# Test Plan and Cases (TPC)

**QuickShip**

**DEV-T**

**Trinh Nguyen (developer)**

**Emmanuel Mendoza (developer)**

**Victor Fateh (developer)**

**David Navarro (developer)**

**03/12/17**

# Version History

| Date | Author | Version | Changes made | Rationale |
| --- | --- | --- | --- | --- |
| 03/13/17 | DN | 1.0 | * Converted original template to test document. | * Initial draft for use with QuickShip |

# Table of Contents

[Test Plan and Cases (TPC) i](#_Toc219626969)

[Version History ii](#_Toc219626970)

[Table of Contents iii](#_Toc219626971)

[Table of Tables iv](#_Toc219626972)

[Table of Figures v](#_Toc219626973)

[1. Introduction 6](#_Toc219626974)

[2. Test Strategy and Preparation 7](#_Toc219626975)

[2.1 Hardware preparation 7](#_Toc219626976)

[2.2 Software preparation 7](#_Toc219626977)

[2.3 Other pre-test preparations 7](#_Toc219626978)

[2.4 Requirements Traceability 7](#_Toc219626979)

[3. Test Identification 8](#_Toc219626980)

[3.1 Test Identifier 8](#_Toc219626981)

[3.2 Test Identifier 10](#_Toc219626982)

[4. Resources and schedule 11](#_Toc219626983)

[4.1 Resources 11](#_Toc219626984)

[4.2 Staffing and Training Needs 11](#_Toc219626985)

[4.3 Schedule 11](#_Toc219626986)

# Table of Tables

[Table 1: Requirements Traceability Matrix 8](#_Toc219626987)

[Table 2: TC-01-01 Check report completeness 9](#_Toc219626988)

[Table 3: TC-01-02 Check report correctness 9](#_Toc219626989)

[Table 4: Testing Schedule 11](#_Toc219626990)

# Table of Figures

[Figure 1: <Figure Title> 21](#_Toc55215008)

### Introduction

As the QuickShip project grows, it is important that all features function to expectation. The vision behind QuickShip is to connect people who are present with each other through a fast-paced strategy game. As the name entails, games should be quick, which inherently means the app is stable. The focus of the testing is to ensure that the app delivers the vision behind QuickShip. The scope of testers will mainly revolve around the coders themselves, due to lack of resources. The testers will utilize methods of testing such as Black Box Testing, Unit Testing, and Incremental Integration Testing.

### Test Strategy and Preparation

Utilizing Agile inherently structures the testing schedule along with scrum meetings. Scrum meetings provide the perfect time to demo the functionality of the recently implemented feature. At this time, we will also allow the other coders to intentionally find bugs and to see if the app delivers the desired functionality. By utilizing the story points fundamental in Agile, we can also use those points to determine the priority of testing new features. The logic is simple, if a story was deemed difficult to complete, it is reasonable to prioritize testing said story over others. In order to keep track of test history, there will be a shared document for coders to keep a list of bugs found during testing. The test environment will be either through Android emulators or an Android device itself. This is made possible by Android Studio.

#### Hardware preparation

Hardware for testing purposes will either be the development machine or Android devices. Within most of our development machines will be an emulator to run Android Nougat and test our app. Android devices will be the most useful for testing, as most devices include required functionality such as Bluetooth. We will just need to ensure that the Android devices are updated to an acceptable version in order for our application to run.

#### Software preparation

As stated before, emulation of Android Nougat can be a method of testing the QuickShip application. However, since emulation does not give us access to some hardware dependent functionality, such as Bluetooth, we will mostly utilize emulation for architectural and GUI testing rather than testing gameplay. This testing will rely on the debugging tools provided by Android Studio.

#### Other pre-test preparations

As expected, all coders and testers should have their environment set up and functioning to expectation. There should also be a document or another form of record to keep testing history such as result and new bugs to fix. Developers should get their hands on a tester Android device, as said device would be the most useful for testing purposes.

#### Requirements Traceability

*Table 1: Requirements Traceability Matrix*

|  |  |  |
| --- | --- | --- |
| **Requirement ID** | **Verification Type** | **Test Case ID (if applicable)** |
| Game Initialization and Termination Test | Testing | TC-01 |
| Touch Input Simulation Test | Testing | TC-02 |
| Bluetooth Connectivity | Testing | TC-03 |

### Test Identification

#### Test Identifier

TC-01 Game Initialization and Termination Test

##### Test Level

Software Level Test

##### Test Class

More Test Class to be added.

GameInitialization.java – We plan to use this class to initiate the game and call the restart game method. We will monitor the memory after every re-initialization to make sure the memory usage remains the same on every restart.

##### Test Completion Criteria

We plan to pass in an argument on how many times the game controller will be restarted. It will log memory usage each time. We anticipate 100 re-runs should be sufficient

##### Test Cases

Table 2: TC-01-01 Game Initialization and Termination Test

|  |  |
| --- | --- |
| Test Case Number |  |
| Test Item |  |
| Test Priority | Should have |
| Pre-conditions | Working game loop code with thread ending identifiers (ie a gameRunning Boolean) |
| Post-conditions | No incremental increased memory usage on each successive re-initialization |
| Input Specifications | Create and run an instance of the java class within the activity startup code |
| Expected Output Specifications | Timestamp, Memory usage amount, re-initialization count |
| Pass/Fail Criteria | Test Pass when memory usage remains close to the same as when test first initialized. Test fail when excessive memory usage is reported on test completion. |
| Assumptions and Constraints | Other aspects of the game is correctly implemented and functional.  Must have core game developed before testing can be done. |
| Dependencies | None |
| Traceability | All results to be printed on Android Monitor console |

#### Test Identifier

TC-02 Touch Input Simulation Test

##### Test Level

Software Level Test

##### Test Class

MultipleTouchInput.java – This class will be used to simulate touch input multiple times. We plan to use this class to monitor performance and correct code execution.

##### Test Completion Criteria

We plan to add a randomizer that decides where on the screen to touch. Values should be returned such as where on the screen and what elements it activates. These are all logged.

##### Test Cases

Table 3: TC-02 Touch Input Simulation Test

|  |  |
| --- | --- |
| Test Case Number |  |
| Test Item |  |
| Test Priority | Should have |
| Pre-conditions | Each area of the screen has defined actions and returns a result |
| Post-conditions | No error messages. Each area touched returns an expected result |
| Input Specifications | Create and run an instance of the java class within the activity startup code |
| Expected Output Specifications | Timestamp, return value logged |
| Pass/Fail Criteria | Test Pass when no error messages are reported on completion and return results are accurate. Test fail when program crashes or incorrect result reported |
| Assumptions and Constraints | Other aspects of the game is correctly implemented and functional.  Must have core game developed before testing can be done. |
| Dependencies | None |
| Traceability | All results to be printed on Android Monitor console |

### Resources and schedule

Some immediate resources we have at our disposable include the coders themselves. It is likely, friends and acquaintances of said coders would be included as human resources by extension. Time varies between each person involved with testing, but we can assume a lower bound of 4 hours a week towards testing. Testing will usually take place on Fridays, since these are the days most available to majority of the coding team. Some testing may occur outside these hours. Budget will remain low since the development team is composed of college students.

#### Resources

The most needed resource for testing is people to test the product. This is where social connections of the coders will come into play, as it is easiest to recruit friends for beta testing. Preferably, these appointed testing duties have access to a physical Android device. In cases where we aim to test a specific function of the application, we will also include a list of instructions for the testers to follow with the intention of pushing the functionality through different test cases.

#### Staffing Needs

*<< Identify the stakeholders responsible for managing, designing, preparing, executing, witnessing, inspecting and resolving test items. In addition, provide the groups responsible for providing items to be tested. Specify test-staffing needs by skill level. Identify training options for providing necessary skills. >>*

#### Schedule

Table 4: Testing Schedule

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Test Identifier** | **Responsible person** | **Resources** | **Training needs** |
| *01/02/09* | *TC-01-01 to TC-01-04* | *John Smith* | *Report test data sets,*  *JUnit* | *N/A* |
|  |  |  |  |  |
|  |  |  |  |  |