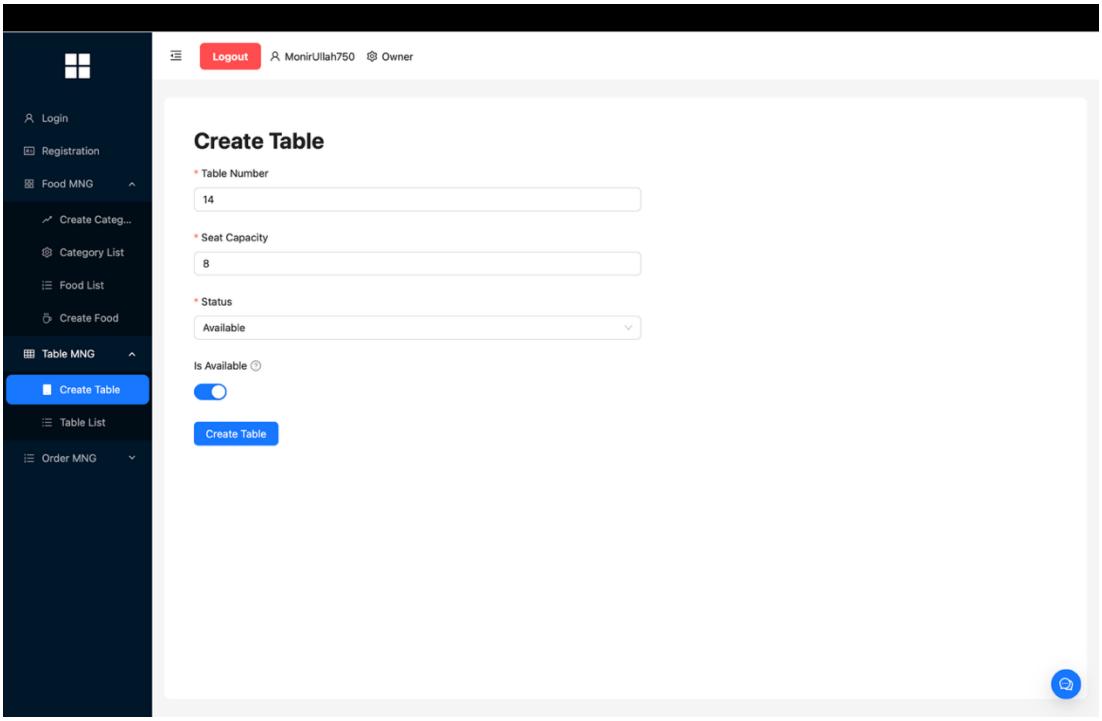


Figure 5—8 : Create Table

- **Table List**

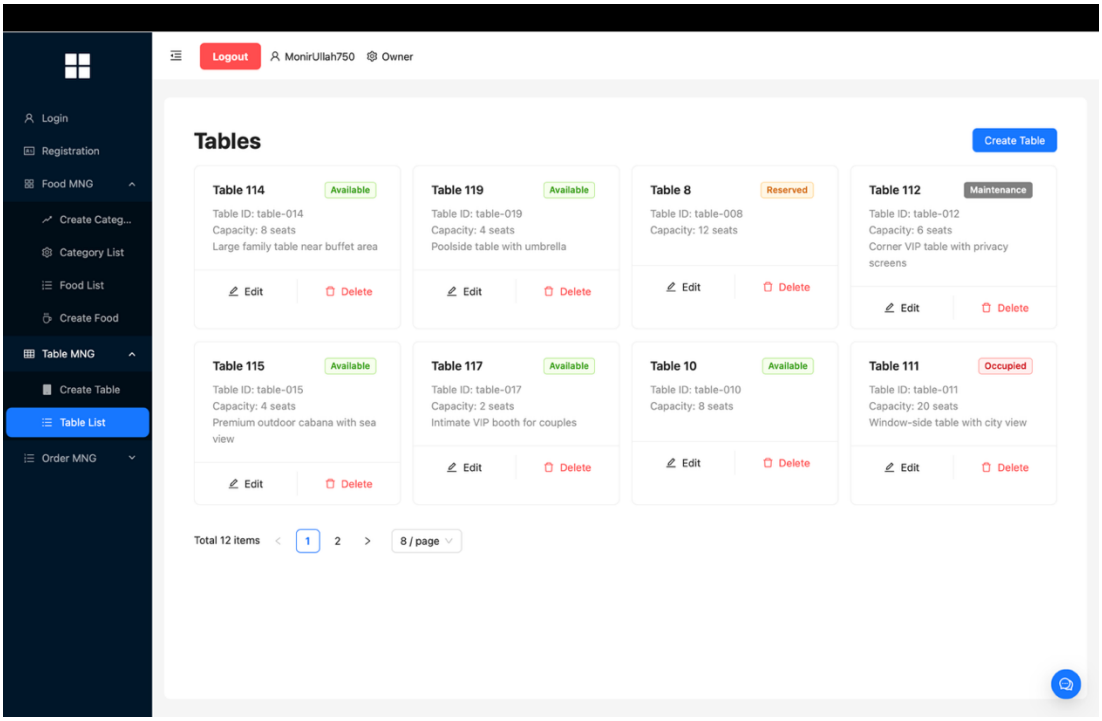


Figure 5—9 : Table List

- **Create Order**

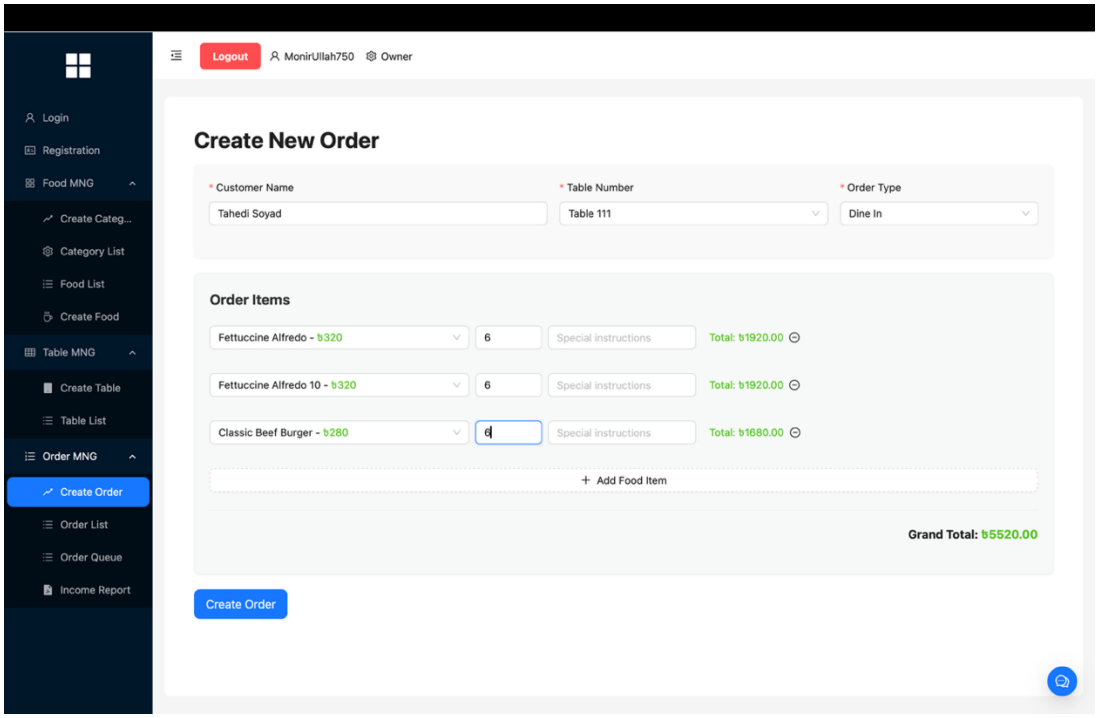


Figure 5—10 : Create Order

- **Order List**

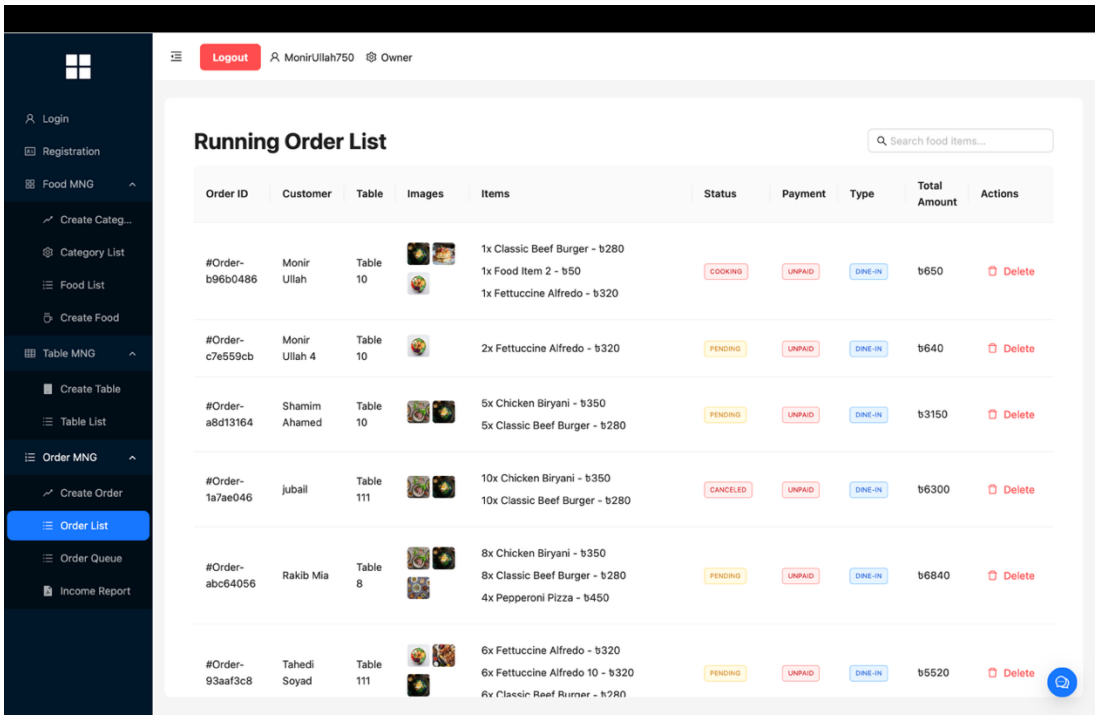


Figure 5—11 : Order List

- **Order Queue**

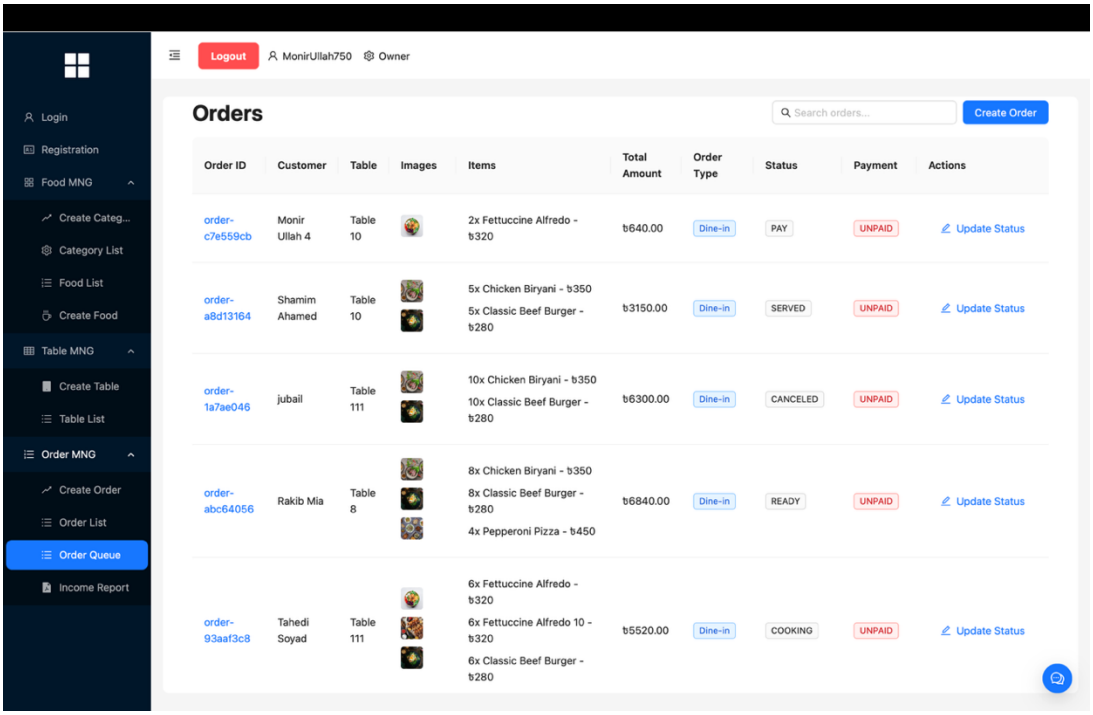


Figure 5—12 : Order Queue

- **Income Report**

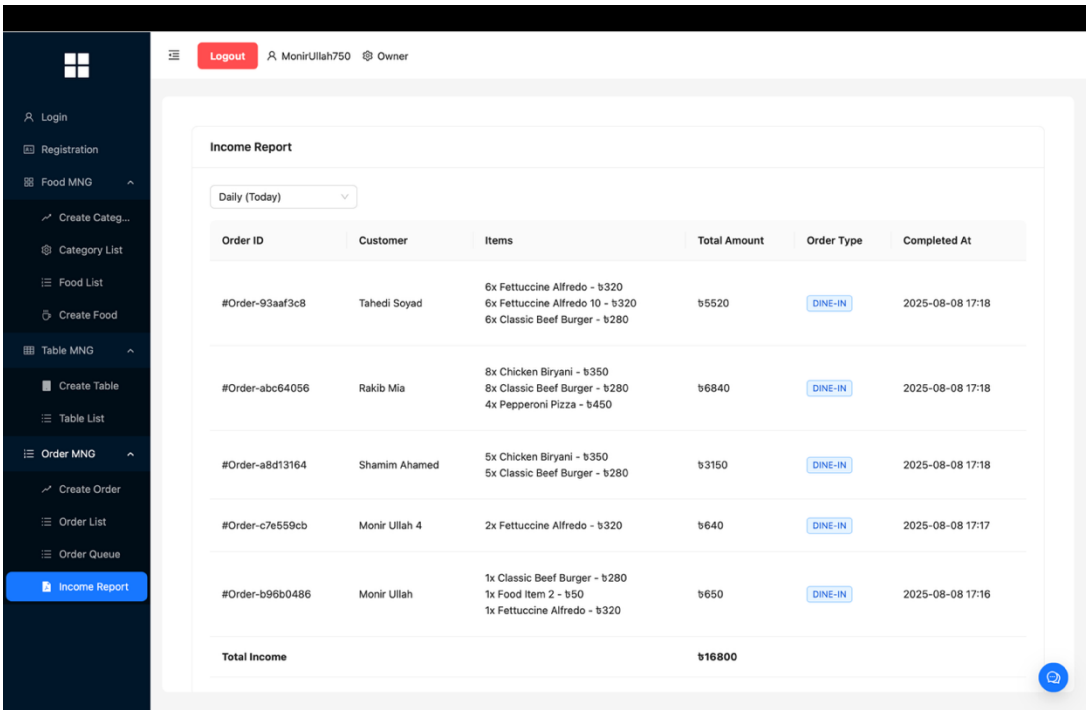


Figure 5—13 : Income Report

- **Chat with your AI Assistant**

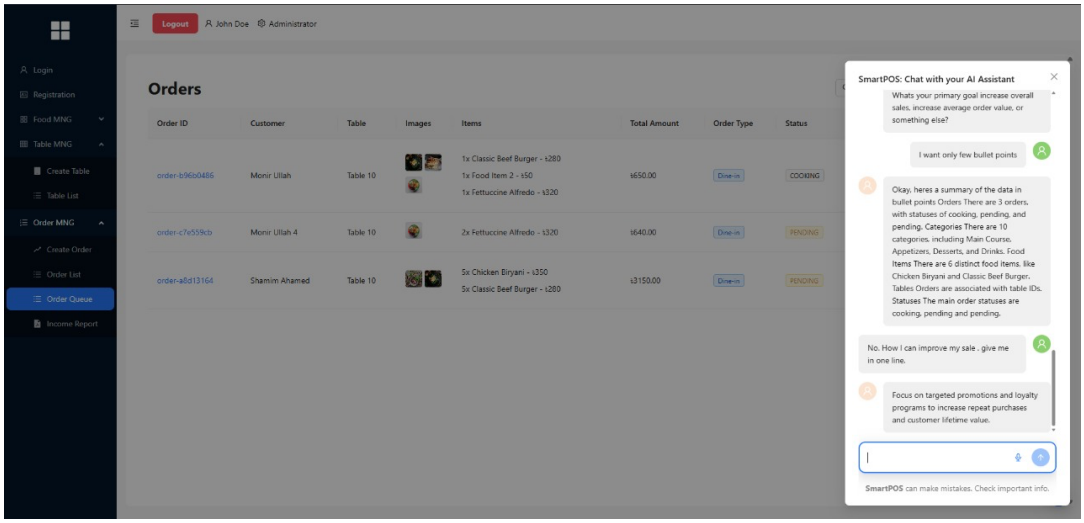


Figure 5—14 : Chat with your AI Assistant

In summary, Chapter 5 equips users with clear, step-by-step instructions for effective utilization of SmartPOS. The manual ensures seamless adoption of system features, enabling restaurants to fully leverage its capabilities with minimal technical barriers.

Chapter 6 : Project Summary

6.1. Introduction

In today's fast-paced restaurant industry, traditional paper-based order management and manual billing processes often lead to errors, delays, and inefficiencies that undermine customer satisfaction and profitability . SmartPOS is a comprehensive web-based Restaurant Management System designed to automate and streamline these core operations . By integrating digital menus, real-time order tracking, automated kitchen notifications, and centralized billing, SmartPOS minimizes human error and accelerates service .

The platform provides role-specific interfaces for administrators, waitstaff, kitchen staff, and cashiers, enabling seamless communication and coordination. SmartPOS is built with scalability in mind: it supports environments from small cafés to large quick-service restaurants and offers multiple payment options along with inventory tracking and sales analytics for data-driven decision making . Overall, SmartPOS enhances operational efficiency and customer experience, helping restaurants reduce costs and improve service quality in a competitive market.

6.2. Project Limitation

Despite its benefits, SmartPOS's current implementation has several technical and operational limitations:

- **Reliance on Internet Connectivity:** As a cloud-based system, SmartPOS requires a stable network connection. Any outage or poor connectivity can interrupt order processing and communication (established POS systems also face issues without offline support).

- **Security and Data Privacy:** The platform handles sensitive customer and payment data, so robust security measures (encryption, authentication, secure payment processing) are essential. The project itself notes that data flow optimization and system security demand careful design and testing .

- **Scalability and Performance:** While designed to scale, the implementation may not have been stress-tested under heavy load. High user volumes or large order backlogs could reveal performance bottlenecks, requiring further optimization.

- **Feature Limitations:** The system covers core functions but lacks some advanced capabilities. For example, it does not currently support offline operation (orders cannot be taken without internet), nor are features like loyalty programs, franchise/multi-location management, or third-party delivery integration implemented.

- **Inventory Management Accuracy:** Although SmartPOS tracks stock and sends low-stock alerts, it relies on accurate data entry. Without integration to real-time inventory hardware, human error in inventory updates could still lead to discrepancies.

- **User Adoption and Training:** Shifting from paper-based workflows to a digital system requires staff training. Even with a user-friendly interface, waiters, cooks, and cashiers need time and guidance to learn SmartPOS. (The proposal explicitly notes user adoption as a key challenge .)

6.3. Project Scope

The implemented features of SmartPOS include the following functionalities:

- **User authentication and roles:** Secure login/registration with role-based access for Admin, Waiter, Kitchen Staff, Cashier, and Customer users.

- **Menu management (Admin):** Administrators can add, edit, or remove menu items and set prices.

- **Order placement and tracking:** Waiters (and customers) can browse the digital menu and place orders by table. Each order is tracked in real time, showing status updates (Pending, Preparing, Ready, Served) to all relevant users.

- **Kitchen processing:** The kitchen interface immediately receives new orders and allows staff to update each order's status (e.g. marking items as "Preparing" or "Ready").

- **Payment processing:** Once meals are ready, customers can make payments (cash, card, or digital wallet), and the cashier finalizes the transaction and records it in the database .

- **Inventory management:** SmartPOS automatically deducts ingredients from stock as orders are served and alerts the admin when items run low.

- **Reporting:** The system generates basic sales reports and summary analytics for restaurant owners/managers to monitor performance and trends.

- **User interface:** A responsive, user-friendly web front-end built with React.js and Tailwind CSS provides an intuitive experience for all users.

6.4. Future Work

Potential enhancements and new modules that could improve or expand SmartPOS include:

- **Online Ordering and Delivery Integration:** Develop a customer-facing online portal or mobile app so patrons can place orders remotely. Integrating with delivery services (e.g. through API connections) would extend the restaurant's reach.

- **Multi-Location Support:** Adapt the system for franchise or chain restaurants by enabling multiple branch management and centralized data synchronization across locations.

- **Mobile App Development:** Create native or cross-platform apps for waitstaff, kitchen staff, and customers. Mobile apps could offer push notifications and a more convenient interface.

- **Loyalty and CRM Modules:** Add features such as customer profiles, rewards/points programs, and automated marketing (e.g. targeted promotions or discounts for repeat customers).

- **Advanced Analytics:** Implement enhanced reporting dashboards with real-time KPIs, sales trends, and predictive analytics (such as demand forecasting or peak-hour analysis) to support strategic decision-making.

- **Offline/Resilience Features:** Introduce an offline mode with local caching so the system can continue taking and processing orders during network outages and sync when connectivity is restored.

- **Third-Party Integrations:** Integrate with additional payment gateways (e.g. Stripe, mobile wallet APIs), accounting or ERP software, supplier/inventory systems, and kitchen hardware (e.g. receipt printers, digital kitchen displays) for greater automation.

- **Localization and Accessibility:** Support multiple languages, currencies, and accessibility features (e.g. voice commands, screen-reader compatibility) to serve a broader range of users.

- **Enhanced Security:** Incorporate stronger security measures such as multi-factor authentication, end-to-end encryption of data, and regular security audits to protect against evolving threats.

- **Emerging Technologies:** Explore innovations like IoT-enabled inventory sensors, AI-driven menu recommendations or upselling prompts, and voice assistants to further modernize the dining experience.

6.5. Conclusion

SmartPOS successfully delivers a unified web platform that meets its original objectives: automating order processing, enabling real-time order tracking, and consolidating billing and inventory management. By replacing manual processes with digital workflows, the project demonstrated improved efficiency and reduced errors in core restaurant tasks. The multi-role design allows administrators, waiters, kitchen staff, and cashiers to coordinate more effectively.

Key lessons from this project include the importance of iterative development and thorough testing to ensure system reliability, as well as the value of a user-centered interface to accelerate staff adoption. We also learned that integrating real-time features (such as live order updates) and secure payment processing requires careful architectural planning. Overall, the project resulted in a functional proof-of-concept that streamlines restaurant operations, and the insights gained will inform future enhancements.

Sources: Information was drawn from the project proposal document for SmartPOS: A Web-Based Restaurant Management Web Application by **Monir Ullah**

REFERENCES

Hisam, K. H. (2019). Restaurant Management System. Universiti Malaysia Pahang. Retrieved
from https://indah.ump.edu.my/ethesis/public/thesis/CS/CC17113.pdf

Nasir, N. Y . M. (2021). An Online Restaurant Management System (District50). Universiti
Teknologi PETRONAS. Retrieved from
https://utpedia.utp.edu.my/24028/1/An%20Online%20Restaurant%20Management%20System%20%28District50%29.pdf

Tripathi, P . P ., Shukla, S., & Mishra, S. (2024). RMS ERP - Revolutionizing Restaurant
Management System (A Smart Solution for Ideal Restaurant Operations). Journal of Emerging Technologies and Innovative Research, 11(2). Retrieved from
https://www.jetir.org/papers/JETIR2402584.pdf

Lanke, P ., Harne, P ., Adhao, A., & Sapkal, A. (2023). Restaurant Management System. Shri
Sant Gajanan Maharaj College of Engineering. Retrieved from
https://www.ssgmce.ac.in/uploads/UG_Projects/it/Gr%20No-10-Project-Report.pdf

SHORT BIOGRAPHY

Myself **Monir Ullah**, a student of Batch 35 with ID **212-35-750** at Daffodil International University. I have a keen interest in software development, web application design, and system automation. Throughout my academic journey, I have worked on various projects, with SmartPOS: A Web-Based Restaurant Management System being one of my key contributions. I am passionate about developing efficient, scalable, and user-friendly applications that enhance business operations. I can be reached via email at monir35-750@diu.edu.bd or by phone at +880 1876-630625.

Plagiarism Report

212-35-750

ORIGINALITY REPORT

12%	10%	2%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

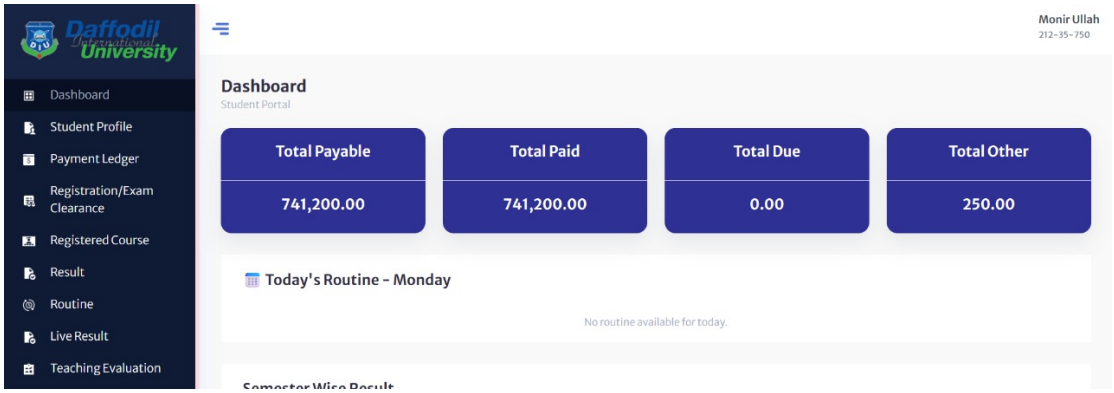
1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	7%
2	testcase.ch Internet Source	<1%
3	Submitted to Daffodil International University Student Paper	<1%
4	Phenikaa University Publication	<1%
5	Submitted to University College Birmingham Student Paper	<1%
6	Submitted to Dundalk Institute of Technology Student Paper	<1%
7	Submitted to South Bank University Student Paper	<1%
8	wiki.smu.edu.sg Internet Source	<1%
9	Submitted to Asia e University Student Paper	<1%
10	www.coursehero.com Internet Source	<1%
11	www.uniselinus.education Internet Source	<1%
12	Luo, Yong. "Nanostructures design and fabrication for magnetic storage applications",	<1%

	Proquest, 20111108 Publication	
13	Submitted to Collin County Community College Student Paper	<1 %
14	Submitted to University of Greenwich Student Paper	<1 %
15	Submitted to University of Strathclyde Student Paper	<1 %
16	Submitted to Coventry University Student Paper	<1 %
17	Submitted to Southern New Hampshire University - Continuing Education Student Paper	<1 %
18	Submitted to Wawasan Open University Student Paper	<1 %
19	Davis, Peter Barden. "The petrotectonic evolution of the Sivrihisar Massif, Turkey", Proquest, 20111004 Publication	<1 %
20	Submitted to National Economics University Student Paper	<1 %
21	Submitted to New York College in Athens, Greece Student Paper	<1 %
22	github.com Internet Source	<1 %
23	www.diva-portal.org Internet Source	<1 %
24	www.padtx.com Internet Source	<1 %

25	Submitted to Universiti Sains Islam Malaysia Student Paper	<1%
26	Submitted to University of Bradford Student Paper	<1%
27	www.buzzybrains.com Internet Source	<1%
28	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
29	www.javatpoint.com Internet Source	<1%
30	getdevdone.com Internet Source	<1%
31	rasya.id Internet Source	<1%
32	www.posmotto.com Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Account Clearance



Library Clearance