Software Requirement Specification

for

Namal Mess Management System

Prepared by: Muhammad Raqib Hayat, Abu Bakar

Instructor: Muhammad Ali Shahid

Course: CS-260 - Software Engineering

Semester: 6th, Spring 2025

Institution: Namal University, Mianwali

Contents

Revision History

Name	**Date**	**Reason For Changes**	**Version**
M. Raqib Hayat & Abubakar	06/05/2025	Initial draft	1.0
M. Raqib Hayat & Abubakar	13/05/2025	Final draft	2.0

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document details the "Mess Management System" requirements - version 1.0. This system is designed to modernize the food service management at Namal University by replacing the traditional cash-based mess system with a digital solution. This document covers the entire system, including the mobile application for students and the web-based administrative dashboard.

1.2 Document Conventions

The following conventions have been used throughout this document:

- **Bold text** - Represents key features and important terms
- *Italic text* - Used for emphasis on critical points
- `Code format` - Used for technical terms and system components
- Priority levels are explicitly stated for each requirement:
  - High: Essential for system functionality
  - Medium: Important but not critical
  - Low: Desirable but can be implemented in later phases

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Development Team**: Should focus on functional requirements (Section 3) and interface specifications
- **Project Managers**: Should review the overall description (Section 2) and non-functional requirements
- **Testers**: Should concentrate on functional requirements (Section 3) for developing test cases
- **University Administrators**: Should understand the product perspective (Section 2.1) and user classes (Section 2.3)
- **Students (End Users)**: May find the product features (Section 2.2) most relevant

For a comprehensive understanding, it is recommended to read the document in sequential order. Those familiar with the project context may skip directly to sections relevant to their role.

1.4 Project Scope

The Mess Management System aims to transform the traditional mess management process at Namal University into a digital, cashless solution. The system will:

- Allow students to browse menus, place orders, and make digital payments through a mobile application
- Provide administrators with a web platform to manage menus, track orders, and generate reports
- Eliminate cash transactions by implementing a digital wallet system
- Streamline the meal ordering and delivery process
- Provide real-time reporting and analytics on consumption patterns
- Improve inventory management and reduce food waste

Key benefits include:

- Convenience for students through digital meal ordering
- Enhanced transparency in billing and payment
- Improved operational efficiency for mess administration
- Better planning and resource allocation based on data analytics
- Reduced wait times and improved service quality

The system will not handle physical food preparation, direct grocery purchasing, external catering services, or meal delivery beyond university premises.

1.5 References

1. Namal University Mess Management Guidelines, 2025
2. Software Engineering Project Guidelines, Department of Computer Science, Namal University, 2025
3. IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)

2 Overall Description

2.1 Product Perspective

The Cashless Mess Management System is a new, self-contained product designed to replace the existing manual mess management system at Namal University. It consists of two main components:

1. **Mobile Application**: For students to browse menus, place orders, and make payments
2. **Administrative Web Portal**: For mess administrators to manage operations

The system will interact with the university's existing student information system to validate student identities and maintain accurate records. It will also integrate with secure payment gateways for processing transactions.

2.2 Product Features

The Mess Management System includes the following major features:

1. **User Authentication and Account Management**
   - Student registration and login
   - Administrator account management
   - Password recovery and profile management
2. **Digital Menu Management**
   - Daily/weekly menu display
   - Category-based food items
   - Pricing and availability information
   - Nutritional information display
3. **Order Processing**
   - Meal selection and customization
   - Cart management
   - Order confirmation and tracking
   - Order history and reordering
4. **Payment System**
   - Digital wallet
   - Balance top-up options
   - Transaction history
   - Payment receipt generation
5. **Inventory Management**
   - Stock tracking
   - Low inventory alerts
   - Consumption patterns analysis
   - Waste reduction metrics
6. **Reporting and Analytics**

- Sales reports
- Consumption trends
- Financial summaries
- Feedback analysis

7. **Notification System**
   - Order status updates
   - Low balance alerts
   - Special menu announcements
   - Important system updates

2.3 Product Benefits

The implementation of the Cashless Mess Management System will provide the following benefits:

- **For Students**:
  - Convenient meal ordering from anywhere on campus
  - Reduced waiting time at mess counters
  - Transparent billing and payment history
  - Ability to customize meal preferences
  - Real-time notifications about order status
- **For Mess Administration**:
  - Streamlined order management process
  - Better demand forecasting and inventory control
  - Reduced paperwork and manual record-keeping
  - Improved financial tracking and reporting
  - Data-driven decision making
- **For University**:
  - Enhanced student satisfaction with mess services
  - Reduced food waste and operational costs
  - Better resource allocation
  - Modern digital campus initiative
  - Improved transparency in mess operations

2.4 User Classes and Characteristics

1. **Students**
   - **Characteristics**: Tech-savvy, frequent users, use mobile devices regularly
   - **Priority**: High (primary users)
   - **Usage Frequency**: Daily
   - **Technical Expertise**: Basic to moderate
   - **Functions Used**: Menu browsing, ordering, payment, profile management
2. **Mess Administrators**
   - **Characteristics**: Staff responsible for day-to-day mess operations
   - **Priority**: High
   - **Usage Frequency**: Daily
   - **Technical Expertise**: Moderate
   - **Functions Used**: Menu creation, order management, reporting, user management
3. **Kitchen Staff**
   - **Characteristics**: Food preparation personnel
   - **Priority**: Medium
   - **Usage Frequency**: Daily
   - **Technical Expertise**: Basic
   - **Functions Used**: Order queue viewing, meal preparation tracking
4. **University Financial Office**
   - **Characteristics**: Administrative staff handling finances
   - **Priority**: Medium
   - **Usage Frequency**: Weekly
   - **Technical Expertise**: Moderate to high
   - **Functions Used**: Financial reports, payment reconciliation
5. **System Administrators**
   - **Characteristics**: IT staff maintaining the system
   - **Priority**: Low (small number but critical)
   - **Usage Frequency**: Occasional
   - **Technical Expertise**: High
   - **Functions Used**: System configuration, backup, user role management

2.5 Operating Environment

The Cashless Mess Management System will operate in the following environment:

Mobile Application:

- **Operating Systems**: Android 8.0+ and iOS 12.0+
- **Platforms**: Smartphones and tablets
- **Network**: Wi-Fi and mobile data (3G/4G/5G)
- **Screen Resolutions**: Multiple device sizes (responsive design)

Web Application:

- **Web Browsers**: Chrome (v80+), Firefox (v75+), Safari (v13+), Edge (v80+)
- **Screen Resolutions**: Desktop and tablet displays
- **Operating Systems**: Windows 10+, macOS 10.14+, Linux (major distributions)

Server Environment:

- **Hosting**: Google Cloud Platform
- **Database**: Firebase Firestore
- **Backend Runtime**: Node.js v14+
- **Authentication**: Firebase Authentication
- **Storage**: Cloud Storage for Firebase

2.6 Design and Implementation Constraints

The development of the Cashless Mess Management System is subject to the following constraints:

1. **Technology Constraints**:
   - Must use Flutter for mobile application development
   - Must use React.js for web application development
   - Must utilize Firebase services for backend functionality
   - Must follow responsive design principles for cross-device compatibility
2. **Security Constraints**:
   - Must comply with university data protection policies
   - Must encrypt all sensitive user data
   - Must implement secure authentication mechanisms
   - Must maintain audit logs of all financial transactions
3. **Regulatory Constraints**:

- Must comply with local financial regulations for digital payments
- Must adhere to data privacy laws
- Must secure appropriate approvals for handling student financial data

4. **Hardware Constraints**:
    - System must function on the university network infrastructure
    - Mobile app must work efficiently on budget to mid-range devices
    - Must operate reliably with limited internet connectivity

5. **Development Constraints**:
    - Must follow Agile Scrum methodology with 2-week sprints
    - Must use Git for version control with GitHub as a repository
    - Must implement continuous integration and deployment
    - Code must meet established coding standards and pass automated tests

2.7 User Documentation

The following documentation will be provided with the system:

1. **For Students**:
    - Mobile application user guide
    - FAQ section within the app
    - Quick start guide (digital format)
    - Video tutorials for common tasks

2. **For Administrators**:
    - Comprehensive system administration manual
    - Web portal user guide
    - Technical reference documentation
    - System troubleshooting guide

3. **For Kitchen Staff**:
    - Order processing guide
    - Quick reference cards for daily operations

4. **For Financial Office**:
    - Financial reporting guide
    - Reconciliation procedures document

All documentation will be available in digital format (PDF) with the option for printed copies upon request. Video tutorials will be accessible through the university's learning management system.

2.8 Assumptions and Dependencies

Assumptions:

- Students have access to smartphones with internet connectivity
- The University will provide reliable internet access in dining areas
- Students are comfortable with digital payment systems
- Mess administration is willing to adapt to the new digital workflow
- University IT infrastructure can support the system requirements
- The University will approve the necessary changes to the existing mess policies

Dependencies:

- Access to student enrollment data from university systems
- Availability of secure payment gateway services
- Firebase infrastructure availability and reliability
- University approval for system implementation
- Sufficient budget allocation for system development and maintenance
- Cooperation from mess staff for system training and adoption
- Availability of technical infrastructure for system deployment

3 Functional Requirements

3.1 User Authentication and Registration

3.1.1 Description and Priority

This feature allows users to create accounts, log in, and manage their credentials. It is the system's entry point and ensures that only authorized users can access it.

Priority: High

3.1.2 Response Sequences

The following sequences illustrate the stimulus-response flow for key authentication scenarios:

- **Student Registration**:
  1. Student accesses the registration page
  2. Student enters university ID, email, and personal details
  3. System validates university ID against the university database
  4. System sends verification email

5. Student verifies email
6. System creates and activates an account

- **User Login**:
  1. User enters credentials (email/ID and password)
  2. System validates credentials
  3. System grants access to the appropriate interface based on the user role
  4. System maintains session until logout or timeout

- **Password Recovery**:
  1. User selects ”Forgot Password” option
  2. User enters registered email address
  3. System sends password reset link
  4. User creates new password
  5. System updates the password in the database

3.1.3 Functional Requirements

REQ-1.1: The system shall allow students to register using their university ID, name, email, and phone number.

REQ-1.2: The system shall verify student identity by cross-checking against the university database.

REQ-1.3: The system shall send a verification email to confirm the user’s email address.

REQ-1.4: The system shall support at least three user roles: student, mess administrator, and system administrator.

REQ-1.5: The system shall allow users to log in using an email/ID and password combination.

REQ-1.6: The system shall implement password reset functionality via email.

REQ-1.7: The system shall enforce secure password policies (minimum 8 characters, mix of letters, numbers, and special characters).

REQ-1.8: The system shall automatically log out inactive users after 30 minutes.

REQ-1.9: The system shall maintain user session information securely.

REQ-1.10: The system shall provide error messages for invalid login attempts without revealing specific reasons for security purposes.

3.2 Menu Management

3.2.1 Description and Priority

This feature allows mess administrators to create, update, and publish daily/weekly menus. Students can view these menus to make food choices.

Priority: High

3.2.2 Stimulus/Response Sequences

- **Creating Menu**:
  1. Admin selects "Create Menu" option
  2. Admin selects a date for the menu
  3. Admin adds food items with prices and descriptions
  4. Admin publishes the menu
  5. The System makes the menu available to students
- **Viewing Menu**:
  1. Student opens the app
  2. Student navigates to the "Menu" section
  3. System displays the current day's menu by default
  4. Student can switch to future dates (if available)
- **Updating Menu**:
  1. Admin selects an existing menu
  2. Admin makes necessary changes
  3. Admin saves the updated menu
  4. System updates menu for all users
  5. System notifies students of significant changes (optional)

3.2.3 Functional Requirements

REQ-2.1 The system shall allow administrators to create daily and weekly menus.

REQ-2.2 The system shall support categorization of food items (breakfast, lunch, dinner, snacks).

REQ-2.3 The system shall allow adding images for food items.

REQ-2.4 The system shall allow setting prices for individual food items.

REQ-2.5 The system shall support adding nutritional information for each food item.

REQ-2.6 The system shall allow administrators to mark items as vegetarian, vegan, halal, or containing allergens.

REQ-2.7 The system shall allow administrators to set availability status for each item.

REQ-2.8 The system shall support bulk upload of menu items via a CSV file.

REQ-2.9 The system shall allow administrators to clone previous menus to create new ones.

REQ-2.10 The system shall display the current day's menu to students by default, with an option to view future menus.

3.3 Order Processing

3.3.1 Description and Priority

This feature enables students to browse menus, select items, place orders, and track their status. It also allows kitchen staff to receive and process orders.
Priority: High

3.3.2 Stimulus/Response Sequences

- **Placing Order**:
  1. Student browses menu
  2. Student adds items to the cart
  3. Student reviews cart
  4. Student confirms order
  5. System verifies sufficient wallet balance
  6. System processes payment
  7. System sends confirmation
  8. System forwards the order to the kitchen queue
- **Tracking Order**:
  1. Student navigates to the "My Orders" section
  2. System displays current and past orders
  3. Student selects a specific order
  4. System shows detailed status and progress
- **Processing Order (Kitchen)**:
  1. Kitchen staff views the incoming order queue
  2. Staff marks order as "in preparation"
  3. Staff marks order as "ready for pickup"
  4. System notifies the student
  5. Student collects the order
  6. Staff marks order as "completed"

3.3.3 Functional Requirements

REQ-3.1 The system shall allow students to browse the menu and add items to the cart.

REQ-3.2 The system shall enable students to modify quantities or remove items from the cart.

REQ-3.3 The system shall calculate the total order amount, including any applicable taxes.

REQ-3.4 The system shall verify sufficient balance before processing a payment.

REQ-3.5 The system shall generate a unique order ID for each order.

REQ-3.6 The system shall maintain order history for at least 30 days.

REQ-3.7 The system shall allow students to reorder from previous orders.

REQ-3.8 The system shall display the estimated preparation time for orders.

REQ-3.9 The system shall allow kitchen staff to update order status (received, in preparation, ready, completed).

REQ-3.10 The system shall queue orders for kitchen staff based on order time.

REQ-3.11 The system shall notify students when their order status changes.

REQ-3.12 The system shall support order cancellation before preparation begins.

3.4 Payment System

3.4.1 Description and Priority

This feature manages all financial transactions within the system, including wallet top-ups, order payments, and financial reporting.
Priority: High

3.4.2 Stimulus/Response Sequences

- **Wallet Top-up**:
  1. Student selects ”Add Money” option
  2. Student enters amount
  3. Student selects payment method
  4. System processes payment
  5. System updates wallet balance
  6. System sends confirmation
- **Order Payment**:
  1. Student confirms order

2. System verifies wallet balance
3. System deducts the amount from the wallet
4. System issues receipt
5. System updates transaction history

- **Viewing Transaction History**:
  1. Student navigates to the "Transactions" section
  2. System displays a list of transactions
  3. Student can filter by date or type
  4. Student can view the receipt for a specific transaction

3.4.3 Functional Requirements

REQ-4.1 The system shall maintain a digital wallet for each student.

REQ-4.2 The system shall support multiple payment methods for wallet top-up (credit/debit card, bank transfer, mobile payment).

REQ-4.3 The system shall process wallet top-ups securely through a payment gateway.

REQ-4.4 The system shall verify sufficient balance before processing an order.

REQ-4.5 The system shall generate digital receipts for all transactions.

REQ-4.6 The system shall maintain a complete transaction history for audit purposes.

REQ-4.7 The system shall allow students to view their transaction history.

REQ-4.8 The system shall enable administrators to process refunds when necessary.

REQ-4.9 The system shall generate daily financial reports for mess administration.

REQ-4.10 The system shall notify students when the wallet balance falls below a configurable threshold.

3.5 Reporting and Analytics

3.5.1 Description and Priority

This feature provides insights into system usage, food consumption patterns, and financial performance through various reports and dashboards.

Priority: Medium

3.5.2 Stimulus/Response Sequences

- **Generating Reports**:
  1. Admin navigates to the "Reports" section
  2. Admin selects report type
  3. Admin specifies parameters (date range, type)
  4. System generates a report
  5. System displays a report with options to export
- **Viewing Analytics Dashboard**:
  1. Admin selects "Dashboard" option
  2. System displays key metrics and trends
  3. Admin can interact with charts and graphs
  4. Admin can filter data by various parameters

3.5.3 Functional Requirements

REQ-5.1 The system shall generate daily sales reports.

REQ-5.2 The system shall provide weekly and monthly financial summaries.

REQ-5.3 The system shall analyze and display food consumption patterns.

REQ-5.4 The system shall track popular menu items and generate popularity rankings.

REQ-5.5 The system shall generate inventory usage reports.

REQ-5.6 The system shall provide user activity statistics (active users, order frequency).

REQ-5.7 The system shall support exporting reports in PDF and Excel formats.

REQ-5.8 The system shall provide visual dashboards with key performance indicators.

REQ-5.9 The system shall allow filtering reports by date range, meal type, and user group.

REQ-5.10 The system shall generate alerts for unusual patterns or potential issues.

3.6 Notifications

3.6.1 Description and Priority

This feature handles all system notifications to keep users informed about important events, updates, and status changes.

Priority: Medium

3.6.2 Stimulus/Response Sequences

- **Order Status Notification**:
  1. Kitchen staff updates order status
  2. System generates a notification
  3. System sends a notification to the student's device
  4. Student receives and views the notification
- **Low Balance Alert**:
  1. System detects wallet balance below threshold
  2. System generates an alert notification
  3. System sends a notification to the student
  4. Student receives a reminder to top up the wallet
- **Special Announcements**:
  1. Admin creates announcement
  2. Admin selects target user groups
  3. System sends notification to selected users
  4. Users receive announcement notification

3.6.3 Functional Requirements

REQ-6.1: The system shall send notifications for order status changes.

REQ-6.2: The system shall alert students when their wallet balance falls below a threshold.

REQ-6.3: The system shall notify students about special menu items or promotions.

REQ-6.4: The system shall allow administrators to send announcements to all users or specific groups.

REQ-6.5: The system shall support in-app notifications and push notifications to mobile devices.

REQ-6.6: The system shall maintain a notification history for users to review.

REQ-6.7: The system shall allow users to set notification preferences.

REQ-6.8: The system shall send reminders for pending orders that haven't been picked up.

REQ-6.9: The system shall notify administrators about system issues or anomalies.

REQ-6.10: The system shall notify kitchen staff about incoming orders.

3.7 User Profile Management

3.7.1 Description and Priority

This feature allows users to manage their personal information, preferences, and account settings.

Priority: Medium

3.7.2 Stimulus/Response Sequences

- **Updating Profile**:
  1. User navigates to the profile section
  2. User selects ”Edit Profile”
  3. User updates information
  4. User saves changes
  5. System verifies and updates the profile
- **Changing Password**:
  1. User selects ”Change Password” option
  2. User enters current and new password
  3. System verifies current password
  4. System updates to a new password
  5. System confirms change via email
- **Setting Preferences**:
  1. User navigates to the preferences section
  2. User adjusts notification settings
  3. User sets dietary preferences
  4. User saves settings
  5. System updates preferences

3.7.3 Functional Requirements

REQ-7.1: The system shall allow users to update personal information (name, email, phone number).

REQ-7.2: The system shall enable users to change their password.

REQ-7.3: The system shall allow users to set dietary preferences (vegetarian, allergies, etc.).

REQ-7.4: The system shall permit users to configure notification preferences.

REQ-7.5: The system shall enable users to view their order history.

REQ-7.6: The system shall allow users to manage payment methods for wallet top-up.

REQ-7.7: The system shall support profile picture uploads.

REQ-7.8: The system shall allow users to delete their account and data.

REQ-7.9: The system shall provide users with a detailed view of their wallet activity.

REQ-7.10: The system shall allow users to save favorite menu items for quick reordering.

3.8 Inventory Management

3.8.1 Description and Priority

This feature helps mess administrators track food inventory, monitor usage, and manage stock levels.
Priority: Medium

3.8.2 Stimulus/Response Sequences

- **Adding Inventory**:
  1. Admin selects "Inventory" section
  2. Admin chooses the "Add Item" option
  3. Admin enters item details and quantity
  4. System updates the inventory database
- **Checking Stock Levels**:
  1. Admin navigates to the inventory dashboard
  2. System displays current stock levels
  3. System highlights low stock items
  4. Admin can filter by category
- **Updating Stock After Usage**:
  1. Admin selects "Update Stock" option
  2. Admin enters items used and quantities
  3. System calculates and updates inventory
  4. System logs the update with a timestamp

3.8.3 Functional Requirements

REQ-8.1: The system shall maintain an inventory database of all food items and ingredients.

REQ-8.2: The system shall track stock levels and usage patterns.

REQ-8.3: The system shall alert administrators when stock levels fall below defined thresholds.

REQ-8.4: The system shall allow adding new inventory items with details (name, category, unit, etc.).

REQ-8.5: The system shall support barcode scanning for quick inventory updates.

REQ-8.6: The system shall automatically reduce inventory based on orders processed.

REQ-8.7: The system shall generate inventory reports by category and period.

REQ-8.8: The system shall calculate projected inventory needs based on historical usage.

REQ-8.9: The system shall support batch updates of inventory data.

REQ-8.10: The system shall maintain inventory transaction history for auditing purposes.

4 External Interface Requirements

4.1 User Interfaces

4.1.1 Mobile Application Interface

The mobile application shall have a clean, intuitive interface following Material Design guidelines. The app shall support both light and dark themes and be responsive to work on various screen sizes. The application shall include the following main screens:

- Login/Registration
- Home Dashboard
- Menu Browser
- Cart
- Order Tracking
- Wallet & Transactions
- Profile & Settings
- Notifications

Main Navigation Structure:

- Bottom navigation bar with icons for: Home, Menu, Orders, Wallet, Profile
- Pull-down notifications accessible from any screen
- Search functionality accessible from relevant screens

Key UI Components:

- Food item cards with images, names, prices, and "Add to Cart" button

- Order tracking with visual status indicators
- Transaction history with clear date, amount, and purpose
- Menu filters for food categories and dietary preferences
- Interactive wallet balance display with top-up button
- Visual order receipt with QR code for verification

4.1.2 Web Application Interface

The administrative web portal shall follow responsive web design principles and be compatible with modern web browsers. The portal shall include the following main sections:

- Dashboard
- User Management
- Menu Management
- Order Management
- Inventory Control
- Reports & Analytics
- System Settings

Main Navigation Structure:

- Sidebar navigation with expandable categories
- Top header with notifications, user profile, and quick actions
- Breadcrumb navigation for deep-linked pages

Key UI Components:

- Interactive dashboard with key metrics and charts
- Data tables with sorting, filtering, and export options
- Form interfaces for data entry and modification
- Calendar views for menu planning
- Real-time order queue display
- Interactive reports with visual graphs

4.2 Hardware Interfaces

4.2.1 Mobile Device Interfaces

The mobile application shall interface with:

- Camera for QR code scanning and profile picture capture
- GPS for location verification (optional feature)
- Notification system for push notifications
- Storage for caching menu data and images
- Network interfaces (Wi-Fi, cellular) for data transmission

4.2.2 Server Hardware Interfaces

The backend system shall interface with:

- Database servers for data storage and retrieval
- File storage systems for image storage
- Network equipment for communication with clients
- Backup systems for data redundancy

4.2.3 Kitchen Display Interface

The system shall support interfaces with kitchen display systems, including:

- Touch screen monitors for order management
- Thermal printers for order tickets
- Sound systems for order alerts

4.3 Software Interfaces

4.3.1 Database Interfaces

- The system shall interface with the Firebase Firestore database
- Database operations shall be performed through the Firebase SDK
- Real-time synchronization shall be implemented for order status updates
- Offline caching shall be supported for essential data

4.3.2 Authentication Interfaces

- The system shall use Firebase Authentication for user management
- The system shall interface with the university's student information system to verify student IDs
- OAuth 2.0 shall be supported for third-party login (Google, university SSO)

4.3.3 Payment Gateway Interfaces

- The system shall interface with secure payment gateways
- Payment processing shall follow industry standard protocols
- Transaction data shall be exchanged using secure APIs

4.3.4 Cloud Services Interfaces

- The system shall utilize Firebase Cloud Functions for serverless computing
- Google Cloud Storage shall be used for file storage
- Firebase Cloud Messaging shall be used for push notifications

4.4 Communications Interfaces

4.4.1 Network Protocols

- The system shall use HTTPS for all client-server communications
- WebSockets shall be used for real-time updates (order status, notifications)
- RESTful APIs shall be implemented for data exchange
- JSON shall be used as the primary data exchange format

4.4.2 API Specifications

- All APIs shall be versioned (starting with v1)
- Authentication shall be implemented using JSON Web Tokens (JWT)
- Rate limiting shall be applied to prevent abuse
- Error responses shall follow a standardized format

4.4.3 External System Interfaces

- The system shall provide APIs for potential integration with other university systems
- APIs shall be documented using the OpenAPI (Swagger) specification
- Data exchange with external systems shall be secured using API keys

5 Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Response Time

- The mobile application shall load initially within 3 seconds on standard networks
- Menu browsing shall respond within 1 second of user interaction
- Order placement shall be completed within 3 seconds after confirmation
- Payment processing shall complete within 5 seconds
- Push notifications shall be delivered within 10 seconds of triggering events

5.1.2 Throughput

- The system shall support up to 500 concurrent users
- The system shall process up to 50 orders per minute during peak hours
- The system shall handle up to 10,000 menu views per day

5.1.3 Capacity

- The database shall store up to 3 years of transaction history
- The system shall support up to 5,000 registered users
- The image storage shall accommodate up to 10,000 food images

5.1.4 Reliability

- The system shall have an uptime of at least 99.5%
- The system shall limit scheduled maintenance to non-peak hours
- The mean time between failures shall exceed 30 days

5.2 Safety Requirements

5.2.1 Food Safety

- The system shall allow labeling of allergens in food items
- The system shall display clear warnings for items containing common allergens
- The system shall maintain food preparation timestamps to ensure freshness
- The system shall support flagging of items for dietary restrictions (halal, kosher, etc.)

5.2.2 Data Safety

- The system shall perform regular automated backups of all data
- The system shall implement disaster recovery procedures
- The system shall maintain data integrity through transaction validation
- The system shall prevent data corruption through appropriate database constraints

5.2.3 User Safety

- The system shall implement account lockout after multiple failed login attempts
- The system shall store personally identifiable information (PII) securely
- The system shall prevent unauthorized access to user data
- The system shall require secure passwords for all accounts

5.3 Security Requirements

5.3.1 Authentication Security

- The system shall use industry-standard encryption for password storage
- The system shall implement multi-factor authentication for administrative accounts
- The system shall enforce password expiration policies
- The system shall maintain an authentication audit trail

5.3.2 Data Security

- All data transmission shall be encrypted using TLS 1.3 or higher
- Sensitive user data shall be encrypted at rest
- Financial transaction data shall be encrypted with AES-256 or equivalent
- Access to data shall be restricted based on user roles and permissions

5.3.3 Application Security

- The system shall be protected against common web vulnerabilities (XSS, CSRF, SQL injection)
- The system shall implement input validation for all user-supplied data
- The system shall restrict API access through authentication and authorization
- The system shall undergo regular security audits and penetration testing

5.4 Software Quality Attributes

5.4.1 Usability

- The system shall be intuitive for first-time users to navigate without training
- The system shall provide clear error messages and recovery options
- The system shall be accessible to users with disabilities (WCAG 2.1 AA compliance)
- The system shall support multiple languages (initially English and Urdu)

5.4.2 Maintainability

- The system shall be implemented using a modular architecture
- The system shall include comprehensive documentation for future development
- The system shall follow consistent coding standards
- The system shall provide detailed error logging for troubleshooting

5.4.3 Scalability

- The system shall be able to scale horizontally to accommodate user growth
- The system shall support database sharding for performance optimization
- The system shall implement caching strategies for frequent operations
- The system shall be able to scale resources during peak usage periods

5.4.4 Portability

- The mobile application shall be compatible with the Android and iOS platforms
- The web application shall be compatible with all major browsers
- The system shall use containerization for deployment flexibility
- The system shall be adaptable to different cloud platforms if necessary

6 Other Requirements

6.1 Business Rules

6.1.1 Operational Rules

- Orders must be placed at least 30 minutes before the desired pickup time
- Menu items must be updated at least 24 hours before becoming available
- Refunds must be processed within 3 business days
- Users with negative wallet balances cannot place new orders

6.1.2 Financial Rules

- A minimum wallet top-up amount of Rs. 500 is required
- All transactions must be logged with a timestamp and a unique reference
- Monthly financial reconciliation must be performed
- Meal prices must include all applicable taxes

6.1.3 Administrative Rules

- Administrator accounts must be approved by the system owner
- System reports must be archived for at least 3 years
- System backups must be performed daily
- User accounts inactive for more than 6 months shall be flagged

6.2 Legal Requirements

6.2.1 Data Protection

- The system shall comply with local data protection regulations
- Users shall have the right to request that their data be deleted
- The Privacy policy shall be presented during registration
- Data sharing with third parties shall be limited and disclosed

6.2.2 Financial Compliance

- The system shall comply with electronic payment regulations
- The system shall maintain financial records as required by law
- The system shall issue receipts for all financial transactions
- The system shall support financial auditing requirements

6.3 Internationalization Requirements

- The system shall support multiple languages (initially English and Urdu)
- The system shall display dates and times in the user's preferred format
- The system shall support multiple currencies for potential future expansion
- The system shall accommodate cultural differences in food categorization

6.4 Documentation Requirements

- User manuals shall be provided for all user roles
- Technical documentation shall be maintained throughout development
- API documentation shall be automatically generated and kept updated
- Regular system status reports shall be generated

7 Appendices

7.1 Appendix A: Glossary

- **API**: Application Programming Interface - a set of rules that allows different software applications to communicate with each other.
- **Authentication**: The process of verifying a user's or system's identity.
- **Authorization**: The process of determining whether a user has permission to access specific resources or functions.
- **Backend**: The server-side of an application, responsible for data processing, business logic, and database operations.
- **Dashboard**: A user interface that organizes and presents information in a way that is easy to read and understand.