Abstract

**Aim:** To select an application and perform testing techniques over it using a testing tool of our choice.

**Application Selected:** Shuup E-Commerce Platform

**Software type:** Open Source Web Application.

**Features of Shuup** include Newsletter, Email, Website Design, Inventory, Payment Gateways, and Shipping etc.

**Built on:** Django (Python web based framework)

**Software Testing Tools selected:**

* Functional Testing Tool: Selenium
* Non-Functional Testing Tool: JMeter

**Types of testing we intend to do:**

* Functional Tests:
  + Unit Testing
  + Black Box Testing
* Non-Functional Tests:
  + Load Testing
  + Stress Testing

**Items not tested**

The following modules were not tested:

* Addons - Shuup contains facilities for installing, detecting, loading and configuring additional functionality with little or no system administration knowledge needed.
* Packages that can be loaded in this way are called Addons.
* Admin - creation of new admin was not tested.
* Settings - System settings, Notifications, Data Import and GDPR were not tested.

**Testing Metrics**

* No. of test cases planned vs executed : Planned (100 ) Executed (100)
* No. of test cases passed/failed: Test Cases passed (100) Test Cases failed (0)
* No of defects identified and their Status & Severity: 1, low severity.
* Defects distribution

- If asterisk field is left unfilled in Order status, Django key error is shown instead of alert box.

**Types of testing performed**

**1 Functional Testing**

**1.1 Unit Testing**

This is a level of software testing where individual units/ components of a software are tested.

The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

Number of test cases: 25

Modules tested:

- Login

- Orders

- Campaigns

- Coupons

- Discounts

**1.2 Black-box Testing**

This is also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.

Number of test cases: 25

Modules tested:

- Reports

- Taxes

- Users

- API

- Customer Tax Groups

- Telemetry

**2 Non-Functional Testing**

**2.1 Usability Testing**

Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system.

Number of test cases: 18

Modules tested:

- Contacts

- Shops

**2.2 Compatibility Testing**

Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, network environments or

Mobile devices.

Number of test cases: 32

Modules tested:

- Forms

**Test Environment**

Test Environment consists of elements that support test execution with software, hardware and network configured. Test environment configuration must mimic the production environment in order to uncover any environment/configuration related issues.

We used Python environment for automated testing. Automated testing is the execution of the test plan by a script instead of a human. Python comes with a set of tools and libraries to enable creation of automated tests for our application.

Environment requirements:

* Python3 must be installed on the system
* Selenium must be installed on the system.

**Test Case Execution**

1. **Functional Testing**
   1. **LOGIN MODULE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | 1 | | **Test Priority** | High | |
| **Test Title** | Verify Successful login | | **Description** | To check if the admin/user is  able to successfully log in to the  application | |
| **Test designed by** |  | | **Test Design Date** |  | |
| **Test executed by** |  | | **Test Execution Date** |  | |
| **Precondition** | | Valid username and password.  Properly configured python test environment.  Pass/Fail criteria should be well known. | | |
| **Dependencies** | | Valid url | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Launch application | http://127.0.0.1:8000/sa | The page is correctly  displayed and the link is not broken. | The page is correctly  displayed and the link  is not broken | Pass |
| 2 | Enter correct username and  Password and hit the login button. | Username: admin  Password:\* | The user successfully logs in to the system. | The user successfully logs in to the system. | Pass |

|  |  |
| --- | --- |
| **Post Conditions** | Home page of the application appears. |

* 1. **ORDER MODULE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | 2 | **Test Priority** | Low |
| **Test Title** | Verify link to order is not broken | **Description** | To check if the order confirmation receipts of all customers is properly printed |
| **Test designed by** |  | **Test Design Date** |  |
| **Test executed by** |  | **Test Execution Date** |  |
| **Precondition** | Valid username and password.  Properly configured python test environment.  Pass/Fail criteria should be well known. | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Test Case 1 |  |  |  | Pass |
| 2 | Go to orders | http://127.0.0.1:8000/sa | The page is correctly  displayed and the link is not broken. | The page is correctly  displayed and the link is not broken. | Pass |

|  |  |
| --- | --- |
| **Post Conditions** | - |

* 1. **NEW MANUFACTURER’S MODULE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | 3 | **Test Priority** | Medium |
| **Test Title** | New Manufacturers | **Description** | To test if new manufacturers can be created |
| **Test designed by** |  | **Test Design Date** |  |
| **Test executed by** |  | **Test Execution Date** |  |
| **Precondition** | Valid username and password.  Properly configured python test environment.  Pass/Fail criteria should be well known. | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Test Case 1 |  |  |  |  |
| 2 | Click New manufacturers |  | New manufacturer’s page opens. | New manufacturer’s page opens. | Pass |
| 3 | Enter details | Name: Bob  Click save | New Manufacturer created | New Manufacturer created | Pass |

|  |  |
| --- | --- |
| **Post Conditions** | Success alert box |

1. Black-box Testing
   1. Module: Report

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | 25 | **Test Priority** | Low |
| **Test Title** | Verify Links to Reports is not broken | **Description** | To check if links to reports are not broken |
| **Test designed by** |  | **Test Design Date** |  |
| **Precondition** | Access the site | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Status** |
| 1 | Test Case 1 |  |  |  |
| 2 | Go to report | http://127.0.0.1:8000/sa/reports/ | The page displays correctly and is not broken | Pass |

|  |  |
| --- | --- |
| **Post Conditions** | **-** |

* 1. MODULE: Products

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Test Case ID** | 45 | **Test Priority** | Low |
| **Test Title** | Verify Link to products is not broken | **Description** | To verify if links to products are not broken |
| **Test designed by** |  | **Test Design Date** |  |
| **Precondition** | Same as previous one | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Test Case 1 |  |  |
| 2 | Go to Products | http://127.0.0.1:8000/sa/products/ | The page displays correctly and is not broken |

|  |  |
| --- | --- |
| **Post Conditions** | **-** |

2.3. Module: Add New Product

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | 46 | **Test Priority** | Medium |
| **Test Title** | Verify addition of new product | **Description** | To test if new product can be added successfully |
| **Test designed by** |  | **Test Design Date** |  |
| **Precondition** | Same as previous one | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Test Case 1 |  |  |
| 2 | Test Case 45 |  |  |
| 3 | Click on new shop product button |  | New shop product page opens |
| 4 | Enter details of new product | Name: Dell Laptop  Short description: Dell Inspiron i5 Personal Laptop  Description: Dell Inspiron Core i5 8th Gen 8250U 2018 (8 GB  RAM /2 TB HDD/Windows 10/MS Office/2 GB Graphics), 3576  Laptop, (15.6 inch, Black)  Primary Product Image: Upload from local folder  SKU: 11  Default Price: 50,000  Click on the Save button | Product gets added into the shop |

|  |  |
| --- | --- |
| **Post Conditions** | **S**uccess alert box |

NON-FUNCTIONAL TESTING

1. Usability Testing

MODULE: Contacts

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | 51 | **Test Priority** | Medium |
| **Test Title** | Add New Company | **Description** | Test if New company can be added |
| **Test designed by** |  | **Test Design Date** |  |
| **Precondition** | Same as previous one | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Test Case 1 |  |  |
| 3 | Click on New Company |  | New company page opens |
| 4 | Enter details | Name: IBM  Tax Number: 12345  Click save | New company created |

|  |  |
| --- | --- |
| **Post Conditions** | **S**uccess alert box |

NON FUNCTIONAL TESTING

2. Compatibility Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | 68 | **Test Priority** | High |
| **Test Title** | Run Application on Chrome | **Description** | Test if application can run on chrome successfully |
| **Test designed by** |  | **Test Design Date** |  |
| **Precondition** | Same as previous one | | |
| **Dependencies** | Valid url | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Install Chrome |  |  |
| 3 | Open Chrome |  |  |
| 4 | Run Application on chrome | http://127.0.0.1:8000 | The application opens successfully |

|  |  |
| --- | --- |
| **Post Conditions** | **S**uccess alert box |

**After generating these test cases, we proceed to executing the tests and then generate the test cases execution report.**