Software Testing Plan

Simpply

**Team members**

Mustafa al Salihi

Michael Yeaple

Jeremy Shulman
Curtis Burtner

**Document History and Distribution**

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Revision #** | **Revision Date** | **Description of Change** |
|  0 | 11/17/14 | Document created |
| 1 | 12/17/14 | Document Modification |

# Table of Contents

[Table of Contents](#h.1fob9te)

[1. Introduction](#h.3znysh7)

[1.1 Objectives](#h.2et92p0)

[1.2 Testing Strategy](#h.tyjcwt)

[1.3 Testing deliverables](#h.3dy6vkm)

[1.4 Definitions and Acronyms](#h.17dp8vu)

[2. Approach](#h.3rdcrjn)

[2.1 Performance Testing](#h.26in1rg)

[2.2 Acceptance Testing](#h.lnxbz9)

[3.  Pass / Fail Criteria](#h.3ardvssw0y2g)

[3.1 Approval Criteria](#h.nd2lf3s0j0p6)

[4. Testing Process](#h.utkaqg3oi9dz)

[4.1 Responsibilities](#h.35nkun2)

[4.2 Resources](#h.1ksv4uv)

[5. Environmental Requirements](#h.2jxsxqh)

[5.1 Software](#h.z337ya)

[5.2 Tools](#h.3j2qqm3)

[5.3 Risks and Assumptions](#h.1y810tw)

#

#

# 1. Introduction

## 1.1 Objectives

Over the course of software development cycle, we aim to perform black box and white box against Simpply application. In this document we have identified the focus of our testing and strategies to test them with. We have assigned responsibility for the various features and team members will devise more detailed test plans for their assigned feature. These tests are expected to be completed at the end of each iteration. This Software Test Plan (STP) will be a living document; over the course of the project, updates will be made to the document as needed. This document will describe the scope, approach, resource, and schedule of all testing activities. The document will also describe what features will be tested, the types of tests that will be performed, the schedule for testing, and the risks associated with the plan.

## 1.2 Testing Strategy

An effective testing strategy includes automated, manual, and exploratory tests to efficiently reduce risk and tighten release cycles. Tests come in several flavors:

* Unit tests validate the smallest components of the system, ensuring they handle known input and outputs correctly. Unit test individual classes in your application to verify they work under expected, boundary, and negative cases.
* Integration tests exercise an entire subsystem and ensure that a set of components play nicely together.
* Functional tests verify end-to-end scenarios that your users will engage in.

## 1.3 Testing deliverables

1. *Feb/ 12 / 2015*
2. *March/ 12/ 2015*
3. *April/9/ 2015*

## 1.4 Definitions and Acronyms

*As we encounter new terms, this section will be updated through the course of testing.*

STP - Software Test Plan

# 2. Approach

Our team’s testing approach is to perform test driven development; each new feature begins with a test case. Unit tests will be self-contained (using mock objects) and will not reach out to any external resources. Developers will perform integration testing continuously throughout the iteration (every time a developer pulls code and commits code) as well as at the end of the iteration. Post-deployment we will retest functions that are integral to the use of the application. For GUI testing, we will perform black-box testing on the application against our wireframes. We will perform heavy dynamic testing. Integration test should have two paths--working and not working.

The GUI will be tested to ensure that using various combinations of the programs features do not conflict with each other. Use cases will be developed from the user stories and combined together along with basic GUI functions such as window resizing, scrolling, etc. Output will be tested against the spec as well as general expectations for the system’s GUI functionality.

Input will be tested by feeding good, flawed, and corrupt keyboard characters in various formats.

User preferences will be tested by applying boundary value analysis, and acceptable input ranges will be identified for each preference. Boundary values for each setting will be tested by testing the engine’s output and functionality related to the preference, error messages, or console output. Comprehensiveness is clearly limited since the number of possible inputs to most settings is large.

## 2.1 Performance Testing

General business expectation is 0.1 second response times in order for the user to feel that the system is reacting instantaneously.

## 2.2 Acceptance Testing

Acceptance testing will be carried out by the users of the software during the end of the iteration demo. A constantly running latest release version of the application will allow users to toy with the system and come up with better feedback.

# 3.  Pass / Fail Criteria

**Acceptance test failure:**

Fails when application does not work as specified by a specific user story, even if that specific combination of features was not explicitly documented.

If invalid input is accepted and the program functions as if the input was not entered it is an indication of failure.

If any of the acceptance criteria is not met, the test can be considered have failed.

**General failure:**

Exception thrown; crash exhibited; interface elements not displayed according to basic system expectations, jumping around, or just plain wrong.

## 3.1 Approval Criteria

Upon completion of the testing phase, bugs will be submitted to TFS and assigned to a developer to address them.

# 4. Testing Process

 Evaluation of test results will be left to the team and their Coach.  Our results will be intended to improve our testing strategy as well as the application.

## 4.1 Responsibilities

|  |  |
| --- | --- |
| Jeremy Shulman | Nunit Tests |
| Mustafa al Salihi | Integration Testing |
| Michael Yeaple | Acceptance Testing |
| CURTIS BURTNER | GUI Testing  |

## 4.2 Resources

Each Tester will utilize their own self provided testing environment to perform tests specified in this document. Additionally, VMs may be utilized if the Tester does not have the necessary operating system on hand.

# 5. Environmental Requirements

## 5.1 Software

We will test basic GUI functionality on Internet Explorer 10 and 11.

## 5.2 Tools

* Visual Studio
* NMock
* NUnit
* NSubstitute

## 5.3 Risks and Assumptions

 The risks involved in this testing plan include: Time constraints on the completion of this document and all testing detailed in this document, the knowledge required to perform all needed tests on the GUI and respective details outlined above, and the availability of testers. Furthermore, if there is a crash in the application somewhere, apart from being a fault, it could delay the testing process.