Table of Contents

[**Test Design** 1](#_Toc477224969)

[**Owners and List of Contacts** 1](#_Toc477224970)

[**Revision History** 1](#_Toc477224972)

[1. Introduction 3](#_Toc477224973)

[1.1 Background 3](#_Toc477224974)

[1.2 References 3](#_Toc477224975)

[1.3 Code Freeze Date 3](#_Toc477224976)

[1.4 Change Control 3](#_Toc477224977)

[2. Items to Be Tested 3](#_Toc477224978)

[2.1 Level of Testing 3](#_Toc477224979)

[2.2 Features to be Tested 6](#_Toc477224980)

[2.3 Test Case Matrix 7](#_Toc477224981)

[2.4 Features Excluded from Testing 7](#_Toc477224982)

[3. Testing Approach 8](#_Toc477224983)

[3.1 Test Deliverables 8](#_Toc477224984)

[3.2 Defect Tracker Setup 10](#_Toc477224985)

[4. Release Criteria 10](#_Toc477224986)

[4.1 Test Case Pass/Fail Criteria 10](#_Toc477224987)

[4.2 Suspension Criteria for failed Smoke Test 11](#_Toc477224988)

[4.3 Resumption Requirements 11](#_Toc477224989)

[4.4 Release to User Acceptance Test Criteria 11](#_Toc477224990)

[4.5 Release to Production Criteria 11](#_Toc477224991)

## 1. Introduction

The primary goal of this document is to establish a plan for the activities that will verify [customer Support tool] as a high quality product that meets the needs of the [Customer Support Tool] business community. These activities will focus upon identifying the following:

* Items to be tested
* Testing approach
* Roles and responsibilities
* Release criteria
* Hardware
* Risks and contingencies

### 1.1 Background

The customer Support tool is a sophisticated user friendly tool that allow user to significantlty

Increase the quality of the M-banking. As a enhancement request new menu is added to the tool

### 1.2 References

List all reference material you used in creating this plan.  
  
**Example:**

1. Functional Specification, Program Management, xxx 1999
2. Testing Computer Software, Second Edition, Kaner / Falk / Nguyen, 1993
3. Detailed Design, Program Management, xxx 1999

### 1.3 Code Freeze Date

Production Code for [Customer support Tool] will be frozen on MM/DD/YY. Our assumption is that any production code changes made after that date is outside of the responsibility of this development project.

### 1.4 Change Control

After baseline, all changes must be approved and documented by the change control board. If it is agreed that the change is necessary, the impact to development and testing must be agreed upon by the test lead, development lead and project manager. This may (or may not) affect the planned completion date of the project.

## 2. Items to Be Tested

### 2.1 Level of Testing

Below is a list of services that testing may provide. Next to each service is the degree of testing that we will perform. Below are the valid level desired:  
  
**High** – High risk area, test this area very hard

**Medium** – Standard testing

**Low** – Low risk area, test if time allows

**None** – No testing desired

|  |  |
| --- | --- |
| **Level Desired** | **Service** |
|  | **Performance Testing:** Performance testing is testing to ensure that the application responds in the time limit set by the user. If this is needed, the client must supply the benchmarks to measure by and we must have a hardware environment that mirrors production. |
|  | **Windows / Internet GUI Standards:** This testing is used to ensure that the application has a standardized look and feel. It may be as simple as ensuring that accelerator keys work properly and font type and size are consistent or it could be as exhaustive as ensuring that the application could be assigned a Windows logo if submitted for one (there are strict guidelines for this). Note: If this level of testing is needed, the client must provide their standards as to allow us to compare to that standard. There is a good book that explains Microsoft standards: The Interface Guidelines for Software Design by Microsoft Press. |
|  | **Platform Testing:** Platform testing is used to warrant that the application will run on multiple platforms (Win 95/98, Win NT, IE 4.0, IE 5.0, Netscape, etc.) (***Specify which ones)*** |
|  | **Localization:**  Localization testing is done to guarantee that the application will work properly in different languages (i.e. Win 95/98 English, German, Spanish, etc.) This also involves ensuring that dates will work in dd/mm/yy format for the UK. (***Specify which ones)*** |
|  | **Stress Testing:** Stress testing is testing to ensure that the application will respond appropriately with many users and activities happening simultaneously. If this is needed, the number of users must be agreed upon beforehand and the hardware environment for system test must mirror production. |
|  | **Conversion:** Conversion testing is used to test any data that must be converted to ensure the application will work properly. This could be conversion from a legacy system or changes needed for the new schema. |
|  | **Parallel Testing:**  Parallel testing is used to test the functionality of the updated system with the functionality of the existing system. This is sometimes used to ensure that the changes did not corrupt existing functionality. |
|  | **Regression of unchanged functionality:**  If regression must occur for functional areas that are not being changed, specify the functional areas to regress and the level of regression needed (positive only or positive and negative testing). |
|  | **Installation Testing:**  Installation testing is testing the setup routine to ensure that the product can be installed fresh, over an existing copy, and with other products. This will test different versions of OCX’s and DLL’s. |
|  | **End to End / Interface Testing:** End to End testing is testing all inputs (super-systems) and outputs (sub-systems) along with the application. A controlled set of transactions is used and the test data is published prior to the test along with the expected results. This testing ensures that the application will interact properly with the other systems. |
|  | **Usability:** Usability is testing to ensure that the application is easy to work with, limits keystrokes, and is easy to understand. The best way to perform this testing is to bring in experienced, medium and novice users and solicit their input on the usability of the application. |
|  | **User’s Guide / Training Guides:** This testing is done to ensure that the user, help and training guides are accurate and easy to use. |
|  | **Input and Boundardy Tests:**  Testing designed toguarantee that the system would only allow valid input. This includes testing to ensure that the maximum number of characters for a field may not be exceeded, boundary conditions such as valid ranges and “off-by-one”, “null”, “max”, “min”, tab order from field to field on the screen, etc. |

### 

### 2.2 Features to be Tested

Below is a list of features that will be tested. ***Note: If features are being dropped into production at different times, add a couple of columns to this grid that identifies what features will be in each drop. The items below map back to the Functional Specifications.***

|  |  |  |  |
| --- | --- | --- | --- |
| **Business Requirements** | **Ref. No.** | **Feature** | **Functional Specification** |
| To Enhance system limit menu in customer support tool | 5.2.1 | System menu should be added to customer support tool | New menu should be added to customer support tool |
|  | 5.2.2 | Adding new rule to the customer | Adding new rule and check to the customer |
|  | 5.2.2.1 | Adding rule based on various payer and payee combination | Adding rule to customer Id based on Payee and payer category set as 0 0r 1 |
|  | 5.2.2.2 | Adding and deleting the rule for the customer | Adding the rule and deleting the rule for the customer. |
|  | 5.2.2.3 | Changing the rule for the customer | Change the rule for the customer |
|  | 5.2.2.4 | Add new customer case limit | Add new customer use case limit |
|  | 5.2.2.5 | Edit or Remove customer use as case | Edit or Remove the use case |
|  | 5.2.2.6 | Add new payment limit | Add new payment limit to the payment instrument selected from the dropdown box. |
|  | 5.2.2.7 | Edit or Delete the payment instrument limit | Allow user to either delete or edit the payment limit . |
|  | 5.2.2.8 | Add new wallet limit | Add new wallet limit for the wallet payment instrument |
|  | 5.2.2.9 | Edit the wallet limit | Edit the wallet limit selected wallet payment instrument |
|  | 5.2.2.10 | Delete the wallet limit | Delete the wallet limit for the wallet payment instrument. |
|  | 5.2.2.11 | Validation for less number of transaction limit | Create customer and set up a transaction for 2 |
|  | 5.2.2.12 | Validation for maximum sum of amount. | Create a customer and validate for maximum sum amount of EUR and $ |
|  | 5.2.2.13 | Validate limit for all use case | Create a customer and set use case limit for all use cases and perform transaction for the limits |
|  | 5.2.2.14 | Validate limit for single use case | Create a customer and set use case limit for single luse cases and perform transaction for the limits |
|  | 5.2.2.15 | Validate payment limit with use case | Create a customer and set payment limit with various use case limit and perform transaction for the limits |
|  | 5.2.2.16 | Validate payment limit with single use case | Create a customer and set payment limit with single use case limit and perform transaction for the limits |
|  | 5.2.2.17 | Validate payment limit with all use case | Create a customer and set payment limit with all use case limit and perform transaction for the limits |
|  | 5.2.2.18 | Validate Wallet payment limit with all use case | Create a customer and set wallet payment limit with all use case limit and perform transaction for the limits |
|  | 5.2.2.19 | Validate wallet payment limit with single use case | Create a customer and set wallet payment limit with all use case limit and perform transaction for the limits |
|  | 5.2.2.20 | Validate wallet payment limit with various use case limit | Create a customer and set wallet payment limit with various use case limit and perform transaction for the limits |
|  | 5.2.2.21 | Validate all the field level value | Validation on all field value |

### 2.3 Test Case Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case Id** | **Func Spec Id** | **Test Case Description** | **Expected Result** |
| Group: inserting new menu and Transaction rule |  |  |  |
| 101 | 5.2.1 | Add the new menu in customer support Tool | System should add new system limit menu |
| 102 | 5.2.2.1 to 5.2.2.3 | Adding /deleting/edit rule to customer | System should allow the user to perform all add /delete and edit operation . |
| Group: Adding payment limit , wallet limit , limit to use case  And editing the same. |  |  |  |
| 103 | 5.2.2.4 to 5.2.2.5 | Perform add/delete and edit operation on use case | System should allow to edit and delete the use case |
| 104 | 5.2.2.6 to 5.2.2.7 | Edit or Delete the payment instrument | System should allow to edit and delete the payment limit |
| 105 | 5.2.2.8 to 5.2.2.10 | To add edit /Delete and change the wallet payment limit | System should allow to perform add/delete and edit wallet payment limit . |
|  |  |  |  |
| Group: Validation on use case limit and Field level value |  |  |  |
| 106 | 5.2.2.11 to 5.2.2.20 | To validate various use case with payment and wallet payment limit | System should restrict the transaction once the limit is reached |
| 107 | 5.2.2.21 | To validation on field value | System should show error message if the limit reached |
|  |  |  |  |
| 109 | 5.2.2.22 | To validate DB level data validation | System should update the data correctly . |

### 2.4 Features Excluded from Testing

Below is a list of features that will NOT be tested.

|  |  |
| --- | --- |
| **Description of Excluded Item** | **Reason Excluded** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 3. Testing Approach

The system test team will begin designing their detailed test plans and test cases, as the development team is designing and coding. Defect Tracker will be used to enter the test cases and to track the defects. It can be accessed from <http://www.DefectTracker.com>.

The builds will be delivered to system test via Visual Source Safe (see cover page for VSS location) drops coordinated by the development team. The development team will be responsible for installing the partial new builds into the existing structure of the system test environment, and updating the client machines if necessary. Build notes with all changes since the last drop and a list of all files to be delivered will accompany each build drop.

Once the build is dropped by the development team, a series of scripts, called the Smoke Test, will be run to ensure that the shipment from development is in a state that is ready for testing. The Smoke Test scripts will test the basic functionality of the system. These scripts may be automated once they are successfully performed manually. If an excessive number of Smoke Test items fail, the product will be shipped back to development and no testing will begin until the Smoke Test passes.

Once the first drop begins, triage meetings will be held to discuss the bug list with the Project/Development Manager. Triage meetings are used to prioritize, set priority and severity and assign bugs. Each week following the first drop, additional drops will be delivered to system test to test the bugs fixed from the prior drops.

Defect Tracker will be used to track, report and analyze bugs. Prior to triage, a Defect Tracker protocol document will be distributed to the project and development manager to ensure that everyone understands how to use Defect Tracker and how to effectively enter bugs.

### 3.1 Test Deliverables

*Below are the deliverables for each phase of the project.*

|  |  |  |
| --- | --- | --- |
| ***Phase*** | ***Deliverable*** | ***Responsible*** |
| Pre-baseline | ***Project Initiation.*** Upon receipt of a functional specification, project initiation will be performed. This includes finding a test lead for the project and setting up a project in Defect Tracker. | Test Lead |
| Pre-baseline | ***Functional Requirement Scrubbing.*** Attend meetings to create functional specifications. Offer suggestions if anything is not testable or poorly designed. | Test Lead |
| Pre-baseline | ***Create Pre-baseline documents  Test Plan.*** The Test Plan will break functionality into logical areas (most often specified in the functional specification). Once completed, the Project Manager, Development Manager, User Project Manager, and Production Support Manager will review it. Once reviewed and amended, it must be approved and signed by the Test Lead, Project Manager, Development Manager, Production Support Manager, and User Project Manager.  ***Create Testing Project Plan and*** . The project plan will be detailed, relating back to the functional specification and Test Plan. | Test Lead |
| Post-baseline | ***Create Test Cases.*** Once the Detailed Test Plan has been created and reviewed by the test and development teams, test cases are created. Test Cases are stored in Defect Tracker. Each test case includes the steps necessary to perform the test, expected results and contains (or refers to) any data needed to perform the test and to verify it works. | Tester, guided by the Test Lead |
| Post-baseline | ***Project and Test Plan Traceability.*** Review the Test Plan to ensure all points of the Functional Specification are accounted for. Likewise, ensure that the Test Cases have traceability with the Test Plan and Functional Spec. Finally, that the Project Plan has traceability with the Test Plan. | Test Lead |
| Once testing begins | ***Triage.*** Once testing begins, triage meetings will be held to prioritize and assign bugs. This is conducted by the Test Lead and will include the Project Manager and Development Lead. Once user testing begins, the User Project Manager will also attend. Triage meetings are usually held 2 to 5 times per week, depending on the need. | Test Lead |
| Bi-Weekly | ***Update Project Plan and Budgeting.*** Update the project plan with % complete for each task and enter notes regarding critical issues that arise. Also determine if the test effort is on budget. | Test Lead |

### 3.2 Defect Tracker Setup

The Test Lead will create a project for Defect Tracker so that bugs can be tracked. The project name in Defect Tracker will be [Bugzilla].

## 4. Release Criteria

### 4.1 Test Case Pass/Fail Criteria

The feature will pass or fail depending upon the results of testing actions. If the actual output from an action is equal to the expected output specified by a test case, then the action passes. Should any action within a test case fail, the entire feature or sub-feature fails. The specific criteria for test case failure will be documented in Defect Tracker.

If a test case fails, it is not assumed that the code is defective. A failure can only be interpreted as a difference between expected results, which is derived from project documentation, and actual results. There is always the possibility that expected results can be in error because of misinterpretation, incomplete, or inaccurate project documentation.

Pass criteria:

1. All processes will execute with no unexpected errors
2. All processes will finish update/execution in an acceptable amount of time based on benchmarks provided by the business analysts and documented by the development team

### 4.2 Suspension Criteria for failed Smoke Test

The system test team may suspend partial or full-testing activities on a given build if any of the following occurs:

* Files are missing from the new build.
* The development team cannot install the new build or a component.
* The development team cannot configure the build or a component.
* There is a fault with a feature that prevents its testing.
* Item does not contain the specified change(s).
* An excessive amount of bugs that should have been caught during the component/unit test phase are found during more advanced phases of testing.
* A severe problem has occurred that does not allow testing to continue.
* Development has not corrected the problem(s) that previously suspended testing.
* A new version of the software is available to test.

### 4.3 Resumption Requirements

The steps necessary to resume testing:

* Clean previous code from machines.
* Re-install the item.
* The problem encountered resulting in suspension is corrected.

Resumption of testing will begin when the following is delivered to the system test team:

* A new build via Visual Source Safe.
* A list of all bugs fixed in the new version.
* A list of all the changes to the modules in the new version and what functionality they affect.

### 4.4 Release to User Acceptance Test Criteria

The release criteria necessary to allow the code to migrate to User Acceptance Testing are as follows:

* There are no open bugs with a severity 1 or 2
* Test cases scheduled for both Integration and system test phases have passed.
* Successfully passes the final regression testing.
* There are no discrepancies between the master setup and the version used during the final regression testing.

### 4.5 Release to Production Criteria

The release criterion necessary to allow the code to migrate to Production is as follows:

* There are no open bugs with a severity 1 or 2
* Test cases scheduled for both Integration and system test phases have passed.
* Successfully passes the final regression testing.
* There are no discrepancies between the master setup and the version used during the final regression testing.
* The User Acceptance Test was successfully completed
* The User Acceptance Criteria was met.