Software Design Document

for

Namal Mess Management System

Students: Muhammad Raqib Hayat, Abu Bakar

Instructor: Muhammad Ali Shahid

Course: CS-260 - Software Engineering

Semester: 6th, Spring 2025

Institution: Namal University, Mianwali

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Overview . . . 3
1.4 Reference Material . . . 4
1.5 Definitions and Acronyms . . . 4

2 System Overview **4**
2.1 System Context . . . 4
2.2 User Roles and Interfaces . . . 5

3 System Architecture **6**
3.1 Architectural Design . . . 6
3.1.1 Client-Server Architecture . . . 7
3.1.2 Model-View-Controller (MVC) Pattern . . . 7
3.2 Decomposition Description . . . 8
3.2.1 Authentication Subsystem . . . 8
3.2.2 Menu Management Subsystem . . . 8
3.2.3 Order Processing Subsystem . . . 8
3.2.4 Payment Subsystem . . . 8
3.2.5 Notification Subsystem . . . 9
3.2.6 Reporting and Analytics Subsystem . . . 9
3.2.7 User Profile Management Subsystem . . . 9
3.3 Design Rationale . . . 9
3.3.1 Client-Server Architecture Selection . . . 9
3.3.2 MVC Pattern Selection . . . 9
3.3.3 Firebase Platform Selection . . . 9

4 Data Design **10**
4.1 Data Description . . . 10
4.1.1 Database Structure . . . 11
4.1.2 Data Relationships . . . 11
4.2 Data Dictionary . . . 11

5 Component Design **13**
5.1 Module: Authentication . . . 13
5.1.1 Responsibilities . . . 13
5.1.2 Interfaces . . . 13
5.1.3 Implementation Details . . . 14
5.2 Module: Order Management . . . 14
5.2.1 Responsibilities . . . 14
5.2.2 Interfaces . . . 14
5.2.3 Implementation Details . . . 15
5.3 Module: Payment System . . . 17
5.3.1 Responsibilities . . . 17
5.3.2 Interfaces . . . 18

5.3.3 Implementation Details . . . 18
5.4 Module: Notification System . . . 19
5.4.1 Responsibilities . . . 19
5.4.2 Interfaces . . . 19
5.4.3 Implementation Details . . . 19

6 Human Interface Design 20
6.1 Overview of User Interface . . . 20
6.2 Screen Images . . . 21
6.2.1 Desktop Screens: . . . 21
6.2.2 Mobile Screens: . . . 23
6.3 Screen Objects and Actions . . . 26
6.3.1 Mobile Application Interface . . . 26
6.3.2 Web Portal Interface . . . 27

7 Requirements Traceability Matrix 28

8 Appendices 30
8.1 Appendix A: Use Case Diagram . . . 30
8.2 Appendix B: Data Dlow Diagrams . . . 31
8.3 Appendix C: System Sequence Diagram . . . 32
8.4 Appendix D: Sequence Diagrams . . . 34
8.5 Appendix E: Class Diagram . . . 37
8.6 Appendix F: Glossary . . . 38

1 Introduction

1.1 Purpose

This Software Design Document (SDD) describes the architectural and detailed design for the Mess Management System at Namal University. It translates the requirements specified in the Software Requirements Specification (SRS) into a design model that can be used as a blueprint for system implementation. This document is intended to provide developers with sufficient information to develop the system without further design input.

1.2 Scope

The design described in this document covers the entire Mess Management System, including:

- Mobile application for students and faculty
- Web-based administrative portal
- Backend services and database design
- Integration with payment systems
- Notification mechanisms

The design encompasses all functional modules identified in the SRS, including user authentication, menu management, order processing, payment handling, reporting, and notifications.

1.3 Overview

This document is organized according to the IEEE STD 1016 format:

- Section 1 provides introductory information about the document.
- Section 2 gives a high-level overview of the system.
- Section 3 details the system architecture, including architectural patterns and component decomposition.
- Section 4 describes the data design, including database structure and data dictionary.
- Section 5 provides detailed component design.
- Section 6 presents the human interface design with UI mockups.
- Section 7 contains the requirements traceability matrix.
- Section 8 includes appendices with supplementary information.

1.4 Reference Material

1. Software Requirements Specification for Mess Management System, Version 2.0, May 15, 2025
2. IEEE Standard for Software Design Descriptions (IEEE Std 1016-2009)
3. Flutter Documentation, `https://flutter.dev/docs`
4. React.js Documentation, `https://reactjs.org/docs`
5. Firebase Documentation, `https://firebase.google.com/docs`

1.5 Definitions and Acronyms

- **API:** Application Programming Interface
- **CRUD:** Create, Read, Update, Delete
- **DFD:** Data Flow Diagram
- **MVC:** Model-View-Controller
- **REST:** Representational State Transfer
- **SDD:** Software Design Document
- **SRS:** Software Requirements Specification
- **UI:** User Interface
- **UX:** User Experience
- **JWT:** JSON Web Token

2 System Overview

2.1 System Context

The Mess Management System is designed to modernize food service management at Namal University by replacing the traditional cash-based mess system with a digital solution. The system consists of two main components:

1. **Mobile Application**: Developed using Flutter, this cross-platform application allows students and faculty to browse menus, place orders, make payments, and track order status.
2. **Administrative Web Portal**: Developed using React.js, this web application enables mess managers, menu managers, and kitchen staff to manage menus, process orders, generate reports, and handle administrative tasks.

Both components are supported by a common backend infrastructure built on Firebase, which provides authentication, real-time database, cloud functions, and storage services. The system follows a client-server architecture with the MVC design pattern to ensure separation of concerns and maintainability.

Key features of the system include:

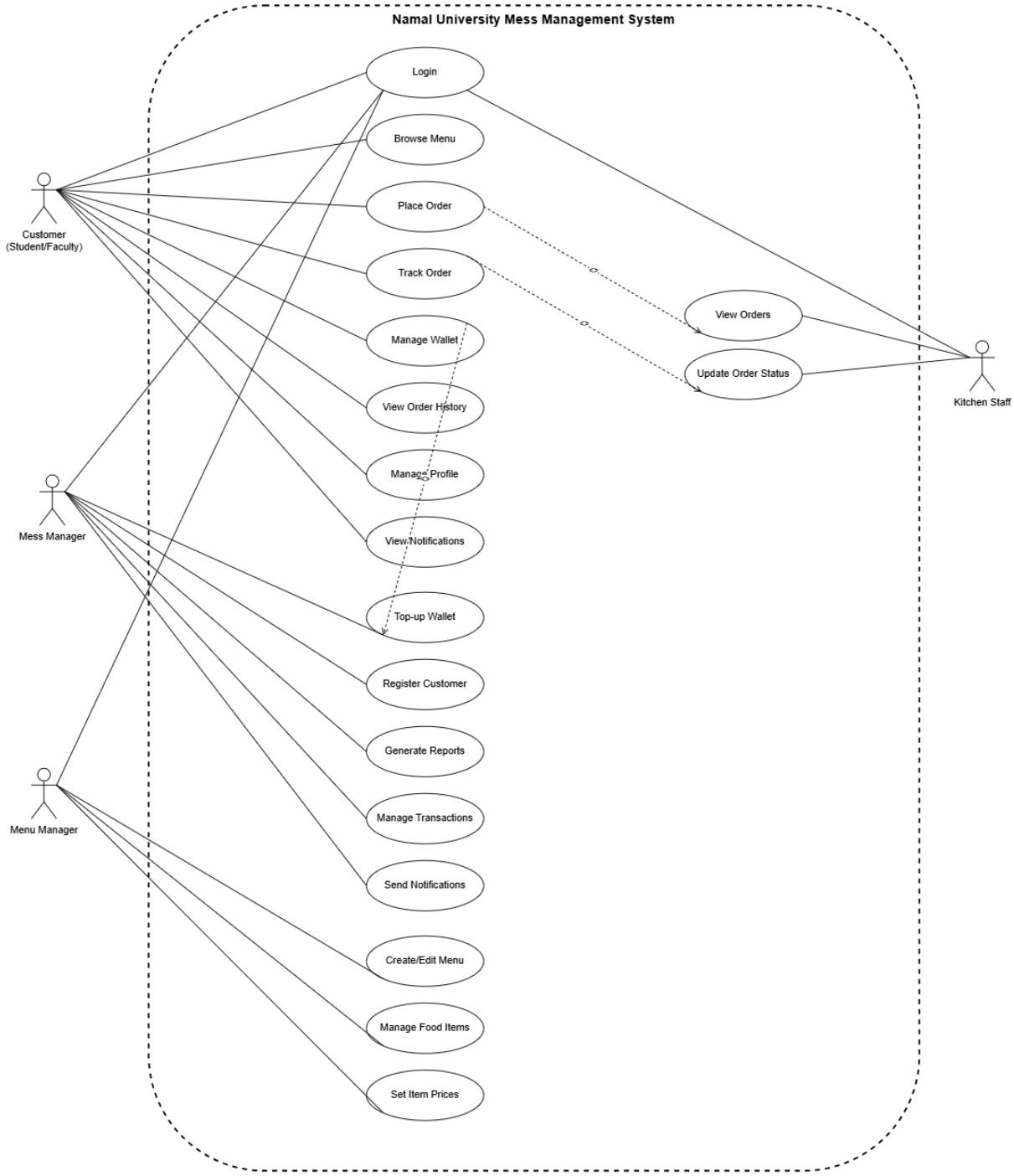
- User authentication and role-based access control
- Digital menu management and browsing
- Order placement, tracking, and management
- Digital wallet for cashless transactions
- Real-time notifications
- Reporting and analytics

2.2 User Roles and Interfaces

The system supports the following user roles, each with specific interfaces and permissions:

1. **Students/Faculty (Customers)**
    - Browse menu items
    - Place and track orders
    - Manage digital wallet
    - View order history
    - Receive notifications
    - Manage profile
2. **Mess Manager**
    - Generate reports
    - Manage transactions
    - View order history
    - Send notifications
    - Manage profiles
3. **Menu Manager**
    - Create and edit menus
    - Manage food items
    - Set item prices
4. **Kitchen Staff**
    - View pending orders

- Update order status
- Deliver orders



3 System Architecture

3.1 Architectural Design

The Mess Management System follows a client-server architecture with the Model-View-Controller (MVC) design pattern. This architecture separates the application into three interconnected components to separate internal representations of information from the ways information is presented to and accepted from the user.

3.1.1 Client-Server Architecture

The client-server architecture divides the system into two main parts:

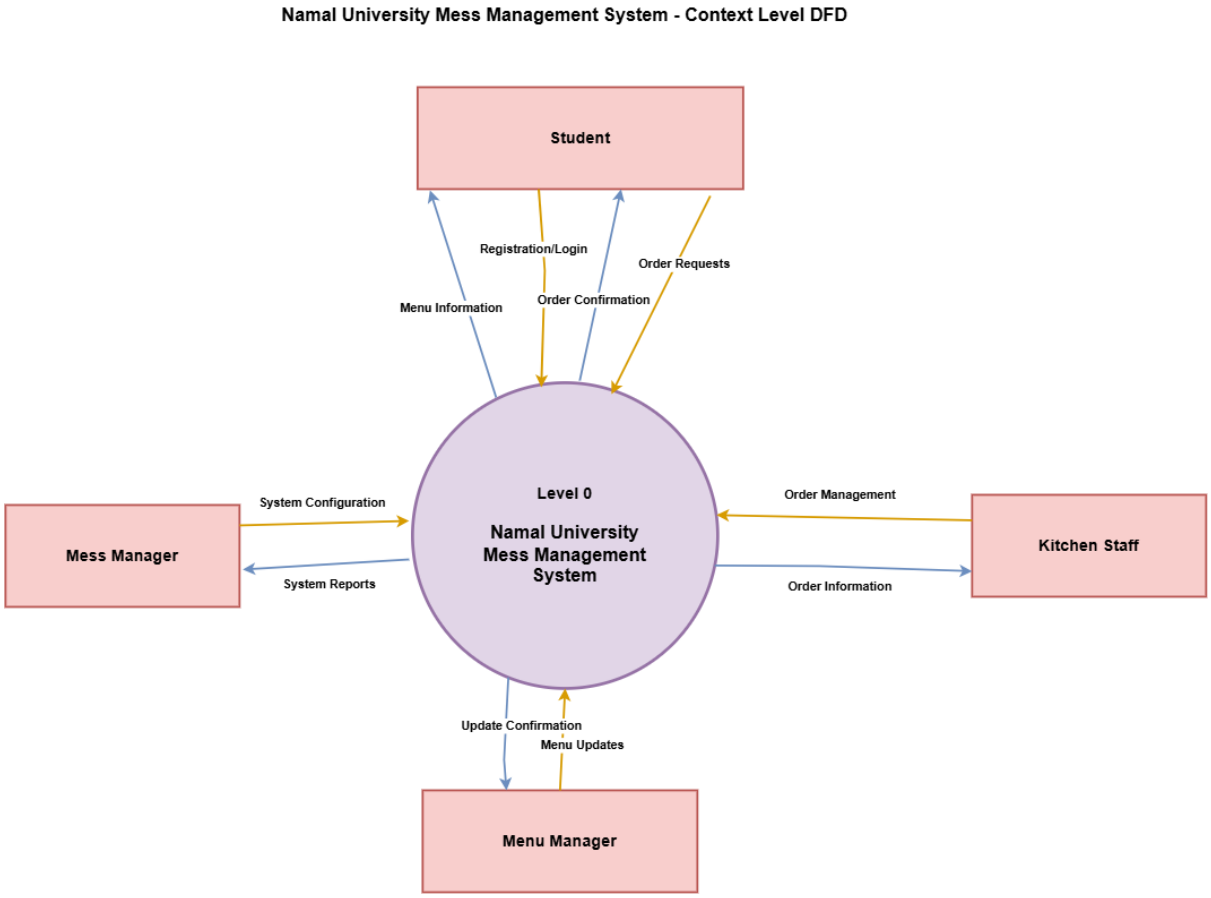
- **Client Side**:
  - Mobile Application (Flutter)
  - Web Portal (React.js)
- **Server Side**:
  - Firebase Authentication
  - Firebase Firestore (NoSQL Database)
  - Firebase Cloud Functions
  - Firebase Cloud Storage

3.1.2 Model-View-Controller (MVC) Pattern

The MVC pattern is implemented as follows:

- **Model**: Represents the data structure and business logic. In this system, models include User, Menu, Order, Payment, and other domain-specific classes.
- **View**: Represents the UI components that display information to users and capture user input. This includes Flutter widgets for the mobile app and React components for the web portal.
- **Controller**: Acts as an intermediary between Model and View, processing user input, manipulating data using the Model, and updating the View. Controllers handle authentication, order processing, payment processing, and other system operations.

3.2 Decomposition Description



The system is decomposed into the following major subsystems:

3.2.1 Authentication Subsystem

Handles user registration, login, logout, and role-based access control. This subsystem leverages Firebase Authentication for secure user management.

3.2.2 Menu Management Subsystem

Enables the creation, updating, and deletion of menu items by authorized personnel. It also handles menu categorization, pricing, and availability scheduling.

3.2.3 Order Processing Subsystem

Manages the entire order lifecycle, from placement to delivery. This includes cart management, order submission, payment processing, order tracking, and status updates.

3.2.4 Payment Subsystem

Handles all financial transactions, including payment processing, and transaction history. It ensures secure and reliable payment operations.

3.2.5 Notification Subsystem

Delivers real-time notifications to users about order status, promotions, and system updates using Firebase Cloud Messaging.

3.2.6 Reporting and Analytics Subsystem

Generates reports on sales, popular items, peak ordering times, and other metrics to support data-driven decision-making.

3.2.7 User Profile Management Subsystem

Allows users to view and update their profile information, preferences, and settings.

3.3 Design Rationale

3.3.1 Client-Server Architecture Selection

The client-server architecture was chosen for the following reasons:

- Enables centralized data management and consistent business logic
- Supports multiple client types (mobile and web) with a common backend
- Facilitates scalability by allowing independent scaling of client and server components
- Enhances security by implementing authentication and authorization at the server level

3.3.2 MVC Pattern Selection

The MVC pattern was selected for the following reasons:

- Promotes separation of concerns, making the codebase more maintainable
- Facilitates parallel development of UI, business logic, and data access components
- Supports code reuse across different parts of the application
- Aligns well with both Flutter and React.js development paradigms

3.3.3 Firebase Platform Selection

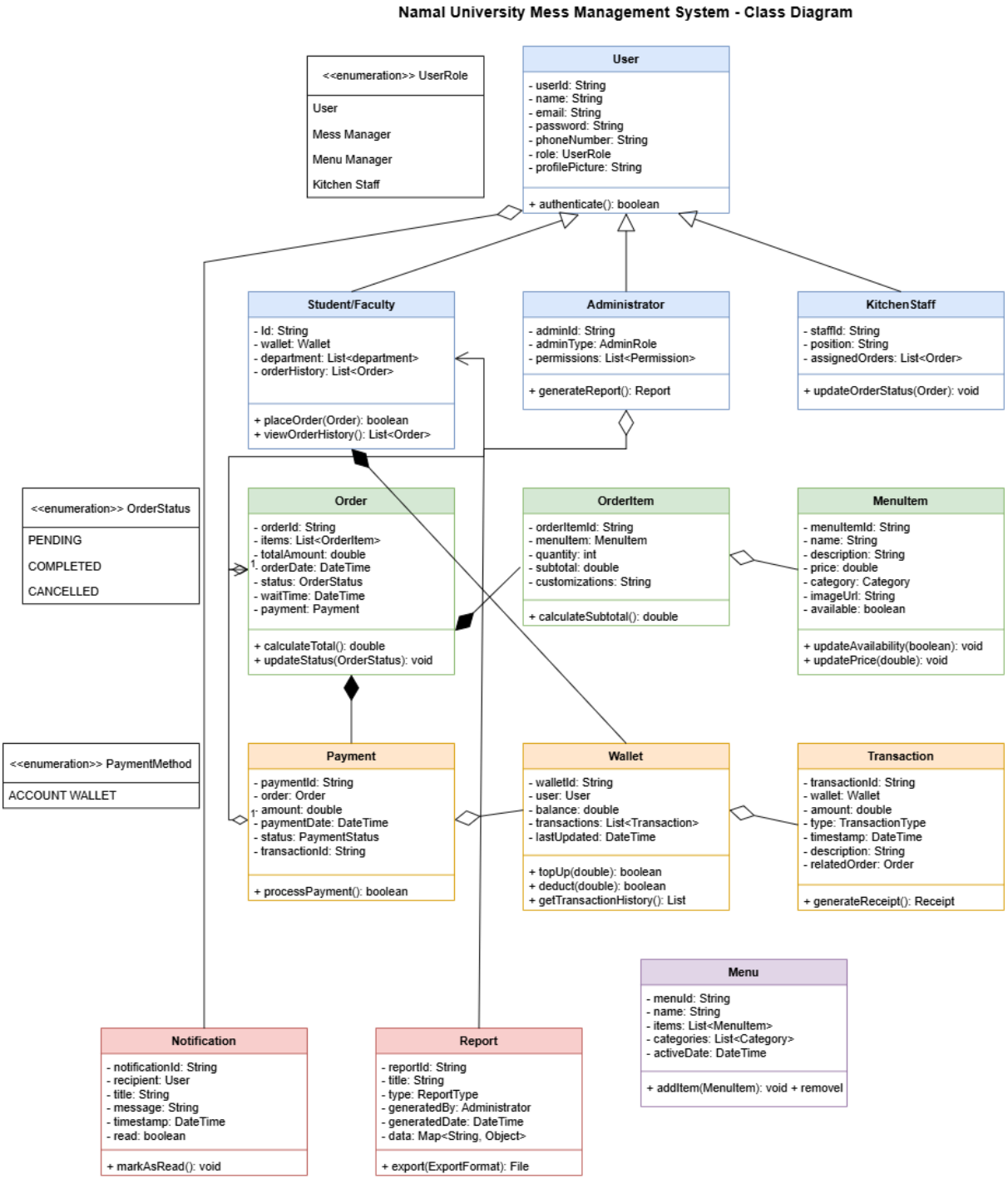
Firebase was chosen as the backend platform for the following reasons:

- Provides a comprehensive suite of services (authentication, database, storage, messaging)
- Offers real-time data synchronization, essential for order tracking and notifications
- Eliminates the need for custom server infrastructure, reducing development and maintenance costs
- Scales automatically to handle varying loads
- Integrates well with both Flutter and React.js

4 Data Design

4.1 Data Description

The data design for the Mess Management System is based on a NoSQL document-oriented database model using Firebase Firestore. This approach was chosen for its flexibility, scalability, and real-time capabilities, which align well with the system requirements.



4.1.1 Database Structure

The database is organized into the following collections:

- **users**: Stores user information, including profile details and role
- **menus**: Contains menu information, including active dates and categories
- **menuItems**: Stores individual food items with details like name, description, price, and availability
- **orders**: Contains order information, including items, status, and timestamps
- **orderItems**: Stores individual items within orders, including customizations
- **payments**: Records payment transactions and their status
- **wallets**: Maintains wallet balances and transaction history for users
- **notifications**: Stores notifications sent to users
- **reports**: Stores generated reports and analytics data

4.1.2 Data Relationships

Key relationships between data entities include:

- One-to-many relationship between users and orders
- One-to-many relationship between users and wallets
- One-to-many relationship between menus and menuItems
- One-to-many relationship between orders and orderItems
- One-to-many relationship between users and notifications

4.2 Data Dictionary

Name	Type	Description
User		
userId	String	Unique identifier for the user
name	String	User's full name
email	String	User's email address
password	String	Encrypted password
phoneNumber	String	User's contact number
role	Enum	User role (Student/Faculty, Mess Manager, Menu Manager, Kitchen Staff)
profilePicture	String	URL to user's profile image

department	String	User's department (for students/faculty)
Menu		
menuId	String	Unique identifier for the menu
name	String	Menu name
activeDate	DateTime	Date when the menu is active
categories	List¡String¿	List of food categories in the menu
items	List¡MenuItem¿	List of menu items
MenuItem		
menuItemId	String	Unique identifier for the menu item
name	String	Item name
description	String	Item description
price	Double	Item price
category	String	Item category
imageUrl	String	URL to item image
available	Boolean	Whether the item is available
Order		
orderId	String	Unique identifier for the order
items	List¡OrderItem¿	List of order items
totalAmount	Double	Total order amount
orderDate	DateTime	Date and time of order placement
status	Enum	Order status (PENDING, COMPLETED, CANCELLED)
waitTime	DateTime	Estimated wait time
payment	Payment	Payment information
OrderItem		
orderItemId	String	Unique identifier for the order item
menuItem	MenuItem	Reference to the menu item
quantity	Integer	Quantity ordered
subtotal	Double	Subtotal for this item
customizations	String	Special instructions or customizations
Payment		
paymentId	String	Unique identifier for the payment
order	Order	Reference to the order
amount	Double	Payment amount
paymentDate	DateTime	Date and time of payment
status	Enum	Payment status
transactionId	String	External transaction identifier
Wallet		
walletId	String	Unique identifier for the wallet
user	User	Reference to the user
balance	Double	Current wallet balance

transactions	List¡Transaction¿	List of transactions
lastUpdated	DateTime	Last update timestamp
Transaction		
transactionId	String	Unique identifier for the transaction
wallet	Wallet	Reference to the wallet
amount	Double	Transaction amount
type	Enum	Transaction type
timestamp	DateTime	Transaction timestamp
description	String	Transaction description
relatedOrder	Order	Reference to related order (if applicable)
Notification		
notificationId	String	Unique identifier for the notification
recipient	User	Reference to the recipient user
title	String	Notification title
message	String	Notification message
timestamp	DateTime	Notification timestamp
read	Boolean	Whether the notification has been read

5 Component Design

5.1 Module: Authentication

5.1.1 Responsibilities

- User registration and account creation
- User login and authentication
- Password reset and recovery
- Session management
- Role-based access control

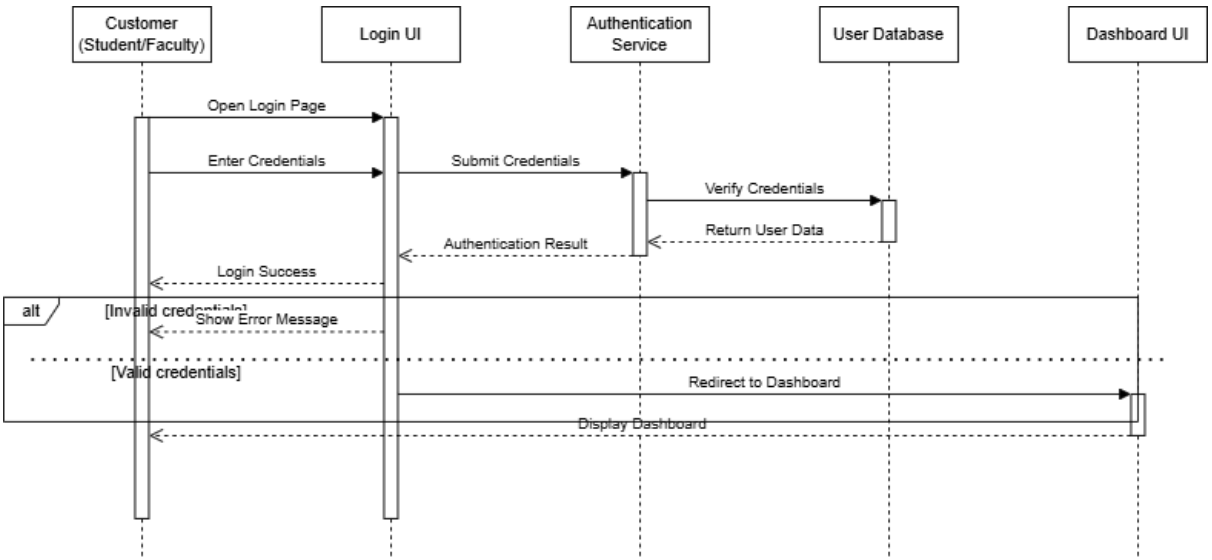
5.1.2 Interfaces

- `authenticate(email, password): User` - Authenticates a user and returns user data
- `register(userData): User` - Registers a new user
- `resetPassword(email): boolean` - Initiates password reset process
- `logout(): boolean` - Logs out the current user
- `getCurrentUser(): User` - Returns the currently authenticated user

- `updateUserProfile(userData): User` - Updates user profile information

5.1.3 Implementation Details

The Authentication component uses Firebase Authentication for user management. It implements JWT-based authentication with secure token storage and automatic token refresh. The component enforces role-based access control through Firebase security rules and custom claims.



5.2 Module: Order Management

5.2.1 Responsibilities

- Handle order creation and submission
- Manage ordering cart functionality
- Process order status updates
- Calculate order totals and wait times
- Track order history
- Menu browsing and item selection

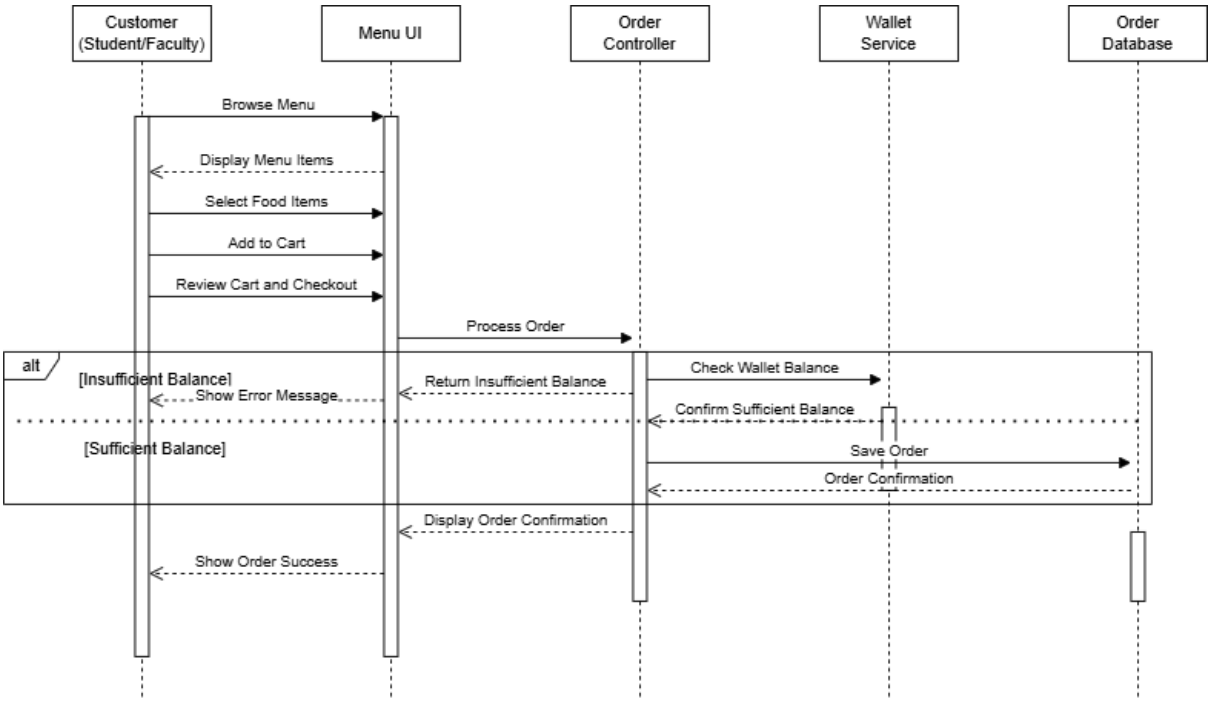
5.2.2 Interfaces

- `createOrder(orderData): Order` - Creates a new order
- `getOrder(id): Order` - Retrieves a specific order
- `updateOrderStatus(id, status): Order` - Updates order status
- `getOrderHistory(userId): List<Order>` - Retrieves order history for a user
- `cancelOrder(id): boolean` - Cancels an order

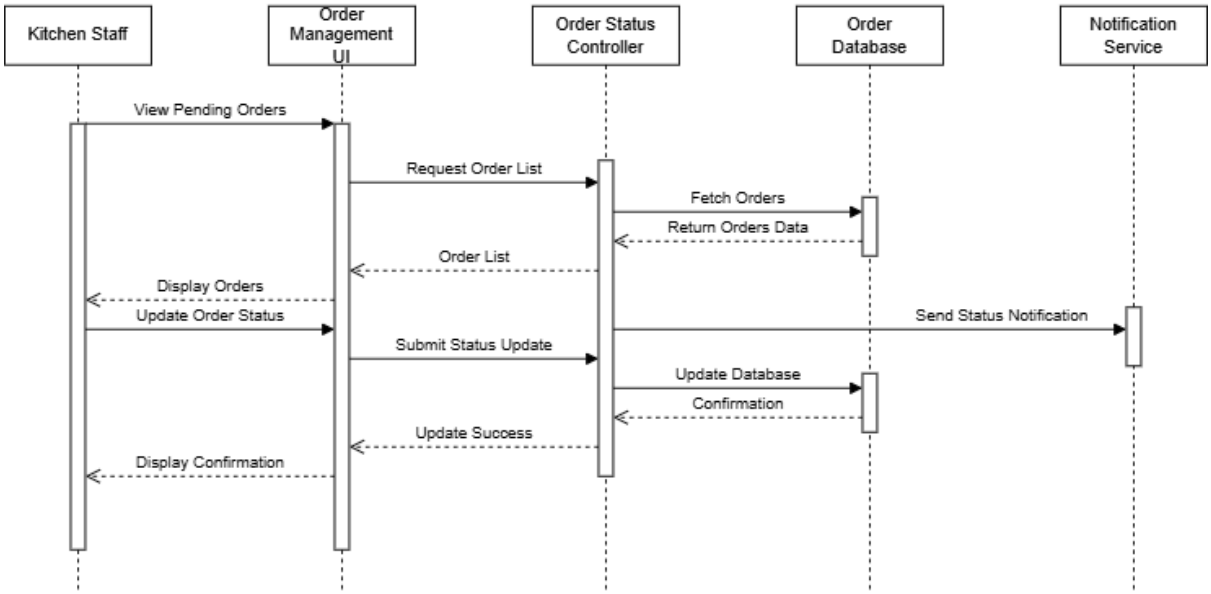
- `addToCart(item): Cart` - Adds an item to the shopping cart
- `removeFromCart(item): Cart` - Removes an item from the shopping cart
- `getMenus(date): List<Menu>` - Retrieves menus for a specific date
- `getMenuItem(id): MenuItem` - Retrieves a specific menu item
- `createMenuItem(itemData): MenuItem` - Creates a new menu item
- `updateMenuItem(id, itemData): MenuItem` - Updates an existing menu item
- `deleteMenuItem(id): boolean` - Deletes a menu item
- `setItemAvailability(id, available): boolean` - Updates item availability

5.2.3 Implementation Details

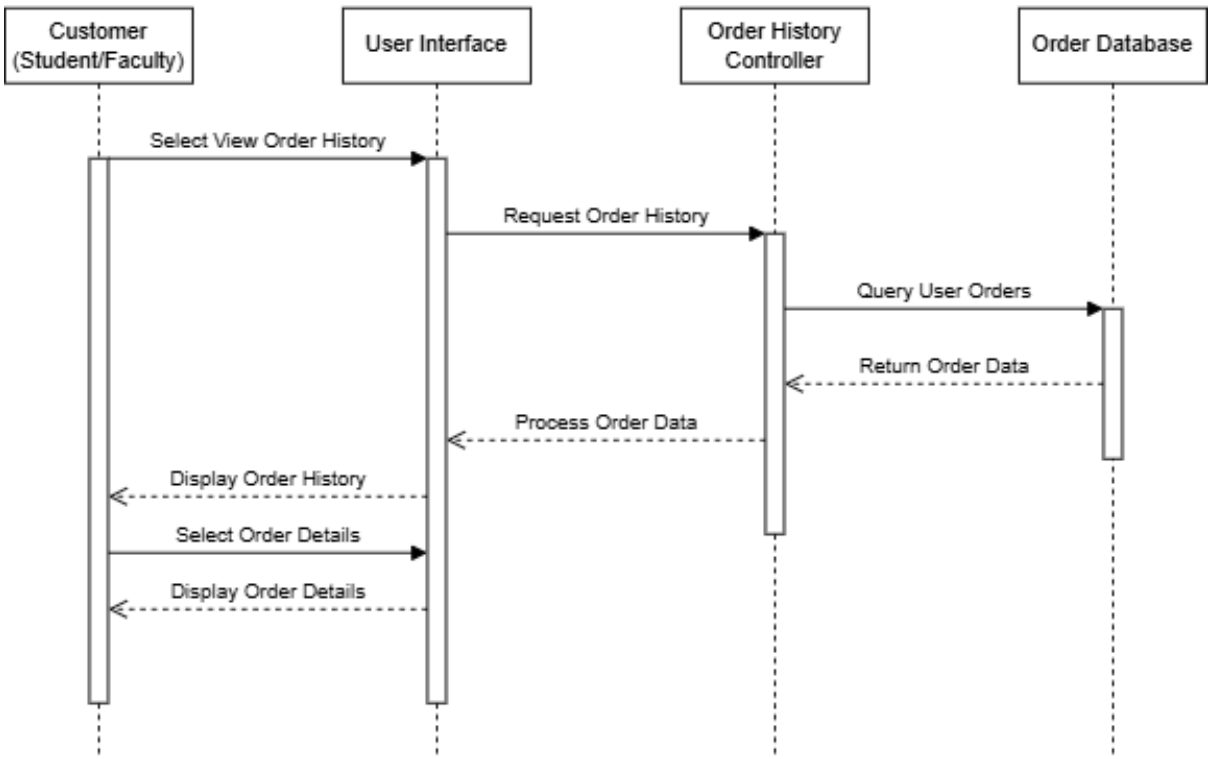
The Order Management component uses Firestore for order storage and real-time updates. It implements transaction management to ensure data consistency when updating order status and inventory. The component includes optimistic UI updates with server validation for a responsive user experience.

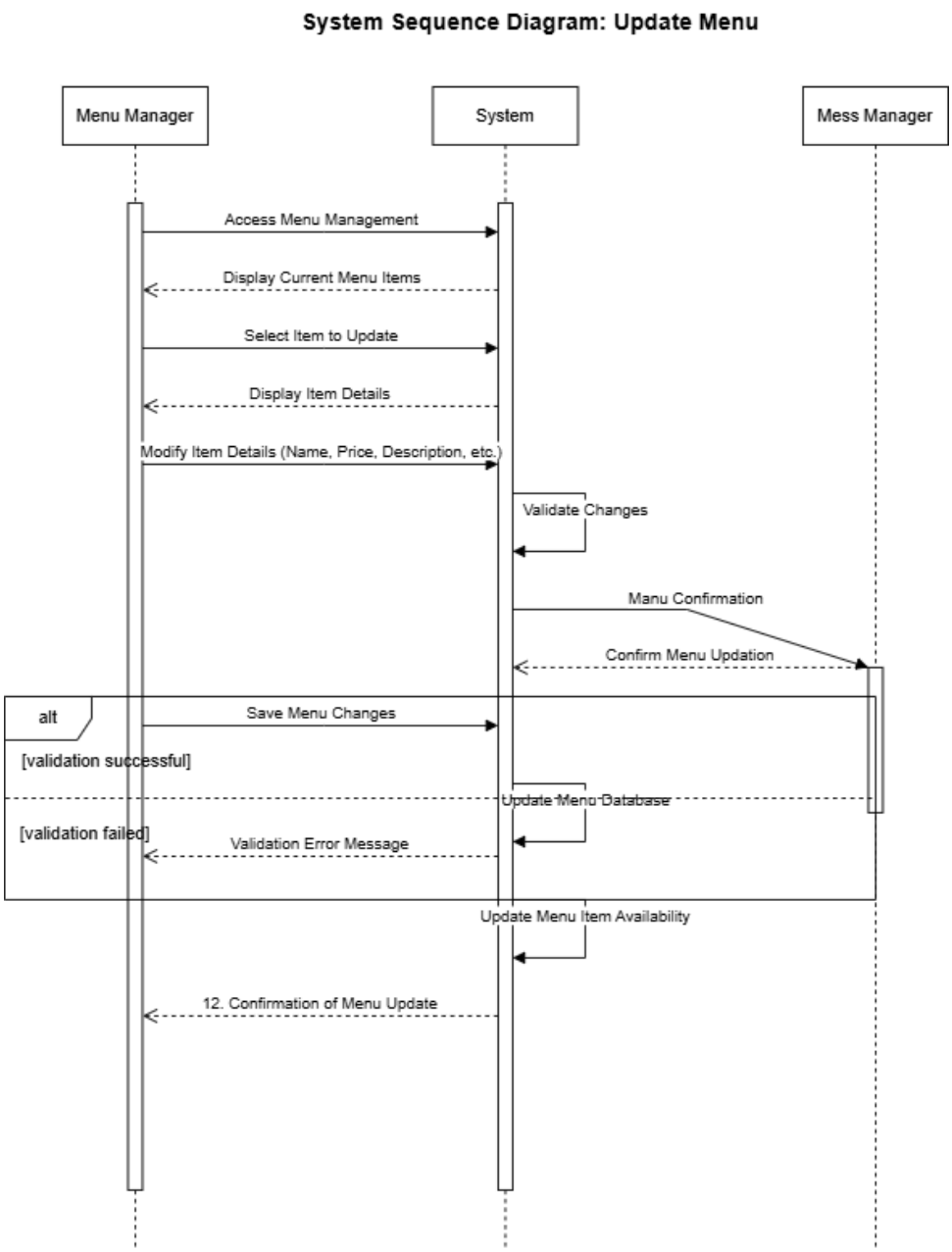


Update Order Menu - Sequential Diagram



View History - Sequential Diagram





5.3 Module: Payment System

5.3.1 Responsibilities

- Process payments for orders
- Manage wallet balances
- Handle wallet top-up operations
- Record transaction history
- Generate payment receipts

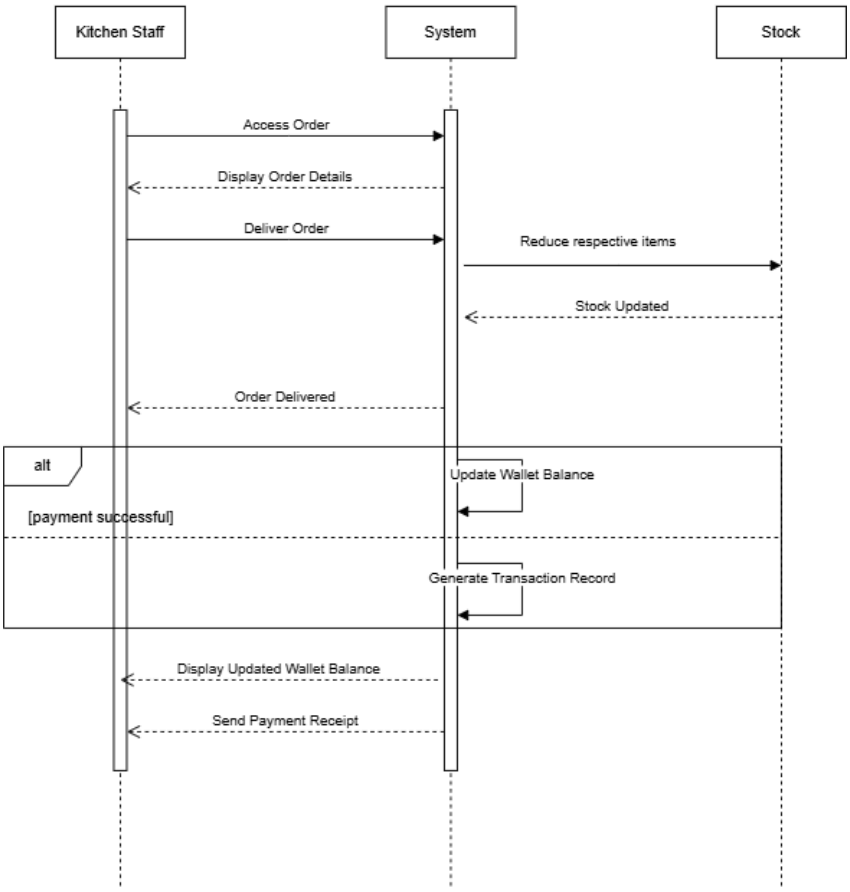
5.3.2 Interfaces

- `processPayment(orderId): Payment` - Processes payment for an order
- `getWalletBalance(userId): double` - Retrieves wallet balance for a user
- `topUpWallet(userId, amount): Wallet` - Adds funds to a user's wallet
- `deductFromWallet(userId, amount): boolean` - Deducts funds from a user's wallet
- `getTransactionHistory(userId): List<Transaction>` - Retrieves transaction history
- `generateReceipt(transactionId): Receipt` - Generates a receipt for a transaction

5.3.3 Implementation Details

The Payment component uses Firestore for wallet and transaction data storage. It implements atomic transactions to ensure financial data consistency. The component includes validation checks for sufficient balance before processing payments and integrates with Firebase Cloud Functions for complex payment operations.

System Sequence Diagram: Make Payment (Order Delivery)



5.4 Module: Notification System

5.4.1 Responsibilities

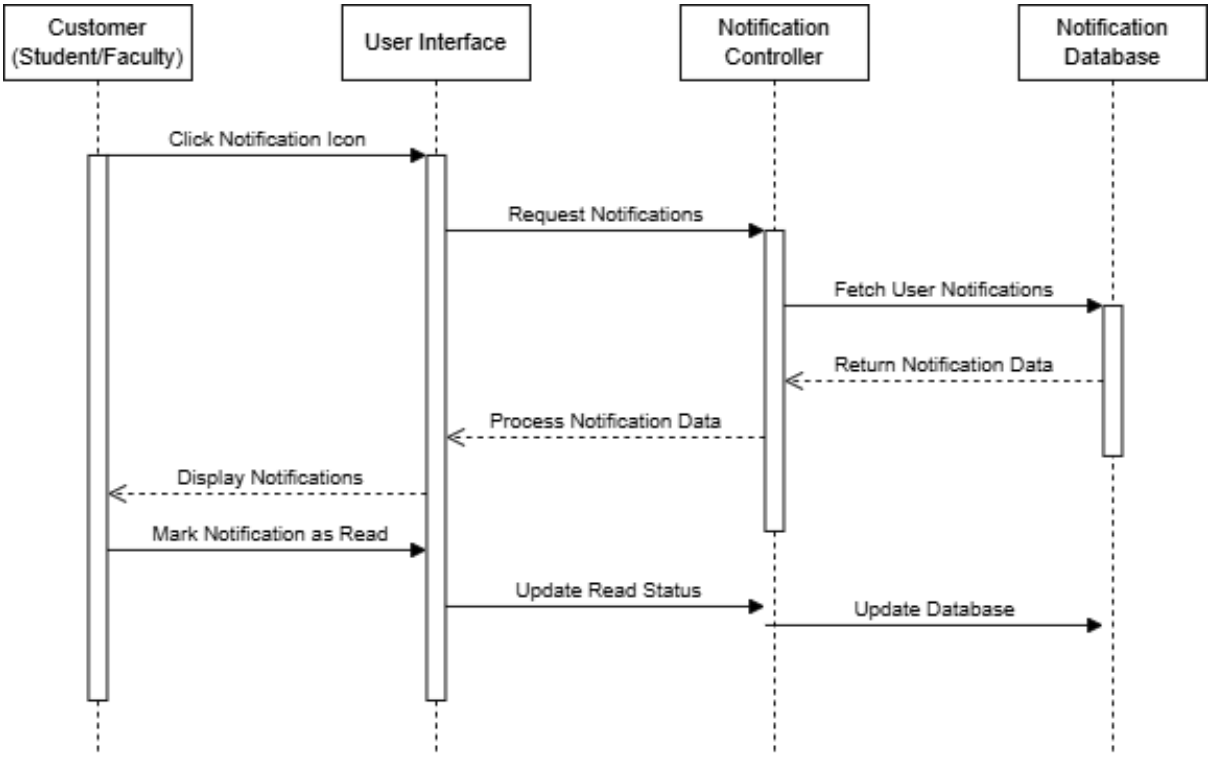
- Send real-time notifications to users
- Store and manage notification history
- Track notification read status
- Support different notification types (order updates, promotions, etc.)

5.4.2 Interfaces

- `sendNotification(userId, notification): boolean` - Sends a notification to a user
- `getNotifications(userId): List<Notification>` - Retrieves notifications for a user
- `markAsRead(notificationId): boolean` - Marks a notification as read
- `deleteNotification(notificationId): boolean` - Deletes a notification

5.4.3 Implementation Details

The Notification component uses Firebase Cloud Messaging for real-time notifications and Firestore for notification storage. It implements background notification handling for both mobile and web clients. The component includes support for notification grouping and prioritization.



6 Human Interface Design

6.1 Overview of User Interface

The Mess Management System features two primary user interfaces:

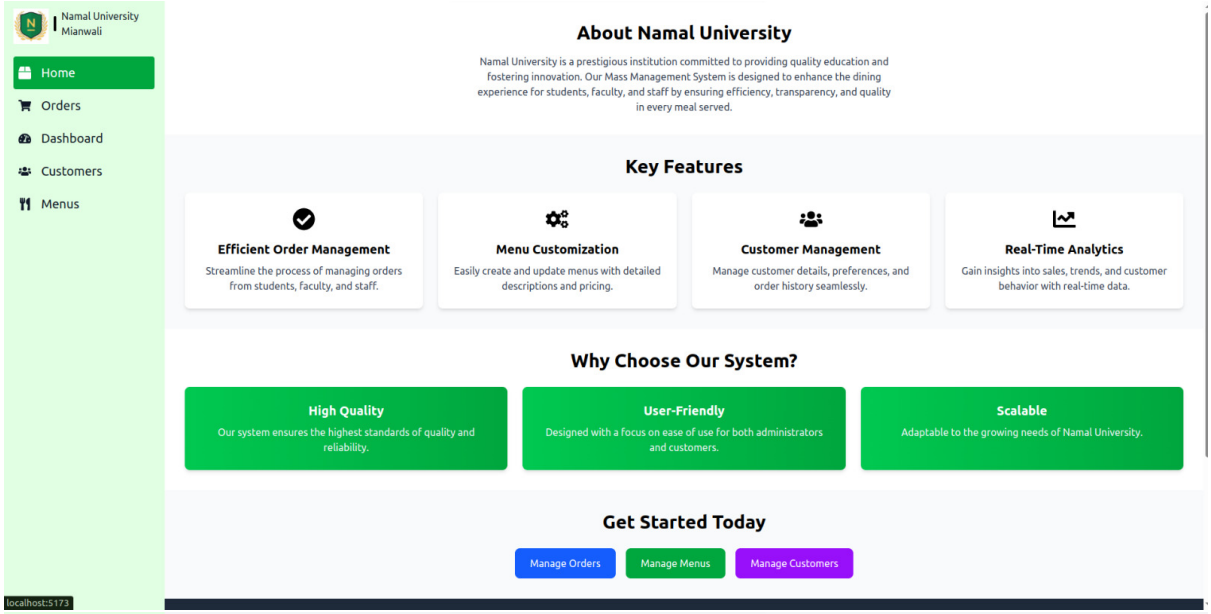
1. **Mobile Application Interface** - Designed for students and faculty to browse menus, place orders, and manage their accounts.
2. **Web Portal Interface** - Designed for administrative users (mess managers, menu managers, kitchen staff) to manage the system.

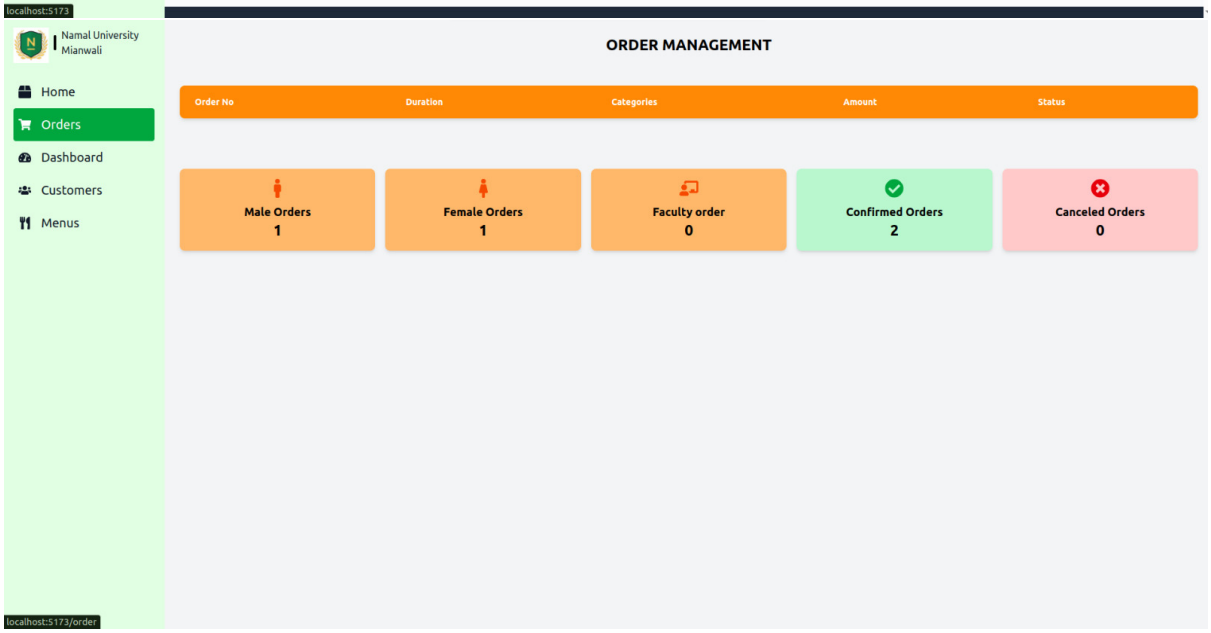
Both interfaces follow these design principles:

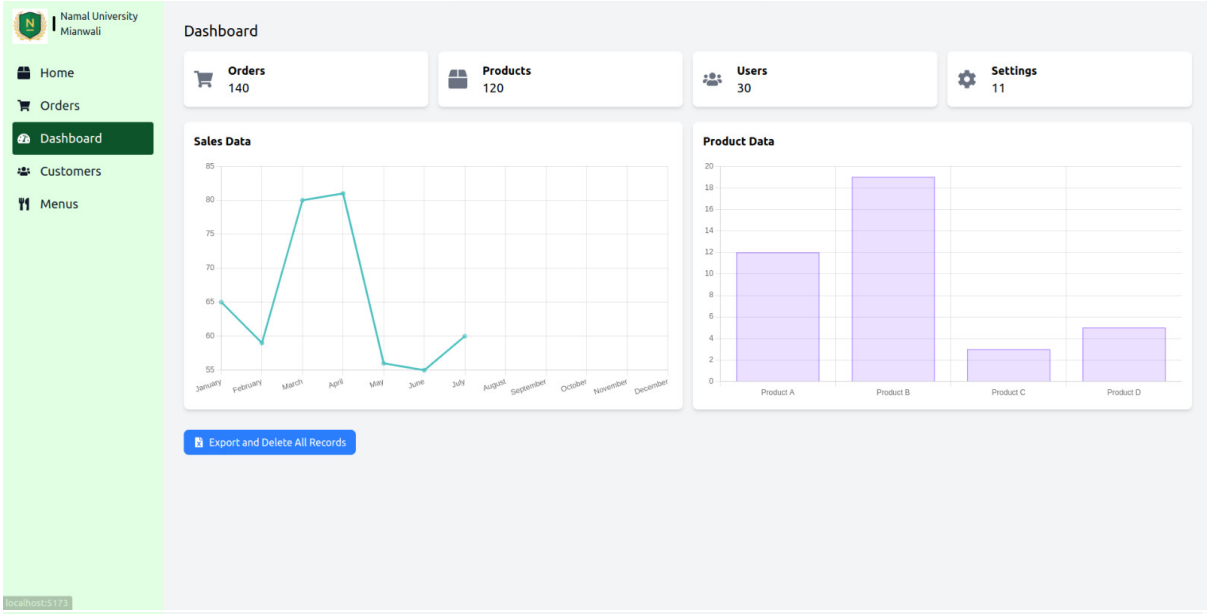
- **Consistency**: Maintaining consistent visual elements, interaction patterns, and terminology across the system.
- **Simplicity**: Focusing on essential features and minimizing cognitive load through clean, uncluttered layouts.
- **Responsiveness**: Ensuring the interfaces adapt to different screen sizes and orientations.
- **Accessibility**: Supporting users with disabilities through proper contrast, text sizing, and screen reader compatibility.
- **Feedback**: Providing clear feedback for user actions through visual cues, animations, and notifications.

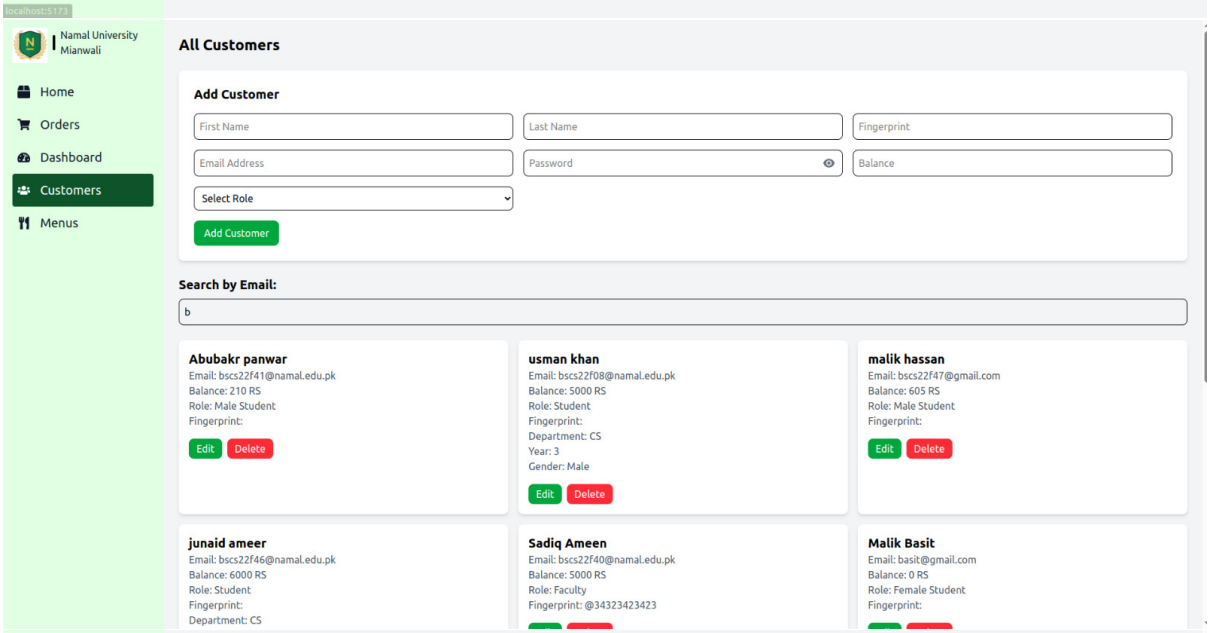
6.2 Screen Images

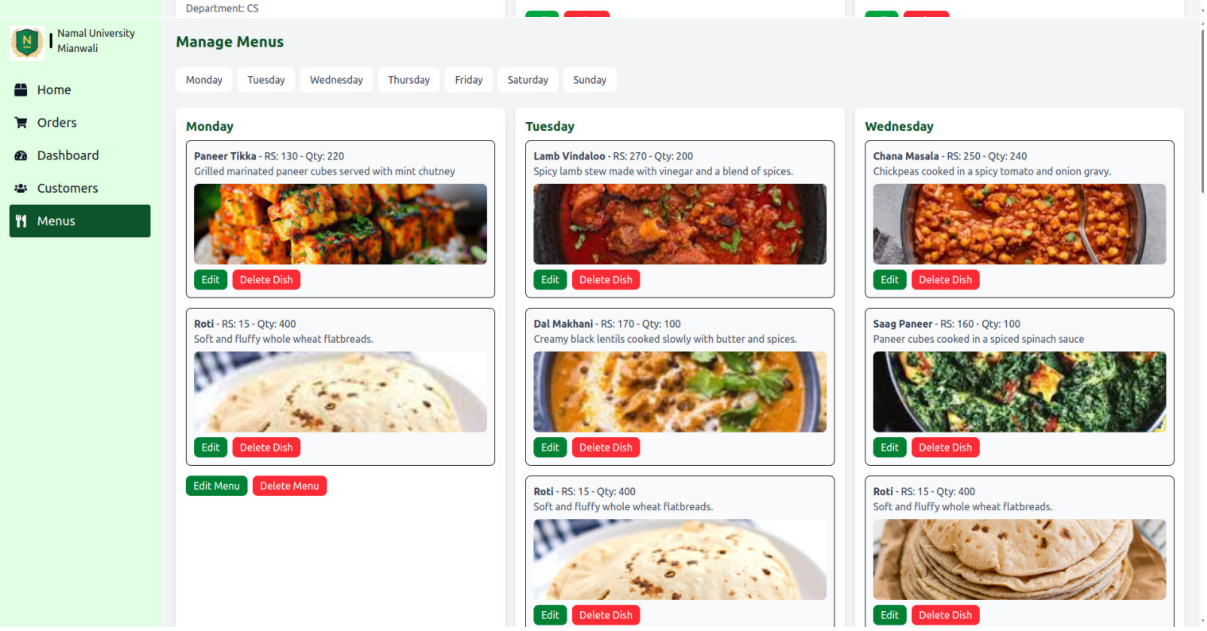
6.2.1 Desktop Screens:



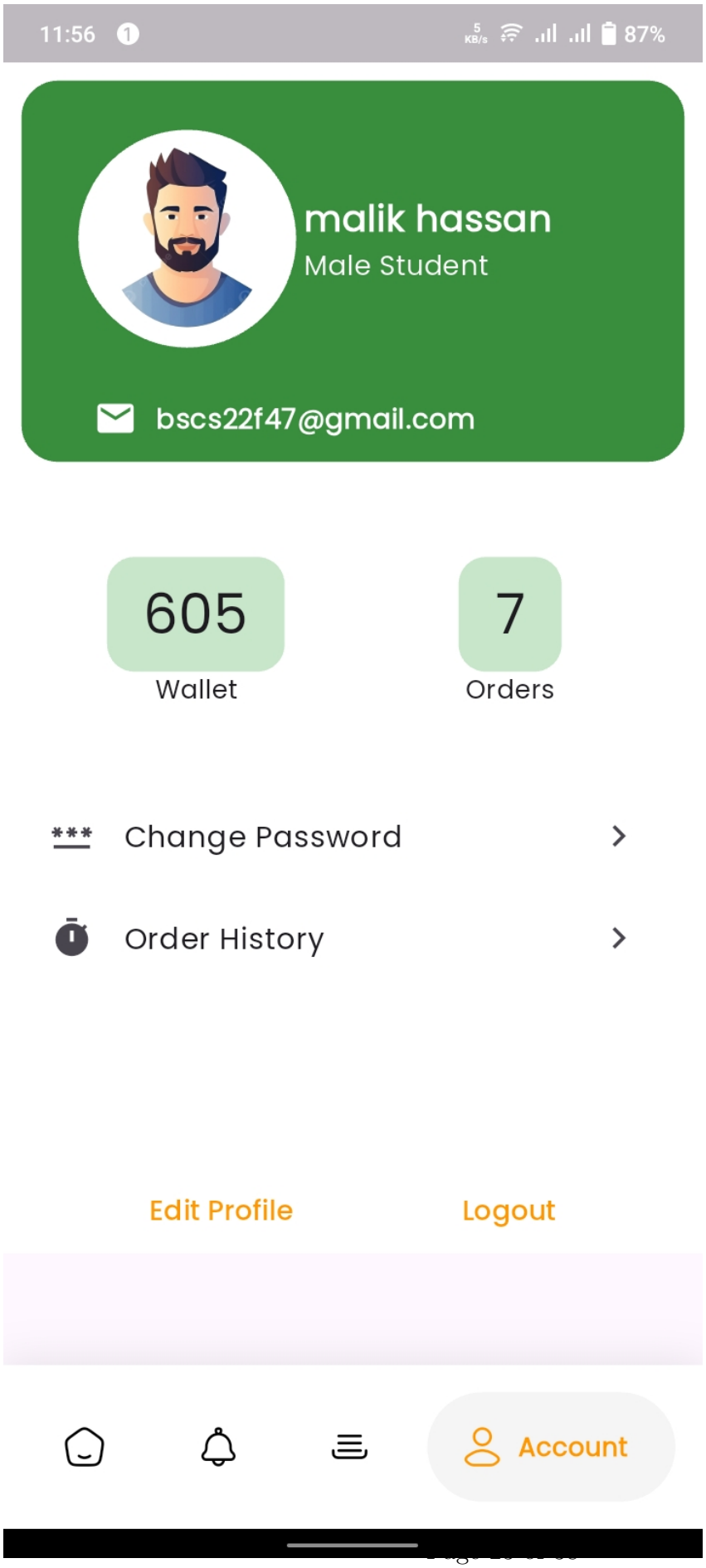


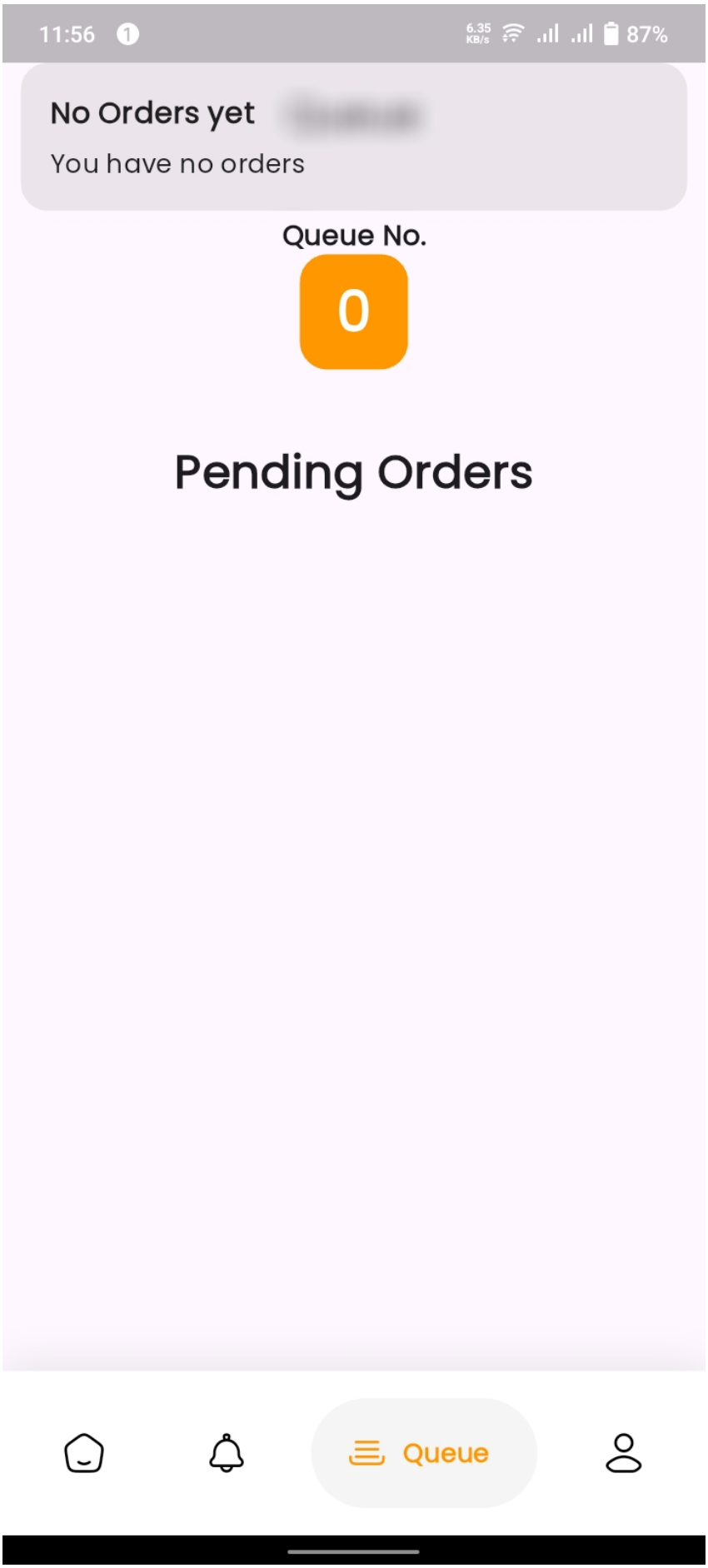

Namal University Mianwali
Home
Orders
Dashboard
Customers
Menus
Dashboard
Orders 140
Products 120
Users 30
Settings 11
Sales Data
Product Data
Product A
Product B
Product C
Product D
Export and Delete All Records

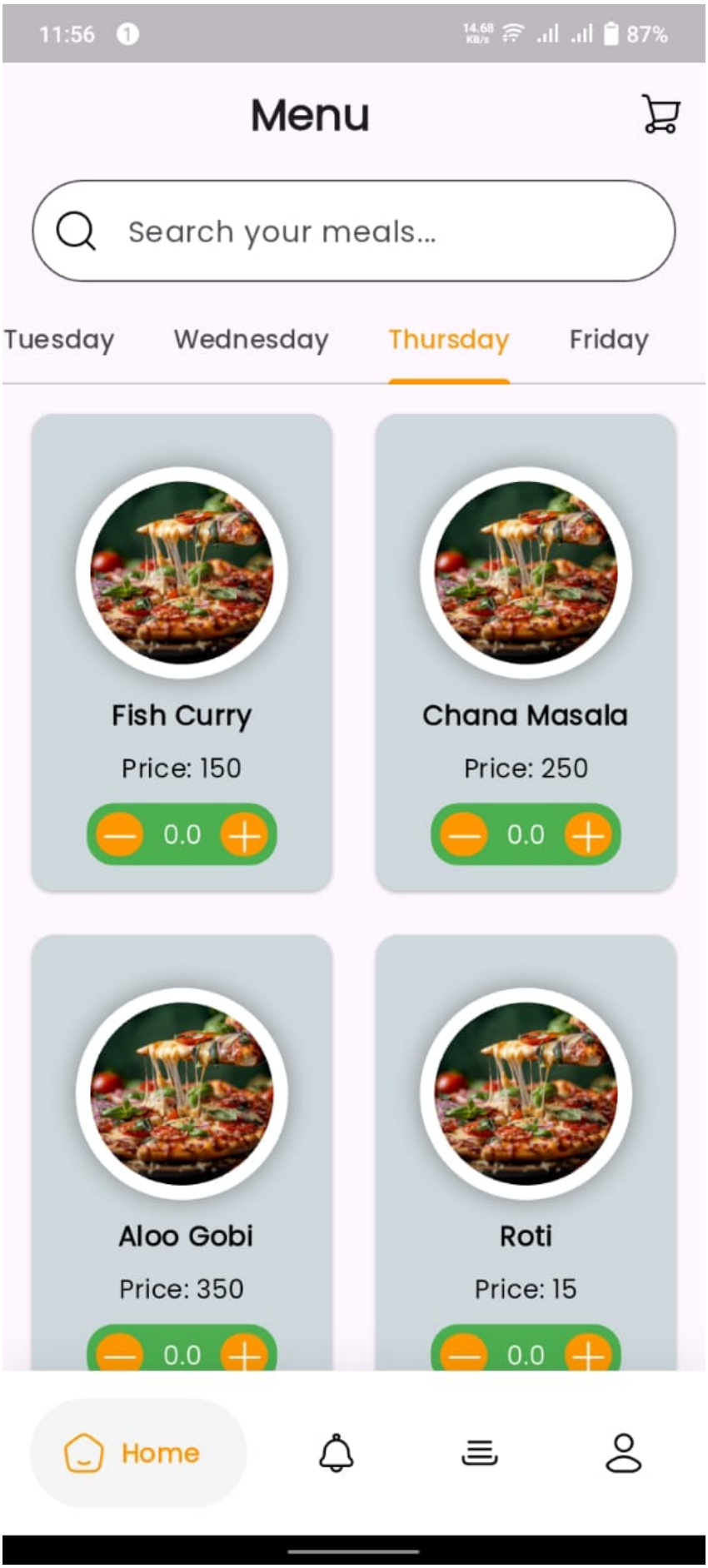

Namal University Mianwali
Home
Orders
Dashboard
Customers
Menus
All Customers
Add Customer
First Name
Last Name
Fingerprint
Email Address
Password
Balance
Select Role
Add Customer
Search by Email:
Abubakr panwar
Email: bscs22f41@namal.edu.pk
Balance: 210 RS
Role: Male Student
Fingerprint:
Edit Delete
usman khan
Email: bscs22f08@namal.edu.pk
Balance: 5000 RS
Role: Student
Fingerprint:
Department: CS
Year: 3
Gender: Male
Edit Delete
malik hassan
Email: bscs22f47@gmail.com
Balance: 605 RS
Role: Male Student
Fingerprint:
Edit Delete
junaid ameer
Email: bscs22f46@namal.edu.pk
Balance: 6000 RS
Role: Student
Fingerprint:
Department: CS
Sadiq Ameen
Email: bscs22f40@namal.edu.pk
Balance: 5000 RS
Role: Faculty
Fingerprint: @34323423423
Malik Basit
Email: basit@gmail.com
Balance: 0 RS
Role: Female Student
Fingerprint:


Namal University Mianwali
Home
Orders
Dashboard
Customers
Menus
Manage Menus
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Monday
Paneer Tikka - RS: 130 - Qty: 220
Grilled marinated paneer cubes served with mint chutney
Edit Delete Dish
Roti - RS: 15 - Qty: 400
Soft and fluffy whole wheat flatbreads.
Edit Delete Dish
Edit Menu Delete Menu
Tuesday
Lamb Vindaloo - RS: 270 - Qty: 200
Spicy lamb stew made with vinegar and a blend of spices.
Edit Delete Dish
Dal Makhani - RS: 170 - Qty: 100
Creamy black lentils cooked slowly with butter and spices.
Edit Delete Dish
Roti - RS: 15 - Qty: 400
Soft and fluffy whole wheat flatbreads.
Edit Delete Dish
Wednesday
Chana Masala - RS: 250 - Qty: 240
Chickpeas cooked in a spicy tomato and onion gravy.
Edit Delete Dish
Saag Paneer - RS: 160 - Qty: 100
Paneer cubes cooked in a spiced spinach sauce
Edit Delete Dish
Roti - RS: 15 - Qty: 400
Soft and fluffy whole wheat flatbreads.
Edit Delete Dish

6.2.2 Mobile Screens:




11:56
87%
No Orders yet
You have no orders
Queue No.
0
Pending Orders
Queue


11:56
87%
Menu
Search your meals...
Tuesday
Wednesday
Thursday
Friday
Fish Curry
Price: 150
0.0
Chana Masala
Price: 250
0.0
Aloo Gobi
Price: 350
0.0
Roti
Price: 15
0.0
Home

6.3 Screen Objects and Actions

6.3.1 Mobile Application Interface

Screen	Objects	Actions
Login Screen	• Email/ID input field • Password input field • Login button • Forgot password link	• Enter credentials • Submit login form • Request password reset
Home Screen	• Featured items carousel • Category icons • Notification badge • Search bar • Bottom navigation bar	• Browse featured items • Select category • View notifications • Search for items • Navigate to other screens
Menu Screen	• Category tabs • Menu item cards • Item images • Price labels • Add to cart buttons	• Switch categories • View item details • Add items to cart • Adjust quantities • Apply customizations
Cart Screen	• Item list • Quantity adjusters • Remove buttons • Subtotal display • Checkout button	• Adjust quantities • Remove items • Apply special instructions • Proceed to checkout • Continue shopping

Order Tracking	• Status indicator • Progress bar • Estimated time display • Order details • Contact button	• View order status • Check estimated time • View order details • Contact kitchen staff • Cancel order (if allowed)
Wallet Screen	• Balance display • Transaction history	• View balance • Browse transactions

6.3.2 Web Portal Interface

Screen	**Objects**	**Actions**
Dashboard	• Statistics cards • Recent orders list • Alert notifications • Quick action buttons • Charts and graphs	• View system metrics • Access recent orders • Respond to alerts • Perform quick actions • Analyze trends
Menu Management	• Category list • Item table • Add/Edit forms • Image uploader • Availability toggles	• Create/edit categories • Add/edit menu items • Upload item images • Set prices • Toggle availability

Order Management	• Order queue • Status filters • Order detail view • Status update buttons • Search and filter tools	• View incoming orders • Update order status • View order details • Search for specific orders • Filter by status/date
Reports	• Report type selector • Date range picker • Data visualization • Export buttons • Filter options	• Select report type • Set date range • View visualized data • Export reports • Apply filters

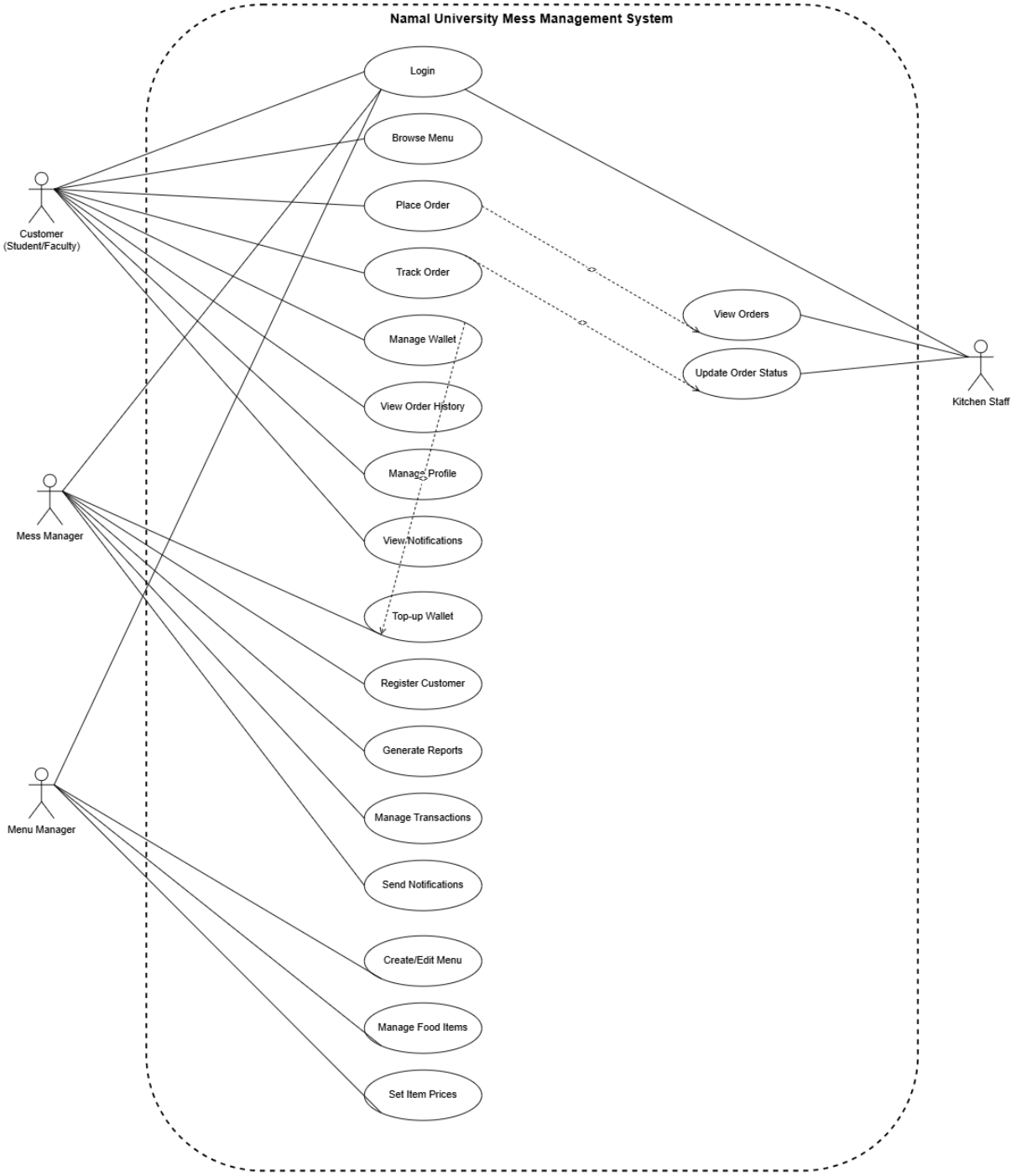
7 Requirements Traceability Matrix

Requirement ID	**Description**	**Component/Module**
REQ-3.1.1	User registration and login	Authentication
REQ-3.1.2	Role-based access control	Authentication
REQ-3.1.3	Password reset functionality	Authentication
REQ-3.2.1	Menu creation and management	Order Management
REQ-3.2.2	Menu categorization	Order Management
REQ-3.2.3	Menu item availability control	Order Management
REQ-3.3.1	Order placement	Order Management
REQ-3.3.2	Order tracking	Order Management
REQ-3.3.3	Order history	Order Management
REQ-3.4.1	Digital wallet	Payment System
REQ-3.4.3	Payment processing	Payment System
REQ-3.5.1	Sales reporting	Order Management
REQ-3.5.2	Consumption analytics	Order Management
REQ-3.6.1	Order notifications	Notification System
REQ-3.6.2	Promotional notifications	Notification System
REQ-3.7.1	Profile management	Authentication
REQ-5.1.1	Response time	All Modules
REQ-5.1.2	Throughput	All Modules

REQ-5.1.3	Capacity	All Modules
REQ-5.3.1	Authentication security	Authentication
REQ-5.3.2	Data security	All Modules
REQ-5.4.1	Usability	All Modules
REQ-5.4.3	Scalability	All Modules

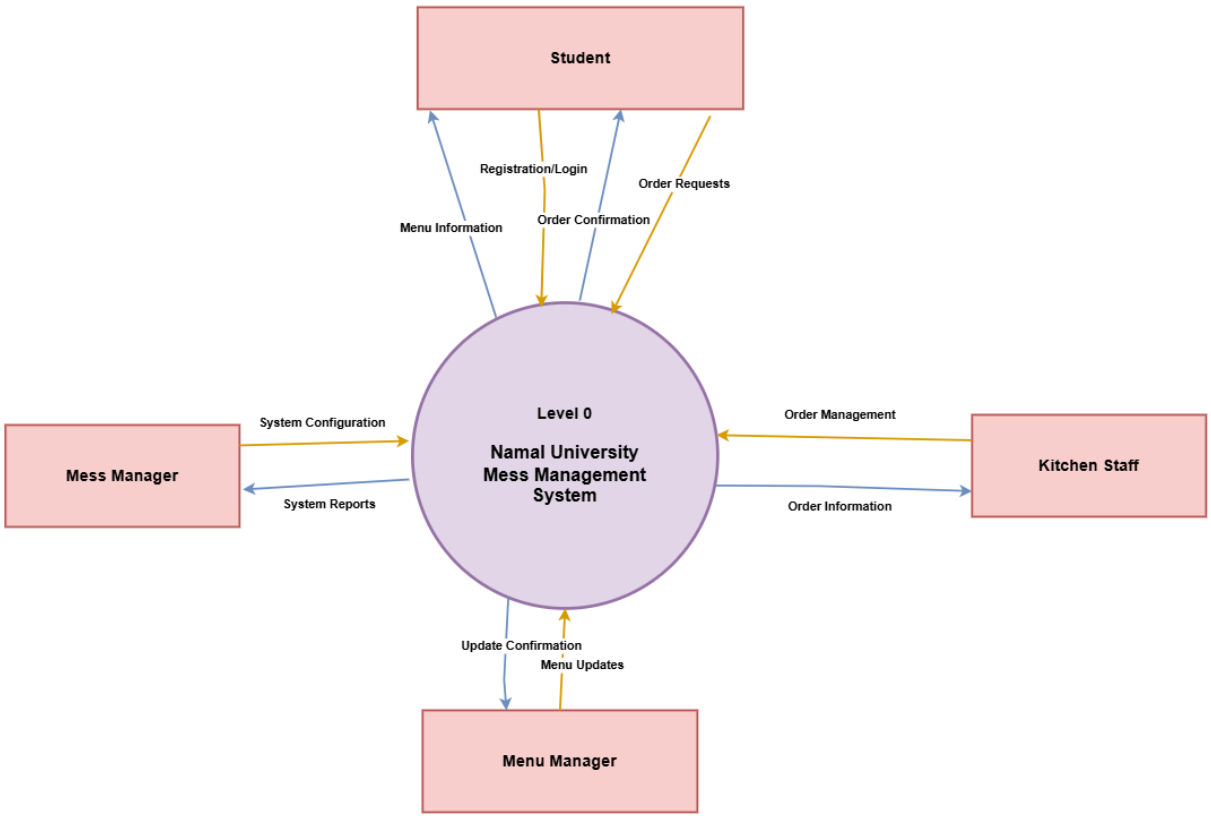
8 Appendices

8.1 Appendix A: Use Case Diagram

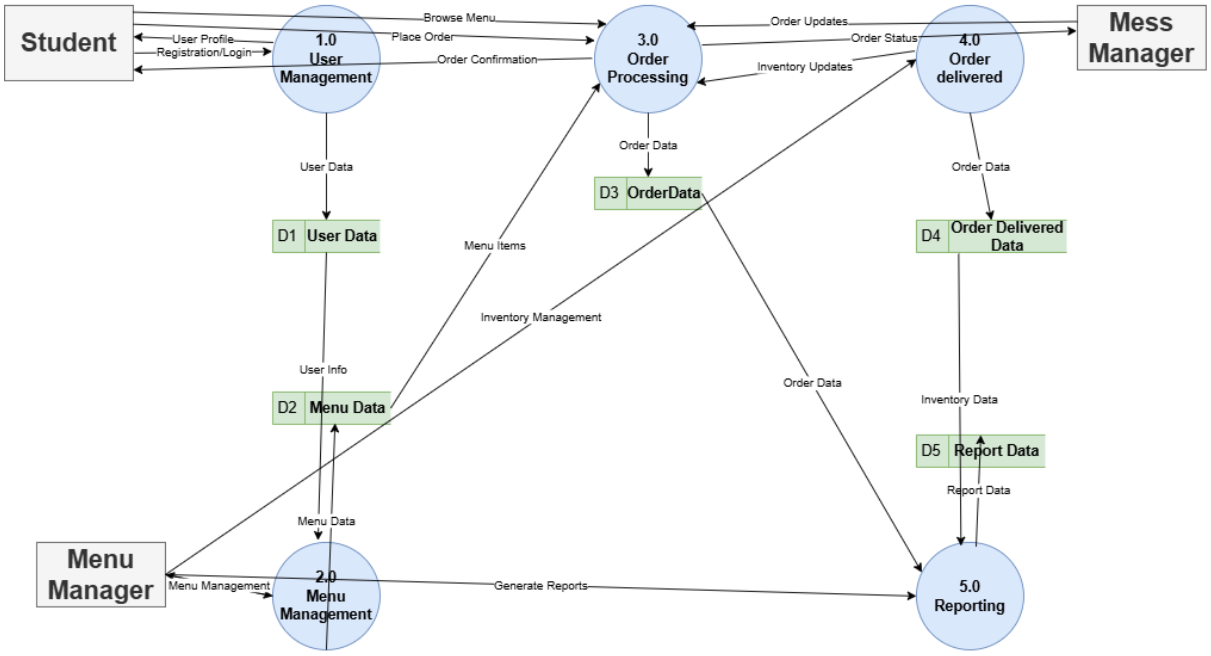


8.2 Appendix B: Data Dlow Diagrams

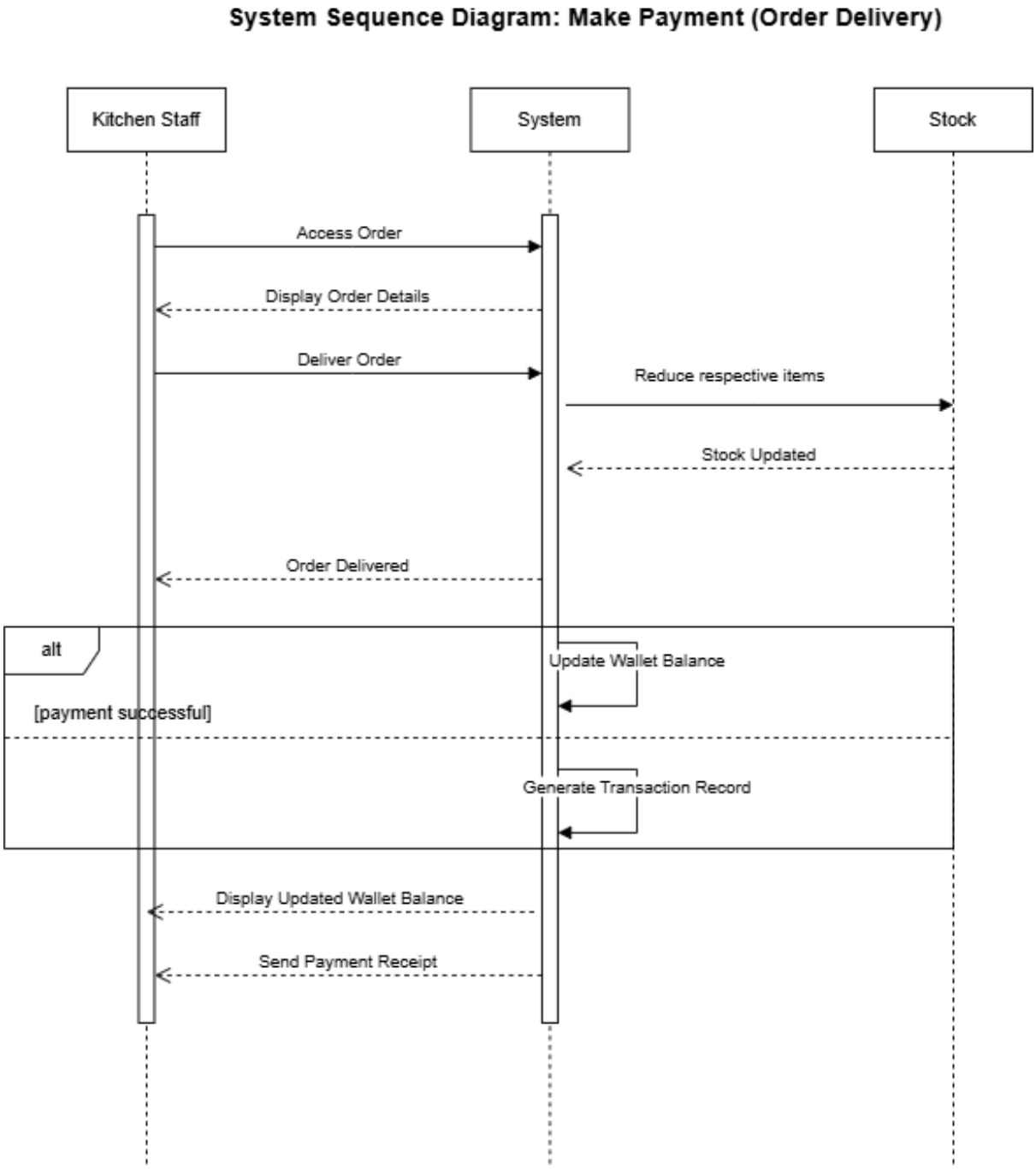
Namal University Mess Management System - Context Level DFD



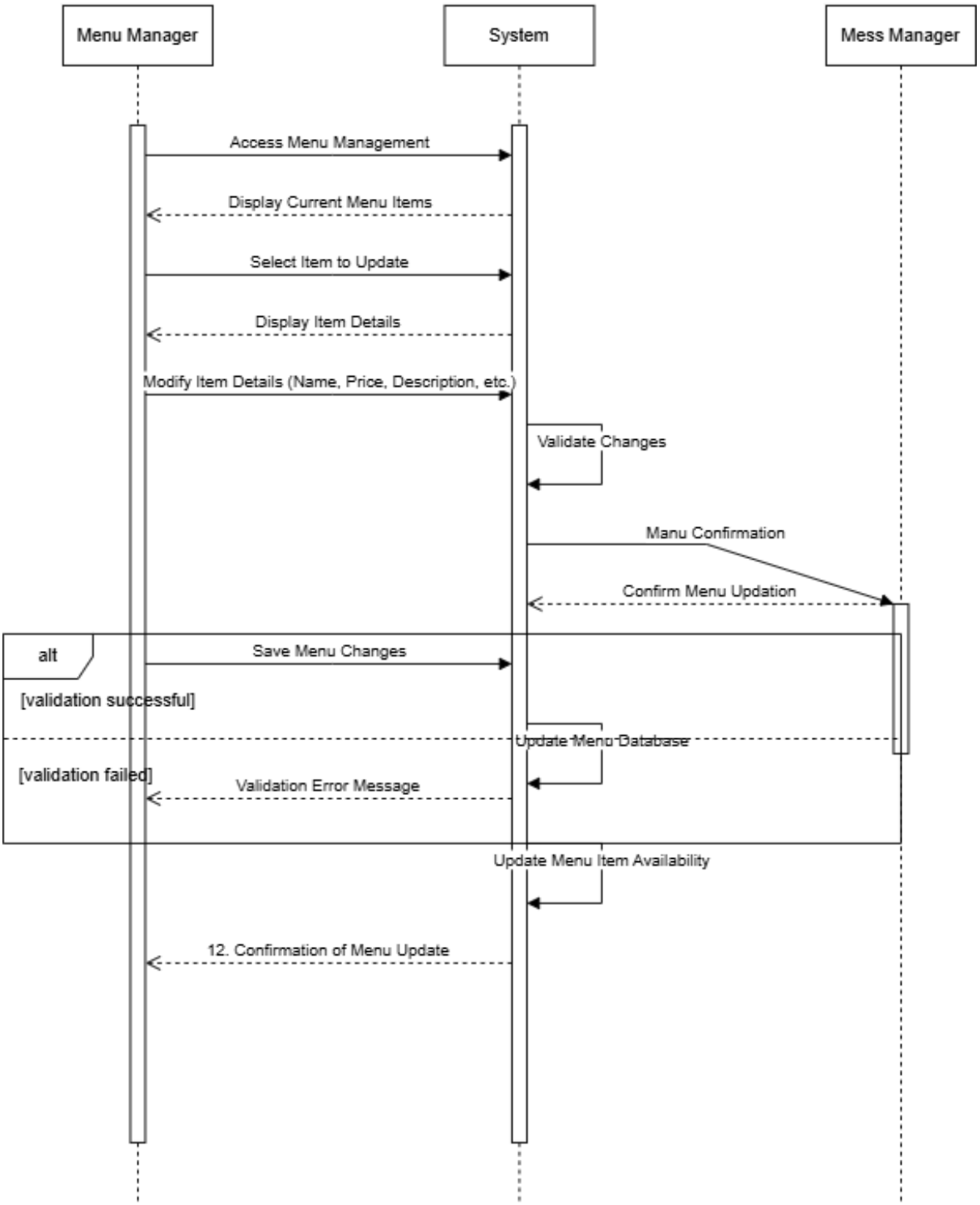
Namal University Mess Management System - Level-1 DFD



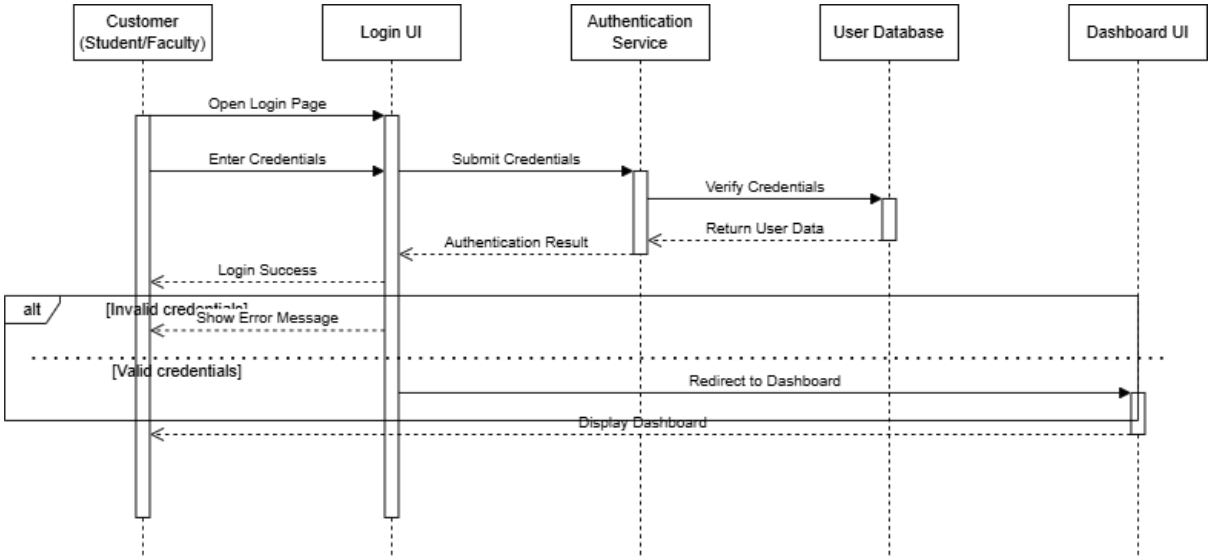
8.3 Appendix C: System Sequence Diagram

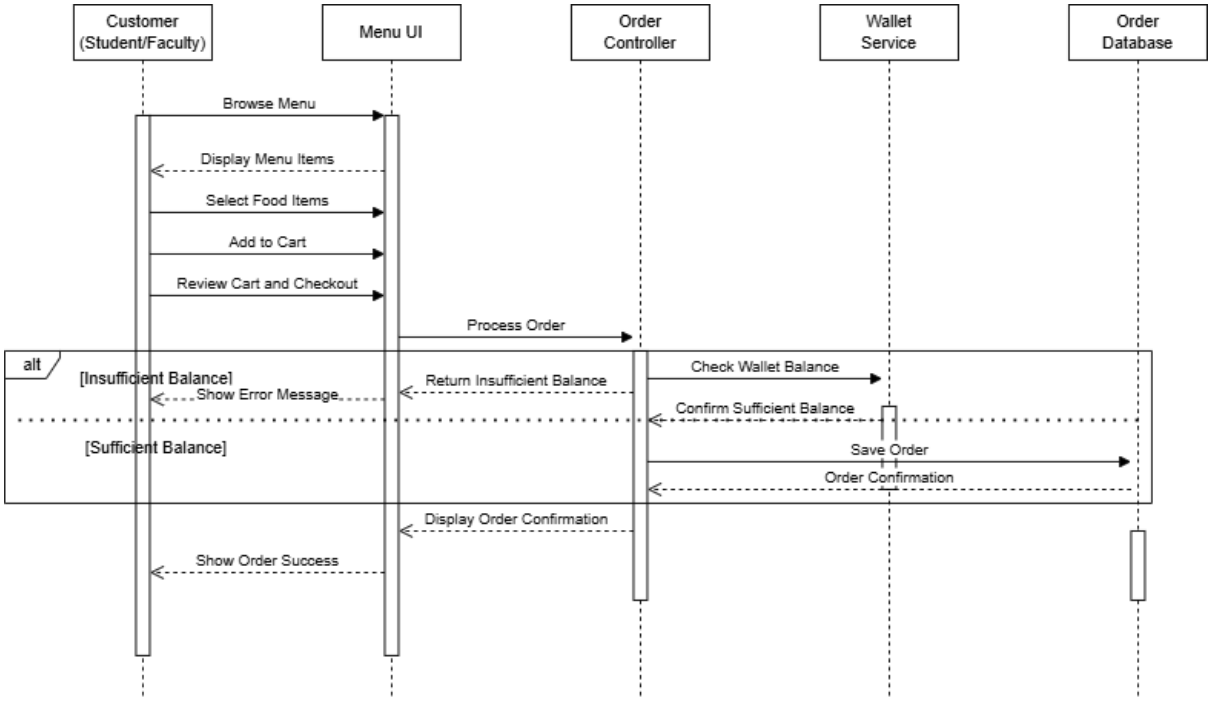


System Sequence Diagram: Update Menu

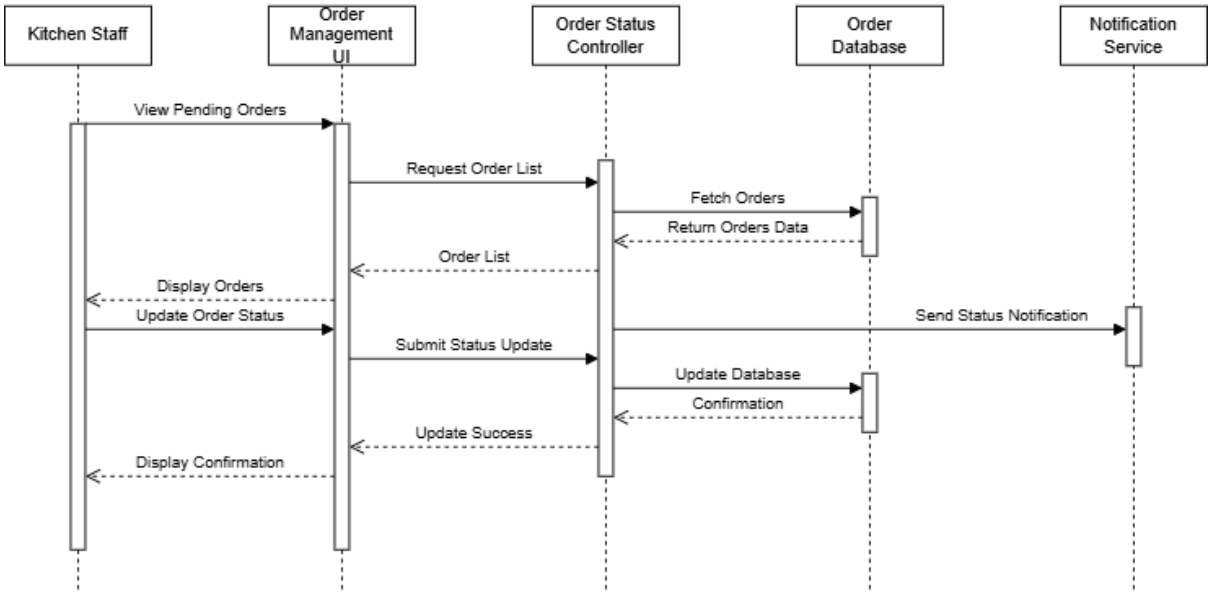


8.4 Appendix D: Sequence Diagrams

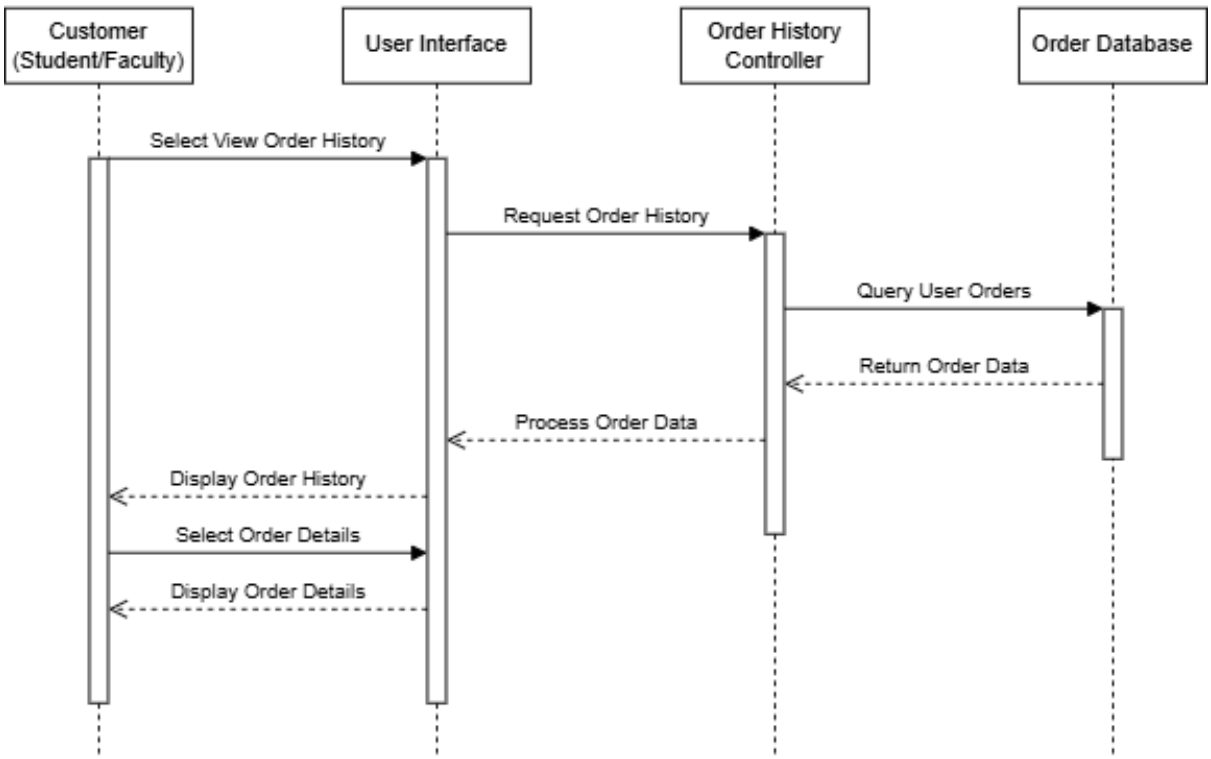


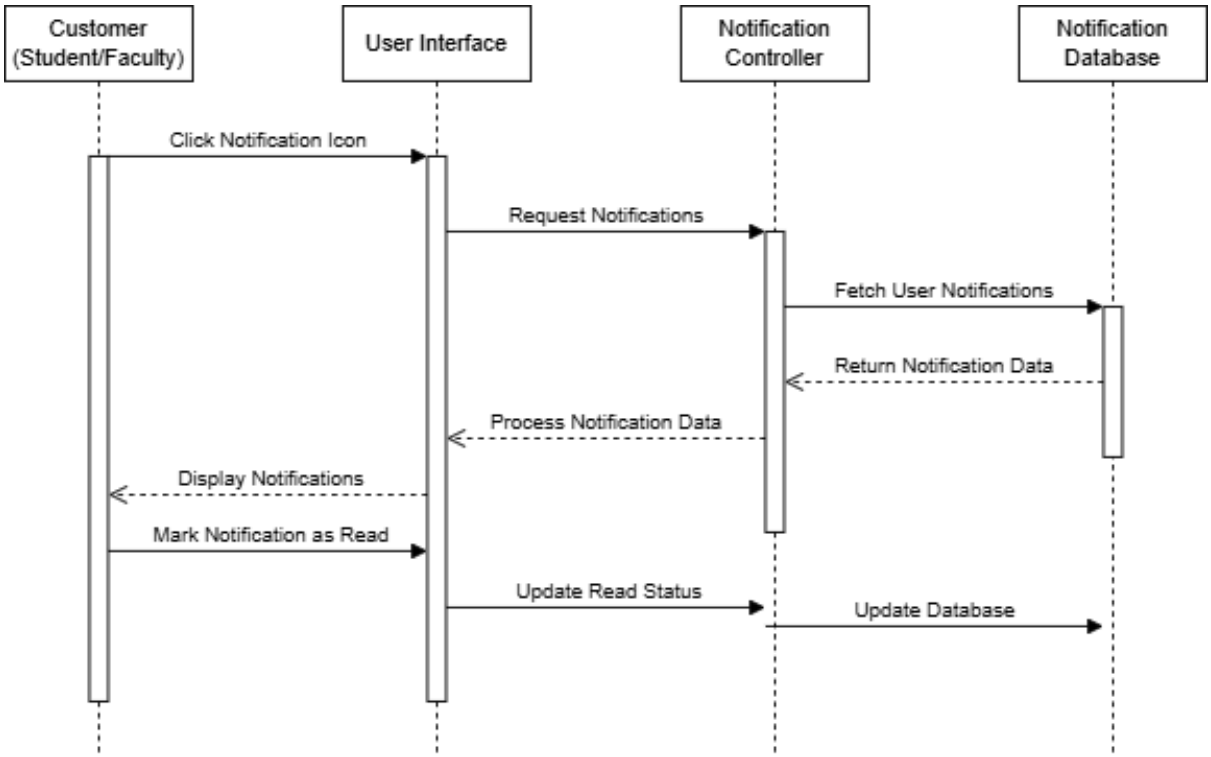


Update Order Menu - Sequential Diagram

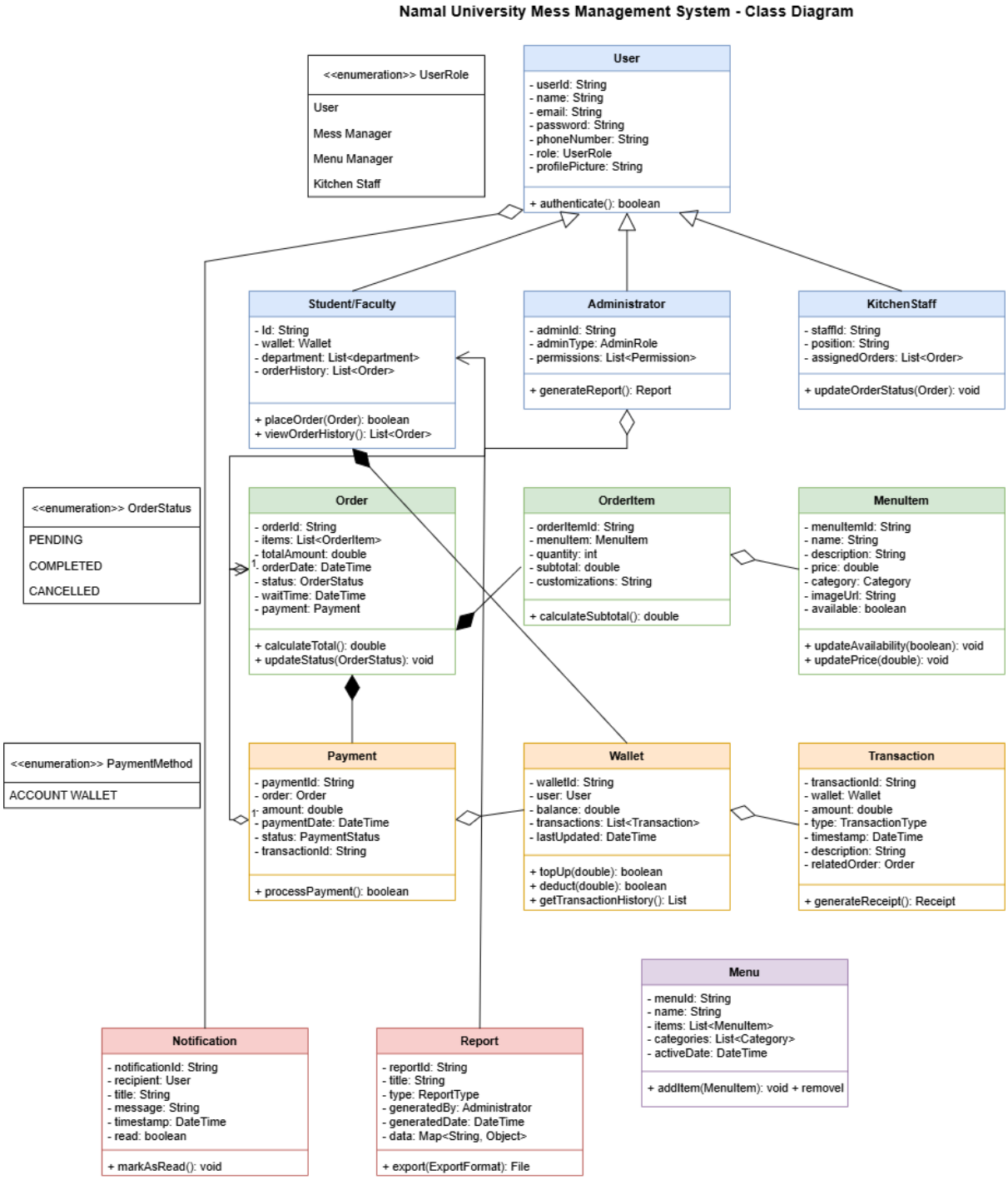


View History - Sequential Diagram




Customer (Student/Faculty)
User Interface
Notification Controller
Notification Database
Click Notification Icon
Request Notifications
Fetch User Notifications
Return Notification Data
Process Notification Data
Display Notifications
Mark Notification as Read
Update Read Status
Update Database

8.5 Appendix E: Class Diagram



8.6 Appendix F: Glossary

- **API:** Application Programming Interface - A set of rules that allows different software applications to communicate with each other.
- **CRUD:** Create, Read, Update, Delete - The four basic operations of persistent storage.
- **DFD:** Data Flow Diagram - A graphical representation of the flow of data through an information system.
- **MVC:** Model-View-Controller - A software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements.
- **REST:** Representational State Transfer - An architectural style for designing networked applications.
- **SDD:** Software Design Document - A document that describes how a software system will be designed and built.
- **SRS:** Software Requirements Specification - A document that describes the requirements of a software system.
- **UI:** User Interface - The space where interactions between humans and machines occur.
- **UX:** User Experience - The overall experience of a person using a product, especially in terms of how easy or pleasing it is to use.
- **JWT:** JSON Web Token - A compact, URL-safe means of representing claims to be transferred between two parties.
- **Firebase:** A platform developed by Google for creating mobile and web applications.
- **Flutter:** An open-source UI software development kit created by Google for building natively compiled applications.
- **React.js:** A JavaScript library for building user interfaces, maintained by Facebook.