Software Engineering Group Project - Test Report

| Author(s): | Craymon Chan [ckc2@aber.ac.uk]  Jim Brown [jib16@aber.ac.uk] |
| --- | --- |
| Config Ref: | N/A |
| Date: | 3 May 2023 |
| Version: | 1.0 |
| Status: | Released |

| Department of Computer Science |
| --- |
| Aberystwyth University |
| Aberystwyth |
| Ceredigion |
| SY23 3DB |
| Copyright © Aberystwyth University 2023 |

| **Test Ref** | **Test Content** | **Result (Pass/Fail)** | **Explanation** |
| --- | --- | --- | --- |
| **SE-F-001** | Test whether the starting screen pops up correctly or functions properly | **Pass** |  |
| **SE-F-002** | Check whether new game works properly | **Pass** |  |
| **SE-F-003** | Check whether ‘Load Unfinished Game’ works properly by using previously played games saved in the log files | **Pass** |  |
| **SE-F-004** | Check whether ‘View Finished Game’ works properly | **Pass** |  |
| **SE-F-005** | Check whether the players entered their names after being prompted | **Pass** |  |
| **SE-F-006** | Check whether name counts exceed 32 characters | **Pass** |  |
| **SE-F-007** | Check whether name characters contain special characters | **Pass** |  |
| **SE-F-008** | Check whether blank names which where no characters are entered | **Pass** |  |
| **SE-F-009** | Check whether White side player’s name(Tony) is at the bottom, whilst Black side player’s name(Sam) is at the top | **Pass** |  |
| **SE-F-010** | Both players names, pieces and colour is tracked correctly | **Pass** |  |
| **SE-F-011** | Detects any Pawn Pieces where they have reached the farthest rank from its original position and their promotion is mandatory while having the ability to promote to a Queen, Knight, Bishop or Rook Piece | **Pass** |  |
| **SE-F-012** | Tutor keeps track of starting player as in white side | **Pass** |  |
| **SE-F-013** | Tutor keeps track of current player and shows visually whos turn it is | **Pass** |  |
| **SE-F-014** | Unambiguous board representation | **Pass** |  |
| **SE-F-015** | Can only make one move per turn | **Pass** |  |
| **SE-F-016** | Only current player can move their own pieces | **Pass** |  |
| **SE-F-017** | Current player cannot move opponent’s pieces | **Pass** |  |
| **SE-F-018** | Check if piece reselection works | **Pass** |  |
| **SE-F-019** | Check if piece selection works when the piece is pinned | **Pass** |  |
| **SE-F-020** | Check if piece selection works when the piece is in check | **Pass** |  |
| **SE-F-021** | Check if piece reselection works when the piece is available to move | **Pass** |  |
| **SE-F-022** | Check whether Pawn Pieces are moving legally in one step | **Pass** |  |
| **SE-F-023** | Check whether Pawn Pieces are moving legally in two steps | **Pass** |  |
| **SE-F-024** | Test functionality of Pawn Pieces for capturing pieces | **Pass** |  |
| **SE-F-025** | Test Pawn Pieces applying En Passant rule which only occurs when a pawn captures a horizontally adjacent enemy pawn piece that made an initial two square advance | **Pass** |  |
| **SE-F-026** | Check if players intend to move Pawn Pieces into restricted spaces | **Pass** |  |
| **SE-F-027** | Check whether Bishop Pieces are moving legally | **Pass** |  |
| **SE-F-028** | Test functionality of Bishop Pieces for capturing enemy pieces | **Pass** |  |
| **SE-F-029** | Test functionality of Bishop making illegal move | **Pass** |  |
| **SE-F-030** | Check whether Knight Pieces are moving legally horizontally | **Pass** |  |
| **SE-F-031** | Check whether Knight Pieces are moving legally on vertically | **Pass** |  |
| **SE-F-032** | Test functionality of Knight Pieces for capturing enemy pieces vertically | **Pass** |  |
| **SE-F-033** | Test functionality of Knight Pieces for capturing enemy pieces on the horizontally | **Pass** |  |
| **SE-F-034** | Test functionality of knight piece making illegal move | **Pass** |  |
| **SE-F-035** | Check whether Rook Pieces are moving legally as in vertically | **Pass** |  |
| **SE-F-036** | Check whether Rook Pieces are moving legally as in horizontally | **Pass** |  |
| **SE-F-037** | Test functionality of Rook Pieces for capturing enemy pieces vertically | **Pass** |  |
| **SE-F-038** | Test functionality of Rook Pieces for capturing enemy pieces horizontally | **Pass** |  |
| **SE-F-039** | Test functionality of rook moving illegally | **Pass** |  |
| **SE-F-040** | Check whether Queen Piece is moving legally | **Pass** |  |
| **SE-F-041** | Test functionality of Queen Piece for capturing enemy pieces | **Pass** |  |
| **SE-F-042** | Test functionality of queen piece making illegal move | **Pass** |  |
| **SE-F-043** | Check whether King Piece is moving legally | **Pass** |  |
| **SE-F-044** | Test functionality of King Piece for capturing enemy pieces | **Pass** |  |
| **SE-F-045** | Test functionality of king piece moving illegally | **Pass** |  |
| **SE-F-046** | Test functionality of king piece moving itself into check | **Pass** |  |
| **SE-F-047** | Test functionality of kingside castling when allowed | **Pass** |  |
| **SE-F-048** | Test functionality of queenside castle when allowed | **Pass** |  |
| **SE-F-049** | Test functionality of kingside castle when not allowed because king or rook previously moved | **Pass** |  |
| **SE-F-050** | Test functionality of queenside castle when not allowed because king or rook previously moved. | **Pass** |  |
| **SE-F-051** | Test functionality of kingside castle when not allowed because there are pieces in the way | **Pass** |  |
| **SE-F-052** | Test functionality of queenside castle when not allowed because there are pieces in the way | **Pass** |  |
| **SE-F-053** | Test functionality of attempting to castle when in check | **Fail** | **Player is allowed to castle when in check to move out of check which shouldn't be allowed.** |
| **SE-F-054** | Test functionality of attempting to castle through check | **Fail** | **Player is allowed to castle when an opponent's piece is covering the path the king would castle through.** |
| **SE-F-055** | Test functionality of Pawn Pieces which are eligible for promotion | **Pass** |  |
| **SE-F-056** | Test error condition if piece is moved off the board | **Pass** |  |
| **SE-F-057** | Tutor detects whether a player’s king is currently in the checked state | **Pass** |  |
| **SE-F-058** | Promoted pieces function correctly when putting opposing king in check | **Pass** |  |
| **SE-F-059** | Tutor detects whether a players king is in the state of checkmate | **Pass** |  |
| **SE-F-060** | Tutor detects whether checkmate is avoidable by moving piece to block check | **Pass** |  |
| **SE-F-061** | Tutor detects whether checkmate is avoidable by moving piece to capture attacking piece | **Pass** |  |
| **SE-F-062** | Tutor detects whether checkmate is avoidable by moving king out of checkmate | **Pass** |  |
| **SE-F-063** | Tutor detects whether the player’s king is in the state of checkmate | **Pass** |  |
| **SE-F-064** | Checkmate | **Pass** |  |
| **SE-F-065** | Resign | **Pass** |  |
| **SE-F-066** | Agree to finish (Offer Draw) | **Pass** |  |
| **SE-F-067** | Disagree to finish  (Offer Draw) | **Pass** |  |
| **SE-F-068** | Resign the game after game is won (checkmate) | **Pass** |  |
| **SE-F-069** | Game quit | **Pass** |  |
| **SE-F-070** | Exit game without saving | **Pass** |  |
| **SE-F-071** | Exit game without quitting | **Pass** |  |
| **SE-F-072** | Game replay | **Pass** |  |
| **SE-F-073** | Replay step forward | **Pass** |  |
| **SE-F-074** | Replay step backwards | **Pass** |  |
| **SE-F-075** | Replaying invalid step forward | **Pass** |  |
| **SE-F-076** | Replaying invalid step backwards | **Pass** |  |
| **SE-F-077** | Attempt to replay game when there is nothing to replay | **Pass** |  |
| **SE-F-078** | Game saved while playing | **Pass** |  |
| **SE-F-079** | Game reloaded | **Pass** |  |
| **SE-F-080** | Save name cannot exceed 32 characters | **Pass** |  |
| **SE-F-081** | Save name cannot have special characters | **Pass** |  |
| **SE-F-082** | Test whether back button works in main menu and in game | **Pass** |  |

# 