**Tic Tac Toe Game**

**Software Quality Assurance Plan**

**Version: 1 Date: (05/01/2016)**

**Document History and Distribution**

1. **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision #** | **Revision Date** | **Description of Change** | **Author** |
| 1 | 02/10/2016 | Introduction. | Ibra |
| 2 | 03/01/2016 | Test items and features to be tested. | Afonso |
| 3 | 03/15/2016 | Features not to be tested and approach. | Alexis |
| 4 | 04/03/2016 | Pass/Fail criteria and Testing Prodcess. | Carrie |
| 5 | 04/21/2016 | Environmental Requirements, change management procedures, and plan approvals. | Eddie |
| 6 | 05/01/2016 | Last finishing touches. | Carrie |

1. **Distribution**

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

**TABLE OF CONTENTS**

**1.**

**INTRODUCTION**

[**2. TEST ITEMS**](#h.4d34og8)

[**3. FEATURES TO BE TESTED**](#h.35nkun2)

[**4. FEATURES NOT TO BE TESTED**](#h.44sinio)

[**5. APPROACH**](#h.z337ya)

[**6. PASS / FAIL CRITERIA**](#h.2bn6wsx)

[**7. TESTING PROCESS**](#h.3as4poj)

[**8. ENVIRONMENTAL REQUIREMENTS**](#h.1pxezwc)

[**9. CHANGE MANAGEMENT PROCEDURES**](#h.2p2csry)

[**10. PLAN APPROVALS**](#h.147n2zr)

# **INTRODUCTION**

(NOTE 1: THE SOFTWARE TEST PLAN GUIDELINES WERE DERIVED AND DEVELOPED FROM IEEE STANDARD FOR SOFTWARE TEST DOCUMENTATION (829-1998)).

*(Note 2: The ordering of Software Test Plan (STP) elements is not meant to imply that the sections or subsections must be developed or presented in that order. The order of presentation is intended for ease of use, not as a guide to preparing the various elements of the Software Test Plan. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content.)*

*The Introduction section of the Software Test Plan (STP) provides an overview of the project and the product test strategy, a list of testing deliverables, the plan for development and evolution of the STP, reference material, and agency definitions and acronyms used in the STP.*

**The Software Test Plan (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities. The plan must identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan**.

**1.1 Objectives**

The test plan followed on this project is based on the functional and non-functional requirements. The black box and glass box testing were implemented to test all the functionalities, described in pur specification document. Thes objective of this tasting phase allows us to make sure that all the modules in our programm work correctly whitin the modules and to verify that these modules interact with eachother smootly. A Test bug report is created from all stages of the testing process and diagrams are utilized to show the overall process.

**1.2 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 testing process starts at the moment the test cases for excution based testing are defined. At this stage we make sure that all the product feautures are traceable to the specifications document and are implemented as the Desing document described them. The tests executed check for the product to be realiable, to have correct output, to be able to hanlde exceptions, and check the overall behaviour of the program. The testing procedes with blak box testing and white box testing. At this stage the feautures of the program are tested first wihtout taken into account the codes, then the white box testing is perfomed to test the program by examinng the code lines. The statements of each line of code in each module is analized and tested to see the outputs created. Ater this, the integration testing is performed, which makes sure all the modules interact correclty. The top-down approach is utilized to show the logical modules first and the operatinal modules last.We made use of diagrams to show this part of the testing process.

Specific test plan components include:

* Purpose for this level of test, is to test the functionalities of the program.
* Robustenss, reliaiability, correctness, perfomance, and utility
* Features to be tested are the login feauture, the sign up featuture, the guest feauture, the reset password feauture, log out feauture.
* Features not to be tested
* Management and technical approach, Execution based testing is used, for non functional and functional. Glass testing is implemented for the code itself , integration testing to test modules and their interactions.
* Pass / Fail criteria of each feautured is specified under the bug report.
* Individual roles and responsibilities, Ibra Cisse-Team lead, developer and architect of the GUI interface. Eddie Aguilar-Sub team lead and architect of the single player mode (PvP). Jason Jense -Architect of the AI. Alfonso Euclides- Secretary, Alexis Franciosi- Secretary, Carrie Dumit- Tester and Secretary.
* Milestones, Getting the AI to work properly. To integrate all functions related to AI functionalities in an AI class, we created modules with high cohesion. Handle all the exceptions from the user scenarios. Test all scenarios by implementing branch testing. Getting the Database to work properly on any computer.
* Schedules, and Responsabilities
* Risk assumptions and constraints

**1.3 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.

After each testing phase on each module an feauture, updates were made to the code itlsef and to the UI to make sure all the feautures were traceable to our specifications and to the requirements of the client. More tests were performed in an iterative manner and changes were made to fix all the issues found. Each time a one of the developers made a change the newest verrsion was uploaded to Github, the verion control utlilized to track all changes, and the testers went back to test all the feautures that failed previously.

**1.4 Reference Material**

(Provide a complete list of all documents and other sources referenced in the Software Test Plan. Reference to the following documents (when they exist) is required for the high-level test plan:

* Project authorization,
* Project management plan
* Test and implementation ,
* Configuration management plan,
* Organization policies and procedures, and
* Relevant standards.)

**1.5 Definitions and Acronyms**

(SD) Specification Document

(DDD) Detailed Design

(TID) Testing and Implementation

# **TEST ITEMS**

(Specify the test items included in the plan. Supply references to the following item documentation:

* Requirements specification,
* Design specification,
* Users guide,
* Operations guide,
* Installation guide,
* Features (availability, response time),
* Defect removal procedures, and
* Verification and validation plans.)

**2.1 Program Modules**

The modules that need testing are as followed:

* mainTicTacToe
* LoginScene
* RegistrationScene
* ResetPassword
* GameMode
* DifficultyLevel
* playerGameOptions
* GameOption
* AIBoard

**2.2 Job Control Procedures**

White box testing and integration testing are perfomed to all modules in the program, this is to ensure that the scheduling between calls in all the functons of each module run smoothly. These two testing pocedures also allows to see the sequence of excecution of all the modules in and therefore allow to test when the calls to the different attributes are made.

**2.3 User Procedures**

The testing that is used on documentation would be functional and non functional testing. To test all the artifacts are traceable to previous stages of the testing process and are in accoradance to our specifications

**2.4 Operator Procedures**

The operator procedure was used to make sure that the application can run and is supportred in a production environment. To do so, the application was tested by creating a build release that was tested on linux, windows, and IOS environment to ensure that it can run properly in different types of environments.

# **3. FEATURES TO BE TESTED**

The main feautures to be tested in the game are on te main wondow. The sign up button, the login button, the exit button, the guest button. Then the second most imprtant feautures to be tested are on the sign up window, here the user enter its infomation and then the program saes it on the database. The login feauture is direcly linked to the signup because the information that is being inpute will be retrieved from the databased to test the login functonality. Finally the AI funcitonalities are tested, speficially to AI vs player and player vs player use cases.

# **4. FEATURES NOT TO BE TESTED**

The features not to be tested are the features that we identified as not critical for the program to functional. Features such as:

1. Help dialog boxes/ buttons
2. The “whats this” button on all dialog menus
3. Graphic rending of the game board on each individual system this program is tested on
4. sRGB color profiles
5. Font size and colors
6. Default window size
7. Application Icons

# **5. APPROACH**

The method used for testing is the Black Box which is an method of testing we use to examine the functionality of the application without looking into the code. Testing was a major part of the development of this application the team members responsible for testing took an average of a month and half on testing. we test for especification, requirements and the design. The test case were generated by first identifying the features that need to tested and a tesing them to ping point the faults features. A table with a test cases was create and for each test case we documented the expected behavior and the actual program behavior and when the program did not funcion as espected we wrote a explanatiion of what the program did wrong. in a table. For each test case, document the expected program behavior and the actual program behavior. The glass box testing was used to examine the the program structure and derives test data from the program code. During this test statement covarage on which we examine the program statement with minimal testing. Branch covarage testing was also used to check if all branch were working accordingly. The white box testing allowed us to identify error in hidden codes and spot line of code that were useless or redundant.

**5.1 Component Testing**

Component testing was used to test separately the components or modules of the program, it helps to test the logical interactions between the modules and to ensure that the perational modules output the correct results.

**5.2 Integration Testing**

Integraton in a top-down apporach is done as well by implementing different diagrams that show all the modules and their relationship with each other.

* Start game test
* Track user name test
* Change between modes test
* Choose level of difficulty test
* Check Score

A call is made based on the returned values to select and enter a difficulty level, then a transition from this mode to the start game mode is made.

**Check if board is full test:** The user gets a notice of completely full board after a function call is made to enter the check board mode. This will detect if all the squares are full and makes a call to a function dedicated to check the scores. The check winner mode is implemented at this point. The user is given a notice of who is the winner and is the option to start a new game. If the user decides to enter into play mode again, the same settings are kept. However, if the user decides not to play again, the exit game mode is implemented and the user would need to login or play as guest before starting a new game.

**GUI applicability test:** The user is able to easily adapt to the interface presented. A few modifications are made based on minor suggestions by the user after this test. The resizing, realignment, the integration of representative icons for buttons, and redesigning of windows is made to create a consistent look throughout all of the interfaces. With these modifications, the user has a better change to adapt easily to all of the modes and options presented.

**5.3 Conversion Testing**

All of the user’s statistics of wins, losses and ties are all stored in the database, however the user must back up and load a the working database onto a new systme format and load it into the project

**5.4 Job Stream Testing**

Currently the most stable build of the program is within the IDE that the user desides to compile it in, Visual Studio, and QT. Since this application was built on QT, migrating it over to Visual Studio will produce different results

**5.5 Interface Testing**

The program has been designed to be a cross-platform application, the most stable release build oustide of the IDE ( Visual Studio, QT) is the Windows executable file. Linux executable file is also availabe for the user to use, a version for Mac OSX has been developed as well.

**5.6 Security Testing**

User database contains all user statistics and passwords for the game, there is currently to way to expliot the user database conatining all of the users paswords. On every single instace of user login, users are verified and program runs through the user database and will then detrmine if the user login is correct. Whenever the user attempts multiple failed attempts, it will keep displaying that the login information is incorrect, this process will run indefinelty.

**5.7 Recovery Testing**

Whenever each update is released, the database that holds all the user information, is always reset upon each update. Whenever the is a major crash in the applicatoin, the datbase will reset all user information. However, the user can choose to backup the database before the program loses any data. The user then can load previously recovered database and load it into the project.

**5.8 Performance Testing**

Our game is simple and easy to use for any user.Our program shows a low probability of having an issue that wipes out our database.The communication is well established between the moduless. We handle exceptions by having a way to manage incorrect inputs and guide the user to the correct utilization of our product. Second guest user, the user is not given instructions to enter “guest” as the username but they receive an error message “Wrong username”.

**5.9 Regression Testing**

Whenever we added a new module or feature, we go back and check that nothing s broken and that communication betwen the mdules is correct.

**5.10 Acceptance Testing**

Acceptance Testing will be implemented when the final product is delivered to the client.

**5.11 Beta Testing**

**Bug report:**

Checking for Correctness on PvP, and PvAI, enter the Username and password should not be empty once submit is pressed

Checking for Correctness Detail: Select option on PvAI and PvP, icons are not displaying.

Checking for Robustness: After choosing the stone and inputting the username, once "Cancel" is clicked, the window is to be closed and program be able to handle a new game by resetting all previous input.

Checking for Utility- Change the window or give instructions to specify to type "guest" as the username.

Checking for Correctness- When username losses its value after the first play from either user or AI.

Checking for Enhancement-Too many files were being included everywhere and repetitively.

Checking for Utility -Database issues with different types of platform of software. Mac OS, Windows, Linux and other type of known software have been having issues to connect to the database.

Checking for Utility- If the close button is clicked, the variables are not reset. They remain within the memory.

Checking for Reliability-AI modes, the AI’s turn

Checking for Utility- On Guest vs AI easy mode, give option to prompt the user to play again or to quit the game.

Checking for Reliability-When canceling the request on password reset, state that the password has not been changed.

Checking for Reliability When clicking on forgot password, the first input fields (first name, last name, and user name) are verified.

# **6. PASS / FAIL CRITERIA**

The criteria to be used to determine whether each item has passed or failed testing is:

* Does the feature work to the user’s advantage?
* Are there any errors when implementing any of the product’s features

**6.1 Suspension Criteria**

The criteria used to suspend all or a portion of the testing activity is only if all members of the team can find no bugs to fix throughout the whole program.

**6.2 Resumption Criteria**

The conditions that need to be met to resume testing activites after suspension are:

* When we need to move on to another phase in the project.
* If any member of the team can find any bugs or issues in the product.
* If the client needs more requirements to be met.

**6.3 Approval Criteria**

The conditions that need to be met to approve test results are:

* All the test cases have been succesfully handled.
* All use cases have been succusfully handled.

# 7. **TESTING PROCESS**

The methods and criteria used in performing test activities are:

* Writing down every step taken
* Using QT software to test our product.

**7.1 Test Deliverables**

The deliverable documents from the test process is the Testing Plan and Implementation document.

**7.2 Testing Tasks**

The set of tasks necessary to prepare for and perform testing activities are:

* Have QT 5.5 installed or Visual Studio 2012, 2013.
* Have Microsoft Word or paper and a pencil.

**7.3 Responsibilities**

The groups responsible for managing, designing, preparing, executing, witnessing, checking, and resolving test activities include:

* Eddie Aguilar
* Carrie Dumit
* Alexis Franciosi
* EuclidesAlfonso

**7.4 Resources**

Execution based testing done by Carrie Dumit

Integration testing done by Alexis Franciosi

Glass Box testing done by Eddie and Carrie

Black Box testing done by Alfonso

**7.5 Schedule**

Test cases for execution based testing to test non functional and function requirments.

Test for reliability, robustness, utility and correctness.

The questions we have according to our test plan include:

· How easy is our product to use?

· How reliable is our database?

· Is the communication well established between all the modules?

· Are we having unacceptable results when utilizing valid inputs?

· How badly does our product react with incorrect use?

· How often are we updating the scores to keep track of the winner?

After this phase on the testing process and having generated a bug report, we went back to the inital state of testing to test each functionality again.

# **8. ENVIRONMENTAL REQUIREMENTS**

The properties of the test environment include:

* Communication between all team members.
* Testing supplies.
  + A Windows, Mac, or Linux computer.
  + QT 5.5 software.
  + Microsoft Word or paper and pencil.
* Time to complete the testing.

**8.1 Hardware**

Our product works on Windows, Mac and Linux and requires no network connection, so any computer is able to complete our test activities.

**8.2 Software**

The software requirements needed to complete our testing activates would be QT because that is where our product was programmed.

**8.3 Security**

The asset protection we have is that a username cannot be used more than once to protect our users from other players wanting to mess up someone’s score.

**8.4 Tools**

The special software tool our team used to employ the testing efforts was QT because that is where most of our programming took place. However, when needed, we used Visual Studio to make sure that our product worked on other software as well.

**8.5 Publications**

The documents and publications needed to support testing activities are our Testing Plan and Implementation document, Detail Design document, test cases for execution based testing, test cases for White Box Testing, and specifications

**8.6 Risks and Assumptions**

Significant constraints that our team encountered when it came to testing were mostly time constraints. We almost always had a way to test our product because it was always up and running. After every test we made, we needed to document what happened.

# **9. CHANGE MANAGEMENT PROCEDURES**

If and when the team needed to change any part of the test plan, we would meet with each other and discuss how the change should be done. However, our team didn’t need to change any part of our test plan since we made it in the beginning.

# **10. PLAN APPROVALS**

Ibra Cisse 05/01

Eddie Aguilar 05/01

Jason Jensen 05/01

Euclides Alfonso 05/01

Alexis Franciosi 05/01

Carrie Dummit 05/01