Software Test Plan Template

(See pages 253-254 for section/sub-section descriptions)

1. **General Information**
   1. Summary: The purpose of this document is to summarize the functions of the Battleship game designed for SE470 and tests to be performed on the application to confirm its operation.
   2. Environment and Pretest Background: This application is being developed as a class project for the SE 470 at Sanford-Brown College in December 2016. The testing will be performed in class and in the homes of the students/developers. This is a new application so has not been any prior testing on this particular application.
   3. Test Objectives: The testing goals include
      1. Requirement completeness and correctness
      2. Design consistency with requirements.
      3. Design correctness.
      4. Design consistency
      5. Functional correctness.
      6. Handling of exceptional and abnormal behavior (See SRS)
      7. Mitigate highest risks to the extent possible including
         1. Status of ongoing project training
         2. Status of documentation in the user areas
         3. Type and Adequacy of configuration management planning
         4. Basis for selection of programming and system software
         5. Status of project maintenance planning.
   4. Expected Defect Rates: No historical data exists for this group of programmers. Using the heuristic metric from the textbook for the course, error rates of 20-60 error per 1000 lines of code will be assumed.
   5. References
      1. See the Software Requirements Specification (SRS) Nov 2016 for list of requirements.
      2. See the Software Summary document (December 2016) for details on schedule, team members and other details on the project.
      3. See the Use Case Document (Dec 2016) for complete list of use cases for testing details.
      4. See the User interface mockups (Dec 2016) for each use case to help with functional testing.
2. **Plan**
   1. Software Description:

|  |  |
| --- | --- |
| **Inputs** | Button/User Interface Clicks via the user interface |
| **Outputs** | Changes to the User Interface via the inputs. Changes include color and textual changes to indicate the progress of the game. |
| **Functions** | The game consists of 6 main functions: Game Initiation, Game piece placement, Guessing Game Play (Turn taking), Game ending, Game Reset |

* 1. Test Team
     1. The test team consists of the 5 developers: Joshua Galbraith, Nathan Grove, Devon Jack, Thomas McElroy and Kenneth Raymond. Each tester will participate in all aspects of testing.
     2. The test manager is the Tony Bladek who also serves as the instructor.
  2. Milestones:
     1. Requirements Review: 12/06/2016
     2. Design Review: 12/13/2016:
     3. Unit Testing Complete: 12/20/2016
     4. Acceptance Testing and prototype release: 01/05/2016
  3. Budgets
     1. Budgets are covered in the class tuition and not a concern for this project. All resources are essentially ‘free’ for the purposes of this project.
  4. Testing (System Checkpoint)
     1. Schedule (and budget): See Milestones above.
     2. Requirements:
        1. Desktop computer provided by the college and laptop computers provided by the students will be used for testing.
        2. Projection equipment in the classroom will be used for requirements and design reviews.
        3. Visual Studio 2012+ will be used for both development and functional testing
        4. NUnit Visual Studio plugin will be used for Unit testing development and testing.
        5. The students listed in Section 2.2.1 will be available for all aspects of testing. Testing shall occur in the classroom and at the homes of the students as necessary.
     3. Testing Materials
        1. Testing shall require the Software Requirements Specification, Use Case Document, Class Diagrams, User Interface Document, and a User’s Manual
        2. Testing shall require a copy of the software to be produced as a separate executable or a executable within the Visual Studio Development environment.
        3. Test Inputs will be generated manually via the user interface. No separate test data is expected to be necessary.
        4. An NUnit plugin for Visual Studio shall be necessary for unit testing.
     4. Test Training:
        1. The testers/developers are being trained as the project is in development via the SE 470 text book “Effective Method for Software Testing Third Edition” By William E. Perry
        2. Testers will be briefly trained in the use of NUnit for unit testing via online tutorials and documentation.
     5. Tests to be Conducted (see below for details)
        1. Initiation Testing
        2. Ship Placement Testing
        3. Game Play Testing
        4. Game Over Testing
        5. Pre-mature Exit Testing
        6. Normal Exit Testing.
        7. Game Reset/Play Again Testing

1. **Specifications and Evaluation**
   1. Specifications
      1. Business Functions:
         1. **Initialization:** Each user is required to start the game and bring up the user interface. This includes initial drawing of the board, game pieces and controls on the user interface.
         2. **Ship Placement:** User must place their game pieces on their Allied Board (See SRS). They must be placed either horizontally or vertically with no overlaps and wholly within the board.
         3. **Game Play:** Each user will take turns guessing the location of the opponent’s ships on the target grid. This is done via the user interface. Hits (successful guesses) or Misses (unsuccessful guesses) will be noted by the user interface.
         4. **Game Over:** The first player to guess all of the locations of the opponent’s ships will be declared the winner. At this point the user can reset the play field and go back to the initialization phase or exit the game whereupon the game will exit.
      2. Structural Functions
         1. Design should follow the Class diagram from previous documentation (see Class Diagram).
         2. Designs should follow the User Interface Mockups from previous documentation (See User Interface Mockups) including placement and presence of controls.
      3. Test/Function Relationships
         1. **Initialization Phase**
            1. Start of the program via the Visual Studio Development environment.
            2. Start the program via the terminal (if available).
            3. Confirm the arrangement and completeness of the user interface according to the User Interface Mock-ups produced previously.
            4. Ensure that the user is able to exit the application at any point during this phase.
         2. **Ship Placement Phase:** 
            1. Test the placement of the ships for each player follows the constraints listed in the SRS and other documentation.
            2. Ensure that placement is correct and that the user cannot perform other operations typical with the game play until placement is complete for both players.
            3. Ensure that at any point during ship placement the user is able to exit the application and/or reset the game and start again.
         3. **Game Play:**
            1. Ensure turn taking is consistent and fair.
            2. Ensure that each user gets one and only one guess per turn.
            3. Ensure operations are only allowed within the correct portion of the user interface based upon the player’s turn. Ensure all other actions are ignored.
            4. Ensure each correct guess and incorrect guess are noted correctly in the user interface, both textually and graphically as per the design.
            5. Ensure that when a game piece’s location has been completely guessed, the user interface notes the event both textually and graphically.
            6. Ensure that at any point during game play, the user is able to exit the applications and/or reset the game and start a new game.
         4. **Game Over:**
            1. Ensure that when a player has guessed all the locations of all of an opponent’s game pieces, the user interface displays the end of the game and which player has won.
            2. Ensure that when conditions in 3.1.2.1.4.1 occur, that the user interface allows the user is able to reset the game interface and play again and/or exit the game and the user interface disappears.
      4. Test Progression:
         1. Tests can proceed in the order listed above for each section (i.e. Initiation, Place Ships, Game Play, Game Over). If the game is exited prematurely, tests will need to recommence from the beginning.
   2. **Methods and Constraints**
      1. Methodology
         1. Testing will proceed using the user interface controls. If controls prove to be unusable for complete testing. The program will be returned to the developers until such time as the controls are available.
      2. Test Tools
         1. The NUnit plugin for Visual Studio 2012+ will be used for Unit testing which will precede the testing listing above.
      3. Extent
         1. The testing shall be complete testing of the application from a user’s point of view.
      4. Data Recording:
         1. Defects will be listed by the testers on a standard form noting date, screen shots and any other graphical information that will help the developer solve the defect.. As this is manual user interface testing, screen shots and other documentation such that the developer can reproduce the defect will be required when necessary.
      5. Constraints
         1. As noted above, testing shall occur manually using the user interface and shall proceed as far as the user interface functionality allows.
   3. Evaluation
      1. Criteria: Given this is manual user interface testing, each test shall be judged to correct based upon the judgment of the tester as to whether the user interface is showing the correct status according to the design, and the rules of the Battleship Game (See the SRS Nov 2016) for details of the game rules.
      2. Data Reduction: This section is not applicable for this application.
2. Test Descriptions
   1. Test (Game Start via IDE)
      1. Control: User shall start the application via Visual Studio
      2. Inputs: None
      3. Outputs: Game User Interface shall appear.
      4. Procedures: Start the program via Visual Studio
   2. Test (Game Start via executable)
      1. Control: User shall locate and run the application via the Windows terminal
      2. Inputs: Application executable
      3. Outputs: Game User Interface shall appear
      4. Procedures: Locate the application executable and run it via the Windows terminal.
   3. Test (User Interface Configuration)
      1. Control: Using either Test 4.1 or 4.2, the application shall be started
      2. Inputs:
         1. Test 4.1 or 4.2
         2. User Interface Mockup Documentation
      3. Output: User Interface configuration.
      4. Procedures: User shall confirm that the actual user interface conforms to the design documents, noting any and all discrepancies.
   4. Test (Place Ship)
      1. Control: Proceeding from 4.3, the user shall attempt to place each user’s game pieces (ships) via the control provided in the user interface.
      2. Inputs: User interface Method provided to place game pieces.
      3. Outputs: Placement of each players ships in the correct manner according to the documentation including
         1. Only horizontal or vertical placement
         2. No overlapping placement
         3. Placement wholly within each player’s own Allied Grid.
         4. Each player places ships separately and within their own turn with Player 1 going first.
      4. Procedures: User shall use the method prescribed in the user interface whether it is automated or manual. The user’s manual shall prescribe the actual method.
   5. Test (Game Play)
      1. Control: Proceeding from Test 4.4., the user shall begin game-play.
      2. Input: Using the provided user interface controls the user shall guess the location of each ship of the opponents. See the user’s manual for directions.
      3. Outputs: Each player shall have one guess per turn and each guess shall produced either a graphical, textual or both indication of the result of the guess. These include color changes within the player’s grid and/or textual indications such as “You missed”. See user’s manual for correct indications.
      4. Procedures: The user shall use the left mouse button to click on a location within an opponent’s grid (or if a computer-based player use an automated method described in the design documents). The indication of the result shall appear within the allotted time (see the SRS Nov. 2016)
   6. Test (Ship Sunk)
      1. Control: Proceeding with the repetitive application of the tests listed in 4.5
      2. Inputs: The user performs the same test as listed in 4.5 until they have guessed all of the locations covered by any one of the game pieces.
      3. Outputs: The user interface shall acknowledge that the user has found and “sunk” a specific game piece (or ship) by name.
      4. Procedure: See Section 4.5 for details on normal Game Play.
   7. Test (Premature Exit)
      1. Control: At any point after test 4.1 or 4.2
      2. Input: The user shall click on the method provided by the user interface to prematurely exit the game (see user interface mock-ups or the user manual for details)
      3. Outputs: The user interface shall confirm that the user really wants to exit the game. If the user confirms the result, the game shall exit. If the user cancels the result, game play shall continue at the point when the exit button was pressed.
   8. Test (Game Won/Over)
      1. Control: Proceeding from Test 4.5 and 4.6 but not 4.7
      2. Inputs: The user shall proceed with the Game Play test (4.5) until the Ship Sunk Test (4.6) has been successful 5 times, thus discovering all of the locations of the player’s ships.
      3. Outputs: When all of a player’s ships have been discovered (and thus “sunk”), the game is over and the user interface shall acknowledge that fact either textually, graphically or both (see User’s manual and/or User Interface Mockups for details).
      4. Procedures: Follow the tests listed in test 4.5 and 4.6 until the event listed above occurs and note the result.
   9. Test (Game Reset)
      1. Control: Proceeding from test 4.8
      2. Input: The user interface shall display a control allowing for either the ability to play the game again or exit the application.
      3. Output: When the user chooses the exit choice, the test shall precede the same as test 4.7 with the confirmation and eventual exit. If the user chooses the “play again” option or cancels out of the game exit option, the user interface shall reset to the state similar to that after test 4.1 or 4.2 with all traces of the previous game removed and the user able to start the game again using the procedures in test 4.4. The user interface may or may not acknowledge the previous winner at this time.