TESTING SUITE

for

Book Shop Automation Software

Version 1.0 approved

Presented by

Rameshwar Bhaskaran (14CS30027)
Aditya Bhagwat (14CS30002)
Group 55

IIT Kharagpur
April 13, 2016

Contents

1 Introduction

1.1 Purpose

This document is a high-level overview defining testing strategy for the Book Automation Software. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. Testing criteria under the white box, black box, and system-testing paradigm will be utilized. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation will be applied.

1.2 Testing overview

- The pattern for testing the system is first identifies i.e the various major functions that may cause bugs.
- The basic philosophy to design the test design suite is to maximise discovery of bugs so that they can be debugged before release.
- First, review the test data and test cases so that they are exhaustive for each unique unit's verification.
- Identify the expected results.
- The expected results are documented
- The tests are performed and the results are compared with the expected results.
- The mismatches are identified and the bug is reported to the development team
- Any specifications/details to be reviewed are reported

2 Testing strategy

The following testing processes are used to test the software:

2.1 Unit testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.For the testing purposes, *JUnit* library is used. The following functions are some of the functions tested.

- addBook() *(BookDao class)*
- getInventoryLevel() *(BookDao class)*
- getStatistics() *(TransactionDao class)*
- getBookByISBN() *(BookDao class)*
- viewRequests() *(NotInCollection class)*
- addRequest() *(NotInCollection class)*
- searchVendorById() *VendorDao class)*

The way we have implemented unit testing is in a **white box** testing fashion as we are testing directly within the application at the source code level.

2.2 Black box testing

Black box testing is the type of testing where the entire application's internal working(and source code) is abstracted and all possible inputs are tested on.
In this testing phase, every major use case is given both right inputs and erroneous inputs. On giving the right inputs, the program should give the expected result and on giving erroneous inputs, the program should give an error message.
The below are some of the major features tested.

- Generate receipt
  - Case when no such book exists

    - Case when successful generation happens

- Fill request for not in collection book
- View statistics
    - Date defined correctly
    - Date defined incorrectly
- View books below threshold
- Update database
    - When no such book exists
        * Publisher not registered
        * Vendor not registered
    - When book exists
- Query for book
    - Book exists
    - Book exists but not in stock
    - No such book exists in database

Here, the application is tested on the interface level and appropriate screenshots are given wherever required.

2.3 White box testing

White box testing is the testing at the application's source code level. Major features and control flows are tested in this process. Unit testing falls under this category. Apart from that, major control deciding elements like buttons were tested whether the right action was taken by writing print statements in the listeners.

2.4 Integration testing

The primary objective is to test whether different modules of the software(which have been tested before through white and black box testing) work together in tandem when integrated.The two main modules of the software are the DAO and GUI modules. A submodule is the PDF generation and viewing module. But as both use functions from two external libraries which have already been tried and tested by a huge user community, the tests have been avoided for that.

2.4.1 GUI Module

- Each button and component is checked for correct behaviour.
- Buttons should either perform a specific task assigned(like generate receipt button) or help navigate to different panels(Logout,Back).
- Checks need to be done whether textfields with compulsory values accept null values
- Situations where the user is ”stuck” in a particular panel i.e the user cannot go to another panel as no navigational component is present are enlisted

2.4.2 DAO module

- Check whether the data received from database is correctly populated on the interface
- Check conditions when either no data is received or too much of data is received

2.5 System Testing

The main goal of system testing is to validate the software as a whole i.e its performance aspects, security, portability and configuration aspects.

The performance aspects and the function validation are done as part of this test plan. For function validation, the behaviour of functions is cross checked with the expected outputs of the corresponding use cases in the SRS document.

As expected, the above system tests have been implemented in a black box testing fashion.