**DINESWIFT REQUIREMENTS TRACEABILITY MATRIX (P1 - PHASE 1)**

**Document Version:** 1.0.0 **Date:** October, 13th, 2025

**Status:** Approved with Component Specification

**Authors:** Product Management Team

**Revision Focus:** Clear component responsibility assignment and phased implementation

# 









|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **RegNo** | **EMAIL** | **PHONE** |
| Mushabe Moses | 23/U/12131/EVE | Mosesmushae9@gmail.com | 0752307875 |
| Drate Hillary | 23/U/23611 | dratehillary@gmail.om | 0758235980 |
| Mukyala Dorcus Nandy | 23/U/11911/EVE | mukyaladorcus@gmail.com | 0755011795 |
| Kiyimba Fahad | 23/U/0628 | kiyimbafwitty@gmail.com | 0762938957 |

This document provides the high-level traceability from Use Cases to Requirements, Test Cases, and Sprints for all components in Phase 1 of the DineSwift project.

Contents

[1](#_Toc211596314)

[1. HIGH PRIORITY USE CASES TO REQUIREMENTS MAPPING 3](#_Toc211596315)

[1.1. Mobile App Use Cases (P1) 3](#_Toc211596316)

[1.2. Local Server Use Cases (P1) 4](#_Toc211596317)

[1.3. Cloud Server Use Cases (P1) 4](#_Toc211596318)

[1.4. Web App Use Cases (P1) 5](#_Toc211596319)

[2. IMPLEMENTATION PRIORITY MATRIX 6](#_Toc211596320)

[2.1. Quadrant 1: High Business Value, High Complexity (Start Immediately) 6](#_Toc211596321)

[2.2. Quadrant 2: High Business Value, Low Complexity (Quick Wins) 6](#_Toc211596322)

[3. DETAILED TEST CASE DEFINITIONS (P1) 7](#_Toc211596323)

[3.1. Mobile App Test Cases (P1) 7](#_Toc211596324)

[3.2. Local Server Test Cases (P1) 11](#_Toc211596325)

[3.3. Cloud Server Test Cases (P1) 13](#_Toc211596326)

[3.4. Web App Test Cases (P1) 16](#_Toc211596327)

[4. DETAILED TRACEABILITY SUMMARY 18](#_Toc211596328)

[4.1. MOBILE-APP Traceability Details 18](#_Toc211596329)

[4.2. LOCAL-SERVER Traceability Details 19](#_Toc211596330)

[4.3. CLOUD-SERVER Traceability Details 19](#_Toc211596331)

[4.4. WEB-APP Traceability Details 20](#_Toc211596332)

[5. SPRINT PLANNING MATRIX 21](#_Toc211596333)

[5.1. Sprint 1: Foundation (Weeks 1-2) 21](#_Toc211596334)

[5.2. Sprint 2: Payments & Bookings (Weeks 3-4) 21](#_Toc211596335)

[6. RISK ASSESSMENT BY USE CASE 23](#_Toc211596336)

[6.1. High Risk Use Cases 23](#_Toc211596337)

[6.2. Medium Risk Use Cases 23](#_Toc211596338)

[7. SUCCESS METRICS TRACKING 24](#_Toc211596339)

# HIGH PRIORITY USE CASES TO REQUIREMENTS MAPPING

## Mobile App Use Cases (P1)

High Priority Use Cases to Requirements Mapping

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Functional Requirements** | **Non-Functional Requirements** | **Priority** | **Test Case ID** | **Sprint** |
| **UC-MOB-ORDER-001** | Scan QR Code | MOBILE-APP-FR-001-P1 | MOBILE-NFR-301-P1, MOBILE-NFR-302-P1 | High | TC-MOB-001 | 1 |
| **UC-MOB-ORDER-002** | Browse Menu | MOBILE-APP-FR-002-P1 | MOBILE-NFR-001-P1, MOBILE-NFR-002-P1 | High | TC-MOB-002 | 1 |
| **UC-MOB-ORDER-003** | Build Order | MOBILE-APP-FR-002-P1 | MOBILE-NFR-002-P1 | High | TC-MOB-003 | 1 |
| **UC-MOB-BOOK-001** | Search Available Tables | MOBILE-APP-FR-101-P1 | MOBILE-NFR-001-P1 | High | TC-MOB-007 | 1 |
| **UC-MOB-BOOK-002** | View Restaurant Layout | MOBILE-APP-FR-101-P1 | MOBILE-NFR-002-P1 | High | TC-MOB-008 | 1 |
| **UC-MOB-BOOK-003** | Select Table & Time | MOBILE-APP-FR-101-P1 | MOBILE-NFR-002-P1 | High | TC-MOB-009 | 1 |
| **UC-MOB-ORDER-004** | Process Payment | MOBILE-APP-FR-004-P1 | MOBILE-NFR-101-P1, MOBILE-NFR-102-P1 | High | TC-MOB-004 | 2 |
| **UC-MOB-ORDER-005** | Display OTP Code | MOBILE-APP-FR-005-P1 | MOBILE-NFR-002-P1 | High | TC-MOB-005 | 2 |
| **UC-MOB-ORDER-006** | Track Order Status | MOBILE-APP-FR-006-P1 | MOBILE-NFR-002-P1 | High | TC-MOB-006 | 2 |
| **UC-MOB-BOOK-004** | Pay Deposit | MOBILE-APP-FR-102-P1 | MOBILE-NFR-101-P1 | High | TC-MOB-010 | 2 |
| **UC-MOB-BOOK-005** | Display Digital Ticket | MOBILE-APP-FR-103-P1 | MOBILE-NFR-002-P1 | High | TC-MOB-011 | 2 |

## Local Server Use Cases (P1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Functional Requirements** | **Non-Functional Requirements** | **Priority** | **Test Case ID** | **Sprint** |
| **UC-LOCAL-ORDER-101** | Cache Menu Data | LOCAL-FR-001-P1 | LOCAL-NFR-201-P1, LOCAL-NFR-101-P1 | High | TC-LOCAL-001 | 1 |
| **UC-LOCAL-ORDER-104** | Process Offline Orders | LOCAL-FR-004-P1 | LOCAL-NFR-201-P1 | High | TC-LOCAL-004 | 1 |
| **UC-LOCAL-ORDER-105** | Sync Orders to Cloud | LOCAL-FR-005-P1 | LOCAL-NFR-202-P1 | High | TC-LOCAL-005 | 1 |
| **UC-LOCAL-ORDER-102** | Generate OTP Codes | LOCAL-FR-002-P1 | LOCAL-NFR-001-P1 | High | TC-LOCAL-002 | 2 |
| **UC-LOCAL-ORDER-103** | Calculate Preparation ETA | LOCAL-FR-003-P1 | LOCAL-NFR-001-P1 | High | TC-LOCAL-003 | 2 |
| **UC-LOCAL-BOOK-101** | Apply Deposit to Bill | LOCAL-FR-104-P1 | LOCAL-NFR-001-P1 | High | TC-LOCAL-006 | 2 |

## Cloud Server Use Cases (P1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Functional Requirements** | **Non-Functional Requirements** | **Priority** | **Test Case ID** | **Sprint** |
| **UC-CLOUD-ORDER-202** | Store Order Records | CLOUD-DR-001-P1 | CLOUD-NFR-002-P1, CLOUD-NFR-005-P1 | High | TC-CLOUD-002 | 1 |
| **UC-CLOUD-MENU-203** | Maintain Menu Master | CLOUD-FR-001-P1 | CLOUD-NFR-001-P1 | High | TC-CLOUD-003 | 1 |
| **UC-CLOUD-BOOK-201** | Manage Booking Calendar | CLOUD-FR-102-P1 | CLOUD-NFR-001-P1 | High | TC-CLOUD-004 | 1 |
| **UC-CLOUD-PAYMENT-201** | Validate Payment | CLOUD-FR-002-P1 | CLOUD-NFR-101-P1, CLOUD-NFR-102-P1 | High | TC-CLOUD-001 | 2 |
| **UC-CLOUD-BOOK-202** | Process Deposit Payments | CLOUD-FR-103-P1 | CLOUD-NFR-101-P1 | High | TC-CLOUD-005 | 2 |
| **UC-CLOUD-BOOK-203** | Generate Digital Tickets | CLOUD-FR-104-P1 | CLOUD-NFR-002-P1 | High | HTC-CLOUD-006 | 2 |

## Web App Use Cases (P1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Use Case ID** | **Use Case Name** | **Functional Requirements** | **Non-Functional Requirements** | **Priority** | **Test Case ID** | **Sprint** |
| **UC-WEB-KITCHEN-301** | View Kitchen Orders | WEB-APP-FR-001-P1 | WEB-NFR-001-P1 | High | TC-WEB-001 | 1 |
| **UC-WEB-KITCHEN-302** | Update Order Status | WEB-APP-FR-002-P1 | WEB-NFR-001-P1 | High | TC-WEB-002 | 1 |
| **UC-WEB-KITCHEN-303** | Mark Order Ready | WEB-APP-FR-003-P1 | WEB-NFR-001-P1 | High | TC-WEB-003 | 1 |
| **UC-WEB-HOST-304** | Validate Booking on Arrival | WEB-APP-FR-004-P1 | WEB-NFR-001-P1 | High | TC-WEB-004 | 2 |
| **UC-WEB-HOST-305** | Apply Deposit to Bill | WEB-APP-FR-005-P1 | WEB-NFR-001-P1 | High | TC-WEB-005 | 2 |

# IMPLEMENTATION PRIORITY MATRIX

## **Quadrant 1: High Business Value, High Complexity (Start Immediately)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Components** | **Requirements** | **Effort** | **Business Value** |
| **UC-MOB-ORDER-001** | Mobile App, Local Server, Cloud Server | MOBILE-APP-FR-001-P1, LOCAL-FR-001-P1, CLOUD-FR-001-P1 | High | Critical |
| **UC-MOB-ORDER-004** | Mobile App, Cloud Server | MOBILE-APP-FR-004-P1, CLOUD-FR-002-P1 | High | Critical |
| **UC-CLOUD-PAYMENT-201** | Cloud Server | CLOUD-FR-002-P1, CLOUD-NFR-101-P1 | High | Critical |

## Quadrant 2: High Business Value, Low Complexity (Quick Wins)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Components** | **Requirements** | **Effort** | **Business Value** |
| **UC-MOB-ORDER-002** | Mobile App, Local Server | MOBILE-APP-FR-002-P1, LOCAL-FR-001-P1 | Medium | High |
| **UC-WEB-KITCHEN-301** | Web App, Local Server | WEB-APP-FR-001-P1, LOCAL-FR-001-P1 | Low | High |
| **UC-MOB-BOOK-001** | Mobile App, Cloud Server | MOBILE-APP-FR-101-P1, CLOUD-FR-102-P1 | Medium | High |

# DETAILED TEST CASE DEFINITIONS (P1)

## Mobile App Test Cases (P1)

These test cases validate the core customer ordering and booking flow on the mobile application, focusing on performance, resilience, and security.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** |
| **TC-MOB-001** | QR code scanning success rate >95% | Valid QR code is available. | 1. Attempt to scan the QR code 100 times under various light/distance conditions. | Successful scan and restaurant/table ID extraction in at least 96 attempts. |
| **TC-MOB-001A** | Offline QR code handling with cached data | App has successfully synced the menu once. Network connectivity is disabled. | 1. Scan a valid QR code while offline. | The app opens the cached menu associated with the scanned table/restaurant without a network error. |
| **TC-MOB-001B** | Error handling for invalid QR codes | Network is enabled. | 1. Scan an invalid QR code (e.g., a generic barcode or a corrupted QR). | A clear, user-friendly error message is displayed (e.g., "Invalid QR Code") with an option for manual entry. |
| **TC-MOB-002** | Menu loading performance <3 seconds | App is connected to the Local Server. | 1. Scan QR to load the menu. 2. Measure the time from scan complete to full menu render. | The full menu content loads and renders in **less than 3 seconds** (MOBILE-NFR-001-P1). |
| **TC-MOB-002A** | Offline menu browsing capability | Menu is cached. Device has no connectivity. | 1. Browse through all menu categories. 2. View item details. | All cached menu data is displayed accurately, allowing full browsing. |
| **TC-MOB-002B** | Shopping cart functionality with real-time totals | User is adding items to the cart. | 1. Add multiple items with different prices. 2. Verify the total price updates instantly and correctly. | Cart total, including tax, is calculated and updated in real-time as items are added/removed. |
| **TC-MOB-003** | Order building with real-time calculations | User is modifying items in the cart. | 1. Add a high-cost modifier. 2. Change the item quantity. 3. Verify the subtotals update instantly. | Subtotals and taxes are calculated correctly and in real-time for all item additions and modifications. |
| **TC-MOB-003A** | Tax calculation accuracy | Cart contains items. | 1. Review the calculated sales tax against the known tax rate. | The displayed tax amount is mathematically accurate according to the configured rate. |
| **TC-MOB-003B** | Special instructions handling | User is on the item detail screen. | 1. Enter a long, multi-line special instruction (e.g., "No onions, extra sauce, and please cut in half"). 2. Save and review in the cart. | Instructions are saved, displayed in the cart summary, and successfully transmitted with the order. |
| **TC-MOB-004** | Payment success rate >98% | User submits a valid payment through the integrated gateway (e.g., Momo). | 1. Execute 100 successful payment attempts with valid credentials. | At least 99 out of 100 payments successfully process and receive confirmation from the Cloud Server. |
| **TC-MOB-004A** | Payment failure recovery with cart preservation | Payment gateway returns a specific failure code (e.g., "Insufficient Funds"). | 1. Initiate a payment that is guaranteed to fail. 2. Check the cart content after failure. | The payment failure is clearly communicated. The user is returned to the cart, and all selected items/modifications are preserved. |
| **TC-MOB-004B** | Security compliance validation | Order processed successfully. | 1. Inspect device storage and network traffic logs during payment. | No sensitive payment information (card numbers, tokens, etc.) is stored locally on the device or transmitted unencrypted (MOBILE-NFR-101-P1, MOBILE-NFR-102-P1). |
| **TC-MOB-005** | OTP display with expiration timer | Order successfully placed and paid. | 1. View the OTP display screen. 2. Monitor the countdown timer. | The OTP is displayed clearly alongside a visible and accurate 5-minute countdown timer. |
| **TC-MOB-005A** | OTP refresh functionality | The expiration timer reaches zero. | 1. Wait for the timer to expire. | A new, unique OTP code is instantly displayed, and the timer is reset to 5 minutes. |
| **TC-MOB-005B** | OTP visibility and copy functionality | OTP is displayed. | 1. Verify OTP is readable in various light conditions. 2. Attempt to copy the OTP text. | The OTP is highly legible. The user can successfully copy the code to the device clipboard. |
| **TC-MOB-006** | Real-time order status updates | Kitchen staff updates the order status (e.g., from "Received" to "In Prep"). | 1. Monitor the order status on the mobile app. | The status update is reflected on the mobile app within **2 seconds** of the update being performed by staff. |
| **TC-MOB-006A** | Status accuracy validation | Staff updates the order to "Ready for Pickup." | 1. Confirm the status displayed on the Mobile App matches the status in the Cloud DB. | The displayed status (e.g., "Ready") is consistent across all application layers. |
| **TC-MOB-006B** | Offline status access | Customer views the order status screen while offline. | 1. Place an order, go offline, and navigate to the order status. | The last known order status is retrieved and displayed with a clear indication that it may be outdated (e.g., "Last updated: 10:30 AM"). |
| **TC-MOB-007** | Table search and filtering functionality | Multiple tables are available and unavailable in the system. | 1. Search for a specific party size. 2. Filter results by an attribute (e.g., 'window seat'). | The search returns only available tables matching all specified criteria and filters. |
| **TC-MOB-007A** | Real-time availability accuracy | Staff manually marks a table as "Reserved" in the Web App. | 1. Immediately refresh the table search view on the Mobile App. | The newly reserved table is marked as unavailable or removed from the list within seconds. |
| **TC-MOB-007B** | Performance validation <3 seconds | Booking search query is executed. | 1. Measure the time taken from submitting search criteria to displaying results. | Search results are loaded and displayed in **less than 3 seconds** (MOBILE-NFR-001-P1). |
| **TC-MOB-008** | Layout rendering accuracy | Customer is viewing the restaurant layout screen. | 1. Compare the rendered digital layout to the actual restaurant floor plan. | The displayed layout accurately reflects the position and relative size of tables. |
| **TC-MOB-008A** | Interactive table selection | Customer taps an available table on the layout. | 1. Tap an available table. 2. Tap an unavailable table. | Tapping an available table proceeds to the confirmation screen. Tapping an unavailable table displays a status message. |
| **TC-MOB-008B** | Maintenance status display | A table is temporarily marked "Out of Service" by management. | 1. View the restaurant layout. | The specific table is visually marked (e.g., grayed out) and non-selectable, showing its maintenance status. |
| **TC-MOB-009** | Table and time selection workflow | Table search is complete. | 1. Select an available table and a time slot. 2. Proceed to the confirmation/deposit screen. | The correct table and time are carried through the workflow without error, and the deposit amount is shown. |
| **TC-MOB-009A** | Conflict detection and prevention | Two users attempt to book the same table/time simultaneously. | 1. User A initiates booking. 2. User B tries to select the same table before User A confirms. | The system detects the conflict and prevents User B from proceeding, prompting them to select an alternative. |
| **TC-MOB-009B** | Special requests handling | User is on the final booking confirmation screen. | 1. Enter a detailed special request (e.g., "Need a high chair"). 2. Complete the booking. | The special request is correctly recorded in the Cloud Booking record. |
| **TC-MOB-010** | Deposit payment processing | User proceeds from table selection to payment. | 1. Submit payment for the deposit. 2. Verify payment success. | The deposit payment is processed securely via the gateway, and a confirmation is received. |
| **TC-MOB-010A** | Refund policy communication | User is on the deposit payment screen. | 1. Review the refund policy link/text. | The refund policy details (e.g., refundable if canceled before 24 hours) are clearly displayed and accurate. |
| **TC-MOB-010B** | Payment security validation | Deposit payment is executed. | 1. Inspect network traffic and logging during the deposit transaction. | The payment transaction adheres to the same security standards as TC-MOB-004B (no local storage of payment data). |
| **TC-MOB-011** | Digital ticket display with QR code | Booking deposit successfully paid. | 1. Navigate to the digital ticket screen. | A clean, readable digital ticket is displayed, including a unique QR code for check-in. |
| **TC-MOB-011A** | Offline ticket access | Device has no connectivity. | 1. Navigate to the digital ticket screen. | The ticket, including the QR code, is cached and displayed successfully for check-in. |
| **TC-MOB-011B** | Check-in QR functionality | Staff uses the Host Web App to check in a customer. | 1. Staff scans the customer's digital ticket QR code. | The Host Web App successfully reads the QR code and loads the corresponding booking record for validation. |

## Local Server Test Cases (P1)

These cases validate the local environment's resilience, caching, and offline functionality, which are crucial for restaurant operations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** |
| **TC-LOCAL-001** | Menu cache synchronization with version control | Menu is updated on the Cloud Server. | 1. Observe the Local Server's menu version identifier. 2. Verify the server pulls the new menu data. | Local Server detects the new Cloud version and successfully downloads/updates the local cache. |
| **TC-LOCAL-001A** | Offline menu access performance | Local Server is offline. Mobile app requests cached data. | 1. Simulate 100 concurrent mobile requests for menu data. 2. Measure the response time for each request. | Menu data is served from the cache with minimal latency, ensuring fast menu browsing performance even when offline. |
| **TC-LOCAL-001B** | Cache encryption validation | Local Server cache storage is physically accessible. | 1. Attempt to read the cached menu file directly from the file system. | The data retrieved is encrypted/obfuscated at rest (LOCAL-NFR-101-P1) and cannot be read as plain text. |
| **TC-LOCAL-002** | OTP generation performance <500ms | A payment success notification is received. | 1. Measure time from payment success notification to OTP generation completion. | The Local Server generates the unique OTP and makes it available for display in **less than 500ms** (LOCAL-NFR-001-P1). |
| **TC-LOCAL-002A** | OTP uniqueness and expiration validation | Multiple concurrent orders are placed. | 1. Generate 10 OTPs concurrently. 2. Validate that all 10 are unique. 3. Verify that all 10 expire precisely after 5 minutes. | All OTPs generated are unique and adhere strictly to the 5-minute expiration rule. |
| **TC-LOCAL-002B** | Order routing to kitchen timing | OTP is successfully generated. | 1. Measure time from OTP generation to the order appearing in the Kitchen Web App. | The order is successfully routed to the Kitchen Web App within the required performance window (e.g., <500ms of confirmation). |
| **TC-LOCAL-003** | ETA calculation accuracy | System has historical data and current kitchen load data. | 1. Submit an order (A) and measure the estimated ETA. 2. Submit a large order (B) and measure its ETA. | Order B's ETA is correctly longer than Order A's, demonstrating that current kitchen load is factored into the calculation. |
| **TC-LOCAL-003A** | Real-time kitchen load factoring | Kitchen staff marks 5 items as "In Prep." | 1. Submit a new order immediately and observe the calculated ETA. | The ETA for the new order is dynamically adjusted (increased) to reflect the recent increase in kitchen load. |
| **TC-LOCAL-003B** | Performance under load | Local Server is under maximum expected load (concurrent orders/lookups). | 1. Execute multiple ETA calculations simultaneously. | ETA calculations are processed within system performance requirements without noticeable delay. |
| **TC-LOCAL-004** | Offline order processing capability | Local Server connectivity is severed. | 1. Mobile App submits an order to the Local Server. | The Local Server successfully processes the order, generates a local ID, and queues it for later sync. |
| **TC-LOCAL-004A** | Order sync after connectivity restore | Multiple offline orders are queued. | 1. Re-establish network connectivity. 2. Monitor the Local Server sync queue. | All queued orders are automatically and successfully synchronized to the Cloud Server. |
| **TC-LOCAL-004B** | CRDT conflict resolution | Two orders are modified on two separate offline devices, causing potential data conflict upon sync. | 1. Restore connectivity and force sync. 2. Verify the final merged order data in the Cloud. | The Conflict-Free Replicated Data Type (CRDT) logic resolves the conflict gracefully, resulting in a consistent final record without data loss. |
| **TC-LOCAL-005** | Order sync functionality and reliability | New orders are processed locally. | 1. Monitor the sync process during normal operation. 2. Verify orders are passed to the cloud. | Orders are reliably synchronized to the Cloud Server as soon as connectivity is available. |
| **TC-LOCAL-005A** | Sync performance validation | A batch of 50 queued orders is synced to the cloud. | 1. Measure the total time taken to synchronize the entire batch. | The sync operation completes within the defined performance threshold (e.g., bulk sync takes less than 2 minutes). |
| **TC-LOCAL-005B** | Data integrity during sync | Compare the local and cloud records after synchronization. | 1. Verify that all items, quantities, special instructions, and timestamps exactly match the local record. | The synchronization process preserves the integrity of all data fields from local storage to the cloud. |
| **TC-LOCAL-006** | Deposit application accuracy | Booking deposit is recorded in the Cloud. | 1. Host processes a bill for a customer with a deposit. 2. Verify the final bill amount. | The Local Server successfully retrieves the deposit record from the Cloud and **accurately deducts** the amount from the final bill. |
| **TC-LOCAL-006A** | Manual adjustment capability | Host attempts to manually adjust the applied deposit amount. | 1. Host attempts to override the automatically applied deposit value (e.g., set to $0 or double it). | The system allows the manual adjustment with required supervisor override/audit logging. |
| **TC-LOCAL-006B** | Validation workflow | Host applies deposit to a bill where the booking has been marked as a "no-show." | 1. Attempt to apply the deposit. | The system prevents the deposit from being applied and displays an error due to the booking's invalid status. |

## Cloud Server Test Cases (P1)

These cases focus on security, data persistence, and master data management in the central cloud environment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** |
| **TC-CLOUD-001** | Payment validation and idempotent handling | Multiple payment requests for the same transaction ID are received. | 1. Submit the same successful payment token twice (rapidly). | The Cloud Server performs the validation call only once and returns a successful confirmation for both requests (idempotent). |
| **TC-CLOUD-001A** | PCI DSS compliance verification | Payment processing system is active. | 1. Conduct an external security audit focusing on payment data handling (e.g., tokenization, storage). | The payment processing infrastructure meets all defined PCI DSS Level 1 compliance requirements (CLOUD-NFR-101-P1). |
| **TC-CLOUD-001B** | Role-based access control validation | An unauthorized user attempts to access order data. | 1. Log in as a Kitchen Staff and try to access Finance/Payment data endpoints. | The system denies access based on the user's role (CLOUD-NFR-102-P1), logging the attempted unauthorized access. |
| **TC-CLOUD-002** | Order storage reliability and performance | Large volume of orders is being synced from Local Servers. | 1. Insert 10,000 order records into the database. 2. Verify all records are present and correct. | All order records are stored persistently and reliably without data loss or corruption. |
| **TC-CLOUD-002A** | Query performance <100ms validation | Execute a complex query (e.g., "Find all orders from Table 5 in the last 7 days"). | 1. Measure the database query execution time under normal load. | The query returns results within the required **100ms P95** threshold (CLOUD-NFR-002-P1). |
| **TC-CLOUD-002B** | Backup and recovery procedures | Production data is active. | 1. Simulate a database failure. 2. Initiate the recovery procedure using the latest 6-hour backup. | The database is successfully restored to a state no older than the last 6-hour backup point. |
| **TC-CLOUD-003** | QR code resolution accuracy | Cloud maps QR codes to specific restaurant menu URLs. | 1. Submit a QR code ID to the Cloud API. 2. Verify the returned menu URL and table map data. | The API correctly resolves the QR ID to the associated restaurant's menu URL and table identifier. |
| **TC-CLOUD-003A** | Multi-tenant support validation | Two different restaurant tenants are configured. | 1. Tenant A attempts to resolve a QR code belonging to Tenant B. | The system returns an error or a "Tenant Not Found" response, preventing cross-tenant data access (CLOUD-NFR-001-P1). |
| **TC-CLOUD-003B** | Menu update propagation | Menu Master is updated. | 1. Measure the time taken for the update to be signaled and pulled by all connected Local Servers. | The menu update is successfully propagated across the entire network within the specified time (e.g., <5 minutes). |
| **TC-CLOUD-004** | Booking calendar management | Various tables are reserved and open across the calendar. | 1. Query the calendar for a specific date/time/table. 2. Query for global availability. | The calendar accurately reflects all reservations, preventing double-booking and showing correct global availability. |
| **TC-CLOUD-004A** | Deposit requirement enforcement | Booking is attempted for a time/date that requires a deposit. | 1. Attempt to confirm a booking without submitting the deposit payment. | The system prevents the booking from being confirmed until a successful deposit payment is processed. |
| **TC-CLOUD-004B** | Policy communication accuracy | Deposit and Cancellation policies are configured. | 1. Query the Cloud API for the current policy text. 2. Verify the text matches what is displayed to the customer. | The policy communication is accurate and consistently presented across all touchpoints (Mobile App, Web App). |
| **TC-CLOUD-005** | Digital ticket generation with QR codes | Deposit payment is successful. | 1. Verify the generated digital ticket in the database. | A unique, cryptographically secure digital ticket with a valid check-in QR code is generated and linked to the booking record. |
| **TC-CLOUD-005A** | Staff notification system | A booking is successfully confirmed. | 1. Check the staff notification logs (e.g., internal messaging/Web App flag). | The appropriate staff (e.g., Host/Manager) is automatically notified of the new booking. |
| **TC-CLOUD-005B** | Security compliance validation | Digital ticket is generated. | 1. Check the QR code data payload for sensitive customer PII. | The QR code payload contains only the necessary, non-sensitive booking ID for check-in validation. |
| **TC-CLOUD-006** | Advance booking requirement enforcement | Advance booking requirement is set to "Minimum 1 hour." | 1. Attempt to book a table 30 minutes in the future. 2. Attempt to book a table 61 minutes in the future. | The system blocks the 30-minute booking (Test 1 fails) and successfully allows the 61-minute booking (Test 2 succeeds). |
| **TC-CLOUD-006A** | Configuration flexibility validation | Management changes the advance booking requirement (e.g., from 1 hour to 4 hours). | 1. Verify the new requirement is immediately applied and enforced across all new booking requests. | The booking policy can be dynamically updated via configuration and takes effect instantly. |
| **TC-CLOUD-006B** | Performance under load | High concurrent requests to the booking API. | 1. Simulate 100 concurrent availability checks and 10 concurrent booking requests. | The booking system maintains performance and accurately enforces availability constraints under high load. |

## Web App Test Cases (P1)

These cases validate the functional and real-time requirements for the staff-facing Web Applications (Kitchen and Host).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Preconditions** | **Test Steps** | **Expected Result** |
| **TC-WEB-001** | Real-time order display accuracy | A new order is successfully synced from the Local Server. | 1. Verify the new order appears on the Kitchen Order screen. 2. Compare item details, quantity, and special instructions. | The order appears instantly on the screen, and all data fields exactly match the submitted order. |
| **TC-WEB-001A** | One-tap action functionality | An order is in the "Received" state. | 1. Staff clicks the "Start Prep" button once. | The order status updates with a single click without any secondary confirmation dialogs (WEB-NFR-001-P1). |
| **TC-WEB-001B** | Performance during peak hours | Kitchen Web App is under maximum expected concurrent staff usage. | 1. Perform 10 status updates within 5 seconds. | The Web App remains responsive, and all updates are registered without freezing or significant lag. |
| **TC-WEB-002** | Status update functionality | Staff changes an order status (e.g., "In Prep" to "Ready"). | 1. Click the status button. 2. Verify the order moves to the next logical state/column. | The status update is successful and the order is correctly moved/categorized in the UI. |
| **TC-WEB-002A** | One-tap status changes | Staff attempts to move an order from "In Prep" to "Ready." | 1. Click the 'Ready' button once. | The status changes instantly, demonstrating one-tap functionality for common status changes. |
| **TC-WEB-002B** | Real-time propagation to mobile apps | Kitchen staff marks an order as "Ready." | 1. Verify the Mobile App status for that order updates instantly. | The status change is immediately pushed and displayed on the customer's Mobile App (UC-MOB-ORDER-006). |
| **TC-WEB-003** | Order ready marking with timestamps | Staff marks an order as ready. | 1. Verify the order record in the Cloud DB. | The order record is updated with a precise timestamp indicating when the order was marked as ready. |
| **TC-WEB-003A** | OTP synchronization with mobile app | Order is marked as ready. | 1. Verify the corresponding OTP is available for the customer on the Mobile App. | The OTP generation/display is triggered and synchronized correctly with the "Order Ready" status change. |
| **TC-WEB-003B** | One-tap ready confirmation | Staff marks the order ready. | 1. Click the 'Ready' button once. | The action completes instantly, confirming the order is ready with a single tap. |
| **TC-WEB-004** | Booking validation accuracy | Host validates a booking record. | 1. Host enters a known booking ID into the lookup field. 2. Verifies customer name, party size, and table assignment. | The correct and current booking details are retrieved from the Cloud Server and displayed accurately for validation. |
| **TC-WEB-004A** | QR code scanning for validation | Host scans the QR code from the customer's digital ticket. | 1. Host uses the built-in scanner to capture the QR. | The system instantly resolves the QR code to the correct booking record. |
| **TC-WEB-004B** | Manual lookup capability | Host searches by customer name or phone number. | 1. Host enters a partial customer name. | The system returns a filtered list of matching bookings, allowing the Host to manually select the correct one. |
| **TC-WEB-005** | Deposit application to bill | A validated booking with a deposit is linked to a customer's bill. | 1. Host initiates the 'Apply Deposit' action on the final bill screen. | The deposit amount is correctly retrieved from the Cloud and automatically deducted from the Local Server's bill total. |
| **TC-WEB-005A** | One-tap application functionality | Host applies the deposit. | 1. Click the 'Apply Deposit' button once. | The deduction is applied to the bill instantly with a single tap. |
| **TC-WEB-005B** | Manual adjustment capability | Host attempts to override the deposit amount or apply a partial amount. | 1. Host manually enters a value different from the applied deposit. | The system allows the manual adjustment, requiring an administrative override/reason, and updates the bill accordingly. |

# DETAILED TRACEABILITY SUMMARY

## MOBILE-APP Traceability Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Implements (FR/NFR)** | **Dependencies** | **Critical Test Cases** |
| **UC-MOB-ORDER-001:** Scan QR Code | MOBILE-APP-FR-001-P1, MOBILE-NFR-301-P1, MOBILE-NFR-302-P1 | LOCAL-FR-001-P1, CLOUD-FR-001-P1 | HTC-MOB-001, TC-MOB-001A, TC-MOB-001B |
| **UC-MOB-ORDER-002:** Browse Menu | MOBILE-APP-FR-002-P1, MOBILE-NFR-001-P1, MOBILE-NFR-002-P1 | LOCAL-FR-001-P1, CLOUD-FR-001-P1 | HTC-MOB-002, TC-MOB-002A, TC-MOB-002B |
| **UC-MOB-ORDER-003:** Build Order | MOBILE-APP-FR-002-P1, MOBILE-NFR-002-P1 | LOCAL-FR-001-P1, LOCAL-FR-004-P1 | HTC-MOB-003, TC-MOB-003A, TC-MOB-003B |
| **UC-MOB-ORDER-004:** Process Payment | MOBILE-APP-FR-004-P1, MOBILE-NFR-101-P1, MOBILE-NFR-102-P1 | CLOUD-FR-002-P1, LOCAL-FR-002-P1 | HTC-MOB-004, TC-MOB-004A, TC-MOB-004B |
| **UC-MOB-ORDER-005:** Display OTP Code | MOBILE-APP-FR-005-P1, MOBILE-NFR-002-P1 | LOCAL-FR-002-P1, LOCAL-FR-003-P1 | HTC-MOB-005, TC-MOB-005A, TC-MOB-005B |
| **UC-MOB-ORDER-006:** Track Order Status | MOBILE-APP-FR-006-P1, MOBILE-NFR-002-P1 | LOCAL-FR-003-P1, CLOUD-FR-001-P1 | HTC-MOB-006, TC-MOB-006A, TC-MOB-006B |
| **UC-MOB-BOOK-001:** Search Available Tables | MOBILE-APP-FR-101-P1, MOBILE-NFR-001-P1 | CLOUD-FR-102-P1, CLOUD-FR-104-P1 | HTC-MOB-007, TC-MOB-007A, TC-MOB-007B |
| **UC-MOB-BOOK-002:** View Restaurant Layout | MOBILE-APP-FR-101-P1, MOBILE-NFR-002-P1 | CLOUD-FR-102-P1, CLOUD-DR-104-P1 | HTC-MOB-008, TC-MOB-008A, TC-MOB-008B |
| **UC-MOB-BOOK-003:** Select Table & Time | MOBILE-APP-FR-101-P1, MOBILE-NFR-002-P1 | CLOUD-FR-102-P1, CLOUD-FR-103-P1 | HTC-MOB-009, TC-MOB-009A, TC-MOB-009B |
| **UC-MOB-BOOK-004:** Pay Deposit | MOBILE-APP-FR-102-P1, MOBILE-NFR-101-P1 | CLOUD-FR-103-P1, CLOUD-FR-104-P1 | HTC-MOB-010, TC-MOB-010A, TC-MOB-010B |
| **UC-MOB-BOOK-005:** Display Digital Ticket | MOBILE-APP-FR-103-P1, MOBILE-NFR-002-P1 | CLOUD-FR-103-P1, LOCAL-FR-104-P1 | HTC-MOB-011, TC-MOB-011A, TC-MOB-011B |

## LOCAL-SERVER Traceability Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Implements (FR/NFR)** | **Dependencies** | **Critical Test Cases** |
| **UC-LOCAL-ORDER-101:** Cache Menu Data | LOCAL-FR-001-P1, LOCAL-NFR-201-P1, LOCAL-NFR-101-P1 | CLOUD-FR-001-P1, LOCAL-DR-001-P1 | HTC-LOCAL-001, TC-LOCAL-001A, TC-LOCAL-001B |
| **UC-LOCAL-ORDER-102:** Generate OTP Codes | LOCAL-FR-002-P1, LOCAL-NFR-001-P1 | LOCAL-FR-001-P1, CLOUD-FR-002-P1 | HTC-LOCAL-002, TC-LOCAL-002A, TC-LOCAL-002B |
| **UC-LOCAL-ORDER-103:** Calculate Preparation ETA | LOCAL-FR-003-P1, LOCAL-NFR-001-P1 | LOCAL-FR-004-P1, CLOUD-FR-001-P1 | HTC-LOCAL-003, TC-LOCAL-003A, TC-LOCAL-003B |
| **UC-LOCAL-ORDER-104:** Process Offline Orders | LOCAL-FR-004-P1, LOCAL-NFR-201-P1, LOCAL-EH-001-P1 | LOCAL-FR-001-P1, LOCAL-FR-005-P1 | HTC-LOCAL-004, TC-LOCAL-004A, TC-LOCAL-004B |
| **UC-LOCAL-ORDER-105:** Sync Orders to Cloud | LOCAL-FR-005-P1, LOCAL-NFR-202-P1 | CLOUD-FR-002-P1, LOCAL-FR-004-P1 | HTC-LOCAL-005, TC-LOCAL-005A, TC-LOCAL-005B |
| **UC-LOCAL-BOOK-101:** Apply Deposit to Bill | LOCAL-FR-104-P1, LOCAL-NFR-001-P1 | CLOUD-FR-103-P1, WEB-APP-FR-005-P1 | HTC-LOCAL-006, TC-LOCAL-006A, TC-LOCAL-006B |

## CLOUD-SERVER Traceability Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Implements (FR/NFR)** | **Dependencies** | **Critical Test Cases** |
| **UC-CLOUD-PAYMENT-201:** Validate Payment | CLOUD-FR-002-P1, CLOUD-NFR-101-P1, CLOUD-NFR-102-P1 | MOBILE-APP-FR-004-P1, External Momo payment gateway API | HTC-CLOUD-001, TC-CLOUD-001A, TC-CLOUD-001B |
| **UC-CLOUD-ORDER-202:** Store Order Records | CLOUD-DR-001-P1, CLOUD-NFR-002-P1, CLOUD-NFR-005-P1 | LOCAL-FR-005-P1, MOBILE-APP-FR-004-P1 | HTC-CLOUD-002, TC-CLOUD-002A, TC-CLOUD-002B |
| **UC-CLOUD-MENU-203:** Maintain Menu Master | CLOUD-FR-001-P1, CLOUD-NFR-001-P1 | LOCAL-FR-001-P1, Restaurant management system integration | HTC-CLOUD-003, TC-CLOUD-003A, TC-CLOUD-003B |
| **UC-CLOUD-BOOK-201:** Manage Booking Calendar | CLOUD-FR-102-P1, CLOUD-NFR-001-P1 | MOBILE-APP-FR-101-P1, WEB-APP-FR-004-P1 | HTC-CLOUD-004, TC-CLOUD-004A, TC-CLOUD-004B |
| **UC-CLOUD-BOOK-202:** Process Deposit Payments | CLOUD-FR-103-P1, CLOUD-NFR-101-P1 | MOBILE-APP-FR-102-P1, CLOUD-FR-104-P1 | HTC-CLOUD-005, TC-CLOUD-005A, TC-CLOUD-005B |
| **UC-CLOUD-BOOK-203:** Generate Digital Tickets | CLOUD-FR-104-P1, CLOUD-NFR-002-P1 | CLOUD-FR-103-P1, MOBILE-APP-FR-103-P1 | HTC-CLOUD-006, TC-CLOUD-006A, TC-CLOUD-006B |

## WEB-APP Traceability Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Implements (FR/NFR)** | **Dependencies** | **Critical Test Cases** |
| **UC-WEB-KITCHEN-301:** View Kitchen Orders | WEB-APP-FR-001-P1, WEB-NFR-001-P1 | LOCAL-FR-001-P1, CLOUD-FR-001-P1 | HTC-WEB-001, TC-WEB-001A, TC-WEB-001B |
| **UC-WEB-KITCHEN-302:** Update Order Status | WEB-APP-FR-002-P1, WEB-NFR-001-P1 | LOCAL-FR-003-P1, MOBILE-APP-FR-006-P1 | HTC-WEB-002, TC-WEB-002A, TC-WEB-002B |
| **UC-WEB-KITCHEN-303:** Mark Order Ready | WEB-APP-FR-003-P1, WEB-NFR-001-P1 | LOCAL-FR-002-P1, MOBILE-APP-FR-005-P1 | HTC-WEB-003, TC-WEB-003A, TC-WEB-003B |
| **UC-WEB-HOST-304:** Validate Booking | WEB-APP-FR-004-P1, WEB-NFR-001-P1 | CLOUD-FR-102-P1, CLOUD-FR-103-P1 | HTC-WEB-004, TC-WEB-004A, TC-WEB-004B |
| **UC-WEB-HOST-305:** Apply Deposit to Bill | WEB-APP-FR-005-P1, WEB-NFR-001-P1 | LOCAL-FR-104-P1, CLOUD-FR-103-P1 | HTC-WEB-005, TC-WEB-005A, TC-WEB-005B |

# SPRINT PLANNING MATRIX

## Sprint 1: Foundation (Weeks 1-2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Component Team** | **Requirements** | **Acceptance Criteria** |
| **UC-MOB-ORDER-001** | Mobile Team | MOBILE-APP-FR-001-P1 | QR scan success rate >95%**Error! Filename not specified.** |
| **UC-MOB-ORDER-002** | Mobile Team | MOBILE-APP-FR-002-P1 | Menu loads <3 seconds, cart functions |
| **UC-MOB-BOOK-001** | Mobile Team | MOBILE-APP-FR-101-P1 | Table search functional |
| **UC-LOCAL-ORDER-101** | Backend Team | LOCAL-FR-001-P1 | Menu caching with version control |
| **UC-LOCAL-ORDER-104** | Backend Team | LOCAL-FR-004-P1 | Offline order processing |
| **UC-CLOUD-MENU-203** | Cloud Team | CLOUD-FR-001-P1 | QR resolution API ready |
| **UC-CLOUD-ORDER-202** | Cloud Team | CLOUD-DR-001-P1 | Order storage operational |
| **UC-CLOUD-BOOK-201** | Cloud Team | CLOUD-FR-102-P1 | Booking calendar management |
| **UC-WEB-KITCHEN-301** | Web Team | WEB-APP-FR-001-P1 | Kitchen order view operational |
| **UC-WEB-KITCHEN-302** | Web Team | WEB-APP-FR-002-P1 | Order status updates working |

## Sprint 2: Payments & Bookings (Weeks 3-4)

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Component Team** | **Requirements** | **Acceptance Criteria** |
| **UC-MOB-ORDER-004** | Mobile Team | MOBILE-APP-FR-004-P1 | Payment success >98%, PCI compliance |
| **UC-MOB-ORDER-005** | Mobile Team | MOBILE-APP-FR-005-P1 | OTP display functional |
| **UC-MOB-ORDER-006** | Mobile Team | MOBILE-APP-FR-006-P1 | Order tracking working |
| **UC-MOB-BOOK-004** | Mobile Team | MOBILE-APP-FR-102-P1 | Deposit payment working |
| **UC-MOB-BOOK-005** | Mobile Team | MOBILE-APP-FR-103-P1 | Digital tickets displayed |
| **UC-LOCAL-ORDER-102** | Backend Team | LOCAL-FR-002-P1 | OTP generation with expiration |
| **UC-LOCAL-ORDER-103** | Backend Team | LOCAL-FR-003-P1 | ETA calculation accurate |
| **UC-LOCAL-ORDER-105** | Backend Team | LOCAL-FR-005-P1 | Cloud sync functional |
| **UC-LOCAL-BOOK-101** | Backend Team | LOCAL-FR-104-P1 | Deposit application ready |
| **UC-CLOUD-PAYMENT-201** | Cloud Team | CLOUD-FR-002-P1 | Momo integration complete |
| **UC-CLOUD-BOOK-202** | Cloud Team | CLOUD-FR-103-P1 | Deposit processing working |
| **UC-CLOUD-BOOK-203** | Cloud Team | CLOUD-FR-104-P1 | Ticket generation operational |
| **UC-WEB-KITCHEN-303** | Web Team | WEB-APP-FR-003-P1 | Order ready marking functional |
| **UC-WEB-HOST-304** | Web Team | WEB-APP-FR-004-P1 | Booking validation working |
| **UC-WEB-HOST-305** | Web Team | WEB-APP-FR-005-P1 | Deposit to bill application ready |

# RISK ASSESSMENT BY USE CASE

## High Risk Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Risk** | **Mitigation** | **Owner** |
| **UC-MOB-ORDER-004** | Payment gateway integration failure | Early Momo API testing, fallback payment methods | Mobile Team Lead |
| **UC-LOCAL-ORDER-104** | Offline sync conflicts during connectivity restoration | CRDT implementation, manual resolution UI | Backend Team Lead |
| **UC-CLOUD-PAYMENT-201** | PCI DSS compliance issues | Early security review, compliance consultant | Security Lead |
| **UC-CLOUD-BOOK-202** | Deposit payment security and refund processing | Tokenization, secure API endpoints, audit trails | Cloud Team Lead |

## Medium Risk Use Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Risk** | **Mitigation** | **Owner** |
| **UC-MOB-ORDER-001** | QR code reliability in various lighting conditions | Multiple scanning libraries, manual entry fallback | Mobile Team |
| **UC-LOCAL-ORDER-101** | Local server performance under concurrent load | Load testing, hardware specifications validation | DevOps |
| **UC-CLOUD-MENU-203** | Menu data consistency across multiple restaurants | Version control, rollback procedures, validation | Cloud Team |
| **UC-WEB-KITCHEN-301** | Real-time order sync during peak hours | WebSocket fallbacks, polling mechanisms | Web Team |

# SUCCESS METRICS TRACKING

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Primary Metric** | **Target** | **Measurement Method** |
| **UC-MOB-ORDER-001** | QR Scan Success Rate | >95% | Analytics dashboard |
| **UC-MOB-ORDER-002** | Menu Load Time | <3 seconds | Performance testing |
| **UC-MOB-ORDER-004** | Payment Success Rate | >98% | Payment gateway logs |
| **UC-MOB-BOOK-001** | Table Search Accuracy | >99% | Booking validation logs |
| **UC-LOCAL-ORDER-101** | Offline Availability | 99.9% during 8-hour outages | System monitoring |
| **UC-LOCAL-ORDER-104** | Offline Order Sync Time | <2 minutes after connectivity | Sync performance logs |
| **UC-CLOUD-PAYMENT-201** | Payment Validation Time | <2 seconds | API response monitoring |
| **UC-CLOUD-MENU-203** | Menu Update Propagation | <5 minutes to all nodes | Update timing logs |
| **UC-WEB-KITCHEN-301** | Order Display Latency | <2 seconds | Real-time performance monitoring |
| **UC-WEB-HOST-304** | Booking Validation Time | <3 seconds | Validation process timing |