Test Plan for Rub BBQ RMS

Version 1.1
February 16, 2011

Prepared for:
Jared Leonard, Owner
RUB BBQ
2407 W Lunt Ave.
Chicago, IL, 60645

Prepared by:
Jerod Hodgkin
Mike Schenk
Jessie Floyd
Scott Leonard
Weber State University
Ogden, UT 84408

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Initial Draft 1.0	Jerod Hodgkin Jessie Floyd	This is the initial test plan for the BBQRMS Restaurant Management System.	02/15/2011

	Mike Schenk Scott Leonard		
1.1	Mike Schenk	Added more details to section 4.2	02/16/2011

Contents

1 Introduction

This document will cover the test plan for the BBQ RMS version 1.0. The main features covered by this plan include Inventory Management, Employee Management, Menu Management, Order entry, Cashiering, and Reports.

1.1 Goals

The goals for this test plan will be to insure coverage of all of the main use cases..

1.2 Assumptions

Unit testing will be written using MS Test. The project will be built and unit tests will pass before each code commit.

1.3 Risks and Assets

Assets: Testing will be limited to a single touch screen capable laptop.

Risks: In the event of system fault the tests will be conducted on a laptop system which may not contain the hardware required to enable touch-screen functionality.

1.4 Terms

Define special terms or acronyms used in this document and not explicitly recorded in another document (Test Process Standard, Project Terms and Acronyms Document, etc.) related to the project.

MSR
: Magnetic Swipe Reader. The hardware device that reads magnetically encoded information from a customer's credit or debit card.

POS
: Point of Sale

1.5 References

BBQRMS SSRS 2011.docx - contains the defined use cases to be tested.
BBQRMS Project Plan.mpp - contains the project plan.

2 Features To Be Tested

2.1 Customer Orders

Enter an order.
Save an order.
Recall a saved order for modification.
Take partial payment on a order and save for later payment.
Take payments on an order.
Read payment card information from hardware MSR.
Refund order on cancelled order with partial payment.
Discount an order by percentage.
Discount order by dollar amount.
Send an order to the cook for completion.
Cook’s screen ability to mark orders completed.
Credit card integration sends correctly formatted information to the API.
System receives payment authorization from credit card processor.
Cancel a customer order.

2.2 Inventory

Enter a new inventory supplier.
Change existing supplier details.
Delete supplier.
Enter new inventory items.
Receive inventory (quantity updates properly.)
Record used inventory (quantity updates properly.)
Delete inventory item.
Quantities update properly on inventory reconciliation.

2.3 Employee Administration

Create a new system user (employee.)
Change employee details from admin screen.
Change employee PIN from admin screen.
Employee can change their own pin.
Delete an employee.
Add a role to an employee.
Remove a role from an employee.
Add and remove privileges from a role.
Create a new role.
Delete a role.

2.4 Menu Management

Add new menu items to the database.
Build a menu from menu items already in the database.
Delete menus from the database.

Delete menu items from the database.

2.5 Reporting

Reports calculate correctly.
Reports generate accurate information.
Inactivated data will show up in reports where required.

2.6 Security

Users can log in to the system.
Users see interface items that appropriately relate to their roles.
Manager approval is required before applying discount.
Deleted (inactivated) users cannot log in.

2.7 Performance

Taking customer orders must occur within the time frames specified in the respective use case specifications.

3 Features Not To Be Tested

POS Hardware integration
- Receipt Printer
- Cash Drawer
- Card Reader Firmware/PC Interaction
- Touch Screen interaction.**

** Limited testing scenarios.

4 Approach

All of the features are documented in Jira. Once a developer marks a feature resolved another developer on the team will then act as a tester. The testing will involve black box testing as well as white box testing depending on the need of the tester. When issues are identified the tester will document them in Jira and assign them to the appropriate developer.

Our main testing effort will be focused on development testing over the first three months of development. System testing and UI Testing will begin after the project has gone into beta and continue for two months.

4.1 Test Objectives

Perform an initial assessment of the advertised functionality. Validating internal processes are sound and the fundamental or core aspects of the application are “workable.” Validate each code unit through unit tests, validate each module through component tests, and validate the completed application through system testing.

4.2 Types of Testing

Unit testing for each unit will be conducted by the developer of that unit at the time of development. Primarily, these will be automated unit tests implemented using the MSTest framework.

Component testing will be conducted by a different team member from the one who developed it. To perform component testing, the tester will exercise the functionality through the application user interface by running the client and server application under the debugger using the Visual Studio development environment. Component tests will be developed based on the list of features to be tested.

System testing will be done by all of the team members. Each major module will be independently tested by at least two team members. System test scenarios will be based on the use cases as documented in the SSRS document.

Performance testing will be done on the order entry screen using the integrated profiler provided by Visual Studio.

User acceptance testing will be coordinated with the client by the client liaison. The user’s availability is limited, therefore the client liaison may perform acceptance testing himself.

5 Artifacts

JIRA - bugs found will be managed and tracked using the JIRA application server.

Microsoft Project - The project manager will update the project document as needed, tracking each users progress and overall project status and dependencies.

Daily Journal - Each team member will update weekly their status and accomplishments.

5.1 Test Design Specification (TDS)

User Interface testing: Each button, entry field, report, and interaction with the user will be

examined for “fit for use”, the tester will then outline what aspect is correct and create a report for items which fail to function properly.

Unit testing: Where appropriate each method, property and class will be tested using a fundamental signature check to ensure each method call , property change and class creation will function as expected and any un-handled exceptions or errors are trapped and managed.

Application Server: Validation from the client component will successfully retrieve data from the server without error ensuring the entire application will perform as needed by the customer.

Application Client: The user interface will be able to retrieve needed services and can inter-operate with the server component.

5.2 Test Case Specification (TCS)

	Test Type	Unit Testing	Manual Interaction	Smoke Test	Performance
Test	---	---	---	---	---
User Interface	---		x	x	x
Server	---	x		x	
Client	---	x	x	x	x

5.3 Validation Checklist

Project builds successfully.
Development use cases for the current cycle are completed.
Unit tests run successfully (All tests pass.)
All GUI elements are functional per GUI smoke test.

5.4 Test Reports

Issue Summary Report - A final report of the open issues included in the current iteration. Will include total number of open issues w/severity, % issues resolved during the course of the iteration.

Unit test Summary Report - A final report of units tested. Names of unit tests to verify completeness. All tests must pass per validation checklist, therefore a list of tests passed is not

required.

Test Summary Report - A final report of the testing results from the project. Will include total number of test cases, number of test cases executed, percent test cases passed.

6 Elements Of Test

6.1 Staffing

The staff of the project includes 4 developers. Each of these developers will also act as quality assurance personnel. Each developer also has an individual role on the project, such as project management, technical lead, client liaison, etc.

6.2 Roles & Responsibilities

Each of the developers are assigned to build separate modules of the application. Once these modules are completed another developer acting as QA will test the module and log issues against it. The original developer will then address the issues and mark them resolved. The QA that created the issue will then have the responsibility to verify that the issue is indeed resolved.

When disagreements arise concerning desired functionality the Client Liaison will take the issue to the customer and solicit a decision. Once the decision has been made the Client Liaison will communicate it to the team.

6.3 Schedule

Development will begin on January 4, 2011.
The Test Plan will be defined by February 16, 2011.
System testing begins on February 17, 2011.
Testing Results will be documented by February 23, 2011.
System Testing Phase 2 begins March 1, 2011.
Beta testing begins on March 15, 2011.
Beta Test results reported by March 23, 2011.
System Testing Phase 3 begins March 24, 2011.
Burn in begins April 5, 2011.
Final delivery to instructor April 3, 2011.
Project Go Live April 18, 2011.

6.4 Resources

Developer’s personal Laptops.
1 small tablet PC.

1 USB MSR card reader.
code.google.com for source repository.
Jira for issue tracking.