|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Description | Steps | Expected | Actual | Result | Comment |
|  |  |  |  |  |  |  |
| 1 | Test the init() method of TicTacToe class to ensure that it initializes the game with a blank board, X as the starting player, and the game\_over flag set to False. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Check that the board attribute of the instance is a list of 9 empty strings.  3. Check that the current\_player attribute of the instance is set to "X".  4. Check that the game\_over attribute of the instance is set to False. | Board attribute is a list of 9 empty strings, current\_player attribute is set to "X", and game\_over attribute is set to False. | Board attribute is a list of 9 empty strings, current\_player attribute is set to "X", and game\_over attribute is set to False. | Pass | init() method of TicTacToe class initializes the game state as expected. |
| 2 | Test the make\_move() method of TicTacToe class to ensure that it updates the board with the correct move and switches the current player. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Call the make\_move() method with index 0.  3. Check that the board attribute of the instance has the first element set to "X".  4. Check that the current\_player attribute of the instance is set to "O".  5. Call the make\_move() method with index 1.  6. Check that the board attribute of the instance has the second element set to "O".  7. Check that the current\_player attribute of the instance is set to "X". | Board attribute has first element set to "X", second element set to "O", and current\_player attribute is set to "X". | Board attribute has first element set to "X", second element set to "O", and current\_player attribute is set to "X". | Pass | make\_move() method of TicTacToe class updates the board with the correct move and switches the current player. |
| 3 | Test the check\_for\_winner() method of TicTacToe class to ensure that it correctly identifies a winner or a tie game. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a list representing a winning game state.  3. Call the check\_for\_winner() method.  4. Check that the method returns "X" as the winner.  5. Set the board attribute of the instance to a list representing a tie game state.  6. Call the check\_for\_winner() method.  7. Check that the method returns "Tie" as the game result. | check\_for\_winner() method returns "X" as the winner for a winning game state, and "Tie" as the game result for a tie game state. | check\_for\_winner() method returns "X" as the winner for a winning game state, and "Tie" as the game result for a tie game state. | Pass | The check\_for\_winner() method correctly identifies a winning game board state where the first player has won by completing a diagonal line from the top left corner to the bottom right corner. |
| 4 | Test the make\_move() method of TicTacToe class to ensure that it updates the board and checks for a win or draw when a move is made. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a specific pattern.  3. Set the current\_player attribute of the instance to "X".  4. Set the game\_over attribute of the instance to False.  5. Call the make\_move() method of the instance with a specific position.  6. Check that the board attribute of the instance is updated with the player's symbol at the specified position.  7. Check that the text of the button at the specified position in the game board is updated with the player's symbol.  8. Check that if the current\_player has won the game, a custom message box is displayed with the player's symbol and the game\_over flag is set to True.  9. Check that if the game ends in a draw, a custom message box is displayed with the message "It's a draw!" and the game\_over flag is set to True.  10. Check that the current\_player attribute of the instance is updated to the other player after the move. | Board attribute is updated with the player's symbol at the specified position, the text of the button at the specified position in the game board is updated with the player's symbol, a custom message box is displayed if the current\_player has won the game or if the game ends in a draw, and the current\_player attribute is updated to the other player after the move. | Board attribute is updated with the player's symbol at the specified position, the text of the button at the specified position in the game board is updated with the player's symbol, a custom message box is displayed if the current\_player has won the game or if the game ends in a draw, and the current\_player attribute is updated to the other player after the move. | Pass | make\_move() method of TicTacToe class updates the board and checks for a win or draw as expected. |
| 5 | Test the check\_win() method of TicTacToe class to ensure it correctly identifies a win condition. | 1. Create an instance of TicTacToe class.  2. Set the board attribute of the instance to a specific pattern representing a win condition for a player.  3. Call the check\_win() method of the instance with the player's symbol. | True | True | Pass | The check\_win() method correctly identified the win condition for the player. |
| 6 | Test the check\_win() method of TicTacToe class to ensure it correctly identifies a non-win condition. | 1. Create an instance of TicTacToe class.  2. Set the board attribute of the instance to a specific pattern that does not represent a win condition for any player.  3. Call the check\_win() method of the instance with a player's symbol. | False | False | Pass | The check\_win() method correctly identified that there is no win condition for the player. |
| 7 | Test the check\_win() method of TicTacToe class to ensure it correctly identifies a win condition on the edge of the board. | 1. Create an instance of TicTacToe class.  2. Set the board attribute of the instance to a specific pattern representing a win condition for a player on the edge of the board.  3. Call the check\_win() method of the instance with the player's symbol. | True | True | Pass | The check\_win() method correctly identified the win condition for the player on the edge of the board. |
| 8 | Test the check\_win() method of TicTacToe class to ensure it correctly identifies a win condition on the diagonal. | 1. Create an instance of TicTacToe class.  2. Set the board attribute of the instance to a specific pattern representing a win condition for a player on the diagonal.  3. Call the check\_win() method of the instance with the player's symbol. | True | True | Pass | The check\_win() method correctly identified the win condition for the player on the diagonal. |
| 9 | Test the check\_draw() method of TicTacToe class to ensure it returns True when the game ends in a draw and False when it doesn't. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a pattern that is not a winning pattern and has no empty spaces.  3. Call the check\_draw() method of the instance. | check\_draw() method returns True. | check\_draw() method returns True. | Pass | check\_draw() method returns True as expected when the game ends in a draw. |
| 10 | Test the check\_draw() method of TicTacToe class to ensure it returns False when the game does not end in a draw. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a pattern that is not a winning pattern and has empty spaces.  3. Call the check\_draw() method of the instance. | check\_draw() method returns False. | check\_draw() method returns False. | Pass | check\_draw() method returns False as expected when the game does not end in a draw. |
| 11 | Test the check\_draw() method of TicTacToe class to ensure that it correctly identifies a draw game. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a pattern that represents a draw game.  3. Call the check\_draw() method of the instance. | check\_draw() returns True. | check\_draw() returns True. | Pass | check\_draw() correctly identifies a draw game. |
| 12 | Test the reset\_game() method of TicTacToe class to ensure it resets the game board to a blank state. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a specific pattern.  3. Set the current\_player attribute of the instance to a specific player.  4. Set the game\_over attribute of the instance to True.  5. Call the reset\_game() method of the instance.  6. Check that the board attribute of the instance is reset to a blank state.  7. Check that the text of all buttons in the game board is empty.  8. Check that the current\_player attribute of the instance is set to "X".  9. Check that the game\_over attribute of the instance is set to False. | Board attribute is reset to a blank state, text of all buttons in the game board is empty, current\_player attribute is set to "X", and game\_over attribute is set to False. | Board attribute is reset to a blank state, text of all buttons in the game board is empty, current\_player attribute is set to "X", and game\_over attribute is set to False. | Pass | reset\_game() method of TicTacToe class resets the game board and other attributes as expected. |
| 13 | Test the reset\_game() method of TicTacToe class to ensure that it resets the game board to a blank state. | 1. Create an instance of TicTacToe class with a Tkinter root window.  2. Set the board attribute of the instance to a specific pattern.  3. Set the current\_player attribute of the instance to "O".  4. Set the game\_over attribute of the instance to True.  5. Call the reset\_game() method of the instance. | Board attribute is reset to a blank state, the text of all buttons in the game board is cleared, the current\_player attribute is reset to "X", and the game\_over attribute is reset to False. | Board attribute is reset to a blank state, the text of all buttons in the game board is cleared, the current\_player attribute is reset to "X", and the game\_over attribute is reset to False. | Pass | reset\_game() correctly resets the game board to a blank state. |