**4stones**

**Software Quality Assurance Plan**

**Document History and Distribution**

1. Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Revision Date | **Description of Change** | **Author** |
| 1 | 10/21/2014 | Initial draft | Dancer |
| 2 | 11/7/2014 | Minor updates to §1 | Pallares |
| 3 | 11/11/2014 | Added items §2 | Pallares |
| 4 | 11/21/2014 | Added items §3–5 | Pallares |
| 5 | 12/4/2014 | Completed all sections | Pallares |
|  |  |  |  |

1. Distribution

|  |  |  |
| --- | --- | --- |
| **Recipient Name** | **Recipient Organization** | **Distribution Method** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

TABLE OF CONTENTS

[1 INTRODUCTION 1](#_Toc405505782)

[2 TEST ITEMS 2](#_Toc405505783)

[3 FEATURES TO BE TESTED 2](#_Toc405505784)

[4 FEATURES NOT TO BE TESTED 3](#_Toc405505785)

[5 APPROACH 3](#_Toc405505786)

[6 PASS / FAIL CRITERIA 5](#_Toc405505787)

[7 TESTING PROCESS 5](#_Toc405505788)

[8 ENVIRONMENTAL REQUIREMENTS 6](#_Toc405505789)

[9 CHANGE MANAGEMENT PROCEDURES 7](#_Toc405505790)

[10 PLAN APPROVALS 8](#_Toc405505791)

# INTRODUCTION

The purpose of this Software Test Plan (STP) is to produce a working tic-tac-toe game. The product test strategy.

## Objectives

The objective of this test plan is to ensure project artifacts are delivered on-time, satisfy user requirements, and are reliable.

## Testing Strategy

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. The application will be tested using white-box tests by the module developers. Black-box testing will be used to validate the output(s) of use-case scenarios. UI components will be tested manually by, both, developers and SQA team.

## Scope

Testing will be performed at several points in the life cycle as the product is constructed. Testing is a very 'dependent' activity. As a result, test planning is a continuing activity performed throughout the system development life cycle. Test plans must be developed for each level of product testing.

Trello will be used to collect, organize, and communicate proposed changes to the test plan. From Trello, updates will be scheduled. Any necessary, unscheduled changes identified during the project’s life-cycle will be submitted to Trello to be reviewed and approved by the team.

Version control will be handled with Git using GitHub as the repository service.

## Reference Material

* Requirements document
* Specifications document
* Software Detail Design document

## Definitions and Acronyms

Refer to “Specifications.docx” for Glossary of Terms.

# TEST ITEMS

* Requirements specification: Specifications.docx
* Design specifications: SoftwareDetailDesign.docx
* Features:

## Program Modules

* Application components:
  + White-box unit tests will be performed for every component (i.e.: classes, methods, and helper functions).
* User Interface:
  + Every feature of the UI will be manually tested to ensure the application operates as expected.
* Database:
  + Create/read/update/delete operations will be executed against the database via a REST API to confirm they function as expected.

## Job Control Procedures

Job control procedures are beyond the scope of this test plan.

## User Procedures

User documentation is beyond the scope of this test plan.

## Operator Procedures

To ensure the application can be run and supported in a production environment, a series of black-box tests will be performed on the final product.

# FEATURES TO BE TESTED

* Register player
* Login as registered player
* Play as guest
* Play vs another guest
* Play vs computer (three difficulties)
* Chose first player
* Open menu screen
* Menu: Navigate back
* Menu: View player history
* Menu: Start new game
* Menu: Restart game
* Menu: Settings (change background)
* Menu: Logout (if playing as registered player)
* Menu: Login (if playing as guest)
* Delete registered player

# FEATURES NOT TO BE TESTED

(Identify all features and specific combinations of features that will not be tested along with the reasons.)

Not all end-game states will be tested, as there are too many permutations of moves that will result in win, loss, or draw.

# APPROACH

## Component Testing

Developers will test the components on which they work whenever possible. Some components are tightly coupled to higher-level components making them difficult to unit test; these will have to be tested when performing system tests.

## Integration Testing

A set of test cases will be created to address the requirements and use cases identified during the requirements gathering and analysis. These tests will be performed manually via the user interface on a fully-integrated and complete system.

## Conversion Testing

Conversion testing is beyond the scope of this test plan.

## Job Stream Testing

The application will be tested on various platforms (e.g.: Windows, OSX, iOS, Android, etc.).

## Interface Testing

The set of test cases used for integration testing are also used for interface testing. Performing each test case will cause the user interface change the display. The software development team will use these test cases to ensure the display renders correctly.

## Security Testing

Security testing is beyond the scope of this test plan.

## Recovery Testing

Recover testing will not be performed, as data will be stored in an online database system that is maintained and supported by the service provider.

## Performance Testing

Performance testing will be done manually and without the use of profiling or analysis tools. Instead, developers and testers will navigate all aspects of the application. If any action is perceived as “taking too long” (generally more than a few seconds), the performance test will be considered a failure for the application.

## Regression Testing

To ensure changes to the system do not cause faults in other components, thorough regression testing will be conducted after every “major” change. Similarly, regression testing will be conducted when implementing and/or debugging modules.

## Acceptance Testing

In lieu of customer acceptance testing, volunteers will assume the role of a user (i.e.: player) of the system. These volunteers will be presented with the customer’s requirements. Using this information and their own opinions and observations, they will be asked if they feel the application satisfies the customer’s requirements. Additionally, they will be asked to provide any additional feedback that can be used to improve the system.

## Beta Testing

Beta versions will be released to the select consumers and individuals via TestFlight for iOS, Google Play and direct download for Android, and download via website for Windows and OS X.

# PASS / FAIL CRITERIA

## Suspension Criteria

If a fault is encountered that prevents further testing of a given test item, the test case executing will be suspended.

## Resumption Criteria

After the cause of the test suspension has been addressed, the test case execution will resume. If necessary, test cases related to the component(s) that caused the fault will be repeated.

## Approval Criteria

Test results must match the expected results defined in the test case for results to be approved.

# TESTING PROCESS

## Test Deliverables

A test report detailing the test cases and pass/fail results will be included in the project deliverables.

## Testing Tasks

* Develop test cases
* Develop scripts are steps for each test case
* Execute test scripts
* Record pass/fail results

## Responsibilities

E. Pallares is responsible for managing, designing, and preparing test activities. Every member of the software development team is responsible for executing tests.

## Resources

Each developer’s personal computer will be used to perform testing activities. Developers are responsible for performing component tests. E. Pallares is responsible for performing regression tests and integration tests. Every member of the software development team is responsible for submitting issues into the issue-tracking system.

## Schedule

* Week 9:
  + Develop test cases
  + Develop test scripts/steps
* Week 10 – 16:
  + Execute tests
  + Report issues
* Week 16:
  + Complete test report

# ENVIRONMENTAL REQUIREMENTS

## Hardware

* 1 computer running Windows 7 (or higher) with WiFi capabilities
* 1 computer running Mac OSX with WiFi capabilities
* 1 mobile device running iOS
* 1 mobile device running Android
* 1 mobile device running Windows Phone 7 (or higher)

## 8.2 Software

* Qt Creator
* GitHub for Windows
* GitHub for Mac
* Parse

## Security

The testing environment does not have security and asset protection requirements.

## Tools

Qt Creator will be used to capture debug statements. Also, Qt Creator will stop at specified “break points” to allow developers to “step-through” the code.

## Publications

* • Test Scripts.docx – lists all the test cases, entry/exit criteria, and test results.

## Risks and Assumptions

Some components are tightly coupled making it difficult to autonomously test them. These will be tested as a set of components.

Components designed to be executed via the user interface may not be able to be tested until the corresponding user interface components have been implemented. As a contingency, a dummy interface that simply executes user interface-dependent modules will be designed and implemented.

# CHANGE MANAGEMENT PROCEDURES

To amend the software test plan, the following change management process must be followed:

1. Propose a change to the development team.
2. The development team will discuss the proposal and reject it, accept it, or accept it with amendments.
3. If the proposal is accepted, it will be implemented.

# PLAN APPROVALS

|  |  |  |
| --- | --- | --- |
| **Name** | **Signature** | **Date** |
| Aaron Dancer |  |  |
| Ernest Pallares |  |  |
| John Molina |  |  |
| Thanh Dang |  |  |
| Hieu Vy |  |  |