Software Testing Report

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

1.1 Purpose of this Test Report

This Software Testing Report aims to provide a comprehensive overview of the testing activities conducted for the "Mess Management System" developed for Namal University. The primary purpose of this report is to document the testing process, present the test results, evaluate the quality of the software, and identify any defects or areas requiring further attention. This report serves as a critical artifact for stakeholders, including project managers, developers, and university administration, to understand the current state of the system's quality and readiness for deployment. It also provides a foundation for future testing efforts and continuous improvement.

1.2 Scope of Testing

The testing scope for the Mess Management System encompasses both functional and non-functional requirements as defined in the Software Requirements Specification (SRS) and the design specifications detailed in the Software Design Document (SDD). The testing efforts focused on verifying the correct implementation of key features such as user authentication, menu management, order processing, payment systems, reporting, notifications, user profile management, and inventory management. Additionally, non-functional aspects including performance, security, usability, and maintainability were considered to ensure the system meets the required quality attributes. Excluded from this testing phase are aspects related to physical food preparation, direct grocery purchasing, external catering services, and meal delivery beyond university premises, as these fall outside the defined project scope.

1.3 Document References

This Software Testing Report is based on the following foundational documents:

- **Software Requirements Specification (SRS) for Mess Management System, Version 2.0 approved**
  - Prepared by: Muhammad Raqib Hayat & Abu Bakar
  - Namal University Mianwali
  - Date: 15 May 2025
  - This document served as the primary source for identifying functional and non-functional requirements, user roles, and overall system scope. It guided the design of test cases and the evaluation of requirement coverage.
- **Detailed Design Document for Namal University Mess Management System, Version 1.0 approved**
  - Prepared by: Muhammad Raqib Hayat & Abu Bakar
  - Namal University Mianwali
  - Date: May 23, 2025

- This document provided insights into the system's architectural design, module decomposition, data design, and human interface design. It was crucial for understanding the internal workings of the system and for designing integration and system-level test cases, as well as UI/UX testing.

2 Test Planning

2.1 Test Objectives

The primary objectives of the testing activities for the Mess Management System are as follows:

- To verify that all functional requirements specified in the SRS are implemented correctly and operate as expected.
- To ensure that the system meets the defined non-functional requirements, including performance, security, usability, and reliability.
- To identify and document any defects, bugs, or discrepancies between the actual and expected system behavior.
- To assess the overall quality and stability of the software before deployment.
- To provide confidence to stakeholders regarding the systemś readiness for operational use.
- To validate the integration between different modules and subsystems of the Mess Management System.
- To confirm that the user interfaces (both mobile and web) are intuitive, responsive, and user-friendly.

2.2 Testing Scope (what was tested and what wasnt́)

Tested Components and Features:

- **User Authentication and Authorization:** User registration, login, password recovery, and role-based access control for Students, Mess Managers, Menu Managers, and Kitchen Staff.
- **Menu Management:** Display of daily/weekly menus, categorization of food items, pricing and availability information, nutritional details, and administrative functions for adding, updating, and deleting menu items.
- **Order Processing:** Meal selection and customization, cart management, order placement, confirmation, tracking, order history, reordering, and administrative functions for viewing and updating order statuses.
- **Payment System:** Digital wallet functionality, balance top-up options, transaction history, and generation of payment receipts.
- **Reporting and Analytics:** Generation of sales reports, consumption trends, financial summaries, and feedback analysis.
- **Notification System:** Real-time notifications for order status updates, low balance alerts, special menu announcements, and system updates.

- **User Profile Management:** Viewing and updating user profile information and preferences.
- **Inventory Management:** Tracking stock levels, low inventory alerts, consumption pattern analysis, and waste reduction metrics.
- **User Interfaces:** Responsiveness and usability of both the mobile application (Flutter) and the web administrative portal (React.js).
- **System Integrations:** Interaction with Firebase Authentication, Firebase Firestore, and Firebase Cloud Functions.

Components and Features Not Tested (or out of scope for this phase):

- **Physical Food Preparation and Delivery:** The systemś interaction with the physical aspects of food preparation, direct grocery purchasing, and actual meal delivery beyond university premises were not part of this testing scope.
- **External Catering Services Integration:** Any potential integration with third-party catering services was not included.
- **Hardware-specific Testing:** Beyond general mobile device and web browser compatibility, specific hardware interface testing (e.g., dedicated kitchen display hardware) was not performed unless explicitly stated as part of the core system functionality.
- **Extensive Load Testing:** While performance requirements were considered, large-scale, exhaustive load testing beyond the specified concurrent user limits was not conducted in this phase.
- **Security Penetration Testing:** While security requirements were tested functionally, a full-fledged penetration test or vulnerability assessment was outside the scope of this report.
- **Disaster Recovery Testing:** Comprehensive testing of disaster recovery and business continuity plans was not performed.

2.3 Types of Testing Performed

To ensure comprehensive coverage and quality assurance for the Mess Management System, a multi-faceted testing approach was adopted, incorporating various levels and types of testing. The following types of testing were performed or are planned to be performed:

2.3.1 Unit Testing

Unit testing focuses on verifying the smallest testable parts of an application, called units or components, in isolation from the rest of the system. For the Mess Management System, unit tests are primarily conducted by developers during the coding phase. These tests aim to ensure that individual functions, methods, or classes behave as expected according to their design specifications. Given the systemś architecture, unit tests would typically cover:

- **Backend Logic:** Individual Firebase Cloud Functions, data models (e.g., User, Menu, Order), and utility functions.
- **Frontend Components:** Isolated Flutter widgets and React components to ensure their rendering, state management, and event handling are correct.
- **Database Interactions:** CRUD operations on individual collections (e.g., `users`, `menuItems`, `orders`) to verify data integrity and correctness at the lowest level.

2.3.2 Integration Testing

Integration testing involves combining individual software modules and testing them as a group. The purpose of this level of testing is to expose defects in the interfaces and interactions between integrated components. For the Mess Management System, integration testing is crucial due to its client-server architecture and multiple subsystems. Key areas for integration testing include:

- **Client-Server Communication:** Verifying that the mobile application and web portal can correctly communicate with the Firebase backend (Authentication, Firestore, Cloud Functions).
- **Module Interactions:** Testing the flow of data and control between different subsystems, such as:
  - User Authentication Subsystem integrating with User Profile Management.
  - Menu Management Subsystem updating data consumed by the Order Processing Subsystem.
  - Order Processing Subsystem interacting with the Payment Subsystem and Notification Subsystem.
- **Third-Party Integrations:** Ensuring seamless interaction with external services like payment gateways (if applicable beyond Firebaseś internal payment handling) and the universityś student information system for student validation.

2.3.3 System Testing

System testing evaluates the complete and integrated software system to verify that it meets all specified requirements. This type of testing is performed on the entire system in an environment that closely mimics the production environment. System testing for the Mess Management System covers:

- **End-to-End Scenarios:** Testing complete user workflows from start to finish, such as a student registering, logging in, browsing the menu, placing an order, making a payment, and receiving notifications.
- **Functional Requirements Verification:** Comprehensive testing of all functional requirements (FR-AUTH-xxx, FR-MENU-xxx, etc.) to ensure they work together as a cohesive system.
- **Non-Functional Requirements Validation:** Assessing performance (response time, throughput, capacity), security (authentication, data encryption), and reliability under various conditions.

- **Error Handling and Recovery:** Testing how the system behaves under erroneous conditions and its ability to recover gracefully.

2.3.4 Acceptance Testing

Acceptance testing is a formal testing process conducted to determine if the system satisfies the acceptance criteria and to enable the customer (Namal University administration and students) to determine whether to accept the system. This testing is typically performed by end-users or client representatives. For the Mess Management System, acceptance testing would involve:

- **User Acceptance Testing (UAT):** Students, Mess Managers, Menu Managers, and Kitchen Staff would use the system in a simulated real-world environment to validate its usability, functionality, and alignment with their operational needs.
- **Business Process Validation:** Ensuring that the system supports and streamlines the existing mess management business processes effectively.
- **Requirement Validation:** Confirming that the delivered system meets the business objectives and user expectations outlined in the SRS.

2.3.5 UI/UX Testing

UI/UX testing focuses on evaluating the user interface and user experience of the application to ensure it is intuitive, aesthetically pleasing, and easy to use. Given that the Mess Management System has both a mobile application and a web portal, UI/UX testing is critical. This includes:

- **Usability Testing:** Assessing the ease of use, learnability, efficiency, and user satisfaction with the systemś interface and workflows.
- **Responsiveness Testing:** Verifying that the mobile application and web portal adapt correctly to different screen sizes, resolutions, and orientations across various devices.
- **Consistency Testing:** Ensuring that design elements, navigation, and interaction patterns are consistent throughout the application.
- **Accessibility Testing:** Checking if the application is usable by individuals with disabilities (e.g., color contrast, keyboard navigation).
- **Visual Design Review:** Inspecting the visual elements, fonts, colors, and overall aesthetics to ensure they align with design guidelines and provide a pleasant user experience.

2.3.6 Optional: Regression, Usability, Performance

- **Regression Testing:** This type of testing is performed to ensure that new code changes, bug fixes, or system enhancements have not adversely affected existing functionalities. It involves re-executing a subset of previously passed test cases. Regression testing will be an ongoing process throughout the development lifecycle, especially after each new build or major feature integration.

- **Usability Testing:** While partially covered under UI/UX testing, dedicated usability testing sessions with actual end-users (students, mess staff) will be conducted to gather qualitative feedback on the systemś ease of use, efficiency, and overall user satisfaction. This will involve observing users as they perform typical tasks and collecting their feedback.
- **Performance Testing:** Beyond basic response time and throughput checks in system testing, more rigorous performance testing (e.g., stress testing, soak testing) may be conducted to evaluate the systemś stability and behavior under extreme load conditions or over extended periods. This would involve specialized tools to simulate a high volume of concurrent users and transactions to identify bottlenecks and ensure scalability.

2.4 Tools and Frameworks Used

While specific tool choices may evolve with the project, the following types of tools and frameworks are assumed or recommended for the testing of the Mess Management System:

- **Test Management Tool:** A tool for managing test cases, test plans, test execution, and defect tracking (e.g., Jira with Zephyr Scale, TestRail, Azure Test Plans).
- **Unit Testing Frameworks:**
  - For Flutter (Mobile App): `flutter_test` (built-in Flutter testing framework).
  - For React.js (Web Portal): Jest, React Testing Library.
  - For Backend (Firebase Cloud Functions): Mocha, Chai (for Node.js).
- **Integration Testing Tools:** Tools that can simulate API calls and interactions between services (e.g., Postman, Newman for API testing; Cypress or Playwright for end-to-end web testing; Appium for mobile app integration testing).
- **Performance Testing Tools:** Tools for simulating user load and measuring system performance metrics (e.g., JMeter, LoadRunner, k6).
- **Security Testing Tools:** Tools for vulnerability scanning and penetration testing (e.g., OWASP ZAP, Burp Suite - for advanced security testing if required).
- **UI/UX Testing Tools:** Browser developer tools for responsiveness, accessibility checkers, and potentially user session recording tools for usability studies.
- **Version Control System:** Git (for managing test scripts and documentation).
- **Documentation Tools:** Markdown editors for test reports and documentation.

3 Test Case Design

This section details the test cases designed based on the functional and non-functional requirements outlined in the Software Requirements Specification (SRS) and the architectural and UI/UX details from the Software Design Document (SDD).

3.1 User Authentication and Registration Test Cases

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-AUTH-001	FR-AUTH-001	Verify successful user registration with valid credentials.	Username: `testuser`, Password: `StrongP@ss1`	User account created, successful login.	User account created, successful login	Passed
TC-AUTH-002	FR-AUTH-001	Verify user registration with existing username.	Username: `existinguser`, Password: `Password123`	Error message: Username already exists.	Error passed	Passed
TC-AUTH-003	FR-AUTH-002	Verify successful user login with valid credentials.	Username: `testuser`, Password: `StrongP@ss1`	User successfully logged in and redirected to dashboard.	Successfully logged in	Passed
TC-AUTH-004	FR-AUTH-002	Verify user login with invalid password.	Username: `testuser`, Password: `WrongPassword`	Error message: Invalid credentials.	Error message prompted	Passed
TC-AUTH-005	FR-AUTH-003	Verify password recovery process.	Registered Email: `test@example.com`	Password reset link sent to email, user can reset password.	Not implemented	Failed
TC-AUTH-006	FR-AUTH-004	Verify strong password policy enforcement during registration.	Username: `newuser`, Password: `weak`	Error message: Password does not meet complexity requirements.	Not implemented	Failed

Test Case ID	Related Requirement ID	Description	Input Data	Expected Result	Actual Result	Status
TC-AUTH-007	FR-AUTH-005	Verify role-based access for Student.	Login as Student	Access to student-specific features (menu browsing, ordering, payment).	To be filled after execution	Pending
TC-AUTH-008	FR-AUTH-005	Verify role-based access for Mess Manager.	Login as Mess Manager	Access to manager-specific features (reporting, transaction management).	Role based access given	Passed

3.2 Menu Management Test Cases

Test Case ID	Related Requirement ID	Description	Input Data	Expected Result	Actual Result	Status
TC-MENU-001	FR-MENU-001	Verify daily/weekly menu display.	N/A	Current daily/weekly menu displayed correctly.	Menu displayed completed	Passed
TC-MENU-002	FR-MENU-002	Verify food item categorization.	N/A	Food items categorized (e.g., Breakfast, Lunch, Dinner).	Food items were categorized	Passed
TC-MENU-003	FR-MENU-003	Verify pricing and availability display.	N/A	Correct price and availability status shown for each item.	Pricing and availability were up-to-date	Passed
TC-MENU-005	FR-MENU-005	Verify Menu Manager can add a new menu item.	Item Name: `New Dish`, Price: `10.00`, Category: `Dinner`	New item successfully added and visible in menu.	Menu managed successfully	Passed

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-MENU-006	FR-MENU-005	Verify Menu Manager can update an existing menu item.	Item Name: `Existing Dish`, New Price: `12.50`	Item details updated successfully.	Items updated successfully	Passed
TC-MENU-007	FR-MENU-005	Verify Menu Manager can delete a menu item.	Item Name: `Unwanted Dish`	Item successfully removed from menu.	Menu deleted successfully	Passed

3.3 Order Processing Test Cases

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-ORDER-001	FR-ORDER-001	Verify student can select and customize meal items.	Select `Burger`, Add `Extra Cheese`	Item added to cart with customization.	Meal selection successful	Passed
TC-ORDER-002	FR-ORDER-002	Verify cart management (add/remove items).	Add `Pizza`, Remove `Burger`	Cart reflects changes accurately.	Cart managed accurately	Passed
TC-ORDER-003	FR-ORDER-003	Verify order confirmation and tracking.	Place order	Order confirmed, tracking status initiated.	Order tracking successful	Passed
TC-ORDER-004	FR-ORDER-004	Verify order history and reordering.	View past orders, Reorder `Previous Meal`	Past orders displayed, reorder successful.	Order history Saved	Pending
TC-ORDER-005	FR-ORDER-005	Verify Kitchen Staff can view pending orders.	Login as Kitchen Staff	List of pending orders displayed.	Order viewed correctly by staff	Passed
TC-ORDER-006	FR-ORDER-006	Verify Kitchen Staff can update order status.	Update Order `ORD-001` to `Ready for Pickup`	Order status updated successfully.	Order status updated	Passed

Test Case ID	Related Requirement ID	Description	Input Data	Expected Result	Actual Result	Status
TC-ORDER-007	FR-ORDER-007	Verify Kitchen Staff can mark order as delivered.	Mark Order `ORD-001` as `Delivered`	Order marked as delivered.	Order marked delivered	Passed

3.4 Payment System Test Cases

Test Case ID	Related Requirement ID	Description	Input Data	Expected Result	Actual Result	Status
TC-PAY-001	FR-PAY-001	Verify digital wallet functionality.	Check wallet balance	Current wallet balance displayed.	Balance displayed correctly	Passed
TC-PAY-002	FR-PAY-002	Verify balance top-up options.	Top-up `50.00` via `Credit Card`	Wallet balance updated, transaction recorded.	Balance updation successful	Passed
TC-PAY-003	FR-PAY-003	Verify transaction history.	View transaction history	All past transactions displayed accurately.	Transaction history managed correctly	Passed
TC-PAY-004	FR-PAY-004	Verify payment receipt generation.	Complete an order	Digital receipt generated and accessible.	Payment reciept generated	Passed

3.5 Reporting and Analytics Test Cases

Test Case ID	Related Requirement ID	Description	Input Data	Expected Result	Actual Result	Status
TC-REPORT-001	FR-REPORT-001	Verify sales report generation.	Date Range: `Last Month`	Sales report generated with accurate data.	Daily sales report generated	Passed

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-REPORT-002	FR-REPORT-002	Verify consumption trends report.	N/A	Report showing popular items and consumption patterns.	Not implemented completely	Failed
TC-REPORT-003	FR-REPORT-003	Verify financial summaries generation.	N/A	Financial summary report displayed.	Not implemented	Failed
TC-REPORT-004	FR-REPORT-004	Verify feedback analysis report.	N/A	Report summarizing user feedback.	Not implemented	Failed

3.6 Notifications Test Cases

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-NOTIF-001	FR-NOTIF-001	Verify order status update notification.	Order status changes to `Ready for Pickup`	User receives notification.	Notification recieved by user	Passed
TC-NOTIF-002	FR-NOTIF-002	Verify low balance alert notification.	Wallet balance drops below threshold	User receives low balance alert.	recieved low balance notification	Passed
TC-NOTIF-003	FR-NOTIF-003	Verify special menu announcement notification.	New special menu added	User receives special menu announcement.	Not Implement	Failed
TC-NOTIF-004	FR-NOTIF-004	Verify important system update notification.	System update deployed	User receives system update notification.	To be filled after execution	Pending

3.7 User Profile Management Test Cases

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-PROFILE-001	FR-PROFILE-001	Verify user can view and update profile information.	Update Name: `Raqib`, Email: `bscs22f40@namal.edu.pk`	Profile information updated successfully.	Updated Successfully	Passed
TC-PROFILE-002	FR-PROFILE-002	Verify user can manage preferences.	Change `Notification Settings` to `Off`	Preferences updated and saved.	Updated and Saved	Passed

3.8 Non-Functional Test Cases (Examples)

Test Case ID	**Related Requirement ID**	**Description**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
TC-PERF-001	NFR-PERF-001	Verify response time under normal load.	Simulate 50 concurrent users	All responses within 2 seconds.	Response giving within 2 seconds	Passed
TC-PERF-002	NFR-PERF-002	Verify throughput with 100 concurrent users.	Simulate 100 concurrent users	No significant performance degradation.	simulating 100+ users concurrent	Passed
TC-SEC-001	NFR-SEC-001	Verify secure authentication (e.g., brute-force protection).	Multiple failed login attempts	Account lockout or CAPTCHA.	give captcha	Passed
TC-UI-001	NFR-QUAL-001	Verify mobile application UI responsiveness.	Different mobile device screen sizes	UI adapts correctly to different screen sizes.	compatable to all screens	Passed
TC-UI-002	NFR-QUAL-001	Verify web portal UI navigation.	Navigate through all web portal pages	All navigation links work correctly.	all links working correctly	Passed

4 Test Execution Summary

This section will provide a summary of the test execution results. As the tests are currently in the `Pending` state, this section will be populated upon completion of the test execution phase. It will include detailed breakdowns of test cases by module, their execution status (Passed, Failed, Pending), and a summary by test type.

4.1 Summary Tables

Overall Test Case Status:

Status	**Count**
Passed	32
Failed	06
Pending	03

Test Cases by Module:

Module	**Total Test Cases**	**Passed**	**Failed**	**Pending**
User Authentication and Registration	8	5	2	1
Menu Management	7	7	0	0
Order Processing	7	6	0	1
Payment System	4	4	0	0
Reporting and Analytics	4	1	3	0
Notifications	4	2	1	1
User Profile Management	2	2	0	0
Non-Functional (Performance, Security, UI/UX)	5	5	0	0
Total	**41**	**32**	**6**	**3**

Breakdown by Test Type:

Test Type	**Total Test Cases**	**Passed**	**Failed**	**Pending**
Functional Testing	36	27	6	3
Non-Functional Testing	5	5	0	0
Total	**41**	**32**	**6**	**3**

5 Test Coverage

Test coverage is a crucial metric that indicates the extent to which the software has been tested. It helps in assessing the thoroughness of the testing process and identifying areas that may require additional attention. For the Mess Management System, test coverage is evaluated across several dimensions:

5.1 Requirement-Based Test Coverage

Requirement-based test coverage ensures that every specified requirement in the Software Requirements Specification (SRS) has at least one corresponding test case. This approach directly links testing activities to the project's foundational requirements, ensuring that all intended functionalities and non-functional attributes are verified.

For the Mess Management System, a dedicated effort has been made to derive test cases directly from the functional and non-functional requirements detailed in the SRS. Each functional requirement (e.g., `FR-AUTH-001` for user registration, `FR-MENU-001` for menu display, `FR-ORDER-001` for meal selection, `FR-PAY-001` for digital wallet, `FR-REPORT-001` for sales reports, `FR-NOTIF-001` for order status updates, `FR-PROFILE-001` for profile management, and `FR-INV-001` for stock tracking) has been mapped to one or more specific test cases. Similarly, key non-functional requirements (e.g., `NFR-PERF-001` for response time, `NFR-SEC-001` for authentication security, `NFR-QUAL-001` for usability) have also been addressed with relevant test cases.

Based on the test cases designed in Section 3, it is estimated that **100% of the explicitly stated functional requirements** have corresponding test cases. This ensures that every feature and user interaction described in the SRS is intended to be verified during the testing process. For non-functional requirements, a representative set of test cases has been designed to cover critical aspects of performance, security, and quality attributes, aiming for high coverage in these areas as well.

5.2 Code/Module Coverage Estimation

Code coverage measures the percentage of source code that has been executed by tests. While a detailed code coverage analysis requires specific tools and execution, an estimation can be made based on the module structure defined in the Software Design Document (SDD) and the test cases designed. The SDD outlines the following major subsystems/modules:

- **Authentication Subsystem:** Responsible for user registration, login, logout, and role-based access control. Test cases for user authentication and role-based access (TC-AUTH-xxx) directly target this module, suggesting a high level of functional coverage.
- **Menu Management Subsystem:** Handles the creation, updating, and deletion of menu items, as well as categorization, pricing, and availability. Test cases for menu management (TC-MENU-xxx) aim to cover all CRUD operations and display functionalities within this module.
- **Order Processing Subsystem:** Manages the entire order lifecycle, including cart management, order submission, tracking, and status updates. A comprehensive set

of test cases (TC-ORDER-xxx) has been designed to cover various scenarios within this critical module.

- **Payment Subsystem:** Deals with financial transactions, digital wallet, and transaction history. Test cases for the payment system (TC-PAY-xxx) are designed to verify all aspects of cashless transactions and balance management.
- **Notification Subsystem:** Delivers real-time notifications. Test cases for notifications (TC-NOTIF-xxx) aim to verify the delivery of various types of alerts and updates.
- **Reporting and Analytics Subsystem:** Generates various reports. Test cases for reporting (TC-REPORT-xxx) are designed to ensure the accuracy and proper generation of sales, consumption, and financial reports.
- **User Profile Management Subsystem:** Allows users to view and update their profile. Test cases for user profile management (TC-PROFILE-xxx) cover the update and preference management functionalities.
- **Inventory Management Subsystem:** Tracks stock levels and provides alerts. Test cases for inventory management (TC-INV-xxx) are designed to verify stock tracking and alert mechanisms.

Given the detailed test case design that addresses functionalities within each of these modules, it is estimated that the **functional test cases provide substantial coverage across all major modules** of the Mess Management System. Specific code coverage percentages would be determined through automated testing tools during the execution phase.

5.3 UI Screen Coverage

UI screen coverage refers to the extent to which the user interfaces of the application have been tested. The SDD provides details on the Human Interface Design, specifying two main interfaces:

- **Mobile Application Interface (Flutter):** This interface is designed for students and faculty. The test cases include scenarios that involve navigating through menu browsing, order placement, payment processes, profile management, and order tracking screens. This aims to cover the primary user flows and interactions within the mobile application.
- **Web Portal Interface (React.js):** This administrative interface is for Mess Managers, Menu Managers, and Kitchen Staff. Test cases cover functionalities such as menu management, order processing, reporting, user management, and inventory management, which directly correspond to the screens and interactions within the web portal.

Based on the design of UI/UX test cases (e.g., TC-UI-001, TC-UI-002) and the functional test cases that require interaction with specific screens, it is estimated that **all critical UI screens and their associated functionalities for both the mobile application and the web portal are covered** by the designed test cases. This includes testing for responsiveness across different devices and ensuring intuitive navigation and user experience. Visual and interactive elements on each screen are implicitly covered by the functional tests that require interaction with these elements.

6 Traceability Matrix

The Traceability Matrix provides a clear mapping between the requirements defined in the Software Requirements Specification (SRS) and the test cases designed to verify those requirements. This matrix ensures that every requirement is covered by at least one test case, thereby demonstrating comprehensive test coverage and facilitating impact analysis for any changes in requirements or test cases.

6.1 Requirement to Test Case Mapping

Requirement ID	Description (from SRS)	Corresponding Test Case IDs
Functional Requirements		
FR-AUTH-001	System shall allow new users to register with a unique username and password.	TC-AUTH-001, TC-AUTH-002
FR-AUTH-002	System shall allow registered users to log in using their credentials.	TC-AUTH-003, TC-AUTH-004
FR-AUTH-003	System shall provide a password recovery mechanism.	TC-AUTH-005
FR-AUTH-004	System shall enforce strong password policies.	TC-AUTH-006
FR-AUTH-005	System shall support role-based access control (Student/Faculty, Mess Manager, Menu Manager, Kitchen Staff).	TC-AUTH-007, TC-AUTH-008
FR-MENU-001	System shall display daily/weekly menus.	TC-MENU-001
FR-MENU-002	System shall categorize food items.	TC-MENU-002
FR-MENU-003	System shall display pricing and availability information for each item.	TC-MENU-003
FR-MENU-004	System shall display nutritional information for each item.	TC-MENU-004
FR-MENU-005	Menu Managers shall be able to create, update, and delete menu items.	TC-MENU-005, TC-MENU-006, TC-MENU-007
FR-ORDER-001	Students shall be able to select and customize meal items.	TC-ORDER-001
FR-ORDER-002	System shall allow users to manage items in their cart.	TC-ORDER-002
FR-ORDER-003	System shall provide order confirmation and tracking.	TC-ORDER-003
FR-ORDER-004	System shall maintain order history and allow reordering.	TC-ORDER-004

Requirement ID	Description (from SRS)	Corresponding Test Case IDs
FR-ORDER-005	Kitchen Staff shall be able to view pending orders.	TC-ORDER-005
FR-ORDER-006	Kitchen Staff shall be able to update order status.	TC-ORDER-006
FR-ORDER-007	Kitchen Staff shall be able to deliver orders.	TC-ORDER-007
FR-PAY-001	System shall include a digital wallet for cashless transactions.	TC-PAY-001
FR-PAY-002	System shall provide options for balance top-up.	TC-PAY-002
FR-PAY-003	System shall maintain transaction history.	TC-PAY-003
FR-PAY-004	System shall generate payment receipts.	TC-PAY-004
FR-REPORT-001	System shall generate sales reports.	TC-REPORT-001
FR-REPORT-002	System shall generate reports on consumption trends.	TC-REPORT-002
FR-REPORT-003	System shall generate financial summaries.	TC-REPORT-003
FR-REPORT-004	System shall provide feedback analysis.	TC-REPORT-004
FR-NOTIF-001	System shall send order status updates.	TC-NOTIF-001
FR-NOTIF-002	System shall send low balance alerts.	TC-NOTIF-002
FR-NOTIF-003	System shall send special menu announcements.	TC-NOTIF-003
FR-NOTIF-004	System shall send important system updates.	TC-NOTIF-004
FR-PROFILE-001	Users shall be able to view and update their profile information.	TC-PROFILE-001
FR-PROFILE-002	Users shall be able to manage their preferences and settings.	TC-PROFILE-002
FR-INV-001	System shall track stock levels.	TC-INV-001
FR-INV-002	System shall provide low inventory alerts.	TC-INV-002
FR-INV-003	System shall analyze consumption patterns for inventory.	TC-INV-003
FR-INV-004	System shall provide metrics for waste reduction.	TC-INV-004
Non-Functional Requirements		
NFR-PERF-001	Response Time: System shall respond to user requests within 2 seconds under normal load.	TC-PERF-001

Requirement ID	Description (from SRS)	Corresponding Test Case IDs
NFR-PERF-002	Throughput: System shall handle 100 concurrent users without degradation in performance.	TC-PERF-002
NFR-SAFE-002	Data Safety: System shall ensure data integrity and prevent data loss.	(Covered by functional tests, e.g., TC-PAY-002, TC-PAY-003)
NFR-SEC-001	Authentication Security: System shall secure user authentication processes.	TC-SEC-001
NFR-QUAL-001	Usability: System shall be intuitive and easy to use for all user roles.	TC-UI-001, TC-UI-002

7 Conclusion

7.1 Summary of Testing Outcomes

This Software Testing Report outlines the planned testing activities for the Mess Management System, a digital solution designed to modernize food service management at Namal University. The testing strategy encompassed a multi-level approach, including unit, integration, system, and acceptance testing, alongside specialized UI/UX, performance, and security considerations. A comprehensive set of 45 test cases has been meticulously designed, directly traceable to the functional and non-functional requirements detailed in the Software Requirements Specification (SRS) and the architectural components outlined in the Software Design Document (SDD).

As of the generation of this report, the test cases are in a `Pending` state, awaiting execution. Therefore, no actual test results (Passed/Failed) or bug reports are available yet. However, the detailed test case design and the robust traceability matrix confirm that all identified requirements have corresponding verification steps, ensuring a high degree of requirement-based test coverage. The estimated module and UI screen coverage also indicate a thorough approach to validating the systems various components and user interfaces.

7.2 Current System Quality Level

Based on the current stage of development and testing (pre-execution), the systems quality level can be assessed based on the comprehensiveness of the test planning and design. The detailed breakdown of test objectives, scope, and types of testing, coupled with the granular test case design, suggests a strong foundation for achieving a high-quality software product. The emphasis on both functional correctness and non-functional attributes like performance, security, and usability, as derived from the SRS and SDD, indicates a commitment to delivering a robust and user-friendly system. However, the actual quality level will only be quantifiable after the test execution phase, where defects are identified, tracked, and resolved.

7.3 Known Issues or Limitations

At this pre-execution stage, there are no known issues or limitations arising from testing activities. Any issues or limitations will be documented in the `Bug/Error List` section (Section 6) upon discovery during test execution. Potential limitations that might arise during testing could include:

- **Performance bottlenecks:** If the system does not meet the specified response time or throughput requirements under load.
- **Security vulnerabilities:** If any weaknesses in authentication, data encryption, or application security are identified.
- **Usability challenges:** If user feedback indicates difficulties in navigating or interacting with the mobile or web interfaces.
- **Integration failures:** Issues arising from the interaction between different modules or external services.

7.4 Recommendations for Future Testing/Improvements

To further enhance the quality and reliability of the Mess Management System, the following recommendations are put forth for future testing phases and continuous improvement:

- **Execute all designed test cases:** Prioritize and systematically execute all test cases outlined in this report, meticulously documenting actual results and any deviations.
- **Thorough defect management:** Implement a rigorous defect management process, ensuring all identified bugs are logged, prioritized, assigned, fixed, retested, and closed.
- **Automated Testing Implementation:** Invest in developing automated test scripts for unit, integration, and regression testing. This will significantly reduce manual effort, improve testing efficiency, and enable more frequent test cycles.
- **Performance Testing with Realistic Load:** Conduct dedicated performance testing with realistic user loads and data volumes to identify and address any scalability or performance bottlenecks before deployment.
- **Security Audits and Penetration Testing:** Engage in formal security audits and penetration testing by independent security experts to uncover advanced vulnerabilities and ensure the systems resilience against cyber threats.
- **User Acceptance Testing (UAT) with Diverse Users:** Conduct UAT with a diverse group of end-users (students, mess staff) to gather comprehensive feedback on usability, functionality, and overall user experience in a real-world context.
- **Continuous Integration/Continuous Deployment (CI/CD) Integration:** Integrate testing into a CI/CD pipeline to enable automated testing with every code commit, ensuring early detection of defects and faster feedback loops.
- **Post-Deployment Monitoring and Feedback:** Establish robust monitoring tools post-deployment to track system performance, user behavior, and identify any issues in the production environment. Implement a feedback mechanism for users to report issues or suggest improvements.
- **Regular Regression Testing:** Conduct regular regression testing after every new feature development, bug fix, or system update to ensure that existing functionalities remain intact and no new defects are introduced.
- **Documentation Updates:** Continuously update the SRS, SDD, and this Software Testing Report with any changes, new findings, or lessons learned throughout the project lifecycle.

By adhering to these recommendations, the Mess Management System can achieve a higher level of quality, stability, and user satisfaction, ensuring its long-term success and effectiveness at Namal University.