**Software Testing Process**

Developers Inc.

**Assignment # 2**

**Comp3520**

Jagjit Singh Bilkhu - T00533766

Fahad Murtaza - T00527466

# **Table of Contents:**

**Table Of Contents: 2**

**Executive Summary: 3**

**Testing Process : 4**

Testing Tools and FrameWorks: 4

Manual Testing during Sprints: 5

Automation Unit and GUI Testing during Sprints: 5

User Testing: 6

Acceptance Testing: 6

**Conclusion: 7**

**References: 8**

# **Executive Summary:**

Developers Inc. our company focuses on developing and delivering native mobile applications to provide cost-effective and robust solutions to the needs of our customers from different business fields. To develop and implement the solutions, we focus on a combination of scrum agile development with bits of extreme agile programming. This allows us to break down the projects into smaller tasks and accomplish them based on their priorities in individual sprint cycles. It also allows us to have customers involved for validation and verification of the progress made in development.

# For testing and validation, we will initially focus on executing manual tests on the software during the initial sprints and development cycles. During the implementation sprints, we will also write unit tests using JUnit to ensure that the individual units operate as required. Once the project has grown and developed over multiple sprints, we will begin automating some of the use cases focusing mainly on those which are repetitive and on features that may not be changed a lot. For the automation, we plan to use Appium testing framework to write and execute automation GUI testing on the applications.

# **Testing Process:**

Our testing process begins with designing manual use cases from the user stories provided. This is done before the implementation of the feature/user story begins, and are further added or changed as new features and changes are requested. We will use JIRA integrated with a test case management tool such as QASymphony to record use case plans, use cases, execution results, bugs and errors and also changes requested.

During each sprint, we will have multiple build cycles (spanning 1-2 days) with changes/additions made in each cycle after which test cases will be run. Features and defects with a high priority will be fixed/implemented and verified first in each sprint and then implementation of new features or minor bugs will be done in the upcoming build cycles. Use cases will be selected in each sprint for execution and will be run multiple times in that sprint.

Some of the testing tools and frameworks we use are:

## **Testing Tools and Frameworks:**

1. JUnit: JUnit is a unit testing framework for the Java programming languages. It has other variations such as NUnit for C# and CPPUnit for C ++ programming languages. The latest release is JUnit 5. Junit would be used to test Native android apps and can also be integrated with Appium for GUI automation testing.

1. Appium: Appium is an open source test automation framework for use with native, hybrid and mobile web apps. It supports iOS, Android, and Windows apps using the WebDriver protocol to automate use cases and do GUI testing by accessing DOM like objects.

1. Jira: Jira is a defect, issue and requirement software designed specifically to work in an agile process. We will use this to keep track of the sprints, build cycles, defects, requirements/user stories and also tasks pending. Software is built for every member of your software team to plan, track, and release great software

1. QASymphony: QASymphony is a test case management tool that allows creating use cases, test cases and also keeps track of the results of their executions and generates reports on the failures and successes in executing tests. It can also be integrated with automated tests to record and save their reports.

1. Jenkins: This is a continuous integration and delivery software to automate the process of building software and creating executables. It provides a pipeline to automatically have the builds ready for testing and releases.

## **Manual Testing during Sprints:**

Use cases will be grouped based on features and user stories and certain groups will be selected for execution in each build cycle based on the areas which were affected in development. The manual test cases will be run in each build cycle and both teammates will work interchangeably to code and test such that both can understand the changes that are made and how they affect the functioning of the application.

As one person does the implementation, the other would be executing the major use cases manually on the previous build and will add more use cases for the feature additions and changes made. Any defects and errors found would be recorded in JIRA and assigned a priority. Depending on the priorities, the defects will be corrected and then verified in the next build cycle of the same sprint.

## **Automation Unit and GUI Testing during Sprints:**

While implementing the features in the application, Unit tests using JUnit will also be written to ensure that the functions/methods/units work correctly individually. The unit tests will be executed every time a build cycle is completed, and also will be combined to do some component testing to see how well the units work together. This automated unit tests will be run with help of Jenkins which only creates successful builds if all unit tests succeed.

We will begin setting up automation GUI testing after around three sprints of production are completed and the progress and stability of the applications is promising. This will be done by splitting our duties interchangeably such than one will begin setting up the automation framework to automate use cases and the other would be working on the application.

We will use Appium testing framework to automate the GUI testing process. Appium is a free open source framework similar to Selenium except that it can be used to test native mobile apps. Appium can be used to test both Android and iOS apps thus it wouldn't need writing test cases multiple times to cover the platforms. Appium also provides support for multiple languages such as Java, Ruby, Python and JS whereas its competitor, Calabash only provides support for Ruby. This would allow writing the test cases much simpler as we can choose the language best fit for us.

## **User Testing:**

The customers will be invited to perform tests and validations to ensure that the changes being made are correct and the application is working as expected.

Some alpha testing will be done before release with help of client representatives who will also be given the application to use to do some ad hoc and black box testing that would help uncover more perspectives to using the application. This would be done every couple of build cycles to allow us enough time to do some significant changes and also not do too many to have trouble restoring previous versions.

For beta testing, we would release the newer version of the application to a portion of the customers and get feedback and make any changes or fixes necessary.

## **Acceptance Testing:**

During the end days of a development sprint, we will divert focus from development to testing and executing regressive, ad hoc and user tests hand in hand with the clients to make sure that the build for release is stable and reliable. Any defects found during these tests would be handled right away if major and then released.

# **Conclusion:**

Our testing process helps validate and approve the application to help build a great product by having repetitive tests after changes have been made in the build cycles. We start with executing manual tests on the software during the initial sprints and development cycles. Even though it will make our testing process start off really slowly but it would really help us in building reliable test cases which we can later use for automate testing. Later on as development progresses, automated testing would be used to execute the most repetitive test cases. These tests will be run together with the manual testing for release and acceptance testing.

Using this testing process would be time consuming and slow but it would allow us to implement the features, and later these same features can be used to implement and setup the GUI testing. The automated testing would help us build a testing framework, which could be used in other similar applications.

This process also allows frequent validation from the clients since they are involved in testing the developed program regularly after a couple of build cycles. Also, since we have short build cycles, it would be easy to have find the causes of problems and what changes caused them.

**References:**

SmartBar - Test Complete. “what is a Test FrameWork?” - Testing Automation frameworks

<https://smartbear.com/learn/automated-testing/test-automation-frameworks/>

StackOverflow -“[advantages of Appium VS Calabash and Calabash VS Appium?”](https://stackoverflow.com/questions/39767341/for-mobile-automation-what-are-the-advantages-of-appium-vs-calabash-and-of-calab)- Sep 17 2017

<https://stackoverflow.com/questions/39767341/for-mobile-automation-what-are-the-advantages-of-appium-vs-calabash-and-of-calab>

Software Testing Help - “Most Popular Test Automation FrameWorks with Pros and Cons of each” - Oct 27 2017

<http://www.softwaretestinghelp.com/test-automation-frameworks-selenium-tutorial-20/>

ISTQB EXAM CERTIFICATION. “ What is fundamental test process in software testing?” Performance Testing - August 27, 2017.

[http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing](http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing/)

Bitbar -”What's trending with mobile Test automation Frameworks” - [16 Feb 2017](https://bitbar.com/whats-trending-with-mobile-test-automation-frameworks/)

<https://bitbar.com/whats-trending-with-mobile-test-automation-frameworks/>