Development of a Modular Web-Based Cafeteria Management System Using MERN Stack: A Case Study on PUST

Project

Course Code: ICE - 4210

B.Sc. (Engineering) Examination - 2023

A thesis paper submitted to the Department of Information and Communication Engineering, Pabna University of Science and Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering in Information and Communication Engineering

Submitted by:

Md. Arif Hossain

Roll No: 190612

Reg. No: 1065337

Session: 2018-2019

Supervised by:Akif Mahdi

Lecturer

Department of Information and Communication Engineering

Pabna University of Science and Technology

Pabna-6600, Bangladesh

Declaration

I hereby declare that the project work entitled **"Cafeteria Management System"** has been carried out by me under the course **ICE-4210 (Thesis/Project)** during the academic session 2018–2019.

This work is the result of my own independent research and has not been submitted previously, either in part or in full, for the award of any other degree or diploma in this or any other institution. Any assistance received from others during the course of this project has been duly acknowledged.

.....

Md. Arif Hossain
Roll No: 190612
Reg. No: 1065337
Department of Information and Communication Engineering
Pabna University of Science and Technology, Pabna-6600, Bangladesh

Certificate of Approval

This is to certify that the thesis entitled **"Cafeteria Management System"** submitted by **Md. Arif Hossain**, bearing Roll No. 190612 and Reg. No. 1065337, is the result of original research work carried out under my supervision as part of the course **ICE-4210 (Thesis/Project)**.

This project fulfills the partial requirement for the award of the degree of Bachelor of Science in Engineering in the Department of Information and Communication Engineering, Pabna University of Science and Technology. To the best of my knowledge, this thesis has not been submitted to any other institution for any degree or diploma.

.....

Akif Mahdi
Lecturer
Department of Information and Communication Engineering
Pabna University of Science and Technology, Pabna-6600, Bangladesh

Dedicated to...

My parents

and

To my beloved all brothers

Acknowledgements

I would like to express my sincere gratitude to my project supervisor, **Akif Mahdi**, for his invaluable guidance, continuous support, and insightful feedback throughout the development and documentation of this project. His expertise and encouragement have been instrumental in shaping the direction and success of my work.

I am also deeply thankful to my family for their unwavering support, patience, and motivation during my final year at university. Their constant encouragement and assistance, especially during the preparation of this report, have been immensely helpful.

— The Author

Abstract

This paper presents a full-stack Cafeteria Management System designed to streamline restaurant operations through digital automation. The system features modular panels for customers, administrators, POS staff, kitchen personnel, and billing units, each developed to serve specific roles within the restaurant workflow. The backend utilizes Node.js with Express and MongoDB, implementing RESTful APIs and middleware for secure data handling and user authentication.The frontend panels, built with React, offer responsive interfaces tailored for user interaction, administrative control, real-time kitchen coordination, and billing operations. Key functionalities include food and inventory management, order processing, table reservations, customer feedback, loyalty points, and report generation. The system improves operational efficiency, reduces manual errors, and enhances the overall dining experience by integrating all restaurant services into a single unified platform.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, the digital transformation of food service systems has become a necessity, particularly in institutional environments such as universities, colleges, and office cafeterias. Cafeteria management has traditionally involved manual processes for order placement, table reservation, and food delivery. These manual systems often result in inefficiencies such as long queues, poor coordination among staff, lack of data-driven decision-making, and an inability to serve customers efficiently during peak hours.

With the rise of automation and the integration of modern web technologies, smart cafeteria systems are now being designed to streamline and digitize the entire food service workflow. These systems are tailored to different roles such as admin, kitchen staff, billing personnel, POS operators, and end users like students, teachers, and organizations. A role-based smart cafeteria system improves operational efficiency, enhances user experience, and supports scalability.

Additionally, modern cafeteria systems are now incorporating features such as scheduled dine-in ordering, table reservation with preferred date and time, and home delivery to cater to the needs of students, teachers, and organizations. These innovations allow better planning, reduce wait times, and improve food service efficiency.

1.2 Problem Statement

During our study at Pabna University of Science and Technology (PUST), we observed several shortcomings in the existing cafeteria service infrastructure. These challenges significantly impact the efficiency and user experience of both customers and cafeteria

staff. The major issues identified are:

- Manual order management often leads to delays and miscommunication between the kitchen and POS staff.
- Users currently have no option to schedule dine-in orders ahead of time, affecting convenience.
- The cafeteria lacks a table reservation system, making it difficult for users to ensure seating availability during peak hours.
- There is no online payment facility in the PUST cafeteria, resulting in cash-only transactions that lead to long queues and decreased convenience.
- The system does not support home delivery, restricting service to only on-premise customers.
- There is no digital feedback mechanism in place, which limits the ability to monitor and improve food quality and customer service.

These limitations reduce customer satisfaction, waste time and resources, and lower the overall efficiency of cafeteria operations.

1.3 Objectives

The main objective of this project is to design and implement a Role-Based Smart Cafeteria Management System that addresses the limitations of traditional methods.

- Develop a centralized platform with distinct roles: Admin, POS, Kitchen, Billing, and User.
- Enable users to place food orders through a user-friendly interface.
- Implement a scheduled dine-in ordering feature allowing users to pre-order meals for a preferred time.
- Allow customers to reserve cafeteria tables by selecting a date and time in advance.
- Integrate a home delivery system with location management and delivery tracking.
- Provide real-time order updates and kitchen status to improve coordination.

- Incorporate secure login, authentication, and role-based access control.
- Collect and manage customer feedback and ratings.

1.4 Scope

This project focuses on developing a full-stack web-based cafeteria management system that supports the following modules:

- **Admin Panel:** Manage menu, inventory, staff, users, table status, and system reports.
- **POS Panel:** Handle food orders, print bills, assign tables, and update order statuses.
- **Kitchen Panel:** View live order queue, update status to *preparing* or *ready*, and notify POS.
- **Billing Panel:** Generate invoices, calculate discounts, manage payments, and track completed orders.
- **User Panel:** Allows students, teachers, and organizations to browse the menu, place orders, schedule dine-in meals, reserve tables, request home delivery, and provide feedback.

The system will be developed using MERN stack (MongoDB, Express.js, React.js, and Node.js) [**?**], ensuring scalability, responsiveness, and real-time communication using Socket.IO [8].

1.5 Significance of the Study

The proposed cafeteria management system introduces a smart, role-based approach that enhances operational workflows and user experience. It ensures that:

- Users can plan meals effectively by pre-ordering for dine-in or reserving tables.
- Real-time synchronization between kitchen and POS improves service speed and accuracy.
- Home delivery expands service beyond the physical premises of the cafeteria.

- Administrators gain insights through feedback and reporting features for better decision-making.
- Time and resource waste is minimized, improving the overall productivity and satisfaction of all stakeholders.

This system sets a precedent for future digital cafeteria models in educational and organizational institutions.

1.6 Languages and Technologies Used

The development of the Cafeteria Management System incorporates several modern technologies and programming languages to ensure a scalable, maintainable, and interactive application experience. Each component of the system leverages technologies appropriate to its function. The detailed description is as follows:

- **React.js:** This JavaScript library is used for building dynamic and responsive user interfaces for the frontend of all panels including User, Admin, POS, Kitchen, and Billing. React's component-based architecture promotes reusability and improves maintainability, which is ideal for large-scale applications with multiple user roles **[1]**.
- **Tailwind CSS:** A utility-first CSS framework used to style the frontend components quickly and consistently. Tailwind enables the creation of modern, mobile-responsive UI designs with minimal custom CSS, which speeds up the development process and ensures a visually cohesive interface across the system **[6]**.
- **HTML5 and JavaScript (ES6+):** HTML5 provides the basic structure of the web pages, while modern JavaScript (ES6 and later) is used for logic, interactivity, and integration between the frontend and backend systems. JavaScript features like arrow functions, async/await, and destructuring significantly simplify the codebase **[1]**.
- **Node.js:** A runtime environment used to build the backend of the system. It handles asynchronous operations, manages API requests, and processes business logic. Node.js allows the use of JavaScript on the server side, enabling full-stack development using a single language **[7]**.

- **Express.js:** A lightweight and flexible Node.js framework used to create RESTful APIs for all system operations such as food item management, order processing, feedback handling, authentication, and report generation. Middleware support in Express allows structured route handling and error management **[2]**.
- **MongoDB:** A NoSQL database used to store and manage application data including user profiles, menu items, orders, inventory, and feedback. Its document-oriented nature and flexibility with unstructured data make it suitable for handling complex and evolving data schemas **[3]**.
- **Mongoose:** An Object Data Modeling (ODM) library for MongoDB and Node.js. It provides schema validation, query building, and middleware features, simplifying interactions with the database and ensuring data integrity **[3]**.
- **JSON Web Token (JWT):** A secure token-based authentication mechanism used to verify user identity across different panels. JWT enables role-based access control (e.g., Admin, Staff, Kitchen, Customer), ensuring that only authorized users can perform specific actions **[4][12]**.
- **Stripe API:** An integrated payment gateway used for processing customer payments in the billing panel. It supports secure transactions and provides tools for managing invoices, receipts, and payment confirmation **[1]**.
- **Socket.IO:** A real-time bidirectional communication library used to facilitate live updates between the POS and Kitchen panels. For example, when an order is placed from the POS, it instantly appears in the Kitchen panel, and status updates are reflected in real-time **[8]**.
- **Axios:** A promise-based HTTP client used in the frontend to communicate with backend APIs. It simplifies sending requests such as placing orders, submitting feedback, or fetching inventory data **[1]**.
- **Dotenv:** Used for managing environment-specific configuration (such as database URIs, JWT secrets, and Stripe API keys) in a secure and organized way, preventing sensitive credentials from being exposed in the codebase **[7]**.
- **Helmet and CORS Middleware:** Helmet helps secure the app by setting various HTTP headers, while CORS enables cross-origin communication between the frontend (e.g., running on Vite or React dev server) and the backend server **[2][6]**.

Each technology plays a crucial role in ensuring the reliability, performance, and security of the overall system. Together, they form a modern full-stack web application that delivers a seamless experience for customers, administrators, and staff.

1.7 Chapter Outline

- **Chapter 2:** Literature Review — overview of similar existing systems and technologies.
- **Chapter 3:** System Design and Methodology — architectural and design decisions, tools used.
- **Chapter 4:** Implementation — code structure, module explanation, and key features.
- **Chapter 5:** Testing and Results — testing approach, sample outputs, and analysis.
- **Chapter 6:** Conclusion and Future Work — summary of contributions and future improvements.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we explore the evolution of cafeteria systems, existing solutions, and the technological landscape that influenced the development of the proposed Cafeteria Management System. We analyze previous systems, identify their limitations, and introduce the modern web technologies that empower our solution.

2.2 Existing Systems and Their Limitations

Traditional cafeteria systems were primarily manual or semi-digital, often relying on paper-based orders or basic POS terminals with limited backend functionality. Some institutions adopted single-interface applications, where all stakeholders (admin, staff, kitchen) interacted with the same interface, leading to confusion, lack of access control, and performance issues.

- **Manual systems:** Prone to human error, delay in service, and lack of traceability.
- **Single panel digital systems:** No role-based segregation, inefficient communication.
- **Lack of real-time updates:** Orders were updated slowly or required manual intervention.

2.3 Technology Overview

Modern cafeteria solutions have evolved to leverage real-time technologies and role-based architectures. The system developed in this project adopts the MERN stack, which offers modularity, scalability, and speed.

2.3.1 MERN Stack Components

- **MongoDB:** NoSQL database for storing flexible order, user, and inventory data [3].
- **Express.js:** Backend framework running on Node.js to handle routing and API communication [2].
- **React:** Component-based frontend library for building fast, dynamic interfaces [1].
- **Node.js:** JavaScript runtime enabling backend operations [7].

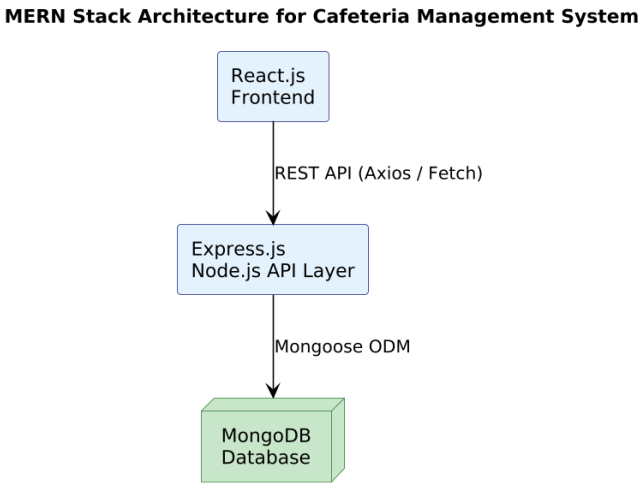

Figure 2.1: Technology Stack: MERN

2.4 System Gaps in Existing Cafeteria Solutions

Through literature and competitive product analysis, the following key issues were identified:

- **Lack of Real-Time Communication:** Delays between order placement and kitchen acknowledgment [8].
- **Role Ambiguity:** Staff from POS and kitchen often share the same interface, leading to operational confusion.
- **No Customer Feedback Integration:** Most systems ignored rating or comment mechanisms.
- **Rigid Table Reservation Systems:** Either no reservation module or lacked multi-role approval.

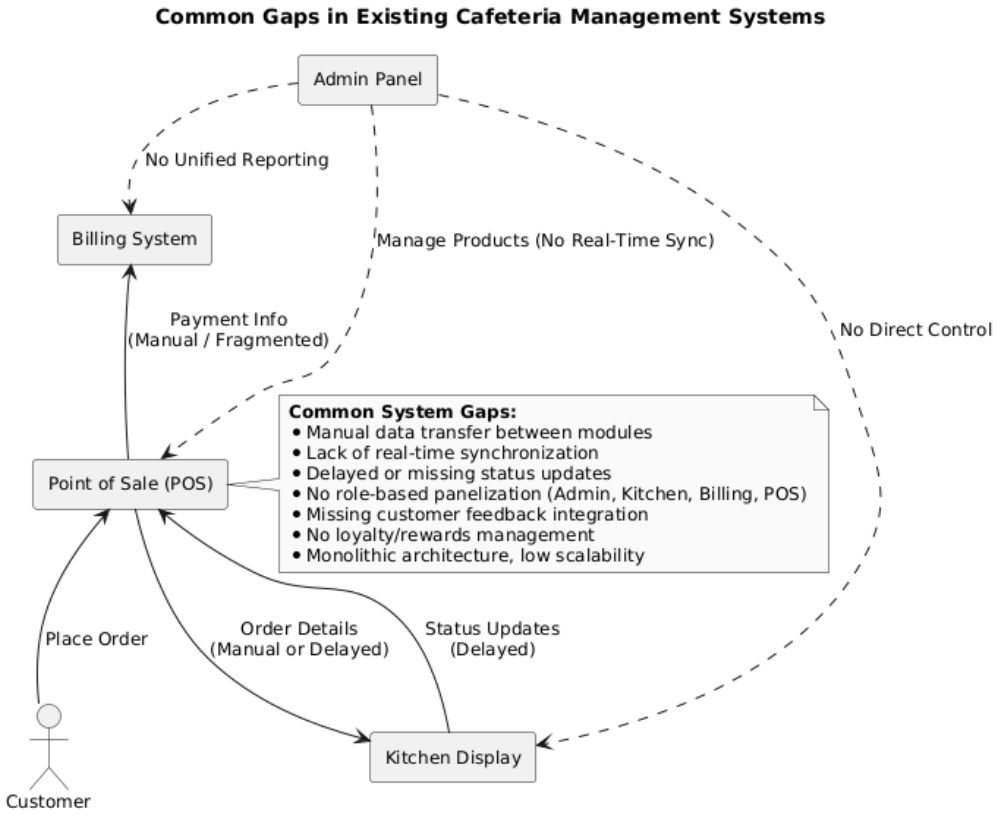

Figure 2.2: Common Gaps in Existing Systems

2.5 Proposed Solution Overview

The Cafeteria Management System addresses these limitations using:

Real-Time Communication: Orders and status updates are synced using WebSocket-based communication [8].

Role-Based Panels: Separate interfaces for Admin, POS, Kitchen, Billing, and Customers ensure workflow clarity.

Feedback System: Each delivered order allows the customer to provide ratings and comments.

Advanced Reservation Module: Users can request table bookings, with admins having final approval rights.

2.6 Summary

This chapter detailed the shortcomings in traditional systems and presented the technological backbone of the proposed solution. By leveraging MERN technologies and addressing communication, role, and feedback issues, the Cafeteria Management System positions itself as a modern, scalable, and user-centric platform.

Chapter 3

System Design and Methodology

3.1 Introduction

This chapter outlines the architectural blueprint and development strategy adopted for the Cafeteria Management System. The system is designed as a scalable, modular, and fault-tolerant web application that separates concerns across multiple role-based panels. Emphasis has been placed on real-time communication, user role segregation, and seamless synchronization of operations to improve efficiency and user satisfaction.

3.2 System Overview

The proposed Cafeteria Management System is a full-stack web application composed of five distinct panels, each tailored to a specific operational role within the cafeteria environment. These panels work independently but are connected through a centralized backend system:

1. **User Panel** — Enables customers (primarily students) to browse the menu, reserve tables, pre-order food for dine-in at a scheduled time, place home delivery orders, and provide feedback.

2. **Admin Panel** — Offers administrative control for managing users, menu items, table reservations, inventory, and system reports.

3. **POS Panel** — Used by cafeteria staff (e.g., waiters) for placing dine-in orders, managing carts, tracking table status, and monitoring reservation approvals.

4. **Kitchen Panel** — Facilitates kitchen operations by displaying incoming orders and allowing chefs to update order statuses (e.g., preparing, ready).

5. **Billing Panel** — Responsible for generating invoices, calculating bills, handling payments, and freeing up tables after order completion.

Each panel operates with clearly defined responsibilities, reducing complexity and improving maintainability. Communication among panels is achieved via secure RESTful APIs and real-time Socket.IO connections, ensuring timely updates and synchronized states across the system.

3.3 Development Methodology

To address dynamic requirements and enable iterative improvements, the Agile software development methodology was adopted. Development was carried out in short sprints, with regular feedback loops from stakeholders ensuring alignment with user expectations and cafeteria workflows.

Technology Stack

The following technologies and tools were employed for the development:

- **Frontend:** React.js, using Vite for optimized bundling and fast hot module replacement.
- **Backend:** Node.js with Express.js for building RESTful API endpoints.
- **Database:** MongoDB, chosen for its NoSQL schema flexibility and scalability.
- **Real-time Communication:** Socket.IO for event-based, bi-directional communication between POS and Kitchen panels.
- **Authentication and Authorization:** JWT (JSON Web Token) for secure token-based authentication and role-specific access control.

This development methodology allowed the system to evolve incrementally, accommodating new features such as table reservation approval, scheduled dine-in orders, and delivery integration without necessitating major structural changes.

3.4 System Architecture

The high-level architecture of the Cafeteria Management System is depicted in Fig. 3.1. Each panel interacts with the backend via defined API routes and listens for real-time updates as necessary. Data persistence is managed through MongoDB, while middleware components handle authentication, logging, and error handling.

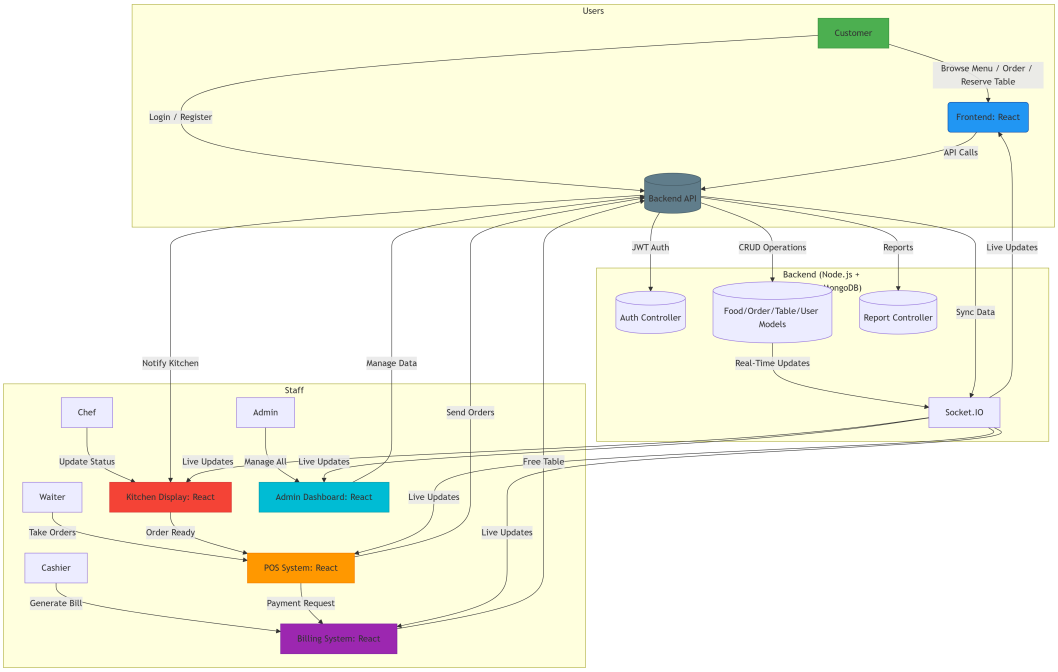

Figure 3.1: Overall System Architecture of the Cafeteria Management System

3.5 Module-wise Functional Design

3.5.1 User Panel

The User Panel of the Cafeteria Management System is designed to provide a seamless, intuitive, and user-friendly interface tailored for end customers. This module allows users to interact with the system to perform core operations such as browsing the food menu, making table reservations, placing and tracking orders, submitting feedback, and managing loyalty points. All features are designed to ensure efficient self-service while enhancing customer satisfaction.

Key Functionalities

- **Menu Browsing:** Customers can explore food items categorized by type, view detailed descriptions, pricing, and availability.
- **Table Reservation:** Users can request to reserve available tables, which are subject to admin approval based on availability.
- **Order Placement:** Enables customers to place food orders directly from the menu and monitor the live status of their orders.
- **Feedback Submission:** Users can submit ratings and textual feedback regarding their dining experience and ordered items.
- **Loyalty Points:** Displays accumulated loyalty points based on completed transactions, encouraging repeat engagement.

Feature Screenshots

- **Secure Login:** This interface enables authenticated access to the user panel. It ensures that only valid users can browse the menu, place orders, or view personal data.

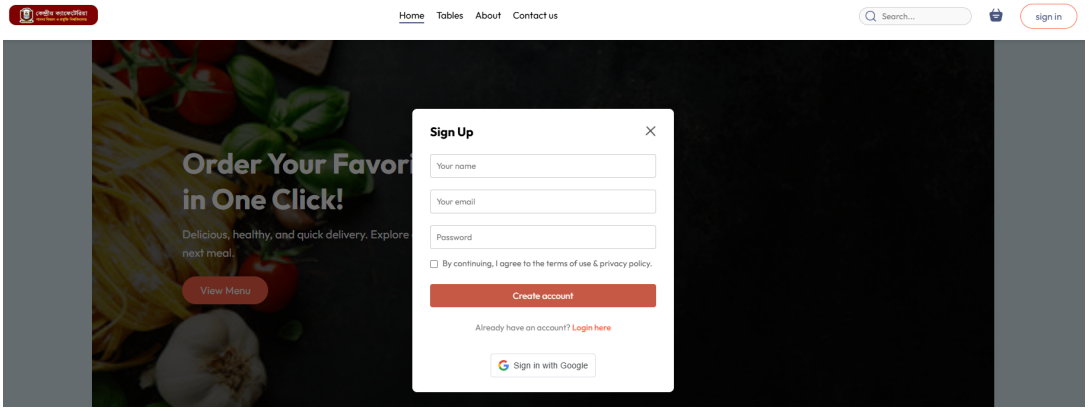

Figure 3.2: Customer Login Screen

- **View Menu and Food:** Users can browse available food items organized by categories. Each item displays relevant information such as name, price, and description.

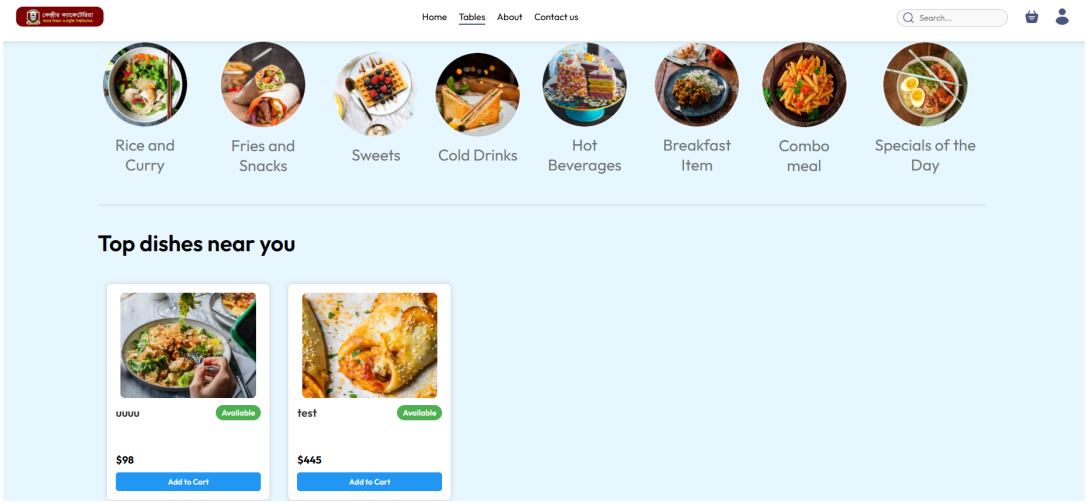

Figure 3.3: Menu Viewing Interface

- **Table Reservation Request:** This feature allows users to request table reservations. Users select preferred time, table type, and provide reservation information.

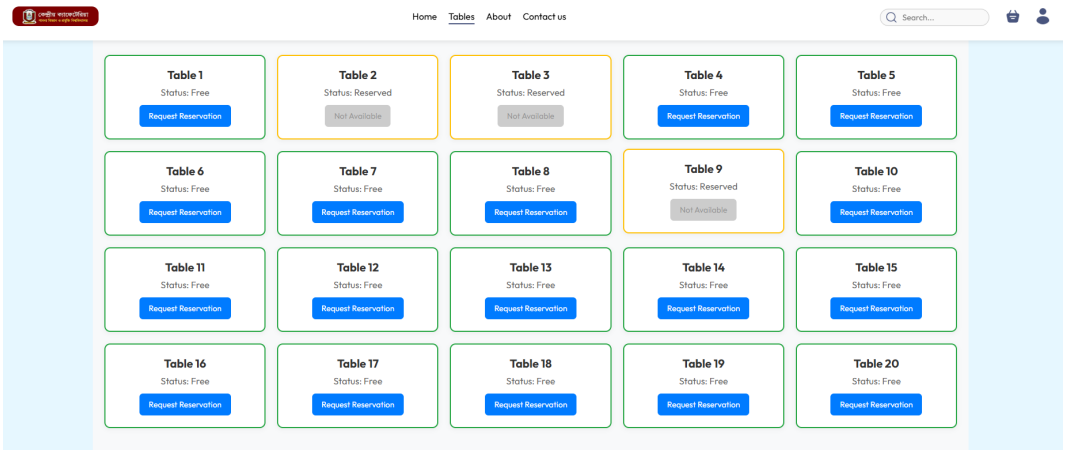

Figure 3.4: Table Reservation System

- **Feedback Submission:** Customers can rate their dining experience and leave feedback for the cafeteria management, helping improve service quality.

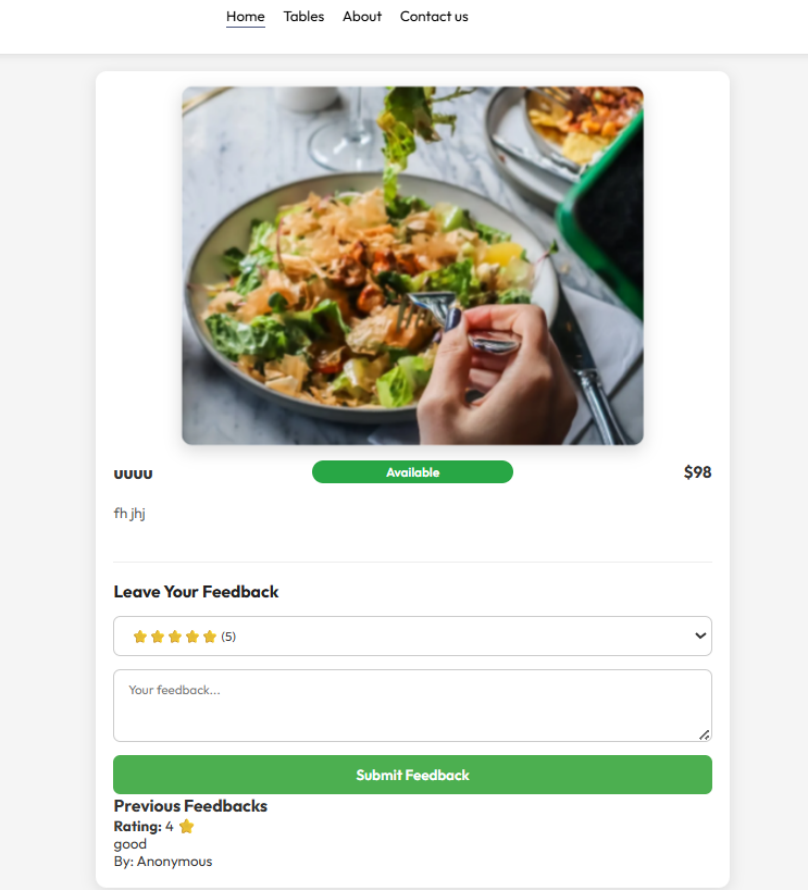

Figure 3.5: Feedback Interface

- **Order Tracking:** Users can track the current status of their placed orders in real-time. This includes updates such as ”Preparing”, ”Ready”, or ”Delivered”.

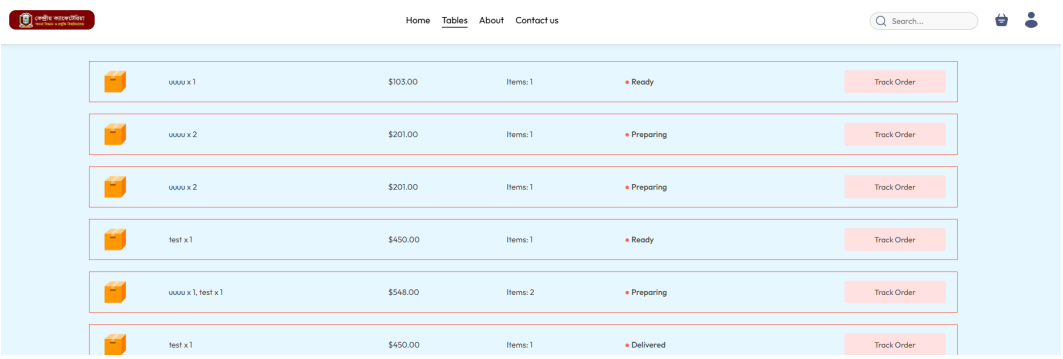

Figure 3.6: Order Management Screen

Use Case Diagram

The Use Case diagram shows how the customer interacts with different features such as login, view menu, place order, track order, submit feedback, and reserve tables.

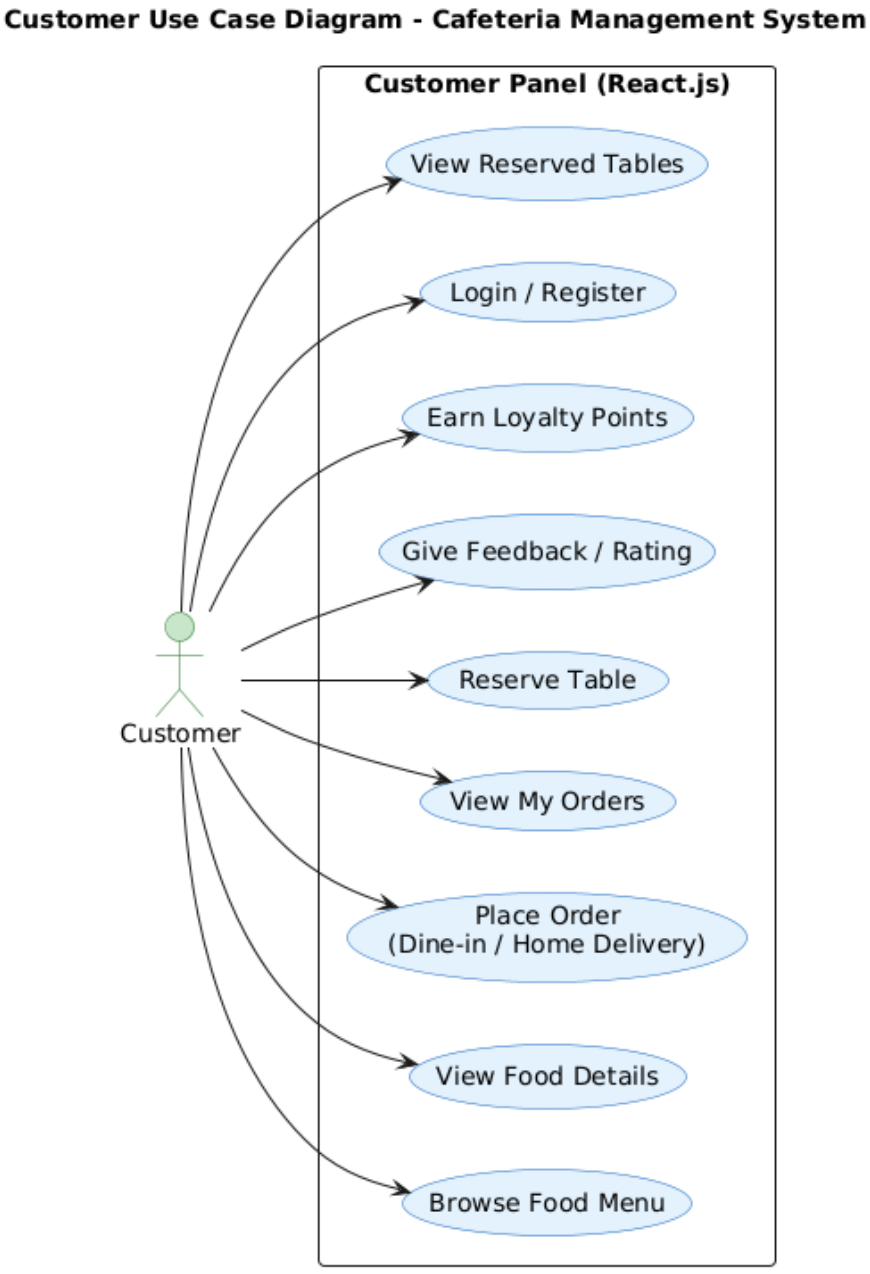

Figure 3.7: Customer Panel Use Case Diagram

Data Flow Diagram

This diagram visualizes how data flows within the system when a customer interacts with functionalities like placing an order or submitting feedback.

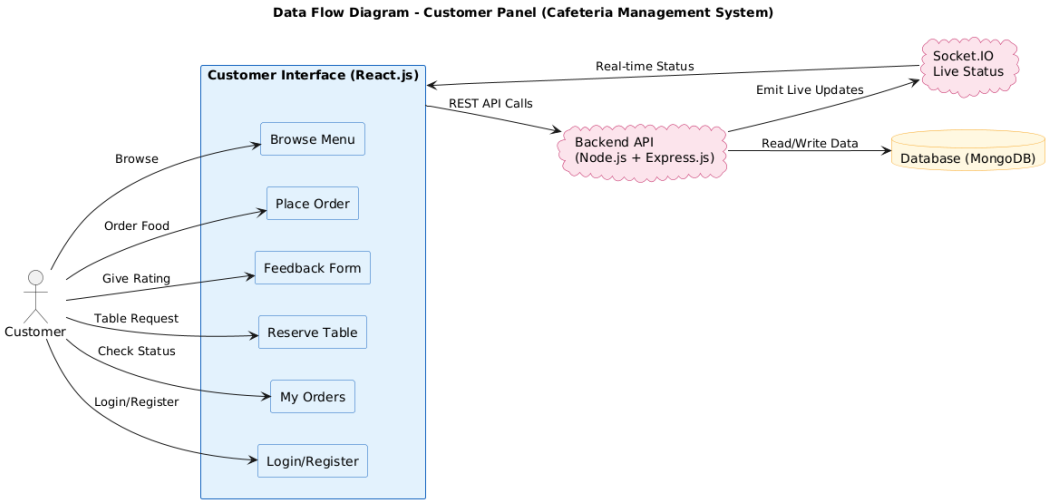

Figure 3.8: Customer Panel Data Flow Diagram

Deployment Diagram

The deployment diagram illustrates the physical deployment of system components, including client devices and backend services involved in customer operations.

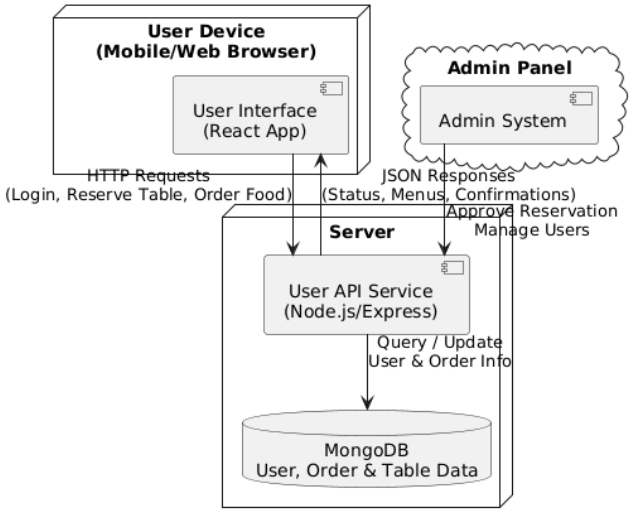

Figure 3.9: Customer Panel Deployment Diagram

System Flowchart

This flowchart represents the overall flow of customer activities such as login, browse menu, reserve table, place orders, and track delivery.

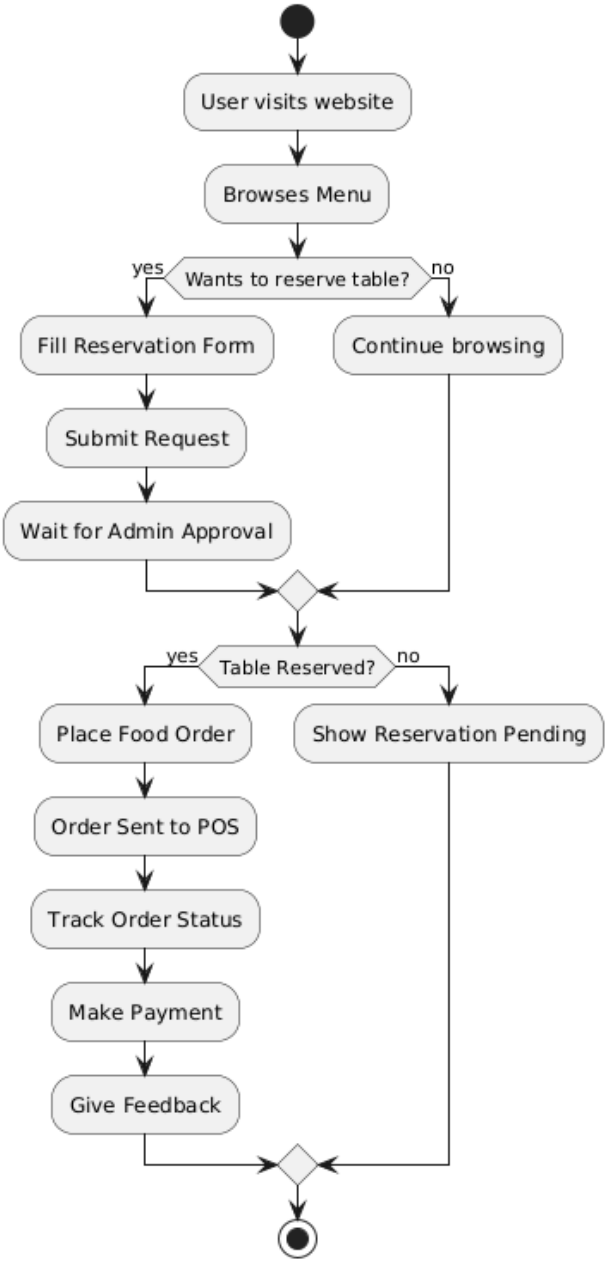

Figure 3.10: Customer Panel System Flowchart

3.5.2 Admin Panel

The Admin Panel serves as the central management hub of the Cafeteria Management System. It empowers administrators to control all major aspects of the system, including inventory, staff, menu, reservations, and customer feedback.

- Add, update, or delete food items from the system
- Approve or reject table reservation requests
- Monitor and manage inventory and kitchen supplies
- View user feedback and generate sales or performance reports
- Manage staff profiles and define role-based access

Admin Login

This is the secure login interface for administrators. Access is restricted to authorized staff only, with role-based JWT authentication ensuring data security.

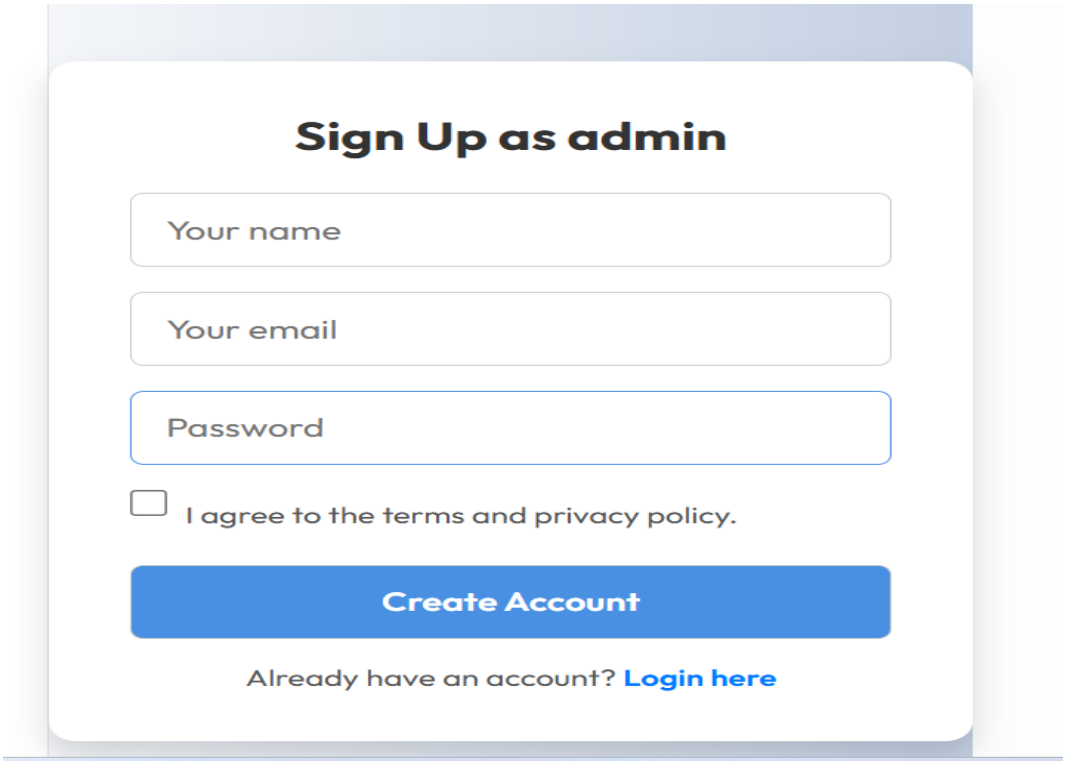

Figure 3.11: Admin Authentication Interface

Product Management

Admins can add new food items with necessary details, modify existing ones, or remove them from the system. This section keeps the menu updated and relevant.

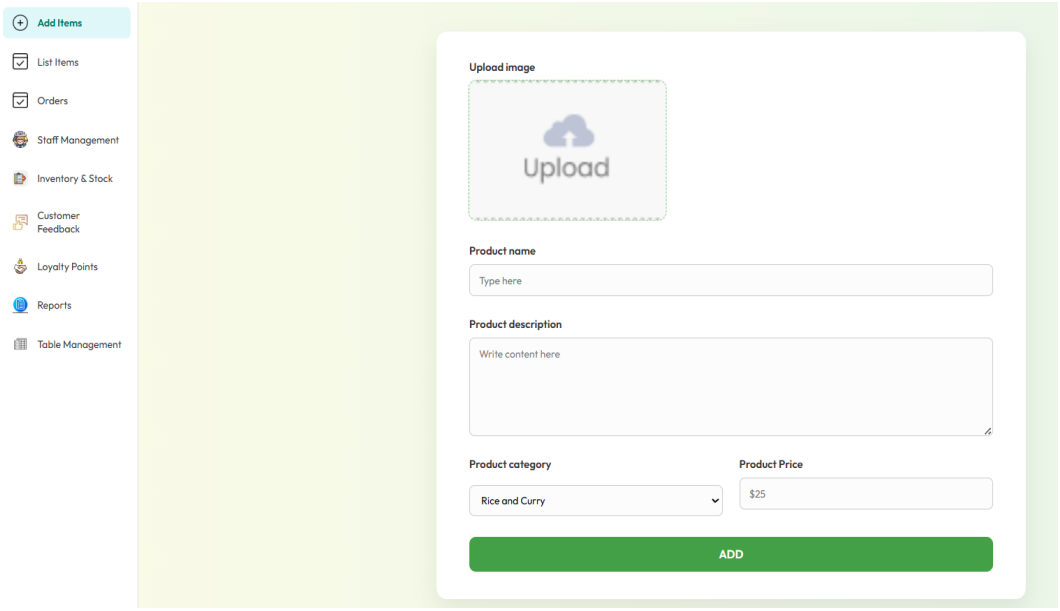

Figure 3.12: Adding a New Product

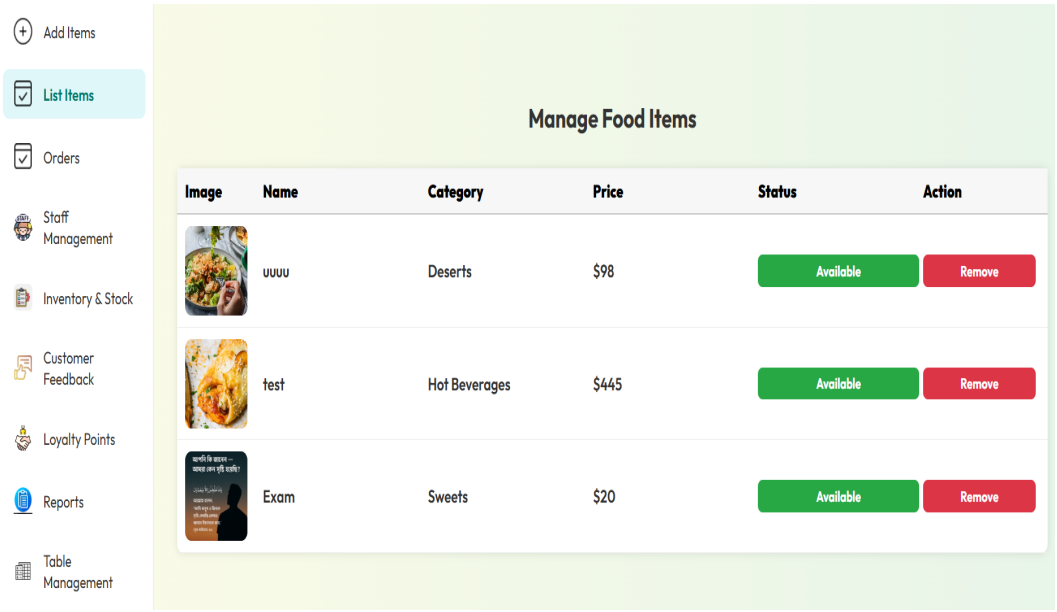

Figure 3.13: Modifying Existing Product

Feedback Management

This dashboard allows admins to view customer reviews, ratings, and feedback. It helps in maintaining service quality and user satisfaction.

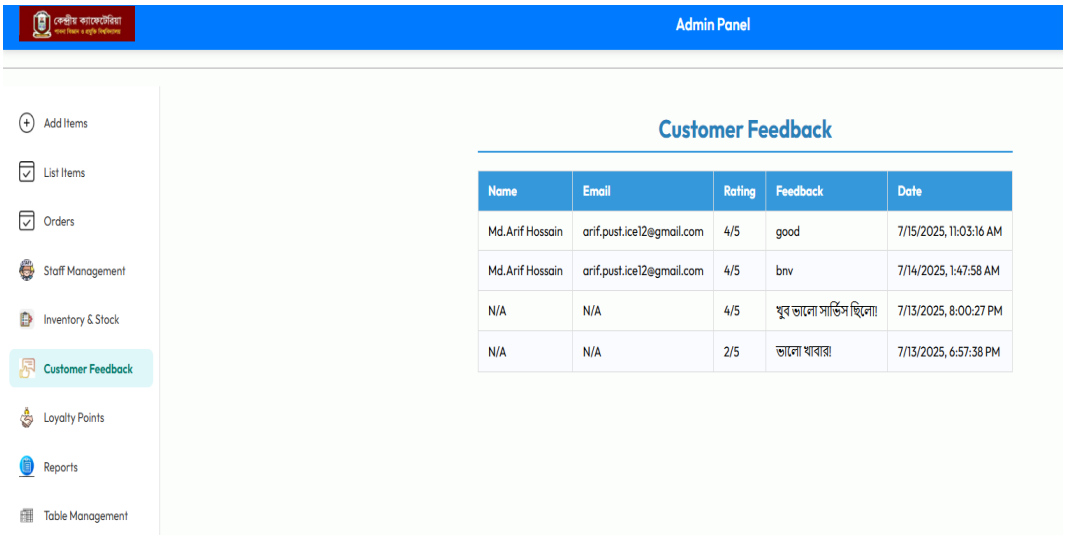

Figure 3.14: Customer Feedback Dashboard

Loyalty System

The loyalty system enables administrators to monitor and manage user reward points based on their purchases and interactions.

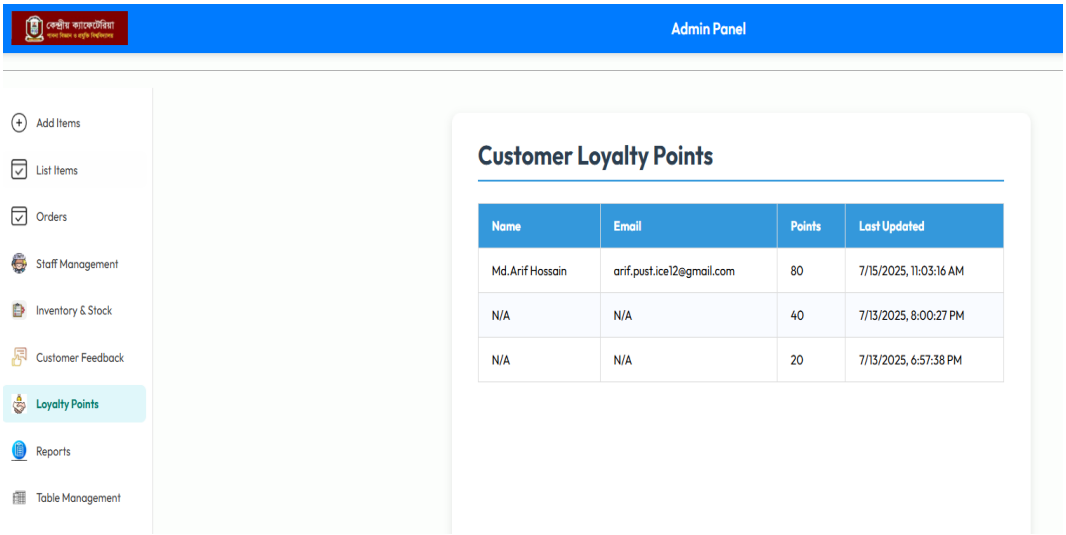

Figure 3.15: Loyalty Points Management

Order Monitoring

Admins can track all live orders, including their current status, associated table, and responsible waiter or kitchen staff.

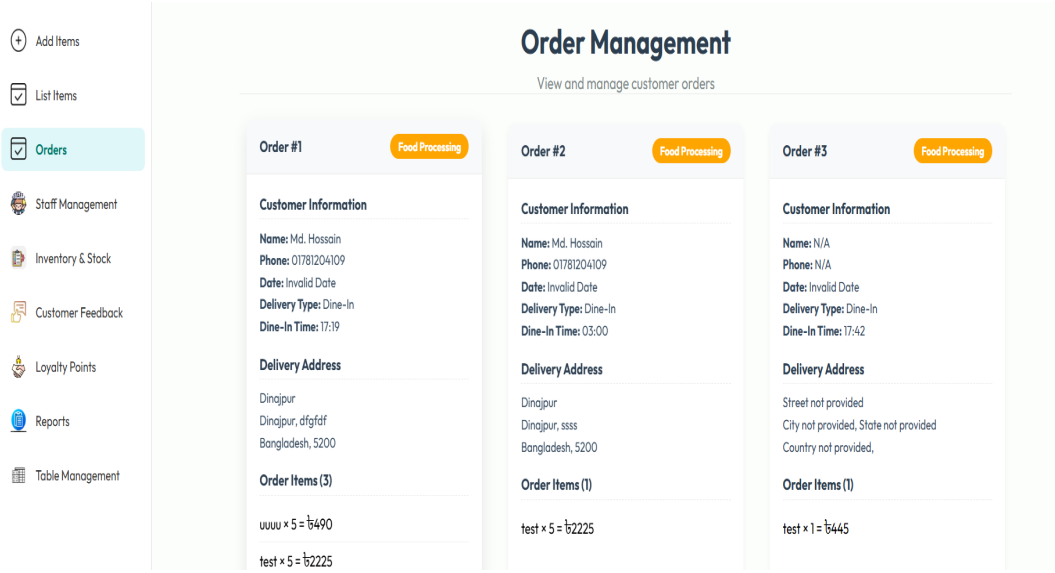

Figure 3.16: Live Order Tracking

Table Management

The Table Management module enables administrators to monitor and manage all cafeteria tables efficiently. Admins can view the current status of each table and handle reservation requests submitted by users. Once a request is approved, the corresponding table is marked as “Reserved” and this update is instantly reflected across all connected system panels.

Moreover, the system enhances operational insights by displaying the following information for each table:

- **Total Orders Placed:** Displays the cumulative number of orders associated with a specific table. This helps in analyzing table usage trends and identifying high-demand areas.
- **Idle Time Duration:** Shows the elapsed time since the last order was placed at the table. This feature is useful for detecting inactive tables and optimizing space utilization during peak hours.

Figure 3.17: Table Overview

Inventory Control

This section presents current stock levels and usage data. Admins can monitor ingredients and trigger restocking as needed.

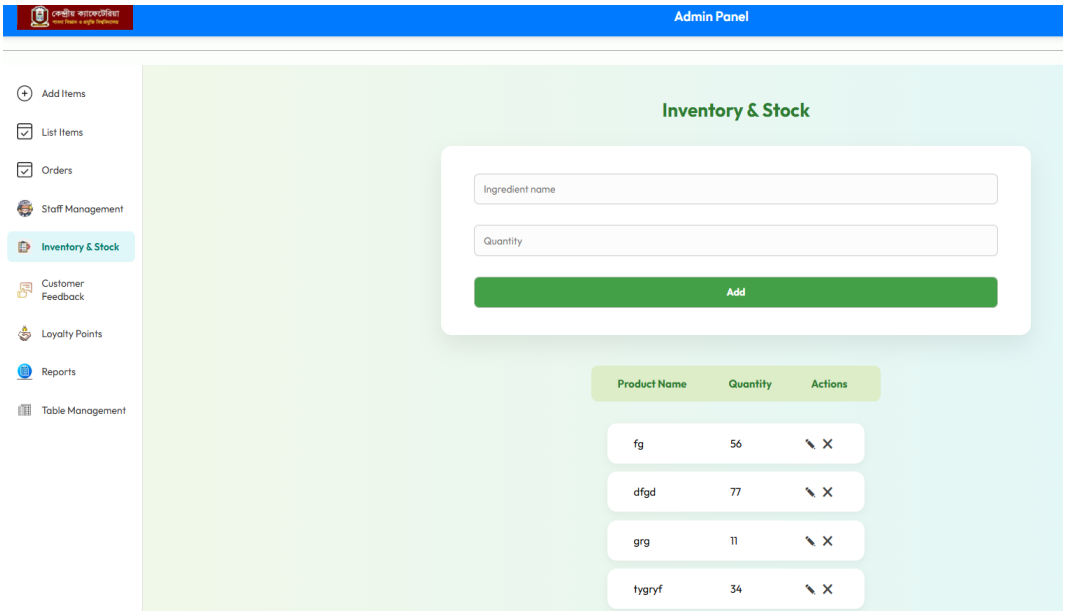

Figure 3.18: Inventory Overview

Staff Administration

Admins can add, edit, or remove staff accounts, assign roles (e.g., chef, waiter), and control their access to specific panels.

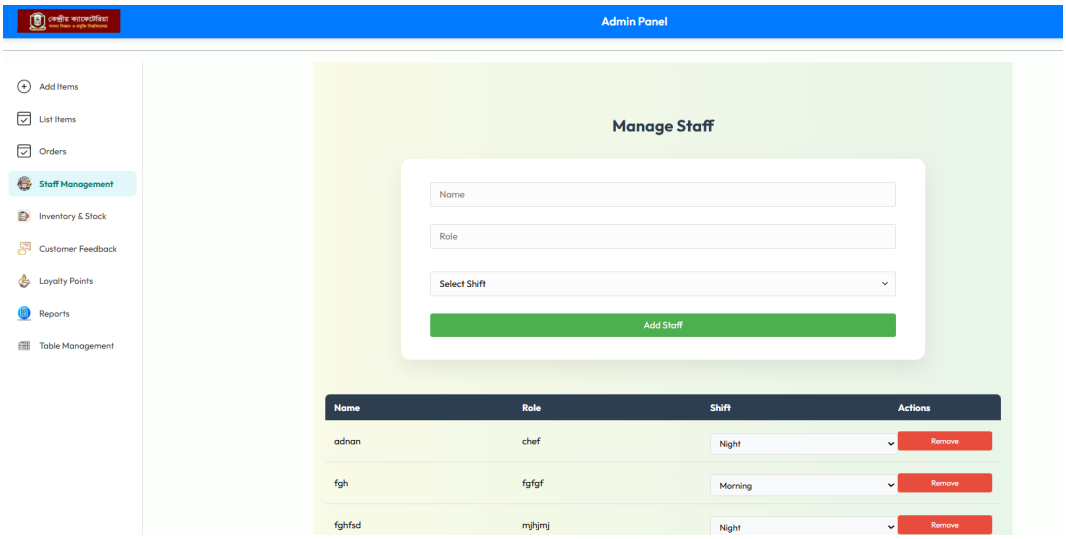

Figure 3.19: Staff and Role Management

Sales Analytics

Visual analytics and detailed reports help in understanding performance trends, top-selling items, and revenue flow.

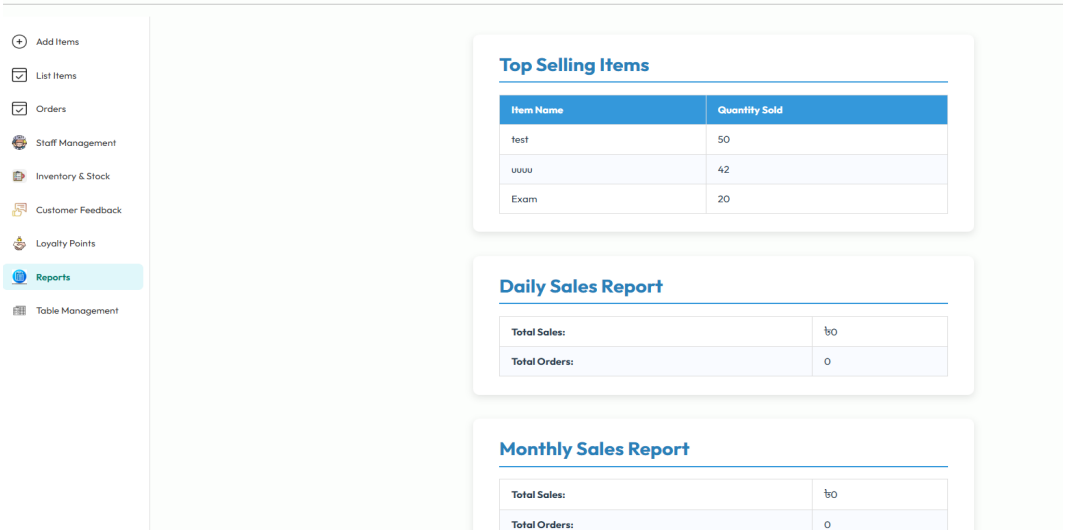

Figure 3.20: Sales Reporting and Insights

Technical Diagrams

The following diagrams offer a structured view of the Admin Panel's functional architecture, workflows, and deployment structure.

Use Case Diagram: Shows different operations available to the admin role.

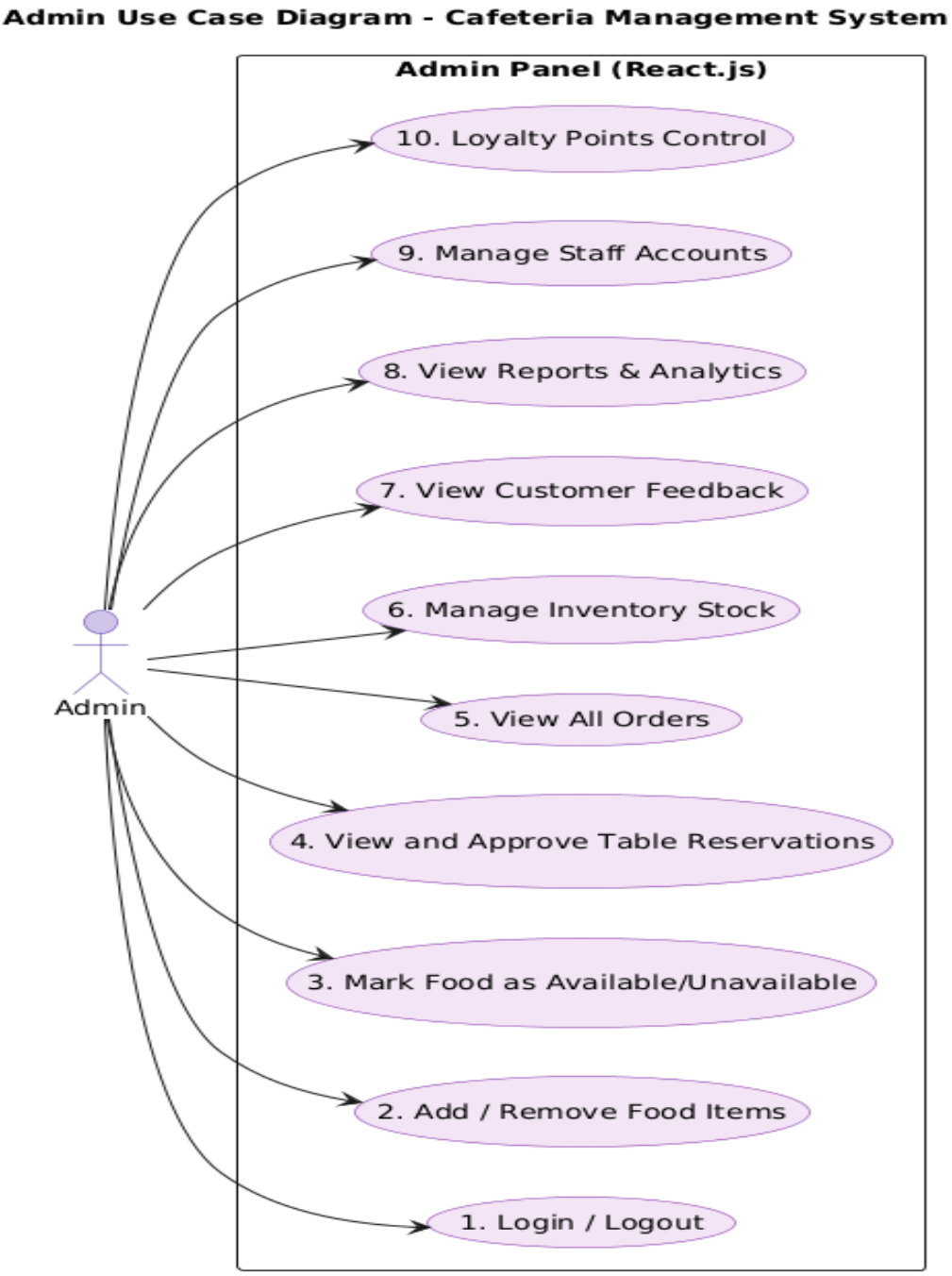

Figure 3.21: Admin Panel Use Case Diagram

Data Flow Diagram (DFD): Illustrates how data moves between the admin and system components.

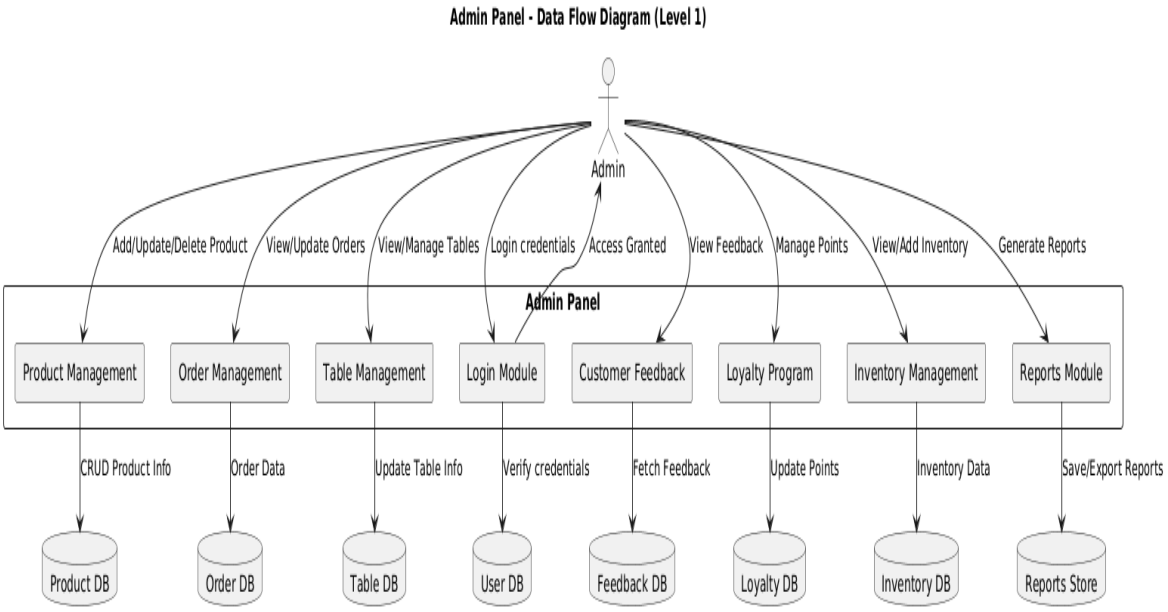

Figure 3.22: Data Flow Diagram for Admin Panel

Sequence Diagram: Depicts time-based interaction between admin and backend modules for operations like adding a product.

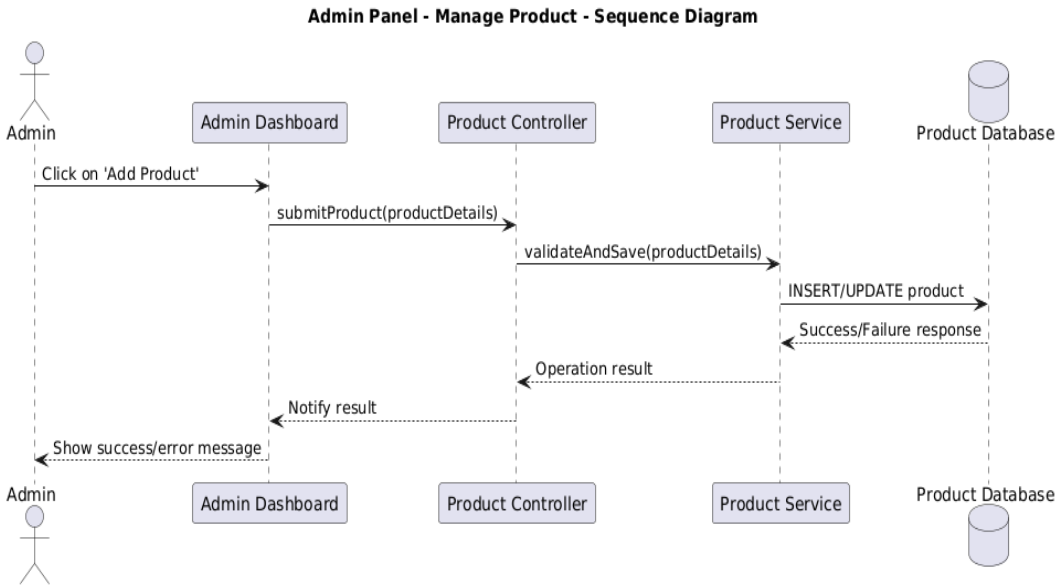

Figure 3.23: Admin Module Sequence Diagram

Activity Diagram: Represents decision-making and parallel tasks in admin workflows such as reservation handling.

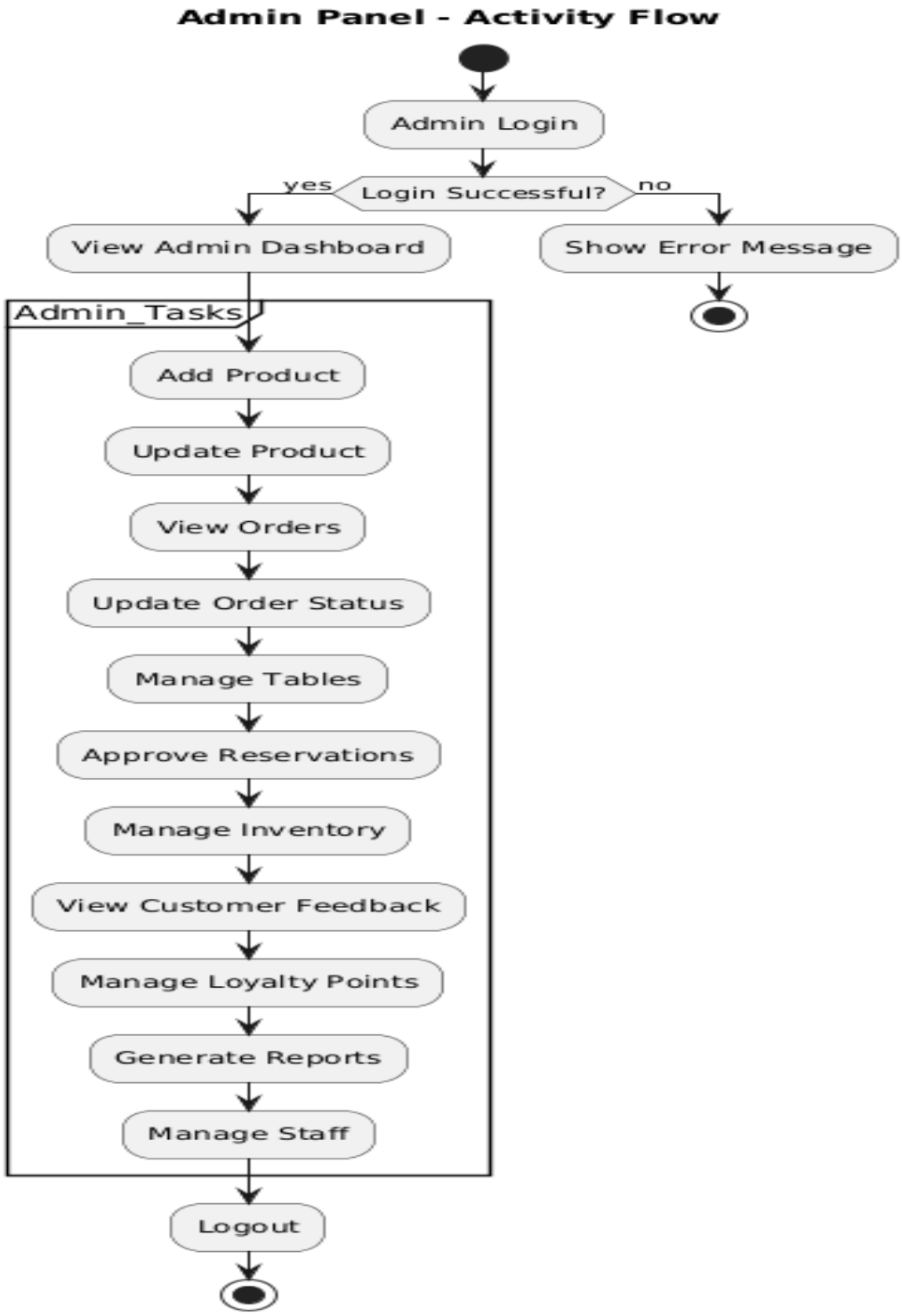

Figure 3.24: Admin Activity Flow

Deployment Diagram: Shows how admin interfaces and services are deployed across client-server infrastructure.

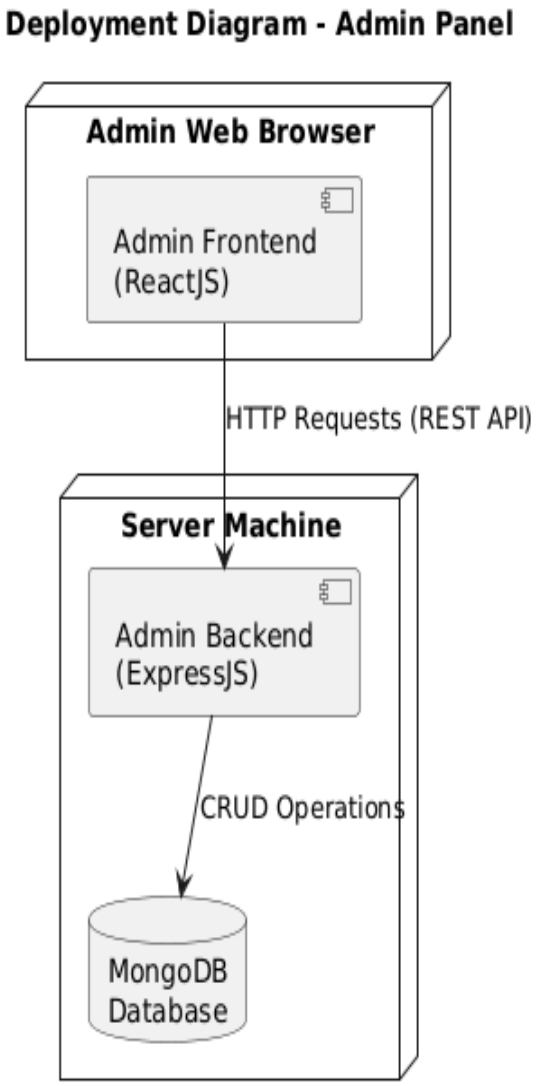

Figure 3.25: Admin Panel Deployment Diagram

3.5.3 POS Panel

The Point of Sale (POS) Panel is primarily used by cafeteria staff, such as waiters, to handle dine-in services efficiently. It allows staff to manage table assignments, place orders, communicate with the kitchen, and track order statuses in real-time.

Key Functionalities

- **Table Selection:** Waiters can view all tables and select available ones for service.
- **Menu-based Ordering:** Food items are organized by category for easy selection.
- **Cart and Placement:** Orders can be added to a cart and placed directly from the interface.
- **Real-time Order Updates:** Orders sent to the kitchen can be tracked until served.

Tables Overview

The POS dashboard displays a comprehensive table management interface where staff can monitor table statuses and initiate orders for available tables.

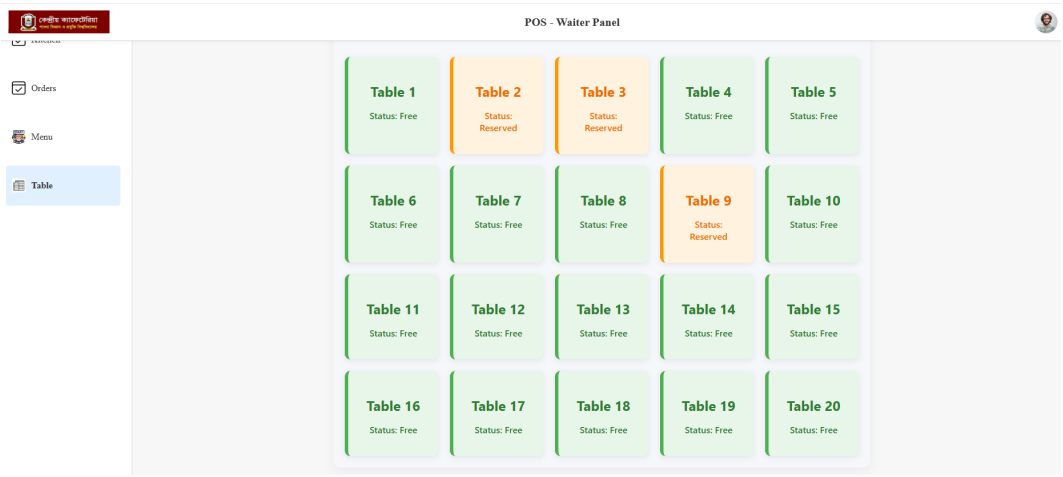

Figure 3.26: POS Table Management Interface

Menu Display

Once a table is selected, the menu is shown, categorized by food types. This interface enables quick item selection for order placement.

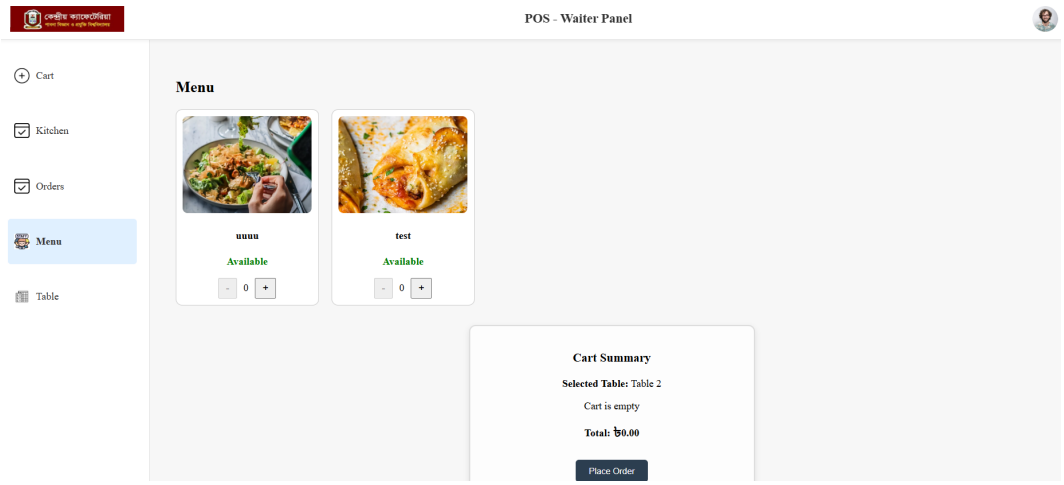

Figure 3.27: Food Menu Selection Screen

Cart and Order Placement

After selecting items, they are listed in the cart. The staff can review and confirm the order for submission to the kitchen.

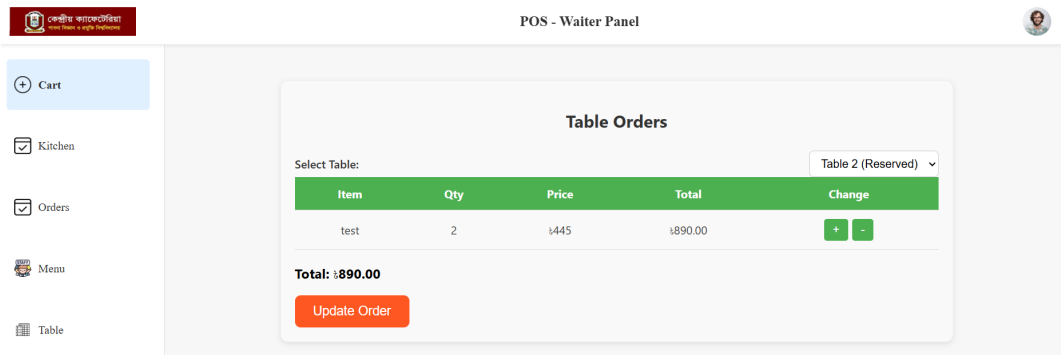

Figure 3.28: Cart Overview and Order Placement

Order Monitoring

All placed orders are displayed with status updates such as “preparing” or “ready.” This helps waiters serve the food on time.

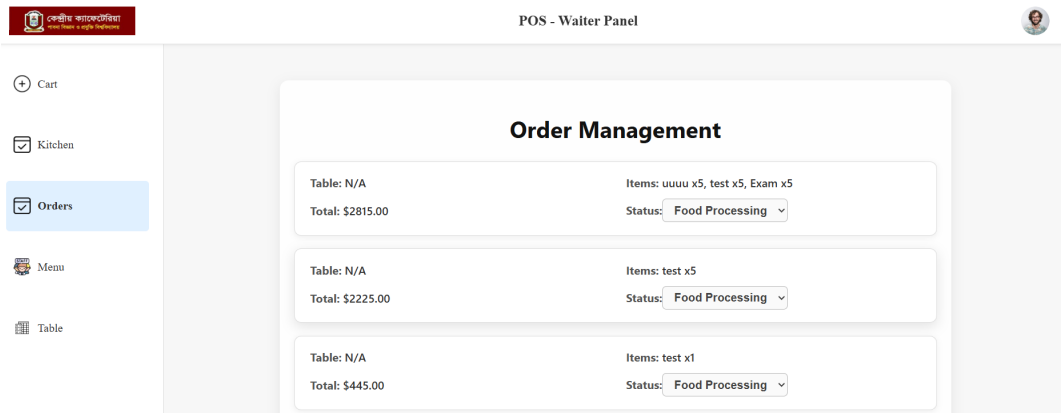

Figure 3.29: Track and Monitor Orders

POS Panel Diagrams

Use Case Diagram: Illustrates the interaction between waiters and the system, showing core functionalities like placing orders and tracking them.

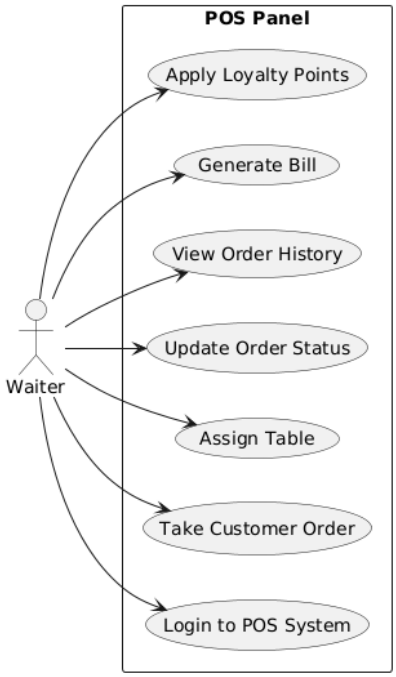

Figure 3.30: POS Use Case Diagram

Data Flow Diagram (DFD): Shows how data flows from table selection through ordering to delivery, involving both frontend and backend components.

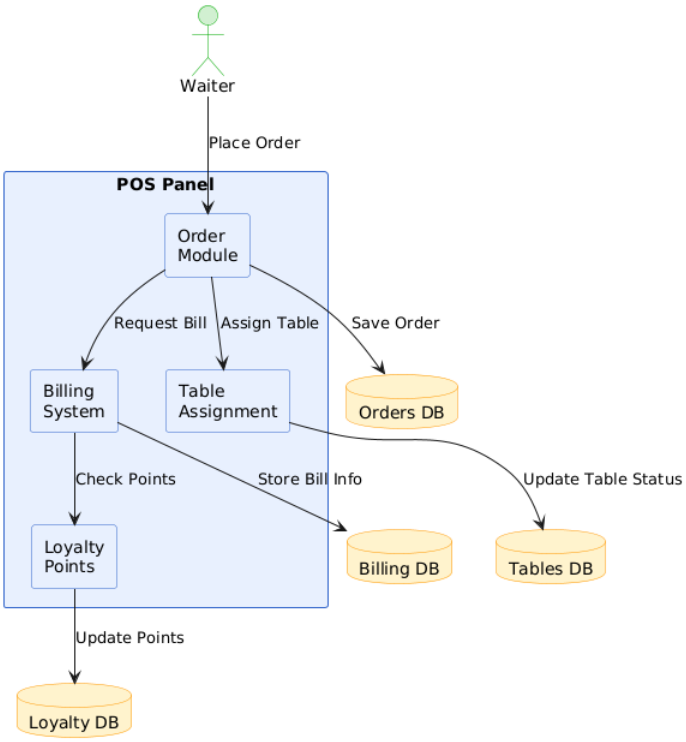

Figure 3.31: POS Data Flow Diagram

Sequence Diagram: Represents the sequence of interactions among waiter, POS interface, order system, and kitchen during the order lifecycle.

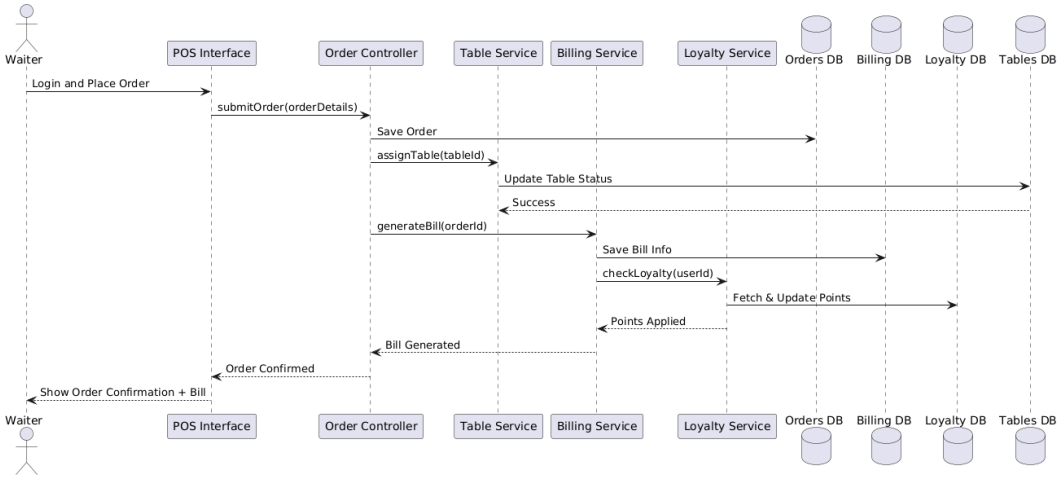

Figure 3.32: POS Order Processing Sequence

3.5.4 Kitchen Panel

The Kitchen Panel is designed for chefs and kitchen staff to efficiently manage food preparation tasks. It ensures seamless communication with the POS and Admin panels, improving the overall workflow from order placement to delivery.

- Receive real-time order notifications
- View detailed order lists including table number and items
- Update order status to *Preparing* or *Ready*
- Notify POS and Admin panels upon food readiness

Update Order Status

The Kitchen Panel allows staff to update the progress of food preparation. As orders arrive, chefs can change the status to "Preparing" and later to "Ready." This triggers real-time updates to other panels, enabling efficient coordination with serving and billing processes.

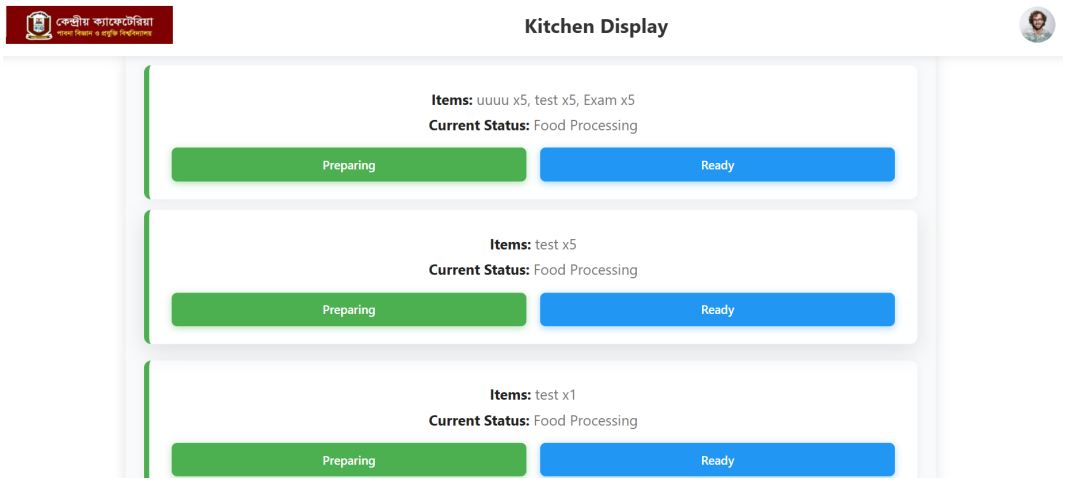

Figure 3.33: Update Order Status (Preparing or Ready)

Kitchen Diagrams

Use Case Diagram: The use case diagram below illustrates the major actions a kitchen user can perform within the system, including receiving orders, updating statuses, and notifying other panels.

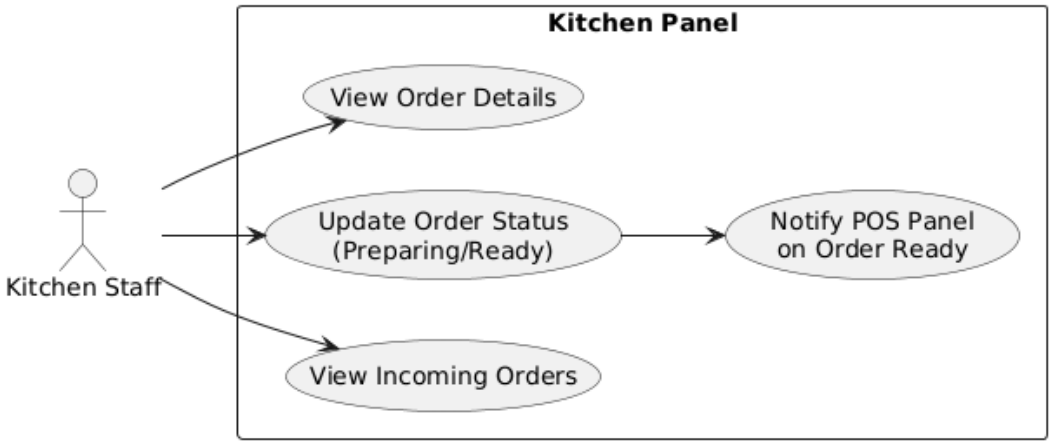

Figure 3.34: Kitchen Use Case Diagram

System Flowchart: The flowchart presents the step-by-step operation inside the kitchen system, starting from order reception to readiness confirmation. It ensures clarity in kitchen workflows.

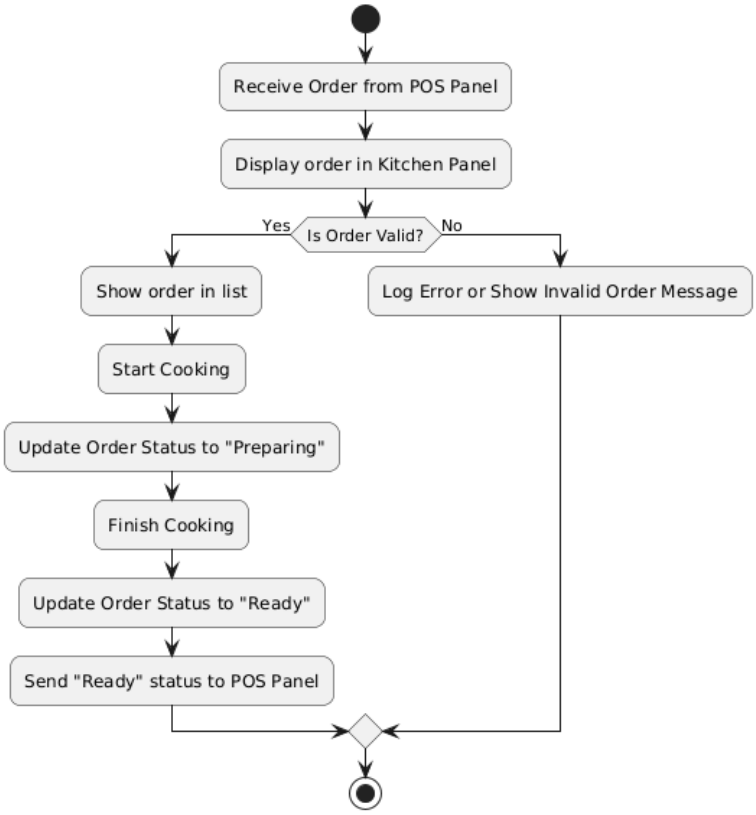

Figure 3.35: Kitchen System Flowchart

Deployment Diagram: This deployment diagram shows how the Kitchen Panel interacts with the server and database. It reflects the panel's integration in the overall system infrastructure.

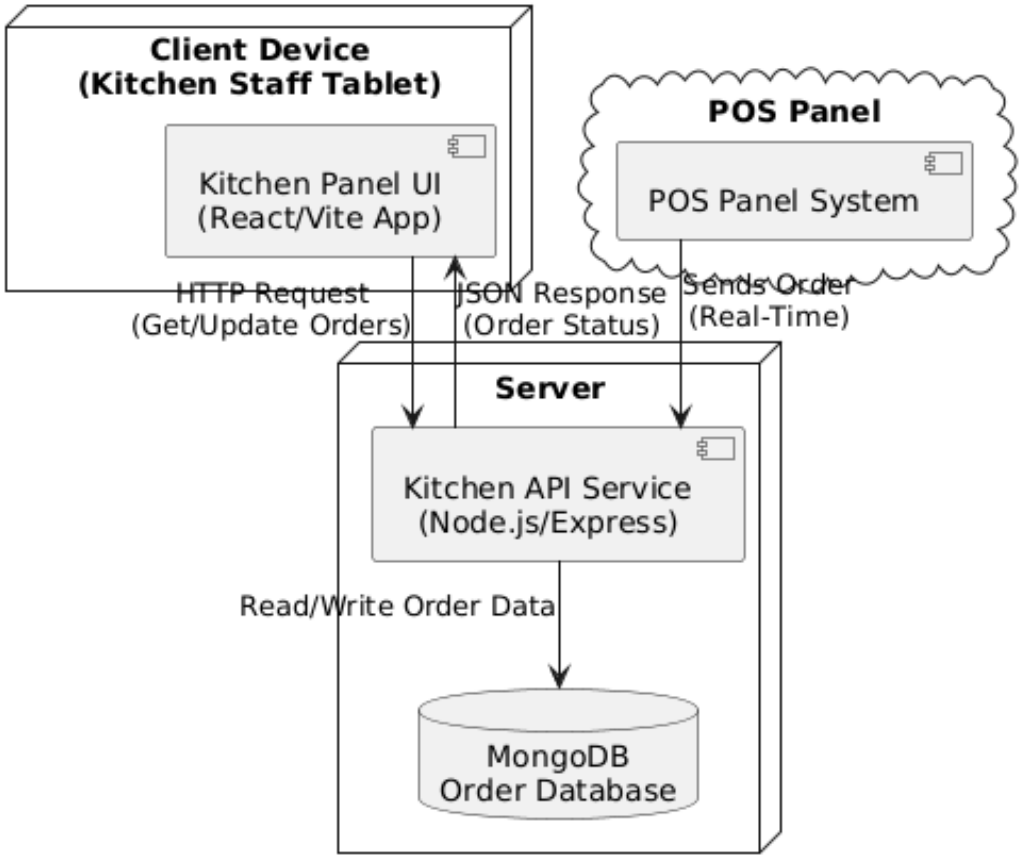

Figure 3.36: Kitchen Deployment Diagram

3.5.5 Billing Panel

The Billing Panel is used by the cashier or billing staff to finalize orders, process payments, and manage invoices. This panel plays a critical role in ensuring smooth order completion and table availability for the next customer.

- Generate invoices and print/download as PDF
- Update payment status (Paid/Unpaid)
- Automatically release table upon payment

Active Orders Billing

This section of the panel displays all active orders that are ready for billing. It allows the cashier to select an order, verify the items, calculate totals, and proceed with billing. The interface provides clarity on table numbers, food items, and order totals.

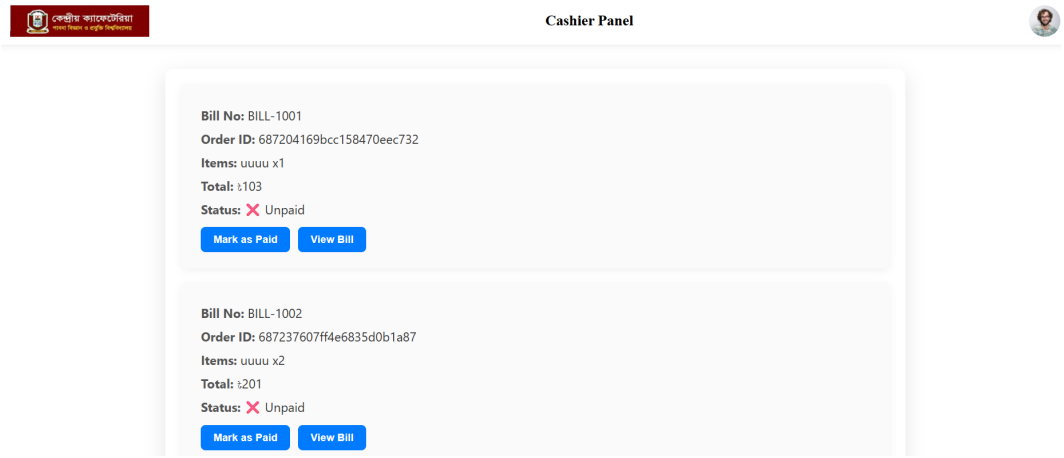

Figure 3.37: Billing and Invoice Processing

Generate Invoices and Print

After verifying the order details, the billing staff can generate an invoice, which can be printed or downloaded as a PDF. The system automatically updates the payment status and frees the table upon successful transaction.

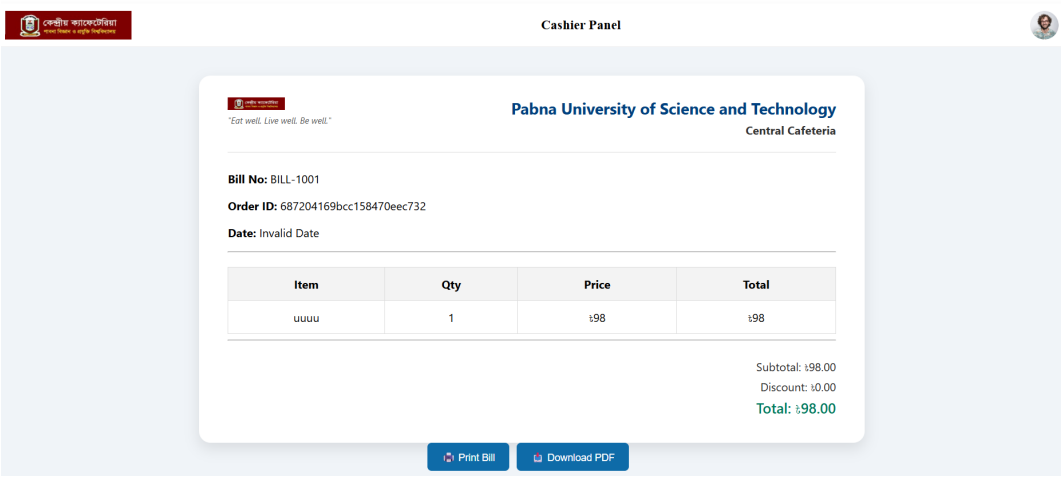

Figure 3.38: Generate Invoices and Print

Billing Diagrams

Use Case Diagram: This diagram outlines the key interactions the billing staff performs in the system, including accessing active orders, generating invoices, and marking payments as completed.

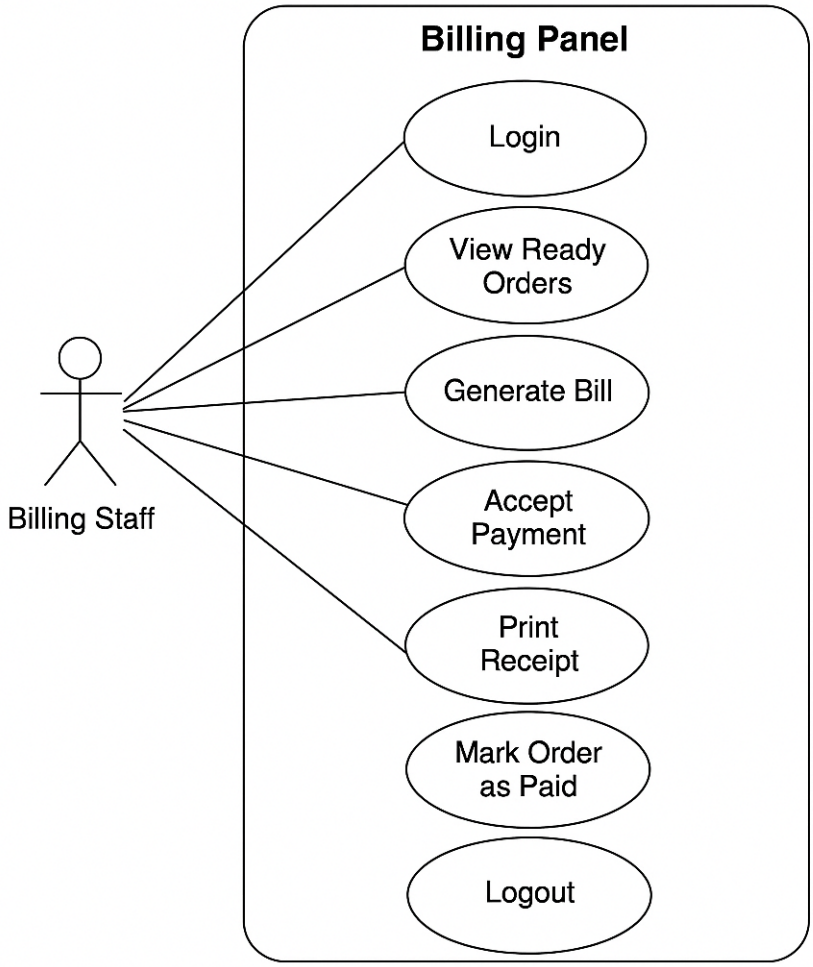

Figure 3.39: Billing Use Case Diagram

System Flowchart: The flowchart below demonstrates the sequential operations of the billing system, from retrieving an order to confirming payment and releasing the table.

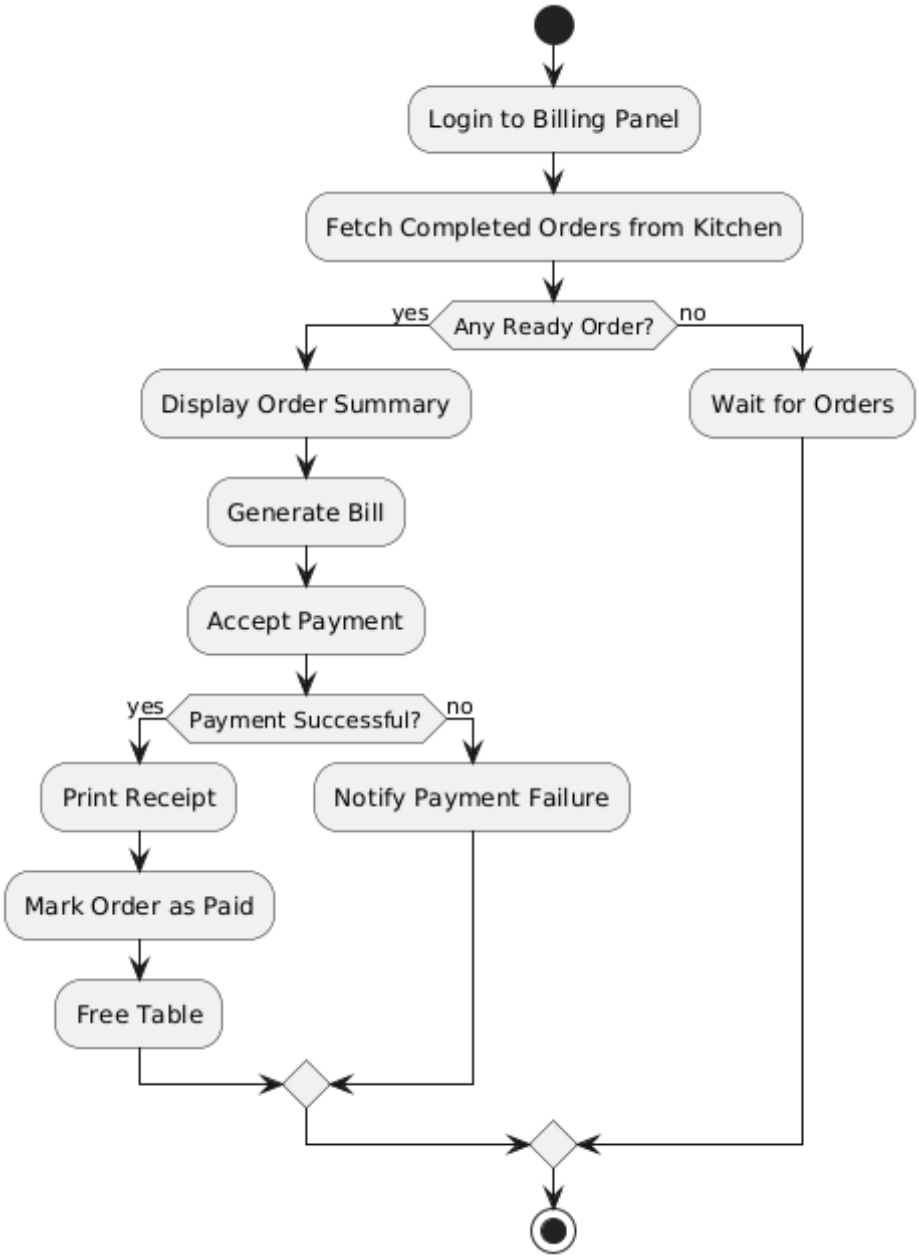

Figure 3.40: Billing System Flowchart

Deployment Diagram: This diagram visualizes how the billing system is deployed within the application infrastructure. It shows the interactions between the billing panel, server, and database.

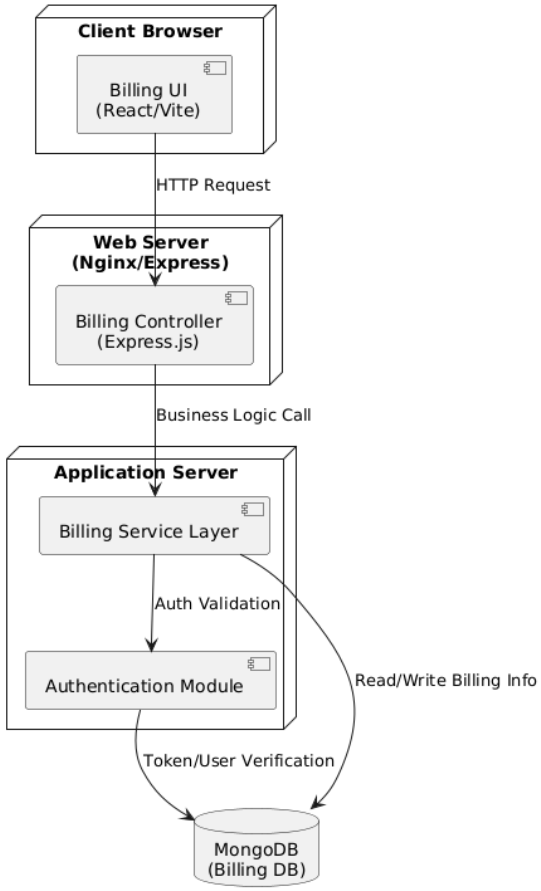

Figure 3.41: Billing Deployment Diagram

3.6 System-wide Diagrams

3.6.1 Use Case Overview

This diagram illustrates all user roles—Admin, POS Staff, Kitchen Staff, Billing Staff, and Customers—and the key functionalities accessible to each role. It helps in visualizing how different actors interact with the system.

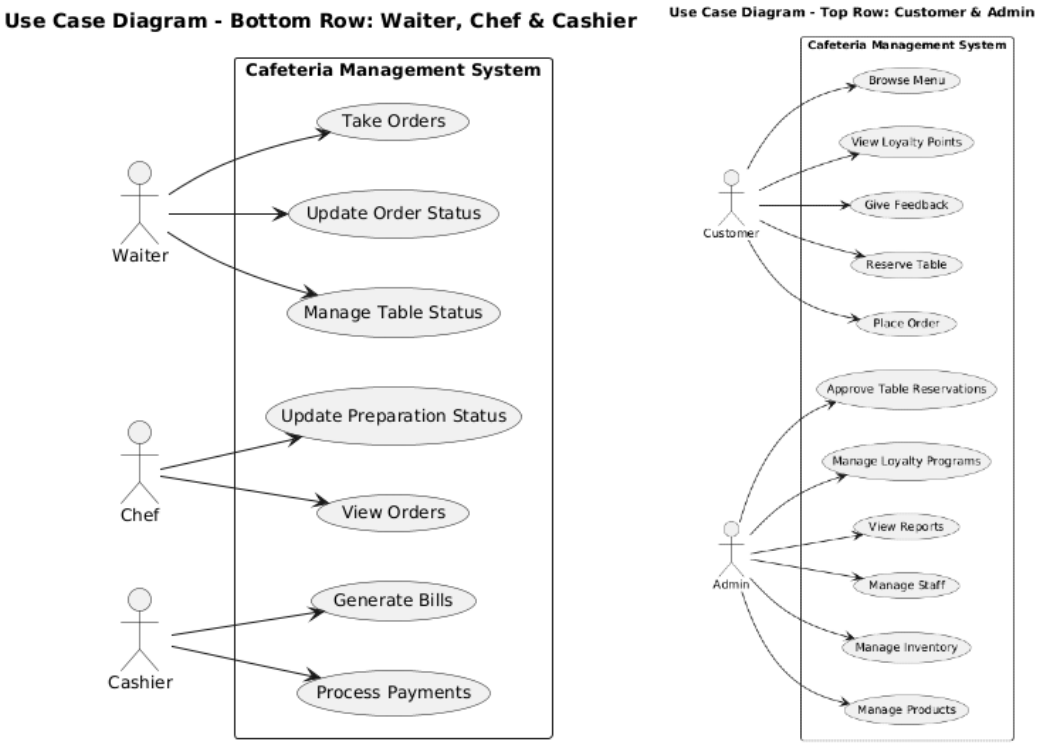

Figure 3.42: Comprehensive Use Case Diagram

3.6.2 Entity-Relationship Diagram

The ER diagram represents the structure of the database, showcasing tables like Users, Orders, Food Items, Reservations, and their interrelationships. It helps in understanding how data flows and connects across the system.

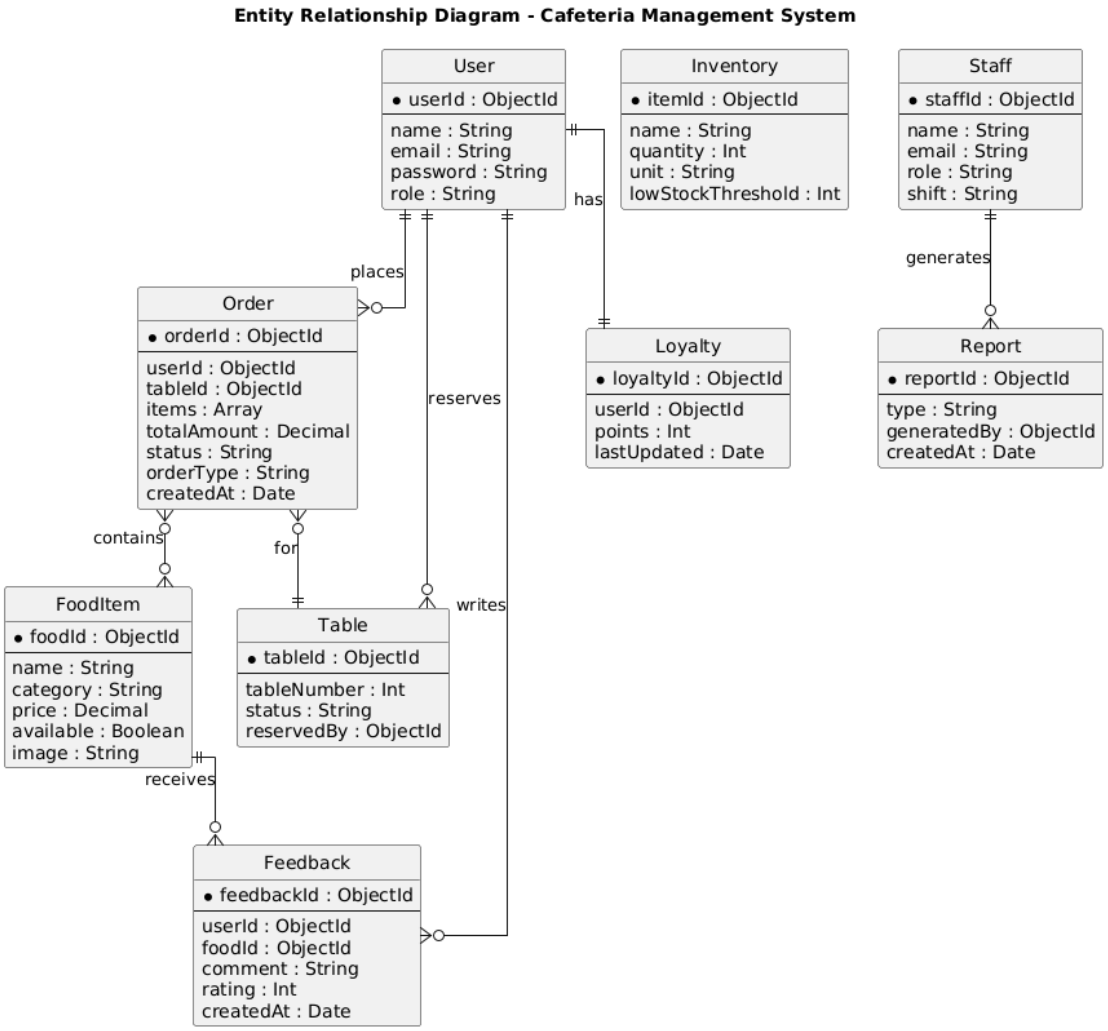

Figure 3.43: Database Schema (ER Diagram)

3.6.3 Component Architecture

This diagram visualizes the major logical components such as Frontend, Backend API, Authentication, Database, and third-party services (e.g., Stripe for payments), and how they interact with each other. It provides a high-level view of the technical architecture.

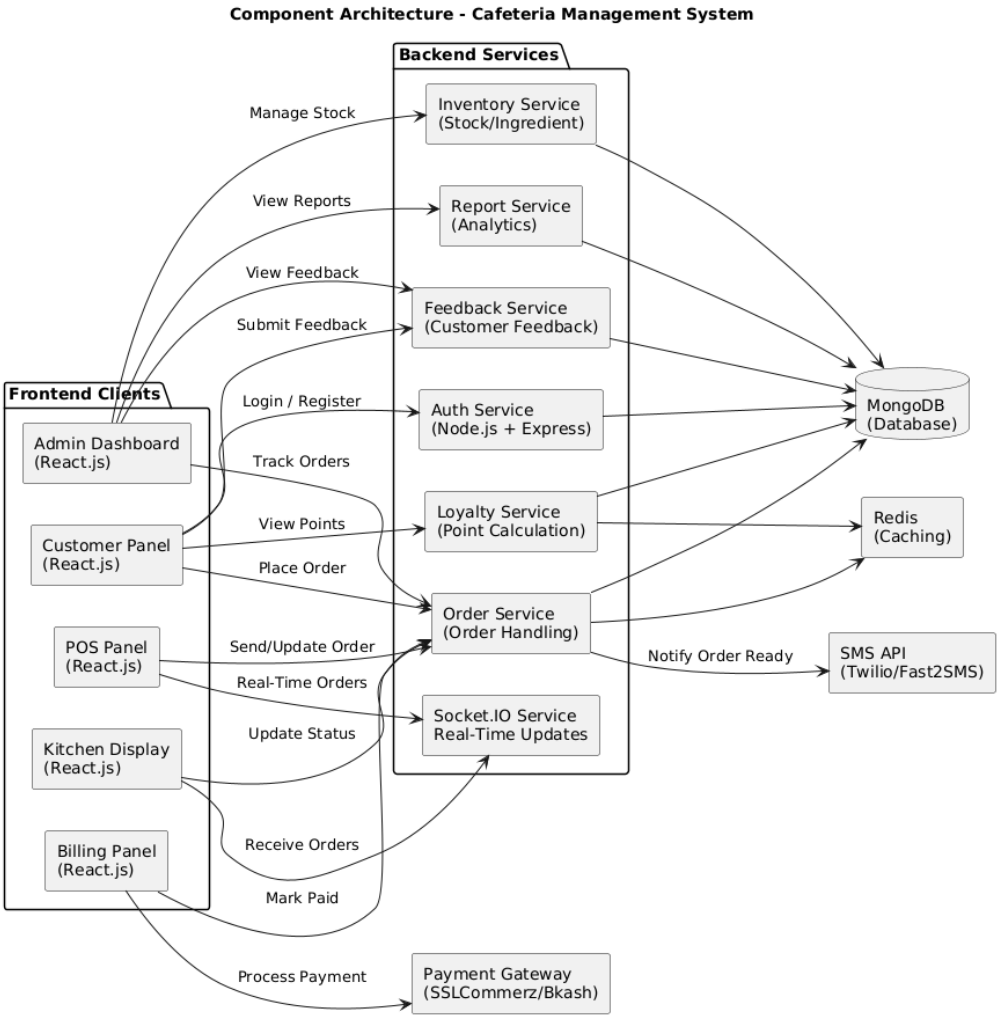

Figure 3.44: System Component Architecture

3.7 Directory Structure

The following table outlines the directory structure of the complete project, highlighting the modular separation across panels for better maintainability and role-based access control.

Table 3.1: Project Directory Layout

Directory	Description
`backend/controllers/`	Contains logic for various APIs such as food management, orders, authentication, and reservations.
`frontend/src/pages/`	User-side pages including Home, Cart, Menu, Feedback, and Reservation features.
`admin/src/pages/`	Admin dashboard pages for managing food items, staff accounts, reservation approvals, and analytics.
`pos/src/pages/`	POS panel pages for table selection, placing orders, managing carts, and real-time order updates.
`kitchen/src/pages/`	Kitchen panel interface where chefs view incoming orders and update their preparation status.
`billing/src/pages/`	Billing panel to generate PDF invoices, mark payments as complete, and auto-release tables.

3.8 Summary

This chapter presented a comprehensive overview of the architectural structure of the Cafeteria Management System. It included system-wide diagrams such as the use case diagram, entity-relationship diagram, and component architecture to describe the overall design and workflow. The modular directory layout ensures ease of development and scalability. The use of real-time communication, well-structured database design, and component-based segmentation make the system robust, extensible, and suitable for modern food service environments.

Chapter 4

Implementation

4.1 Introduction

This chapter presents the practical implementation of the proposed Cafeteria Management System, developed to streamline food service operations in institutional or commercial environments. The system is divided into five modular panels—Admin, POS (Point of Sale), Kitchen, Billing, and Customer—each implemented as a standalone React application that communicates with a centralized backend via RESTful APIs and real-time WebSocket channels. The backend architecture is built using Express.js and MongoDB to ensure scalability, modularity, and real-time responsiveness. The goal is to deliver a role-based digital solution that enhances operational efficiency, improves user interaction, and ensures data-driven coordination across all functional units.

4.2 System Overview

The system architecture follows the MERN stack paradigm—MongoDB, Express.js, React.js, and Node.js—selected for its robust ecosystem, community support, and compatibility for full-stack JavaScript development. Each panel is designed to fulfill a specific operational role, integrated via a shared backend for unified data access and communication. This modular structure ensures clear role separation, maintainability, and real-time synchronization across components.

4.2.1 Technology Stack

The frontend is developed using React.js, enhanced by Vite for optimized build performance. Routing is handled by React Router DOM, and Axios is used for HTTP communication. TailwindCSS is employed for responsive UI design. The backend employs Express.js and Socket.IO for API and real-time capabilities, respectively. MongoDB serves as the cloud-hosted NoSQL database. JWT handles authentication, while third-party tools such as Cloudinary (for media), Stripe (for payments), Puppeteer (for PDF generation), and Thunder Client (for API testing) complement the ecosystem.

4.3 Backend Implementation

The backend is structured into modular directories including controllers, routes, middleware, and configuration files. Each controller handles its corresponding entity (e.g., food, order, reservation) with clear separation of concerns.

4.4 Backend Implementation

Directory Structure

```
backend/

 controllers/
    authController.js
    foodController.js
    orderController.js
    reservationController.js
    ...
 models/
    userModel.js
    orderModel.js
    foodModel.js
    ...
 routes/
```

```
    authRoute.js
    foodRoute.js
    ...
middleware/
    authMiddleware.js
config/
    db.js
server.js
.env
```

Key Features: The backend supports user authentication via JWT, REST API endpoints for CRUD operations, role-based access control, real-time updates using Socket.IO, and payment processing via Stripe. Admin functionalities include dashboard analytics, food item management, and reservation control.

4.5 Frontend Panels

Customer Panel

This panel allows users to browse food items, place orders, submit feedback, and request table reservations. Users can register under specific roles (Student, Teacher, or Organization) which tailor the reservation form accordingly.

POS Panel

The POS panel is designed for waiters and staff to manage table assignments, create and monitor orders, and receive kitchen updates in real time. It also displays reservation statuses for each table.

Kitchen Panel

This panel receives orders directly from the POS panel via Socket.IO. It enables staff to update order statuses (e.g., Preparing, Ready) and notifies POS instantly upon status changes.

Admin Panel

The admin panel includes a secure login and privileges to manage inventory, approve/reject reservations, monitor order flows, analyze feedback, and generate reports.

Billing Panel

The billing panel accesses delivered orders and finalizes payment using Stripe or cash. It provides downloadable PDF invoices generated via Puppeteer, and updates order status accordingly.

4.6 Database Schema Overview

The system maintains a NoSQL schema using MongoDB, consisting of interconnected collections:

- **Users:** Stores user credentials, roles, and session tokens.
- **FoodItems:** Maintains product data including image links and ratings.
- **Orders:** Tracks each placed order along with status, timestamp, and table ID.
- **Reservations:** Records table booking requests and approval statuses.
- **Feedback:** Captures customer reviews and ratings.

4.7 Authentication and Security

User authentication is implemented using JWT. Tokens are stored securely on the client and validated via middleware on protected routes. Role-based access control restricts unauthorized operations. CORS policies ensure cross-domain request safety.

4.8 Real-Time Communication

Socket.IO facilitates real-time bidirectional communication, primarily between the POS and Kitchen panels. Events such as `newOrder`, `orderUpdate`, and

`statusChange` ensure order progression is reflected without delay.

4.9 PDF Billing Generation

Invoices are generated from HTML templates on the backend using Puppeteer. These invoices are available as downloadable PDFs and can be optionally sent via email. Each invoice contains item-wise details, taxes, table ID, timestamp, and transaction mode.

4.10 Challenges Faced

Development involved several challenges:

- **Socket.IO Coordination:** Synchronizing event emissions between panels to reflect accurate status in real-time.
- **Multi-role Authentication:** Handling login for distinct roles with secure session management.
- **Table Reservation Workflow:** Designing a reservation system involving multi-stage approval and feedback.
- **PDF Styling:** Ensuring layout consistency across devices during PDF rendering.
- **Micro-frontend Routing:** Managing conditional routing within each frontend based on role-specific access.

4.11 Conclusion

This chapter has elaborated the complete implementation of the Cafeteria Management System. The project demonstrates how modular design, secure API communication, real-time synchronization, and scalable architecture can be integrated to develop an intelligent cafeteria platform. Each panel's independence ensures maintainability, while the shared backend promotes efficient data exchange and consistency.

Chapter 5

Testing and Results

5.1 Introduction

This chapter presents the testing and validation process of the Cafeteria Management System. The testing phase aimed to verify that all individual modules, components, and integrated parts function as intended and fulfill the system's functional and non-functional requirements. A variety of testing methodologies were applied to ensure system reliability, security, and user satisfaction.

5.2 Testing Methodologies

Multiple levels of testing were employed throughout the development lifecycle to validate both backend and frontend functionalities:

- **Unit Testing:** Backend routes and frontend component logic were individually tested to ensure correctness.
- **Integration Testing:** Communication among frontend, backend, and database layers was verified using API testing tools like Postman and Thunder Client.
- **System Testing:** Ensured that the complete system adhered to the specified functional and performance requirements.
- **User Acceptance Testing (UAT):** Conducted with a small group of real users, including students and cafeteria staff, to validate the system's usability and usefulness.

5.3 Functional Testing

All five system panels were tested thoroughly to validate core functionalities. The test summary is provided in Table 5.1.

Table 5.1: Functional Testing Summary by Panel

Panel	**Tested Features**
User Panel	Menu browsing, category filtering, table reservation, order placement, dine-in scheduling, feedback and rating submission, loyalty point tracking
Admin Panel	Manage food items (add/update/delete), approve/reject reservations, view feedback, generate reports
POS Panel	Place orders, update order status, view real-time table status
Kitchen Panel	View new orders, update preparation status, send ready notifications to POS/Admin
Billing Panel	Generate downloadable PDF invoices, mark payment completion, release reserved tables

5.4 Test Case Sample

To validate the core functionality of the Cafeteria Management System, various test cases were executed. These covered major user interactions such as login, table reservation, order tracking, and billing. Testing was done using tools like Postman, manual UI verification, and database inspection.

Representative test cases across critical user journeys are listed in Table 5.2. Each test case includes the scenario, expected behavior, test tools/methods used, and the actual result.

Table 5.2: Sample Test Cases

Scenario	Expected Outcome	Test Performed	Result
User logs in with valid credentials	Dashboard is loaded with JWT token issued	Postman used to verify token flow	Passed
Student reserves table with dine-in time	Reservation appears in Admin panel with specified time	Database verified for accuracy	Passed
Kitchen updates order to ready	POS reflects new order status in real time	Socket.io real-time test verified	Passed
Admin deletes food item	Menu updates instantly without item	Tested with JWT-secured DELETE route	Passed
Billing panel generates invoice	PDF downloads successfully	Verified file type and layout integrity	Passed

The results confirm that the system performs as intended under various user scenarios. All tested components passed their respective criteria, ensuring system robustness and reliability in real-world use.

5.5 Non-Functional Testing

5.5.1 Performance

All RESTful APIs responded within 300ms under standard load. The system maintained responsiveness under concurrent usage by 20+ users interacting with different panels simultaneously.

5.5.2 Security

Security was enforced at both route and application levels:

- JWT-based authentication for all protected APIs
- Role-based access control (RBAC) to restrict unauthorized access

- Frontend gracefully handles token expiration with auto-logout

5.5.3 Usability

A usability test was conducted with 5 participants from the student community and cafeteria staff. Key findings included:

- Clear and intuitive navigation across panels
- Positive feedback for the dine-in scheduling and feedback features

5.5.4 Responsiveness

Responsive design was validated using:

- Desktop browsers (Chrome, Edge, Firefox)
- Android smartphones (Chrome browser)
- Tablets (both portrait and landscape)

All layouts adjusted correctly with no visual glitches.

5.6 Bug Tracking and Fixes

Table 5.3 outlines the major bugs encountered during testing and their resolutions.

Table 5.3: Notable Bugs and Fixes

Bug Description	**Cause and Resolution**	**Status**
Reservation status not syncing between User and Admin	Reservation field missing in DB model; added and synchronized via API logic	Resolved
Token expiration crashing dashboard	Missing frontend logic; handled with auto-logout and message prompt	Resolved
Corrupt PDF in mobile view	Incorrect `Content-Type`; fixed with proper MIME type headers	Resolved
Dine-in schedule not displaying in POS	Field not passed in order payload; fixed in `placeOrder` API route	Resolved

5.7 Visual Test Reports

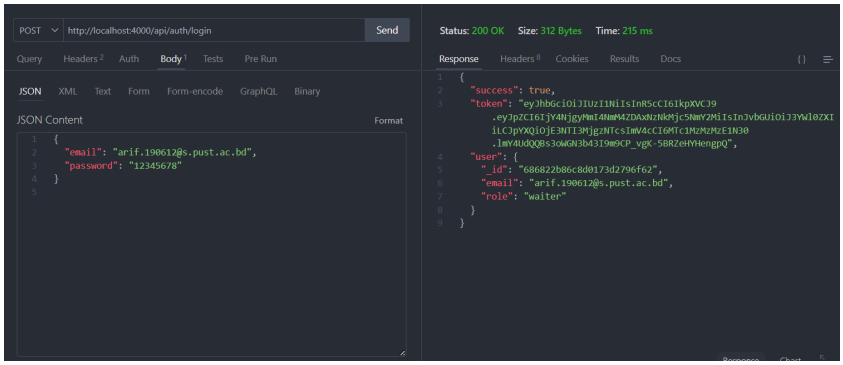

Figure 5.1: Authentication APIs tested using Thunder Client

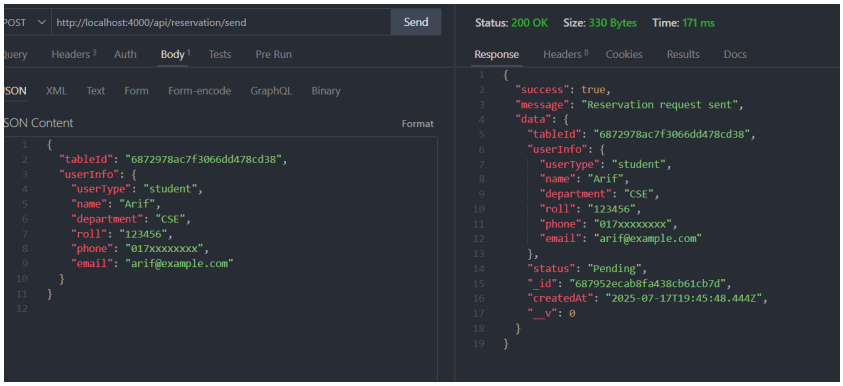

Figure 5.2: Table Reservation APIs tested using Thunder Client

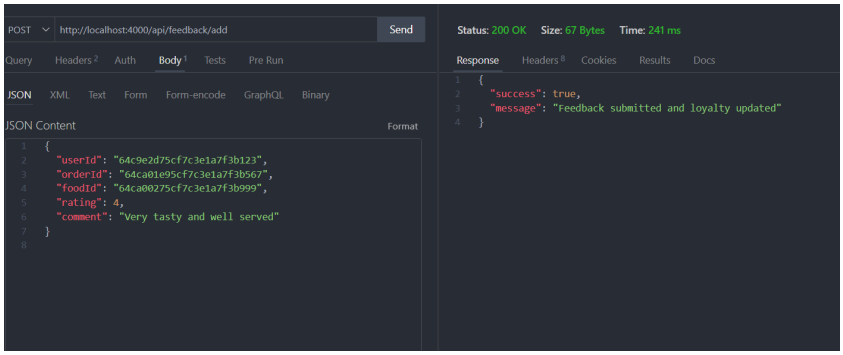

Figure 5.3: User Panel - Feedback and Rating Submission Test

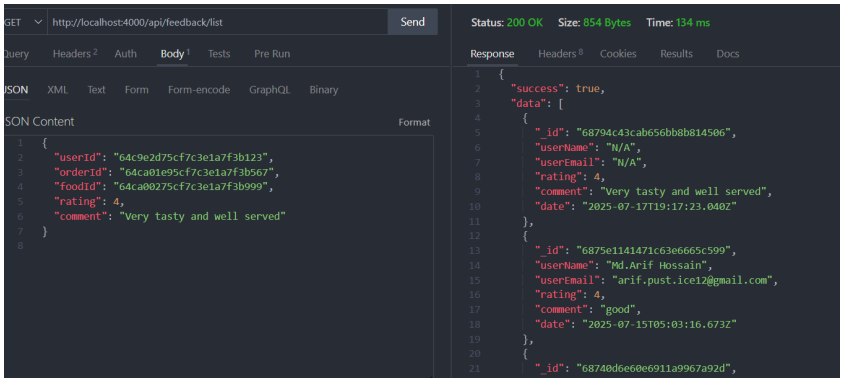

Figure 5.4: User Panel - Feedback Display Validation

5.8 Conclusion

The Cafeteria Management System was subjected to rigorous testing across all layers. Both functional and non-functional aspects were validated using industry-standard tools and real-user feedback. The system meets its design objectives in terms of correctness, performance, usability, and security. Based on the results, the system is considered ready for production deployment.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The Cafeteria Management System has been successfully developed as a modular, scalable, and full-stack digital solution to meet the operational demands of modern restaurants and institutional cafeterias. The system encompasses five distinct panels — User, Admin, POS, Kitchen, and Billing — each tailored to specific stakeholder responsibilities and workflows.

By digitizing traditionally manual tasks, the platform significantly improves staff coordination, service efficiency, and customer satisfaction, effectively demonstrating the value of digital transformation in food service environments.

6.1.1 Key Achievements

The system's core achievements include:

- Seamless food ordering workflow with real-time kitchen communication and order tracking.
- Role-based access control and interface customization aligned with stakeholder responsibilities.
- Digitized table reservation system with admin-mediated approval process.
- Integrated customer feedback and loyalty features to enhance engagement and retention.

- Secure authentication using JSON Web Tokens (JWT) and robust backend API structure.
- Automated invoice generation and billing system for financial transparency and accuracy.

The solution has been implemented using the MERN stack — comprising React.js, Node.js, Express.js, and MongoDB — and adheres to modular design principles, RESTful APIs, and secure development standards. System testing confirms that the platform is functionally stable, secure, and deployable in real-world cafeteria settings.

6.2 System Impact

The development and deployment of this system have shown or are expected to yield the following positive outcomes:

- **Efficiency:** Reduced average service time due to streamlined workflows and real-time communication.
- **Accuracy:** Significantly fewer order and billing errors as a result of digitization.
- **Transparency:** Improved visibility into operations, order progress, and staff activity.
- **Scalability:** Architecture designed to support cloud deployment, future modules, and larger user bases.

6.3 Future Work

While the current implementation fulfills its primary objectives, further improvements can enhance the platform's functionality, scalability, and commercial viability.

6.3.1 Suggested Enhancements

- **Mobile App Integration:** Native or cross-platform mobile applications for users and staff using React Native or Flutter.

- **Push Notification System:** Real-time alerts and updates via Firebase Cloud Messaging (FCM).
- **Payment Gateway Integration:** Secure digital payment options using Stripe, SSLCommerz, or bKash.
- **Multilingual Interface:** Support for multiple languages including English and Bangla for better accessibility.
- **AI-Based Analytics Module:** Intelligent analytics to forecast demand, optimize inventory, and personalize user experience.
- **QR-Based Ordering:** Contactless menu browsing and ordering via tableside QR codes.
- **Cloud Deployment and CI/CD Pipelines:** Hosting on AWS, Vercel, or Heroku with automated deployment through CI/CD.

6.4 Final Remarks

The Cafeteria Management System stands as a technically sound, modular, and future-ready platform suitable for adoption across diverse food service domains. Its clean architecture supports long-term maintainability and offers a solid foundation for integrating next-generation features.

> *"Good software is not just what works today — but what is prepared to scale tomorrow."*

This project not only highlights advanced full-stack development and architectural design skills but also contributes meaningfully to the digitization of cafeteria operations — making it a relevant asset both academically and professionally.

Bibliography

[1] React Documentation, “React – A JavaScript library for building user interfaces,” Meta (Facebook), [Online]. Available: https://reactjs.org. [Accessed: Jul. 2025].

[2] Express.js Documentation, “Fast, unopinionated, minimalist web framework for Node.js,” [Online]. Available: https://expressjs.com. [Accessed: Jul. 2025].

[3] MongoDB Documentation, “The application data platform for modern applications,” [Online]. Available: https://www.mongodb.com. [Accessed: Jul. 2025].

[4] JWT.IO, “JSON Web Tokens - Introduction and Documentation,” [Online]. Available: https://jwt.io. [Accessed: Jul. 2025].

[5] Thunder Client, “Lightweight Rest API Client for VS Code,” [Online]. Available: https://www.thunderclient.com. [Accessed: Jul. 2025].

[6] Vite Documentation, “Next Generation Frontend Tooling,” [Online]. Available: https://vitejs.dev. [Accessed: Jul. 2025].

[7] Node.js Documentation, “Node.js JavaScript runtime,” [Online]. Available: https://nodejs.org. [Accessed: Jul. 2025].

[8] Socket.IO Documentation, “Realtime communication engine,” [Online]. Available: https://socket.io. [Accessed: Jul. 2025].

[9] Vercel Docs, “Frontend Cloud: Develop. Preview. Ship.,” [Online]. Available: https://vercel.com/docs. [Accessed: Jul. 2025].

[10] Figma, “Collaborative interface design tool,” [Online]. Available: https://www.figma.com. [Accessed: Jul. 2025].

[11] Firebase, “Firebase Cloud Messaging for Push Notification,” [Online]. Available: https://firebase.google.com/products/cloud-messaging. [Accessed: Jul. 2025].

[12] RFC 7519, “JSON Web Token (JWT),” IETF, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519. [Accessed: Jul. 2025].