Software Test Plan Document

Date: 12/7/2015

Version: Release 2

Team: TEAM-C-TeamSquad

1.0 INTRODUCTION

1.1 Objectives

The tests will be performed through using the Django application to be testing multiple cases. These cases are highlighted in the TestCaseTracker. New cases will be added as errors come up and use cases change. This document is meant to provide an overall idea of what tests will occur and how our team will approach testing our Django application.

1.2 Testing Approach

The specific test cases for each of the approaches listed in Clause 4 of this document are identified in the document "Test Case Tracker Document."

The testing approach taken involved checking to see whether external cases were handled for each functionality after the test for it's original functionality was operable.

The modules that needed testing, were ones involving the functions in views as they carried out most of the functionality.

The approach taken to select features were conducted through trial and error. If there were features needed, it was all up to the team to check whether it would meet the standards of the customer.

Certain features that did not need testing were based on the constraints given to us, which involved requiring certain features and functions to be functional for the specified release of the deliverable.

The technical approach involved making cases where the system would crash and correcting and finding solutions for these cases. While some members worked on testing the functionality, other members would continue to work on testing different aspects of the system such as testing the quality of the web-page that was displayed to customers.

Even though each member had an individual role, they all took part in each others role when certain things were in question. So at the end of the product delivery, each member fundamentally had experience in managing the different aspects of the software. Responsibility was held solely to the each member who managed a certain role and each issues were consulted with these members and solutions were produced.

1.3 Responsibilities

The responsibility of our team as a whole is to not only test each use case on an individual bases but also to be able to come up with new test and therefore new test cases. Each member of the team has their own responsibilities, but also providing the test coordinator with different tests in their mind they believe are important will benefit the team in finding new approaches to new problems. The main responsibility is the test coordinator in testing the program for the main use cases.

1.4 Resources

The tools that will be utilized is the test case coordinator going through and manually testing each use case. This is the best way to analyze individual problems as well as seeing what needs to change in the django program. Individuals that can test will also be other team members as they utilize the application, they will be able to giving additional tests that the test coordinator should do.

1.5 Risks and Assumptions

The most important constraint is time. The risks are not doing important tests that are essential to the program, but are not seen by the test case coordinator or any of the team members. This could be anything from security to testing the basic functionality of the program. Anything that a user might attempt to do either by mistake or on purpose needs to be analyzed ahead of time. The approach to documenting our tests can all be seen in the test case tracker document. This will allow for anyone on the team to see if their specific test was done.

2.0 FEATURES TO BE TESTED

- Django web framework
  - input and output from forms and database
- Applications of different types of users
  - users with different permissions may only access certain attributes

3.0 FEATURES NOT TO BE TESTED

- Different users outside of the main requirements

4.0 SPECIFIC APPROACHES AND STRATEGIES

4.1 Unit Testing - Component Testing

Our unit testing strategy focuses heavily on the technical capabilities of our forms and models. We feel our acceptance tests sufficiently cover the user functionality aspect, as well as being much more difficult to write unit tests for.

We also plan to factor our unit testing into our regression tests that we will do after any significant changes.

Refer to the Unit Test Sheet in the Test Case Tracker Document for further details.

4.2 Integration Testing

We will use a bottom up integration test approach, given the time to do so. This will make it so that all of our lowest level apps can be tested individually before we merge them all.

Refer to the Integration Test Sheet in the Test Case Tracker Document for further details.

4.3 Acceptance Testing

Our acceptance testing will be done primarily by third party testers, such as the SWEN-261 testing and our fellow cross team testers. We also plan to make it the center of our formal R2 presentation. In terms of implementation, our acceptance tests will follow very, very close to the

Refer to the Acceptance Test Sheet in the Test Case Tracker Document