# Test Plan for Restaurant Management System (RMS)

## 1. Introduction

### 1.1 Scope

The test plan is focused on ensuring that the Restaurant Management System (RMS) operates as expected across all modules including menu management, order processing, reservation handling, kitchen coordination, billing, and POS integration. The system will be tested across multiple platforms (desktops and tablets) to ensure compatibility and seamless operation with external hardware like printers, card readers, and cash registers.

**In Scope:**

- Functional testing of core modules (menu, orders, reservations, billing).
- Performance and load testing to ensure scalability.
- Usability testing for both customer and staff interfaces.
- Security testing for sensitive data protection (e.g., customer information and payment data).
- Integration testing with POS systems and hardware devices.

**Out of Scope:**

- Testing of external payment gateways beyond initial integration.
- Hardware testing for specific models of printers or card readers (assumed to be provided by third parties).

### 1.2 Quality Objectives

- **Functionality:** Ensure each module functions as expected and meets the requirements defined in the architecture document.
- **Performance:** Ensure the system handles high traffic efficiently without delays (e.g., during peak restaurant hours).
- **Security:** Verify that sensitive customer and restaurant data (e.g., payment info, order history) are securely handled.
- **Usability:** Verify that the system offers a smooth, user-friendly experience for both customers and restaurant staff.

### 1.3 Roles and Responsibilities

- **Test Manager:** Oversee the test execution and ensure deadlines are met.
- **Test Engineers:** Design test cases, execute tests, report bugs, and validate bug fixes.
- **Developers:** Fix bugs and help identify issues during integration and testing.
- **Product Owners:** Provide clarification on functional requirements during the test cycles.

---

## 2. Test Methodology

### 2.1 Overview

The testing methodology will follow the Agile approach, where testing is conducted iteratively after every development sprint. Testing will include both manual and automated testing based on the scope and nature of the modules.

### 2.2 Test Levels

- **Unit Testing:** Focused on individual components (e.g., API endpoints for menu, orders).
- **Integration Testing:** Ensuring that modules like kitchen coordination and billing work together.
- **System Testing:** Validate the end-to-end workflow from ordering to payment.
- **Acceptance Testing:** Ensure the system meets the business requirements and is ready for deployment.

### 2.3 Bug Triage

Bugs will be classified by severity:

- **Critical:** Blocks core functionality (e.g., order placement fails).
- **High:** Significantly impacts functionality but has workarounds (e.g., reservation not syncing).
- **Medium:** Minor issues (e.g., UI misalignment).
- **Low:** Cosmetic or non-functional issues (e.g., text formatting).

### 2.4 Suspension Criteria and Resumption Requirements

Testing will be suspended if:

- A critical bug blocks major functionality (e.g., order placement or billing stops working).
- Development environments are unstable or non-functional. Testing will resume once blocking issues are resolved, and a stable build is available.

### 2.5 Test Completeness

Testing will be considered complete when:

- All high and critical defects are resolved.
- All test cases are executed and meet acceptance criteria.
- The system successfully passes performance, security, and usability testing.

---

## 3. Test Deliverables

- **Test Strategy Document:** Outlining the overall testing approach.

- **Test Case Document:** Detailed list of test cases for all modules.
- **Test Results:** Logs of all test case executions with pass/fail status.
- **Defect Reports:** A detailed list of defects, including their severity and status.
- **Performance Reports:** Results of load and stress testing for key modules.
- **Final Test Report:** A summary of the overall testing process, including major findings and final recommendations.

---

## 4. Resource & Environment Needs

### 4.1 Testing Tools

- **Manual Testing**: Used for UI/UX and functionality verification.
- **Browser Developer Tools**: For inspecting and debugging the application in Chrome.

### 4.2 Test Environment

- **IDE**: Visual Studio Code (VS Code) for development.
- **Local Server**: Run the project using Live Server extension in VS Code.
- **Browser**: Google Chrome to test the UI and functionality.
- **Database**: MySQL for testing SQL queries.

---

## 5. Test Deliverables

- **Test Cases:** Cover all core functionalities such as order placement, kitchen coordination, reservations, and billing.
- **Test Logs:** Track test execution status and capture pass/fail results.
- **Defect Reports:** Document all bugs encountered with steps to reproduce and severity levels.
- **Performance Metrics:** Load, stress, and response time analysis for high-traffic scenarios.
- **Final Test Report:** Comprehensive report summarising all test phases, major findings, and readiness for deployment.