****

**Lecturer: Nguyen Tran Thi Van**

**Members: Nguyen Van Dan 17110010**

**Doan Viet Thuan 17110080**

**Nguyen Tan Phat 17110059**

**REPORT**

1. **COMMERCE WEBSITE FOR**

**SELLING BOOK**

**SOFTWARE TESTING**

**Ho Chi Minh, 12/2020**

**HO CHI MINH UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC KINH BẮC**

**Score**

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Content | Presentation | Overall |
| Score |  |  |  |

Observations of Instructor

Evaluation:

Nguyen Van Dan:   
Doan Viet Thuan:   
Nguyen Tan Phat:

Lecturer

Nguyen Tran Thi Van

Contents

[Chapter 01 Introduction 8](#_Toc62051941)

[1.1 General description of the project: 8](#_Toc62051942)

[1.2 Plan for Final Project 8](#_Toc62051943)

[Chapter 02 System specification, requirement modeling, use-case diagrams, use-case specifications. 11](#_Toc62051944)

[2.1 System specification 11](#_Toc62051945)

[2.1.1 Website connstruction goals 11](#_Toc62051946)

[2.1.2 Characteristics 11](#_Toc62051947)

[2.2 Requirement modeling 12](#_Toc62051948)

[2.2.1 Module product 12](#_Toc62051949)

[2.2.2 Shopping cart module 12](#_Toc62051950)

[2.2.3 Module member registration and system login 12](#_Toc62051951)

[2.2.4 Module search for products, news, advice 12](#_Toc62051952)

[2.2.5 Module to manage products and orders 13](#_Toc62051953)

[2.3 Usecase diagrams 13](#_Toc62051954)

[2.4 Uscase specifications 13](#_Toc62051955)

[2.4.1 Login 14](#_Toc62051956)

[2.4.2 Register an account 15](#_Toc62051957)

[2.4.3 Search product information 17](#_Toc62051958)

[2.4.4 Add new product information 19](#_Toc62051959)

[2.4.5 Delete product information 20](#_Toc62051960)

[2.4.6 Update product information 22](#_Toc62051961)

[Chapter 03 Test plan 24](#_Toc62051962)

[3.1 Introduction 24](#_Toc62051963)

[3.1.1 Purpose 24](#_Toc62051964)

[3.1.2 Project Overview 25](#_Toc62051965)

[3.1.3 Audience 26](#_Toc62051966)

[3.1.4 Definitions, Acronyms, and Abbreviations 27](#_Toc62051967)

[3.1.5 References 27](#_Toc62051968)

[3.2 Test strategy 28](#_Toc62051969)

[3.2.1 Test Objectives 28](#_Toc62051970)

[3.2.2 Test Principles 28](#_Toc62051971)

[3.2.3 Scope 28](#_Toc62051972)

[3.2.4 Functional Testing 29](#_Toc62051973)

[3.2.5 Regression Testing 30](#_Toc62051974)

[3.2.6 Security and Access Control Testing 30](#_Toc62051975)

[3.2.7. MILESTONE LIST 31](#_Toc62051976)

[3.2.8 Test tool 31](#_Toc62051977)

[3.3 Execution strategy 32](#_Toc62051978)

[3.3.1 Entry and Exit Criteria 32](#_Toc62051979)

[3.3.2 Defect tracking & Reporting 32](#_Toc62051980)

[3.4 TEST MANAGEMENT PROCESS 33](#_Toc62051981)

[3.4.1.Role Expectations 33](#_Toc62051982)

[3.5. TEST ENVIRONMENT 33](#_Toc62051983)

[3.5.1. System 33](#_Toc62051984)

[Chapter 04 Describe in details about the testing methods and techniques 34](#_Toc62051985)

[4.1 Unit Testing 34](#_Toc62051986)

[4.1.1 What is Unit Testing? 34](#_Toc62051987)

[4.1.2 Why Unit Testing? 34](#_Toc62051988)

[4.1.3 How to do Unit Testing 34](#_Toc62051989)

[4.1.4 Unit Testing Techniques 35](#_Toc62051990)

[4.1.5 Unit Testing Example: Mock Objects 35](#_Toc62051991)

[4.1.6 Unit Testing Tools 35](#_Toc62051992)

[4.1.7 Unit Testing Advantage 36](#_Toc62051993)

[4.1.8 Unit Testing Disadvantages 36](#_Toc62051994)

[4.2 Integration testing 36](#_Toc62051995)

[4.2.1 What is Integration Testing? 36](#_Toc62051996)

[4.2.2 Why do Integration Testing? 36](#_Toc62051997)

[4.2.3 Example of Integration Test Case 37](#_Toc62051998)

[4.2.4 Approaches, Strategies, Methodologies of Integration Testing 37](#_Toc62051999)

[4.2.5 How to do Integration Testing? 38](#_Toc62052000)

[4.2.6 Entry and Exit Criteria of Integration Testing 38](#_Toc62052001)

[4.3 Sandwich Testing 39](#_Toc62052002)

[4.3.1 What is System Testing? 39](#_Toc62052003)

[4.3.2 What do you verify in System Testing? 39](#_Toc62052004)

[4.3.3 Different Types of System Testing 40](#_Toc62052005)

[4.3.4 What Types of System Testing Should Testers Use? 40](#_Toc62052006)

[4.4 Acceptance Testing 41](#_Toc62052007)

[4.4.1 What is Acceptance Testing? 41](#_Toc62052008)

[4.4.2 Why Acceptance Tests? 41](#_Toc62052009)

[4.4.3 Types 42](#_Toc62052010)

[4.4.4 Qualities of Acceptance Testers 43](#_Toc62052011)

[4.4.5 Impact of Issues found during this testing 44](#_Toc62052012)

[Chapter 05 Choose at least 3 code units / functions / APIs / methods, etc. to perform whitebox testing 45](#_Toc62052013)

[5.1 Authorize Admin 45](#_Toc62052014)

[5.1.1. Create Control flow graph 45](#_Toc62052015)

[5.1.2 Calculate Cyclomatic complexity to identify independent linear execution paths. 46](#_Toc62052016)

[5.1.3 Design test cases for each path 46](#_Toc62052017)

[5.1.4 Create the general data flow graph 46](#_Toc62052018)

[5.2 Update user 47](#_Toc62052019)

[5.2.1 Create Control flow graph 47](#_Toc62052020)

[5.2.2 Calculate Cyclomatic complexity to identify independent linear execution paths. 48](#_Toc62052021)

[5.2.3 Design test cases for each path 48](#_Toc62052022)

[5.2.4 Create the general data flow graph 48](#_Toc62052023)

[5.3 Delete User 49](#_Toc62052024)

[5.3.1 Create Control flow graph 49](#_Toc62052025)

[5.3.2 Calculate Cyclomatic complexity to identify independent linear execution paths. 49](#_Toc62052026)

[5.3.3 Design test cases for each path 49](#_Toc62052027)

[5.3.4 Create the general data flow graph 50](#_Toc62052028)

[Chapter 06 Bug Report 51](#_Toc62052029)

[6.1 Bug Report Template: B1 (G1-31) 51](#_Toc62052030)

[6.2 Bug Report Template: B2 (G1-22) 52](#_Toc62052031)

[6.3 Bug Report Template: B3 (G1-38) 53](#_Toc62052032)

[6.4 Bug Report Template: B4 (G1-03) 54](#_Toc62052033)

[6.5 Bug Report Template: B5 (G1-19) 55](#_Toc62052034)

[6.6 Bug Report Template: B6 (G1-15) 56](#_Toc62052035)

[6.7 Bug Report Template: B7(G1- 05) 57](#_Toc62052036)

[Chapter 07 Bug Fix 59](#_Toc62052037)

[7.1 Test case ID (G1-31) 59](#_Toc62052038)

[7.2 Test case ID (G1-22) 59](#_Toc62052039)

[7.3 Test case ID (G1-38) 59](#_Toc62052040)

[7.4 Test case ID (G1-03) 60](#_Toc62052041)

[7.5 Test case ID (G1-19) 60](#_Toc62052042)

[7.6 Test case ID (G1-15) 61](#_Toc62052043)

[7.7 Test case ID (G1-05) 61](#_Toc62052044)

# Chapter 01 Introduction

## 1.1 General description of the project:

Building an online book selling website provides an alternative means of e-commerce for selling school books and vulnerable books for the students and remaining people. The website also comes with a great benefit which is that it also provides an inquiry of books and also the uploading and purchasing a book with a single click on the button. It also provides the user to enquire about the books through message. The objective is to develop a fully functional online bookselling website to allow students to sell and buy books and obtain the full utilization of the books. To further make the system complete an inquiry system is put into place to determine if a buyer or uploader wants to.

## 1.2 Plan for Final Project

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Week | Duration | Tasks | Student in charge | Expected results / products | Date of completion | Completion Rate |
| 1 | 14/12-20/12 | 1) Specification and modeling requirements.  2) Create the usecase diagrams and write the usecase specification for the system.  3) Create the Test Plan for the whole project. | - Doan Viet Thuan (1)  - Nguyen Van Dan (2)  - Nguyen Tan Phat (3) | - Complete specification and modeling requirements. (1)  - Make usecase diagrams and detail descriptions of the system's functions. (2).  - Complete creation of the detailed test plan for the entire project.(3) |  |  |
| 2 | 21/12-27/12 | 1) Describe in details about the testing methods and techniques used during the test process of the project.  2) Choose at least 3 code units / functions / APIs / methods, etc. to perform whitebox testing | - Doan Viet Thuan (1)  - Nguyen Van Dan (2)  - Nguyen Tan Phat (2) | - A Word file included all of the testing methods and techniques. (1)  -A Word file include result of whitebox testing.(2) |  |  |
| 3 | 28/12-03/01 | 1) Based on 8 blackbox testing techniques to design at least 50 test cases. | All members | -An Excel file included 50 test cases based on blackbox testing.(1) |  |  |
| 4 | 04/01-10/01 | 1) Write bug reports for the unpassed test cases in the above part, using the the given template. | Divide the number of test cases equally for each member | - Archive 50 blackbox testcases.(1)  - Write the above to the report.(1) |  |  |
| 5 | 11/01-17/01 | 1 ) Write report presentation. | All members | -The report should be presented in the form of a complete written reports, conforming to the regulations given in the attached instructional document.(1)    -The test suite of test cases must be organized in a separate Excel document.(1) |  |  |

# Chapter 02 System specification, requirement modeling, use-case diagrams, use-case specifications.

## 2.1 System specification

### 2.1.1 Website connstruction goals

The object of the website

Website is built to serve two main objects: Admin (administrators value) and Customers with the following functions:

**Admin**

* Website Login
* View, update, delete product information.
* Manage orders
* View, respond to customer comments, suggestions, and feedback or delete such information from customers.
* View, delete information of customers but cannot right to change that information.
* News updates

**Customers**

**Guest**: Visitors

* View product information as well as other news
* Membership registration

**User**:

* Already have an account
* Have the right to login, log out, and change passwords
* Order products
* Enjoy privileges like new product announcements most, download files,etc.

### 2.1.2 Characteristics

Build a simple, user-friendly, easy-to-use online clothing sales system that allows customers to view information and place orders online, and administrators to manage information about products as well as users.

Website designed with

* User-friendly interface, making it easy for users to use.
* The homepage will display a list of the newest and best-selling products that make it easier for the user to find.
* Customers can easily find detailed information about the clothes they are interested in.
* Customers can choose to buy the clothes they need based on their financial ability and required functionality by adding them to the shopping cart
* Have the function of registration, login.

Customers can send feedback and suggestions to the Website to contribute to enriching and developing the Website.

Build a simple, user-friendly, easy-to-use online clothing sales system that allows customers to view information and place orders online, and administrators to manage information about products as well as users.

## 2.2 Requirement modeling

### 2.2.1 Module product

Display information and classify products in virtual booths. Products displayed on the website will be displayed full information about that product such as image, product name, product's features, price, etc.

### 2.2.2 Shopping cart module

When referring to full product information, customers can order products right at the Website through the shopping cart function without having to go to the transaction location, the shopping cart is simulated as a real shopping cart. can add, remove, or pay for purchased products. When choosing to pay for the shopping cart, the customer must fully record personal information, this information is stored and processed by the system.

### 2.2.3 Module member registration and system login

Each customer transacting at the Website will be entitled to register a separate account. This account will be used when requested by the system. An account registered by the customer will store the personal information of the customer.

### 2.2.4 Module search for products, news, advice

Customers will be provided with a search function on the Website.

- News sites: Advise customers who want to learn about the recipe and baking method ...

- Introduction page: Introduce information about the store, sales motto ...

- Contact page: Customers can contact the sales staff with their questions and opinions.

### 2.2.5 Module to manage products and orders

Administrators can update information on items, types of goods, and manage order information.

## 2.3 Usecase diagrams



## 2.4 Uscase specifications

### 2.4.1 Login

|  |  |
| --- | --- |
| Name | Log In |
| Brief Description | A user of the Website login to the Website. |
| Actor(s) | Logged In User (Customer, Admin) |
| Flow of Events | |
| Basic Flow | |
| This use case starts when a website user is not logged in to the website and goes to the login page.  1. The website prompts the user for a username and password or register new account  2. The user enters his/her username and password.  3. The website validates the entered username and password, making sure that the entered username is a valid username in the website, and that the required password is entered for the entered username.  4. The user is signed in and returned to the home page as a Logged In User.  5. The use case ends. | |
| Alternate Flows | |
| Title | Description |
| User Fails Authentication | If the User entered an invalid username and/or password, the following occurs:  1. The website describes the reasons why the User failed authentication.  2. The website presents the User with suggestions for changes necessary to allow the User to pass authentication.  3. The website prompts the User to re-enter the valid information.  4. The Basic Flow continues where the User enters new information (see step 2 of the Basic Flow). |
| Pre-Conditions | |
| Title | Description |
| (none) | User have registered an account on the website |
| Post-Conditions | |
| Title | Description |
| Success | The User is authenticated and the website displays a home page based on the user type. |
| Failure | User is unable to log in for one or more reasons. |
| Extension Points | |
| None | |

### 2.4.2 Register an account

|  |  |
| --- | --- |
| Name | Register an account |
| Brief Description | A user of the website creates an account |
| Actor(s) | Customer |
| Flow of Events | |
| Basic Flow | |
| This use case starts when a website user is not logged in to the system and goes to the login page.  1. The website prompts the user for a username and password or register new account.  2. The user selects registration option.  3. The website prompts user for registration information, Username, password, etc  4. The user enters in their information.  5. Website verifies information and creates account.  6. The use case ends. | |
| Alternate Flows | |
| Title | Description |
| Cancel Registration | 1. The user selects the cancel option.  2. The website returns the user to the home page without the user being logged in and any information entered has been erased. |
| Invalid Information Entered | 1. User clicks submit after entering information system asked for.  2. Website displays information with appropriate message to correct invalid information.  3. User re-enters information. |
| Pre-Conditions | |
| Title | Description |
| (none) |  |
| Post-Conditions | |
| Title | Description |
| Success | The user entered successful information and is returned to the home page as a Logged In User |
| Failure | User is unable to log in for one or more reasons and is returned to the home page as a Guest. |
| Extension Points | |
| None | |

### 2.4.3 Search product information

|  |  |
| --- | --- |
| Name | Search product information |
| Brief Description | An user searches product information |
| Actor(s) | Customer |
| Flow of Events | |
| Basic Flow | |
| This use case starts when a Customer accesses the “Search” feature of the website.  1. The website displays the search submission box  2. The Guest enters search criteria and submits  3. The system displays results  4. Use Case ends. | |
| Alternate Flows | |
| Title | Description |
| Cancel Search | 1. Customer selects the “Cancel” option.  2. Website returns Customer to Home page. |
| Pre-Conditions | |
| Title | Description |
|  |  |
| Post-Conditions | |
| Title | Description |
| Success | The search is successfully executed |
| Failure | The search fails for one or more reasons |
| Extension Points | |
| None | |

### 2.4.4 Add new product information

|  |  |
| --- | --- |
| Name | Add new product information |
| Brief Description | An User of the Website adds a new product to the database |
| Actor(s) | User ( Admin, Customer) |
| Flow of Events | |
| Basic Flow | |
| This use case starts when an user accesses the “Add new product information" feature of the Website.  The Website asks the User to enter the appropriate information for the product to be entered (e.g. Product Name, Publisher, etc...).  User enters appropriate information and clicks submit.  Website validates product information.  User is returned to User panel.  Use Case ends. | |
| Alternate Flows | |
| Title | Description |
| Invalid Information Entered | The Webiste displays an “Invalid Product Information” error after the user submits the information and asks the User to re-enter the information.  User re-enters information and clicks submit. |
| Cancel Add New Product Informaion | User clicks cancel after selecting the “Add new product information ” feature.  Website returns User to the User panel. |
| Pre-Conditions | |
| Title | Description |
|  |  |
| Post-Conditions | |
| Title | Description |
| Success | The new product information is added into the database and is now searchable. |
| Failure | The product information is not added for one or more reasons. |
| Extension Points | |
| None | |

### 2.4.5 Delete product information

|  |  |
| --- | --- |
| Name | Delete product information |
| Brief Description | An User of the Website deletes a product to the database |
| Actor(s) | User ( Admin, Customer) |
| Flow of Events | |
| Basic Flow | |
| This use case starts when an user accesses the “Delete product information" feature of the Website.  The Website shows a list of product information.  User enters a product information and clicks Delete.  Website validates product information.  User is returned to User panel.  Use Case ends. | |
| Alternate Flows | |
| Title | Description |
| Invalid Information Entered | The Webiste displays an “Invalid Product Information” error after the user deletes the information and asks the User to re-enter the information.  User re-enters information and clicks delete. |
| Cancel Delete Product Informaion | User clicks cancel after selecting the “Delete product information ” feature.  Website returns User to the User panel. |
| Pre-Conditions | |
| Title | Description |
|  |  |
| Post-Conditions | |
| Title | Description |
| Success | The product information is deleted from the database. |
| Failure | The product information is not deleted for one or more reasons. |
| Extension Points | |
| None | |

### 2.4.6 Update product information

|  |  |
| --- | --- |
| Name | Update product information |
| Brief Description | An User of the System updates an existing product in the database |
| Actor(s) | User ( Admin, Customer) |
| Flow of Events | |
| Basic Flow | |
| This use case starts when an User accesses the “ Update product information” feature of the system.  The Website displays the current list of product.  The User selects the product wants to edit.  The Website displays the corresponding information to the product selected.  The User edits the information accordingly and submits.  The Website updates the information.  Use Case ends. | |
| Alternate Flows | |
| Title | Description |
| Cancel Update product | User selects the “Cancel” option.  Website returns User to user panel. |
| Invalid Product Entered | The Website displays an “Invalid Information” error after the user submits the information and asks the User to re-enter the information.  User re-enters information and clicks submit. |
| Pre-Conditions | |
| Title | Description |
|  |  |
| Post-Conditions | |
| Title | Description |
| Success | The product information is changed. |
| Failure | The product information is unchanged. |
| Extension Points | |
| None | |

# Chapter 03 Test plan

## 3.1 Introduction

### 3.1.1 Purpose

This is a test plan outlining the work, methods and manpower to test the e-commerce website for selling book .

Document include 5 parts:

Part 1:INTRODUCTION

Part 2:TEST STRATEGY

Part 3:EXECUTION STRATEGY

Part 4:TEST MANAGEMENT PROCESS

Part 5:TEST ENVIRONMENT

Test Strategy: rules the test will be based on, including the givens of the project (e.g.: start / end dates, objectives, assumptions); description of the process to set up a valid test (e.g.: entry / exit criteria, creation of test cases, specific tasks to perform, scheduling, data strategy).

Execution Strategy: describes how the test will be performed and process to identify and report defects, and to fix and implement fixes.

Test Management: process to handle the logistics of the test and all the events that come up during execution (e.g.: communications, escalation procedures, risk and mitigation, team roster)

### 3.1.2 Project Overview

Our website is a powerful tool providing customers with the ability to view relevant information such as product information, personal information, and updating personal information.

This website is a place where people can easily find and buy books that you looking for.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Function Code | Function | Description | Test Purpose |
| 1 | LI | Login system | Check that the login function has implemented the correct permissions according to the user's rights | Check information: username, password |
| 2 | LF | Look for information | Find information about book, as well as customer information | Enter any message, the corresponding information will be displayed |
| 3 | MPI | Manage product information | Add, Edit product information | Check whether the added or updated data is correct or not of the data type |
| 4 | MCI | Manage  Customer Information | Manage customer information with actions such as Add, Edit, and Delete customer information | Make sure the data entered into the database is correct |
| 5 | MA | Manage  Account | Manage information of members | Ensuring data is put into the database while ensuring the decentralization of members |
| 6 | MP | Manage promotions | Manage the launch of promotions | Make sure the data entered is correct: promotional time ... |
| 7 | MCF | Manage cart functions | Manage the adding and removing of items that customers have selected | Correct display information about the items that customers have chosen such as name, price, quantity ... |
| 8 | TTKH | Payment to customers | Manage customer payments | Check that information that customers buy, and at the same time check that information about the items that customers buy is correct. |

### 3.1.3 Audience

Project team members perform tasks specified in this document and provide input and recommendations on this document.

Project Manager Plans for the testing activities in the overall project schedule, reviews the document, tracks the performance of the test according to the task herein specified, approves the document and is accountable for the results.

Technical Team ensures that the test plan and deliverables are in line with the design, provides the environment for testing and follows the procedures related to the fixes of defects.

Business analysts will provide their inputs on functional changes.

### 3.1.4 Definitions, Acronyms, and Abbreviations

| Abbreviations | Description | Note |
| --- | --- | --- |
| AT | Acceptance test |  |
| DMS | Defect Management System (Fsoft tool) |  |
| IT | Integration test |  |
| PM | Project Manager |  |
| PTL | Project Technical Leader |  |
| QA | Quality Assurance |  |
| SRS | Software Requirement Specification |  |
| ST | System test |  |
| TP | Test Plan |  |
| TC | Test Case |  |
| TR | Test Report |  |
| UAT | User Acceptance test |  |
| UT | Unit test |  |

### 3.1.5 References

| Title/File name | Author | Version | Effective Date |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 3.2 Test strategy

### 3.2.1 Test Objectives

* Check that whether website functionality(payment, looking, add, edit…) is working as expected without any error or bugs in real business environment
* Check that the external interface of the website such as UI is working as expected and & meet the customer need
* Verify the usability of the website. Are those functionalities convenient for user or not?

### 3.2.2 Test Principles

* Testing will be focused on meeting business objectives, cost efficiency, and quality.
* There will be common, consistent procedures for all teams supporting testing activities.
* Testing processes will be well defined, yet flexible, with the ability to change as needed.
* Testing activities will build upon previous stages to avoid redundancy or duplication of effort.
* Testing environment and data will emulate a production environment as much as possible.
* Testing will be a repeatable, quantifiable, and measurable activity.
* Testing will be divided into distinct phases, each with clearly defined objectives and goals.
* There will be an entrance and exit criteria.

### 3.2.3 Scope

In-scope items:[Functional Testing](https://www.guru99.com/functional-testing.html), Api Testing

Out of scope items: [Database Testing](https://www.guru99.com/data-testing.html), hardware & any other external interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Unit testing | Intergration Testing | System Testing | Acceptance testing |
| Functions are tested | class, function, procedure |  |  |  |
| Test techniques | White box test | Black box test |  |  |

### 3.2.4 Functional Testing

-Function Testing

|  |  |
| --- | --- |
| Test purpose: | Ensuring the correct functional test objective, including orientation, input, processing and data received |
| Way: | Execute each UC, UC cycle or function, using valid and invalid data to check:  Expected results with valid data.  - An appropriate error or message is displayed when the data is not valid.  - Each business rule is applied correctly |
| Completion conditions: | The whole test plan has been done.  - All detected errors have been recorded. |
| Special issues: | Identify or describe problems (internal or external) that affect functionality testing. |

-User Interface Testing

|  |  |
| --- | --- |
| Test purpose: | Check:  Use through test targets accurately reflects business functions and requirements, including screen to screen, field to school, and access methods (tabs, hover, key combinations )  Screen objects and properties such as menus, size, position, state, and focus on being compliant with standards |
| Way: | Create and edit tests for each screen to check proper usage and status of objects for each screen and app object |
| Completion conditions: | Each screen has been successfully tested according to the test version or acceptable range |
| Special issues: | Not all properties of the objects are accessible |

### 3.2.5 Regression Testing

|  |  |
| --- | --- |
| Test purpose: | Regression test is used to check corrected parts in software, to make sure that the changes did not cause errors in other parts. |
| Way: | Reuse TCs from previous tests to test repaired modules.  • Use the Rational Robot tool: Create some functional test scripts. Define a schedule for them  • 80% of all TCs are randomly selected |
| Completion conditions: | All TCs are done and satisfactory  All selected organizations are implemented and met the requirements |

### 3.2.6 Security and Access Control Testing

|  |  |
| --- | --- |
| Test purpose: | Application level security: Ensuring that a user can only access functions or data for which that user group is allowed  System-level security: Ensure that only those authorized to access the system and applications are allowed to access them |
| Way: | Application security: Identify and list each user group and the functions or data they are allowed to access  Create test cases for each user group and test each permission by creating defined transactions for each group  • Modify the user group and re-run the test case for the same users. For each case, check whether the functions are added or the data is correct or denied.  • System level access: refer to the special conditions below |
| Completion conditions: | For each user group has the appropriate functions or data, and all transactional functions are as expected and run in previous application functionality tests |
| Special issues: | Access to the system must be reviewed or discussed with the system administrator or network administrator, may not be required if it is a function of the network administrator or system administrator |

### 3.2.7. MILESTONE LIST

The milestone list is tentative and may change due to below reasons

a) Any issues in the System environment readiness

b) Any change in scope/addition in scope

c) Any other dependency that impacts efforts and timelines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone** | **Deliverables** | **Duration** | **Start Date** | **End Date** |
| Create Testplan | Testplan |  |  |  |
| Create Testcase | Testcase |  |  |  |
| Run Testcase |  |  |  |  |
| Review Testplan+Testcase |  |  |  |  |

### 3.2.8 Test tool

|  |  |  |  |
| --- | --- | --- | --- |
| Purpose | Tool | Supplier / Self-built | Version |
|  |  |  |  |

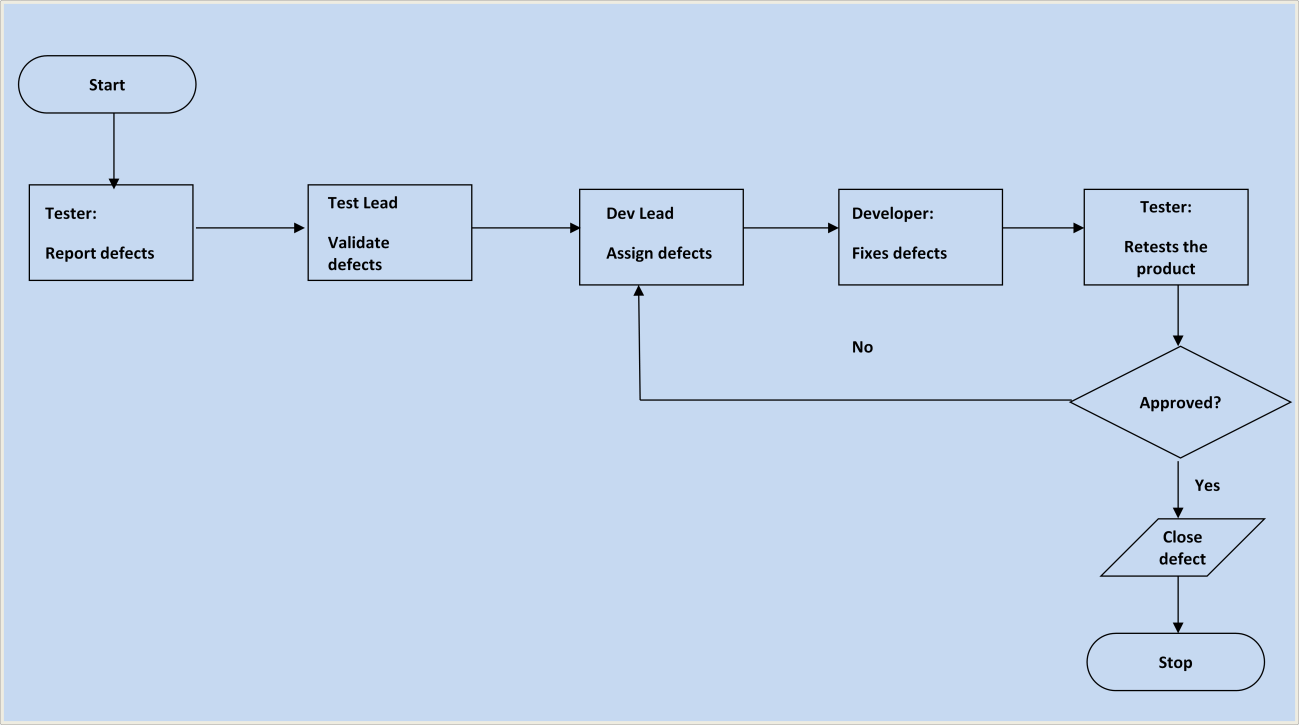
## 3.3 Execution strategy

### 3.3.1 Entry and Exit Criteria

* The entry criteria refer to the desirable conditions in order to start test execution; only the migration of the code and fixes need to be assessed at the end of each cycle.
* The exit criteria are the desirable conditions that need to be met in order proceed with the implementation.
* Entry and exit criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation. All this is input to the project manager for a final “go-no go” decision.
* Entry criteria to start the execution phase of the test: the activities listed in the Test Planning section of the schedule are 100% completed.
* Entry criteria to start each cycle: the activities listed in the Test Execution section of the schedule are 100% completed at each cycle.

### 3.3.2 Defect tracking & Reporting

Following flowchart depicts Defect Tracking Process:



## 3.4 TEST MANAGEMENT PROCESS

### 3.4.1.Role Expectations

|  |  |  |  |
| --- | --- | --- | --- |
| No | Roles | Name | Contact Info |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 3.5. TEST ENVIRONMENT

Window 10

### 3.5.1. System

Hardware:PC

Software: Visual studio code

PRODUCT

| No | Product | Finished date | File/Link | Submitter |
| --- | --- | --- | --- | --- |
| 1 | Test cases |  |  |  |
| 2 | Test report |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Chapter 04 Describe in details about the testing methods and techniques

## 4.1 Unit Testing

### 4.1.1 What is Unit Testing?

Unit testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a White-Box testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

### 4.1.2 Why Unit Testing?

Unit Testing is important because software developers sometimes try saving time doing minimal unit testing and this is myth because inappropriate unit testing leads to high cost Defect fixing during System Testing, Integration Testing and even Beta Testing after application is built. If proper unit testing is done in early development, then it saves time and money in the end.

Here, are the key reasons to perform unit testing :

1. Unit tests help to fix bugs early in the development cycle and save costs.
2. It helps the developers to understand the code base and enables them to make changes quickly
3. Good unit tests serve as project documentation
4. Unit tests help with code re-use. Migrate both your code and your tests to your new project. Tweak the code until the tests run again.

### 4.1.3 How to do Unit Testing

In order to do Unit Testing, developers write a section of code to test a specific function in software application. Developers can also isolate this function to test more rigorously which reveals unnecessary dependencies between function being tested and other units so the dependencies can be eliminated. Developers generally use UnitTest framework to develop automated test cases for unit testing.

Unit Testing is of two types

* Manual
* Automated

Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document.

Under the automated approach-

* A developer writes a section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is deployed.
* A developer could also isolate the function to test it more rigorously. This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment. Isolating the code helps in revealing unnecessary dependencies between the code being tested and other units or data spaces in the product. These dependencies can then be eliminated.
* A coder generally uses a UnitTest Framework to develop automated test cases. Using an automation framework, the developer codes criteria into the test to verify the correctness of the code. During execution of the test cases, the framework logs failing test cases. Many frameworks will also automatically flag and report, in summary, these failed test cases. Depending on the severity of a failure, the framework may halt subsequent testing.
* The workflow of Unit Testing is 1) Create Test Cases 2) Review/Rework 3) Baseline 4) Execute Test Cases.

### 4.1.4 Unit Testing Techniques

The Unit Testing Techniques are mainly categorized into three parts which are Black box testing that involves testing of user interface along with input and output, White box testing that involves testing the functional behaviour of the software application and Gray box testing that is used to execute test suites, test methods, test cases and performing risk analysis.

Code coverage techniques used in Unit Testing are listed below:

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Condition Coverage
* Finite State Machine Coverage

### 4.1.5 Unit Testing Example: Mock Objects

Unit testing relies on mock objects being created to test sections of code that are not yet part of a complete application. Mock objects fill in for the missing parts of the program.

For example, you might have a function that needs variables or objects that are not created yet. In unit testing, those will be accounted for in the form of mock objects created solely for the purpose of the unit testing done on that section of code.

### 4.1.6 Unit Testing Tools

There are several automated tools available to assist with unit testing. We will provide a few examples below:

1. Junit: Junit is a free to use testing tool used for Java programming language. It provides assertions to identify test method. This tool test data first and then inserted in the piece of code.
2. NUnit: NUnit is widely used unit-testing framework use for all .net languages. It is an open source tool which allows writing scripts manually. It supports data-driven tests which can run in parallel.
3. JMockit: JMockit is open source Unit testing tool. It is a code coverage tool with line and path metrics. It allows mocking API with recording and verification syntax. This tool offers Line coverage, Path Coverage, and Data Coverage.
4. EMMA: EMMA is an open-source toolkit for analyzing and reporting code written in Java language. Emma support coverage types like method, line, basic block. It is Java-based so it is without external library dependencies and can access the source code.
5. PHPUnit: PHPUnit is a unit testing tool for PHP programmer. It takes small portions of code which is called units and test each of them separately. The tool also allows developers to use pre-define assertion methods to assert that a system behave in a certain manner.

Those are just a few of the available unit testing tools. There are lots more, especially for C languages and Java, but you are sure to find a unit testing tool for your programming needs regardless of the language you use.

### 4.1.7 Unit Testing Advantage

Developers looking to learn what functionality is provided by a unit and how to use it can look at the unit tests to gain a basic understanding of the unit API.

Unit testing allows the programmer to refactor code at a later date, and make sure the module still works correctly (i.e. Regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified and fixed.

Due to the modular nature of the unit testing, we can test parts of the project without waiting for others to be completed.

### 4.1.8 Unit Testing Disadvantages

Unit testing can't be expected to catch every error in a program. It is not possible to evaluate all execution paths even in the most trivial programs

Unit testing by its very nature focuses on a unit of code. Hence it can't catch integration errors or broad system level errors.

## 4.2 Integration testing

### 4.2.1 What is Integration Testing?

INTEGRATION TESTING is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated.

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'.

### 4.2.2 Why do Integration Testing?

Although each software module is unit tested, defects still exist for various reasons like

* A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

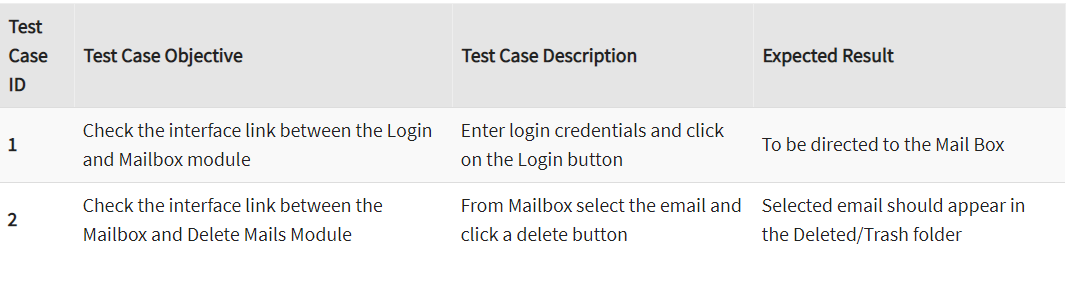
### 4.2.3 Example of Integration Test Case

Integration Test Case differs from other test cases in the sense it focuses mainly on the interfaces & flow of data/information between the modules. Here priority is to be given for the integrating links rather than the unit functions which are already tested.

Sample Integration Test Cases for the following scenario: Application has 3 modules say 'Login Page', 'Mailbox' and 'Delete emails' and each of them is integrated logically.

Here do not concentrate much on the Login Page testing as it's already been done in Unit Testing. But check how it's linked to the Mail Box Page.

Similarly Mail Box: Check its integration to the Delete Mails Module.



### 4.2.4 Approaches, Strategies, Methodologies of Integration Testing

Software Engineering defines variety of strategies to execute Integration testing, viz.

* Big Bang Approach :
* Incremental Approach: which is further divided into the following
* Top Down Approach
* Bottom Up Approach
* Sandwich Approach - Combination of Top Down and Bottom Up

Below are the different strategies, the way they are executed and their limitations as well advantages.

**1) Big Bang Testing**

Advantages:

* Convenient for small systems.

Disadvantages:

* Fault Localization is difficult.
* Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.
* Since the Integration testing can commence only after "all" the modules are designed, the testing team will have less time for execution in the testing phase.
* Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

**2) Incremental Testing**

In the Incremental Testing approach, testing is done by integrating two or more modules that are logically related to each other and then tested for proper functioning of the application. Then the other related modules are integrated incrementally and the process continues until all the logically related modules are integrated and tested successfully.

Incremental Approach, in turn, is carried out by two different Methods:

* Bottom Up
* Top Down

### 4.2.5 How to do Integration Testing?

The Integration test procedure irrespective of the Software testing strategies (discussed above):

1. Prepare the Integration Tests Plan
2. Design the Test Scenarios, Cases, and Scripts.
3. Executing the test Cases followed by reporting the defects.
4. Tracking & re-testing the defects.
5. Steps 3 and 4 are repeated until the completion of Integration is successful.

### 4.2.6 Entry and Exit Criteria of Integration Testing

Entry and Exit Criteria to Integration testing phase in any software development model

**Entry Criteria:**

* Unit Tested Components/Modules
* All High prioritized bugs fixed and closed
* All Modules to be code completed and integrated successfully.
* Integration tests Plan, test case, scenarios to be signed off and documented.
* Required Test Environment to be set up for Integration testing

**Exit Criteria:**

* Successful Testing of Integrated Application.
* Executed Test Cases are documented
* All High prioritized bugs fixed and closed
* Technical documents to be submitted followed by release Notes.

## 4.3 Sandwich Testing

### 4.3.1 What is System Testing?

**System testing** is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

**System Testing is Blackbox**

Two Category of Software Testing

* Black Box Testing
* White Box Testing

System test falls under the black box testing category of software testing.

White box testing is the testing of the internal workings or code of a software application. In contrast, black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

### 4.3.2 What do you verify in System Testing?

System Testing involves testing the software code for following

* Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
* Verify thorough testing of every input in the application to check for desired outputs.
* Testing of the user's experience with the application.

That is a very basic description of what is involved in system testing. You need to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

### 4.3.3 Different Types of System Testing

There are more than 50 types of System Testing. For an exhaustive list of software testing types click here. Below we have listed types of system testing a large software development company would typically use

* Usability Testing- mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives
* Load Testing- is necessary to know that a software solution will perform under real-life loads.
* Regression Testing- involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.
* Recovery testing - is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.
* Migration testing- is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.
* Functional Testing - Also known as functional completeness testing, Functional Testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.
* Hardware/Software Testing - IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.

### 4.3.4 What Types of System Testing Should Testers Use?

There are over 50 different types of system testing. The specific types used by a tester depend on several variables. Those variables include:

* Who the tester works for - This is a major factor in determining the types of system testing a tester will use. Methods used by large companies are different than that used by medium and small companies.
* Time available for testing - Ultimately, all 50 testing types could be used. Time is often what limits us to using only the types that are most relevant for the software project.
* Resources available to the tester - Of course some testers will not have the necessary resources to conduct a testing type. For example, if you are a tester working for a large software development firm, you are likely to have expensive automated testing software not available to others.
* Software Tester's Education- There is a certain learning curve for each type of software testing available. To use some of the software involved, a tester has to learn how to use it.
* Testing Budget - Money becomes a factor not just for smaller companies and individual software developers but large companies as well.

## 4.4 Acceptance Testing

### 4.4.1 What is Acceptance Testing?

Once the System Testing process is completed by the testing team and is signed-off, the entire Product/application is handed over to the customer/few users of customers/both, to test for its acceptability i.e., Product/application should be flawless in meeting both the critical and major Business requirements. Also, end-to-end business flows are verified similar as in real-time scenario.

The production-like environment will be the testing environment for Accepting Testing (Usually termed as Staging, Pre-Prod, Fail-Over, UAT environment).

This is a [black-box testing technique](https://www.softwaretestinghelp.com/black-box-testing/) where only the functionality is verified to ensure that the product meets the specified acceptance criteria (no need for design/implementation knowledge).

### 4.4.2 Why Acceptance Tests?

Though System testing has been completed successfully, the Acceptance test is demanded by the customer. Tests conducted here are repetitive, as they would have been covered in System testing.

Then, why is this testing is conducted by customers?

This is because:

* To gain confidence in the product that is getting released to the market.
* To ensure that the product is working in the way it has to.
* To ensure that the product matches current market standards and is competitive enough with the other similar products in the market.

### 4.4.3 Types

**4.4.3.1 User Acceptance Testing (UAT)**

UAT is to assess whether the Product is working for the user, correctly for the usage. Specific requirements which are quite often used by the end-users are primarily picked for the testing purpose. This is also termed as End-User Testing.

The term “User” here signifies the end-users to whom the Product/application is intended and hence, testing is performed from the end-users perspective and from their point of view.

#### 4.4.3.2 Business Acceptance Testing (BAT)

This is to assess whether the Product meets the business goals and purposes or not.

BAT mainly focuses on business benefits (finances) which are quite challenging due to the changing market conditions/advancing technologies so that the current implementation may have to undergo changes which result in extra budgets.

Even the Product passing the technical requirements may fail BAT due to these reasons.

#### 4.4.3.3 Contract Acceptance Testing (CAT)

This is a contract which specifies that once the Product goes live, within a predetermined period, the acceptance test must be performed and it should pass all the acceptance use cases.

Contract signed here is termed as Service Level Agreement (SLA), which includes the terms where the payment will be made only if the Product services are in-line with all the requirements, which means the contract is fulfilled.

Sometimes, this contract may happen before the Product goes live. Either the ways, a contract should be well defined in terms of the period of testing, areas of testing, conditions on issues encountered at later stages, payments, etc.

#### 4.4.3.4 Regulations/Compliance Acceptance Testing (RAT)

This is to assess whether the Product violates the rules and regulations that are defined by the government of the country where it is being released. This may be unintentional but will impact negatively on the business.

Usually, the developed Product/application that is intended to be released all over the world, has to undergo RAT, as different countries/regions have different rules and regulations defined by its governing bodies.

If any of the rules and regulations are violated for any country, then that country or the specific region in that country will not be allowed to use the Product and is considered as a Failure. Vendors of the Product will be directly responsible if the Product is released even though there is a violation.

#### 4.4.3.5 Operational Acceptance Testing (OAT)

This is to assess the operational readiness of the Product and is a non-functional testing. It mainly includes testing of recovery, compatibility, maintainability, technical support availability, reliability, fail-over, localization etc.

OAT mainly assures the stability of the Product before releasing it to the production.

#### 4.4.3.6 Alpha Testing

This is to assess the Product in the development/testing environment by a specialized testers team usually called alpha testers. Here, the testers feedback, suggestions help to improve the Product usage and also to fix certain bugs.

Here, testing happens in a controlled manner.

#### 4.4.3.7 Beta Testing/Field Testing

This is to assess the Product by exposing it to the real end-users, usually called beta testers/beta users, in their environment. Continuous feedback from the users is collected and the issues are fixed. Also, this helps in enhancing/improving the Product to give a rich user experience.

Testing happens in an uncontrolled manner, which means a user has no restrictions on the way in which the Product is being used.

### 4.4.4 Qualities of Acceptance Testers

Testers with the below qualities are qualified as Acceptance testers:

* Ability to think logically and analytically.
* Good domain knowledge.
* Able to study the competitive products in the market and analyze the same in the developed product.
* Having end-user perception while testing.
* Understand business need for each requirement and test accordingly.

### 4.4.5 Impact of Issues found during this testing

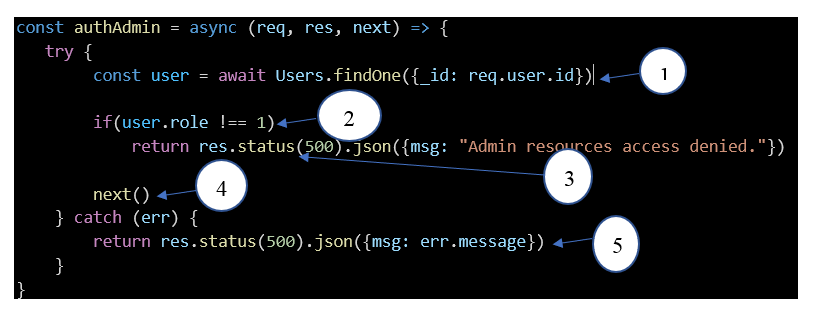
Any issues encountered in Acceptance test phase should be considered as a high priority one and fixed immediately. This also requires Root Cause Analysis to be performed on each and every issue that is found.

The testing team plays a major role in providing RCA’s for Acceptance issues. These also help in determining how efficiently testing is performed.

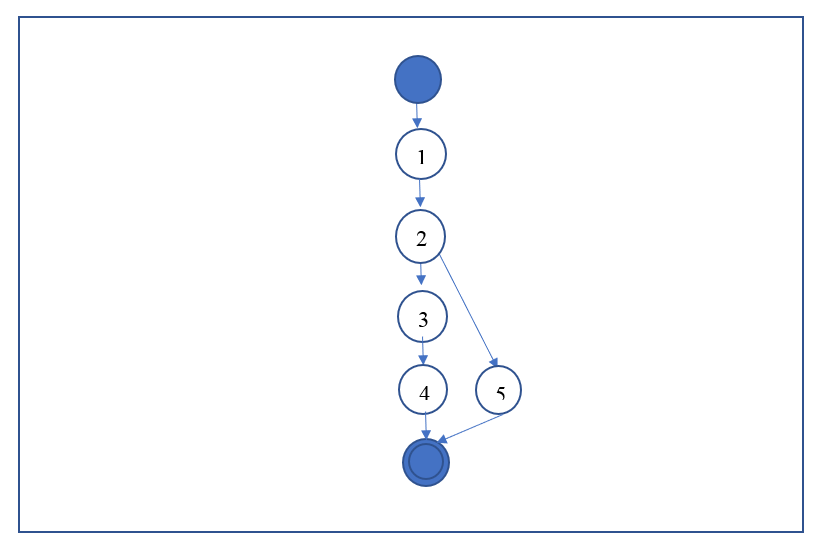
Also, valid issues in acceptance test will hit both the testing and the development team efforts in terms of impression, ratings, customer surveys, etc. Sometimes, if any ignorance from the testing team on validations is found, it leads to escalations as well.

# Chapter 05 Choose at least 3 code units / functions / APIs / methods, etc. to perform whitebox testing

## 5.1 Authorize Admin



### 5.1.1. Create Control flow graph



### 5.1.2 Calculate Cyclomatic complexity to identify independent linear execution paths.

The graph above has 1 binary decision node so Cyclomatic complexity C = 1+1 = 2

### 5.1.3 Design test cases for each path

The 2 basic independent linear execution lines are

* 1 => 2 => 3 => 4
* 1 => 2 => 5

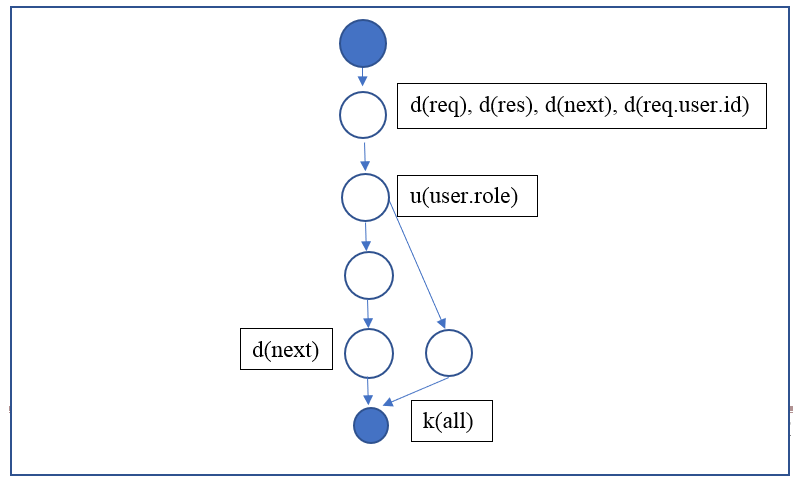
Design test case for line 1: 1 ->2 -> 3 -> 4

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected result** | **Actual result** |
| Req =1 | Access successfully | Access successfully |

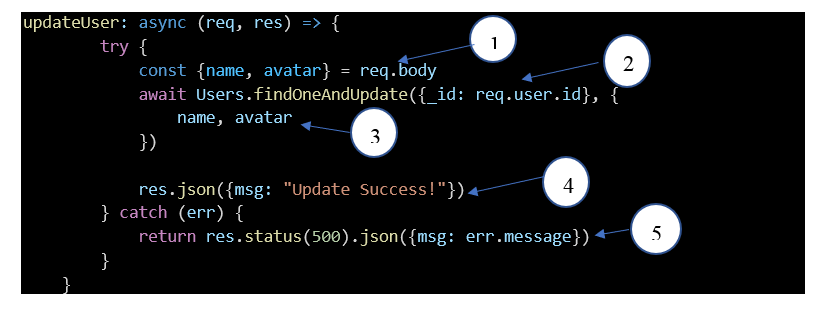
Design test case for line 2: 1 ->2 -> 5

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected result** | **Actual result** |
| Req = 0 | Access failed | Access failed |

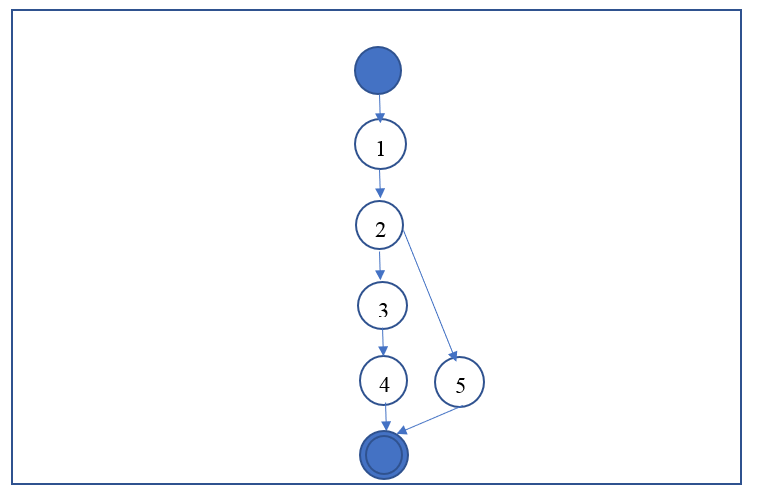
### 5.1.4 Create the general data flow graph



## 5.2 Update user

****

### 5.2.1 Create Control flow graph



### 5.2.2 Calculate Cyclomatic complexity to identify independent linear execution paths.

The graph above has 1 binary decision node so Cyclomatic complexity C = 1+1 = 2

### 5.2.3 Design test cases for each path

The 2 basic independent linear execution lines are

* 1 => 2 =>3 =>4
* 1 =>2 => 5

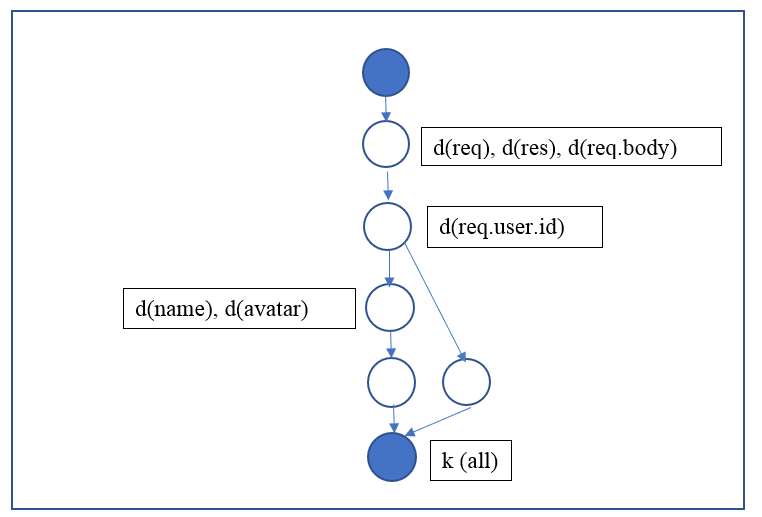
Design test case for line 1: 1 =>2 => 3 => 4

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected result** | **Actual result** |
| Req =1 | Update successfully | Update successfully |

Design test case for line 2: 1 ->2 -> 5

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected result** | **Actual result** |
| Req = 1000 | Update failed | Update failed |

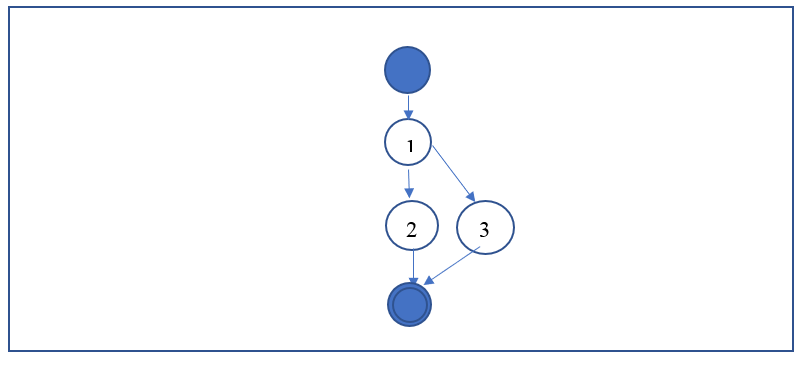
### 5.2.4 Create the general data flow graph



## 5.3 Delete User



### 5.3.1 Create Control flow graph

****

### 5.3.2 Calculate Cyclomatic complexity to identify independent linear execution paths.

The graph above has 1 binary decision node so Cyclomatic complexity C = 1+1 = 2

### 5.3.3 Design test cases for each path

The 2 basic independent linear execution lines are

* 1 -> 2
* 1 -> 3

Design test case for line 1: 1 ->2

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected result** | **Actual result** |
| Req = 10 | Delete successfully | Delete successfully |

Design test case for line 2: 1 ->3

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected result** | **Actual result** |
| Req = 1000 | Delete failed | Delete failed |

### 5.3.4 Create the general data flow graph

d (req), d (res), d(req.params.id)

k (all)

# Chapter 06 Bug Report

## 6.1 Bug Report Template: B1 (G1-31)

|  |  |
| --- | --- |
| **ID number** | B1 (G1-31) |
| **Name** | Test any number of selected product |
| **Reporter** | Jason |
| **Submit Date** | 20/01/2021 |
| **Summary** | When I select number of product greater than data storage, Ido not receive a notification. |
| **URL** | [**https://testing-g1.herokuapp.com/**](https://testing-g1.herokuapp.com/) |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10, 64-bit |
| **Browser** | Chrome |
| **Severity** | Major |
| **Assigned to** | / |
| **Priority** | High |

**Description**

When I select number of product greater than data storage, I do not receive a notification.

**Steps to reproduce**

* Open Website
* Login as User
* Select “Tô bình yên vẽ hạnh phúc”

**Expected result**

Show table messages if values greater than specified max limit

**Actual result**

Do not have a notification

**Notes**

## 6.2 Bug Report Template: B2 (G1-22)

|  |  |
| --- | --- |
| **ID number** | B2 (G1-22) |
| **Name** | Test any specific input character |
| **Reporter** | Jason |
| **Submit Date** | 20/01/2021 |
| **Summary** | When I input “###” to email box for register, I do not receive any announcement |
| **URL** | [**https://testing-g1.herokuapp.com/**](https://testing-g1.herokuapp.com/) |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10, 64-bit |
| **Browser** | Chrome |
| **Severity** | Major |
| **Assigned to** | / |
| **Priority** | High |

**Description**

When I input “###” to email box for register, I do not receive any announcement

**Steps to reproduce**

* Open Website
* Register
* Input a “###” in message box

**Expected result**

Show table messages if values greater than specified max limit

**Actual result**

Show message to announce for special characters

**Notes**

## 6.3 Bug Report Template: B3 (G1-38)

|  |  |
| --- | --- |
| **ID number** | B3 (G1-38) |
| **Name** | Test error page load |
| **Reporter** | Jason |
| **Submit Date** | 20/01/2021 |
| **Summary** | Check error page |
| **URL** | [**https://testing-g1.herokuapp.com/**](https://testing-g1.herokuapp.com/) |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10, 64-bit |
| **Browser** | Chrome |
| **Severity** | Major |
| **Assigned to** | / |
| **Priority** | High |

**Description**

Check error page

**Steps to reproduce**

* Open Website
* Login as User
* Choose filters do not have product

**Expected result**

When I choose filters without product, I redirect to error page

**Actual result**

The page loading.

**Notes**

## 6.4 Bug Report Template: B4 (G1-03)

|  |  |
| --- | --- |
| **ID number** | B4 (G1-03) |
| **Name** | Security Testing Test Scenarios - Check for SQL injection attacks. |
| **Reporter** | Nguyen Van Dan |
| **Submit Date** | January 2, 2021 |
| **Summary** | Input speacial characters “###” in login page but can not access to main page by admin role. |
| **URL** | https://testing-g1.herokuapp.com/login |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10 Pro |
| **Browser** | Chrome |
| **Severity** | Major |
| **Assigned to** | Doan Viet Thuan |
| **Priority** | High |

**Description**

When you in put speacial characters "###" in value email address in login page page but can not access to main page by admin role.

**Steps to reproduce**

> Open the webpage

> Input speacial characters "###" in value email address

**Expected result**

Access to main page by admin role.

**Actual result**

Unexpected

**Notes**

## 6.5 Bug Report Template: B5 (G1-19)

|  |  |
| --- | --- |
| **ID number** | B5 (G1-19) |
| **Name** | Security Testing Test Scenarios - Verify CAPTCHA functionality. |
| **Reporter** | Nguyen Van Dan |
| **Submit Date** | January 2, 2021 |
| **Summary** | In page login have not verify CAPTCHA functionality. |
| **URL** | https://testing-g1.herokuapp.com/login |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10 Home Single Language |
| **Browser** | Chrome |
| **Severity** | Major |
| **Assigned to** | Doan Viet Thuan |
| **Priority** | Medium |

**Description**

In page login have not verify CAPTCHA functionality.

**Steps to reproduce**

> Open the webpage

> Login as user

> Verify CAPTCHA functionality in login page

**Expected result**

Form Captcha appear

**Actual result**

Unexpected

**Notes**

## 6.6 Bug Report Template: B6 (G1-15)

|  |  |
| --- | --- |
| **ID number** | B6 (G1-15) |
| **Name** | Test Scenarios For Result Grid - Check for Next, Previous, First and Last page pagination functionality. |
| **Reporter** | Nguyen Van Dan |
| **Submit Date** | January 2, 2021 |
| **Summary** | Do some activities such as add, edit, delete then check for Next, Previous, First and Last page pagination functionality. But just Previous functionality work |
| **URL** | https://testing-g1.herokuapp.com/edit\_product/5fe89c07f6bd1337b4b4e04b |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10 Home 64-bit |
| **Browser** | Chrome |
| **Severity** | Major |
| **Assigned to** | Doan Viet Thuan |
| **Priority** | High |

**Description**

Do some activities such as add, edit, remove then check for Next, Previous, First and Last page pagination functionality. But just Previous functionality work

**Steps to reproduce**

> Open the webpage

> Login as student

> Check Previous functionality

> Check Next functionality

> Check First Page functionality

> Check Last Page functionality

**Expected result**

The webpage should move to Previous Page/ Next page/ First Page/Last Page

**Actual result**

Unexpected (Except Previous)

**Notes**

## 6.7 Bug Report Template: B7(G1- 05)

|  |  |
| --- | --- |
| **ID number** | B7(G1- 05) |
| **Name** | Security Testing Test Scenarios - Secure pages should use the HTTPS protocol. |
| **Reporter** | Nguyen Van Dan |
| **Submit Date** | January 2, 2021 |
| **Summary** | The page does not use the HTTPS protocol |
| **URL** | https://testing-g1.herokuapp.com/create\_product |
| **Screenshot** |  |
| **Platform** | Window |
| **Operating System** | Window 10 Home Single Language |
| **Browser** | Google Chrome |
| **Severity** | Major |
| **Assigned to** | Doan Viet Thuan |
| **Priority** | High |

**Description**

The page does not use the HTTPS protocol

**Steps to reproduce**

>Open the webpage

>Login as admin

>Check page

**Expected result**

Secure pages should use the HTTPs protocol

**Actual result**

Unexpected

**Notes**

# Chapter 07 Bug Fix

## 7.1 Test case ID (G1-31)

|  |  |
| --- | --- |
| **Developer’s Name** | Jason |
| **Fix date** | 20/01/2021 |
| **Test case ID** | G1-31 |
| **Test Description** | Test any number of selected product |
| **Cause of the bug** | Lack of add handle exception |
| **How to fix** | Add handle exception of the value of quantity if the user add more than 100 |
| **Notes** | No notes |

## 7.2 Test case ID (G1-22)

|  |  |
| --- | --- |
| **Developer’s Name** | Jason |
| **Fix date** | 20/01/2021 |
| **Test case ID** | G1-22 |
| **Test Description** | Test any specific input character |
| **Cause of the bug** | Have not handle exception in value of email address |
| **How to fix** | Add handle exception in value of email address to show the message if the user enter the special characters |
| **Notes** | No notes |

## 7.3 Test case ID (G1-38)

|  |  |
| --- | --- |
| **Developer’s Name** | Jason |
| **Fix date** | 20/01/2021 |
| **Test case ID** | G1-38 |
| **Test Description** | Check error page |
| **Cause of the bug** | Do not have a error page |
| **How to fix** | Create error page and redireted to the error page when the user enter the unvailable page |
| **Notes** | No notes |
| **Developer’s Name** | Doan Viet Thuan |
| **Fix date** | January 2, 2021 |
| **Test case ID** | G1-03 |
| **Test Description** | When you in put speacial characters "###" in value email address in log in page but but can not access to main page by admin role. |
| **Cause of the bug** | Have not handle exception in value of email address |
| **How to fix** | Add handle exception in value of email address access to main page by admin role. |
| **Notes** |  |

## 7.4 Test case ID (G1-03)

|  |  |
| --- | --- |
| **Developer’s Name** | Doan Viet Thuan |
| **Fix date** | January 2, 2021 |
| **Test case ID** | G1-03 |
| **Test Description** | When you in put speacial characters "###" in value email address in log in page but but can not access to main page by admin role. |
| **Cause of the bug** | Have not handle exception in value of email address |
| **How to fix** | Add handle exception in value of email address access to main page by admin role. |
| **Notes** |  |

## 7.5 Test case ID (G1-19)

|  |  |
| --- | --- |
| **Developer’s Name** | Doan Viet Thuan |
| **Fix date** | January 2, 2021 |
| **Test case ID** | G1-19 |
| **Test Description** | Test verify CAPTCHA functionality. |
| **Cause of the bug** | Missing the code that verify captcha while login |
| **How to fix** | Add a code to verify captcha in page login |
| **Notes** |  |

## 7.6 Test case ID (G1-15)

|  |  |
| --- | --- |
| **Developer’s Name** | Doan Viet Thuan |
| **Fix date** | January 2, 2021 |
| **Test case ID** | G1-15 |
| **Test Description** | Do some activities such as add, edit, remove then check for Next, Previous, First and Last page pagination functionality. |
| **Cause of the bug** | Did not add Next, First and Last page pagination functionality |
| **How to fix** | Add function Next, First and Last page |
| **Notes** |  |

## 7.7 Test case ID (G1-05)

|  |  |
| --- | --- |
| **Developer’s Name** | Doan Viet Thuan |
| **Fix date** | January 2, 2021 |
| **Test case ID** | G1- 05 |
| **Test Description** | Check secure page should use the HTTPS protocol |
| **Cause of the bug** | Did not register certificate SSL in server and verification code CSR in hosting |
| **How to fix** | Register certificate SSL in server and create verification code CSR in hosting |
| **Notes** |  |

**REFERENCE**

* Beizer, B. (2003). Software testing techniques. Dreamtech Press.
* Sawant, A. A., Bari, P. H., & Chawan, P. M. (2012). Software testing techniques and strategies. International Journal of Engineering Research and Applications (IJERA), 2(3), 980-986.
* Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2016, November). Software testing techniques: A literature review. In 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M) (pp. 177-182). IEEE.
* Hooda, I., & Chhillar, R. S. (2015). Software test process, testing types and techniques. International Journal of Computer Applications, 111(13).
* Nidhra, S., & Dondeti, J. (2012). Black box and white box testing techniques-a literature review. International Journal of Embedded Systems and Applications (IJESA), 2(2), 29-50.
* Henard, C., Papadakis, M., Harman, M., Jia, Y., & Le Traon, Y. (2016, May). Comparing white-box and black-box test prioritization. In 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE) (pp. 523-534). IEEE.