Assignment 1
Names: Stephanie Saenz, Sofia Torres, Mia Marte
CAP 4630
Dr.Marques

Project 1 Tic-Tac-Toe

Project Roles:

- ➢ Architect role - *Mia Marte*
  The architect plays a crucial role in designing the overall structure and logic of the Tic-Tac-Toe game.
  - ★ Game Logic Design: define the game rules, logic, and mechanics. They need to determine how players will interact with the game, handle user inputs, and decide the winning conditions.
  - ★ Class and Function Design: design the classes and functions required for the game. define the functions for checking win conditions, displaying the game board, and handling player turns.
  - ★ Overall Project Structure: plan the project's file structure, organize modules and scripts, and decide on a suitable approach for code organization and separation of concerns.

- ➢ Developer role - *Stephanie Saenz*
  The developer will implement the game based on the design provided by the architect.
  - ★ Implementing Classes and Functions: The developer will write the code for the classes and functions defined by the architect. This includes creating the game board, implementing player actions, checking win conditions, and managing the game flow.
  - ★ Error Handling: The developer should handle potential errors and exceptions, ensuring that the game handles unexpected inputs gracefully and provides appropriate feedback to the players.
  - ★ Unit Testing: The developer should write unit tests to ensure that each function and class works as intended. This will help catch any bugs or issues early in the development process and ensure the game functions correctly.
  - ★ Code Integration: The developer will integrate the individual components and functionalities developed by themselves and other team members, ensuring smooth interactions between different parts of the game.

- ➢ Reporter role - *Sofia Torres*

The reporter is responsible for ensuring effective communication within the team and documenting the progress of the project.

- ★ Team Coordination: The reporter should schedule regular meetings and discussions with the architect and developer to review progress, clarify requirements, and address any questions or issues that may arise.
- ★ Progress Tracking: The reporter will keep track of the project's milestones, deadlines, and overall progress. They should regularly update the team on the status of each task and ensure that everyone is aware of their responsibilities.
- ★ Documentation and Reporting: The reporter should maintain documentation of the project, including meeting notes, decisions made, and any changes or updates to the initial design. This documentation will help with future maintenance and troubleshooting.
- ★ User Documentation: The reporter can also work on creating user documentation, such as a user manual or guide, to help users understand how to play the game and navigate its features.

Overall Design Decisions:

We implemented the code given from Kylie Yings' Youtube video but made modifications to it, so it can fit the requirements of the project guideline. For this assignment we are to use adversarial search, specifically MinMax search. As stated in the book, "min-max search aims to build a tree of possible outcomes based on moves that each player could make and favor paths that are advantageous to the agent while avoiding paths that are favorable to the opponent. Min-max search simulates the turns taken by each player, so the depth specified is directly linked to the number of turns between both players. A depth of 4, for example, means that each player has had 2 turns. Player A makes a move, player B makes a move, player A makes another move, and Player B makes another move."

Implementation:

These are the functions that we used to implement in our code. Each function has its definition and what are they supposed to do. We mainly got help from the book and for sources online, such as Youtube and Google for reference. We had several classes, and we had functions to make the code easier and readable for us. One of our class names was Player, which mainly prioritized the player.py file and made sure that it met the criteria. The other class name was Tic Tac Toe which was based on the instructions and how the game is supposed to be played. However, we also used Streamlit to deploy our app, which caused us to change our code to their syntax. We also changed the name of the files as front end and back end to make the process easier. The front end is the "game.py" that we had originally, and the back end is the "player.py" that we got from the beginning.

If we look at the screenshots below, you can see that our code is completely different from the one from Streamlit but the functions are the same. In our original code, we had the user play multiple times and asked the user to play again. Meanwhile, in our web-app the user has the availability to choose to either play against the AI generator or themselves. This made the process fun, because instead of asking the same questions all over again they were able to do it at the moment. Both our original code and our web-app acts the same, they just have different implementations.

Below are some our class/ definitions of our original code:

- Using Class Player - Name of the class
- Initialize function def _init_ - for which letter the player will use (x or o)
- Def getMove() - player
- make_board(): This static method creates and returns a list representing the Tic Tac Toe board. It initializes the board with empty spaces.
- print_board(): This will print the output and ask for input
- print_board_nums(): Prints numbers that correspond to the positions.
- make_move(square, letter): This method is used to make a move on the Tic Tac Toe board. It takes X or O as the letter.
- winner(square, letter): This method checks if the given move (square and letter) results in a winning move. It also checks the row, column, and diagonals of the board.
- empty_squares(): Checks if there are any empty squares remaining on the board.
- num_empty_squares(): Counts the number of empty squares remaining on the board and returns them.
- available_moves(): This method returns a list of available spots left in the board.
- __init__(self): This is the constructor of the TicTacToe class. It initializes the board using make_board() and sets current_winner to None.
- __init__(self, letter): This is the constructor of the Player class. It initializes a player with a given letter (X or O).
- get_move(self, game): This method is implemented differently in different player classes (HumanPlayer, RandomComputerPlayer, and SmartComputerPlayer).
- play(game, x_player, o_player, print_game=True): This function is used to play the Tic Tac Toe game. It takes the game instance, two players (X and O), and an optional parameter to print the game's progress.

Future Improvements:

We would like to add an option for multiplayer or online capabilities. Where users are able to access our game from anywhere in the world, with their computers or phones. With this option, they will have the chance to create a profile where they can save their

progress. The progress will allow you to go from beginner to intermediate and finally to expert. Where also the AI gets harder as you increase your level. This feature will be good for friends that like to challenge each other, but in a positive and friendly way.

Another thing we could add to our game is to implement virtual reality. Where users can bring this project into real life and interact with the pieces as if you were actually playing in real life. This will make the game not repetitive, instead it will be much fun since now you can actually play “in real life.” Not only that, but it will also be good to keep this game alive, and not let it die.

Streamlit web-app:
https://softorres-tictactoegame-frontend-nepywi.streamlit.app/

Link to our Git-hub repository:
https://github.com/softorres/TicTacToeGame.git

Screenshots of Our Solution:

Playing against Random AI (X winning)

```
Welcome to our Tic-Tac-Toe Game!

This program allows you to play a classic game of Tic-Tac-Toe against an AI generator.
The game is played on a 3x3 grid, and the objective is to be the first player to get three of their marks (X or O) in a horizontal, vertical, or diagonal line.


Follow our intstructions below to know how to play:

-> To play the game, two players take turns making their moves.

-> Player 1 will be assigned the mark 'X', and Player 2 will be assigned the mark 'O'.

-> The players will enter the coordinates of their desired move (row and column) to place their mark on the grid.

Here's an example of the grid layout:

  0 | 1 | 2
-----------
  3 | 4 | 5
-----------
  6 | 7 | 8


-> To make your move, enter the corresponding number when prompted.

-> For example, if you want to place your mark in the top-right corner, enter '3'.

-> The game will continue until one player wins by getting three marks in a row or if the entire grid is filled without a winner, resulting in a draw.
```

```
Now that you read our instructions, are you ready to play? (Y/N): y
Choose your opponent:
1 = Random AI
2 = Perfect AI
Enter the opponent's number (1 or 2): 1
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
O makes a move to square 3
|   |   |   |
| O |   |   |
|   |   |   |

X's turn. Input move (0-8): 0
X makes a move to square 0
```

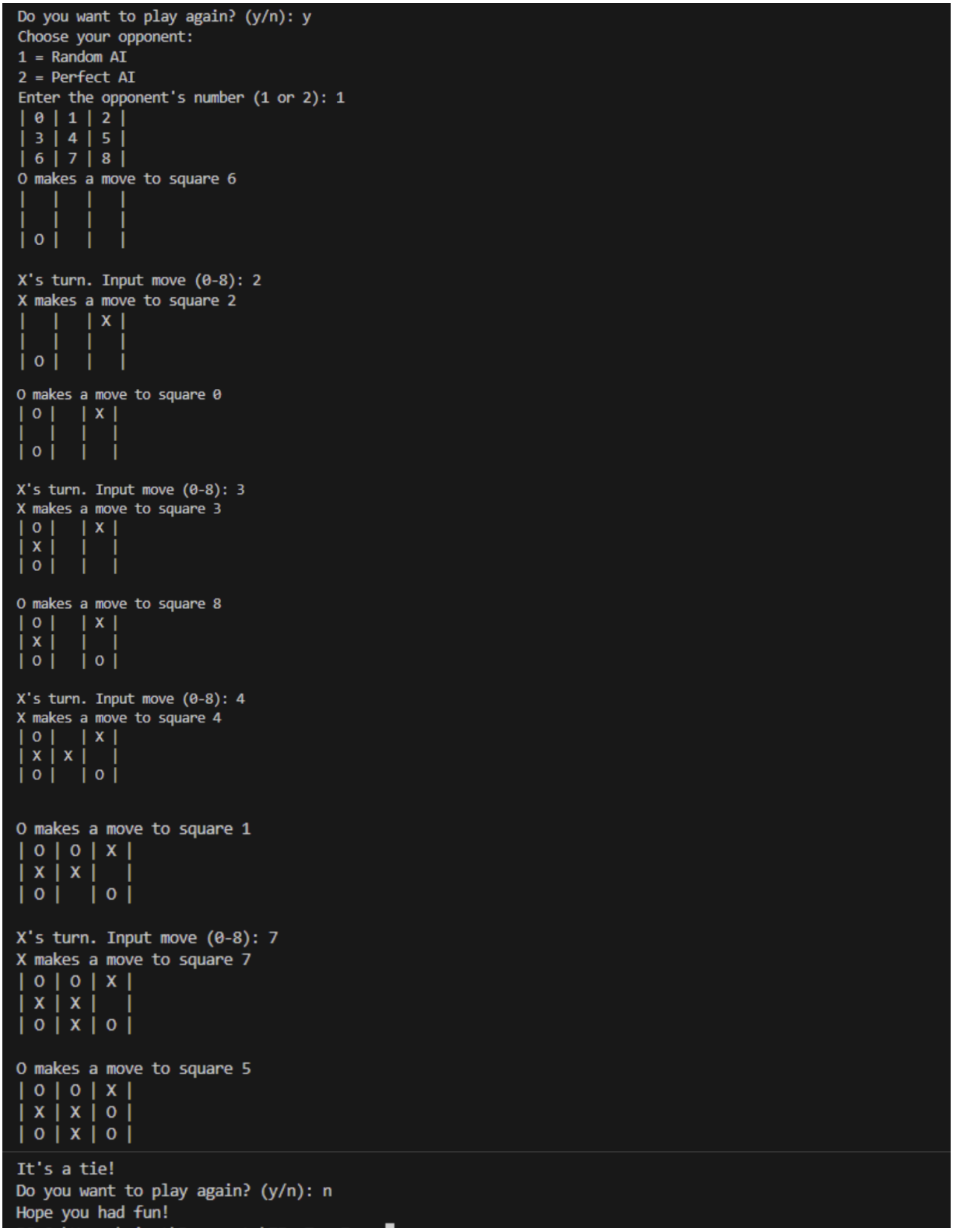
```
X's turn. Input move (0-8): 0
X makes a move to square 0
| X |   |   |
| O |   |   |
|   |   |   |

O makes a move to square 2
| X |   | O |
| O |   |   |
|   |   |   |

X's turn. Input move (0-8): 4
X makes a move to square 4
| X |   | O |
| O | X |   |
|   |   |   |

O makes a move to square 6
| X |   | O |
| O | X |   |
| O |   |   |

X's turn. Input move (0-8): 8
X makes a move to square 8
```

```
X's turn. Input move (0-8): 8
X makes a move to square 8
| X |   | O |
| O | X |   |
| O |   | X |

X wins!
```

Playing against Random AI (O winning)

```
X wins!
Do you want to play again? (y/n): y
Choose your opponent:
1 = Random AI
2 = Perfect AI
Enter the opponent's number (1 or 2): 2
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
O makes a move to square 1
|   | O |   |
|   |   |   |
|   |   |   |

X's turn. Input move (0-8): 3
X makes a move to square 3
|   | O |   |
| X |   |   |
|   |   |   |

O makes a move to square 0
| O | O |   |
| X |   |   |
|   |   |   |

X's turn. Input move (0-8): 8
X makes a move to square 8
| O | O |   |
| X |   |   |
|   |   | X |

O makes a move to square 2
| O | O | O |
| X |   |   |
|   |   | X |

O wins!
Do you want to play again? (y/n):
```

Playing against Random AI (Tie)



Playing against Perfect AI (Tie)

```
After reading our instructions. Are you ready to play? (Y/N): y
Choose your opponent:
1 = Random AI
2 = Perfect AI
Enter the opponent's number (1 or 2): 2
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
O makes a move to square 1
|   | O |   |
|   |   |   |
|   |   |   |

X's turn. Input move (0-8): 0
X makes a move to square 0
| X | O |   |
|   |   |   |
|   |   |   |

O makes a move to square 3
| X | O |   |
| O |   |   |
|   |   |   |

X's turn. Input move (0-8): 4
X makes a move to square 4
| X | O |   |
| O | X |   |
|   |   |   |

O makes a move to square 8
| X | O |   |
| O | X |   |
|   |   | O |

X's turn. Input move (0-8): 5
X makes a move to square 5
| X | O |   |
| O | X | X |
|   |   | O |

O makes a move to square 2
| X | O | O |
| O | X | X |
|   |   | O |

X's turn. Input move (0-8): 7
X makes a move to square 7
| X | O | O |
| O | X | X |
|   | X | O |

O makes a move to square 6
| X | O | O |
| O | X | X |
| O | X | O |

It's a tie!
```

Playing against Perfect AI (O wins)

```
It's a tie!
Do you want to play again? (y/n): y
Choose your opponent:
1 = Random AI
2 = Perfect AI
Enter the opponent's number (1 or 2): 2
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
O makes a move to square 3
|   |   |   |
| O |   |   |
|   |   |   |

X's turn. Input move (0-8): 1
X makes a move to square 1
|   | X |   |
| O |   |   |
|   |   |   |

O makes a move to square 0
| O | X |   |
| O |   |   |
|   |   |   |

X's turn. Input move (0-8): 5
X makes a move to square 5
| O | X |   |
| O |   | X |
|   |   |   |

O makes a move to square 6
| O | X |   |
| O |   | X |
| O |   |   |

O wins!
Do you want to play again? (y/n): n
Hope you had fun!
```

References

YouTube. (2022). *Build a Website in only 12 minutes using Python & Streamlit*. *YouTube*. Retrieved June 7, 2023, from https://www.youtube.com/watch?v=VqgUkExPvLY.

YouTube. (2020, December 29). *How to deploy a Streamlit app - part 4*. YouTube. https://www.youtube.com/watch?v=B0MUXtmSpiA&t=179s

YouTube. (2021, October 14). *Quantum Tic Tac Toe with IBM Qiskit and streamlit | quantum superposition | quantum computing*. YouTube. https://www.youtube.com/watch?v=DeFzlOJeWI0&t=1633s

YouTube. (2020a, December 9). *12 beginner python projects - coding course*. YouTube. https://www.youtube.com/watch?v=8ext9G7xspg

Hurbans, R. (2020). In Grokking artificial intelligence algorithms (pp. 83-85). Manning Publications.