**Table of Contents**

1.0 Introduction

2.0 Application Overview

3.0 Testing Scope

3.1 Items Tested

3.2 Items not tested

4.0 Metrics

5.0 Types of testing performed

6.0 [Test Environment & Tools](http://www.softwaretestinghelp.com/test-bed-test-environment-management-best-practices/)

7.0 Lessons Learned

8.0 Recommendations

9.0 Best Practices

10.0 Exit Criteria

11.0 Conclusion/Sign Off

**1.0 Introduction**

This document explains the various activities performed as part of Testing for the Online Bookstore application developed by Alliance..

**2.0 Application Overview**

“Forwarding Mail” application is a GUI utilizing C# and was built in Microsoft Visual Studio 2013. Forwarding mail labels can be printed from the program. Student information is received from a shared file in the Post Office network. The student information is placed on the label featuring five of the following: a regular forwarding label, forwarding time expired with the student address, forwarding time expired with no address, moved left no forwarding, or addressee unknown. Administrators can add, update, and remove student information. Administrators can also make other admins, or update their own information.

**3.0 Testing Scope**

This section explains about the functions/modules in scope & out of scope for testing, as well as any items which are not tested due to any constraints/dependencies/restrictions.

1. Items tested
2. Items not tested

**3.1 Items Tested**

Below are the functional testing Alliance used to determine the functionality of the Online Bookstore website.

* Login
  + Login
  + Registration
  + Logout
* Administrator Functions
  + Login
  + Removing book from database
  + Removing user from database
* Sale process
  + Selecting book
  + Adding book to cart
  + Viewing cart
  + Proceeding to checkout
  + Filling out shipping information
  + Filling credit card information
  + Processing order

**3.2 Items not tested**

Below are the functional tests Alliance has not completed on the Online Bookstore website due to the features not being implemented.

* Administrator Actions
  + Manage comments
  + Manage the announcements
  + L

**4.0 Metrics**

The following section gives a visual representation of the test cases Alliance’s testing team used to detect and fix any possible defects. Table 4.1 shows the amount of closed test cases from the open test cases left.

|  |  |
| --- | --- |
| **Closed Test Cases** | **Open Test Cases** |
| 35 | 0 |

*Test Case Table 4.1*

**5.0 Types of testing performed**

The following is the types of tests performed by Alliance. System testing was conducted until the end of the SDLC.

.

1. **Unit testing** – Testing of individual software components or modules. Completed by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code.
   1. This type of test was utilized because the programmers needed to be sure that the modules were functioning correctly when run.
2. **Integration testing** – Testing of combined software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code.
   1. This type of test was utilized because the programmers needed to be certain that when more than one modules or components were combined that the application would still behave normally.
3. **Usability testing** – User-friendliness check. Application flow is tested. Can a new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.
   1. This type of test was utilized to determine if the application required unnecessarily complicated instructions for its use.
4. **System testing** - Testing conducted on a complete and integrated system to determine if the software complies with the specified requirements. Typically done by a tester who has no knowledge of the code.
   1. This type of test was utilized to evaluate the complete and functioning application. System testing checks to see if all modules and components are operating correctly, and if there are any defects, major or minor, in the application.

**6.0** [**Test Environment & Tools**](http://www.softwaretestinghelp.com/test-bed-test-environment-management-best-practices/)

The following is the developmental environment Alliance was in while developing the Online Bookstore.

* Notepad++;
* Digital Ocean LAMP server running Ubuntu 16.04 (Xerial Xerus)
* Minimum of 128 megabytes of system RAM
* Hard drive with 20 gigabytes of free space
* Internet Connection
* Bootstrap.
* PHP CRUD API

**7.0 Lessons Learned**

This section describe the critical issues Alliance faced and our solutions. Lessons learnt helped to make proactive decisions during testing, by avoiding common mistakes or finding a suitable workaround for more efficient testing.

|  |  |
| --- | --- |
| **Issues Faced** | **Solution** |
| Database Crashes | Table was dropped from database. Sanitize database requests or utilize API. |
| Database disappears | Back up of database was taken once a week. |
| JavaScript ran asynchronously | Utilize .done and .then functions after gets and posts. |
| Items were not properly alligned | Utilize bootstrap to get a design that worked well for the site |

*Table 7.0 Issues Alliance Faced and Overcame*

**8.0 Recommendations**

Web applications require tedious and efficient programming. Utilization of database calls may be more efficient than API calls since API calls request a page from the server. This however brings a trade off with security. Database calls need to be sanitized or a table may dropped or the database may become at risk for damage.

Integration testing was the greatest tactic at our disposal when used because when many JavaScript functions were placed into one page, some of them did not run correctly.

**9.0 Best Practices**

Templating certain areas of the webpages allowed for modification of one file to cascade throughout all webpages minimizing the number of pages that had to be individually updated.

Use of bootstrap allowed the website to dynamically resize to fit the viewport. Therefore, the website worked well on both big screens and mobile devices

**10.0 Exit Criteria**

* All test cases for complete sections should be executed – **Yes**
* All sections should be completed and be functional – **No**.
* All current issues should be noted for next session of SDLC. – **Yes**

**11.0 Conclusion/Sign Off**

As the Exit criteria was not met, Alliance Software Development suggests that this website be worked on further and will turn over all code, documents and relevant information to our professor, Dr. Catherine Stringfellow.