Team A

Test Plan

CIS-470 – Senior Project



Table of Contents

[Scope of Testing 2](#_Toc448858292)

[Unit Testing 2](#_Toc448858293)

[Unit Hierarchy 3](#_Toc448858294)

[Unit Test Policy 4](#_Toc448858295)

[Unit Testing Procedures and Responsibilities 4](#_Toc448858296)

[Integration Testing 6](#_Toc448858297)

[Integration Testing Policy 6](#_Toc448858298)

[Integration Strategy 7](#_Toc448858299)

[Integration Test Procedures 7](#_Toc448858300)

[Integration Testing Results 9](#_Toc448858301)

**Test Plan**

|  |  |
| --- | --- |
| Team: | Team A |
| Team Members: | John Boley, Justin Byrne, James Coltman, and Marshall Gibson |
| Date: | 04/10/2016 |
| Project Title: | Williams Specialty Company - - Business Automation |

**Williams Specialty Company - - Business Automation Test Plan**

**Scope of Testing**

Each process that has been implemented into this Automation System, originating from its (original) source requirements, will be tested (exclusively), as well as any other additional functions integrated to ensure that these required processes operate effectively; from a user’s perspective. Each unit test will be designed to predict, identify, and rectify any errors associated with their functionality (or performance) as well as how well they catch any foreseeable (and elementary) user errors.

**Unit Testing**

There will be two types of unit tests conducted: A PHP/JSON generated test for functionality will be a coded, automated, test written in the PHP project and will require no direct human testing outside of executing the test module. Automated tests, which are coded to cover only the functionality of the components, allow an extra layer of testing free of human error or interaction. The results of these tests will be included with our project documentation at the conclusion of this project.

A battery of in-depth, user experience tests in which a tester will manually navigate through the graphical user interface will also be conducted. User

experience tests cover functionality and an examination of the look and feel of the application as well as the navigation/flow.

**Unit Hierarchy**

* **Login Panel**: *where user credentials will be passed for user authentication*
  + Special permissions should be included for the following employee(s):
    - *Sales Clerk*
    - *Specialist*
    - *Stockroom Clerk*
    - *Operations Manager*
* **Control Panel**: *where user(s) can review, manage, and alter various internal office jobs, records, and notifications*
  + Get Existing Work-Order(s)
  + Save (or Establish) New Work-Order(s)
  + Validate New/Existing Work-Order(s)
  + Mark Existing Work-Order’s as Complete
* **Inventory Search Panel**: *where user(s) can search for inventory items by ID, name of the item, or the item’s manufacturer name*
* **Notification Panel**: *where user(s) can generate/obtain interoffice notifications (or memos) with other inter-office employees associated with a particular work-order, which should include:*
  + New Order Notification(s)
  + Validation Status Notification(s)
  + Inventory Request Notification(s)
  + Inventory Request Fulfilled Notification(s)
  + Inventory Request Ordered Notification(s)
  + Work Complete Notification(s)
  + QA Pass/Fail Notification(s)
  + Ready for Delivery Notification(s)

**Unit Test Policy**

The purpose of the Unit Test Policy is to ensure each unit/component operates properly and is free of bugs and/or defects. These tests focus on not only the functionality of each component of the application, but the ease of use for the user and workflow from beginning to end.

These tests are designed to cover the application workflow from start to finish and ensure that each component/unit functions properly, is easy to navigate and use, and satisfies the criteria outlined in the Statement of Work.

**Unit Testing Procedures and Responsibilities**

The unit test procedures are clearly defined for each case. During testing, any and all tests cases will be divided amongst the developers in the group and executed per the acceptance criteria defined in the Statement of Work.

The test cases should be performed in order, but needn’t be. The workflow is closely followed in the test plan, so proceeding in order would make testing easier on the developer.

Prior to beginning a test the tester should do the following:

* Read the entirety of the test case
* Understand what it is the goal of each test case is
* Verify that the prerequisites are met:
  + Data needed to perform tests exists in the database
  + Ensure the data to be used is understood, particularly in the way of passwords as zeros and o’s, ones and lower case L’s are sometimes difficult to distinguish from one another.
* Familiarize yourself with the Defect ID Key; understand how to classify failures or issues you find.

The plan is formatted as follows:

* Tester – Full name
* ID # - Prepopulated
* Test Condition – Prepopulated
* Procedure – Required Steps to complete test
* Data – Provided information to complete the test
* Expected Result – What should happen when the steps are followed
* Actual Result – What actually happened when steps were followed
* Negative Test – Criteria that would cause a failure
* Pass/Fail – Did the test pass or fail
* Defect ID # - See below

During testing, it is imperative that the tester record all thoughts, both good and bad, in the comments, notes section located at the bottom of the test plan worksheet. Lastly, use the following defect ID key to classify your reasons for failure. The Defect ID should be formatted as follows in the example of a notification being sent to the wrong employee: Initials-DefectID

The properly formatted Defect ID on the test above being performed by a tester named Bob Williams would be: **BW-BN4**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Identifier** | **Test Type** | **Defect Description** | **Defect ID** |
| **A** | **FORMS** | Form Functionality | AF1 |
| Form Aesthetics | AF2 |
| Data Field Functionality | AF3 |
| Control Functionality | AF4 |
| Workflow Issues | AF5 |
| Navigation Issues | AF6 |
| **B** | **NOTIFICATIONS** | Notification Not Generated | BN1 |
| Notification Alert Not Visible | BN2 |
| Notification Not Received | BN3 |
| Note Sent to Wrong Employee | BN4 |
| Notification contents incorrect | BN5 |
| **C** | **PERMISSIONS** | See less than allowed | CP1 |
| See more than allowed | CP2 |
| **D** | **OTHER** | Typos | DO1 |
| Poor Form Construction | DO2 |
| Difficulty with Flow | DO3 |

**Integration Testing**

**Integration Testing Policy**

The purpose of this testing policy is to insure that each piece of the application interacts as designed and that all the functionality is working. This testing will include interactions between all layers of the application as a complete end-to-end test of the functionality.

The development team will be responsible for carrying out this test plan and in certifying that the testing has been complete. The testing must be recorded and include the following:

* Tester
* Test ID
* Test Condition
* Procedure
* Expected Result
* Actual Result
* Negative Test
* Pass or Fail

**Integration Strategy**

Due to the size of the application the Big Bang Integration Strategy will be used. This testing strategy integrates all components together at once and then the tests are run.

The subsequent steps will be followed to perform the Integration Test:

* All testers will have a thorough understanding of the application including how each feature works and how the features are related or linked to each other.
* All possible scenarios are identified.
* Test all scenarios.
* Report any defects to the development team.
* Both positive and negative integration testing will be done.
  + Positive testing will ensure the application works as expected. If an error occurs the test fails.
  + Negative testing will ensure the application can handle unexpected user behavior or invalid input without crashing and returns proper error message.

**Integration Test Procedures**

1. Login Component:
   1. Ensure login using proper credentials
   2. Verify failure while using incorrect credentials
2. Work Order Creation:
   1. Application accepts valid data input and creates work order
   2. Application rejects invalid data
   3. Verify data written to database
3. Input of Customer Data:
   1. Application accepts valid customer data
   2. Application rejects invalid data
   3. Verify customer data written to database
4. Input Billing Data:
   1. Application accepts valid data input
   2. Application rejects invalid data
   3. Verify data written to database
5. Input Shipping Data:
   1. Application accepts valid data input
   2. Application rejects invalid data
   3. Verify data written to database
6. Notification:
   1. Accepts valid data (Order ID) in Order ID text box
   2. Rejects invalid data in Order ID text box
   3. Radio buttons work correctly
   4. Accepts input in Details text box
   5. Verify data saved to database
7. Search Customers:
   1. Drop down list functions correctly
   2. Search text box accepts data
   3. Returns correct customer data
   4. Returns message when no data found
8. Search Employees:
   1. Drop down list functions correctly
   2. Search text box accepts data
   3. Returns correct employee data
   4. Returns message when no data found
9. Search Orders:
   1. Drop down list functions correctly
   2. Search text box accepts data
   3. Returns correct order
   4. Returns message when no data found
10. Search Inventory:
    1. Drop down list functions correctly
11. Search text box accepts data
12. Returns correct inventory item
13. Returns message when no data found

## Integration Testing Results

See attached Excel file labeled **Integration\_Test\_Cases**