Chessborne

Test Case Specifications/Results

|  |  |
| --- | --- |
| **Group Members** | **Ryan Kalan**  **Kevin Tayah**  **Jace Beaudoin**  **Brandon Kolle**  **Tharunkumar Mallireddigari** |

**Table of Contents**

1. **Introduction**

1.1 Purpose of Document

1.2 Scope of Document

1.3 Definition Glossary

1. **Acceptance Testing**

2.1 Functionality Test Scenarios

2.2 Performance/Stress/Reliability Test Scenarios

2.3 Security Test Scenarios

2.4 Recovery/Failover Test Scenarios

1. **Unit Testing**

3.1 Server Unit Testing

1. Introduction

**1.1 Purpose of Document**

This document will provide actual results, along with pass/fail status, for various test cases and acceptance criteria outlined for the game Chessborne. This document serves as an outline for all developers and shareholders to ensure that the final version of the game contains all necessary features/requirements. In addition, test cases are designed to verify and validate the game’s intended behavior.

**1.2 Scope of Document**

This document provides a thorough report of all necessary tests that must be met before the product can be released to consumers. Areas like feature functionality, performance, and security are considered when performing acceptance testing.

**1.3 Definition Glossary**

* **Check:** The state a player is in when an opponent’s piece can capture the king on the next turn. The defending player must move their king out of harm's way or capture the attack piece. If they cannot, they are in checkmate.
* **Checkmate:** A win state for a player in which the opponent’s King is in check, and cannot move their king, or capture the attacking piece.
* **Castling:** A move where as long as the King is not in check, has not moved, and if any of the Rooks have not moved can make a move where the King moves 2 spaces towards the rook of choice and the rook will pass by the King and land on the square adjacent to the King on the other side. This can happen only if above conditions are met and if the King will not move through check and if the shares between the King and rook are clear.
* **En Passant:** If a pawn moves two square forward and lands adjacent to another pawn to its East or West side, the adjacent piece can capture the pawn en passant or in passing by landing on where the previous moved pawn would have landed if it had moved one square forward.
* **Piece:** The game pieces in chess and their movements as long as another piece doesn’t occupy that tile. For clarity, directions are described as North, South, East and West as if the player is looking at the board head-on.
  + King: Can move one space in it’s adjacent 8 spaces
  + Queen: Can move any distance in 8 directions
  + Rook: Can move any distance in the 4 cardinal directions (North, South, East, West)
  + Knight: Can move in an L-shaped pattern. For example, North->North->East. The first two directions must be the same, and the final direction must be perpendicular to the first two directions.
  + Bishop: Can move any distance diagonally in 4 directions (North-East, North-West, South-East, South-West)
  + Pawn: Can move two spaces North on their first move, or one space North on their subsequent moves. They can only capture on diagonal tiles in front of them.
* **Capture:** A piece can eliminate another piece on the board by moving on it’s tile, as long as it’s an opponent's piece.
* **Board:** A grid consisting of 8x8 tiles, usually with alternating colors on every other tile.
* **Turn:** A turn consists of one move, from one player.

2. Acceptance Testing

**2.1 Functionality Test Scenarios**

|  |  |
| --- | --- |
| **ID** | FUNC000 |
| **Title** | The purpose of this test case is to ensure a join option exists where players can enter the game. |
| **Priority** | High |
| **Description** | Once the user interface (web page) loads, an option to join a chess game will populate. The first two users to the webpage will be assigned player 1 and player 2 titles. Following users will be titled as audience members of the game. |
| **Expected Results** | Initial web page has a join feature that users can click and once clicked, the user is redirected to the main chessboard webpage after the first two members (player 1 and player 2) join. Player 1 will be assigned white chess pieces. Player 2 is assigned black chess pieces. The players will have an option of selection/movement of pieces. Any users joining after player 1 and player 2 will be redirected to the main game board but with only view options. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | The first two players to the join feature are assigned the players (1 and 2). Any user who joins after the first two are considered spectators and can watch the match between player 1 and player 2. |

|  |  |
| --- | --- |
| **ID** | FUNC001 |
| **Title** | The purpose of this test case is to ensure that player 1 has the first move |
| **Priority** | High |
| **Description** | Once the chess match begins, player 1 (white chess pieces) should have the first move. |
| **Expected Results** | As the game starts, player 1 should be prompted to move first since they hold the first move. Following moves will alternate between both players. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Verify player 1 has first move once the game is loaded. Player 1 should be able to move their desired piece first. |

|  |  |
| --- | --- |
| **ID** | FUNC002 |
| **Title** | The purpose of this test case is to ensure that players can only move their chess pieces when it is their turn. |
| **Priority** | High |
| **Description** | Once the chess match begins, player 1 should only be able to move white pieces when it is player 1 move. Similarly, player 2 should only be able to move black pieces when it is player 2 move. |
| **Expected Results** | When it is player 1 move, player 2 should have no functionality to move and player 1 should only be able to move player 1 pieces (white).  When it is player 2 move, player 1 should have no functionality to move and player 2 should only be able to move player 2 pieces (black).. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Both players should be prohibited from moving opponents' pieces. |

|  |  |
| --- | --- |
| **ID** | FUNC003 |
| **Title** | The purpose of this test case is to ensure that players can only move their chess pieces to valid spots. |
| **Priority** | High |
| **Description** | When a player attempts to move a chess piece, spots on the chessboard where the move is valid should light up. The piece should only be able to be moved to the lit up spots. |
| **Expected Results** | When either player attempts to move a chess piece, the chess piece should only be moved to lit up spots, which indicates that the move is valid. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Both players should be prohibited from moving to invalid spots. |

|  |  |
| --- | --- |
| **ID** | FUNC004 |
| **Title** | The purpose of this test case is to ensure that players can only move their King chess piece when they are in check state, or a piece that can capture the checking piece. |
| **Priority** | High |
| **Description** | When a player is in check state, they must move their king piece to continue the game. Hence, when the king is in check, the player needs to move to a lit up spot on the chess piece, which will remove the king from check. (\*check FUNC005 for when the king is in check and does not have any spots to move)  This also applies to pieces that can capture the piece checking the King. |
| **Expected Results** | When a check occurs, the player in check should only be allowed to move their king piece to a safe lit up spot (removing the king from check) or capturing a checking piece, or blocking the check from happening. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Only the king chess piece is to be moved when in check. Other pieces will not be allowed to be moved. |

|  |  |
| --- | --- |
| **ID** | FUNC005 |
| **Title** | The purpose of this test case is to ensure that the game is over once a king is in checkmate. The king cannot be moved to another secure spot. |
| **Priority** | High |
| **Description** | When a player faces a checkmate, the game will not continue and the winner who implemented the checkmate is crowned. |
| **Expected Results** | When a checkmate occurs, the game is over and a message signaling the winner is prompted. |
| **Actual Results** | **FAILED** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Checkmate signals the game is over. |

|  |  |
| --- | --- |
| **ID** | FUNC006 |
| **Title** | The purpose of this test case is to ensure that players receive the modified state board once the opponent makes a move. |
| **Priority** | High |
| **Description** | When a player makes their move, the modified board state will be sent to the other player, who will then see their board update with the most recent move. After viewing the recent move, the player can then make their move. |
| **Expected Results** | Once a player makes a move, the new board state is updated on the opponents screen. The new move should be made visible to the opponent in this process. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Board screens should be updated after each move over the course of the game. |

|  |  |
| --- | --- |
| **ID** | FUNC007 |
| **Title** | The purpose of this test case is to ensure that players receive a promotion of a game piece once a pawn makes it to the opponent’s (other end) end of the game board. |
| **Priority** | Med |
| **Description** | If a pawn makes it to the opponent's end of board, the user will get to choose what piece he/she would like to promote the pawn to, and that newly promoted piece will be placed on that tile. |
| **Expected Results** | If a pawn makes it to the opponent’s end of the board, a pop-up screen should populate which allows the player to select the piece they would like the pawn to promote to. Once the promotion occurs, both players should see the selected piece in place of the pawn that made its way to other end of board |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | New promoted pieces should replace the pawn that made it to the other side of the board. |

|  |  |
| --- | --- |
| **ID** | FUNC008 |
| **Title** | The purpose of this test case is to ensure that invalid moves and mismatch gameboard don’t occur during gameplay.The gameplay should not error out. |
| **Priority** | High |
| **Description** | During gameplay, invalid chess moves should not occur for either player. In addition, both players should be viewing the synced game board. The board for both players should display the same piece layout after each move. |
| **Expected Results** | Once a player makes a move, the new board state is updated on the opponents screen. In addition, the players should be prohibited from moving chess pieces to spots that are not lit up. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Board screens should be updated after each move over the course of the game. In addition, chess pieces should not be allowed to move to invalid locations. |

|  |  |
| --- | --- |
| **ID** | FUNC009 |
| **Title** | Player should be able to Castle |
| **Priority** | Med |
| **Description** | Players should be allowed to castle his King. As spoken in the glossary, that is to move the King more than one space towards an unmoved rook. |
| **Expected Results** | Player is able to make this move |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | N/A |

|  |  |
| --- | --- |
| **ID** | FUNC010 |
| **Title** | Player is able to perform En Passant |
| **Priority** | Low |
| **Description** | Players should be allowed to capture a Pawn, using a pawn because of En Passant. As spoken in the glossary, this is a diagonal capture beyond another pawn. |
| **Expected Results** | Player is able to make this move |
| **Actual Results** | **FAIL**- Actual Results are different to those outlined in Expected Results. En Passant was attempted, but did not work in the end. |
| **Comments** | N/A |

**2.2 Performance/Stress/Reliability** **Test Scenarios**

|  |  |
| --- | --- |
| **ID** | PERF000 |
| **Title** | The purpose of this test case is to ensure that lag issues do not occur during gameplay. |
| **Priority** | Med |
| **Description** | In the entirety of finding an opponent in the lobby and game execution, there should remain little to no lag between screens, browsers, and moves within the game. |
| **Expected Results** | Users should be able to join and play chess without lag between screens, browsers, and moves with the game. Updates in the game board should be made simultaneously and lag should not be an issue. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Lag should not exist at any point during gameplay. |

|  |  |
| --- | --- |
| **ID** | PERF001 |
| **Title** | The purpose of this test case is to ensure that performance does not suffer when high volumes of users are on the chess game. |
| **Priority** | Low |
| **Description** | Higher number of spectators observing a game should not affect performance with increased lag. |
| **Expected Results** | Gameplay performance should be similar with either 2 or with 200 users. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Performance should not be an issue with increased traffic. |

|  |  |
| --- | --- |
| **ID** | PERF002 |
| **Title** | The purpose of this test case is to ensure that gameplay is available on the specified operating systems: Windows 7, 8, and 10, along with Mac OS (version 10.15.6 and newer versions). |
| **Priority** | High |
| **Description** | Gameplay is available on Windows 7, 8, and 10, along with Mac OS (version 10.15.6 and newer versions). Game should work similarly on all operating versions stated. |
| **Expected Results** | Gameplay should be available on Windows 7, 8, and 10, along with Mac OS (version 10.15.6 and newer versions). Other operating systems will not be required. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | The specified operating systems should allow gameplay. |

**2.3 Security Test Scenarios**

|  |  |
| --- | --- |
| **ID** | SEC000 |
| **Title** | The purpose of this test case is to ensure that server details are not visible on UI. |
| **Priority** | High |
| **Description** | The server should remain secure during game play. The implementation details should remain hidden to protect application against potential threats. Ensure that important server details are not leaked to users on the user interface side. |
| **Expected Results** | None of the server details should be accessible or visible during gameplay. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Server implementation details are to be hidden from users. |

|  |  |
| --- | --- |
| **ID** | SEC001 |
| **Title** | The purpose of this test case is to ensure that users (can be viewers) cannot gain access to any of the game players and interfere with gameplay. |
| **Priority** | High |
| **Description** | Once a game starts, players' connections are secure and viewers (and other users) cannot gain access to gameplay and they shall remain viewers of the game. |
| **Expected Results** | Players should have access throughout gameplay and viewers cannot obtain access in any way. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Secure connections for players. |

**2.4 Recovery/Failover Test Scenarios**

|  |  |
| --- | --- |
| **ID** | FAIL000 |
| **Title** | The purpose of this test case is to ensure that a game restarts properly when the prior game finishes or either player leaves during the middle of a game. |
| **Priority** | High |
| **Description** | Once a user is crowned winner or they leave during the middle of gameplay, the application restarts and can be used to play the game again. |
| **Expected Results** | When a game is completed, the game should restart with the next two users as players. A game should also restart similarly when a player leaves during the middle of the game. |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | Restart should occur swiftly. |

3. Unit Testing

**3.1 Server**

|  |  |
| --- | --- |
| **ID** | UNIT000 |
| **Title** | Initialization Test |
| **Priority** | High |
| **Description** | Node.js/Express server should properly start up |
| **Expected Results** | Server should initialize without errors, and be ready to serve pages and process endpoint functionality |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | This does not include testing endpoints or other functionality |

|  |  |
| --- | --- |
| **ID** | UNIT001 |
| **Title** | Homepage Test |
| **Priority** | High |
| **Description** | Homepage should be served when hitting the default url (hostname/) |
| **Expected Results** | 200 HTTP response on a request to the server to ‘/’, with a page being served back |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | This does not test that the homepage was correctly rendered or even was properly sent, just that a page was sent |

|  |  |
| --- | --- |
| **ID** | UNIT002 |
| **Title** | Register Endpoint Success on New Registration |
| **Priority** | High |
| **Description** | When a new registration is submitted in the correct format, the user should be successfully registered |
| **Expected Results** | 200 response on a new registration (new being they have not registered before) |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | This test makes no distinction between player and spectator registrations |

|  |  |
| --- | --- |
| **ID** | UNIT003 |
| **Title** | Register Endpoint Fail on Old Registration |
| **Priority** | Med |
| **Description** | If a user is already registered, they should not be able to register twice |
| **Expected Results** | Return 403 error when a registration is attempted to be made twice |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | This test makes no distinction between player and spectator registrations |

|  |  |
| --- | --- |
| **ID** | UNIT004 |
| **Title** | Register Endpoint Fail on Improper Input |
| **Priority** | High |
| **Description** | If bad input is sent to the register endpoint, the request should be rejected |
| **Expected Results** | Return 400 error when the endpoint receives a request with bad input |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | This test makes no distinction between player and spectator registrations |

|  |  |
| --- | --- |
| **ID** | UNIT005 |
| **Title** | Register Endpoint First Registration Is Player |
| **Priority** | High |
| **Description** | The first player to register should be registered as a player |
| **Expected Results** | Return 200 with playerStatus listed as ‘player’ |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** |  |

|  |  |
| --- | --- |
| **ID** | UNIT006 |
| **Title** | Register Endpoint Second Registration is Player |
| **Priority** | High |
| **Description** | The second player to register should be registered as a player |
| **Expected Results** | Return 200 with playerStatus listed as ‘player’ |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** |  |

|  |  |
| --- | --- |
| **ID** | UNIT007 |
| **Title** | Register Endpoint Any Other Registration is Spectator |
| **Priority** | Low |
| **Description** | The any player to register after the second registration should be registered as a spectator |
| **Expected Results** | Return 200 with playerStatus listed as ‘spectator’ |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** |  |

|  |  |
| --- | --- |
| **ID** | UNIT008 |
| **Title** | Send Endpoint Improper Input Should Fail |
| **Priority** | High |
| **Description** | If a state object sent to the Send endpoint is improperly formatted, the request should be rejected |
| **Expected Results** | Return 400 when improper input is received |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** | State schema will have to be maintained between frontend and backend |

|  |  |
| --- | --- |
| **ID** | UNIT009 |
| **Title** | Send Endpoint Proper Input Success |
| **Priority** | High |
| **Description** | A properly constructed state object sent to the Send endpoint should result in success |
| **Expected Results** | Return 200 with success set to true |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** |  |

|  |  |
| --- | --- |
| **ID** | UNIT010 |
| **Title** | Send Endpoint Proper Input State Imported |
| **Priority** | High |
| **Description** | When a properly constructed state object is sent to the Send endpoint, the server should import and store the state internally |
| **Expected Results** | State is stored, overwriting previously stored state in the server |
| **Actual Results** | **PASS** - Actual Results are equivalent to those outlined in Expected Results. |
| **Comments** |  |