CHAPTER I: INTRODUCTION

1 Background of the Project

In today's fast-paced and digitally-driven environment, cafés and small food establishments are increasingly relying on technology to streamline their operations, enhance customer service, and remain competitive. A Point of Sale (POS) system is a critical component in achieving these goals. It serves not just as a cash register but as a comprehensive platform for managing sales transactions, inventory, and customer interactions.

Traditional manual methods of sales and inventory tracking often lead to inefficiencies such as human error, slow processing times, and difficulty in record-keeping. A digital POS system can significantly mitigate these issues by automating sales processing, billing and generating useful sales reports. This is especially beneficial for small businesses like cafés, where daily operations need to be smooth and responsive to customer needs.

2 Problem Statement

In many small-scale cafés, daily operations such as order taking, table management, billing, and sales tracking are often handled manually or with outdated systems. This can lead to various issues such as slow service during peak hours, human errors in billing, difficulty in managing customer data, and lack of proper records for business analysis. Additionally, staff may require more time to learn complex systems, and café owners may not have the budget for expensive POS software or specialized hardware.

To address these challenges, there is a need for a lightweight, easy-to-use, and cost-effective Point of Sale (POS) system that automates key operations. The system should allow café staff to take orders quickly, generate bills automatically, manage tables efficiently, and provide an admin dashboard to monitor daily sales and manage products. It should also ensure secure user authentication and store data reliably.

This project emphasizes the development of a web-based POS system to help small cafés improve operational efficiency, reduce manual workload, minimize errors, and enhance customer service—all while keeping costs low and the system easy to use.

3 Objective of the Project

The main objective of this project is

- To design and develop a Point of Sale (POS) system tailored for cafe management.
- To provide a user-friendly interface for cafe staff to streamline the transaction process and to automate billing.

4 Literature Review

A Point of Sale (POS) system is an essential tool for managing retail and food service operations. It allows businesses to process transactions, automate billing and gain insights through sales reporting. This chapter reviews relevant literature on POS systems, their functionalities, and their impact on small business efficiency, particularly in café settings.

1.4.1 Overview of Point of Sale Systems

A POS system is more than just a payment processor; it is an integrated system that helps track sales, inventory, and customer preferences. According to Laudon and Laudon (2020), POS systems form a part of the broader Management Information Systems (MIS) that help businesses make strategic decisions by collecting, processing, and storing data. They play a vital role in enhancing operational efficiency and ensuring accuracy in sales records.

The integration of POS systems has significantly contributed to improving operational processes within restaurant businesses. Kocaman (2021) emphasized that POS systems enhance operational outcomes by improving inventory management, customer service efficiency, and overall transaction accuracy. Similarly, Parkan (2003) measured the impact of a newly implemented POS system in drugstore operations, finding substantial improvements in transaction speed and service quality. These findings highlight how POS technologies align business operations with technological advancements to streamline service delivery.

1.4.2 Importance of POS Systems in Small Businesses

Small businesses like cafés benefit from using POS systems due to their ability to streamline operations. Turban et al. (2018) stated that POS systems reduce the complexity of daily transactions and inventory management by automating manual tasks. This is particularly important in small food establishments where quick service and accuracy are essential. Furthermore, Smith and Chang (2019) found that businesses that adopted digital POS systems reported a 25% increase in order accuracy and a 30% reduction in processing time.

Kızılırmak and Ergun (2022) observed that staff in restaurants using such technology perceived improvements in sales, operational management, and standardization of service production. Moreover, chain restaurants experienced more substantial benefits than smaller establishments, suggesting that the size and complexity of a business influence the perceived usefulness of such systems.

The impact of POS systems on business performance has also been quantified. Parkan (2003) applied the Operational Competitiveness Rating Analysis (OCRA) to assess the effect of POS deployment in Hong Kong drugstores, confirming a statistically

significant improvement in performance metrics such as sales and operational efficiency after implementation.

A case study by Wijaya (2021) on coffee shop POS implementation reported improvements in inventory tracking, demand prediction, and transaction speed. Despite challenges such as employee training and initial investment costs, the long-term benefits included higher profitability and better customer service. These findings are consistent with broader trends in POS deployment across foodservice sectors.

Modern POS systems are increasingly leveraged as platforms for enhancing customer engagement. Garaus, Wagner, and Rainer (2021) explored the use of digital signage integrated with facial recognition technology at the POS, finding that emotional targeting could significantly personalize customer experiences and drive purchase behavior. By adapting marketing strategies based on real-time emotional data, businesses can deepen customer loyalty and increase overall sales conversion rates.

1.4.3 Features of an Effective POS System

An effective POS system should include user-friendly interfaces, inventory tracking, billing modules, and reporting tools. According to Rouse (2021), POS systems should also offer real-time data tracking and customizable interfaces to meet the specific needs of different businesses. In café operations, where menu items and prices change frequently, flexibility and ease of use are critical features

1.4.4 Challenges in POS System Implementation

Despite their benefits, POS systems can present challenges, especially in terms of cost, user training, and system integration. Kumar and Srinivas (2020) highlight that small businesses may struggle with the initial investment and technical skills required to operate complex POS software. Moreover, issues such as system downtime or lack of internet connectivity can affect business continuity, especially for cloud-based systems.

1.4.5 Recent Trends and Developments

The POS industry has seen major advancements with the introduction of cloud-based and mobile POS systems. These systems offer portability, scalability, and remote access, which are particularly beneficial for small cafés. According to a report by Statista (2023), cloud-based POS adoption has increased significantly, with 58% of small food businesses in urban areas adopting such systems due to their low upfront cost and ease of maintenance.Recent innovations in AI and automation have extended the capabilities of POS systems. According to Rachinger et al. (2023), AI-driven tools like chatbots, mobile kiosks, and robotic servers are being explored to revolutionize customer service in high-contact industries such as hospitality. These technologies can augment human labor and bring process innovations, although full implementation remains limited due to emotional and interpersonal service needs.

5 Development Methodology

The method used for this project development is Agile Software Development Methodology which is flexible, iterative, and ideal for evolving requirements during development.This POS system includes multiple independent modules (e.g., booking, billing, authentication), which has made agile methodology viable approach. Here's the methodology structure:

1. Requirement Gathering

- Identified stakeholders : café owner, staff, customers.
- Core features: order taking, billing and receipt generating, table booking, sales reporting, etc.

Output: Product Vision + Feature List

2. Planning & Prioritization

Break the system into modules:

- Order Management
- Payment Processing
- Sales Reports
- Admin Dashboard

Prioritize MVP (Minimum Viable Product) features: Basic UI, order management, biling, authentication.

3. Design

- **System Architecture:**
  - MERN stack (MongoDB, Express, React, Node.js)
  - Consider offline-first or local-server-based architecture if the café lacks stable internet.
- **UI/UX Design:**
  - Wireframes for cashier terminal, order screen, admin dashboard
  - Mobile/tablet compatibility if needed
- **Database Design:**
  - Entities like User, Order, Item, Invoice and Table

4. **Development**

- Short development sprints (1–2 weeks each).
- Implement modules incrementally:

  Sprint 1: Basic UI, authentication

  Sprint 2: Order & table booking logic.

  Sprint 3: Billing logic

  Sprint 4: UI polishing, receipt printing, etc.

5. **Testing**

- **Unit testing** : for individual modules has been done in this phase i.e. functions and APIs
- **Integration testing** : End-to-end workflows

  Staff: → Create customer profile → Table booking → Order → Generate bill

  Admin:: → Add product Category → List products →Check sales report

- **User Acceptance Testing (UAT)** : Tested by café staff in real use-case scenarios

6. **Deployment**

- Deploy to a local server.
- Prepare user manuals or training materials.

7. **Maintenance & Feedback**

- Bugs and performance has been regularly monitored.
- Feedback from café users has been continuously considered.
- Release updates based on usage insights in future.

6 Scope and Limitations

The scope of this project includes basic functionalities such as product listing, order taking, billing, and sales reporting. The system has been developed with a focus on small to medium-sized cafés.

However, the project is limited by the following:

- It will not include advanced features such as customer loyalty programs or multi-branch support.
- The system will be designed for a single-user interface at a time.
- Integration with external hardware (e.g., receipt printers) will be simulated but not implemented.

7 Report Organization

This section provides an overview of how the BIM 6th Semester Project Report is structured.

Table 0.1.1: Report organizaton

Chapter No.	Chapter Title	Description
Chapter 1	**Introduction**	Provides an overview of the project, objectives, problem statement, scope, and significance. Discusses existing research, technologies, and related works that support the project.
Chapter 2	**System Development Process**	
	System Analysis	Explains user requirements, functional & non-functional requirements, and feasibility analysis.
	System Design	Describes system architecture, data flow diagrams (DFD), entity-relationship diagrams (ERD), and interface design.
	Implementation	Covers coding standards, naming conventions, frameworks, and database design used in development.
	Testing & Evaluation	Describes testing strategies, test cases, and system evaluation results.
Chapter 3	**Conclusion & Recommendations**	Summarizes findings, achievements, challenges, and future enhancements.

—	**References**	Lists books, research papers, websites, and other sources cited in the report.
—	**Appendices**	Includes screenshots, user manuals, and additional information.

CHAPTER II: SYSTEM DEVELOPMENT PROCESS

2.1 Analysis

2.1.1 Requirement Analysis

The requirements were gathered through discussions with the café owner and observation of day-to-day operations. Stakeholders identified include café staff (cashiers, baristas), the café owner (admin), and customers (indirect users).

2.1.1.1 Functional Requirement

- Product and category listing
- Order placement and modification
- Automated billing and receipt generation
- Daily/weekly/monthly sales reporting
- Login authentication for staff

i. Functional requirement table

Table 2.1: Functional requirement table

Requirement ID	Requirement Description	Inputs	Outputs
POS_01	Staff should be able to register an account	Name, Email, Phone number Password, role	Confirmation
POS_02	Staff should be able to log in	Email, Password	Dashboard/Home page loaded
POS_03	System should redirect user to the Home page after login	Authenticated credentials	Home page with POS options
POS_04	Staff should be able to create an order with customer details	Customer Name, Contact, no. of guests	Order created and saved to system

POS_05	Staff should be able to book a table for a customer	Table number, Customer ID	Table booking confirmation.
POS_06	Staff should be able to add or remove menu items to a cart	Menu item ID, Quantity	Cart updated with items
POS_07	Staff should be able to place the order and generate a bill	Cart items, Payment method(only cash)	Order saved, Printable bill generated

ii. Use case

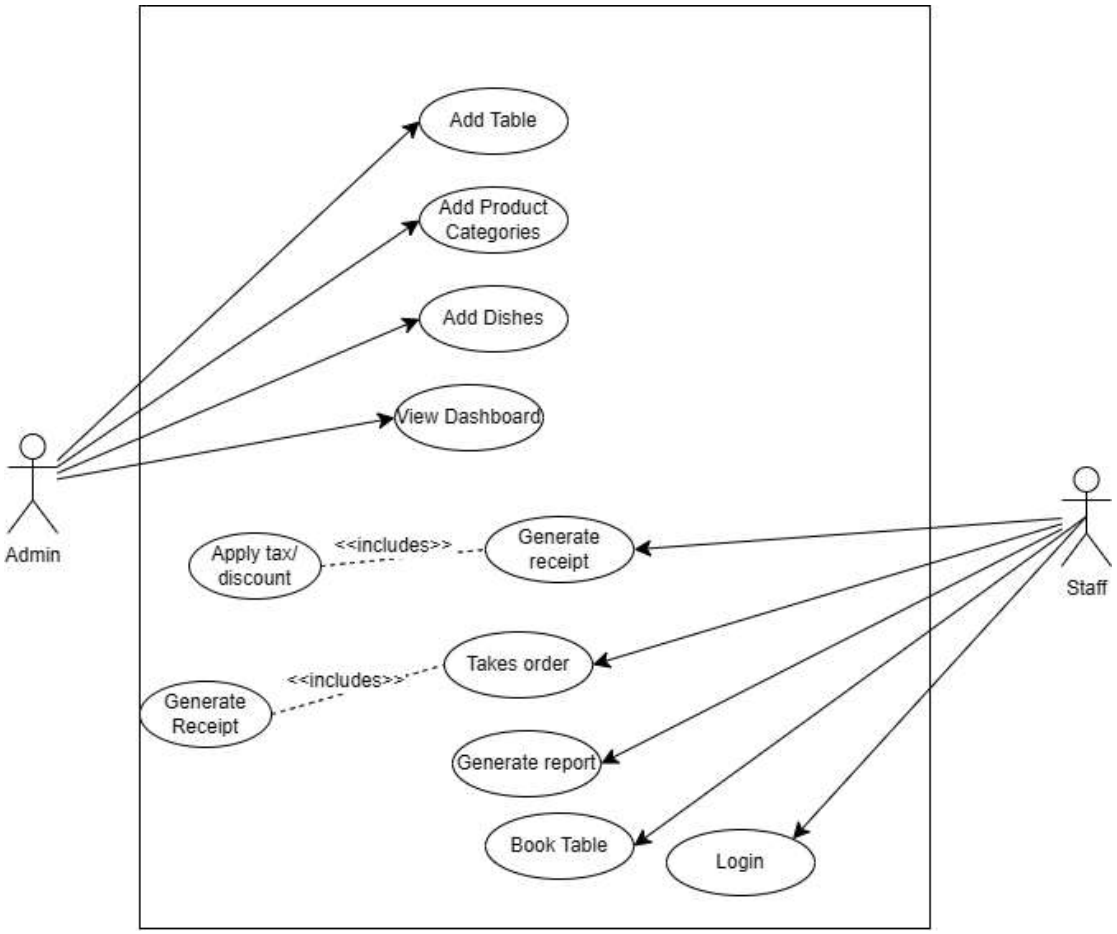

Figure 2.1 : Use case Diagram

Use Case: Login

Table 2.2: Login Use Case

Use Case	**Login**
Actors	Admin, Staff
Preconditions	User must be registered
Steps	Navigate to login page Enter email and password Click "Login" System verifies credentials Redirect user to appropriate dashboard

Use Case: Add Dishes

Table 2.3: Add Dishes Use Case

Use Case	**Add Dishes**
Actors	Admin
Preconditions	Admin must be logged in
Steps	Navigate to "Add Dishes" page Enter dish name, price, category Click "Add Dishes" System adds dish to the menu

Use Case: Add Product Categories

Table 2.4: Add Product Categories Use Case

Use Case	**Add Dishes**
Actors	Admin
Preconditions	Admin must be logged in

Steps	Navigate to "Add Category" page Enter category name Click "Add Category" System stores new category

Use Case: View Dashboard

Table 2.5: View Dashboard Use Case

Use Case	**Add Dishes**
Actors	Admin
Preconditions	Admin must be logged in
Steps	Login to admin account Navigate to Dashboard System shows reports and stats

Use Case: Takes Order

Table 2.6: Order Use Case

Use Case	**Add Dishes**
Actors	Staff
Preconditions	Staff must be logged in
Steps	1. Select create order button 2. Add customer details 3.Choose available table 3. Add to order list 4. Click "Place Order"

Use Case: Generate Receipt

Table 2.7: Generate receipt Use Case

Use Case	**Add Dishes**
Actors	Staff
Preconditions	Valid completed order
Steps	Open order details Click "Generate Receipt" System calculates total Receipt is displayed

Use Case: Book Table

Table 2.8: Book table Use Case

Use Case	**Book Table**
Actors	Staff
Preconditions	Table must be available
Steps	Check available tables Select table Enter customer orders

Use Case: Generate Report/ overview of performance

Table 2.9: Generate Report Use Case

Use Case	**Generate Report**
Actors	Admin, Staff
Preconditions	User must be logged in
Steps	Navigate to Dashboard Choose date range

	System shows overall performance of the cafe

2.1.1.2 Non- Functional Requirement

- User-friendly interface with minimal learning curve
- Quick response time for transaction processing
- Secure data handling

Table 2.10 : Non-Functional Requirement

Requirement ID	Category	Description	Example
POS_01	Performance	The system should load the dashboard within 2 seconds for 90% of users.	After login, the POS dashboard should not take more than 2 seconds.
POS_02	Scalability	The system should support up to 50 simultaneous orders without slowdown.	System should perform well during busy hours (e.g., lunch time).
POS_03	Security	User credentials must be encrypted and sensitive data securely stored.	Implemented bcrypt to hash passwords and JET token sent from/to client
POS_04	Usability	Interface should be intuitive for quick learning with minimal training.	POS layout should follow Material Design; allow first-time use in 15 -20 min.
POS_05	Reliability	The system should run smoothly with minimal crashes and work at least 95% of the time.	Should not be down for more than 4 hours.
POS_06	Maintainability	The system should allow easy menu and pricing updates.	Enable admins to update items without developer intervention.

POS_07	Compatibility	The system should work on all major modern browsers.	POS must work on Chrome, Safari, Edge
POS_08	Compliance	The system must comply with customer request / permission for customer data.	Customer info (e.g., contact) must be anonymized upon request.
POS_09	Backup & Recovery	The system should automatically backup data.	Use secure cloud services; allow recovery from last backup.

2.1.2 Feasibility Study

2.1.2.1 Technical feasibility

The system is technically possible because it uses simple and modern web tools that work well for a small café. The frontend (what users see) is built using React, which makes the interface fast and easy to use for staff and admins. The backend (the server part) uses Node.js a

nd Express, which can handle many order requests at the same time without slowing down. For storing data, the system uses MongoDB, which is flexible and can be used either on the café computer or online using MongoDB Atlas. Since the system is made to run locally, it can work on a normal PC or laptop in the café, without needing expensive internet or cloud services. Also, all the tools used are open source, free, well-documented, and easy to find help for. This makes the system easy to build, use, and maintain even by a small team or one person. So, it is technically a good and simple choice for a local café POS system.

2.1.2.2 Economic Feasibility

- Built entirely using open-source technologies: No licensing fees.
- Runs on existing café computers or affordable new devices.
- Minimal operating costs: No recurring cloud server fees if deployed locally.
- Reduces paperwork and manual billing, saving staff time and errors

Since the project will be built using open-source technologies and does not rely on expensive hardware integration, development costs are minimal, which make sit cost effective for small cafes with tight budget.

2.1.2.3 Legal Feasibility

The POS system is legally safe to use for a small café because it only stores basic information like customer orders and, if needed, names or contact details for things like table bookings or receipts. It does not handle sensitive information like credit card numbers or medical records, so there are fewer legal concerns. If customer data is collected, the system can be designed to allow deleting or hiding that information if the customer requests it, which supports basic privacy rights. While the system does not need to follow strict data protection laws like GDPR for local use, it can be improved in the future if the café grows or starts handling more personal data. Overall, the system is legally acceptable for small-scale use with simple privacy features.

2.1.2.4 Operational Feasibility

- The POS interface is simple, designed for café staff with little to no tech training.
- User testing shows most users can learn in under 15 minutes.
- Staff can:
    - Take orders with a few clicks
    - Print or view bills
    - Update menus via admin dashboard
- Offline/local operation avoids internet dependency, which is ideal for many small cafés.

Hence, Café operations can continue smoothly with minimal disruption or training.Simple UI design and guided workflows ensure minimal training.

2.1.2.5 Schedule Feasibility

Month 1: Gather requirements, list features, prioritize MVP, and plan sprints.

Month 2: Design system architecture and UI wireframes; set up project structure.

Month 3: Develop user authentication and basic frontend/backend setup.

Month 4: Build order placement, table booking, and billing features.

Month 5: Create admin dashboard, reports, and polish the UI.

Month 6: Test thoroughly, fix bugs, deploy locally, and train café staff.

Over the course of six months, the POS system has been developed step-by-step, starting with gathering requirements and planning, followed by designing the

architecture and user interface. Development has focused first on authentication and basic features, then move on to order handling, billing, and the admin dashboard. The final month has been dedicated to thorough testing, fixing issues, deploying the system locally, and training café staff. By the end, we have a stable and fully functional POS system ready for everyday use, with staff confident in operating it.

2.1.3 Object oriented modeling

2.1.3.1 System Overview

The Point of Sale (POS) system is designed to streamline and automate the café's daily operations by allowing Staff to handle all sales-related functions. Staff interact with the system to take customer orders, process payments, issue receipts, and update inventory in real-time.The system also generates sales summaries and inventory status reports for internal review and recordkeeping. This centralized system reduces manual effort, increases accuracy, and enhances the speed of service, ensuring smooth and efficient café operations.

2.1.3.2 Class Diagram

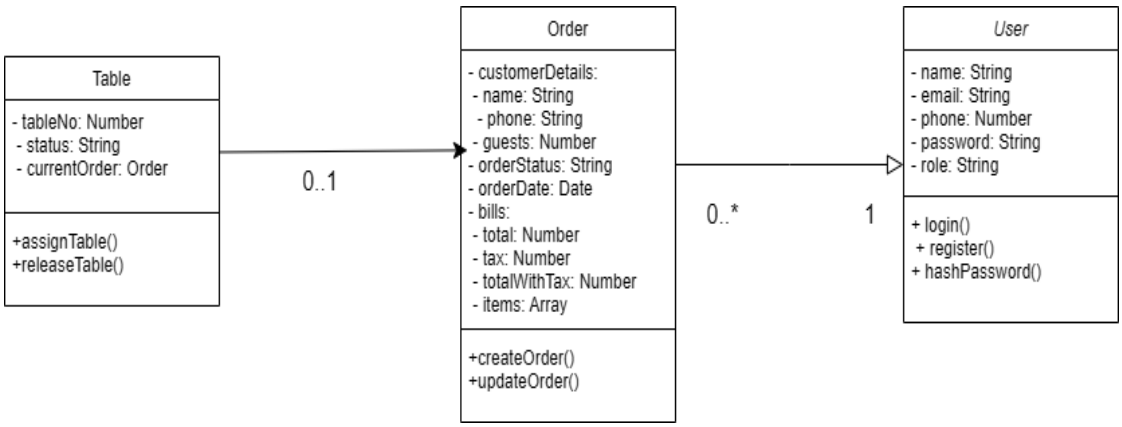

Figure 2.2 :Class Diagram

Relationships:

- A **User** can place **many Orders**.
- A **Table** can be assigned to **one Order** at a time.
- An **Order** contains a reference to a **Table**

2.1.3.3 Object Diagram

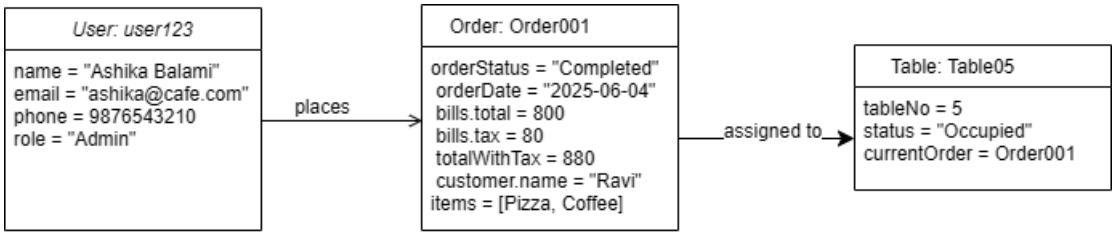

Figure 2.3: Object Diagram

2.1.3.4 Sequence Diagram

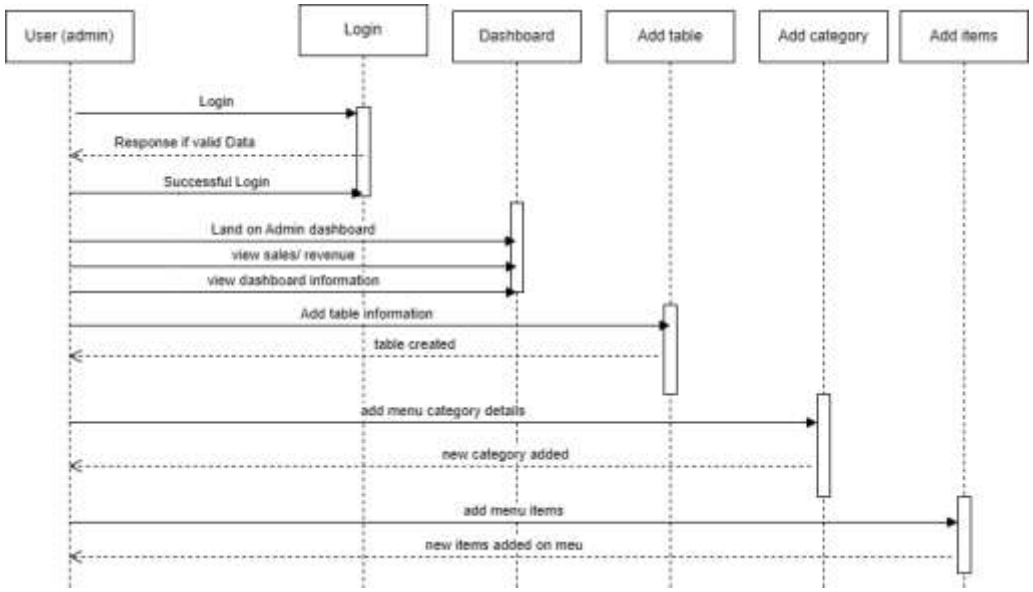

Figure 2.4: Admin sequence diagram

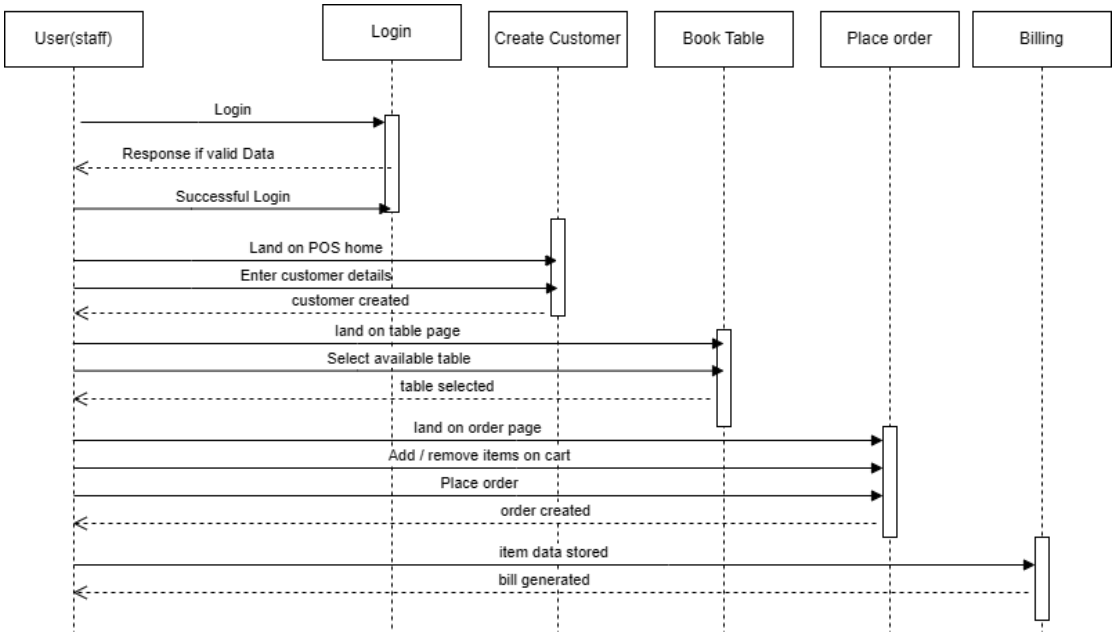

Figure 2.5: Staff sequence diagram

Admin Sequence Diagram shows the flow for an **Admin** user**:**

- Logs in and accesses the dashboard.
- Adds new tables, menu categories, and items.
- System confirms each addition.

Staff Sequence Diagram **s**hows the flow for a **Staff** user**:**

- Logs in, creates a customer, and books a table.
- Adds items to the cart and places the order.
- System stores order and generates the bill.

2.1.4 System Requirement Specification (SRS)

Table 2.10 System Requirement Specification Table

Category	Requirement	Specification
Server Processor	Basic Intel Core i5 or similar	Fast processing to handle concurrent orders and database requests.
Server RAM	At least 8GB or higher	Handles multiple simultaneous sessions and database operations.
Server Storage	Minimum: 256GB SSD	Medium storage for menu items, orders, and logs.
Client Processor	Minimum: Intel Core i3 or equivalent Recommended: Intel Core i5/i7	Smooth operation for billing, order creation, and table management.
Client RAM	Minimum: 4GB Recommended: 8GB or higher	Smooth UI and multi-tab functionality.
Client Storage	Minimum: 128GB SSD or more	Local caching, offline mode capability.
Network	Minimum: 25 Mbps for smooth flow	Stable internet for web app and data sync.
Display	Minimum: 1366×768 resolution	Better and clear UI layout for order and table management.

2.2 Design

1. User Interface Design

The user interface (UI) of the POS system is designed to be simple, clear, and easy to use for both café staff and administrators. Key screens include:

- Login Screen: Allows users to securely sign in with their credentials.
- POS Terminal Interface: Displays the product list, current order summary, and a clear "Place Order" button for quick order submission.
- Admin Dashboard: Provides tools for managing products, updating menus, and viewing sales reports.

- **Design Principles:**

- The layout follows a minimalist approach, avoiding clutter and focusing on essential actions.
- Color-coded buttons are used to help users quickly identify important functions and navigate efficiently.
- The interface is responsive and optimized for desktop screens, ensuring smooth operation on typical cafe computers.

2. Database design Models

1. User Collection

Table 2.11 : User Collection

Field	Type	Description
_id	ObjectId	Unique user ID
name	String	User's full name
email	String	User's email (unique)
phone	Number	User's 10-digit phone number
password	String	Hashed password
role	String	User role (e.g., admin, staff)

createdAt	Date	Account creation time
updatedAt	Date	Last update time

2. Table Collection

Table 2.12 : Table Collection

Field	Type	Description
_id	ObjectId	Unique table ID
tableNo	Number	Table number (unique)
status	String	Table status (Available, Occupied)
currentOrder	ObjectId	Reference to active Order (_id)

3. Order Collection

Table 2.13 : Order Collection

Field	Type	Description
_id	ObjectId	Unique order ID
customerDetails	Object	Customer info (name, phone, guests)
name	String	Customer name
phone	String	Customer phone
guests	Number	Number of guests
orderStatus	String	Status of the order (e.g., Pending, Completed)
orderDate	Date	Date and time of order
bills	Object	Billing info (total, tax, totalWithTax)
total	Number	Total amount before tax

tax	Number	Tax amount
totalWithTax	Number	Total amount including tax
items	Array	List of ordered items (product details or references)
table	ObjectId	Reference to Table (_id)
createdAt	Date	Created timestamp
updatedAt	Date	Last updated timestamp

Relationships Overview

- **Order** references **Table** via table field.
- **Table** references current active Order via **currentOrder.**
- **Users** have roles to separate **admin** and **staff** functionality.

2.3 Implementation

2.3.1 Tools and Development Environment

1. **IDE:** Visual Studio Code (VS Code) – used for front-end and back-end development.
2. **Frameworks/Libraries:**
   a. **Front-end**: React.js, Redux, Tailwind CSS
   b. **Back-end**: Node.js, Express.js, Mongoose (ODM)
3. **Database**: MongoDB (Local via MongoDB Atlas)
4. **Version Control**: Git with GitHub for repository management
5. **Postman**: For testing RESTful APIs
6. **Browser DevTools**: For real-time front-end debugging and inspection
7. **Development Hardware Setup**

- **Client Machine:**

1. Intel Core i5 or higher
2. 8GB RAM minimum
3. SSD Storage (256GB+)

- **Server Environment:**

1. **Backend Runtime:** Node.js (JavaScript runtime environment)
2. **Framework:** Express.js (for building the server and APIs)
3. **Database:** MongoDB (hosted on **MongoDB Atlas** – a cloud-based NoSQL database)
4. **Environment Hosting (Optional):** To be hosted locally.

2.3.2 System Modules and Their Implementation

Table 2.14 : System Modules

Module	Description & Functionality
User Authentication	Handles staff registration and login using JWT.
Order Management	Allows staff to create, update, and delete orders with item and customer details.
Table Booking	Enables booking of tables with validation on availability and scheduling.
Menu Management	Allows CRUD operations for menu items with real-time availability updates.
Billing and Payment	Generates printable bills with tax and total calculations.
Dashboard	Displays analytics like total sales/ revenue, active orders, and table status.

2.3.3 Coding Standards and Conventions

Naming Conventions:

- CamelCase for JavaScript variables and functions (orderDetails, handleLogin)
- PascalCase for React components (OrderForm, TableBooking)
- Snake_case for MongoDB field names where required

File Organization:

- Modular directory structure (components, services, routes, models, controllers)

Documentation:

- JSDoc-style comments above models and key functions to describe their purpose.
- Inline comments for logic explanation

Best Practices:

- Use of .env for environment variables
- **Validation:** Used validation for user inputs (email, phone number, required fields), which is important for data integrity.
- **Encryption:** Passwords are securely hashed using bcrypt before saving.
- **Timestamps:** Enabled timestamps for all models to automatically track creation and update times.

2.3.4 Security Considerations

- **Authentication:** JSON Web Tokens (JWT) used for session management.
- **Password Handling:** Encrypted using bcrypt before storage.
- **Aut**horization: Role-based access control (e.g., admin vs staff).
- **Data Validation:**
    - **Backend**: Express-validator & Mongoose schema validation
    - **Frontend**: Form validation (required fields, regex checks)

2.3.5 Testing and Debugging

.a Unit Testing: Conducted on individual functions (e.g., price calculation, login validation, table booking, taking order)

.b Integration Testing: Ensures seamless flow from order creation to sales recording

.c User Acceptance Testing (UAT): Conducted with café staff to verify ease of use and accuracy

1. Login Testing

Table 2.15 : Login Testing

Test case ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status
TC_Login_01	Valid login credentials	1.Enter registered email 2.Enter correct password 3. Click Login	Dashboard is loaded	Dashboard loaded	Pass
TC_Login_02	Invalid login credentials	1. Enter wrong email/password 2. Click Login	Error message "Invalid credentials"	Error shown	Pass
TC_Login_03	Empty fields on login form	1. Leave email and/or password blank 2. Click Login	Validation message displayed	Validation shown	Pass

3. Order Processing Testing

Table 2.16 :Order Processig Testing

Test case ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status
TC_Order_01	Place new order	1. Login as staff 2. Select items3. Click "Place Order"	Order saved and receipt generated	Success	Pass
TC_Order_02	Place order with no items	1. Login as staff 2. Click "Place Order" without selecting items	Validation message "No items selected"	Message shown	Pass
TC_Order_03	Modify on-processing order	1. Select existing order 2. Remove items clicking on remove icon	Order updated successfully	Updated	Pass

		3. Click "Place Order"			

4. Table Booking Testing

Table 2.17 : Security Testing

Test case ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status
TC_Book_01	Book available table	1. Create customer. Select available tables	Booking confirmed	Confirmed	Pass
TC_Book_02	Book already booked table	1. Select an already booked table.	Cant click booked table	Error shown	Pass

5. Billing Testing

Table 2.18 : Billing Testing

Testcase ID	Test Scenario	Tst Steps	Expected Output	Actual Output	Status
TC_UI_01	Navigation ease for new staff	1. Log in as a new user 2. Try to place an order	Task completed without external help	Task completed	Pass
TC_UI_02	Visibility of menu options	1. Log in and view dashboard	All options clearly visible and labeled	Clear layout	Pass
TC_UI_03	Font and color accessibility	1. Open on tablet/mobile	Fonts are readable and contrast is accessible	Clear and readable	Pass

6. Database Testing

Table 2.19:Database Testing

Testcase ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status
TC_DB_01	Data integrity after order	1. Place order 2. Query database	Order saved with all fields	Order data saved	Pass
TC_DB_02	Prevent duplicate orders	1. Try submitting same order twice rapidly	Only one entry in database	One order saved	Pass
TC_DB_03	Check valid foreign keys	1. Place order with invalid customer ID	Order is rejected	Order rejected	Pass

7. Security Testing

Table 2.20 : Login Testing

Testcase ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status
TC_Rec_01	Recovery from previous backup	1. Simulate system crash 2. Restore from last backup	System restores to last known good state	Recovery successful	Pass

5. Admin Testing

Table 2.21 : Login Testing

Testcase ID	Test Scenario	Test Steps	Expected Output	Actual Output	Status
TC_Rec_01	Recovery from	1. Simulate system	System restores to last	Recovery successful	Pass

	previous backup	crash 2. Restore from last backup	known good state		

2.4 Deployment

The POS system will be deployed locally on a desktop or laptop within the café premises. The backend server (Node.js) and the frontend (React.js) will run on the same machine, with MongoDB Atlas used as the remote database for easier data access and backup. The local deployment ensures fast access, low latency, and offline accessibility when internet is unavailable. The application can be accessed through a browser on the same device or across devices within the same network using the server's local IP. Proper instructions for starting the server and accessing the POS interface will be provided for the café staff.

CHAPTER III CONCLUSION AND RECOMMENDATION

3.1 Summary

This project focused on the design and development of a Point of Sale (POS) system tailored for café management. Using Agile methodology, the development proceeded through iterative phases, including requirement analysis, modular planning, UI/UX design, backend implementation, and testing. The system streamlines order processing, automates billing, and generates sales reports through a user-friendly interface. Technologies like React.js, Node.js, and MongoDB were used to ensure scalability, responsiveness, and ease of maintenance. The project addressed the practical needs of café operations while maintaining simplicity for end-users

3.1.1 Findings

- Developed a POS system tailored for small cafés.
- Streamlined order management, automated billing, and generated sales reports.
- Created an intuitive interface using React.js and Node.js, backed by MySQL

3.1.2 Limitations of the Current System

- No integration with physical hardware (e.g., receipt printers).
- Designed for single-user operation; does not support multi-terminal use.
- Lacks features like customer loyalty programs and real-time stock alerts

3.1.3 Future Enhancements

- Add support for multi-user access and real-time updates.
- Integrate external hardware like barcode scanners and receipt printers.
- Implement modules for customer loyalty, analytics, and mobile/tablet POS interface.

3.1.4 Lessons Learned

- **Technical:** Improved skills in full-stack development, database design, and modular coding using Agile methodology.
- **Non-technical:** Enhanced teamwork, project planning, user-focused design, and adaptability to client feedback.

3.2 Conclusion

The developed POS system effectively meets the core requirements of small to medium-sized cafés by simplifying transactions, providing a structured admin panel, and automating billing. The system proved functionally sound through testing and user feedback, demonstrating reliable performance, ease of use, and clear reporting functionalities. Although limited to single-device usage , the system lays a solid foundation for future enhancements.

3.3 Recommendation

To further improve the system, the following recommendations are proposed:

- Integrate real receipt printers and barcode scanners for full hardware functionality.
- Extend multi-user access and real-time inventory updates across multiple terminals.
- Implement data analytics features to offer sales trends and customer insights.
- Develop mobile or tablet support to enhance portability and counter space efficiency.