Testing Policy

Cache Money

September 2022

Contents

1 Introduction **2**
- 1.1 Purpose of Document . . . 2
- 1.2 Why is Testing required? . . . 2

2 The Testing process **2**
- 2.1 Testing Requirements . . . 2
- 2.2 Types of Tests to be carried out . . . 2
- 2.3 Unit Testing . . . 3
- 2.4 Integration Testing . . . 3
- 2.5 Non-Functional Testing . . . 3

3 Defect Categorization **4**

4 Defect Tracking and Reporting **4**

5 Testing Responsibilities **5**

6 Tools for Testing **5**

1 Introduction

1.1 Purpose of Document

This testing policy describes the overall approach that will drive the testing of the ReverseHand project being built by Cache Money. This document will provide a summary of the testing procedures used, minimum requirements for test coverage as well as what requirements need to be met.

1.2 Why is Testing required?

Testing is required to ensure the following in the project:

- Ensure the requirements of the project are met.
- Allow developers to have confidence that the resulting software product is working as it should.
- Detect any potential defects in the software which might result in severe losses or damage to individuals.

2 The Testing process

2.1 Testing Requirements

The following are the requirements that must be met in the project:

- All testing must be automated in a CI-CD pipeline(Continuous Integration - Continuous Development).
- There should be tests to ensure that each functional requirement from the System Requirements Specification is validated to be in working condition.
- No code can be added to the code base in the presence of any failing tests.
- Test cases should be inline with functional requirements as much as possible.
- A minimum of 50 percent for test coverage is required.

2.2 Types of Tests to be carried out

The following tests must be present in the project at all times:

- Unit Tests: Individual units of code are tested to verify they perform their actions correctly isolated from the rest of the units. Mock data must be used
- Integration Tests: These tests will verify that the components are interacting together as expected and no mock data must be used.

- Non-Functional Tests: These tests will verify important quality requirements are met. Includes but not limited to: Performance Testing, Usability Testing and Security Testing.

2.3 Unit Testing

The goal of the unit tests is to identify defects in and verify the functioning of testable modules and methods of classes. The tests will verify that individual components are working correctly.

The following should be used as a guideline for writing unit tests:

- Each of the control flow paths from if statements, loops and switch statements must be exercised during the tests.

  This means that every if branch, else branch and else if branch must be tested with values to trigger that branch of the program to verify it is functioning correctly.

  Loop conditions must be tested to verify they will execute on correct conditions and that they will be skipped when conditions are not met.

- Boundary value analysis should be performed where applicable. Test with values at the boundary as well as near the boundary and evaluate whether correct program execution is taking place.

The unit tests will be written by individuals that understand the functionality of the code well as test cases are derived mainly from the component. It is essential that any dependencies are mocked out and that the unit tests are named descriptively.

2.4 Integration Testing

It must be ensured that these tests are carried out with no mock data. The goal of these tests is to ensure that components are working as they are supposed to be with other components.

The scope of the Integration Tests should be as small as possible. This makes it easier to identify defects between the components which might otherwise not be picked up with a large scope.

The key areas of testing will be functionality, unit interoperability and compatibility.

2.5 Non-Functional Testing

Security Testing is the most important nonfunctional requirement that must be satisfied. The following must be used as a guideline:

- Ensure that all user passwords are hashed and not stored as plain-text.

- Before releasing the product to production ensure all logs which were used for debugging purposes are removed.
- Consult the OWASP Mobile Security Guide for an extensive guide to ensure adequate security for the project.

In performance testing the major area of interest is loading and response times. These should be kept below a minute always and preferably a few seconds at all times. Finally it is essential to perform usability testing of the software product as it is heavily user dependent. The following can be used as guidelines when performing the test:

- Identify the major parts of the system you want to test.
- Create a set of tasks to be performed on those parts.
- Define a standard for success.
- Create a script which will take not more than 10 minutes to complete.
- Find participants and conduct the test with.
- Evaluate the data received.

3 Defect Categorization

4 levels of severity are used to define the effect of a defect.

- Level 1 represents a critical bug that crushes the system and prevents the product from functioning. The lead tester should be notified immediately and the responsible parties must make it highest priority to fix the issue.
- Level 2 represents a bug that causes a specific part of the system to not function. There is however a workaround to keep the system working
- Level 3 the bug degrades overall quality of the system but can be easily fixed within less than an hour.
- Level 4 is a low impact bug that has little effect on the use of the system

4 Defect Tracking and Reporting

The following process is to be followed when defects are detected:

1. The tester is notified of the defects.
2. The tester helps developer fix the test.
3. The developer reruns the test.
4. If test fails, step 2 is started again else the defect is marked as solved.

5 Testing Responsibilities

A dedicated tester is assigned the responsibilities for developing and maintaining the tests in project. The rest of the team is obliged to assist the tester with any questions about their code that might be asked. It is also the responsibility of the developers to notify the tester when defects are detected. The urgency of such notification depends on the level of severity of the defect. For unit tests, if the developer can fix the failing test without the assistance of the tester they are encouraged to.

6 Tools for Testing

Below is a list of tools used to assist in the testing process:

1. GitHub Actions: This is used to automate the execution of the tests and validate new changes being added into the code base automatically.
2. Jest: This is used to perform unit tests and integration tests for various JavaScript code in the system.
3. flutter_test: This is a flutter library used to perform unit tests on Dart code.
4. integration_test: This is a flutter library used to perform integration tests on Dart code.
5. CodeCov: A test coverage generator to assist in automatically determining the test coverage of the project.
6. Flutter DevTools: This is used to perform performace profiling on the mobile application
7. MobSF(Mobile Security Framework): This is used to perform security testing on the mobile application.