Development of Banglar Heshel: A Comprehensive Restaurant Management System

Tomal Mujumder
ID: 22103038

A Practicum in the Partial Fulfillment of the Requirements

for the Award of Bachelor of Computer Science and Engineering (BCSE)

Department of Computer Science and Engineering
College of Engineering and Technology
IUBAT International University of Business Agriculture and Technology

Fall 2025

Development of Banglar Heshel: A Comprehensive Restaurant Management System

Tomal Mujumder

A Practicum in the Partial Fulfillment of the Requirements for the Award of Bachelor of Computer Science and Engineering (BCSE)
The practicum has been examined and approved,

____
Prof. Dr. Utpal Kanti Das
Chairman

____
Shahinur Alam
Co-supervisor, Coordinator and Assistant Professor

____
Toyeer-E-Ferdoush
Supervisor and Assistant Professor

Department of Computer Science and Engineering
College of Engineering and Technology
IUBAT International University of Business Agriculture and Technology

Fall 2025

Letter of Transmittal

14 January 2026
The Chair
Practicum Defense Committee

Department of Computer Science and Engineering

IUBAT International University of Business Agriculture and Technology
4 Embankment Drive Road, Sector 10, Uttara Model Town
Dhaka 1230, Bangladesh.

Subject: Letter of Transmittal.

Dear Sir,

With due respect, I submit my project report titled **"Development of Banglar Heshel: A Comprehensive Restaurant Management System"**, completed as part of the practicum requirement for the Bachelor of Computer Science and Engineering (BCSE) degree. The project, conducted under the supervision of Soft-Tech Solutions, focuses on developing a full-stack web application that streamlines restaurant operations, online food ordering, table reservations, and inventory management using modern software architecture.

This work provided practical exposure to the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, **Redux state management**, **SSLCommerz payment gateway integration**, **Cloudinary cloud storage**, and **Nodemailer email services**, strengthening my full-stack development, database design, and problem-solving skills. I sincerely appreciate your time in reviewing this report and hope it meets the required academic and professional standards.

Yours sincerely,

____

Tomal Mujumder
ID: 22103038

Organization’s Certificate

Place your organization’s certificate scanned copy here.

Student’s Declaration

I hereby declare that the project report entitled **“Development of Banglar Heshel: A Comprehensive Restaurant Management System”** is my original work and has been prepared by me in partial fulfillment of the requirements for the practicum. All sources of information and references used in this report have been properly acknowledged.

I confirm that the contents of this report are original and that no part of this work has been submitted previously for any degree or academic purpose. I have adhered to ethical guidelines in the collection and presentation of data and take full responsibility for the authenticity of this work.

I sincerely hope that this report meets the standards and expectations of the Practicum Defense Committee and respectfully submit it for review and evaluation.

Tomal Mujumder
ID: 22103038

Supervisor’s Certification

This is to certify that the project report entitled **“Development of Banglar Heshel: A Comprehensive Restaurant Management System”** has been prepared by **Tomal Mujumder**, **ID: 22103038**, in partial fulfillment of the requirements for the practicum.

I have reviewed the report and confirm that the work has been carried out under my supervision. All sources, references, and materials used in this report have been properly cited.

I hereby certify that the report meets the academic standards required by the Department of Computer Science and Engineering, IUBAT, and is suitable for submission and evaluation.

Toyeer-E-Ferdoush

Supervisor and Assistant Professor
Department of Computer Science and Engineering
IUBAT International University of Business Agriculture and Technology

Abstract

This practicum report presents the design and implementation of **"Banglar Heshel,"** a full-stack restaurant management system developed to modernize restaurant operations and customer service delivery. The project addresses critical challenges in traditional restaurant management including manual order processing, inefficient inventory tracking, limited customer engagement, and lack of digital payment integration. Built using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, the system implements **role-based architecture** supporting three user types: Administrators, Managers, and Customers, each with tailored functionalities. Key features include **secure authentication with OTP-based email verification**, **comprehensive menu management** with multi-image uploads via Cloudinary, **shopping cart and wishlist** functionality, **table reservation system** with real-time availability, **SSLCommerz payment gateway integration** with unique token generation for order tracking, **customer review and rating system** (1-5 stars), **inventory management** with automated low-stock alerts, **supplier and purchase order management**, and **analytics dashboards** with sales visualizations using Recharts. The system employs **JWT-based authentication**, **bcrypt password hashing**, **Redux state management**, and **RESTful API architecture** to ensure security and scalability. Development followed the **Incremental Process Model**, allowing iterative refinement of authentication, ordering, payment, and management modules. The application successfully demonstrates how modern web technologies can digitize restaurant operations, reduce processing time, minimize errors, and enhance customer satisfaction through seamless online ordering and reservation experiences. Current limitations include internet dependency and sandbox payment testing. Future enhancements will include **mobile application development**, **real-time order tracking**, **AI-powered menu recommendations**, **offline capabilities**, and **production deployment** with live payment processing. This project showcases the effective application of full-stack development principles to solve real-world challenges in the hospitality industry.

Acknowledgments

I would like to take this opportunity to sincerely thank all those who supported and encouraged me throughout the completion of my practicum program and the preparation of the project report entitled **“Development of Banglar Heshel: A Comprehensive Restaurant Management System.”** First and foremost, I express my gratitude to Almighty for granting me the patience, strength, and perseverance necessary to complete this work. I am also thankful to the International University of Business, Agriculture and Technology (IUBAT) for providing an academic environment that fosters both personal and professional growth. I would like to pay special tribute to **Late Prof. Dr. M. Alimullah Miyan**, the Founder of IUBAT, whose visionary leadership opened new avenues for higher education in the non-government sector of Bangladesh. I also extend my sincere appreciation to the honorable Vice-Chancellor, **Prof. Abdur Rab**, for his continued leadership and guidance, which have greatly benefited the institution and its students. My heartfelt thanks go to **Prof. Dr. Utpal Kanti Das**, Professor and Chair, for his valuable advice and encouragement throughout this academic journey. Above all, I am deeply indebted to my supervisor, **Toyeer-E-Ferdoush**, Assistant Professor, Department of Computer Science and Engineering, IUBAT, for his constant guidance, constructive feedback, and encouragement, which were instrumental in shaping this work. Completing this practicum has enhanced my understanding of the subject matter, strengthened my technical skills, and provided valuable experience in research and report writing. I am truly grateful for the opportunity to work on this project, which has been both academically enriching and professionally rewarding.

Table of Contents

List of Figures

List of Tables

Chapter 1.
Introduction

The contemporary food service industry is characterized by a paradigm shift towards digital transformation and on-demand convenience, fundamentally altering consumer behavior and restaurant operational methodologies. In an era where efficiency and seamless digital experiences are paramount, the traditional methods of restaurant management relying on manual order processing, paper-based reservation systems, physical inventory ledgers, and cash-dominant transactions have become increasingly inefficient and obsolete. "Banglar Heshel - A Comprehensive Restaurant Management System" represents a strategic response to this gap, engineered as a full-stack web-based platform during academic practicum. This system is designed to seamlessly connect customers seeking authentic Bengali cuisine and convenient dining experiences with comprehensive restaurant services including online ordering, table reservations, and real-time inventory management, all integrated within their immediate digital accessibility.

Developed leveraging the robust capabilities of the MERN stack (MongoDB, Express.js, React.js, Node.js) and adhering to the rigorous principles of modular architecture, Banglar Heshel integrates secure payment processing via SSLCommerz and harnesses cloud storage through Cloudinary and automated email services via Nodemailer to optimize operational workflows. The platform facilitates the entire restaurant service lifecycle, encompassing menu discovery, cart management, booking, secure payment processing with unique token generation, and review management. By digitizing traditional restaurant operations and providing a structured, secure framework for food service transactions, Banglar Heshel aims to empower local food businesses, reduce friction in service delivery, enhance inventory control, and provide unparalleled convenience to urban food enthusiasts. This report exhaustively details the project's inception, architectural design, implementation challenges, database schema, security implementations, and strategic implications for the digital restaurant management landscape.

1.1 Background of Study

The global landscape for restaurant technology and food service automation is currently undergoing an unprecedented expansion, catalyzed by rapid digital transformation, the ubiquitous proliferation of online ordering platforms, and a profound shift in consumer preference toward contactless, technology-driven dining experiences. Market analysis indicates that the global restaurant management software market size was valued at approximately USD 4.59 billion in 2023 and is projected to exceed USD 10.02 billion by 2030, growing at a compound annual growth rate (CAGR) of 11.8% (Grand View Research, 2023). This exponential growth is fueled by the increasing digitization of food ordering, delivery services, and inventory management systems, transforming how modern restaurants operate and interact with their customers.

Despite the dominance of major players in specific verticals like food delivery or point-of-sale systems, the comprehensive restaurant management ecosystem remains highly fragmented and operationally inefficient. Trust and quality assurance remain significant barriers; customers often struggle to find authentic cuisine with verified reviews and reliable service, while restaurant owners lack integrated digital infrastructure to manage inventory, reservations, payments, and customer relationships effectively within a single unified platform (McKinsey & Company, 2023). Traditional restaurant operations rely on disconnected systems for ordering, reservation management, and inventory tracking, while existing platforms often charge substantial commission fees that reduce profit margins for small and medium-sized establishments. Furthermore, the integration of data analytics and automated workflow management represents a shift toward "intelligent" restaurant operations, a key trend identified for 2025 and beyond, where real-time insights and personalized customer experiences become the competitive norm (Deloitte, 2024). Banglar Heshel addresses these gaps by offering a unified, comprehensive management platform that prioritizes operational efficiency and customer satisfaction. Unlike generic point-of-sale systems, it employs integrated role-based architecture to connect menu management, inventory control, payment processing, reservation workflows, and analytics dashboards within a cohesive ecosystem, reducing operational overhead and enhancing service quality for both restaurant staff and customers.

1.2 Methodology

The development of Banglar Heshel followed a rigorous, structured methodology that combined agile software development practices with user-centered design principles and iterative testing workflows. This approach ensured that the final product was not only technically robust and scalable but also aligned precisely with the operational realities and needs of restaurant owners, management staff, and customers. The methodology was divided into distinct phases of requirements gathering, system design, incremental development, testing, and deployment validation.

1.2.1 Primary Sources

Primary data collection was instrumental in shaping the functional requirements and user experience design of the platform. This phase involved direct engagement with the target demographic to understand their immediate pain points and operational necessities.

- Primary data collection was instrumental in shaping the functional requirements and user experience design of the platform. This phase involved direct engagement with the target demographic to understand their immediate pain points and operational necessities.

- Direct Observation of Restaurant Operations: We conducted field observations of current offline methods for restaurant management, noting the friction points such as manual order taking leading to errors, inefficient inventory tracking causing stock shortages or wastage, paper-based reservation systems resulting in double bookings, and lack of integrated payment processing creating accounting discrepancies. This highlighted the necessity for a unified digital platform with automated workflows and real-time data synchronization.

- Stakeholder Interviews: Structured interviews were conducted with potential users, ranging from restaurant owners and managers to front-line staff and regular customers, to understand their technological literacy, workflow preferences, and pain points. These insights directly influenced the design of the role-based dashboard system, the decision to implement intuitive menu management with drag-and-drop image uploads, and the simplified reservation workflow with email confirmations.

- User Feedback Loops: During the development phase, iterative prototypes were tested with a focus group of restaurant staff and customers to refine the UI/UX. Particular attention was paid to the "Order Placement" flow, shopping cart functionality, and mobile responsiveness of the platform, ensuring that the payment process was seamless and the table reservation system was intuitive and non-intrusive (Nielsen Norman Group, 2023).

1.2.2 Secondary Sources

Secondary research focused on establishing technical feasibility, adhering to architectural best practices, and understanding broader market trends in restaurant technology.

- **Technical Documentation:** An extensive review of MongoDB database design patterns, Express.js middleware architecture, React.js component lifecycle documentation, Node.js server-side programming guides, SSLCommerz payment gateway integration documentation, Cloudinary API specifications, and Nodemailer SMTP configuration provided the technical backbone for the system's architecture. This ensured that the chosen technologies were compatible and capable of meeting performance benchmarks (MongoDB Documentation, 2024; React Documentation, 2024).
- **Market Research Reports:** Reports from Grand View Research and Deloitte provided critical data on restaurant management software market size, growth trajectories (CAGR of 11.8%), and emerging trends like cloud-based operations, mobile ordering integration, and data analytics adoption. This data validated the business case for the project and informed the feature prioritization strategy, emphasizing inventory management, payment integration, and analytics dashboards (Grand View Research, 2023; Deloitte, 2024).
- **Academic Journals on Web Application Security:** Research papers regarding authentication best practices, specifically JWT token security, bcrypt password hashing, and OTP-based verification systems, informed the design of the security modules. This ensured that the application would remain secure against common vulnerabilities like SQL injection, cross-site scripting (XSS), and unauthorized access, critical factors in maintaining user trust and data integrity (OWASP, 2024; IEEE Software Engineering, 2023).

1.3 Objectives

The project was driven by a strategic vision to create a scalable, secure, and user-focused platform that addresses the specific challenges of modern restaurant management and digital food service delivery.

1.3.1 Broad Objective

The broad objective of this practicum is to design, develop, and deploy "Banglar Heshel," a comprehensive restaurant management system that streamlines restaurant operations, online food ordering, table reservations, inventory control, and customer engagement through integrated digital workflows, secure payment processing, and data-driven analytics. The goal is to enhance operational efficiency, improve customer satisfaction, and stimulate business growth for local food establishments by removing the technological barriers that currently exist in traditional restaurant management processes.

1.3.2 Specific Objectives

- **Implement a Modular MERN Stack Architecture:** To build a maintainable and scalable full-stack application using MongoDB for database management, Express.js for backend API development, React.js for frontend user interface, and Node.js for server-side operations, ensuring a clear separation of concerns between data layer, business logic, and presentation components. This architecture facilitates future feature expansion and ease of maintenance (MongoDB Architecture Guide, 2024).
- **Integrate Secure Payment Gateway Services:** To utilize SSLCommerz payment gateway APIs for secure online transaction processing, enabling customers to complete orders with credit/debit cards, mobile banking, and digital wallets. This objective ensures trust and convenience, allowing users to pay online and receive unique token numbers for order tracking and pickup verification (SSLCommerz Documentation, 2024).
- **Develop Multi-Role Access Control System:** To implement role-based authentication and authorization mechanisms supporting three distinct user types: Administrators with complete system oversight, Managers with operational control over menu, inventory, payments, and reservations, and Customers with ordering and reservation capabilities. This enhances security by restricting access based on user roles and responsibilities (JWT Best Practices, 2024).
- **Ensure Data Security and Privacy:** To implement robust authentication mechanisms utilizing JWT tokens, bcrypt password hashing, and OTP-based email verification through Nodemailer integration. This includes protecting sensitive user data, payment information, and personal identifiable information (PII) in strict compliance with security best practices and data protection standards (OWASP Security Guidelines, 2024).
- **Create a Responsive User Interface:** To design a mobile-first frontend using React.js and Tailwind CSS that allows seamless browsing, ordering, and reservation management across various devices. Recognizing the dominance of smartphone usage

in the target market, the interface must be touch-friendly, visually appealing, and responsive to varying screen sizes from mobile phones to desktop computers (Nielsen Norman Group, 2023).

- **Implement Comprehensive Inventory Management:** To develop automated stock tracking functionality with real-time quantity updates, minimum and maximum threshold configurations, and low-stock alert notifications. This objective ensures efficient inventory control, reduces wastage, prevents stock-outs, and facilitates data-driven purchasing decisions (Restaurant Inventory Management Best Practices, 2024).
- **Develop Analytics and Reporting Capabilities:** To integrate data visualization dashboards using Recharts library, providing restaurant management with actionable insights on sales trends, popular menu items, revenue patterns, payment statistics, and customer engagement metrics. This enables informed business decisions and strategic planning (Business Intelligence for Restaurants, 2024).

1.4 Process Model

The Incremental Process Model was selected as the optimal development methodology for Banglar Heshel. This model was chosen because the project required the integration of distinct, complex modules—User Authentication, Menu Management, Shopping Cart, Payment Processing, Table Reservations, Inventory Control, and Analytics Dashboards—that could be developed, tested, and refined independently before integration into the final system.

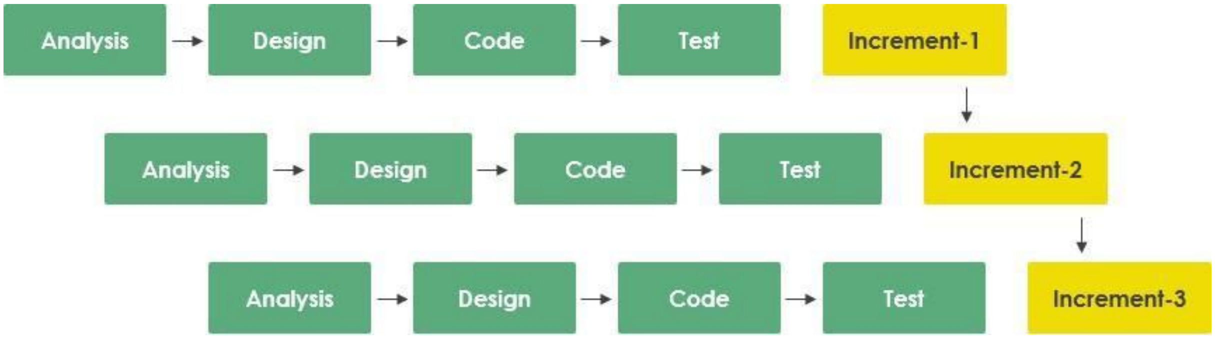


Figure 1.1: Incremental Process Model

1.4.1 Reasons for Choosing the Incremental Process Model.

The rationale for selecting the Incremental Process Model over other methodologies like Waterfall or pure Agile Scrum includes several key factors relevant to the practicum environment:

- **Risk Mitigation**: By developing the core authentication and menu management modules first, we could validate the primary value proposition and technical architecture before investing time in secondary, high-risk features like payment gateway integration and analytics. This reduced the likelihood of project failure due to technical roadblocks in advanced features.

- **Flexibility**: The incremental approach allowed for the incorporation of feedback from early testing phases (e.g., refining the shopping cart workflow and reservation form based on initial user data) into subsequent iterations without derailing the entire project schedule.
- **Parallel Development**: The separation of the project into distinct increments (e.g., Increment 1: Authentication & Profiles; Increment 2: Menu Management & Browsing; Increment 3: Cart & Ordering; Increment 4: Payment & Reservations; Increment 5: Inventory & Analytics) allowed for focused development sprints, ensuring that each component was robust before moving to the next.
- **Early Delivery of Functionality**: A functional version of the product was available early in the lifecycle, facilitating early testing of critical paths like the "Order Food" and "Reserve Table" workflows. This allowed for continuous validation against user requirements and stakeholder expectations.
- **Independent Module Testing**: Each increment could be tested independently, allowing bugs to be identified and resolved in isolation before system integration, resulting in a more stable final product with fewer integration issues.

1.5 Feasibility Study

A comprehensive feasibility study was conducted to ensure the project's viability across technical, economic, and operational dimensions. This analysis confirmed that the project was not only achievable within the constraints of the practicum but also held significant potential for real-world application in the restaurant management sector.

1.5.1 Technical Feasibility.

The project is technically feasible, leveraging the mature and robust MERN stack ecosystem. MongoDB provides flexible, document-based data storage capable of handling complex nested data structures for orders, reservations, and inventory records with excellent scalability for growing restaurant operations. Express.js offers a minimalist, unopinionated framework for building RESTful APIs with robust middleware support for authentication, validation, and error handling. React.js ensures a dynamic, component-based user interface with efficient state management through Redux, enabling seamless user interactions and real-time updates. Node.js provides an event-driven, non-blocking I/O model ideal for handling concurrent connections from multiple customers and restaurant staff simultaneously (Node.js Documentation, 2024). Integration with SSLCommerz relies on well-documented RESTful APIs with comprehensive implementation guides, making secure payment processing technically straightforward and reliable for the Bangladeshi market (SSLCommerz Integration Guide, 2024). Furthermore, Cloudinary provides serverless cloud storage with automatic image optimization and CDN delivery, removing the need for local file storage infrastructure. This makes image management technically viable on standard hosting environments without requiring specialized hardware (Cloudinary Documentation, 2024). The integration of Nodemailer with Gmail SMTP ensures reliable email delivery for OTP verification, order confirmations, and reservation notifications using industry-standard protocols (Nodemailer Documentation, 2024).

1.5.2 Economic Feasibility.

The project demonstrates strong economic viability. The development phase utilized open-source technologies (MongoDB, Express.js, React.js, Node.js) and free-tier services for cloud storage (Cloudinary free tier), email services (Gmail SMTP), and initial deployment options (Vercel, Render, MongoDB Atlas free tier), significantly minimizing upfront development costs. The potential operational cost is low, primarily consisting of cloud hosting fees (e.g., MongoDB Atlas, AWS, or DigitalOcean) which scale with usage, and SSLCommerz transaction fees which are only incurred on successful payments. Given the market projection of the restaurant management software sector reaching over USD 10 billion by 2030 with a CAGR of 11.8% (Grand View Research, 2023), the potential return on investment (ROI) through subscription models for restaurant owners, transaction-based revenue sharing, or premium feature monetization is substantial. The low barrier to entry and high scalability of the platform suggest a sustainable business model for both individual restaurants and multi-location chains..

1.5.3 Operational Feasibility.

Operationally, the system is designed to be self-sustaining with minimal manual intervention required for day-to-day activities. Automated workflows for order notifications, email confirmations, payment receipts, reservation confirmations, and low-stock alerts reduce the administrative burden on restaurant staff. The platform's user interface is designed for intuitive use with clear navigation, familiar e-commerce patterns, and responsive design, minimizing the need for extensive user training or technical support. The modular MERN architecture with separated concerns between frontend components, backend API routes, and database models ensures that the system is maintainable and adaptable to future changes in business requirements, technology updates, or feature additions, ensuring long-term operational stability (React Architecture Best Practices, 2024). The role-based access control system allows restaurants to onboard new staff members with appropriate permissions without system-wide configuration changes, making operational scaling straightforward and secure.

1.6 Structure of the Report

The report is structured systematically into ten chapters to provide a comprehensive overview of the project. Following this Introduction, **Chapter 2** provides an in-depth overview of Soft-Tech Solutions, the host organization. **Chapter 3** details Requirement Engineering, outlining user and system needs. **Chapter 4** covers Analysis, presenting detailed architectural diagrams including Activity, Swim-lane, and Class diagrams. **Chapter 5** discusses Project Management and Risk mitigation strategies. **Chapter 6** outlines Project Planning and

Estimation metrics. **Chapter 7** details System Design, including Interface, Data Flow, and Database design. **Chapter 8** covers System Quality and Testing methodologies. **Chapter 9** addresses Ethical Considerations in software development. Finally, **Chapter 10** concludes the report with a summary of achievements and future work.

Chapter 2.

Organizational Overview

Soft-Tech Solution Limited is a software development and logistics technology company. It provides Enterprise Resource Planning (ERP) solutions, custom software development, cloud computing, and mobile application development. The company has a particular focus on logistics and courier management systems. Its work combines practical problem-solving with innovative technology. Soft-Tech Solution is based in Dhaka and is gradually expanding its services beyond the city. The team consists of experienced professionals and young developers, creating a balance of industry knowledge and fresh technical skills.

2.1 Organization Vision

The vision of Soft-Tech Solution Limited is to establish itself as a recognized technology provider that supports business growth through reliable and scalable software. The organization seeks to contribute to the digital transformation of industries by offering practical and cost-effective solutions that simplify operations and improve efficiency.

2.2 Organization Mission

The mission of Soft-Tech Solution Limited is to design and deliver software solutions that are tailor to client needs. The company emphasizes high-quality development, professional guidance, and long-term client support. In addition to serving organizations, Soft-Tech Solution also focuses on developing new talent by providing interns and junior developers with firsthand learning opportunities. Through teamwork, continuous improvement, and a focus on usability, the company aims to contribute positively to the local technology community.

2.3 Organization Services

Soft-Tech Solution Limited provides a range of services, including:

1. **Custom Software Development:** Development of software solutions designed for specific business requirements across industries.
2. **Enterprise Resource Planning (ERP) Solutions:** Implementation of ERP systems to integrate operations, increase efficiency, and support decision-making.
3. **Cloud Computing Services:** Cloud migration, secure storage

4. **E-commerce Platform Development:** Creation of e-commerce websites and mobile applications that support online transactions and business growth.
5. **Product Development and Software Integration:** Assistance in building new digital products and integrating them with existing systems.
6. **Software Maintenance and Testing:** Ongoing updates, performance monitoring, and security testing to ensure system reliability.
7. **Training:** Training programs for client teams to support the effective use and management of new systems.
8. **Mobile App Development:** Design and development of mobile applications for Android and iOS platforms.

2.4 Organizational Structure

Soft-Tech Solution Limited is a small but growing company. As a startup, it follows a simple organizational structure, as shown in Figure 2.1.

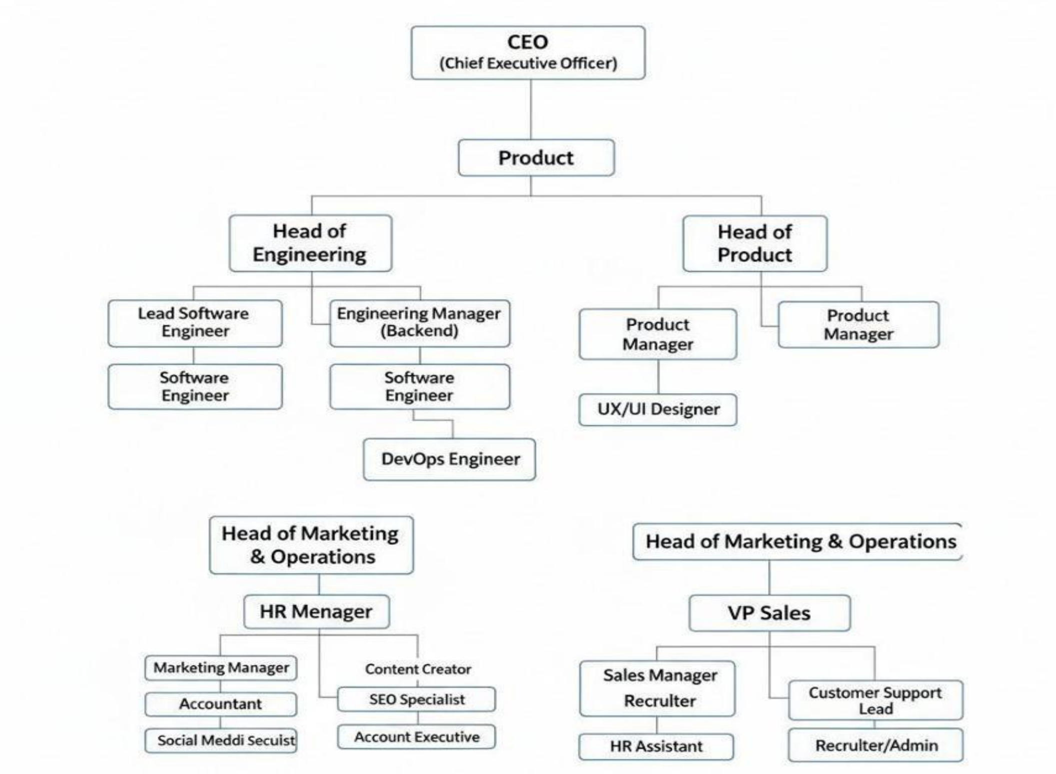


Figure 2.1 Organizational Structure

2.5 My Position in this Organization

I served as a **Junior Front-End Developer Intern** within the Engineering Department. My responsibilities were integral to the development of the **Banglar Heshel** platform. I was tasked with assisting in full-stack development, specifically focusing on React.js component development for the user interface, Express.js RESTful API architecture for backend services, integrating third-party services such as SSLCommerz payment gateway, Cloudinary cloud storage for image uploads, and Nodemailer for automated email notifications with OTP verification, and designing the NoSQL database schema using MongoDB with Mongoose for data modeling. I worked under different mentorship actively participating in daily stand-ups, code reviews, and sprint planning sessions, which provided invaluable exposure to professional software development workflows and agile methodologies.[15]

2.6 Address of the Organization

Soft-Tech Solution

Amin Tower, North side of Mascot Plaza, Sector 9, Road 1/A
Uttara, Dhaka 1230, Bangladesh.15

Chapter 3.

Requirement Engineering

Requirement engineering was a pivotal phase in defining the scope, functionality, and user experience of Banglar Heshel. We employed a systematic and iterative approach, engaging with various stakeholders including restaurant owners, management staff, and potential customers to elicit, analyze, and validate requirements, ensuring the final product effectively solved the identified operational challenges in the food service industry.

3.1 Requirement Analysis

The analysis phase focused on identifying the core problem: the lack of an integrated, efficient, and user-friendly digital platform for comprehensive restaurant management that addresses both operational needs and customer experience. We conducted a comparative analysis of existing restaurant management solutions and online ordering platforms (e.g., Toast POS, Square for Restaurants, Foodpanda, Pathao Food) and identified significant gaps in their integration capabilities, cost-effectiveness for small to medium-sized restaurants, and comprehensive feature sets combining ordering, reservations, inventory, and analytics in a single platform. The analysis confirmed the critical need for a system that prioritizes operational efficiency—to reduce manual workload and human errors—and customer satisfaction—to ensure seamless ordering and reservation experiences. The "comprehensive management" nature of the solution meant the system had to handle multiple interconnected workflows efficiently including menu management, real-time inventory tracking, secure payment processing, table reservation coordination, customer reviews, supplier management, and provide real-time business analytics to restaurant owners and managers.[2]

3.2 Requirement Engineering

The requirements were categorized into User, System, Functional, and Non-Functional requirements to ensure a holistic definition of the system.

3.2.1 User and System Requirements

This section maps high-level user needs to specific, testable system requirements.

1. Authentication and Account Management: Users, Managers, and Administrators require a secure and distinct access mechanism to the platform.

1.1 The system shall provide a multi-role registration system, allowing actors to register as Customers with email verification via OTP.

1.2 The system shall enforce a comprehensive registration flow, including username, email, password validation, and 6-digit OTP verification sent via Nodemailer.

1.3 The system requires email confirmation through OTP before account activation to verify user identity, with OTP expiry set to 5 minutes.

1.4 The system shall support secure login using email and password with JWT token-based authentication and bcrypt password hashing.

1.5 The system shall allow users to reset their passwords via a secure OTP sent to their registered email address.

1.6 The system shall enforce role-based access control (RBAC) covering Admin, Manager, and User roles with distinct dashboard interfaces and permissions.

1.7 The system shall support Google OAuth integration via Firebase for seamless third-party authentication.

2. Profile and Menu Management: Users need to browse menus and manage their profiles, while restaurant staff need to manage food offerings.

2.1 The system allow users to update their personal profile information, including username, email, and profile picture via Cloudinary upload.

2.2 Managers shall be able to create, read, update, and delete food items with comprehensive details including food ID, name, description, category, price, old price, discount percentage, and multiple images (up to 6).

2.3 The system should integrate Cloudinary API for efficient image storage, automatic optimization, and CDN delivery.

2.4 The system shall provide image management with automatic cleanup of removed images from cloud storage.

2.5 The system shall support dynamic food category management for organized menu browsing.

3. Food Discovery and Search: Users must be able to find and explore menu items efficiently.

3.1 The system shall allow users to browse food items by dynamically managed categories (e.g., Main Course, Appetizers, Desserts, Beverages).

3.2 The system shall provide a search bar allowing keyword-based queries for food items by name.

3.3 The system shall sort search results based on relevance, creation date (ascending/descending), and price.

3.4 The system shall display food cards with key information, including Food Name, Price, Discount Badge, Category, Rating, and Images.

3.5 The system shall implement loading skeletons and optimized data fetching to handle large menu catalogs without performance degradation.

4. **Shopping Cart and Wishlist:** Users need to manage their food selections before ordering.

4.1 The system shall enable users to add food items to a shopping cart with quantity selection.

4.2 The system shall implement cart persistence using local storage, maintaining cart state across sessions.

4.3 The system shall display real-time cart item count in the navigation bar.

4.4 The system shall allow users to update item quantities or remove items from the cart.

4.5 The system shall calculate and display real-time cart totals including subtotal and applicable discounts.

4.6 The system should provide Wishlist functionality allowing users to save favorite food items for future reference.

4.7 The system display Wishlist item count in the navigation bar with heart icon indicator.

5. **Table Reservations:** Users need to book tables for dining experiences.

5.1 The system should provide a reservation form collecting customer name, email, phone number, date, time, and party size.

5.2 The system shall validate reservation inputs including email format, phone number, future date selection, and party size limits.

5.3 The system shall store reservation requests with status tracking (Pending, Confirmed, Cancelled).

5.4 The system shall allow Managers to view, search, and manage all reservations with filtering capabilities.

5.5 The system shall generate daily reservation reports in PDF format with customer details and statistics.

6. **Payments and Transactions:** Users need a secure method to pay for orders, and restaurant staff need to track revenue.

6.1 The system shall integrate SSLCommerz payment gateway for handling secure online payments supporting credit/debit cards, mobile banking, and digital wallets.

6.2 The system shall generate unique token numbers for each successful payment transaction for order tracking and pickup verification.

6.3 The system shall generate automated PDF receipts with jsPDF library for completed transactions including itemized order details, payment method, transaction ID, and token number.

6.4 The system shall maintain a comprehensive payment history accessible to customers showing all past transactions with timestamps.

6.5 The system shall provide Managers with payment management dashboard displaying all transactions with search, filter, and verification capabilities.

6.6 The system shall ensure transactional integrity by storing complete cart data, user information, and payment details in the database upon successful payment confirmation.

7.Reviews and Ratings: The platform requires mechanisms to ensure quality feedback and trust.

7.1 The system shall allow authenticated users to submit reviews for food items with 1–5-star ratings and text comments.

7.2 The system shall display all reviews for each food item with reviewer username, rating, comment, and timestamp.

7.3 The system shall calculate and display the average rating for each food item based on all submitted reviews.

7.4 The system shall allow users to delete their own reviews and allow Managers to delete any review for moderation purposes.

7.5 The system shall implement review ownership validation to prevent unauthorized deletions.

8. Inventory and Stock Management: Restaurant staff need to track food item availability and supplies.

8.1 The system shall automatically create stock entries for each new food item with configurable quantity, unit type (pieces, kg, liters), minimum threshold, and maximum threshold.

8.2 The system shall allow Managers to update stock quantities in real-time through the stock management dashboard.

8.3 The system shall display low-stock warnings when quantities fall below minimum thresholds.

8.4 The system shall provide stock level visibility to prevent orders for out-of-stock items.

9. Supplier Management: Restaurant operations require tracking supplier relationships and purchase orders.

9.1 The system shall allow Managers to add, edit, and delete supplier information including company name, contact person, email, phone, address, and rating.
9.2 The system shall provide search functionality for suppliers by name, contact person, or email.
9.3 The system shall enable creation of purchase orders linked to specific suppliers with product details, quantities, unit prices, and total amounts.
9.4 The system shall track purchase order status (Pending, Completed, Cancelled) with timestamps.
9.5 The system shall allow Managers to rate suppliers (1-5 stars) for performance evaluation.

10. Analytics and Reporting: Management needs data-driven insights for business decisions.
10.1 The system shall provide analytics dashboard with visual charts using Recharts library displaying sales trends, revenue patterns, and popular items.
10.2 The system shall calculate and display key performance indicators including total users, total food items, total reservations, and revenue metrics.
10.3 The system shall generate comprehensive reports for reservations, payments, and inventory in PDF format.
10.4 The system shall provide real-time statistics on the admin dashboard for quick business overview.

11. Employee and Manager Management: Administrators need to manage staff access and roles.
11.1 The system shall allow Administrators to add new Managers and employees with role assignments and profile information.
11.2 The system shall provide employee search functionality by ID, name, email, or NIC number.
11.3 The system shall display employee lists with filtering by role and sorting capabilities.
11.4 The system shall allow Administrators to delete or deactivate employee accounts.
11.5 The system shall generate employee list reports in PDF format with complete details.

12. Email Notifications: Users and staff need automated email communications.
12.1 The system should integrate Nodemailer with Gmail SMTP for reliable email delivery.
12.2 The system should send professional HTML-formatted emails for OTP verification during registration with 6-digit code and 5-minute expiry notice.
12.3 The system shall send password reset OTP emails with security warnings and expiry information.
12.4 The system should send order confirmation emails to customers with order details and token number.
12.5 The system shall send reservation confirmation emails to customers with booking details and restaurant contact information.

3.2.2 Functional Requirements

The core functionalities of the Banglar Heshel system include several key operational areas that enable end-to-end restaurant management and food service delivery on the platform.

1. 1. User Authentication and Identity: The system provides secure registration, login, and logout functionality for all actors. It supports email verification with 6-digit OTP and password recovery mechanisms to ensure account security. Role management is enforced to maintain strict separation of privileges among Users, Managers, and Administrators. Google OAuth integration provides alternative authentication for enhanced user convenience.
2. 2. Profile Management: Users are able to update their personal details and profile pictures via Cloudinary cloud storage. Managers can manage comprehensive food item profiles including multiple images, detailed descriptions, pricing structures, and category assignments. Administrators have visibility into all user profiles and their corresponding access permissions.
3. 3. Menu and Food Management: Managers can create, read, update, and delete food items with complete information including names, descriptions, categories, pricing, discounts, and multiple images. Each food item includes comprehensive details such as food ID for tracking, base price and optional old price for discount display, percentage-based discount calculations, and support for up to six high-quality images stored via Cloudinary with automatic optimization and CDN delivery.
4. 4. Shopping Cart and Wishlist Management: The system supports a complete shopping experience with persistent cart functionality across sessions. Users can add items to cart with quantity selection, update quantities, remove items, and view real-time totals. The Wishlist feature allows users to save favorite items for future orders, with both cart and Wishlist counts displayed prominently in the navigation bar.
5. 5. Table Reservation System: The platform provides comprehensive table booking functionality allowing customers to reserve tables by specifying date, time, party size, and contact information. Managers can view, search, filter, and update reservation statuses (Pending, Confirmed, Cancelled), and generate daily PDF reports with complete reservation statistics and customer details.
6. 6. Payment Processing: The system integrates with the SSLCommerz payment gateway to handle transactions securely. It supports both sandbox testing and production environments, generates unique token numbers for order tracking and pickup verification, creates verifiable digital receipts in PDF format for completed payments, and maintains comprehensive transaction history accessible to both customers and management staff.
7. 7. Review and Rating System: The platform enables customers to submit reviews and ratings (1-5 stars) for food items after purchase, view all reviews with timestamps and user information, and delete their own reviews. Managers have moderation capabilities

to remove inappropriate reviews, and the system automatically calculates and displays average ratings for each food item to assist customer decision-making.

8. 8. Inventory and Stock Management: The system provides real-time inventory tracking with automated stock entry creation for new food items, configurable minimum and maximum thresholds, low-stock alert notifications, and manager-accessible stock update interfaces to prevent stockouts and reduce wastage.
9. 9. Supplier and Purchase Order Management: Restaurant operations are supported through supplier relationship management including supplier contact information storage, supplier rating system, purchase order creation and tracking, and status management (Pending, Completed, Cancelled) for procurement workflows.
10. 10. Analytics and Business Intelligence: Management requires data-driven insights through comprehensive analytics dashboards displaying sales trends, revenue patterns, popular menu items, payment statistics, reservation analytics, and inventory levels using interactive charts and visualizations powered by Recharts library..

3.2.3 Non-Functional Requirements

Non-functional requirements define system quality and performance standards.

1. Security: The system shall ensure secure handling of user data by hashing passwords using bcrypt with salt rounds, enforcing HTTPS for all communications, and implementing JWT-based authentication with secure token storage. Role-based authorization shall restrict access to system resources based on user roles (Admin, Manager, User), and all inputs shall be validated on both client and server sides to prevent common security attacks including SQL injection, cross-site scripting (XSS), and unauthorized access.
2. Performance: The system shall deliver responsive performance with food search results returned within 2 seconds under normal load conditions and main pages loading within 3 seconds on standard broadband connections. Time-consuming operations such as email delivery, PDF generation, and image uploads shall be processed asynchronously to prevent UI blocking. The system shall handle at least 100 concurrent users without significant performance degradation.
3. Scalability: The system shall be designed to support future growth by following modular MERN architecture principles with separated concerns between frontend components, backend API routes, and database models. The MongoDB database shall use appropriate indexing on frequently queried fields (userId, foodId, email) to maintain query performance as data volume grows. The system shall support horizontal scaling through stateless API design and cloud-based infrastructure deployment.
4. Reliability and Availability: The platform shall maintain high availability during peak dining hours (lunch and dinner periods) and ensure reliable operation through proper error handling, logging, and graceful degradation. All critical transactions including

payments, orders, and reservations shall follow ACID principles in MongoDB transactions to maintain data consistency and prevent data loss.

5. Usability: The user interface shall be fully responsive across all devices (mobile phones, tablets, desktops) and follow modern UX design principles with intuitive navigation. The system shall provide clear and immediate feedback for user actions including success notifications, error messages, and loading indicators to enhance usability. The ordering and reservation workflows shall be streamlined to minimize steps required for task completion.
6. Maintainability: The system shall follow modular MERN stack architecture with clear separation of concerns between frontend components, backend API routes, database models, and service layers. Proper code documentation, meaningful variable naming, and consistent coding standards shall be maintained to support future development and maintenance. Version control through Git and GitHub shall ensure code history and collaborative development capabilities.
7. Performance: The system shall deliver responsive performance with API response times under 2 seconds for standard operations and page load times under 3 seconds on standard broadband connections. Image loading shall be optimized through Cloudinary CDN and lazy loading techniques. Database queries shall be optimized with appropriate indexing on frequently accessed fields.
8. Scalability: The system shall be designed to support future growth through horizontal scaling capabilities of MongoDB and Node.js, utilizing cloud infrastructure services (MongoDB Atlas, AWS, or Digital Ocean) that can scale with increasing user load and data volume.

3.3 Use Case Diagram of the System

The Use Case diagram visualizes the high-level interactions between the actors (Customer, Manager, Admin) and the system, defining the scope of the application.

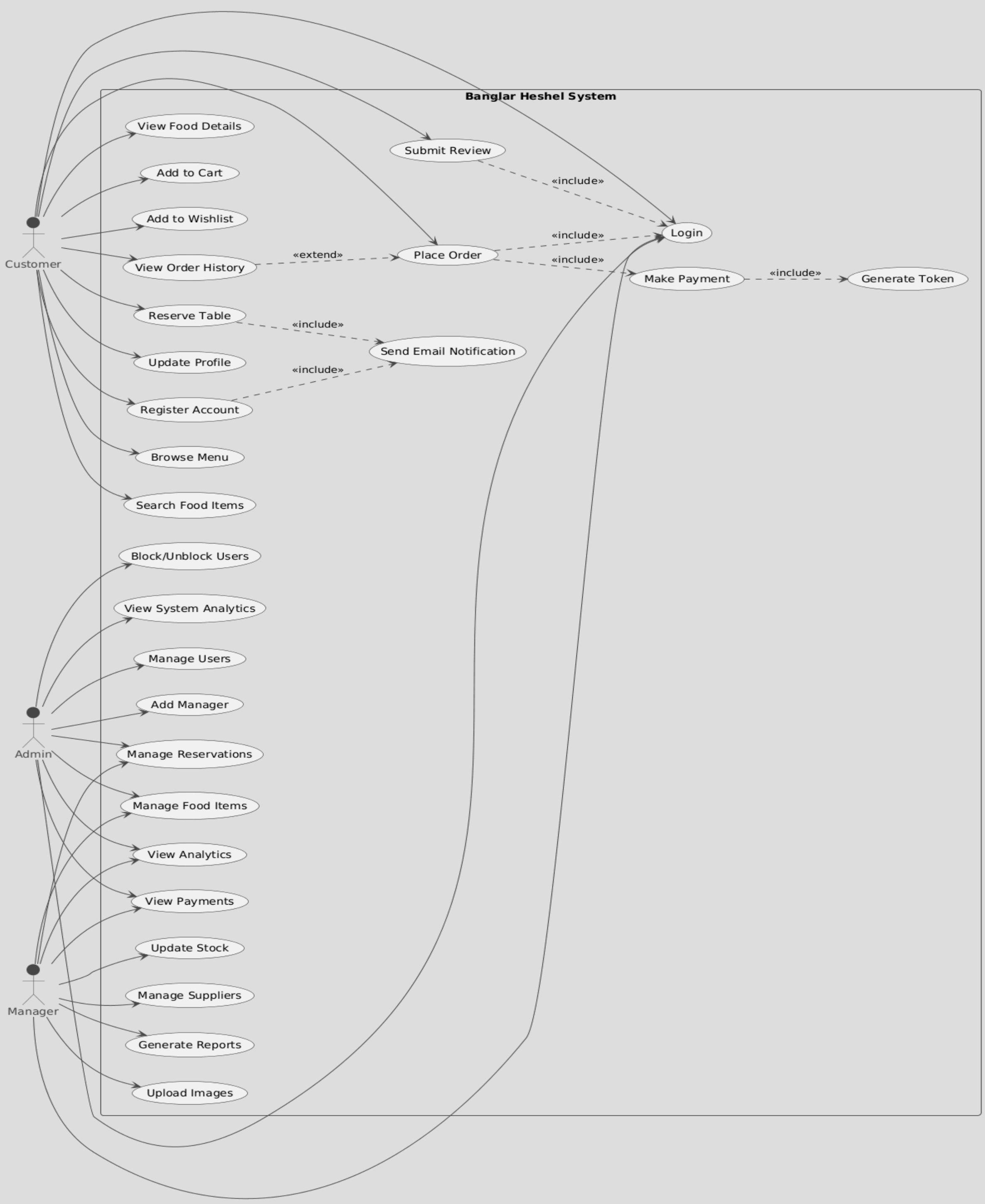


Figure 3.1: Use Case Diagram

Actors: Customer (User), Manager, Administrator.

Key Use Cases:

Customer: The primary actor representing end-users can perform the following actions: Register Account, Login, Browse Menu, Search Food Items, View Food Details, Add to Cart, Add to Wishlist, Place Order, Make Payment, Reserve Table, Submit Review and Rating, View Order History, and Update Profile. These use cases enable customers to explore the menu, place orders, make reservations, and provide feedback on their dining experience.

Manager: The operational staff member responsible for day-to-day restaurant management can perform: Login, Manage Food Items including Add, Edit, and Delete operations, Upload Food Images via Cloudinary integration, View Payments and transaction history, Manage Reservations with status updates, Update Stock Levels with inventory tracking, Manage Suppliers and supplier relationships, Create Purchase Orders for supply chain management, View Analytics Dashboard for business insights, and Generate Reports for various operational metrics.

Admin: The system administrator with complete access privileges can perform: Login, Manage Users with full CRUD operations, Add or Delete Managers and staff accounts, Block or Unblock Users for security and moderation, View System Analytics for platform-wide insights, Manage Food Categories for menu organization, View All Payments across the system, View All Reservations regardless of status, and Access All Reports for comprehensive oversight.

Relationships:

The use case diagram incorporates several relationship types to represent dependencies and extensions between use cases. "Includes" relationships represent mandatory dependencies where one-use case cannot be completed without another. Login is required and included for Place Order, ensuring only authenticated users can make purchases. Login is also required and included for Submit Review, maintaining accountability for feedback. Making Payment is required and included for Order Completion, ensuring transactional integrity. Send Email Notification is included in Register Account for OTP verification and included in Reserve Table for booking confirmation. Generate Token is included in Make Payment for order tracking purposes.

"Extends" relationships represent optional functionalities that enhance base use cases. View Order History extends from Place Order, providing customers with transaction records. Generate Receipt extends from Make Payment, offering downloadable proof of purchase. Update Profile extends from Login, allowing authenticated users to modify their account information.

Chapter 4.
Analysis

This chapter presents the structural and behavioral analysis of the Banglar Heshel system using standard UML diagrams. These diagrams are employed to illustrate the overall system logic, data flow, component responsibilities, and interactions between objects. The analysis helps in understanding how various parts of the system collaborate to fulfill functional requirements and supports better system design, implementation, and maintenance.

4.1 Software Analysis Pattern

We adopted the MERN Stack Architecture (MongoDB, Express.js, React.js, Node.js) for Banglar Heshel. This pattern organizes code into distinct layers with clear separation of concerns, ensuring that the front-end presentation logic, backend business logic, and data persistence remain independent and maintainable. This promotes scalability, testability, and ease of maintenance of the codebase.

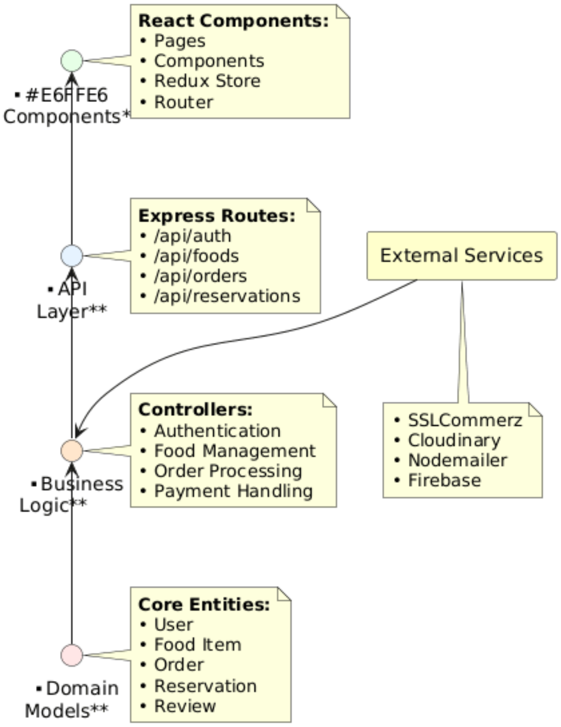


Figure 4.1: MERN Stack Architecture for Banglar Heshel

- **Frontend Layer (React.js):** The presentation layer containing UI components, Redux state management for global state (cart, user authentication, wishlist), and React Router for navigation. It communicates with the backend via Axios HTTP client and handles user interactions including form validations and dynamic content rendering.
- **Backend Layer (Node.js + Express.js):** The application layer containing RESTful API routes, authentication middleware using JWT tokens, and business logic controllers. It orchestrates data flow between frontend and database, manages third-party integrations (SSLCommerz payments, Cloudinary uploads, Nodemailer

emails), and enforces role-based access control for Admin, Manager, and User roles.

- **Database Layer (MongoDB + Mongoose):** The data persistence layer using MongoDB with Mongoose ODM for schema validation and modeling. It contains collections for Users, Food Items, Orders, Reservations, Reviews, Stock, Suppliers, and Purchase Orders with proper indexing for query optimization and referential integrity between related documents.
- **External Services Layer:** Integration with third-party APIs including SSLCommerz for secure payment processing, Cloudinary for image storage and CDN delivery, Nodemailer for automated email notifications with OTP verification, and Firebase for Google OAuth authentication. This architecture ensures single responsibility for each layer, loose coupling between components, and ease of testing and maintenance as requirements evolve..

4.2 Activity Diagrams

Activity diagrams illustrate the dynamic nature of the system by modeling the flow of control from activity to activity.

4.2.1 Activity Diagram for User Registration

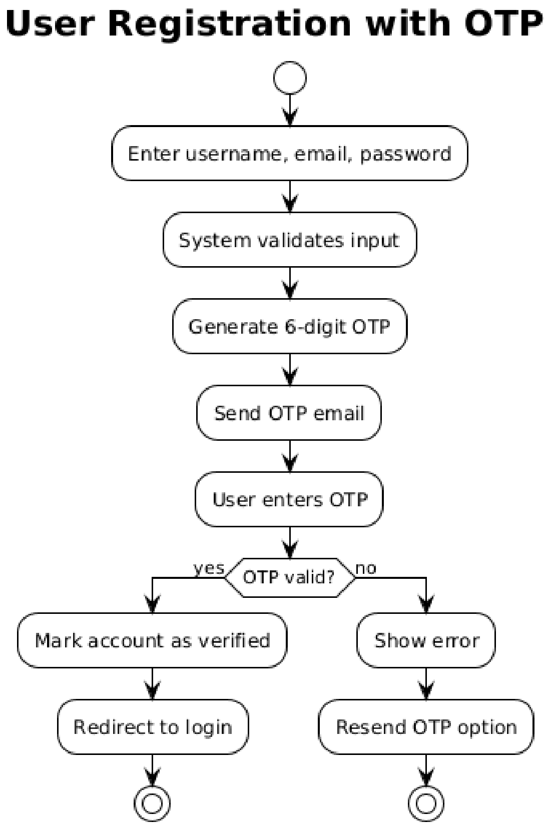


Figure 4.2: Activity Diagram for User Registration

4.2.2 Activity Diagram of User Login

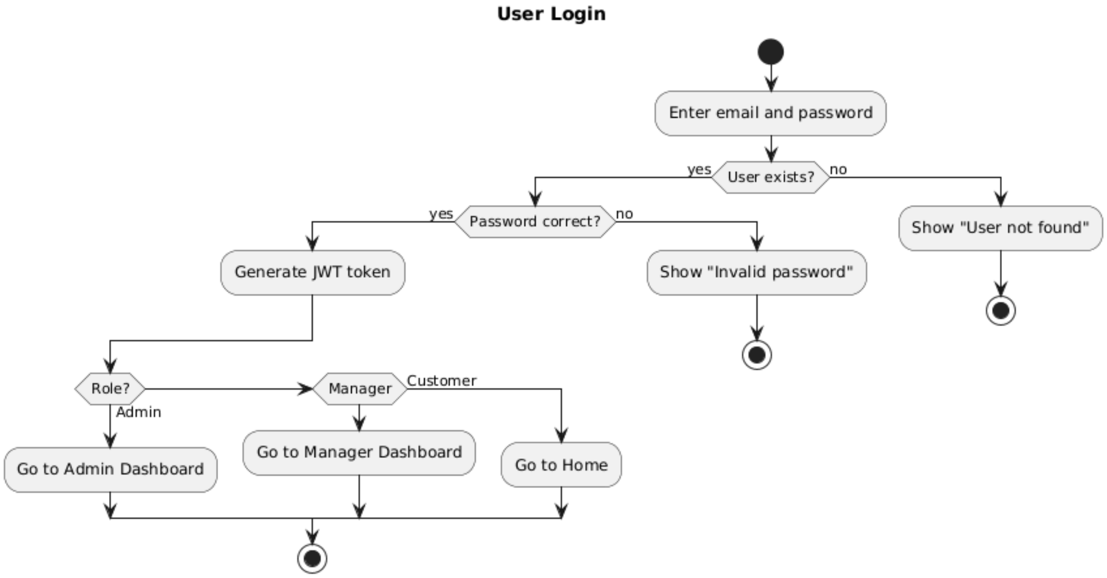


Figure 4.3: Activity diagram for User Login

4.2.3 Activity Diagram of Place Order

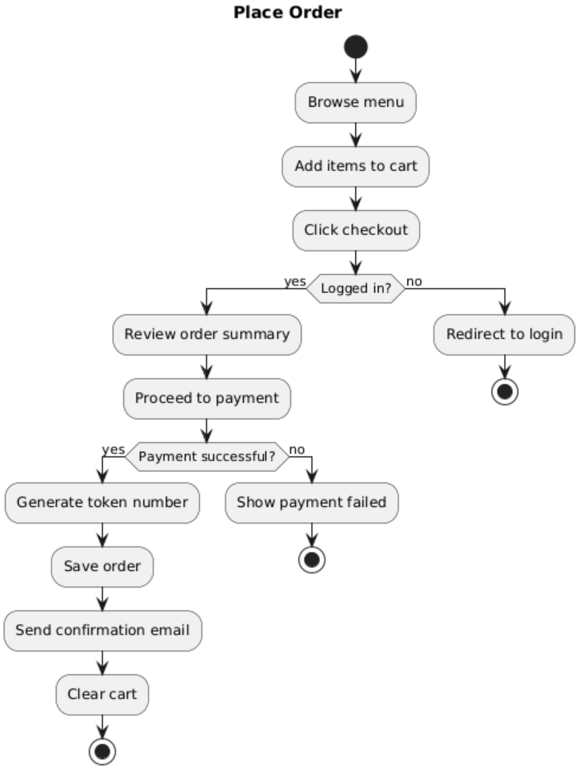


Figure 4.4: Activity Diagram of Place Order

4.2.4 Activity Diagram of Table Reservation

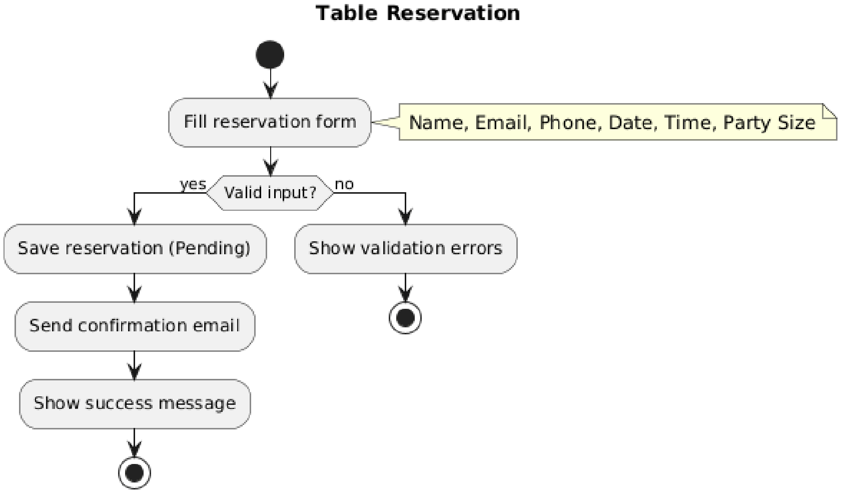


Figure 4.5: Activity Diagram of Table Reservation

4.2.5 Activity Diagram of Add Food Item

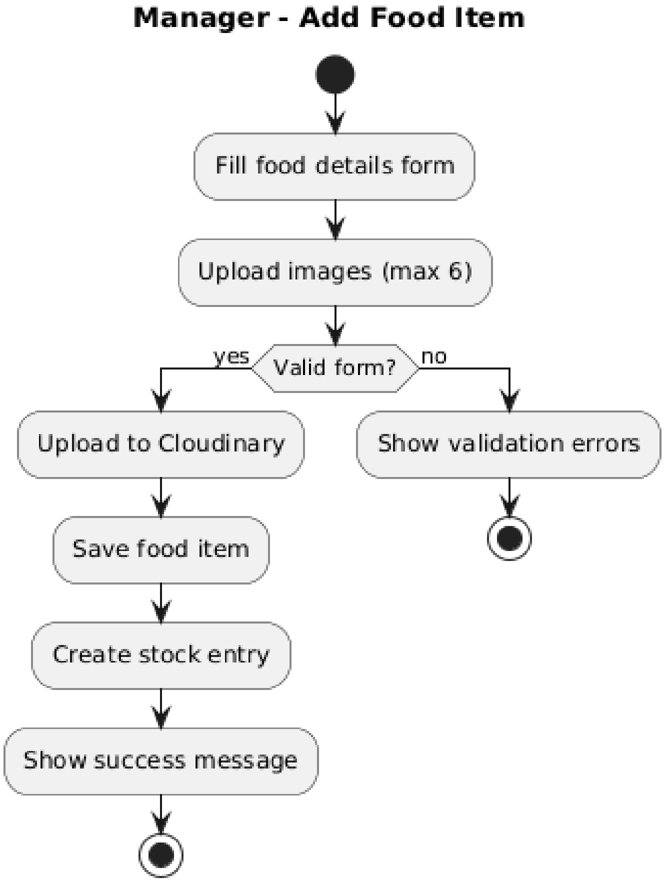


Figure 4.6: Activity Diagram of Add Food Item

4.2.6 Activity Diagram of Manager Update Stock

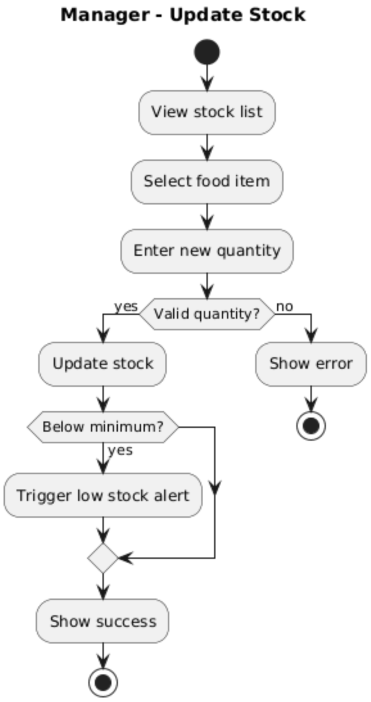


Figure 4.7: Activity Diagram of Manager Update Stock

4.2.7 Activity Diagram of Submit Review

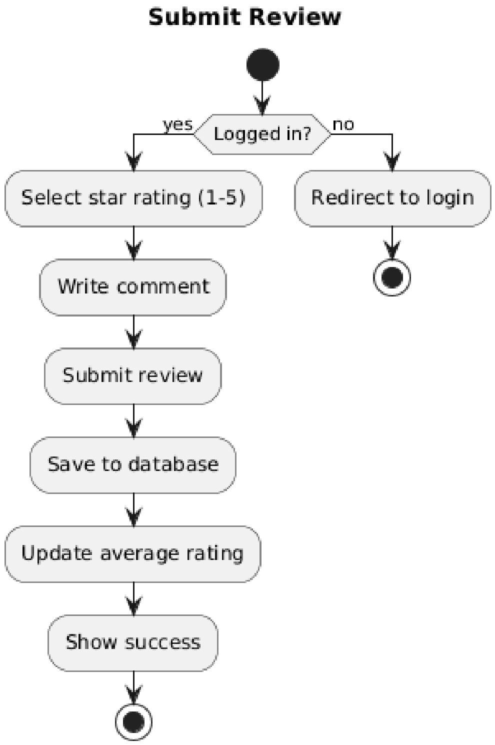


Figure 4.8: Activity Diagram of submit review

4.2.8 Activity Diagram of Forgot Password

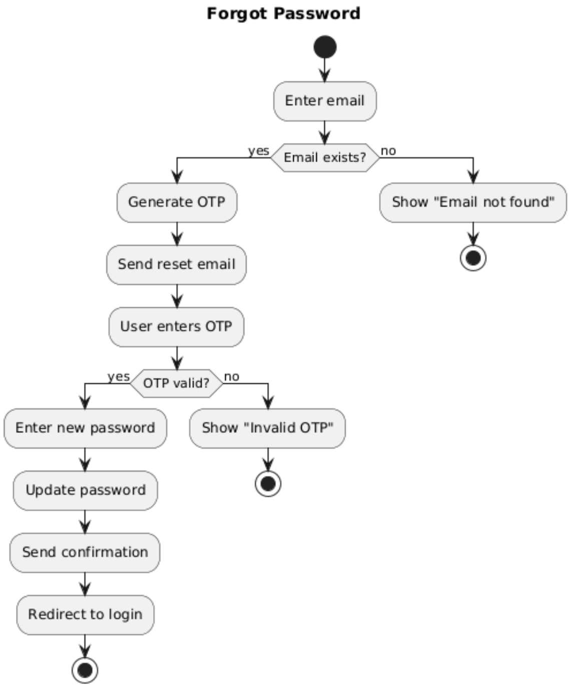


Figure 4.9: Activity Diagram of forgot Password

4.2.9 Activity Diagram of Manager view payments

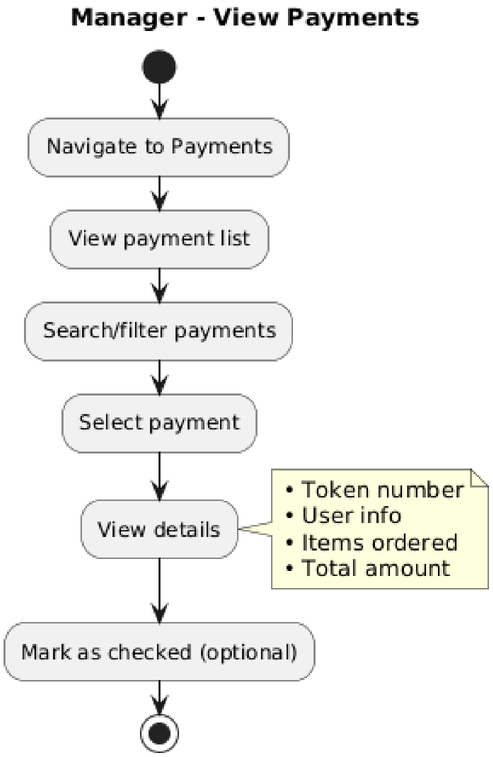


Figure 4.10: Activity Diagram of manager view payments

4.2.10 Activity Diagram of Manage Reservations

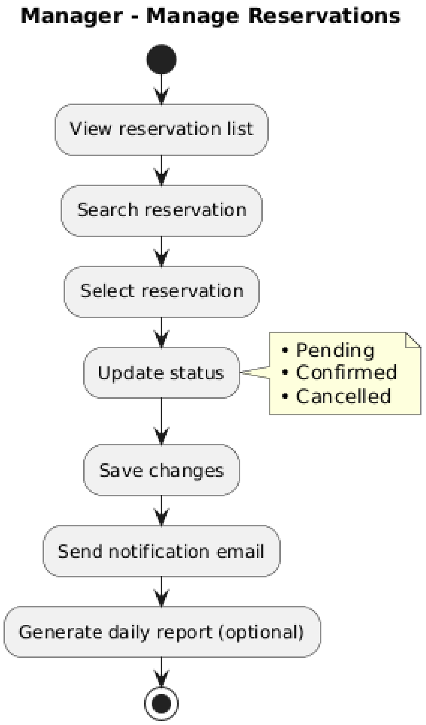


Figure 4.11: Activity Diagram of managing reservations

4.2.11 Activity Diagram of Add Manager

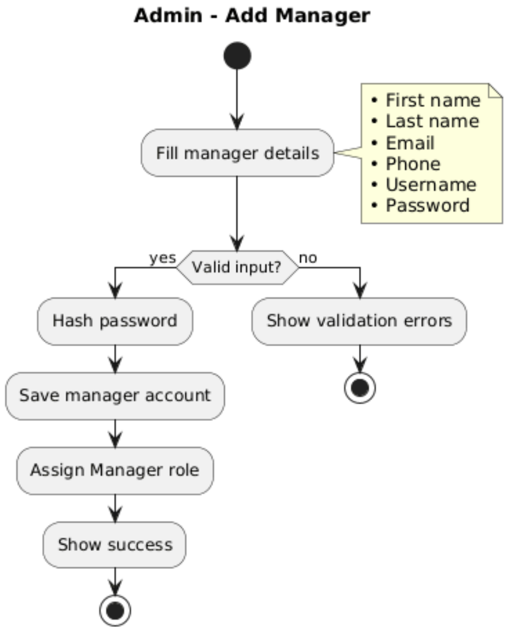


Figure 4.12: Activity Diagram of add manager

4.2.12 Activity Diagram of User Management

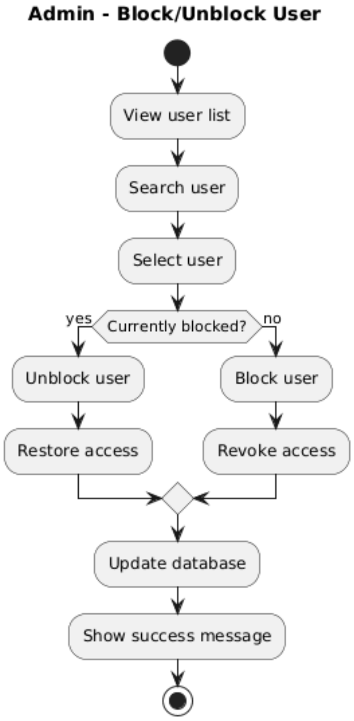


Figure 4.13: Activity Diagram of user management

4.3 Swim-lane Diagrams

Swim-lane diagrams delineate specific responsibilities within a process, clarifying the interaction between the User, System, Database, and External Services.

4.3.1 Swim-lane Diagram for User Registration

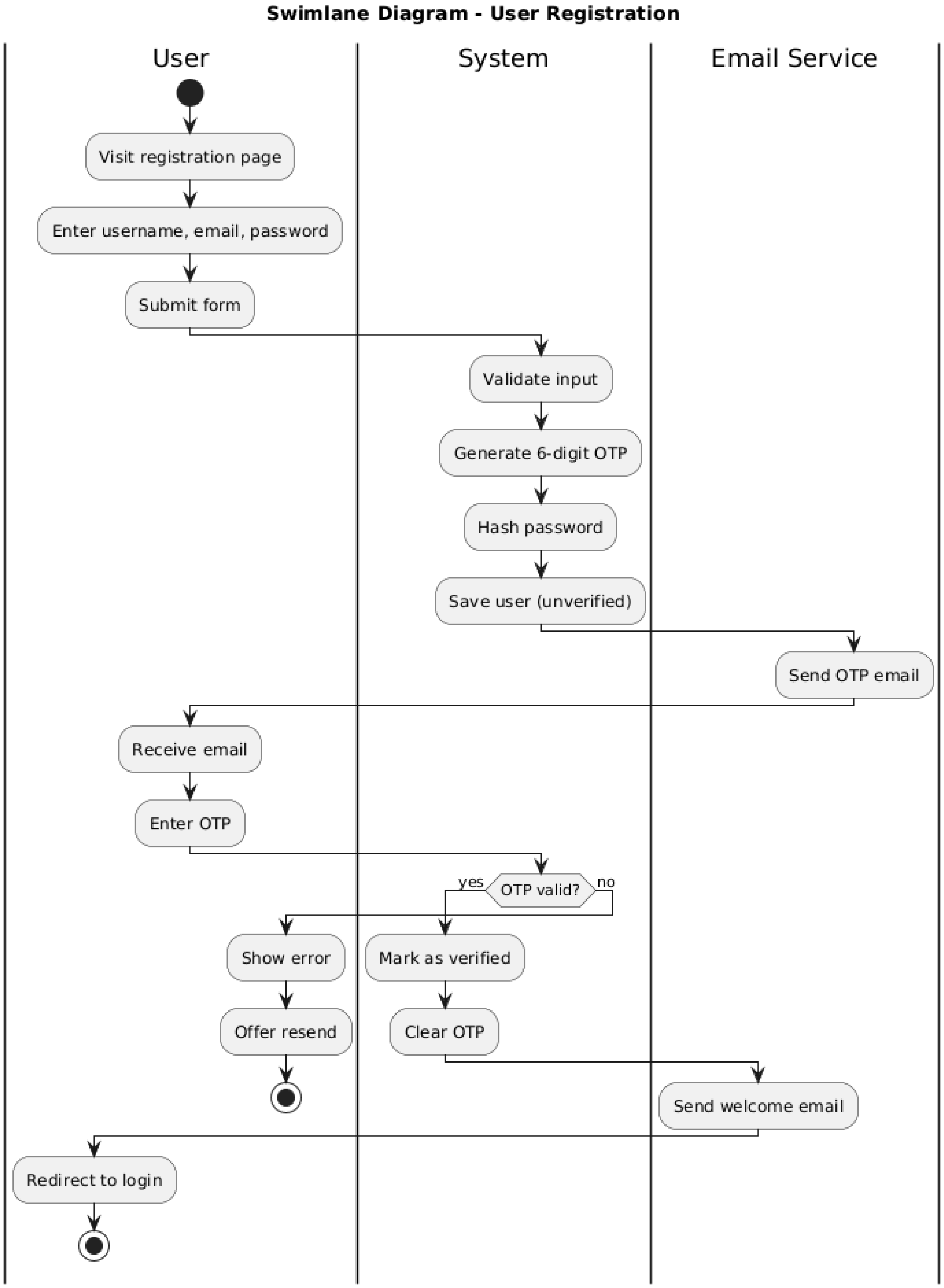


Figure 4.14: Swim-lane Diagram for User Registration

4.3.2 Swim-lane Diagram for Place Order

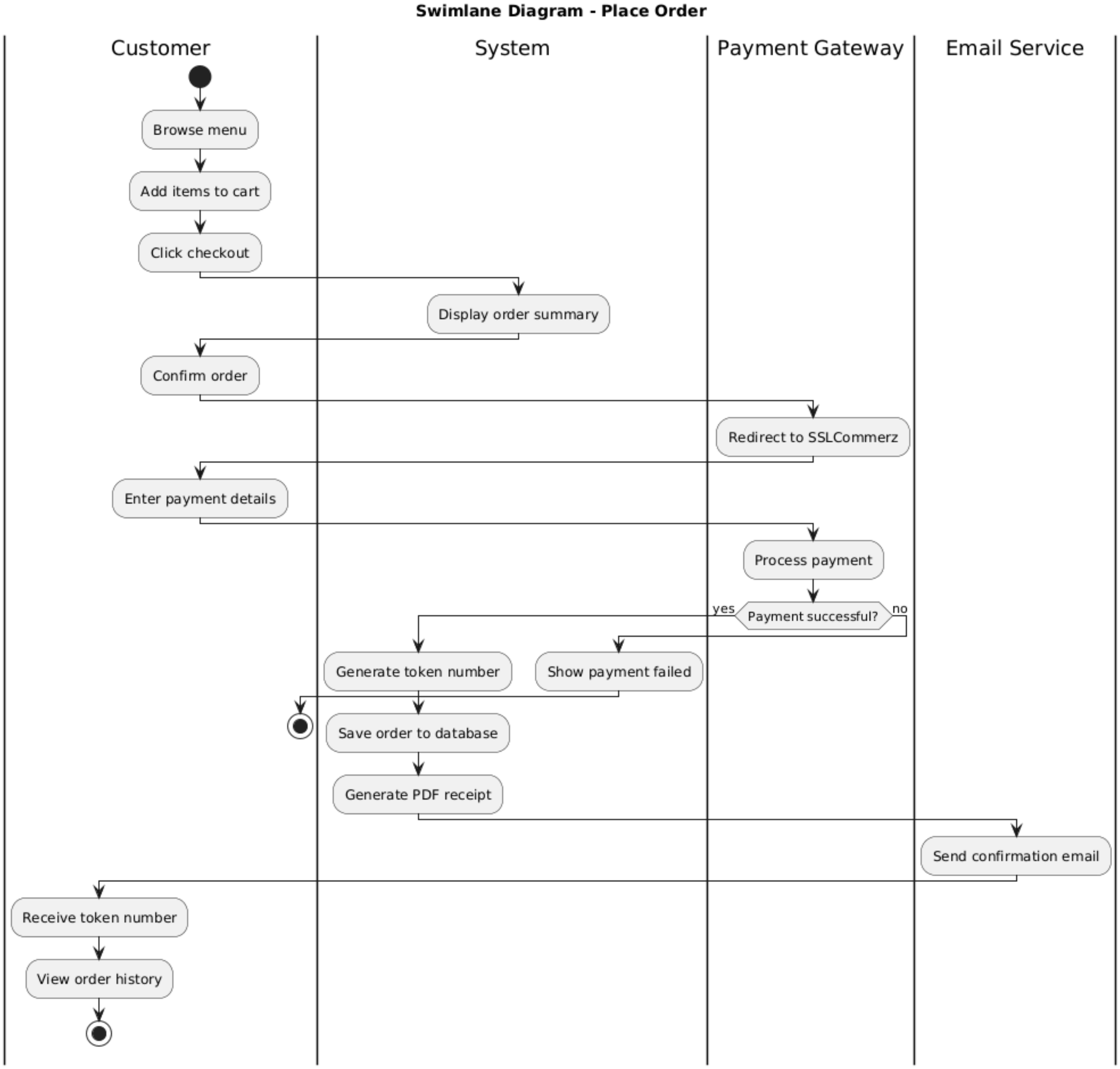


Figure 4.15: Swim-lane Diagram for Place Order

4.3.3 Swim-lane Diagram for Table Reservation

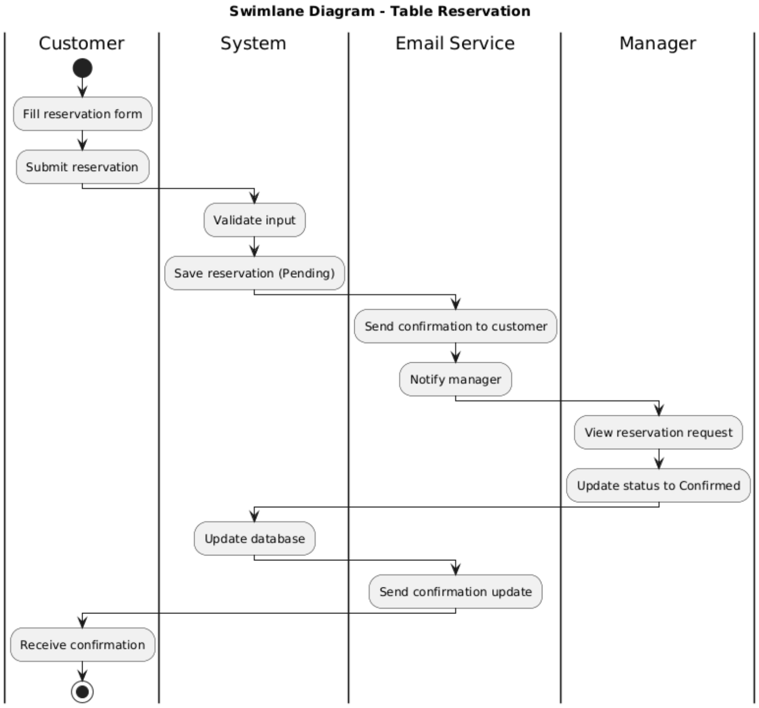


Figure 4.16: Swim-lane Diagram for Table Reservation

4.3.4 Swim-lane Diagram for Add Food Item

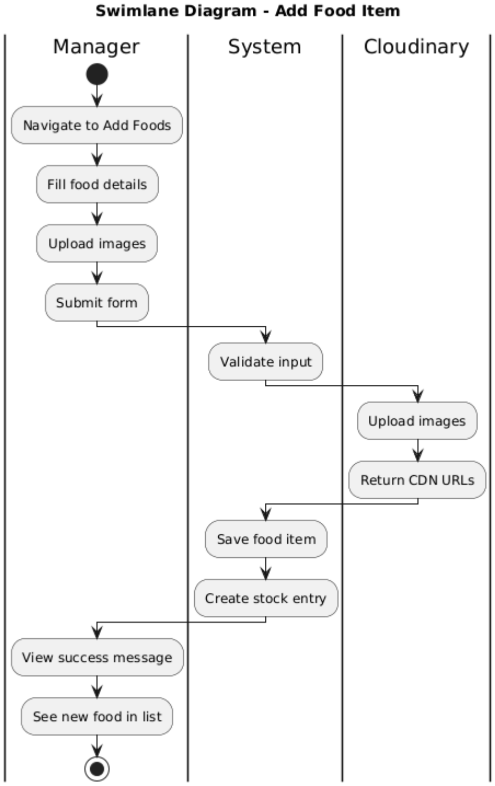


Figure 4.17: Swim-lane Diagram for Add Food Item

4.3.5 Swim-lane Diagram for Forgot Password

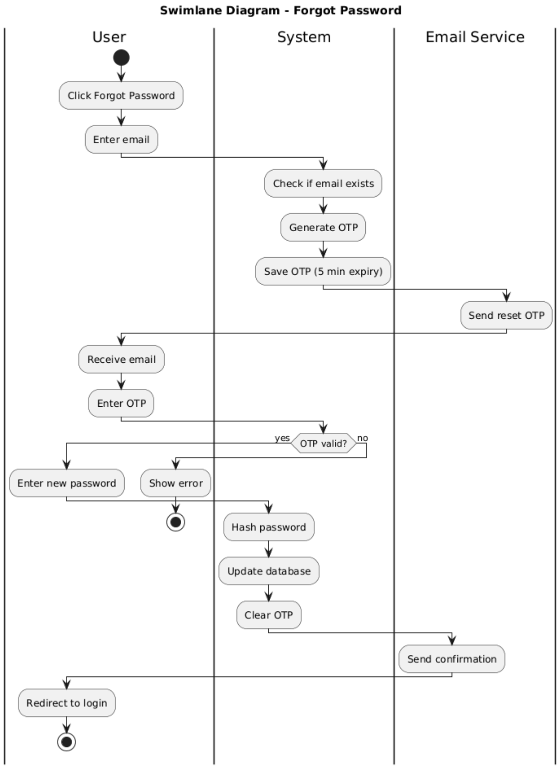


Figure 4.18: Swim-lane Diagram for Forgot Password

4.3.6 Swim-lane Diagram for Add Manager

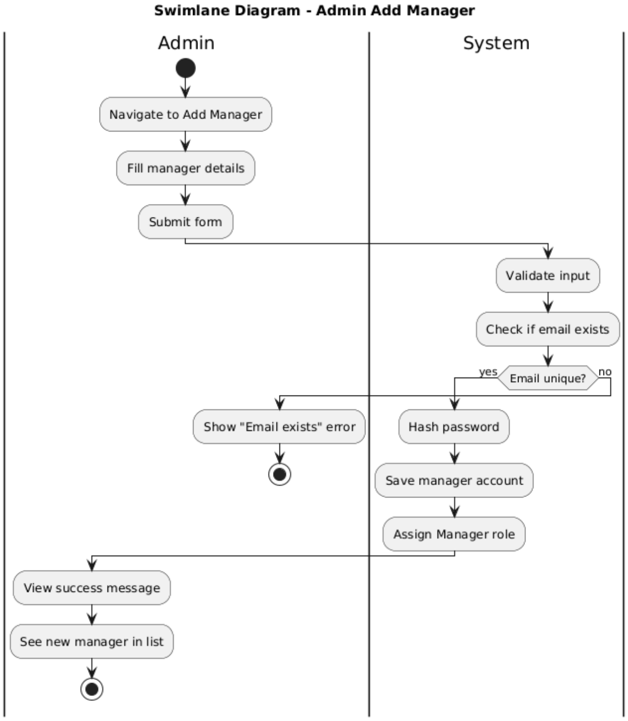


Figure 4.19: Swim-lane Diagram for Add Manager

4.4 Sequence Diagrams

Sequence diagrams capture the detailed interaction between objects over time for specific use cases.

4.4.1 Sequence Diagram for User Registration

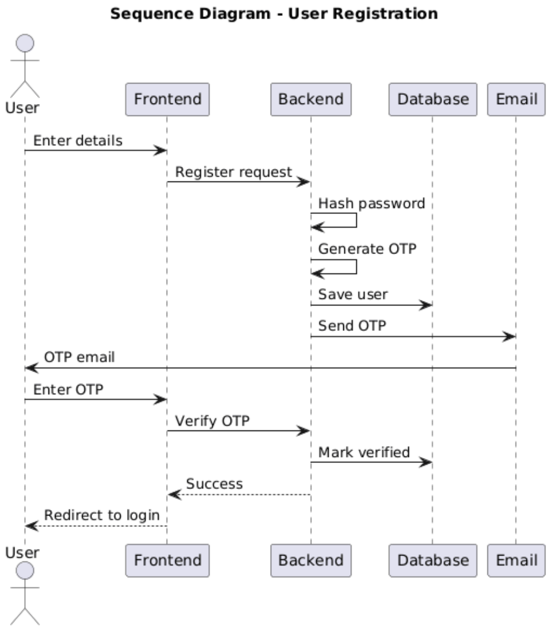


Figure 4.20: Sequence Diagram for User Registration

4.4.2 Sequence Diagram for Place Order

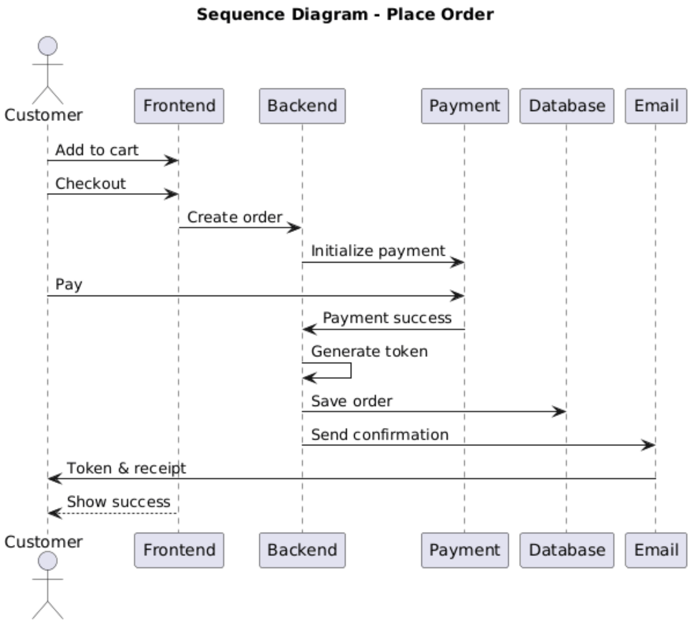


Figure 4.21: Sequence Diagram for Place Order

4.4.3 Sequence Diagram for Table Reservation

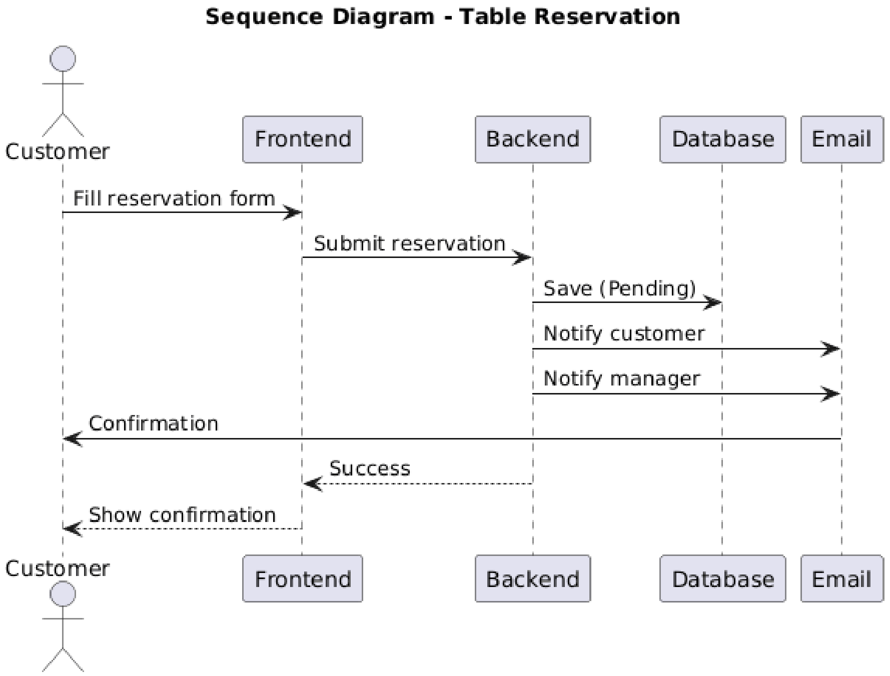


Figure 4.22: Sequence Diagram for Table Reservation

4.4.4 Sequence Diagram for Add Food Item

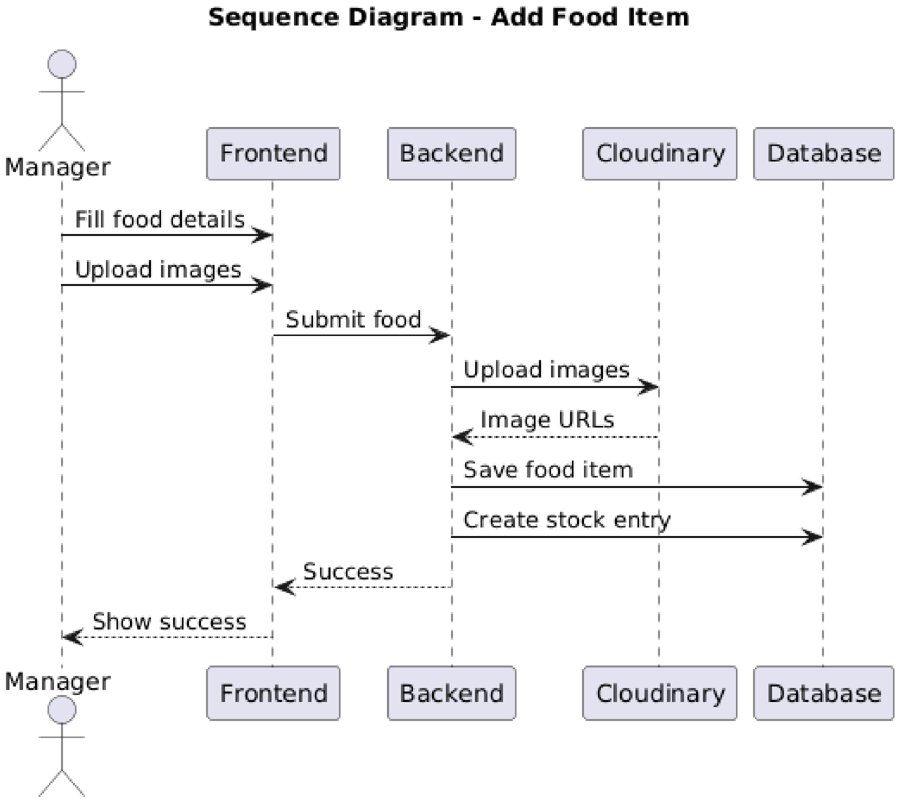


Figure 4.23: Sequence Diagram for Add Food Item

4.4.5 Sequence Diagram for Forgot Password

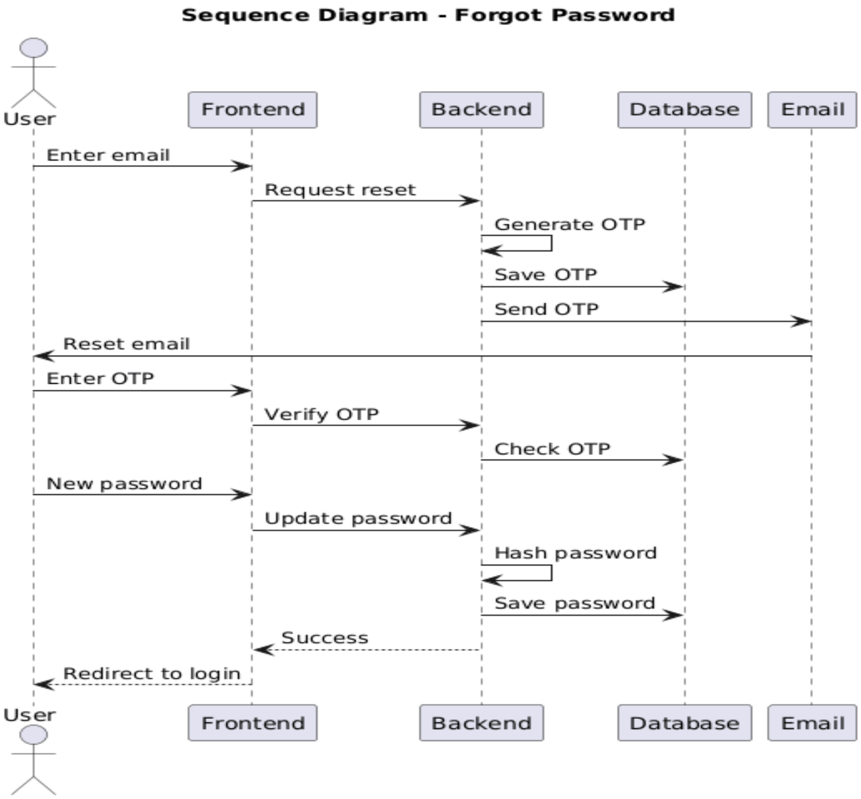


Figure 4.24: Sequence Diagram for Forgot Password

4.4.6 Sequence Diagram for Add Manager

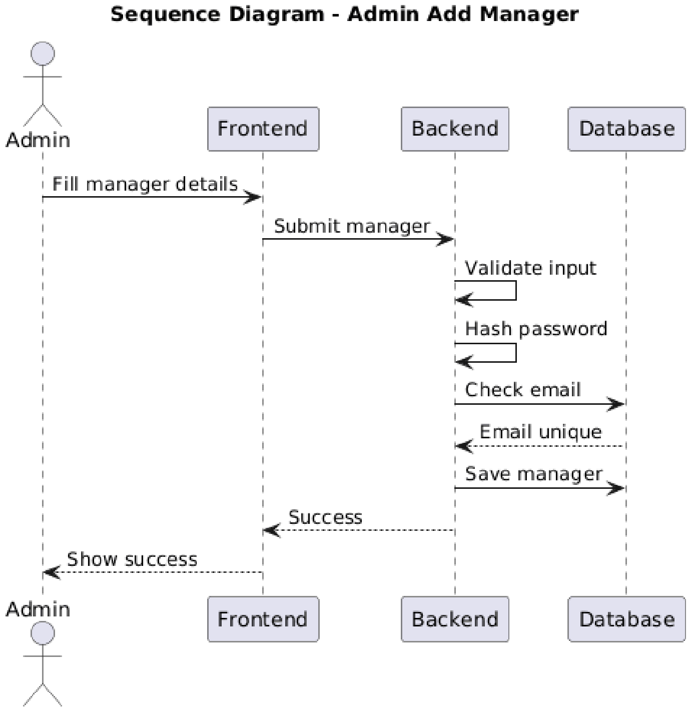


Figure 4.25: Sequence Diagram for Add Manager

4.4.7 Sequence Diagram for Update Stock

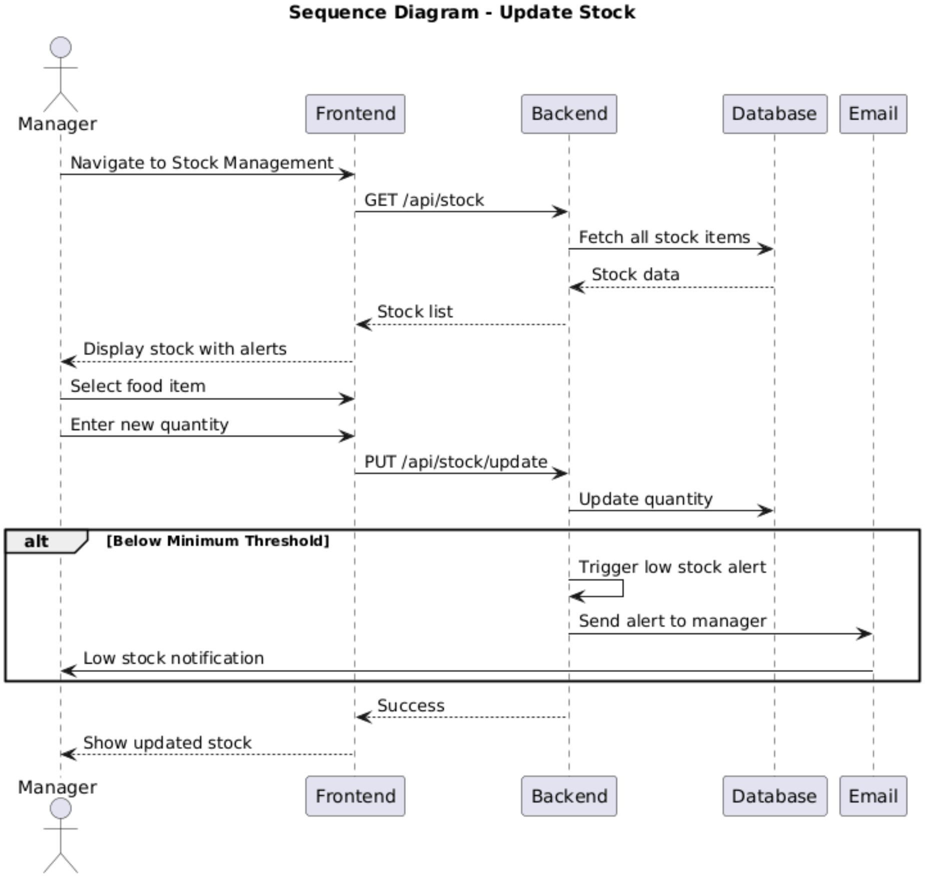


Figure 4.26: Sequence Diagram for Update Stock

4.5 Class Diagram of Banglar Heshel System

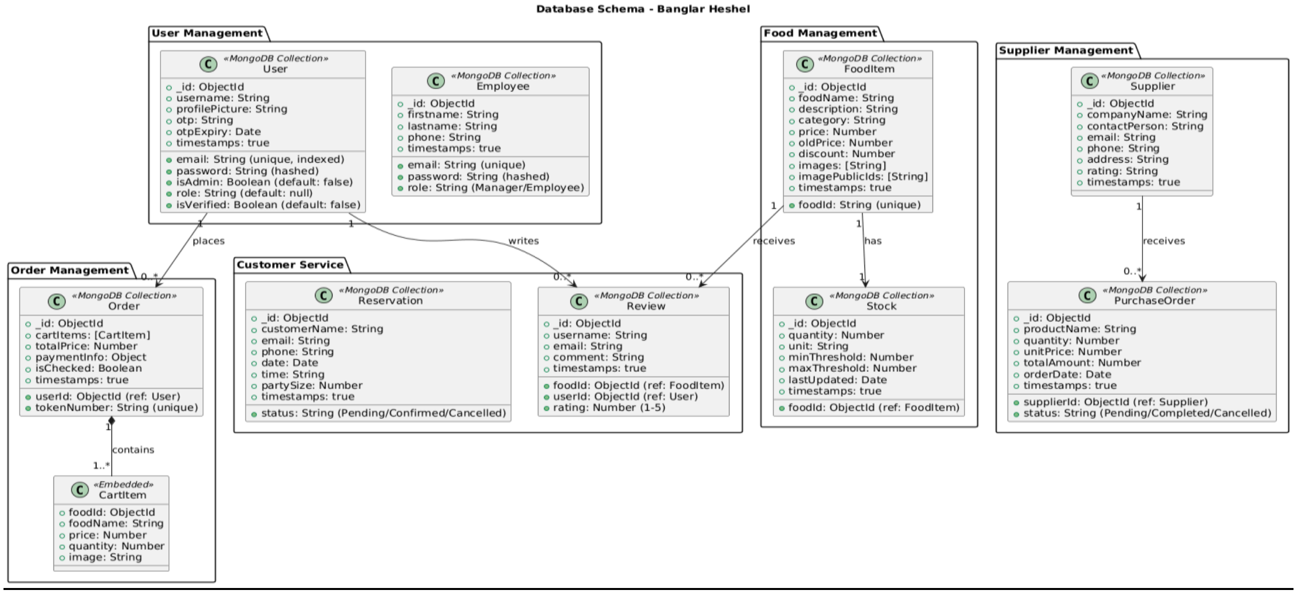


Figure 4.27: Class Diagram of Banglar Heshel System

Chapter 5.

Project Management

Effective project management was essential for the successful delivery of the Banglar Heshel platform. This chapter details the risk management strategies employed to ensure the project was completed on time, within budget, and to the specified quality standards. Given the system's reliance on external services (SSLCommerz payment gateway, Cloudinary cloud storage, Nodemailer email service, and Firebase authentication) and real-time data processing, proactive risk management was critical. The following sections outline the risk identification, analysis, planning, and the comprehensive Risk Mitigation, Monitoring, and Management (RMMM) plan.

5.1 Risk Identification

Risk identification involved a systematic analysis of potential threats that could impact the project's timeline, budget, technical functionality, or quality. The risks were categorized into technical, operational, and external factors.

5.1.1 Technical Risks

TR-1: **Payment Gateway Integration Failure**

The integration with SSLCommerz payment gateway posed significant risk due to complex API requirements, sandbox-to-production migration challenges, and potential security vulnerabilities in handling sensitive payment data.

TR-2: **Database Performance Issues**

MongoDB performance degradation under heavy load, particularly for complex queries involving multiple collections (orders with populated user and food data), could impact user experience during peak ordering times.

TR-3: **Image Upload and Storage Limitations**

Cloudinary integration risks included upload failures, storage quota exceeded, CDN delivery issues, and image optimization problems affecting page load times.

TR-4: **Email Delivery Failures**

Nodemailer with Gmail SMTP dependency posed risks including emails marked as spam, Gmail account suspension due to high volume, OTP delivery delays, and SMTP authentication failures.

TR-5: **Authentication Security Vulnerabilities**

JWT token management, OTP generation security, password hashing implementation, and session management could introduce security vulnerabilities if not properly implemented.

TR-6: **Third-Party API Dependency**

Reliance on external services (SSLCommerz, Cloudinary, Firebase) introduced risks of service outages, API changes, rate limiting, and unexpected cost increases.

5.1.2 Operational Risks

OR-1: **Data Loss or Corruption**

Without proper backup mechanisms, database failures could result in loss of critical data including orders, payments, reservations, and user accounts.

OR-2: **Scalability Challenges**

The system's ability to handle increasing numbers of concurrent users, orders, and file uploads during peak hours (lunch and dinner times) was uncertain.

OR-3: **Insufficient Testing Coverage**

Limited time for comprehensive testing of all user flows, edge cases, payment scenarios, and role-based access control could result in production bugs.

OR-4: **Inadequate Documentation**

Poor code documentation and lack of system documentation could hinder future maintenance, feature additions, and knowledge transfer.

5.1.3 External Risks

ER-1: **Third-Party Service Pricing Changes**

SSLCommerz transaction fees, Cloudinary storage costs, and potential MongoDB Atlas pricing changes could impact project budget and operational costs.

ER-2: **Security and Compliance Requirements**

Payment Card Industry Data Security Standard (PCI DSS) compliance requirements, data protection regulations, and security best practices needed adherence to avoid legal issues.

ER-3: **Browser Compatibility Issues**

Variations in browser behavior, particularly for payment redirects and file uploads, could affect user experience across different browsers and devices

5.2 Risk Analysis

Following identification, each risk was analyzed to determine its probability and impact. This allowed us to calculate the **Risk Exposure** and assign a **Priority** level to focus our mitigation efforts effectively. **RE = Probability × Impact**

Risk ID	Risk Description	Probability	Impact	Risk Exposure	Priority

TR-1	Payment Gateway Integration Failure	3	5	15	High
TR-2	Database Performance issues	2	4	8	Medium
TR-3	Image upload and Stroage Limitations	3	3	9	Medium
TR-4	Email Delivery Failures	4	3	12	High
TR-5	Authentication Security Vulnerabilities	2	5	10	High
TR-6	Third-Party API Dependency	3	4	12	High
OR-1	Data Loss or Corruption	2	5	10	High
OR-2	Scalability	3	3	9	Medium
OR-3	Insufficient Testing Coverage	4	3	12	High
OR-4	Inadequate Documentation	3	2	6	Low

5.3 Risk Planning

For each identified risk, a specific response strategy was developed. This planning phase ensured that the team was prepared to act immediately should any risk materialize.

Table 5.3: Risk Planning Table

Risk ID	Risk	Mitigation Strategy
TR1	Payment Gateway Failure	Sandbox testing, error handling, webhooks, fallback mechanisms

TR-4	Email Delivery Failures	App-specific passwords, retry queue, OTP expiry, HTML, Templates, rate limiting
TR-5	Security Vulnerabilities	Bcrypt hashing, JWT tokens, Secure OTPs, HTTP-only cookies, password requirements
TR-6	API dependency	Try-catch blocks, timeouts, graceful degradation, caching, environment variables
OR-1	Data Loss	Automated backups, point-in-time recovery, Schema validation, transactions
OR-3	Testing Coverage	Incremental development, test scenarios, UAT, role-based testing, sandbox testing

5.4 The RMMM Plan

The Risk Mitigation, Monitoring, and Management (RMMM) plan provided a framework for ongoing risk oversight. Managing these risks was a continuous process throughout the development of the Banglar Heshel system. By anticipating issues such as payment gateway integration challenges, email delivery failures, and third-party API dependencies, we allocated resources efficiently to maintain steady progress..

1. **Risk Mitigation**
    - **Incremental Development:** Adopted an agile approach, delivering core features (User Authentication, Menu Management) first before complex integrations (Payment Gateway, Email Service, Image Upload).
    - **External Service Abstraction:** Wrapped third-party APIs (SSLCommerz, Cloudinary, Nodemailer) in service interfaces to allow easy swapping or mocking during testing.
    - **Security First:** Applied authentication best practices including bcrypt password hashing, JWT token management, and OTP-based verification early in the development phase.
    - **Comprehensive Testing:** Implemented sandbox testing for payment gateway before production deployment to identify and resolve integration issues.
    - **Data Protection:** Configured MongoDB Atlas automated backups and implemented Mongoose schema validation to prevent data loss and corruption.
2. **Risk Monitoring**
    - **Weekly Progress Reviews:** Discussed potential blockers and new risks during project review sessions.
    - **Log Monitoring:** Tracked API response times, payment success rates, email delivery rates, and error logs for all external services.
    - **Performance Tracking:** Monitored database query performance, image upload success rates, and application response times during development and testing phases.
    - **User Feedback Loop:** Monitored beta tester feedback regarding ordering workflow, payment process, reservation system, and overall user experience.
3. **Risk Management**
    - **Fallback Provisions:** Maintained default placeholder images and graceful error messages in case external services (Cloudinary, Nodemailer) failed to load.
    - **Backup Email Service:** Prepared alternative email provider (SendGrid) configuration in case Gmail SMTP issues arose.

- **Manual Verification Option:** Created manual order verification process as backup for payment gateway failures.
- **Rate Limiting:** Implemented request throttling for image uploads and API calls to prevent quota exhaustion and service suspension.
- **Documentation:** Maintained detailed API integration documentation, error handling procedures, and troubleshooting guides to resolve integration issues quickly.

Table 5.4: Risk Mitigation, Monitoring, and Management Table

Risk ID	Risk	**Mitigation Strategy**	**Monitoring Method**
TR-1	Payment Gateway Failure	Sandbox testing, error handling, webhooks, fallback mechanisms	Log transactions, alert failures, review success rates
TR-4	Email Delivery Failures	App-specific passwords, retry queue, OTP expiry, HTML templates, rate limiting.	Track delivery rates, monitor account, log OTPs
TR-5	Security Vulnerabilities	Bcrypt hashing, JWT tokens, secure OTPs, HTTP-only cookies, password requirements.	Log attempts, monitor patterns, track OTPs
TR-6	API Dependency	Try-catch blocks, timeouts, graceful degradation, caching, environment variables.	Track response times, monitor status, quota alerts
OR-1	Data Loss	Automated backups, point-in-time recovery, schema validation, transactions	Verify backups, monitor storage, track errors
OR-3	Testing Coverage	Incremental development, test scenarios, UAT, role-based testing, sandbox testing	Track bugs, monitor feedback, log errors

Chapter 6.

Project Planning

To ensure successful execution, the **Banglar Heshel** project required a structured project plan. This involved defining project objectives, system scope, and critical milestones such as requirements analysis, system design, development, testing, and deployment. The plan also addressed resource allocation, budget estimation, and risk management strategies. Key deliverables include administrative features (user management, manager oversight, system analytics), manager features (food management, inventory control, payment tracking, reservation management, supplier coordination), and customer features (menu browsing, online ordering, secure payments, table reservations, review submission). The system ensures a smooth interface with features like shopping cart management, real-time stock alerts, token-based order tracking, and secure role-based access control.

The following activities of project planning were followed in this system:

1. Estimation of project effort and resources - Assessed development time for each module including authentication, menu management, payment integration, and inventory system.

2. Task prioritization based on dependencies and risks - Core features (authentication, database setup) were prioritized before advanced features (payment gateway, analytics dashboard).

3. Personnel requirements for development and testing - Identified skills needed for MERN stack development, third-party API integration, and comprehensive system testing.

4. Estimation of project cost and schedule - Calculated development timeline, cloud service costs (MongoDB Atlas, Cloudinary), and payment gateway transaction fees.

5. Version control for proper code and document management - Utilized Git and GitHub for source code versioning, collaboration, and project documentation maintenance.

6.1 System Project Estimation

The accuracy of estimation in the Banglar Heshel system depends on:

1. **Estimating product size:** Scope includes modules such as User/Manager/Admin Authentication with OTP verification, Food and Menu Management with multi-image upload, Shopping Cart and Wishlist functionality, Table Reservation System, Payment Gateway Integration (SSLCommerz) with token generation, Review and Rating System, Inventory and Stock Management with automated alerts, Supplier and Purchase Order Management, and comprehensive Admin/Manager Dashboards with analytics.
2. **Effort, Calendar Time, and Cost:** Time and cost are estimated considering system

complexity (specifically payment gateway integration, cloud storage management, and email automation), number of modules, and skill requirements of the MERN stack development team. This involves effort for modular architecture setup, responsive frontend design with React and Tailwind CSS, RESTful API development, MongoDB database schema design, and third-party API integrations (SSLCommerz, Cloudinary, Nodemailer, Firebase).

3. **Consistency of requirements:** Requirements were kept stable throughout development, focused on the core "Restaurant Management System" with clearly defined features for online ordering, table reservations, inventory control, and multi-role access management. Changes were minimal and primarily involved refinements to user interface elements and workflow optimizations based on testing feedback.

6.2 Function-Oriented Metrics

The project estimation is based on function point analysis, which considers both data functions and transactional functions.

6.2.1. Data Functions

- **Internal Logical Files (ILF):** Data maintained within the system, such as Users (with authentication and profile data), Food Items (with images and pricing), Orders (with cart items and payment details), Reservations (with customer and booking information), Reviews (with ratings and comments), Stock (with inventory levels and thresholds), Suppliers (with contact and rating information), Purchase Orders (with order details and status), and Employees/Managers (with role-based access credentials).
- **External Interface Files (EIF):** Interfaces connecting to external services like SSLCommerz (payment gateway for secure transactions), Cloudinary (cloud storage for image management and CDN delivery), Nodemailer (email service for OTP verification and notifications), and Firebase (Google OAuth authentication).

6.2.2. Transactional Functions

- **External Inputs (EI):** User actions such as registering with email, verifying OTP, creating food items with image uploads, adding items to cart, placing orders, making payments through SSLCommerz, submitting table reservations, writing reviews with star ratings, updating stock quantities, creating purchase orders, and managing supplier information.

- **External Outputs (EO):** Order confirmations with unique token numbers, digital PDF receipts with itemized details, reservation confirmation emails, payment success notifications, low stock alert emails, daily reservation reports, employee list reports, and analytics dashboard visualizations.
- **External Inquiries (EQ):** Real-time queries such as browsing food menu by category, searching food items by keywords, viewing food details with reviews and ratings, checking stock availability, filtering payments by token or user, searching reservations by customer information, viewing order history, and accessing analytics with sales trends.

6.3 Identifying Complexity

6.3.1 Identifying Complexity of Transaction Functions

- **Low complexity:** Simple inputs such as user login, logout, viewing menu pages, profile updates, and browsing food categories.
- **Medium complexity:** Food item creation (requires multi-image upload to Cloudinary), submitting table reservations with email notifications, writing reviews with star ratings, updating stock quantities, and keyword-based food search with filtering.
- **High complexity:** The order placement process (involves multiple steps: cart management → payment gateway redirect → SSLCommerz transaction → token generation → order confirmation → email notification), secure payment processing (SSLCommerz integration with webhooks and callbacks), inventory management with automated stock alerts (triggers email notifications when quantity falls below minimum threshold), and analytics dashboard generation (aggregates data from multiple collections and creates visualized reports).

6.3.2 Identifying Complexity of Data Functions

- **Low complexity:** Static data such as food categories, user roles (Admin, Manager, User), and reservation status options (Pending, Confirmed, Cancelled).
- **Medium complexity:** User profiles (authentication data with OTP verification + profile information + role-based permissions), review records (links to food items and users with ratings and timestamps), and supplier information (contact details and performance ratings).
- **High complexity:** Food item records (links to categories, multiple images with Cloudinary URLs, pricing with discounts, stock information, and aggregated review ratings), order records (links to users, contains embedded cart items array with food

details, payment information from SSLCommerz, unique token numbers, and timestamps), stock records (links to food items with quantity tracking, unit types, minimum/maximum thresholds, and last updated timestamps), and purchase orders (links to suppliers with product details, pricing calculations, order status history, and date tracking).

6.3.3 Unadjusted Function Point Contribution

Table 6.1: Unadjusted Function Point Contribution (Transaction Function)

Transaction Functions	FTRs	DETs	Complexity	UFP
User Registration with OTP (EI)	2	12	Average	4
Authentication (Login/OTP Verify) (EI)	1	6	Low	3
Browse Food Menu (EQ)	1	8	Low	3
Search for Food Items (EQ)	2	10	Average	4
View Food Details with Reviews (EQ)	3	18	High	6
Add/Update Food Item (EI)	2	20	Average	4
Upload Images to Cloudinary (EI)	1	8	Average	4
Add to Cart/Wishlist (EI)	1	6	Low	3
Place Order with Payment (EI)	3	15	High	6
Process Payment (SSLCommerz) (EO)	2	12	High	7
Create Table Reservation (EI)	1	10	Average	4
Manage Reservations (EI)	2	12	Average	4
Submit Review & Rating (EI)	2	8	Average	4
Update Stock Levels (EI)	1	8	Low	3
Create Purchase Order (EI)	2	10	Average	4
Admin User Management (EI)	1	8	Low	3
Generate PDF Reports (EO)	2	12	Average	5
Send Email Notifications (EO)	1	8	Average	5
View Analytics Dashboard (EQ)	2	15	Average	4

Total (Transaction)				**70**

6.3.4 Unadjusted Function Point Contribution

Table 6.2: Unadjusted Function Point Contribution (Data Function)

Data Function (ILF/EIF)	FTRs	DETs	Complexity	UFP
User Collection (ILF)	2	15	Average	10
Employee Collection (ILF)	2	12	Average	10
Food Item Collection (ILF)	3	18	High	15
Order Collection (ILF)	4	20	High	15
Reservation Collection (ILF)	2	10	Average	10
Review Collection (ILF)	2	8	Low	7
Stock Collection (ILF)	2	10	Average	10
Supplier Collection (ILF)	1	8	Low	7
PurchaseOrder Collection (ILF)	2	10	Average	10
Food Categories (ILF)	1	5	Low	7
SSLCommerz API (EIF)	1	12	Average	7
Cloudinary API (EIF)	1	8	Average	7
Nodemailer Service (EIF)	1	6	Low	5
Firebase OAuth (EIF)	1	8	Average	7
Total (Data)				**127**

6.3.5 Performance and Environmental Impact

Table 6.3: Performance and Environmental Impact (GSC)

General System Characteristic	Rating (0–5)
Data Communications	5
Distributed Data Processing	3
Performance (Real-time ordering)	4
Heavily Used Configuration	3
Transaction Rate	4
Online Data Entry	5
End-user Efficiency	4
Online Update	5
Complex Processing (Payment/Stock)	4
Reusability (MERN Stack)	4
Installation Ease	4
Operational Ease	4
Multiple Sites	2
Facilitate Change	4
Total Degree of Influence (TDI)	**55**

6.3.6 Counting Adjusted Function Point

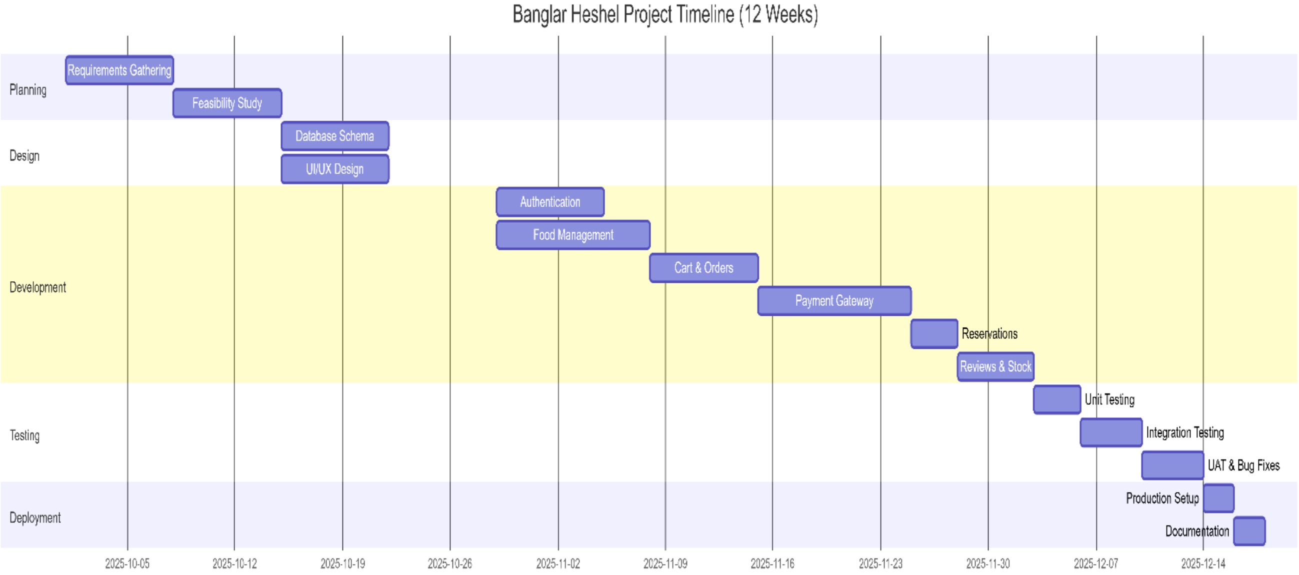
Function Point Calculation

- UFP for Transaction = 70
- UFP for Data = 127
- Total UFP = 197
- Value Adjustment Factor (VAF) = 0.65 + (0.01 × 55) = 1.10
- Adjusted FP (AFP) = 197 × 1.10 = 216.7 ≈ 217

- Effort = AFP × Productivity (Assumed MERN stack productivity ≈ 18 FP/month for small teams) = 217 ÷ 18 ≈ 12 person-months
- With a team of 4 developers: Duration ≈ 12 ÷ 4 ≈ 3 months (approx. 12 weeks)

6.4 Project Schedule Chart (12- week timeframe)

The Banglar Heshel system project follows a structured 12-week timeline running from October 2025 to January 2026, using an incremental, phase-based approach with parallel development streams. The project begins with a 2-week Planning phase focused on requirements gathering and system analysis, followed by a 2-week Design phase covering database schema design, system architecture, and UI/UX mockups.



The core Development phase spans 6 weeks, during which the team of 4 developers worked on different modules in parallel including authentication and user management, food and menu management with Cloudinary integration, shopping cart and order placement, payment gateway integration with SSLCommerz, table reservation system, review and rating functionality, inventory and stock management, and supplier/purchase order management.

The Testing phase requires 1.5 weeks for comprehensive unit testing, integration testing, payment flow testing, and user acceptance testing. Finally, the Deployment and Documentation phase takes 0.5 weeks for production deployment setup, system documentation, and final project report preparation.

6.5 Accounts Table

Project costs are estimated based on a standard development environment for a medium-scale academic project utilizing the MERN stack and cloud services.

6.5.1 Personnel Cost

The largest portion of the budget is dedicated to personnel, as skilled manpower is essential for successful development and deployment.

Table 6.4: Personal Cost

Role	**No. of Persons**	**Est. Cost (Monthly)**	**Duration**	**Total**
Project Manager	1	70,000 BDT	3 Months	210,000 BDT
Backend Developer (Node.js/Express)	2	45,000 BDT	3 Months	270,000 BDT
Frontend Developer (React.js)	1	40,000 BDT	3 Months	120,000 BDT
Total Personnel				**600,000 BDT**

6.5.2 Expected Hardware Cost

To support development, testing, and deployment, certain hardware resources are required.

Table 6.5: Expected Hardware Cost

Item	**Quantity**	**Unit Price**	**Total Cost**
Development Laptops	4	(Existing Assets)	0 BDT
Testing Devices (Mobile/Tablet)	3	(Existing Assets)	0 BDT
Total Hardware			**0 BDT**

6.5.3 Expected Software Cost

Cloud services and development tools are considered in the cost structure.

Table 6.6: Expected Software Cost

Item	Type	Cost
VS Code	IDE	Free
MongoDB Atlas	Database (Free Tier)	Free
Node.js & npm	Runtime	Free
React.js	Frontend Framework	Free
Cloudinary	Image Storage (Free Tier)	Free
SSLCommerz	Payment Gateway (Sandbox)	Free
Nodemailer	Email Service (Gmail SMTP)	Free
Firebase	Authentication (Free Tier)	Free
Postman	API Testing	Free
Git/GitHub	Version Control	Free
Render/Railway	Backend Hosting (Free Tier)	Free
Domain Name	Web	1,200 BDT
Total Software		**1,200 BDT**

6.5.4 Expected Other Cost

Other costs include miscellaneous but essential expenses.

Table 6.7: Expected Other Cost

Item	Description	Cost
Internet & Utility	High-speed connection (3 months)	6,000 BDT
Documentation	Printing & Binding	2,000 BDT
Miscellaneous	Contingency fund (API upgrades, testing)	5,000 BDT
Total Other		**13,000 BDT**

6.5.5 Total Project Cost Summary

Cost Category	**Amount (BDT)**
Personnel Cost	600,000
Hardware Cost	0
Software Cost	1,200
Other Cost	13,000
Grand Total Estimated Project Cost	**614,200 BDT**

Grand Total Estimated Project Cost: 614,200 BDT

6.6 Clarification of Academic vs. Industry Estimates

It is important to note that this budget represents an industry implementation scenario. In the context of this practicum report, the actual cost was close to zero because the project was completed as an academic exercise using free or personal resources. The detailed accounts table therefore illustrates what an organization might expect to spend if deploying the system commercially as a restaurant management platform, while the student project itself required minimal financial investment.

Chapter 7.
Designing

The design phase translated the requirements into detailed technical specifications, focusing on usability, data flow, system architecture, and database integrity.

7.1 Interface Design

The User Interface (UI) follows a modern, mobile-first design language built with React.js and Tailwind CSS, acknowledging that most users will access the platform via smartphones and tablets. Key design principles include:

1. **Simplicity:** Clear navigation menus for Home, Menu, Cart, Reservations, Dashboard, and Profile with intuitive icons and labels.

2. **Consistency:** Uniform design across all modules using Tailwind CSS utility classes ensuring consistent fonts (Poppins), color scheme (primary gradient colors), buttons (rounded with hover effects), and spacing throughout the application.

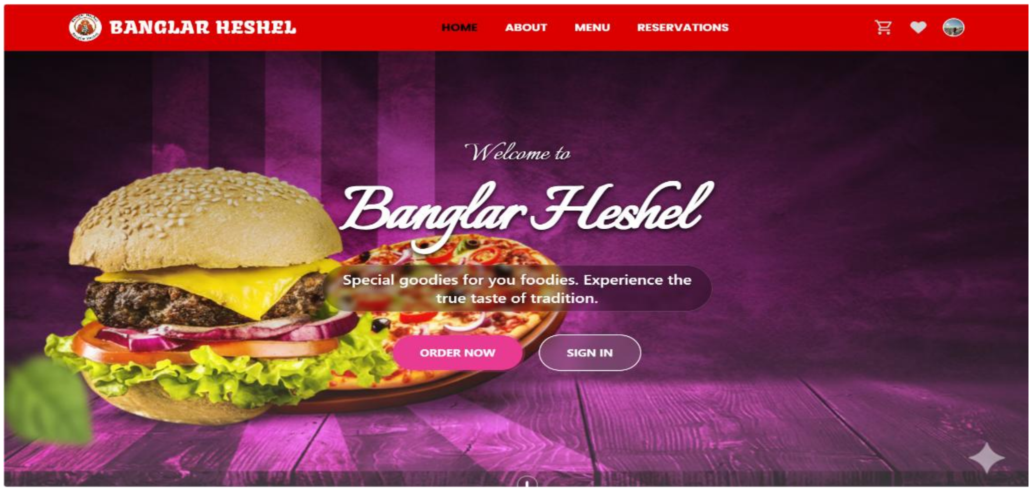
3. **Feedback:** Real-time confirmation messages using toast notifications (React Toastify, Notistack) after actions such as adding to cart, placing orders, making payments, submitting reservations, writing reviews, and updating stock levels.

4. **Accessibility:** Mobile-friendly responsive design adaptable to multiple screen sizes (320px to 1920px+) using Tailwind's responsive breakpoints, ensuring seamless experience across mobile phones, tablets, and desktop computers.

5. **Visual Hierarchy:** Strategic use of color gradients, card-based layouts, loading skeletons, and iconography (React Icons) to guide user attention and improve information comprehension.

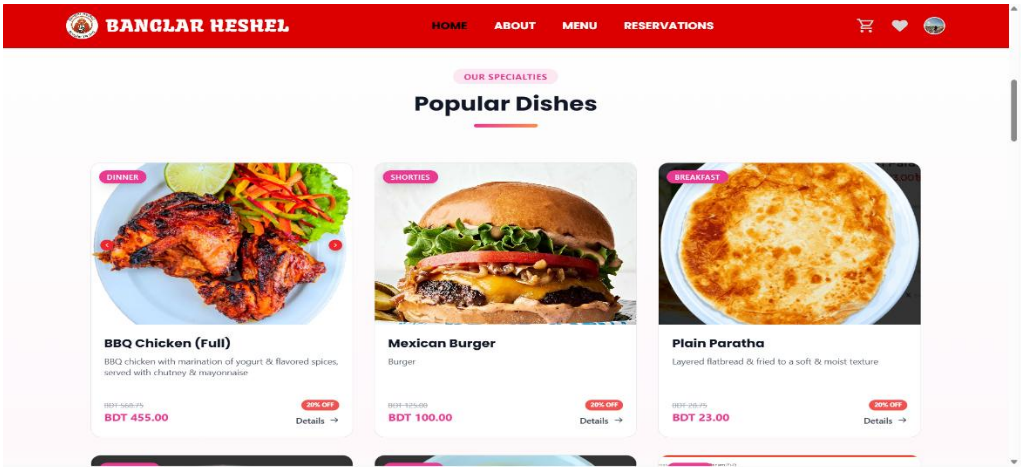
6. **Performance:** Optimized image loading with Cloudinary CDN, lazy loading for off-screen content, and efficient React component rendering to ensure fast page loads even on slower connections.

7.1.1 Non-Logged in User Features



Special goodies for you foodies. Experience the true taste of tradition.

Figure 7.1: Landing Page



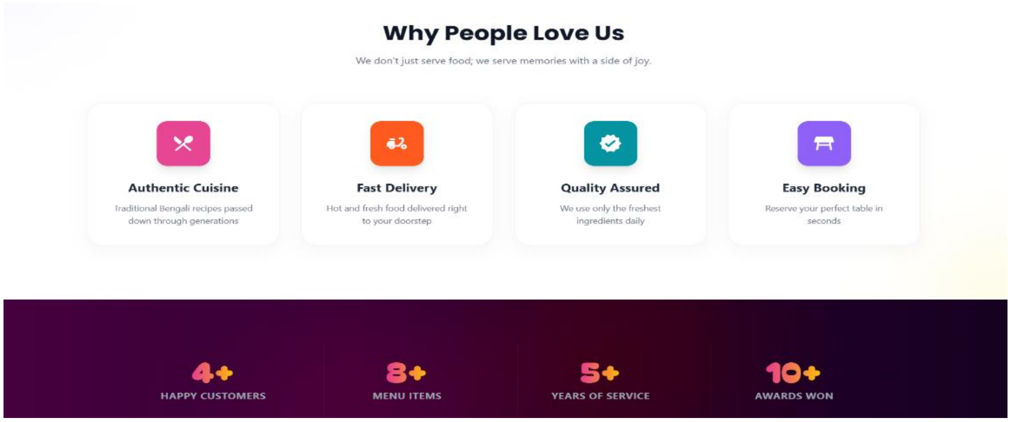
Popular Dishes

BBQ Chicken (Full)
BBQ chicken with marination of yogurt & flavored spices, served with chutney & mayonnaise
~~BDT 568.75~~
BDT 455.00

Mexican Burger
Burger
~~BDT 125.00~~
BDT 100.00

Plain Paratha
Layered flatbread & fried to a soft & moist texture
~~BDT 28.75~~
BDT 23.00

Figure 7.2: Popular Dishes



Why People Love Us

We don't just serve food; we serve memories with a side of joy.

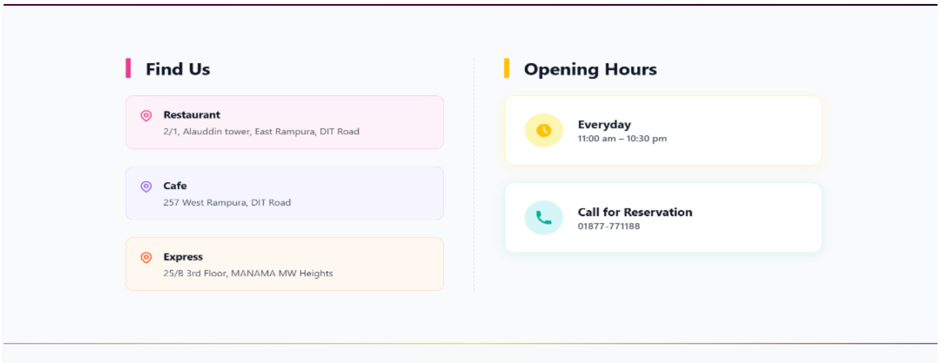
Authentic Cuisine
Traditional Bengali recipes passed down through generations

Fast Delivery
Hot and fresh food delivered right to your doorstep

Quality Assured
We use only the freshest ingredients daily

Easy Booking
Reserve your perfect table in seconds

Figure 7.3: Marketing



Find Us

Restaurant
2/1, Alauddin tower, East Rampura, DIT Road

Cafe
257 West Rampura, DIT Road

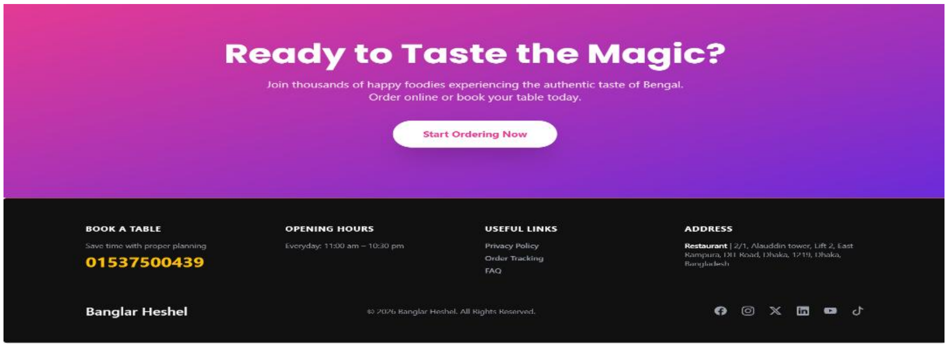
Express
25/B 3rd Floor, MANAMA MW Heights

Opening Hours

Everyday
11:00 am – 10:30 pm

Call for Reservation
01877-771188

Figure 7.4: Location & Contact Section



Ready to Taste the Magic?

Join thousands of happy foodies experiencing the authentic taste of Bengal.
Order online or book your table today.

BOOK A TABLE
Save time with proper planning
01537500439

OPENING HOURS
Everyday: 11:00 am – 10:30 pm

USEFUL LINKS
Privacy Policy
Order Tracking
FAQ

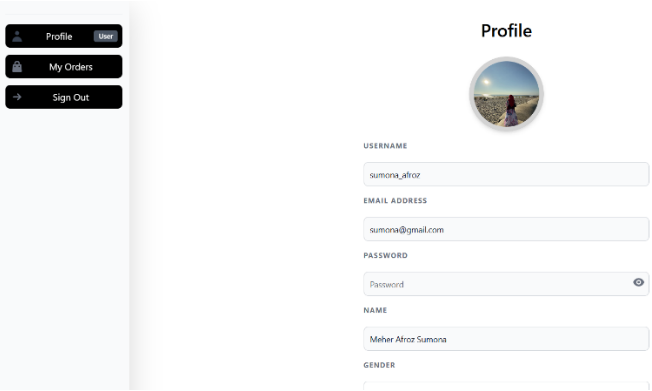
ADDRESS
Restaurant | 2/1, Alauddin tower, Lift 2, East Rampura, DIT Road, Dhaka, 1219, Dhaka, Bangladesh

Banglar Heshel

© 2026 Banglar Heshel. All Rights Reserved.

Figure 7.5: Footer Section

7.1.2 User Features



Profile

USERNAME
sumona_afroz

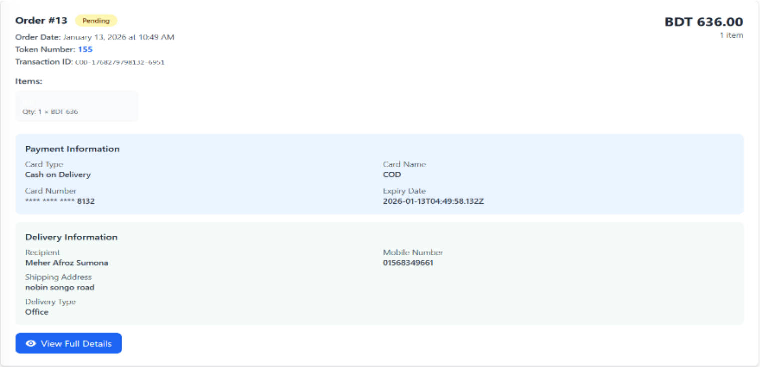
EMAIL ADDRESS
sumona@gmail.com

PASSWORD
Password

NAME
Meher Afroz Sumona

GENDER

Figure 7.6: user profile



Order #13 Pending

Order Date: January 13, 2026 at 10:49 AM
Token Number: 155
Transaction ID: COD-1768279798132-6951

Items:

Qty: 1 × BDT 636

BDT 636.00
1 item

Payment Information

Card Type
Cash on Delivery

Card Name
COD

Card Number
**** **** **** 8132

Expiry Date
2026-01-13T04:49:58.132Z

Delivery Information

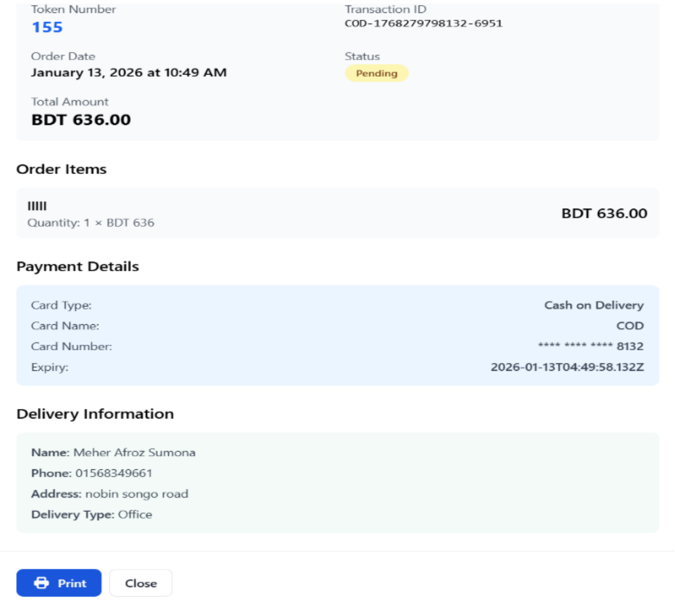
Recipient
Meher Afroz Sumona

Mobile Number
01568349661

Shipping Address
nobin songo road

Delivery Type
Office

Figure 7.7: user order record section



Token Number
155

Transaction ID
COD-1768279798132-6951

Order Date
January 13, 2026 at 10:49 AM

Status
Pending

Total Amount
BDT 636.00

Order Items

IIIII
Quantity: 1 × BDT 636 — BDT 636.00

Payment Details

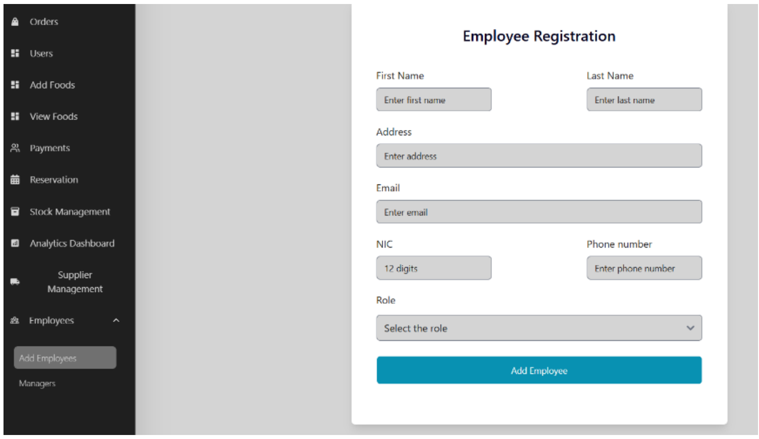
Card Type:	Cash on Delivery
Card Name:	COD
Card Number:	**** **** **** 8132
Expiry:	2026-01-13T04:49:58.132Z

Delivery Information

Name: Meher Afroz Sumona
Phone: 01568349661
Address: nobin songo road
Delivery Type: Office

Figure 7.8: user order details

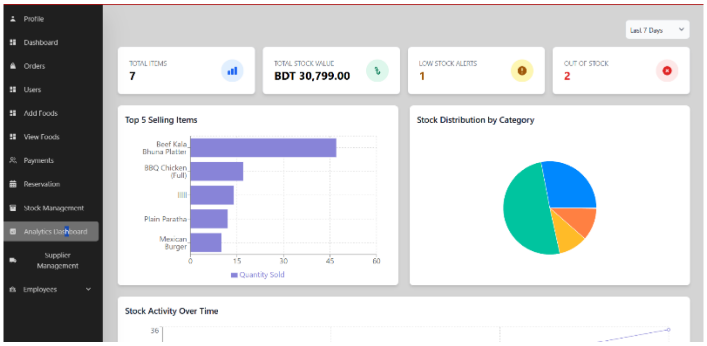
7.1.3 Admin Features



Employee Registration

First Name: Enter first name
Last Name: Enter last name
Address: Enter address
Email: Enter email
NIC: 12 digits
Phone number: Enter phone number
Role: Select the role

Figure 7.9: Add Employee



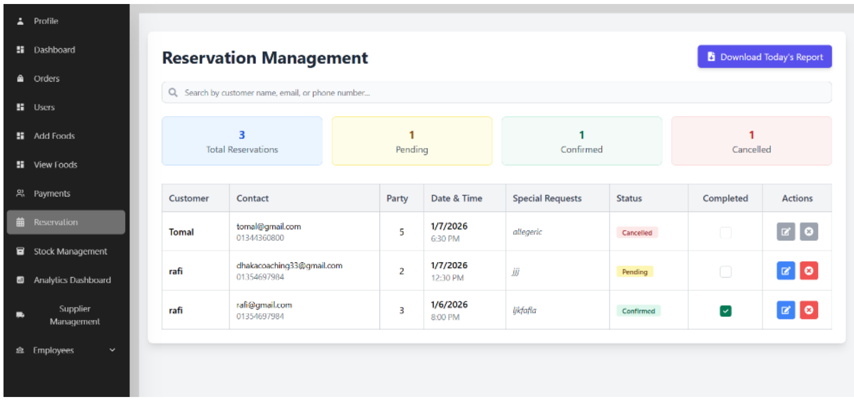
TOTAL ITEMS
7

TOTAL STOCK VALUE
BDT 30,799.00

LOW STOCK ALERTS
1

OUT OF STOCK
2

Figure 7.10: Analytics Dashboard



Reservation Management

3	1	1	1
Total Reservations	Pending	Confirmed	Cancelled

Customer	Contact	Party	Date & Time	Special Requests	Status	Completed	Actions
Tomal	tomal@gmail.com 01344360800	5	1/7/2026 6:30 PM	*allegeric*	Cancelled	□	
rafi	dhakacoaching33@gmail.com 01354697984	2	1/7/2026 12:30 PM	*jjj*	Pending	□	
rafi	rafi@gmail.com 01354697984	3	1/6/2026 8:00 PM	*ljkfafla*	Confirmed	☑	

Figure 7.11: Reservation Management

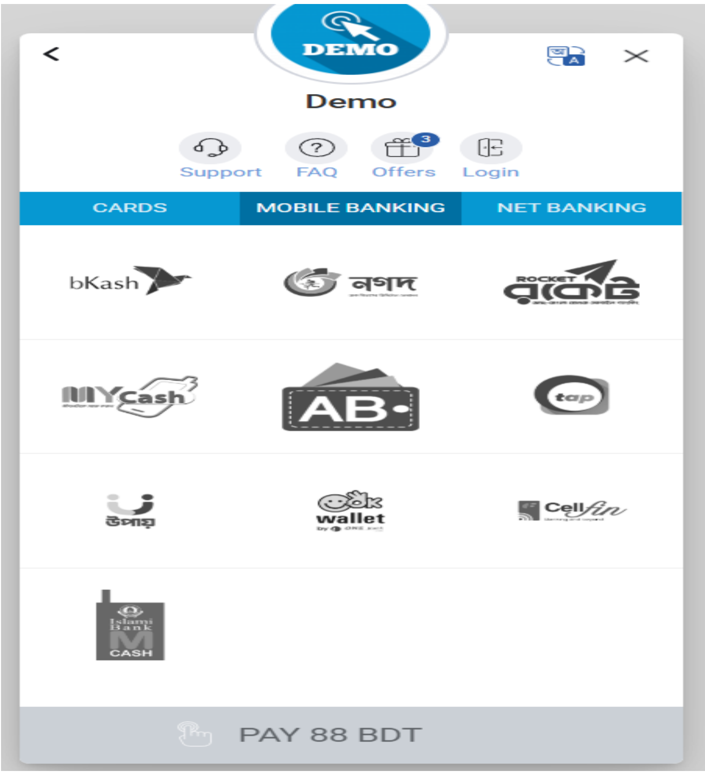
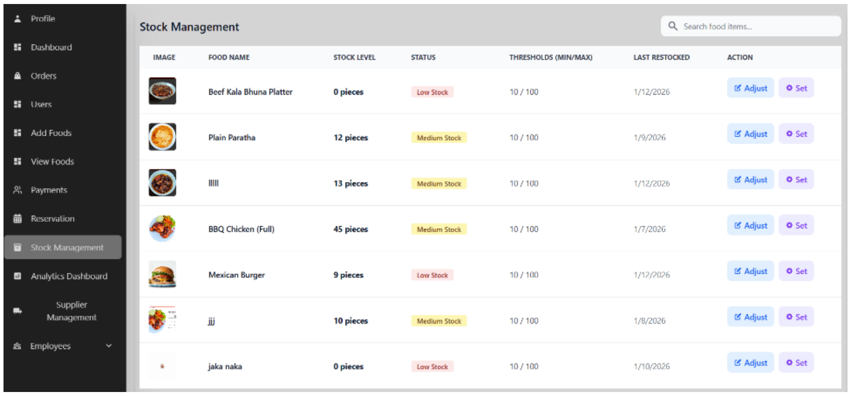


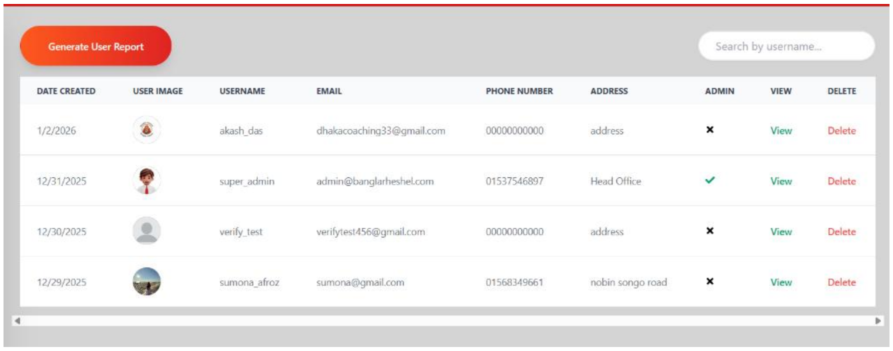
Figure 7.12: SSLCommerz Payment Gateway



Stock Management

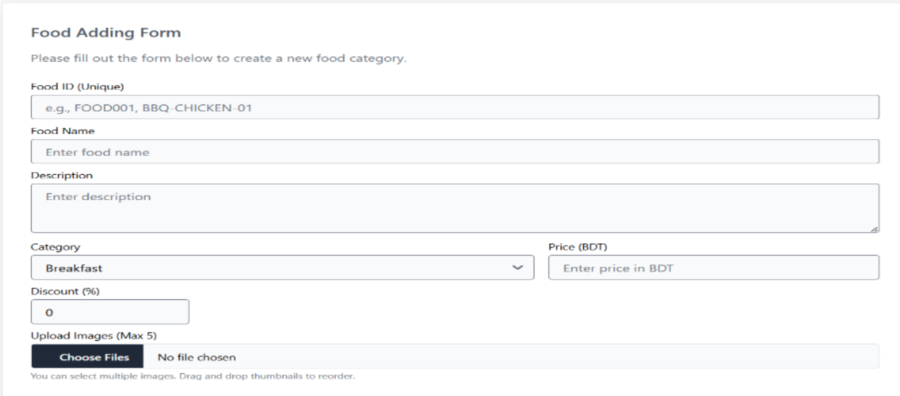
IMAGE	FOOD NAME	STOCK LEVEL	STATUS	THRESHOLDS (MIN/MAX)	LAST RESTOCKED	ACTION
	Beef Kala Bhuna Platter	0 pieces	Low Stock	10 / 100	1/12/2026	Adjust Set
	Plain Paratha	12 pieces	Medium Stock	10 / 100	1/9/2026	Adjust Set
	lllll	13 pieces	Medium Stock	10 / 100	1/12/2026	Adjust Set
	BBQ Chicken (Full)	45 pieces	Medium Stock	10 / 100	1/7/2026	Adjust Set
	Mexican Burger	9 pieces	Low Stock	10 / 100	1/12/2026	Adjust Set
	jjj	10 pieces	Medium Stock	10 / 100	1/8/2026	Adjust Set
	jaka naka	0 pieces	Low Stock	10 / 100	1/10/2026	Adjust Set

Figure 7.13: Stock Management



DATE CREATED	USER IMAGE	USERNAME	EMAIL	PHONE NUMBER	ADDRESS	ADMIN	VIEW	DELETE
1/2/2026		akash_das	dhakacoaching33@gmail.com	00000000000	address	✕	View	Delete
12/31/2025		super_admin	admin@banglarheshel.com	01537546897	Head Office	✓	View	Delete
12/30/2025		verify_test	verifytest456@gmail.com	00000000000	address	✕	View	Delete
12/29/2025		sumona_afroz	sumona@gmail.com	01568349661	nobin songo road	✕	View	Delete

Figure 7.14: User Dashboard



Food Adding Form

Please fill out the form below to create a new food category.

Food ID (Unique): e.g., FOOD001, BBQ-CHICKEN-01

Food Name: Enter food name

Description: Enter description

Category: Breakfast

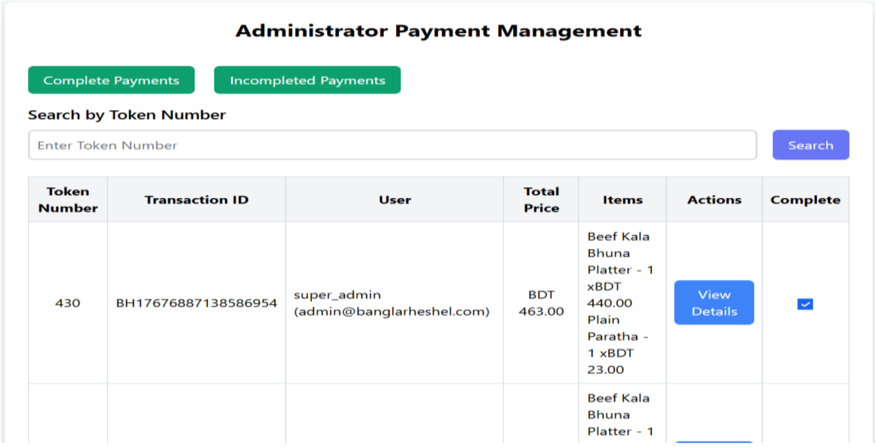
Price (BDT): Enter price in BDT

Discount (%): 0

Upload Images (Max 5)

You can select multiple images. Drag and drop thumbnails to reorder.

Figure 7.15: Food Adding Form

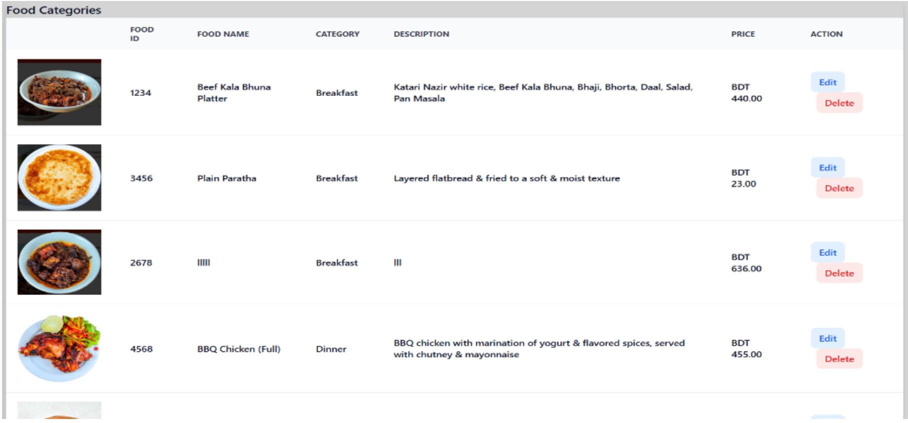


Administrator Payment Management

Search by Token Number

Token Number	Transaction ID	User	Total Price	Items	Actions	Complete
430	BH17676887138586954	super_admin (admin@banglarheshel.com)	BDT 463.00	Beef Kala Bhuna Platter - 1 xBDT 440.00 Plain Paratha - 1 xBDT 23.00	View Details	☑
				Beef Kala Bhuna Platter - 1		

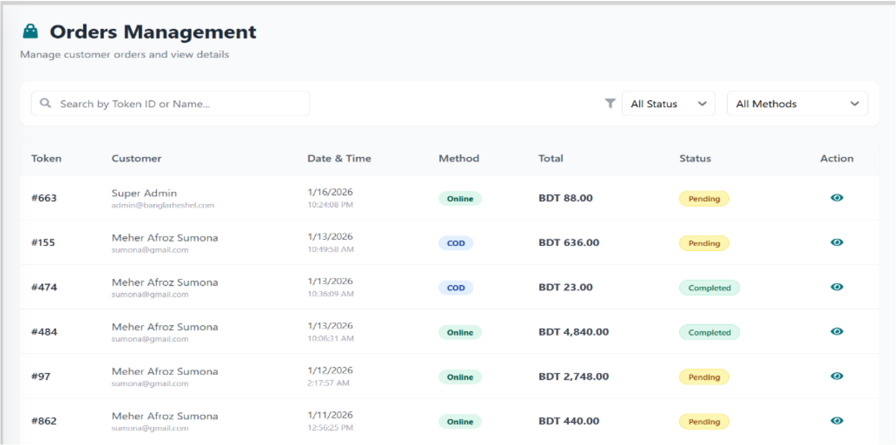
Figure 7.16: Payment Management



Food Categories

	FOOD ID	FOOD NAME	CATEGORY	DESCRIPTION	PRICE	ACTION
	1234	Beef Kala Bhuna Platter	Breakfast	Katari Nazir white rice, Beef Kala Bhuna, Bhaji, Bhorta, Daal, Salad, Pan Masala	BDT 440.00	Edit Delete
	3456	Plain Paratha	Breakfast	Layered flatbread & fried to a soft & moist texture	BDT 23.00	Edit Delete
	2678	IIIII	Breakfast	III	BDT 636.00	Edit Delete
	4568	BBQ Chicken (Full)	Dinner	BBQ chicken with marination of yogurt & flavored spices, served with chutney & mayonnaise	BDT 455.00	Edit Delete

Figure 7.17: Food CRUD System



Orders Management

Manage customer orders and view details

Token	Customer	Date & Time	Method	Total	Status	Action
#663	Super Admin admin@banglarheshel.com	1/16/2026 10:24:08 PM	Online	BDT 88.00	Pending	👁
#155	Meher Afroz Sumona sumona@gmail.com	1/13/2026 10:49:58 AM	COD	BDT 636.00	Pending	👁
#474	Meher Afroz Sumona sumona@gmail.com	1/13/2026 10:36:09 AM	COD	BDT 23.00	Completed	👁
#484	Meher Afroz Sumona sumona@gmail.com	1/13/2026 10:06:31 AM	Online	BDT 4,840.00	Completed	👁
#97	Meher Afroz Sumona sumona@gmail.com	1/12/2026 2:17:57 AM	Online	BDT 2,748.00	Pending	👁
#862	Meher Afroz Sumona sumona@gmail.com	1/11/2026 12:56:25 PM	Online	BDT 440.00	Pending	👁

Figure 7.18: Order Management

7.2 Data Flow Diagram (DFD)

A **Data Flow Diagram (DFD)** is a graphical representation used to illustrate how data moves through an information system. It focuses on the flow of data between different system components rather than on program logic or technical implementation. DFDs help analysts and developers understand how data is received, processed, stored, and produced as output within the Banglar Heshel restaurant management system.

A Data Flow Diagram is composed of four main elements: external entities, processes, data stores, and data flows. External entities represent users (customers, managers, administrators) and external systems (SSLCommerz, Cloudinary, Nodemailer, Firebase) that interact with the system. Processes define the activities that transform input data into meaningful output, such as user authentication, order processing, payment handling, and inventory management. Data stores are MongoDB collections where data is saved, including users, food items, orders, reservations, reviews, and stock. Data flows show the direction and type of data moving between entities, processes, and data stores.

DFDs are developed at multiple levels to provide increasing detail. The Context Diagram (Level 0) presents the entire system as a single process and shows its interaction with external entities. Level 1 and Level 2 DFDs further decompose the system into smaller, more detailed processes, making the complex restaurant management workflows easier to understand.

Overall, Data Flow Diagrams are essential for system analysis and design, as they improve clarity, support requirement validation, and enhance communication among stakeholders throughout the development lifecycle.

7.2.1 Context-Level DFD (Level 0)

The Level 0 DFD represents the entire Bangler Heshel system as a single process interacting with external entities.

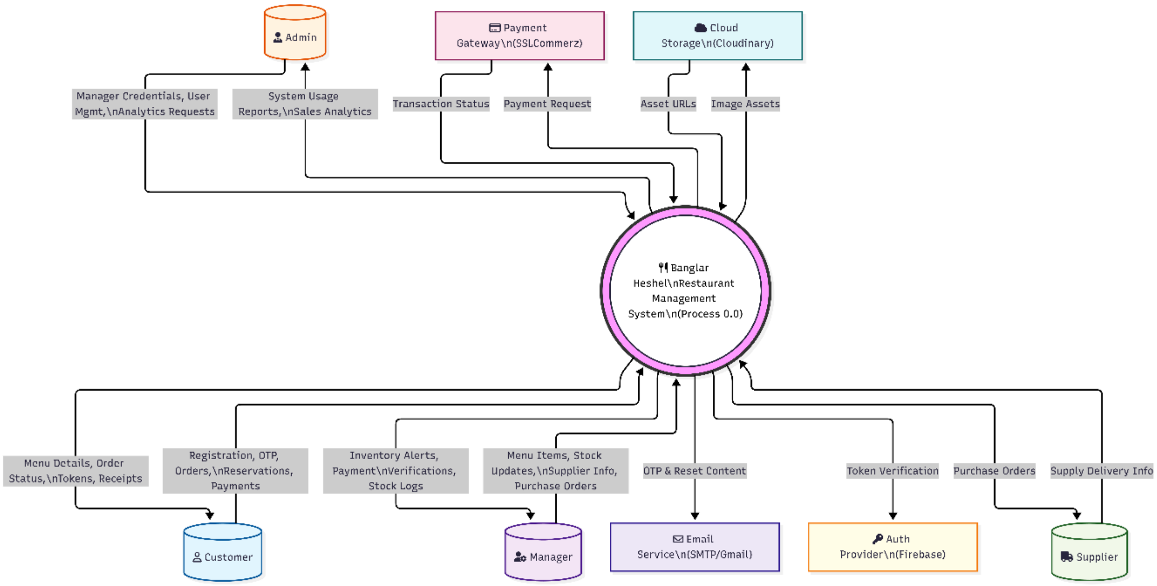

Admin
Payment Gateway\n(SSLCommerz)
Cloud Storage\n(Cloudinary)
Manager Credentials, User Mgmt,\nAnalytics Requests
System Usage Reports,\nSales Analytics
Transaction Status
Payment Request
Asset URLs
Image Assets
Banglar Heshel\nRestaurant Management System\n(Process 0.0)
Menu Details, Order Status,\nTokens, Receipts
Registration, OTP, Orders,\nReservations, Payments
Inventory Alerts, Payment\nVerifications, Stock Logs
Menu Items, Stock Updates,\nSupplier Info, Purchase Orders
OTP & Reset Content
Token Verification
Purchase Orders
Supply Delivery Info
Customer
Manager
Email Service\n(SMTP/Gmail)
Auth Provider\n(Firebase)
Supplier

Figure 7.19: Context Level DFD (Level 0)

- **External Entities:**

  (a) Customer (User):

  (i) Sends: Registration data, login credentials, food search queries, orders, payments, reservations, reviews.

  (ii) Receives: Menu listings, order confirmations, payment receipts, reservation confirmations, email notifications.

  (b) Manager:

  (i) Sends: Food item details, images, stock updates, supplier data, reservation status updates.

  (ii) Receives: Payment reports, order history, low stock alerts, analytics dashboards, reservation lists.

  (c) Administrator:

  (i) Sends: User management commands (block/unblock), manager assignments, role permissions.

  (ii) Receives: System reports, user statistics, revenue summaries, platform usage data.

(d) Payment Gateway (SSLCommerz):

(i) Receives: Payment requests with transaction details**.**

(ii) Sends: Transaction status (success/failure), payment confirmations, transaction IDs**.**

(e) Image Storage Service (Cloudinary):

(i) Receives: Food images, profile pictures**.**

(ii) Sends: CDN URLs, optimized images, upload confirmations**.**

(f) Email Service (Nodemailer):

(i) Receives: OTP codes, order confirmations, reservation details, password reset links**.**

(ii) Sends: Email delivery status**.**

(g) Authentication Service (Firebase):

(i) Receives: Google OAuth login requests**.**

(ii) **Sends:** Authentication tokens, user profile data.

7.2.2 DFD Level 1

The Level 1 DFD decomposes the main system into its primary sub-processes:

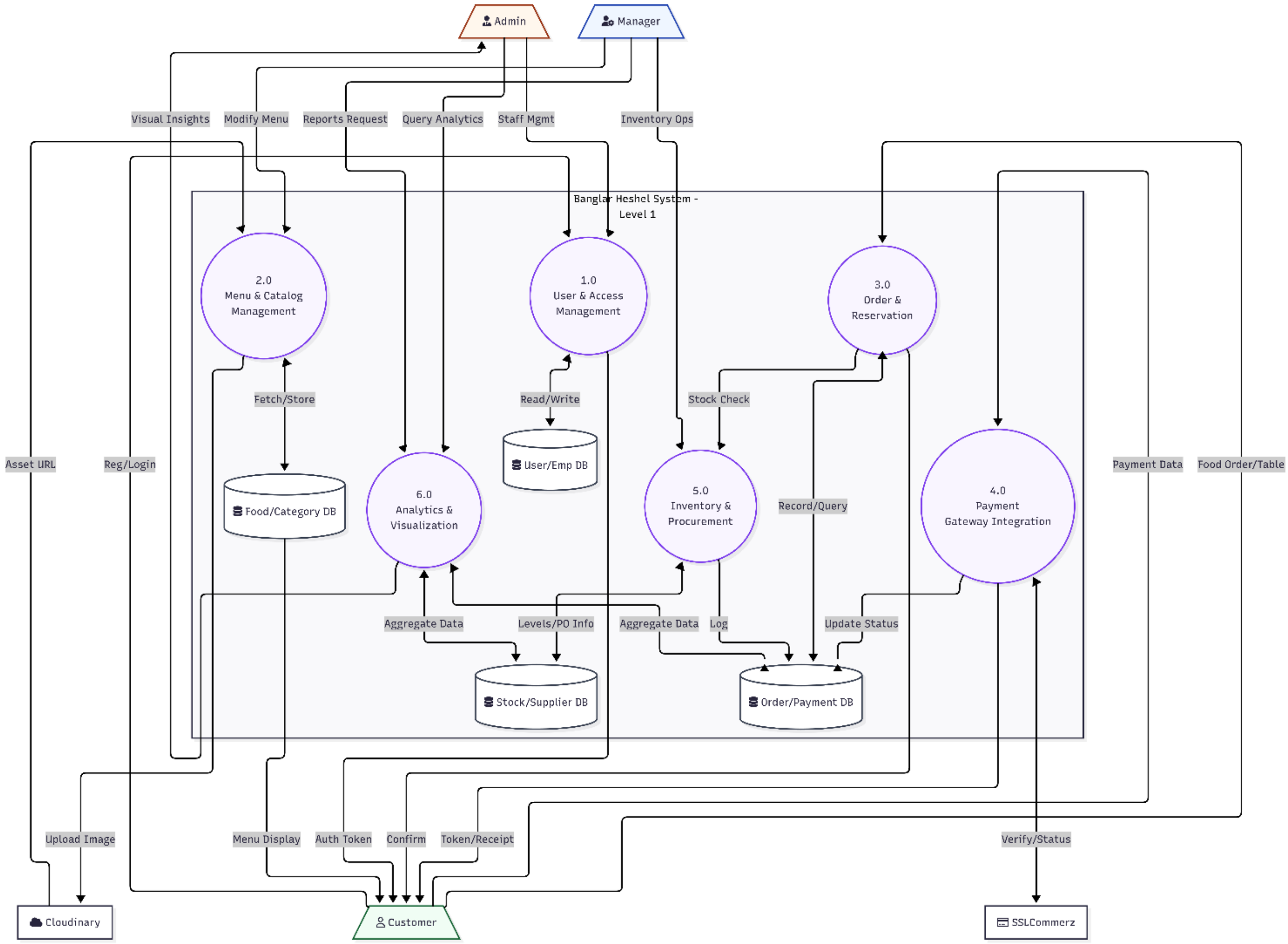


Figure 7.20: Level 1 DFD

1. **User Management:** Handles user registration with OTP verification, login authentication with JWT tokens, password reset, profile management, and maintenance of user and employee records.
2. **Food & Menu Management:** Manages food item listings with multi-image uploads, category organization, menu browsing, search queries by food name, filtering by category, and food details display.
3. **Order Management:** Handles shopping cart operations, wishlist management, order placement requests, payment processing integration, order confirmation with token generation, and order history tracking.
4. **Payment Processing:** Processes payment information through SSLCommerz gateway, handles payment redirects, receives transaction callbacks, generates PDF receipts, and records payment status in the database.
5. **Reservation Management:** Manages table reservation requests with date/time validation, reservation status updates (Pending/Confirmed/Cancelled), email confirmations, and generates daily reservation reports.

6. **Review & Rating System:** Manages user reviews and ratings (1-5 stars) after order completion, stores feedback data with timestamps, calculates average ratings, and handles review deletion.
7. **Inventory & Stock Management:** Oversees stock level tracking, automated stock entry creation for new food items, low stock alerts when quantity falls below minimum threshold, and stock update operations.
8. **Supplier & Purchase Order Management:** Manages supplier information, creates and tracks purchase orders, monitors order status (Pending/Completed/Cancelled), and maintains supplier performance ratings.
9. **Notification Processing:** Generates and sends OTP verification emails, order confirmations, payment receipts, reservation confirmations, low stock alerts, and password reset emails through Nodemailer service.
10. **Admin Management:** Oversees system administration including user management (block/unblock), manager/employee assignments, role-based access control, system analytics, and comprehensive reporting.

7.2.3 DFD Level 2

P2.1 User Management:

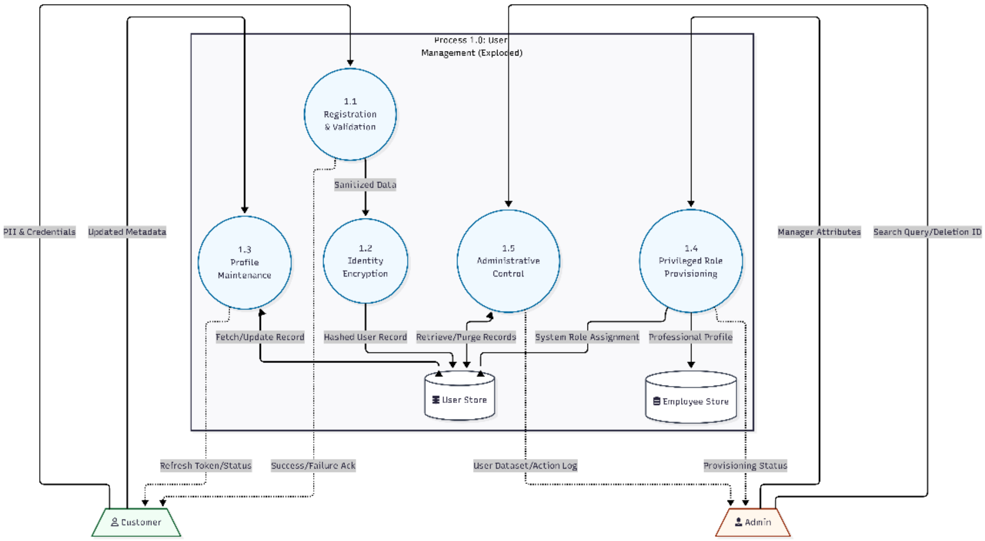


Figure 7.21: DFD Level 2 Process 1

P2.2 Menu Management:

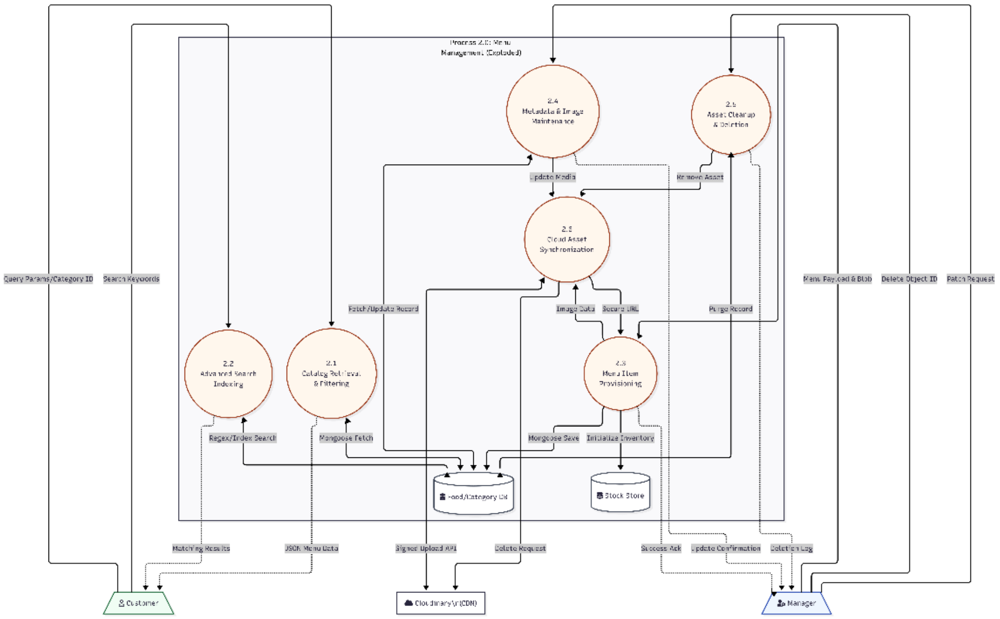


Figure 7.22: DFD Level 2 Process 2

P2.3 Order Management:

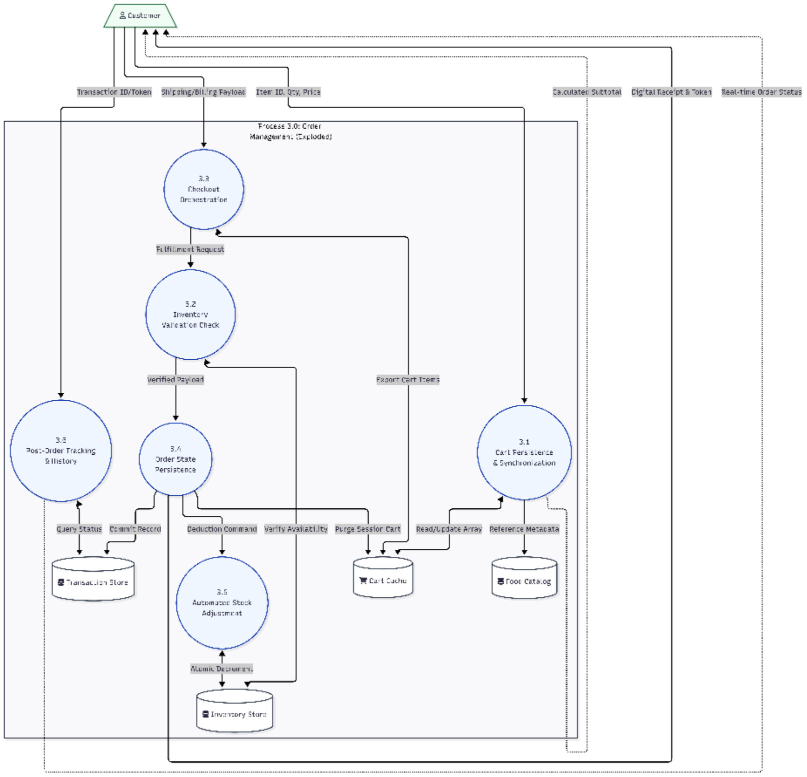


Figure 7.23: DFD Level 2 Process 3

P2.4 Payment Processing:

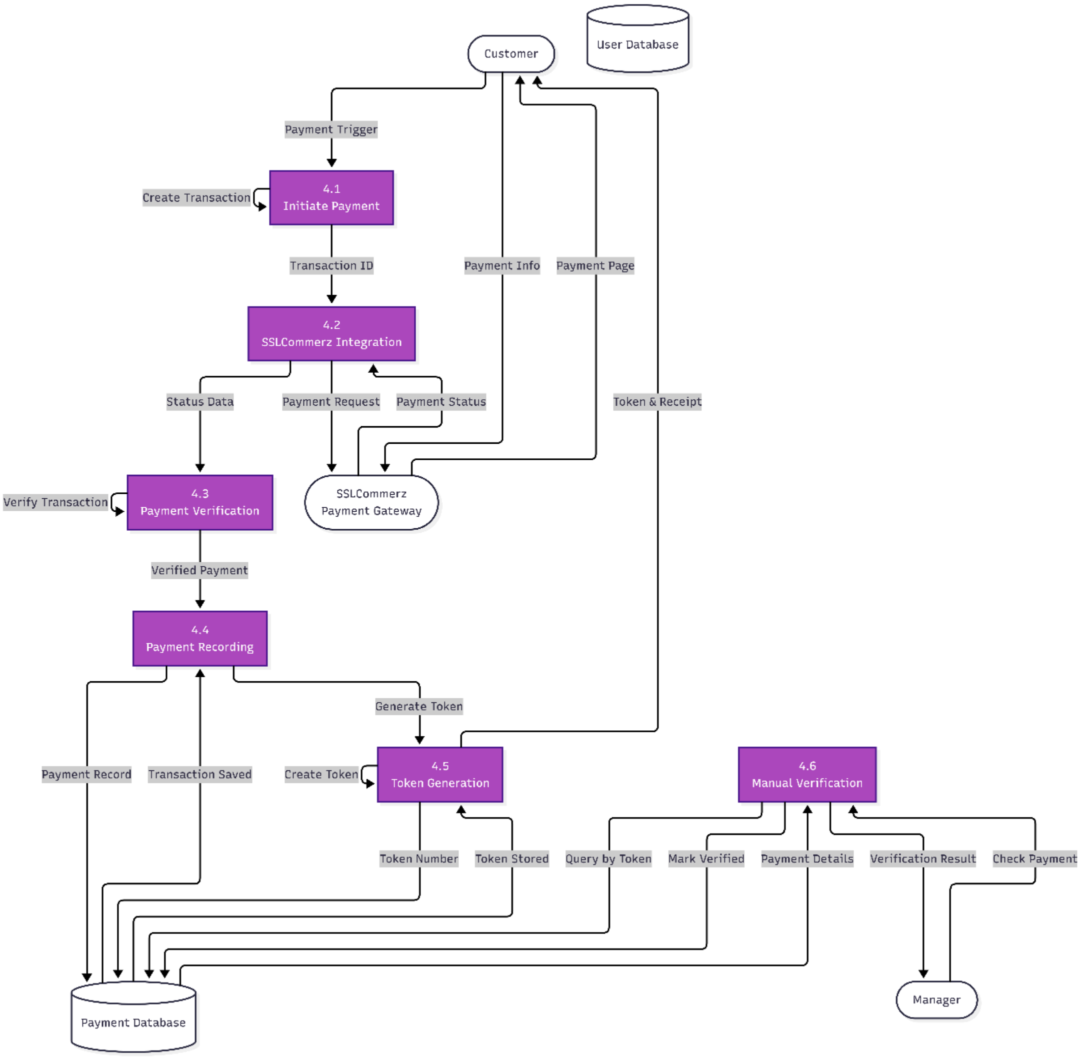


Figure 7.24: DFD Level 2 Process 4

P2.5 Stock management:

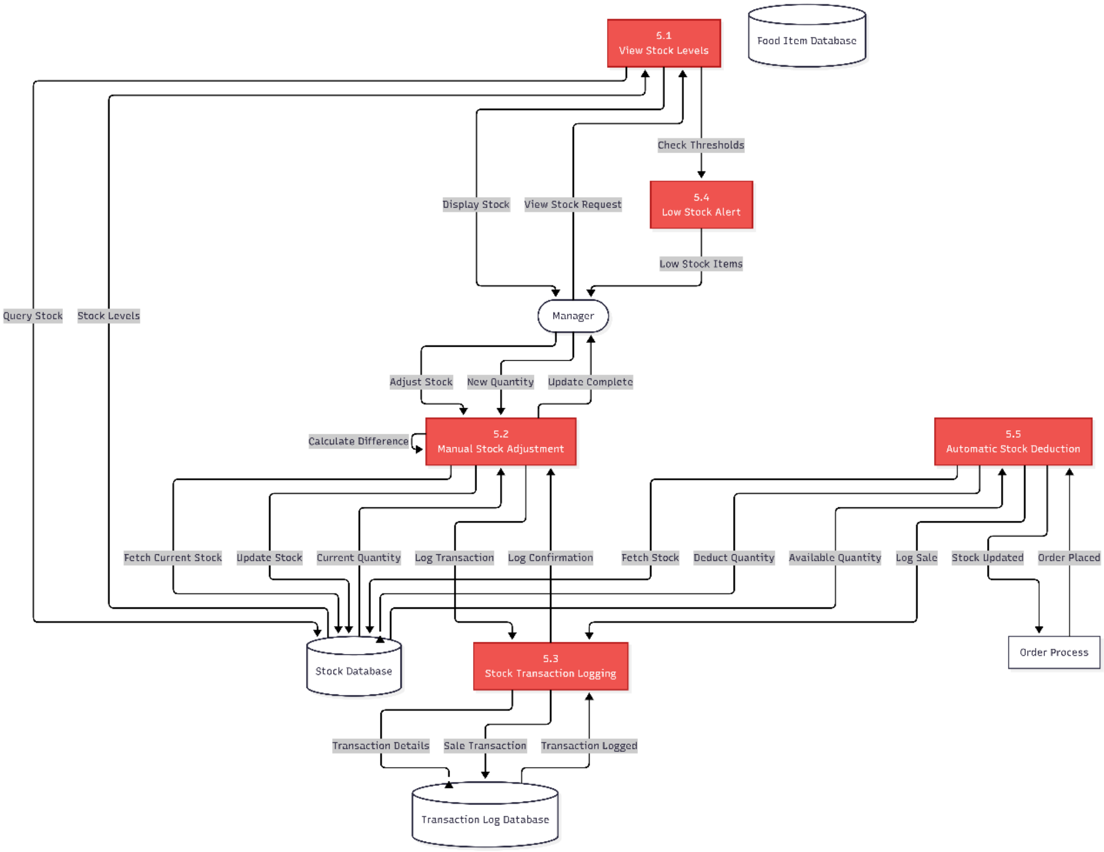


Figure 7.25: DFD Level 2 Process 5

P2.6 Authentication & Authorization:

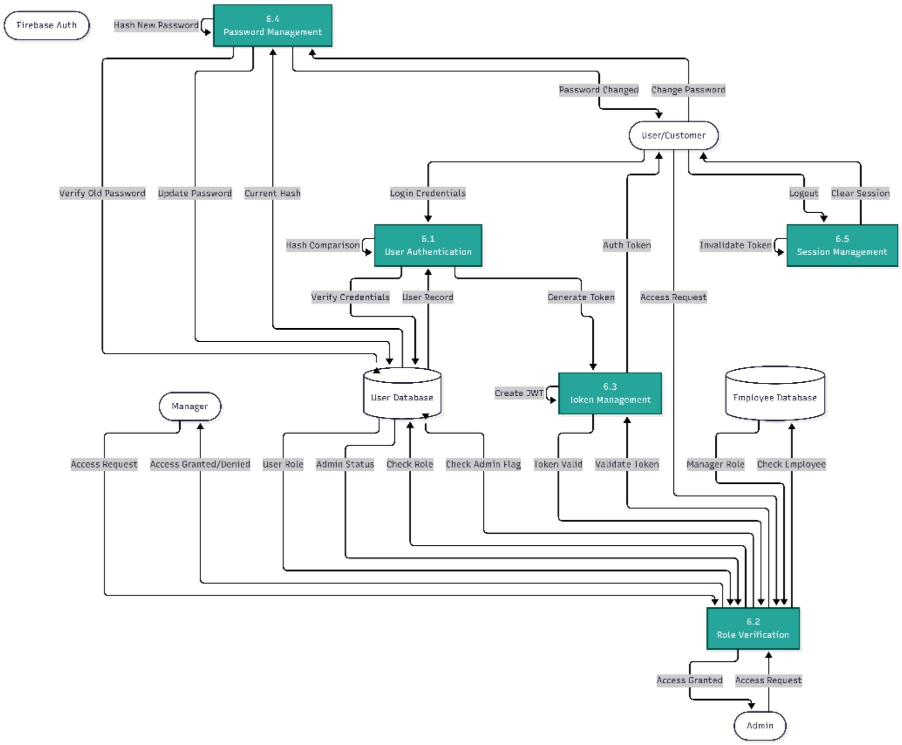


Figure 7.26: DFD Level 2 Process 6

P2.7 Analytics & Reporting:

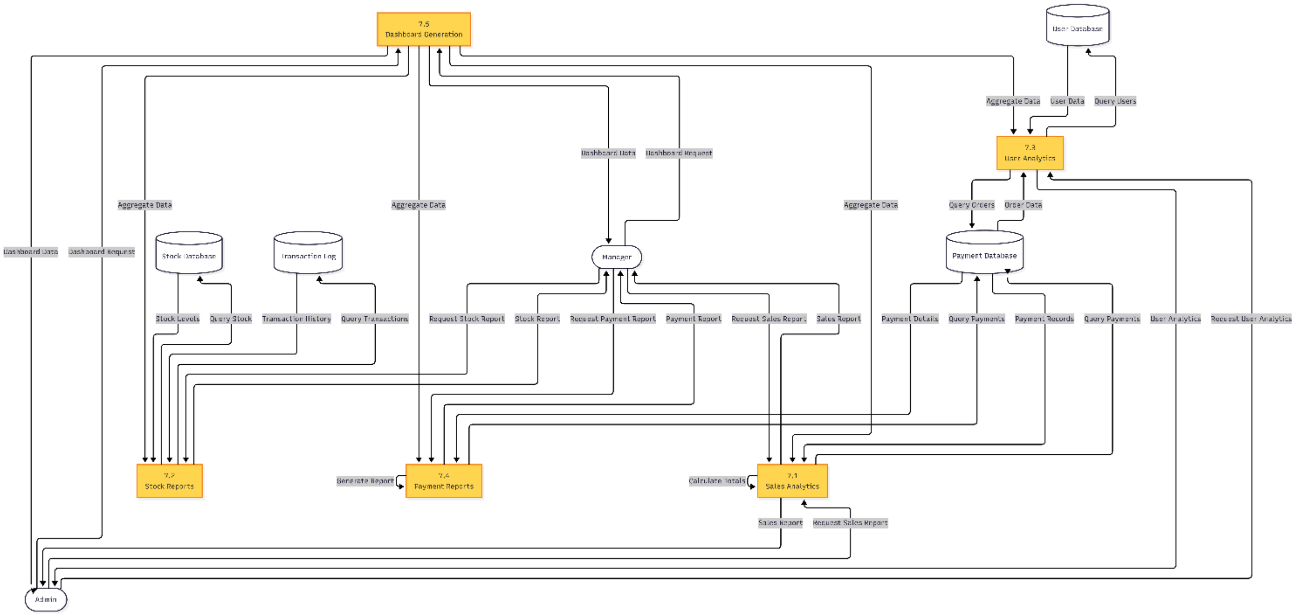


Figure 7.27: DFD Level 2 Process 7

P2.8 Inventory & Procurement Management:

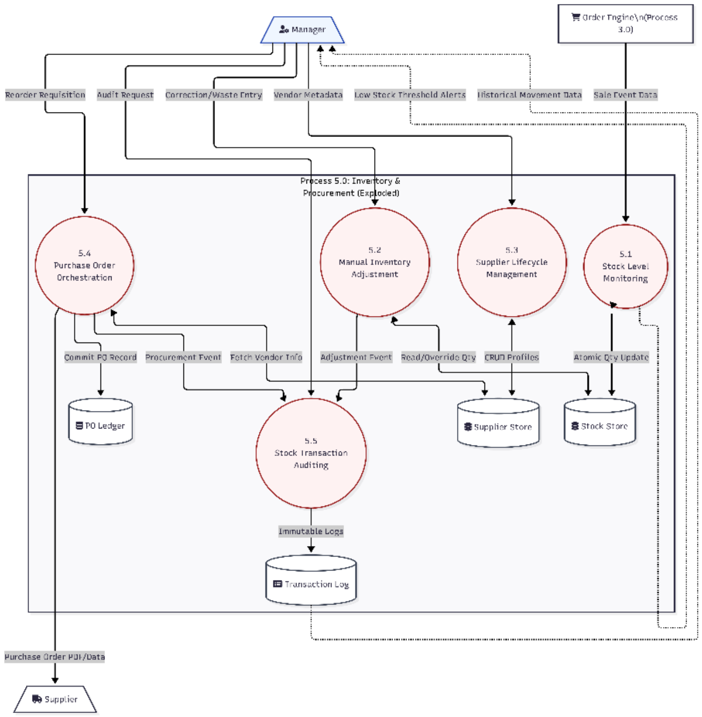


Figure 7.28: DFD Level 2 Process 8

7.3 Database Design

Database design plays a critical role in ensuring that the Banglar Heshel Restaurant Management System is efficient, scalable, and reliable. The database is designed to manage users, managers, administrators, food items, orders, payments, reservations, reviews, stock inventory, suppliers, and purchase orders. A NoSQL document-based approach using MongoDB is adopted to provide flexibility, horizontal scalability, and efficient handling of nested data structures such as cart items and image arrays.

The design supports core system functionalities including menu management, online ordering, payment processing with token generation, table reservations, review and rating system, inventory control with automated alerts, supplier management, and multi-role access control.

7.3.1 Data Organization and Schema Design

While MongoDB is a NoSQL database that doesn't enforce strict normalization, the Banglar Heshel database follows logical data organization principles to ensure consistency and minimize redundancy.

a. **Atomic Data Storage**

   i. All collections contain atomic values in their fields.

   ii. No repeating groups or multivalued attributes exist within single fields.

   iii. **Example:** Food images are stored as arrays of URLs (images: [url1, url2, ...]) rather than concatenated strings, and cart items are stored as embedded documents within orders.

b. **Logical Data Independence**

   i. Related data is stored in separate collections with references (foreign keys).

   ii. Non-key attributes depend entirely on the document's primary identifier (_id).

   iii. **Example:** In the Order collection, attributes such as tokenNumber, totalPrice, and isChecked depend entirely on the order's _id.

c. **Reduced Data Duplication**

   i. Core entity data is stored only in its primary collection and referenced elsewhere.

   ii. **Example:** User profile details (username, email, profilePicture) are stored only in the User collection. Orders and reviews reference the userId rather than duplicating user information.

This data organization approach improves data integrity, reduces update anomalies through Mongoose schema validation, and enhances database maintainability while leveraging MongoDB's flexibility for embedded documents where appropriate (such as cart items within orders)

7.3.2 Entity Relationship (ER) Model

The ER model of the Banglar Heshel system defines key collections and their relationships as follows:

Core Collections:

1. **User**: Represents customers who browse menu, place orders, make reservations, and submit reviews. Stores personal information (username, email, password hash), profile picture with Cloudinary URLs, authentication data (OTP, otpExpiry, isVerified), and role information (isAdmin, role).

2. **Employee**: Represents managers and staff members. Stores professional information (firstname, lastname, username, email, password hash), role designation (Manager/Employee), contact details (phone, NIC, address), and profile picture.

3. **FoodItem**: Represents menu items offered by the restaurant. Each food item belongs to a specific category and contains details including foodId (unique identifier), foodName, description, category, pricing (price, oldPrice, discount), and up to 6 images stored as Cloudinary URLs with corresponding public IDs for cleanup.

4. **Category**: Defines food categories under which menu items are grouped (Main Course, Appetizers, Desserts, Beverages, etc.).

5. **Order**: Represents customer orders placed through the system. Links a user with their selected cart items (embedded documents containing foodId, foodName, quantity, price, image), payment information from SSLCommerz, unique token number for tracking, total price calculation, and order status (isChecked).

6. **Reservation**: Stores table booking requests made by customers. Contains customer contact information (customerName, email, phone), reservation details (date, time, partySize), and status tracking (Pending, Confirmed, Cancelled).

7. **Review**: Stores customer feedback and ratings for food items. Links userId and foodId with rating (1-5 stars), text comment, username for display, and timestamp for ordering.

8. **Stock**: Manages inventory levels for food items. Each stock entry is linked to a foodId and contains quantity, unit type (pieces, kg, liters), minimum threshold for alerts, maximum threshold for ordering, and lastUpdated timestamp.

9. **Supplier**: Stores restaurant supplier information including companyName, contactPerson, contact details (email, phone, address), and performance rating for vendor management.

10. **PurchaseOrder**: Records supply orders placed with suppliers. Links supplierId with product details (productName, quantity, unitPrice, totalAmount), order date, and status tracking (Pending, Completed, Cancelled).

Relationships:

The MongoDB collections maintain relationships through referenced ObjectIds:

1. **One User to Many Orders:** A single user can place multiple orders over time.

2. **One User to Many Reviews:** A user can submit reviews for multiple food items.

3. **One FoodItem to Many Reviews:** Each food item can receive multiple customer reviews.

4. **One FoodItem to One Stock:** Each food item has exactly one stock record for inventory tracking.

5. **One Order contains Many CartItems:** Orders embed an array of cart item documents.

6. **One Supplier to Many PurchaseOrders:** Suppliers can receive multiple purchase orders.

7. **Many FoodItems to One Category:** Food items are grouped under categories.

8. **External Service Integrations:**

    a. **SSLCommerz Payment Gateway:** Referenced in Order collection through paymentInfo object containing transaction details.

    b. **Cloudinary Image Storage:** Referenced in FoodItem and User collections through image URL arrays and public ID tracking.

    c. **Nodemailer Email Service:** Triggered by system events (registration, orders, reservations) but not stored as a collection.

    d. **Firebase Authentication:** Used for Google OAuth but user data is stored in User collection

7.3.3 Database Table Structure

The following tables represent the core schema of the system:

Table 7.1: User registration Table

Field Name	Data Type	Constraint	Description
username	String	Unique, Required	A unique identifier for the user to log into the system.
email	String	Unique, Required	The user's email address, used for communication and OTP verification.
password	String	Required	The user's security credential: it is hashed using Bcrypt before storage for security.
profilePicture	String	Default Value	A URL link to the user's avatar, stored as a string.
isAdmin	Boolean	Default: false	A flag identifying if the user has administrative privileges.
isVerified	Boolean	Default: false	Tracks whether the user has successfully completed the email/OTP verification process.

Table 7.2: Employee Registration table

Field Name	Data Type	Description	Constraints
_id	ObjectId	Unique system-generated identifier for the user record.	Primary Key, Auto-generated
firstname	String	User's specific first name.	required
lastname	String	User's specific last name.	Optional
username	String	Unique alphanumeric identifier used for system login.	Required, Unique
email	String	User's electronic mail address is used for communication and OTP.	Required, Unique

password	String	Security credentials stored as a one-way hash for protection.	Required, Hashed (Bcrypt)
gender	String	Identified gender of the user.	Optional
dateOfBirth	Date	User's birth date is used for age verification or records.	Date Format (ISO 8601)
nic	String	National Identity Card number for formal identification.	Optional, Unique
address	String	Physical residential or business location of the user.	Optional
phone	String	Secondary or alternative contact number.	Optional
contactNumber	String	Primary mobile or landline phone number.	Optional
profilePicture	String	Secure URL pointing to the user's avatar image.	Optional, Hosted on Cloudinary
isAdmin	Boolean	Authorization flag granting full system access.	Default: false

Table 7.3: Fooditems table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the food record.	Primary Key, Auto-generated
foodId	String	A specific unique identifier for the food item.	Required, Unique
foodName	String	The official name of the food item or category.	Required
description	String	Detailed information regarding ingredients and preparation.	Required
category	String	Classification for filtering (e.g., Appetizer, Main Course).	Required
price	Int32	The current selling price of the item in BDT.	Required, Minimum: 0

oldPrice	Double	The original price of the item before any discount was applied.	Optional
discount	Int32	The specific amount or percentage deducted from the old price.	Optional, Default: 0
images	Array	A collection of secure URL strings pointing to Cloudinary assets.	Required

Table 7.4: Payment Table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the order record.	Primary Key, auto generated
userId	ObjectId	Reference to the User model identifying the customer.	Required, Foreign Key
transactionId	String	Unique identifier for the financial transaction (SSLCommerz tran_id).	Required, Unique
cartItems	Array	Collection of food items including foodId, quantity, and price.	Required
totalPrice	Int32	The final total cost of the order in BDT.	Required, Minimum: 0
paymentInfo	Object	Detailed object containing payment status and method details.	Required
shipmentData	Object	Object containing shipping address, city, and postal code.	Required for delivery orders
tokenNumber	Int32	Secure alphanumeric code for order pickup or identification.	Unique, Required
isChecked	Boolean	Flag used by managers to verify and process the order.	Default: false

Table 7.5: Reservation Table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the reservation record.	Primary Key, auto generated
customerName	String	Full name of the individual making the booking.	Required
email	String	Customer's electronic mail address for booking confirmation.	Required, Valid Email Format
phoneNumber	String	Primary contact number for the reservation.	Required
partySize	Int32	Total number of guests for table booking.	Required, Minimum: 1
reservationDate	Date	The scheduled calendar date for the dining visit.	Required
reservationTime	String	The specific time slot was selected for the reservation.	Required

specialRequests	String	Optional notes or requirements provided by the customer.	Optional

Table 7.6: Review Table

Field Name	**Data Type**	**Description**	**Constraints**
foodId	ObjectId	Reference to the FoodItem being reviewed.	Required, Foreign Key
userId	ObjectId	Reference to the User who authored the review.	Required, Foreign Key
username	String	Display name of the reviewer at the time of posting.	Required
email	String	Email address of the reviewer for verification or contact.	Required, Valid Email Format
rating	Int32	Numeric score representing user satisfaction (typically 1-5).	Required, Range: 1-5
comment	String	Detailed textual feedback provided by the customer.	Required

Table 7.7: Purchase Table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the purchase order record.	Primary Key, Auto-generated
orderNumber	String	Unique tracking number for the purchase order (e.g., PO-2026-001).	Required, Unique
supplierId	ObjectId	Reference to the Supplier model identifying the vendor.	Required, Foreign Key
supplierName	String	Denormalized name of the supplier for faster reporting.	Required
items	Array	Collection of objects containing itemName, quantity, and unitPrice.	Required
subtotal	Int32	Total cost of all items before applying taxes.	Required, Minimum: 0
tax	Int32	Value-added tax or other applicable charges.	Default: 0
totalAmount	Int32	Final calculation of subtotal + tax.	Required
orderDate	Date	The specific date the purchase order was issued.	Required

Table 7.8: Stock Table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the inventory record.	Primary Key, Auto-generated
foodId	ObjectId	Reference to the FoodItem model; links stock to a specific menu item.	Required, Foreign Key

quantity	Int32	The current number of units physically available in the restaurant's stock.	Required, Minimum: 0
unit	String	The unit of measurement (e.g., 'kg', 'pcs', 'liters', 'servings').	Required
minThreshold	Int32	The lower limit that triggers a "Low Stock" alert for management.	Default: 10
maxThreshold	Int32	The target capacity to prevent overstocking and reduce waste.	Optional
costPerUnit	Int32	The purchase price paid for a single unit of the item.	Required for COGS calculation
lastRestocked	Date	The timestamp of the most recent inventory replenishment.	Optional

Table 7.9: Stock Transaction Table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the transaction audit record.	Primary Key, Auto-generated
foodId	ObjectId	Reference to the FoodItem associated with this inventory movement.	Required, Foreign Key
transactionType	String	The nature of the change (e.g., 'sale', 'restock', 'adjustment', 'waste').	Required
quantity	Int32	The specific amount of stock added or removed during this event.	Required
previousQty	Int32	The exact stock level as it existed immediately before this transaction.	Required
newQty	Int32	The resulting stock level after the transaction was successfully committed.	Required
reason	String	A descriptive note explaining the purpose of the adjustment (e.g., "Order #1234").	Optional
performedBy	String	Identifier (Username or ID) of the staff member who initiated the change.	Required

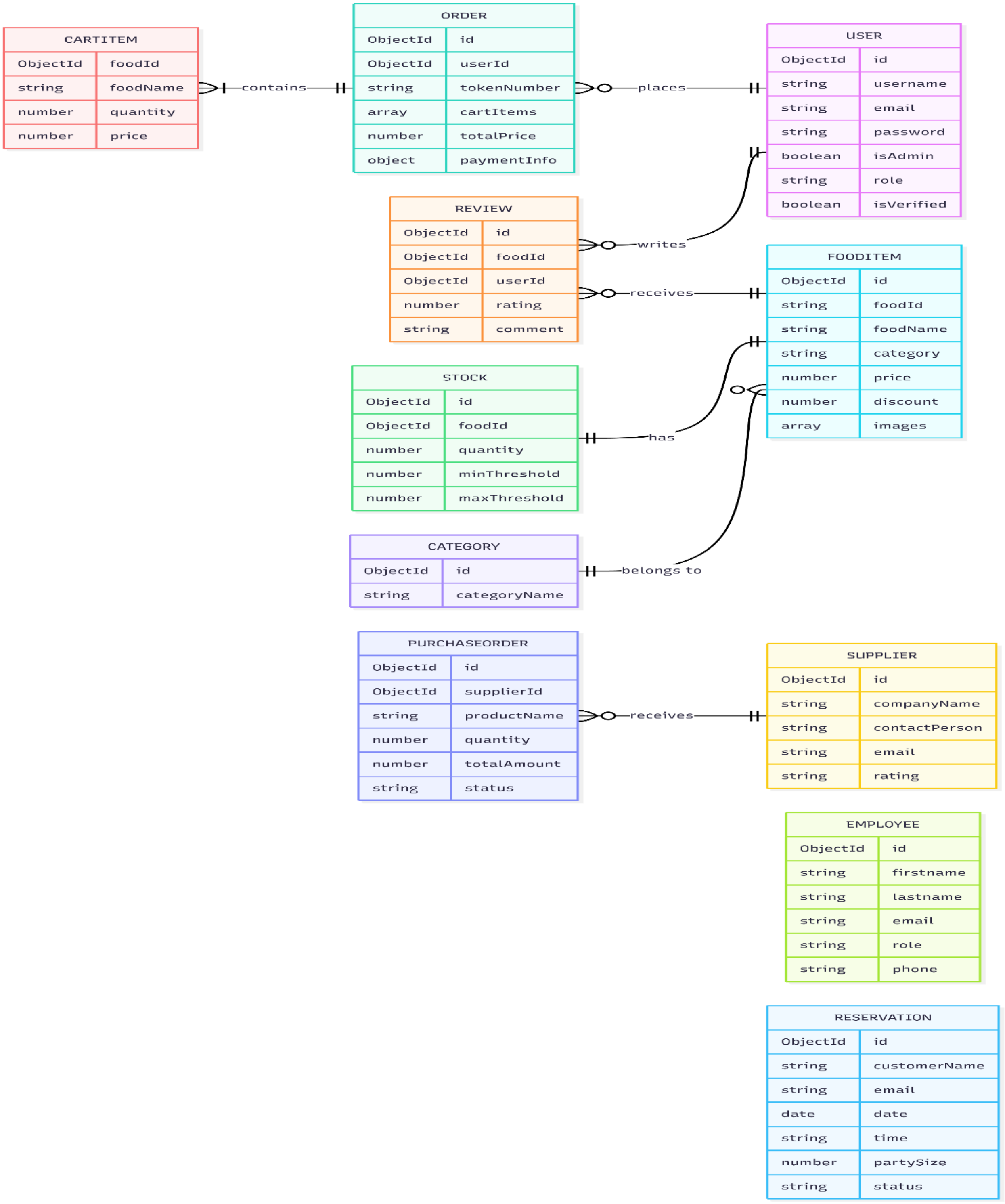
Table 7.10: Suppliers Table

Field Name	**Data Type**	**Description**	**Constraints**
_id	ObjectId	Unique system-generated identifier for the supplier record.	Primary Key, Auto-

			generated
companyName	String	The registered business name of the supplier company.	Required, Unique
contactPerson	String	The name of the primary representative or account manager.	Required
phone	String	Primary contact phone number for procurement and coordination.	Required
email	String	The supplier's official email address for purchase orders.	Required, Valid Email Format
address	String	The physical location or warehouse address of the supplier.	Required
itemsSupplied	Array	A collection of strings or IDs representing categories or specific food items provided.	Optional
rating	String	A qualitative or numeric assessment of the supplier's reliability.	Default: "Unrated"

7.4 ER Diagram for Banglar Heshel System

The Entity-Relationship (ER) Diagram of the Banglar Heshel system visually represents the database structure and the relationships between key collections. It shows how users interact with the menu system, including food browsing, cart management, and order placement processes. The diagram also illustrates how orders are associated with payments through SSLCommerz integration and how reviews are linked to specific food items, ensuring accurate tracking of transactions and customer feedback. Additionally, it represents inventory management functions such as stock tracking with automated alerts, supplier management, and purchase order workflows, which support operational efficiency and quality control. Administrative functions including user management, manager assignments, and role-based access control are also depicted. Overall, the ER Diagram provides a clear understanding of data organization and relationships within the Banglar Heshel system, supporting efficient database design using MongoDB and system implementation following the MERN stack architecture



CARTITEM	
ObjectId	foodId
string	foodName
number	quantity
number	price

ORDER	
ObjectId	id
ObjectId	userId
string	tokenNumber
array	cartItems
number	totalPrice
object	paymentInfo

USER	
ObjectId	id
string	username
string	email
string	password
boolean	isAdmin
string	role
boolean	isVerified

REVIEW	
ObjectId	id
ObjectId	foodId
ObjectId	userId
number	rating
string	comment

FOODITEM	
ObjectId	id
string	foodId
string	foodName
string	category
number	price
number	discount
array	images

STOCK	
ObjectId	id
ObjectId	foodId
number	quantity
number	minThreshold
number	maxThreshold

CATEGORY	
ObjectId	id
string	categoryName

PURCHASEORDER	
ObjectId	id
ObjectId	supplierId
string	productName
number	quantity
number	totalAmount
string	status

SUPPLIER	
ObjectId	id
string	companyName
string	contactPerson
string	email
string	rating

EMPLOYEE	
ObjectId	id
string	firstname
string	lastname
string	email
string	role
string	phone

RESERVATION	
ObjectId	id
string	customerName
string	email
date	date
string	time
number	partySize
string	status

Figure 7.29: ER Diagram of Banglar Heshel System

Chapter 8.

System Quality and Testing

8.1 System Quality Management

System quality management ensures that the developed Banglar Heshel Restaurant Management System meets its functional requirements, performance expectations, and user satisfaction goals. The quality of the system was managed throughout the project by applying structured testing methods, code reviews, continuous validation of requirements, and adherence to modern development best practices in the MERN stack ecosystem.

8.1.1 Software Quality Management Process

The quality management process followed these steps:

1. Requirement Verification: All customer, manager, and administrator requirements (food menu management, online ordering with cart functionality, payment processing through SSLCommerz, table reservations, review and rating system, inventory management with stock alerts, supplier coordination, etc.) were documented and confirmed with stakeholders before development. Requirements were validated against real-world restaurant operational needs to ensure practical applicability.

2. Design Validation: The MongoDB database schema, API architecture, React component hierarchy, and system architecture were validated to ensure consistency, logical data organization, efficient query patterns, and scalability. UI/UX mockups were reviewed for usability and responsive design across multiple devices.

3. Code Quality Assurance: Coding standards (JavaScript ES6+ conventions, React component best practices, RESTful API design principles, modular architecture patterns) were maintained throughout development. Code reviews were performed to avoid logical errors, eliminate redundancy, ensure proper error handling, and improve maintainability. Version control with Git and meaningful commit messages ensured code traceability.

4. Testing: Multiple types of testing, such as unit testing for individual functions, integration testing for API endpoints, system testing for complete workflows (registration to order completion), and user acceptance testing (UAT) with beta testers, were performed to verify interaction between different modules (User authentication, Food management, Order processing, Payment gateway, Reservation system, Stock management).

5. Performance Evaluation: Load testing was carried out to ensure the system could handle multiple simultaneous users browsing the menu, adding items to cart, placing orders, and making reservations without performance degradation. Database query optimization with MongoDB indexing and React component optimization with proper state management were implemented to maintain responsiveness.

6. Error Monitoring and Debugging: Bugs identified during testing (e.g., OTP expiry issues, payment callback failures, image upload errors, stock calculation discrepancies, cart persistence problems) were logged with detailed descriptions, categorized by severity (critical, high, medium, low), and systematically fixed before final deployment. Console logging and error tracking helped identify and resolve issues quickly.

7. Security Testing: Authentication mechanisms (JWT token validation, bcrypt password hashing, OTP verification), payment integration security (SSLCommerz webhook validation), input validation (both client-side and server-side), and protection against common vulnerabilities (SQL injection prevention through Mongoose, XSS protection, CSRF tokens) were thoroughly tested.

8. User Feedback Integration: End users (customers, restaurant managers) and administrative testers were involved in beta testing to provide feedback on usability, workflow efficiency, interface intuitiveness, and feature completeness. Feedback was analyzed and integrated into iterative development cycles to refine the user experience before final release.

1 .

8.1.2 Software Test Cases

The system was tested using multiple scenarios to ensure functionality, security, and reliability. Some key test cases are listed below:

Table 8.1: Test Cases for the System

Test ID	Feature Category	Input / Action	Expected Technical Outcome	Status
TC1	Authentication	Registration + OTP intake	Account committed to MongoDB; isVerified set to true.	Passed
TC2	Authorization	Login with valid credentials	JWT Token issued; redirected via role-based routing.	Passed
TC8	Financials	Place Order & Checkout	Handshake with SSLCommerz; redirection to secure gateway.	Passed
TC9	Financials	Success Callback (IPN)	tran_id stored; tokenNumber generated; cart purged.	Passed
TC11	Reservations	Valid form submission	Record saved in Reservation store with 'Pending' status.	Passed
TC13	Reviews	Submit rating & comment	Review linked via foodId; average rating recalculated.	Passed
TC14	Menu Mgmt	Add food + 6 images	Assets hosted on Cloudinary; Stock entry initialized.	Passed
TC15	Inventory	Stock update < threshold	quantity updated; lowStockThreshold alert triggered.	Passed

TC16	Payment Audit	Search by tokenNumber	Retrieves payment details, user info, and cart items from Payment store.	Passed
TC17	Booking Mgmt	Update status to "Confirmed"	Reservation record updated; trigger email service for customer notification.	Passed
TC18	Privilege Mgmt	Admin submits Manager form	New User created with 'Manager' role; synced with Employee list.	Passed
TC19	User Control	Admin clicks "Block"	isBlocked flag set to true in MongoDB; JWT authentication denied.	Passed
TC20	Recovery	Email + OTP + New Password	Password hashed via Bcrypt and updated in User store.	Passed
TC21	OAuth	Google Account Auth	Firebase token verification; session synchronization with local DB.	Passed

Chapter 9.

Ethical Consideration

9.1 Ethical Considerations in the Software Development Process

Developing the Banglar Heshel platform involved navigating complex ethical landscapes, particularly given its nature as a digital restaurant management system handling customer data, financial transactions, and operational business information.

i. **Data Privacy and Security**

1. In a restaurant management system, protecting user privacy is paramount. The system handles highly sensitive Personally Identifiable Information (PII), including customer names, email addresses, phone numbers, delivery addresses, and payment transaction details. To ensure confidentiality and compliance with data protection standards:
2. Encryption: Passwords are hashed using bcryptjs with salt rounds, ensuring one-way encryption that prevents password recovery even in case of database breach. All data in transit is encrypted via HTTPS in production deployment.
3. Access Control: Strict Role-Based Access Control (RBAC) ensures that only authorized Admins can manage users and employees, only Managers can access payment reports and inventory data, and Customers can only view their own orders and reservations.
4. Payment Security: Financial data is never stored locally in the database. The integration with SSLCommerz ensures that sensitive card details are handled by a PCI-DSS compliant payment gateway, with only transaction confirmation data (token numbers, status) stored in the system.
5. Secure Authentication: OTP-based email verification with 5-minute expiry prevents unauthorized account creation. JWT tokens with appropriate expiration times ensure secure session management without exposing user credentials.
6. Data Minimization: The system only collects information necessary for core functionality. Optional fields like profile pictures are not mandatory, and user data is not shared with third parties beyond essential service providers (Cloudinary for images, Nodemailer for emails).

ii. **Intellectual Property**

1. The development of Banglar Heshel respected all intellectual property rights. The solution utilizes open-source technologies such as Node.js, React.js, Express.js, MongoDB, and Tailwind CSS, adhering strictly to their MIT or Apache 2.0 licenses. External

services like SSLCommerz (payment gateway), Cloudinary (image storage), and Firebase (authentication) were integrated according to their Terms of Service, ensuring proper attribution and usage within free-tier limits during development. The core business logic, database schema, API architecture, and custom React components remain the intellectual property of the development team. All third-party libraries used (Redux Toolkit, Mongoose, React Router, React Icons, Recharts, jsPDF) are properly credited and used in compliance with their respective open-source licenses.

iii. **User Impact**

1. The system significantly impacts the daily operations of restaurant businesses and customer dining experiences. For Restaurant Managers and Staff, the platform provides critical tools for inventory management, order tracking, and customer service coordination. For Customers, it offers a convenient, efficient way to browse menus, place orders, make reservations, and provide feedback. The design prioritizes "Trust and Transparency"—features like order token numbers for tracking, email confirmations for all transactions, clear pricing with discount displays, and public review systems were implemented to build customer confidence. Accessibility best practices were followed in the UI design using responsive layouts and clear typography, ensuring the application is usable by customers of varying technical abilities and on different devices (mobile, tablet, desktop).

iv. **Bias and Fairness**

1. Banglar Heshel is designed to be an equitable platform serving all customers and operational staff fairly.
2. Algorithm Neutrality: The menu search and display algorithms prioritize relevance (keyword matching), category organization, and user ratings. All food items have equal visibility in search results and category browsing, ensuring fair representation of the entire menu.
3. Review Authenticity: The review and rating system (1-5 stars with comments) allows genuine customer feedback without algorithmic manipulation. All reviews are displayed chronologically with timestamps, providing transparency about customer experiences.
4. Fair Access: The platform does not discriminate based on customer demographics. All authenticated users have equal access to menu browsing, ordering, and reservation features. Role-based restrictions apply only to operational functions (Manager/Admin dashboards)

based on legitimate business needs.

5. Transparent Pricing: Food prices, discounts, and old prices are clearly displayed. The order summary shows itemized costs before payment, ensuring customers can make informed purchasing decisions without hidden charges.

v. **Responsibility to Stakeholders**

1. The primary stakeholders are Restaurant Customers, Restaurant Management and Staff, and System Administrators.
2. Customers: The development team is responsible for delivering a functional, reliable experience where orders are processed accurately, payments are secure, reservations are confirmed promptly, and customer data is protected. Email notifications keep customers informed about order status and reservations.
3. Management and Staff: The system must accurately track orders with token numbers, maintain inventory with low-stock alerts, process payments reliably, and provide analytics for business decision-making. The stock management system prevents overselling by tracking quantities in real-time.
4. Administrators: The platform empowers admins with tools to maintain system integrity (user management, manager oversight, platform monitoring) effectively while respecting user privacy and operational boundaries.
5. Regular feedback during beta testing with actual restaurant staff and customers helped align the software functionality with these diverse stakeholder needs and refine workflows for practical usability.

vi. **Professional Responsibility**

1. The development team adhered to software engineering ethical principles and best practices. This specifically meant:
2. Honesty: Clearly communicating system limitations (internet dependency for real-time operations, sandbox payment environment during development, free-tier cloud service constraints) to stakeholders without overselling capabilities.
3. Competence: Ensuring the payment integration, authentication security, and data validation modules were implemented using industry-standard best practices (JWT tokens, bcrypt hashing, Mongoose schema validation, SSLCommerz webhook verification) to prevent financial loss and security breaches.
4. Quality Assurance: Conducting thorough testing (unit tests,

integration tests, payment flow tests, security tests) before deployment to ensure the system functions reliably under normal and edge-case scenarios.

5. Maintenance Commitment: Establishing clear error logging and monitoring to identify issues quickly, and committing to resolving critical bugs (payment failures, authentication issues, data corruption risks) promptly to prevent customer disruption and business losses.
6. Transparency: Maintaining clear documentation of system architecture, API endpoints, database schema, and deployment procedures to facilitate future maintenance and knowledge transfer.
7. Sustainability: Designing the system with modular architecture and clean code practices to ensure long-term maintainability and extensibility as business requirements evolve.

vii. **Environmental and Social Responsibility**

1. While primarily a software system, Banglar Heshel also considers broader impacts:
2. Digital Efficiency: Cloud-based architecture (MongoDB Atlas, Cloudinary CDN) reduces the need for local server infrastructure, potentially lowering energy consumption compared to traditional on-premises systems.
3. Waste Reduction: Digital order tracking with token numbers and PDF receipts reduces paper waste from manual order management and printed receipts.
4. Local Business Support: By providing affordable digital tools (leveraging free-tier cloud services and open-source technologies), the platform helps small and medium-sized restaurants compete with larger chains that have custom technology investments

9.2 Sustainability in the Software Development Process

Sustainability is integrated into the core architecture of Banglar Heshel, addressing environmental, economic, and social dimensions.

1. Environmental Sustainability

- **Paperless Workflow:** By digitizing the entire service lifecycle—from Booking requests to Digital Receipts and Invoices—Banglar Heshel significantly reduces paper waste involved in traditional contracting and billing.
- **Efficiency:** The verification process is entirely digital, eliminating the need for physical mail or travel for document submission.

- **Server Optimization:** Database queries are optimized using Entity Framework's specialized projection to fetch only necessary data, reducing CPU cycles and energy consumption in data centers.

2. Economic Sustainability

- **Gig Economy Support:** The platform directly fosters the local economy by lowering the barrier to entry for skilled workers (plumbers, tutors, technicians), allowing them to start businesses with minimal overhead costs.
- **Cost Efficiency:** For users, finding a local provider reduces need for long-distance travel, saving fuel and time costs.
- **Scalability:** The modular "Clean Architecture" design allows the system to scale to new cities without complete redevelopment, ensuring long-term economic viability of the software asset itself. Pressman and Maxim (2014).

3. Social Sustainability

- **Trust Building:** By implementing a verification and review system, Banglar Heshel builds social capital and trust between strangers in a community.
- **Inclusivity:** The platform provides equal opportunity for independent workers who might not have the capital for traditional advertising store-fronts.
- **Employment:** Long-term, the platform acts as an engine for local employment, helping skilled individuals find consistent work. Turban and Volonino (2010).

Sustainability
Social Sustainability
Economic Sustainability
Environmental Sustainability (Green *level*)

Figure 9.1: Sustainability Considerations in Software Development

Chapter 10.
Conclusion

10.1 Brief Overview of the Project

The **Banglar Heshel** system was designed to provide a comprehensive digital restaurant management platform that streamlines operations and enhances customer experience. It serves as a modern solution for restaurant businesses, replacing fragmented manual processes (order taking on paper, phone-based reservations, physical inventory tracking, cash transactions) with a centralized, efficient digital platform. By introducing features such as online menu browsing with search and filtering, shopping cart and wishlist functionality, secure payment processing through SSLCommerz with unique token-based order tracking, table reservation management, customer review and rating system, real-time inventory management with automated low-stock alerts, supplier coordination with purchase order tracking, and comprehensive analytics dashboards, the project aims to modernize restaurant operations. On the administrative side, the system offers robust tools for multi-role access control (Admin, Manager, Customer), user management, and comprehensive reporting, ensuring efficient business oversight and customer service management.

10.2 Proposed System Benefits

The developed system offers a range of benefits for both customers and restaurant operations:

1. **Customer Convenience**: Users can easily browse the complete menu with images and pricing, search for specific dishes, add items to cart or wishlist, place orders online with secure payment, make table reservations with email confirmations, and provide reviews and ratings to share their dining experiences.
2. **Operational Efficiency**: Provides restaurant managers with digital tools to manage food items with multi-image uploads via Cloudinary, track orders with unique token numbers, monitor payments and revenue, manage table reservations with status updates, control inventory with automated alerts, coordinate with suppliers through purchase order management, and access analytics dashboards for data-driven decision making.
3. **Trust & Transparency**: The token-based order tracking system, email confirmations for all transactions (orders, payments, reservations), public review and rating display, clear pricing with discount information, and PDF receipt generation ensure transparency and build customer confidence in the platform.
4. **Inventory Management**: Automated stock tracking with configurable minimum and maximum thresholds, real-time low-stock email alerts to managers, automatic stock entry creation for new food items, and integration with supplier management reduce waste, prevent stock-outs, and optimize purchasing decisions.

5. **Secure Transactions**: Integration with SSLCommerz payment gateway provides a PCI-DSS compliant and secure environment for financial exchanges, supporting multiple payment methods (credit/debit cards, mobile banking, digital wallets), minimizing fraud risk through webhook verification, and ensuring customer payment data is never stored locally.
6. **Cost-Effective Solution**: Built entirely on open-source technologies (MERN stack) and leveraging free-tier cloud services (MongoDB Atlas, Cloudinary, Firebase, Nodemailer), the platform provides enterprise-grade features at minimal operational cost, making it accessible for small and medium-sized restaurants.
7. **Scalable Architecture**: Modular design with separated frontend (React), backend (Express), and database (MongoDB) layers, cloud-native deployment, RESTful API architecture, and efficient state management (Redux) enable the system to grow with business needs without major architectural changes.

10.3 Limitations of the Project

While the system meets its primary objectives, several limitations remain, which highlight areas for improvement:

1. Internet Dependence: As a cloud-hosted web application, it requires a stable internet connection for all operations, which may limit accessibility in areas with poor network coverage or during internet outages, affecting both customer ordering and restaurant operations.

2. Payment Gateway Sandbox: The current implementation uses the SSLCommerz Sandbox environment for testing purposes; a live production deployment would require merchant account setup, compliance auditing, legal agreements, and activation of production API credentials.

3. Lack of Native Mobile App: The system is a responsive web application (PWA-capable), which lacks native mobile features like push notifications for real-time order updates, offline menu browsing, native camera integration for faster image uploads, and home screen installation prompts.

4. Limited Real-Time Features: The system currently lacks real-time updates for order status changes, requiring page refreshes to see new data. Implementation of WebSocket connections would enable instant notifications for new orders, reservation confirmations, and low-stock alerts without manual refresh.

5. Single Payment Gateway: The system currently supports only SSLCommerz, limiting payment options. Integration with additional payment methods like bKash, Nagad, Rocket (mobile financial services popular in Bangladesh), and international options like Stripe would improve accessibility.

6. Cloud Service Dependencies: Reliance on free-tier cloud services (MongoDB Atlas 512MB, Cloudinary 25GB, Gmail SMTP limits) means the system may face quota limitations under heavy usage, requiring paid upgrades for production deployment at scale.

7. Basic Analytics: The current analytics dashboard provides fundamental metrics (total revenue, order counts, popular items), but lacks advanced features like predictive analytics for demand forecasting, customer segmentation, seasonal trend analysis, and automated business intelligence reporting.

These limitations emphasize that while the project demonstrates strong architectural principles, comprehensive functionality, and practical value for restaurant operations, further refinement and feature additions are needed for commercial-grade deployment serving high-traffic restaurant businesses

10.4 Practicum and Its Value

The practicum offered meaningful real-world learning opportunities by connecting theoretical knowledge with practical application in modern web development. The experience included:

1. Technical Mastery: Deepened expertise in the MERN stack ecosystem (MongoDB with Mongoose ODM, Express.js RESTful API development, React.js with Hooks and functional components, Node.js server-side programming), along with modern development tools (Redux Toolkit for state management, Tailwind CSS for responsive design, Git/GitHub for version control).

2. Third-Party Integration: Gained practical experience integrating diverse external services such as SSLCommerz (payment gateway with webhook handling), Cloudinary (cloud image storage with CDN delivery and automatic optimization), Nodemailer (SMTP email service for OTP and notifications), and Firebase (Google OAuth authentication), understanding API documentation, authentication flows, and error handling.

3. Complex Logic Implementation: Solved challenging problems related to shopping cart state management with localStorage persistence, order processing workflows with payment gateway integration, OTP generation and expiry handling, inventory tracking with threshold-based alerts, role-based access control implementation, and database schema design with proper relationships and data validation.

4. Professional Development: Reinforced the importance of code organization with modular architecture, implementing security best practices (JWT authentication, password hashing, input validation), conducting systematic testing across multiple scenarios, maintaining comprehensive documentation, following RESTful API design principles, and adhering to modern JavaScript coding standards (ES6+ syntax, async/await patterns, error handling).

5. Full-Stack Understanding: Gained holistic perspective on web application development, understanding how frontend UI/UX decisions impact backend API design, how database schema choices affect query performance, how security measures must be implemented at multiple layers, and how user experience depends on seamless integration of all system components.

Overall, the practicum bridged the gap between academic concepts and industry-standard software engineering practices, providing hands-on experience with technologies widely used in professional web development and preparing for real-world software engineering challenges

.

10.5 Future Plan

Future development of the system will focus on addressing the identified limitations and preparing it for larger-scale deployment. Planned improvements include:

1. Native Mobile Application: Developing cross-platform mobile apps using React Native to leverage native features including push notifications for order status updates and reservation confirmations, offline menu browsing with cached data, native camera integration for faster photo uploads, biometric authentication (

2. Real-Time Features: Implementing WebSocket connections (using Socket.io) to enable real-time order notifications to kitchen staff, live order status updates for customers, instant reservation confirmations, real-time stock level updates across multiple devices, and in-app messaging between customers and restaurant staff for order clarifications.

4. Advanced Analytics & Business Intelligence: Using data aggregation and visualization libraries to provide predictive analytics for demand forecasting, customer behavior analysis and segmentation, seasonal trend identification, inventory optimization recommendations, revenue forecasting, popular item t

5. Enhanced Inventory Features: Implementing barcode/QR code scanning for faster stock updates, automated reorder point calculations based on historical consumption patterns, vendor performance tracking and comparison, batch tracking for perishable items with expiry date monitoring, and integration with POS systems for real-time inventory synchronization.

6. AI-Powered Recommendations: Using machine learning algorithms to suggest menu items based on customer preferences and order history, predict popular items during specific time periods, optimize menu pricing based on demand patterns, and provide personalized dining suggestions.

7. Delivery Integration: Integrating with third-party delivery services (Foodpanda, Pathao Food) APIs, implementing in-house delivery management with driver assignment and tracking, real-time order tracking with GPS, estimated delivery time calculations, and customer delivery feedback collection

.

Bibliography

[1] Grand View Research, "Restaurant Management Software Market Size, Share & Trends Analysis Report," 2023. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/restaurant-management-software-market

[2] McKinsey & Company, "Digital transformation in the restaurant industry," 2023. [Online]. Available: https://www.mckinsey.com/industries/retail/our-insights/digital-transformation-restaurant-industry

[3] Deloitte, "Restaurant industry trends and technology adoption," Deloitte Insights, 2024. [Online]. Available: https://www2.deloitte.com/us/en/insights/industry/retail-distribution/restaurant-industry-trends.html

[4] Restaurant Technology News, "Cloud-based restaurant management systems: Benefits and implementation," 2024. [Online]. Available: https://restauranttechnologynews.com/cloud-restaurant-management

[5] Food Service Technology, "Inventory management best practices for restaurants," 2025. [Online]. Available: https://www.foodservicetechnology.com/inventory-management-restaurants

[6] MongoDB Inc., "MongoDB Manual - Database Design Patterns," MongoDB Documentation, 2024. [Online]. Available: https://www.mongodb.com/docs/manual/

[7] Meta Open Source, "React - A JavaScript library for building user interfaces," React Documentation, 2024. [Online]. Available: https://react.dev/

[8] OpenJS Foundation, "Node.js Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs/

[9] SSLCommerz, "SSLCommerz Integration Guide," 2024. [Online]. Available: https://developer.sslcommerz.com/

[10] Cloudinary Ltd., "Cloudinary Developer Documentation," 2024. [Online]. Available: https://cloudinary.com/documentation

[11] Andris Reinman, "Nodemailer Email Service Documentation," 2024. [Online]. Available: https://nodemailer.com/about/

[12] Google LLC, "Firebase Authentication Documentation," Firebase, 2024. [Online]. Available: https://firebase.google.com/docs/auth

[13] Nielsen Norman Group, "UX Research and Design Best Practices," 2023. [Online]. Available: https://www.nngroup.com/

[14] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2024. [Online]. Available: https://owasp.org/www-project-top-ten/

[15] IETF, "JSON Web Token Best Practices," RFC 8725, 2024. [Online]. Available: https://tools.ietf.org/html/rfc8725

[16] Modern Restaurant Management, "Restaurant Inventory Management Best Practices," 2024. [Online]. Available: https://modernrestaurantmanagement.com/inventory-management-best-practices/

[17] Restaurant Business Online, "Restaurant Business Intelligence and Analytics Guide," 2024. [Online]. Available: https://restaurantbusinessonline.com/technology/business-intelligence

[18] Meta Open Source, "React Application Architecture Best Practices," 2024. [Online]. Available: https://reactjs.org/docs/thinking-in-react.html

[19] B. P. Mramba and S. F. Kaijage, "Design of an Interactive Mobile Application for Maternal, Neonatal and Infant Care Support for Tanzania," J. Softw. Eng. Appl., vol. 11, no. 12, pp. 569–584, 2018.

[20] R. S. Pressman and B. R. Maxim, Software Engineering: A Practitioner's Approach, 8th ed. New York: McGraw-Hill Education, 2014.

[21] C. Ghezzi, M. Jazayeri, and D. Mandrioli, Fundamentals of Software Engineering, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[22] I. Sommerville, Requirements Engineering, 9th ed. Boston, MA: Pearson, 2011.

[23] I. Sommerville, Requirements Engineering: From System Goals to UML Models to Software Specifications. London, UK: Pearson Education, 2011.

[24] I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson, 2016.

[25] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed. Boston, MA: Addison-Wesley, 2012.

[26] E. Turban and L. Volonino, Information Technology for Management, 7th ed. Hoboken, NJ: John Wiley & Sons, 2010.

[27] A. R. Hevner and S. Chatterjee, Design Research in Information Systems: Theory and Practice. New York, NY: Springer, 2010.

[28] IEEE Computer Society, "IEEE Software Engineering Standards," 2023. [Online]. Available: https://www.computer.org/education/bodies-of-knowledge/software-engineering