Table of Contents

WebLoanApp Application

Product Line:	Loan Management
Release:	Moonlight
Product Manager:	Oleg Vertlib
Author:	Mamata Lama

Distribution List	
Feature Product Manager:	Oleg Vertlib
Project Manager:	Oleg Vertlib
Developer:	Igor Gudkov
AC (optional):	
Tech Comm (optional):	
Services (optional):	

Revision History

Date (mm/dd/yy)	Author	**Version**	**Description**
09/08/2022 1:22 PM	Mamata Lama	2.01	First Update
10/08/2022 3:15 PM	Mamata Lama	2.02	Update based on requirement change
11/08/2022 11.30 AM	Mamata Lama	2.03	Update after review

Review

Date	Author	Version	Reviewed By
2022-08-15	Mamata Lama		Igor Gudhov

Related Documentation

Functional requirements for WebLoanApp Application are available on:
http://webloanappsoft/sites/PM/Features.aspx

WebLoanApp Application Internal Design document can be found on http://webloanappsoft/sites/RD/oroject.aspx in the X project Introduction

The scope of this document is to provide testing framework to fully test WebApp software. The test cases will be created and used to test an application as a part of a smoke, system and regression test.

1 Introduction

1.1 Functional Overview (Description of the product)

Loan Prequalifier consolidates many different functions of loan management processes into a single integrated solution. WebLoanApp is a digital platform that helps add users and customers to the database for using the loan services from a bank. Furthermore, the process involves calculating interest rates and adding more users from different locations depending on where the admin is located.

1.2 Current Release Scope

The Test Plan is designed to prescribe the scope, approach, resources, and schedule of all testing activities of a bank.

The plan identifies the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

1.3 Risks

This software supposes to be compatible with different web browsers such as Google Chrome, Safari, Mozilla Firefox, MS Edge, and many Open-Source Browsers. However, due to time constraints compatibility testing with the latest release of Chromium and Vivaldi won't be conducted. That creates a risk of not identifying potential compatibility problems.

1.4 Test Schedule

System familiarization 20/11 – 21/11/2022
System Test 22/11 – 28/12/2022
Documentation Test 28/12 - 29/12/2022

Test Schedule is comprised of major activities Test Planning and Test Execution:

1.4.1 Test Planning

Assumptions:

Timeframe required for requirements and design analysis = 10% of the Time to Write Test Cases

Timeframe for writing Test Plan = 10% of the Time to Write Test Cases

Timeframe for design and writing test cases:

number of builds = 8
Number of test cases = 9
number of testers on the project = 1.
Test Case time = 7-10 minutes per bug (0.12 – 0.16 hour)
So, Average Test Case time = 0.14 hour

Calculation:

Total Timeframe to write Test Cases per build = Total number of test cases * Test Case Time = 9 * 0.14 hour = 1.26 hours per build

Timeframe for creating test planning materials = 10% of the Time to Write Test Cases

So, Total Test Planning Time = 1.3 * Total timeframe to write Test Cases = 1.3 * 1.26 = 1.64 hours

1.4.2 Test Execution

Assumptions:

number of builds = 8
number of bugs = 50
number of test cases = 9
Test Case time = 7-10 minutes per bug (0.12 – 0.16 hour)
So, Average Test Case time = 0.14 hour
number of testers on the project = 1
estimate time to report a bug = 10 min (0.17 hour)

Calculations:

Total timeframe to execute test cases = Number of builds * Total number of test cases per build * 0.10 hour
= 8 * 9 * 0.14 hours = 10.08 hours

Timeframe to report bugs and verify fixes = 2 * 0.17 hours * 50 = 17 hours

Now,

Total Test Execution Time = 1.5 X (Time to Execute Test Cases + Time to Repot Bugs And Verify Fixes) = 1.5 * (10.08 + 17) = 40.62 hours

Total Testing Time = Total Test Planning Time +Total Test Execution Time
= 1.64 hours + 40.62 hours = 42.26 hours

Testers usually can only write up to 6 hours test cases per day for preparing status reports, attend meetings, emails, and other communications. So, 42.26/6 = 7 days approximately to do the complete testing

1.5 Total Resource Estimates

Total Conduct Time: 7 days

Total Personnel: 1 person

1.6 Test Strategy

The test strategy consists of a series of different tests that will fully exercise the loan management system. The primary purpose of these tests is to uncover the system's limitations and measure its full capabilities. A list of the various planned tests and a brief explanation follows below.

1.6.1 Smoke Test

The smoke test will be performed daily on each new build to define if it stable enough for further testing

1.6.2 Unit Test

The unit test will be performed once a week to determine which statements have been executed at least once. This will include temporary programs, namely stubs and drivers to enable testing to check if the users module and customers module work together in the system.

1.6.3 System Test

The System tests will focus on the behaviour of the loan management system. User scenarios will be executed against the system as well as screen mapping and error message testing. Overall, the system tests will test the integrated system and verify that it meets the requirements defined in the requirements document.

1.6.4 Documentation Test

Tests will be conducted to check the accuracy of the user documentation. These tests will ensure that no features are missing, and the contents can be easily understood.

1.6.5 Bug Reporting

All bugs should be reported and tracked using the company Defect Tracking System. Triage meetings to assign priority should be held daily during the test execution stage.

1.6.6 Entry Criteria for Test Execution

- All Functional Design documents have been signed off
- All Test cases and other test supporting documents have been completed and approved
- All test data requirements have been identified and test data has been mapped to the test cases
- Test environments have been setup as per the application requirements
- Test management tool has been setup with required accesses to all testers

1.6.7 Exit Criteria for Test Execution

- All Test scripts/cases need to be run at least once
- All defects irrespective of the severity has to be reported to the stakeholders
- All defects with severity Critical, High, Medium need to be fixed and retested successfully. (NOTE. All minor/cosmetic defects can be deferred to be fixed in later stages)
- All defects need to be moved to terminal stages such as Closed, Non-Defect, and Deferred. No defect has to be in any stage other than the terminal stages.
- All regression defects fixed and retested
- Passed to a planned ratio of the tests should be at least 95%
- All Functional Design documents have been signed

1.6 Tools

The following testing tools will be utilized during the project:

- Elementool Defect Tracking System
- TestLink Test Management Utility
- UFT automation tool for functional and regression testing
- WebLOAD automation tool for performance and stress testing
- LAN Gigabit and an internet line with the speed at least 5 Mb/s

1.7 Environment Requirements

1.7.1 Workstations

3 IBM compatible PCs (one with Win10, one with Win11, one with LINUX/Ubuntu), 1 MAC

3.5 GHz processor (minimum)

IE, Mozilla Firefox, Google Chrome, Opera, Safari

8 Gb RAM

512 Gb Hard Drive

A network-attached printer

2 Test Coverage

The following areas will to be tested:

Install/Uninstall

Help

Print

Graphical User interface

Windows integration

Hardware compatibility

Software compatibility

3 Test Cases

Provided separately

Pareto Principle

The Pareto principle (also known as the 80/20 rule) is a phenomenon that states that roughly 80% of outcomes come from 20% of causes, asserting an unequal relationship between inputs and outputs. This principle serves as a general reminder that the relationship between inputs and outputs is not balanced. The Pareto Principle is also known as the Pareto Rule or the 80/20 Rule.

The Pareto Principle can be applied in a wide range of areas such as manufacturing, management, and human resources. For instance, the efforts of 20% of a corporation's staff could drive 80% of the firm's profits. The Pareto Principle can be applied especially those businesses that are client-service based. It has been adopted by a variety of coaching and customer relationship management (CRM) software programs.

It can also be applied on a personal level. Time management is the most common use for the Pareto Principle, as most people tend to thinly spread out their time instead of focusing on the most important tasks. In terms of personal time management, 80% of your work-related output could come from only 20% of your time at work.

History

The Pareto Principle was named after Italian economist Vilfredo Pareto in 1896. Pareto observed that 80% of the land in Italy was owned by only 20% of the population. He also

witnessed this happening with plants in his garden—20% of his plants were bearing 80% of the fruit. This relationship is best mathematically described as a power law distribution between two quantities, in which a change in one quantity results in a relevant change into the other.

This phenomenon also goes by a couple of different names:

- Pareto principle
- The 80/20 rule (most common)
- Law of the vital few
- Principle of factor sparsity

How Pareto Principle can be applied to QA and testing?

Pareto principle/analysis is a technique used for business decision-making, but which also has applications in several different fields of quality control, analysis, and testing. It is based largely on the "80-20 rule." As a decision-making technique, Pareto analysis statistically separates a limited number of input factors—either desirable or undesirable—which have the greatest impact on an outcome.

A Test Case is an algorithm that should be performed during testing. The Test Case looks like a textual description of what steps must be done. It is written by a manual testing specialist based on User Stories and according to the Software Requirements Specification Document.

In custom software testing, test cases are divided into two types:

1. The positive test cases to check whether the software works without bugs in popular and obvious scenarios of behavior, which 80% of this software users reproduce.
2. The negative test cases to check whether the software works without bugs in the less possible scenarios, which 20% of this software users reproduce.

If a QA specialist is trying to run both positive and negative test cases for every recently developed feature simultaneously, a programmer is waiting for the results from the tester. What if there is 1 tester for every 4-5 programmers and all of them have just sent some recently developed features for testing? What about more complex projects with 2 testers and 8-10 programmers?

As a result, all developers are just waiting for the feedback from testers. It causes low productivity and client’s dissatisfaction.

Pareto analysis is premised on the idea that 80% of a project's benefit can be achieved by doing 20% of the work—or, conversely, 80% of problems can be traced to 20% of the causes. Pareto analysis is a powerful quality and decision-making tool. In the most general sense, it is a

technique for getting the necessary facts needed for setting priorities. Here are some scenarios relevant to QA and testing where Pareto analysis might be applicable:

- Try to sort out the defects according to their causes, not consequences. It means you should not group the bugs that cause the same results. It is better to group the issues that occur in the same module.
- Collaborate with developers to find out new ways to group the issues. For example, they may use the same underlying library for the modules where most of the bugs have been found.
- Spend enough time and effort on identifying the problem areas in code rather than finding random bugs.
- Prioritize test cases to start with the most critical ones.
- Analyze the users' feedback. It will help to identify the risk areas as well.

Conclusion

When there seems to be too many options to choose from or its difficult to assess what is most important within a company, Pareto analysis attempts to identify the more crucial and impactful options. The analysis helps identify which tasks hold the most weight as opposed to which tasks have less of an impact. By leveraging Pareto analysis, a company can more efficiently and effectively approach its QA and testing process.

References

Kenton, W. (2022). Pareto Analysis. Retrieved from https://www.investopedia.com/terms/p/pareto-analysis.asp

Laoyan, S. (2022). Understanding the Pareto principle (The 80/20 rule). Retrieved from https://asana.com/resources/pareto-principle-80-20-rule

Vasylyna, N. (2011). Pareto Principle in Software Testing - QATestLab Blog. Retrieved 16 August 2022, from https://blog.qatestlab.com/2011/02/25/pareto-principle-in-software-testing/#:~:text=Quality%20Assurance%20(QA)%20is%20one,%25%20and%2070%25%2C%20etc.