CO421: SOFTWARE TESTING

Test Plan for the testing of Shuup E-Commerce Platform

Submitted by:
16CO129 Palak Singhal
16CO140 Sharanya Kamath

Submitted to:
Prof. Santhi Thilagam
Professor at CSE Dept, NITK

Date:
12th Sept 2019

Table of Contents

1.SCOPE

The objective of the testing is to find as many software defects as possible, or ensure that the software under test is bug free.

Login and Security

1. Check the login capability
2. Penetration testing to find out the security vulnerabilities in the application and API's.

Browser Compatibility

1. Test for browsers which support the application on the following platforms(Windows, Linux)
2. Check if the application lacks the support for early browsers or not.

Page Display

1. Check if the pages are displayed correctly.
2. Detect dead hyperlinks , plugin dependencies, font sizing etc
3. Detect poor page download time
4. Check if runtime error messages are displayed correctly.
5. Check if alerts are displayed whenever required.

Usability

1. Check if the design is non intuitive.
2. Check if the site navigation is poor.
3. Check if help-support feature is poor.
4. Check if catalog navigation for the customer is not proper.

Shopping order processing and purchasing

1. Check if the shopping cart functionality works properly.
2. Check if the order processing happens as desired.
3. Check if payment processing works properly.
4. Check if the order tracking feature is proper.

2. OUT OF SCOPE

The following features will not be tested by us during the testing phase due to lack of resources and expertise in the field.

Session Testing

1. Session Expiration: To test whether session timeout is properly enforced by the server.
2. To test whether the data stored in sessionStorage gets cleared when the page session ends.

Content Analysis

1. Check for misleading, offensive and litigious content in the webpages.
2. Royalty free images and copyright infringement.

Availability Testing

1. Test for denial of service (DoS) attacks.
2. Unacceptable levels of unavailability.

Back-up and Recovery Testing

1. Test how each component of the module will react in case of failure.
2. Check for backup failures.
3. Test the fault tolerance capability of the application.

Performance Testing

1. Check for performance bottlenecks in the application.
2. Check how much load can the application handle.
3. Check for how much stress can be handled by the website.
4. Check for scalability for the application.

3. ASSUMPTIONS

1. Availability of test environment:
   The technical infrastructure (hardware, software, and required files) that is necessary for the test should be controllable for the test management and available in accordance with the schedule. The development environment should be up and running.

2. Access to testing tools:
   The necessary testing tool and access to the tool are available and will be provided to the testing team.

4. SCHEDULES

Test step	**Start date**	**End date**	**Responsibility**
Information Gathering	**1/08/19**	**18/08/19**	
1. Go through application documentation	1/08/19	10/08/19	Read the documentation available about the application. Understand the application thoroughly, the technologies used and the components.
2. Summarize findings	10/08/19	18/08/19	Create a summary of the important findings after doing the above step so as to highlight the important aspects of the application.
Test Planning	**18/08/19**	**12/09/19**	
1. Build test plan	18/08/19	1/09/19	Create a test plan stating the scope and schedule, responsibilities, and testing tools which will be used for testing.
2. Define metric objectives	1/09/19	10/09/19	Specify the objectives, the features that will be tested and that won’t be tested
3. Review/approve plan	10/09/19	12/09/19	Review the test plan created

			above before starting with test case designing process.
Test Case Design and Development	**12/09/19**	**14/10/19**	
1. Design function/unit tests	12/09/19	17/09/19	Design functional tests for each component of the application and document the test case along with the input/ expected output etc
2. Design GUI Tests	17/09/19	22/09/19	Design GUI test cases for the application and document the test cases along with the input/ expected output etc
3. Define security/penetration tests	22/09/19	27/09/19	Design security/ penetration tests for the application and document the test case along with the input/ expected output etc
4. Review/approve design	27/09/19	1/10/19	Review the test cases developed before proceeding with test case execution.
5. Develop test scripts	1/10/19	12/10/19	Develop test case scripts for test execution.
6. Review/approve development	12/10/19	14/10/19	Review the test script development before starting off with the execution.
Test Execution/Evaluation	**14/10/19**	**4/11/19**	
1. Setup and testing	14/10/19	20/10/19	Execute the test scripts created after setting up the execution environment.
2. Evaluation	20/10/19	4/11/19	Evaluate the test case execution whether the test case passes or fails.
Summarize Report	**4/11/19**	**8/11/19**	
1. Perform data reduction	4/11/19	5/11/19	Select the important test cases to be mentioned in the report which bring out some defect in the application if any.

2. Prepare final test report	5/11/19	7/11/19	Prepare final test report based on the test cases selected above and perform a detailed analysis to be mentioned in the report.
3. Review/approve final test report	7/11/19	8/11/19	Review the test report before presenting it for final evaluation.

Test Documentation

Test documentation is documentation of artifacts created before or during the testing of software. It helps the testing team to estimate testing effort needed, test coverage, resource tracking, execution progress, etc. It is a complete suite of documents that allows us to describe and document test planning, test design, test execution, test results that are drawn from the testing activity.

1. Test Plan: A test plan is a complete planning document which contains the scope, approach, resources, schedule, etc. of testing activities.

2. Test Case: It is a group of input values, execution preconditions, expected execution postconditions and results. It is developed for a Test Scenario

3. Test Data: It is a data which exists before a test is executed. It used to execute the test case.

4. Test Summary Report: Test summary report is a high-level document which summarizes testing activities conducted as well as the test result.

5. ROLES AND RESPONSIBILITIES

Team Members:

Palak Singhal (16CO129)
Sharanya Kamath (16CO140)

Both the team members will be assuming the following roles and performing the responsibilities associated with each of these roles:

Test Manager Role:

1. Define the test activities for the application selected.
2. Check if the resources necessary for carrying out the testing are available like installation of required softwares etc.
3. To check if the testing is going according to the timeline specified.
4. Prepare the status report document of testing activities.

Test Engineer Role:

1. Read all the documentation available with respect to the application and the tools selected for testing and decide what in the application should be tested.
2. Decide a strategy to test the above mentioned tests.
3. Develop test cases and decide on a timeline to perform them.
4. Execute all the test cases and report defects, define severity and priority for each defect.

In addition to the above, the following responsibilities will be carried individually by the team members:
Palak Singhal:

- Prepare the environment for unit testing.
- Be incharge of penetration testing and make sure login and logout functionalities are up to mark.

Sharanya Kamath

- Prepare the environment for blackbox testing.
- Be incharge of UI testing and make sure each UI component is taken into consideration.

6. DELIVERABLES

The documents that will be produced throughout the testing phase are as follows:

1. **Generation of Test Cases for each type of testing**
   (September 11- October 14 2019)

   This will consist of precondition, input, validation point, output, post condition. Each of the test case which is to be carried out will be mentioned in detail in this document and will possibly help us to find defects in the application.

2. **Test Execution**
   (October 14- November 4 2019)

   Test cases mentioned above will be executed using the specified tools and the results will be recorded with respect to the behaviour of the application on their execution. The result will include what the test case intended to test and what is the result of the test case and the expected and the actual output.

3. **Test Report**
   (November 4- November 8 2019)

   A test report will be made which will contain the findings of the testing work and also the specific tests done alone with the relevant data used. Each and every test case will be explained along with why we chose a particular test case and how we went about testing the application against that test and the techniques used to carry out that test.

7. 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 will be using 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
- Pytest: It is a (unit) test framework that is ideal to make little tests. It is a framework that scales. This means that it is also capable to have more complex tests. It can run tests in parallel.

Problems:

1. **Test process problems** often occur when testing and engineering processes are poorly integrated. Testing may not be adequately prioritized so that functional testing may be overemphasized. Testing of components, subsystems, or the system may begin before they are sufficiently mature for testing. Other problems include inadequate test evaluations and inadequate test maintenance.

2. **Test tools and environments problems** include an over-reliance on testing tools. Often, there are an insufficient number of test environments. Some of the test environments may also have poor quality (excessive defects) or insufficient fidelity to the actual system being tested. Moreover, the system and software under test may behave differently during testing than during operation. Other common problems are that tests were not delivered or the test software, test data, and test environments were not under sufficient configuration control.

8. TOOLS

1. **Selenium**

   Selenium is an open-source tool that automates web browsers. We plan to use selenium-python that lets us write test scripts in python.
   Selenium Webdriver then executes these scripts on a browser-instance on our device.
   We will perform functional tests using Selenium.

2. **Lambda Test**

   LambdaTest is a free browser compatibility testing tool in the cloud. It is one of the best tools to ensure the streamlined functioning of your web applications on almost any desktop and mobile browser available today. This tool lets you test your applications on real browsers running on real operating systems and machines.

3. **Arachni**

   Appropriate for both penetration testers and admins, Arachni is designed to identify security issues within a web application. The open source security testing tool is capable of uncovering a number of vulnerabilities, including invalid redirect, local and remote file inclusion, SQL injection and XSS injection.

9. DEFECT MANAGEMENT

Defect management can be defined as a process of detecting bugs and fixing them.
The process of defect management usually includes four steps.

1. **Defect detecting.** We have already mentioned that it can be conducted either by the team of developers or by the users. Regardless of the type of testing, its main goal is to detect all bugs in the final product or its part.

2. **Formulation of bug reports.** These are the documents that include all necessary information about certain bugs. Usually, they contain data on the type of bug, and the possible ways of its correction.

3. **Bug fixing.** After the bugs are fixed, they should be tested once more to make sure that the software works properly.

4. **Bug list is created.** This is the document that contains information about all bugs that occurred during the project's performance. The team often uses the bug list because similar bugs' occurrence is not rare.

Reporting the defects

- As the application chosen by us (Shuup E-commerce platform) is an open-source application, the source code of the development version is available on Github.
- Defects/bugs found in the application will be reported to the application owners in the form of issues on Github.
- In addition to issue creation, bug reports and amendments in the code will be sent to the developers in the form of pull requests on Github.
- The bug reports will include screenshots as well as readings recorded by the testing tools.

10. RISKS AND RISK MANAGEMENT

In software testing Risks are the possible problems that might endanger the objectives of the project stakeholders. It is the possibility of a negative or undesirable outcome. A risk is something that has not happened yet and it may never happen; it is a potential problem.

1.Schedule / Time-Related / Planning Risks :
These risks are related to running behind schedule and are essential time-related risks, which directly impact the delivery of the project. Some reasons for such risks may be:

- Incorrect Time Estimation, and consequently an incorrect project schedule
- Improper Resource Allocation
- Underutilization of Resources
- Superficial Understanding of Project Complexities
- Unexpected Expansion of Project Scope

2.Operational / Procedural Risks
These are risks which are associated with the day-to-day operational activities of the project. These could be due to any of the below reasons:

- Improper Process Implementation
- Conflicting Priorities
- Lack of conflict resolution / team spirit
- Lack of clarity in responsibilities

- Breakdown in communications
- Lack of sufficient training

3. Technical / Functional / Performance Risks

- If the software skips some key function that the customers specified, the users required or the stakeholders were promised.
- If the software is unreliable and frequently fails to work.
- If the software has problems related to a particular quality characteristic, which might not be functionality, but rather security, reliability, usability, maintainability or performance.

11. EXIT CRITERIA

It specifies the criteria that denote a successful completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development.
We will focus on the following exit criteria:

1. **Run Rate**
   Verify if all the tests specified have been run. We expect the run rate to be 100%.
2. **Requirement Coverage**
   Verify if the level of requirement coverage has been met.
3. **Severe Defects**
   Verify if there are NO Critical or high severity defects that are left outstanding.
4. **Pass Rate**
   It is the ratio between numbers test cases passed / test cases executed. Our goal is to achieve a high pass rate.

▣▣▣