# Design and implementation Document

**I decided to use the Haskell language with functional paradigms to implement the modified tic-tac-toe program. Below is the overall pseudocode of the modified tic-tac-toe programme.**

A screenshot of a white page with black text

Description automatically generated

Figure 1- Pseudocode of tic-tac-toe

**In Figure 1 I have shown the overall pseudocode, so I have written pseudocode for each function below:**

Display Board Function

A screenshot of a computer program

Description automatically generated

Figure - pseudocode of display board function

Check Winner Function

A screenshot of a computer game

Description automatically generated

Figure - pseudocode of check winner function

Check Draw Function

A screenshot of a computer

Description automatically generated

Figure - pseudocode of check draw function

Game Condition Function

A screenshot of a game

Description automatically generated

Figure - pseudocode of game condition function

Check Game Over Function

A screenshot of a computer game

Description automatically generated

Figure -- pseudocode of check game over function

Available Moves Function

A screenshot of a computer program

Description automatically generated

Figure - pseudocode of available moves function

Valid Move Function

A screenshot of a computer program

Description automatically generated

Figure - pseudocode of valid move function

Update List Function

A white background with black text

Description automatically generated

Figure - pseudocode update list function

Make Move Function

A screenshot of a computer program

Description automatically generated

Figure - pseudocode of make move function

Switch Player Function

A white background with black text

Description automatically generated

Figure - pseudocode of switch player function

Game Function

A screenshot of a computer game

Description automatically generated

Figure - pseudocode of game function

Main Function

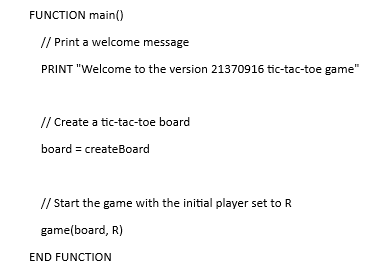


Figure - pseudocode of main function

**Follow the pseudocode function, I have written the code for each function below:**

Display Board Function

A computer screen with text on it

Description automatically generated

Figure 14- Display board function

Check Winner Function

A computer screen shot of text

Description automatically generated

Figure 15- Check winner function

Check Draw Function

A screenshot of a computer program

Description automatically generated

Figure 16- Check draw function

Game Condition Function

A screenshot of a computer screen

Description automatically generated

Figure 17-Game condition function

Check Game Over Function

A screenshot of a computer screen

Description automatically generated

Figure 18- Check game over function

Available Moves Function

A black background with colorful text

Description automatically generated

Figure 19- Available moves function

Valid Move Function

A black background with white text

Description automatically generated

Figure 20- Valid move function

Update List Function

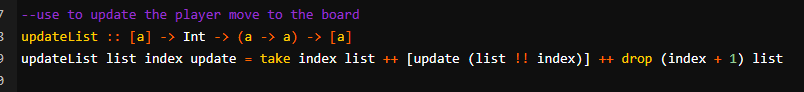


Figure 21- Update list function

Make Move Function

A screen shot of a computer program

Description automatically generated

Figure 22- Make move function

Switch Player Function

A screen shot of a computer

Description automatically generated

Figure 23- Switch player function

Game Function

A computer screen shot of a game

Description automatically generated

Figure 24- Game function

Main Function

A screenshot of a computer

Description automatically generated

Figure 25- Main function

**In the iteration code of my implementation process, I wrote the following code:**

A screenshot of a computer program

Description automatically generated

Figure 26- Iteration code part 1

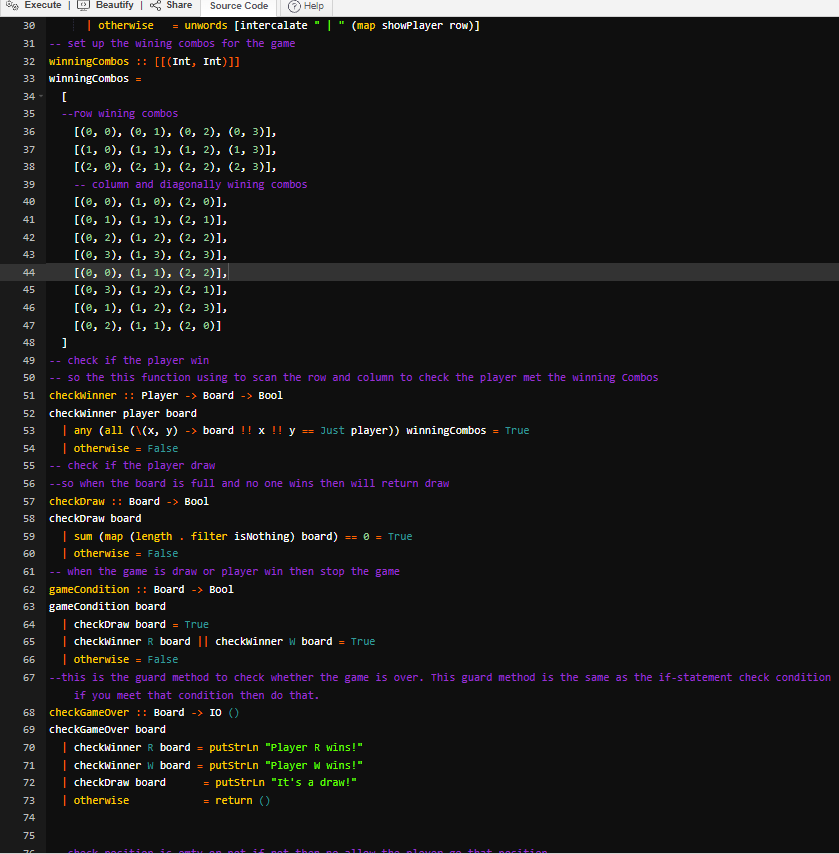


Figure 27- Iteration code part 2

A screen shot of a computer program

Description automatically generated

Figure 28- Iteration code part 3

A screenshot of a computer program

Description automatically generated

Figure 29- Iteration code part 4

A computer screen shot of a program code

Description automatically generated

Figure 30- Iteration code part 5

**In the iteration of my implementation, I have found some error, so I have updated the code**

**A screenshot of a computer screen

Description automatically generated**

Figure 31- Error find from the display board

A computer screen with text and images

Description automatically generated with medium confidence

Figure 32- Mention the "showPlayer " but haven’t create "showPlayer" function

A screen shot of a computer program

Description automatically generated

Figure 33- Updated code - Add the “showPlayer” function

A screenshot of a computer program

Description automatically generated

Figure 34- Error: Still can't show the board because the player input position 1-12 need to convert to coordination because the board and winning combos use the coordination for location

A computer screen shot of text

Description automatically generated

Figure 35- Update code: Add the positionToCoordinates function and use in the makeMove function

A screenshot of a computer

Description automatically generated

Figure 36- Updated code result - Display the board when run the code

A screenshot of a computer program

Description automatically generated

Figure 37 Error the board is show 4x3 grids not 3x4

A screenshot of a computer program

Description automatically generated

Figure 38- Error: The list set to create 1x3 list and create four times of that 1x3 list

A computer screen shot of text

Description automatically generated

Figure 39- Update list to create a 1x4 list and create three times of 1x4 list

A computer screen with a black background

Description automatically generated

Figure 40- Result after updated the code

**When the whole program working, and I have tested the programming. These are the tests that I conducted:**

1. Run to the end with R player winning

A screen shot of a computer

Description automatically generated

Figure 41- Test R player wining (col)

A screenshot of a computer

Description automatically generated

Figure 42- Test R player wining (diagonal)

A black screen with white text

Description automatically generated

Figure 43- Test R player wining (row)

1. Run to the end with W player winning

A computer screen with white text

Description automatically generated

Figure 44- Test W player wining (col)

A screen shot of a computer

Description automatically generated

Figure 45- Test W player wining (diagonal)

A screenshot of a computer

Description automatically generated

Figure 46- Test W player wining (row)

1. Run to the end with a draw.

A screen shot of a computer

Description automatically generated

Figure 47- Test the game draw

1. Validation of input the out-of-range number

A screenshot of a computer

Description automatically generated

Figure 48- Test input out of range number

1. Validation of input a letter

A screenshot of a computer

Description automatically generated

Figure 49- Test input a letter

1. Validation of input of the symbol to the cell that already exists

**A screenshot of a computer program

Description automatically generated**

Figure 50- Test input of the symbol to the cell that already exists

# Appendix

I learned how to use the guard in Haskell, so I asked the ChatGPT to provide an example of how to use guard and I used the example for gathering an idea and to implement my “checkWinner” and “checkDraw” etc guard functions in the program.

A screenshot of a chat

Description automatically generated

Figure 51- Interaction with ChatGPT about how to use guard in Haskell

I asked the ChatGPT to provide an example of how to use the recursion function in Haskell and use the idea for implement the “game” function.A screenshot of a computer

Description automatically generated

Figure 52- Interaction with ChatGPT about how to use recursion function in Haskell