

Otholo AI Project

Members:

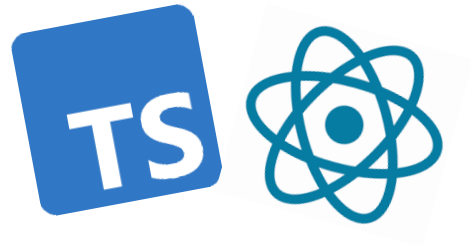
Student's Name	Id
Mansour Mohamed	1901567

GitHub Repo: <https://github.com/Mans1611/Otholo>

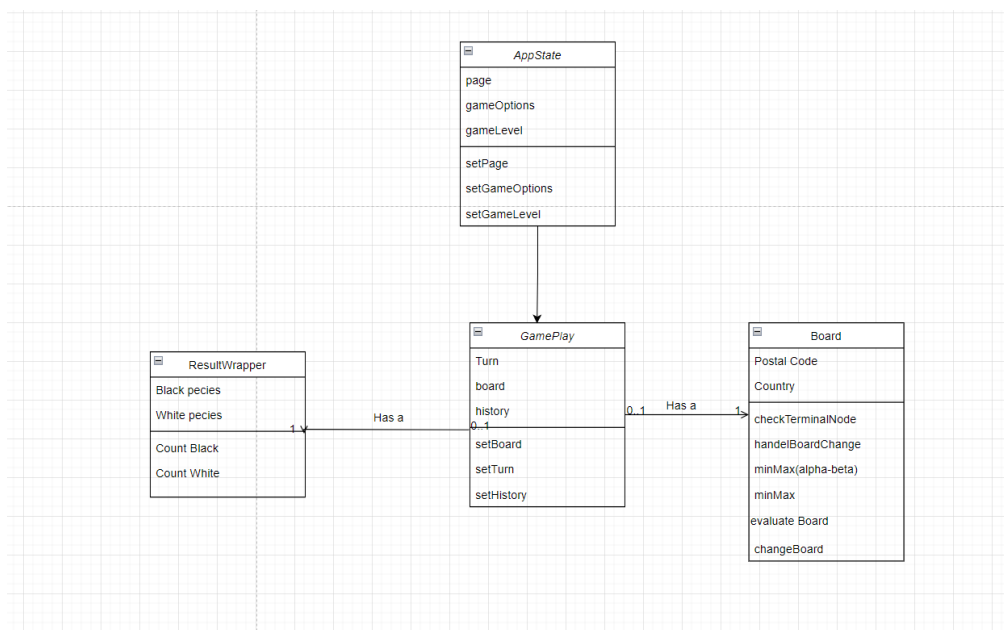
Video Link: <https://youtu.be/GJihYpFWN7g>

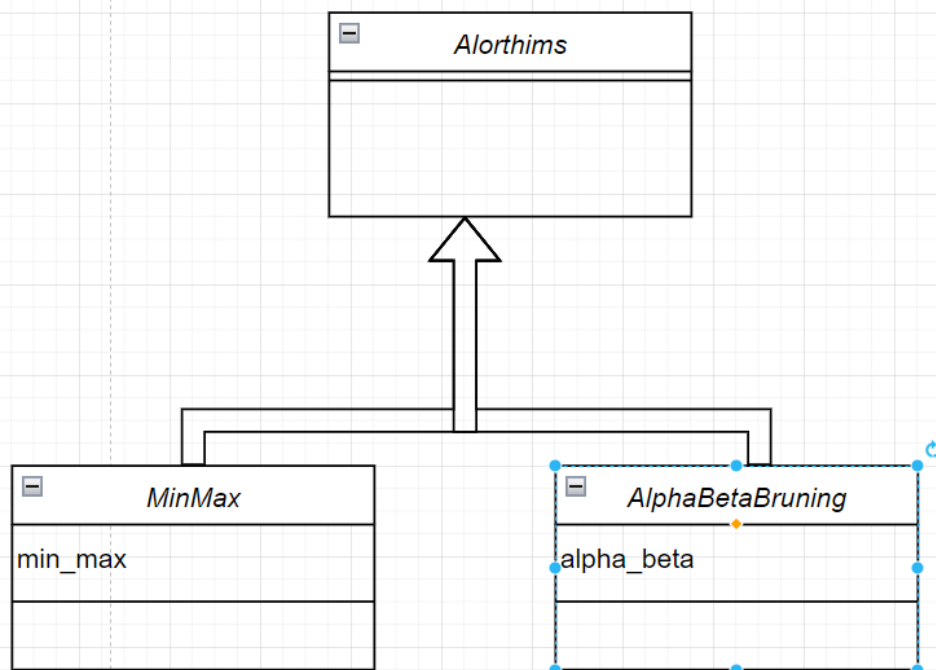
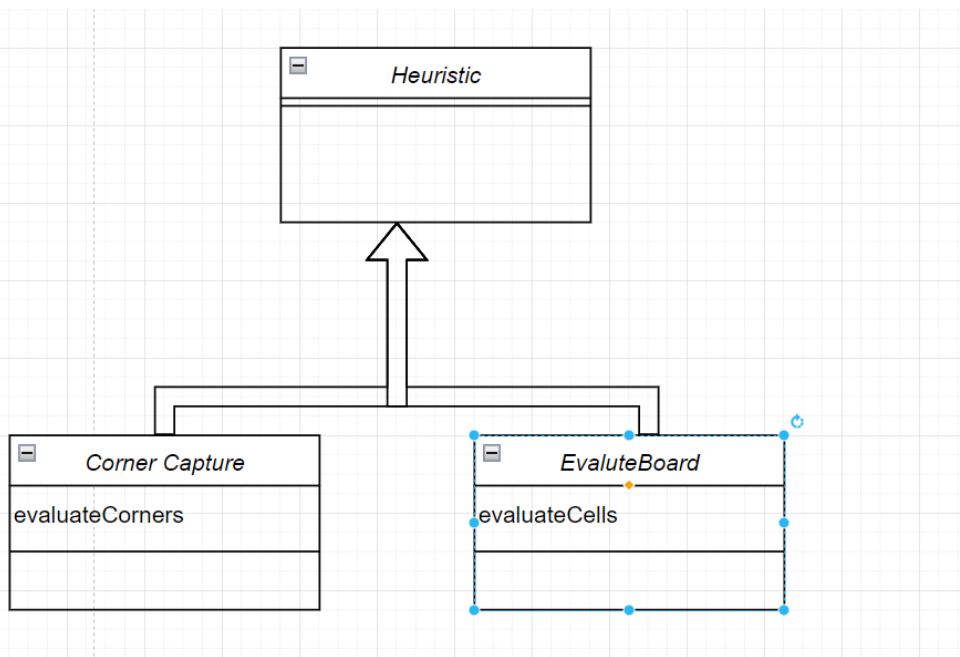
Programming Language and Frameworks:

- JavaScript
- TypeScript
- React
- SaSS



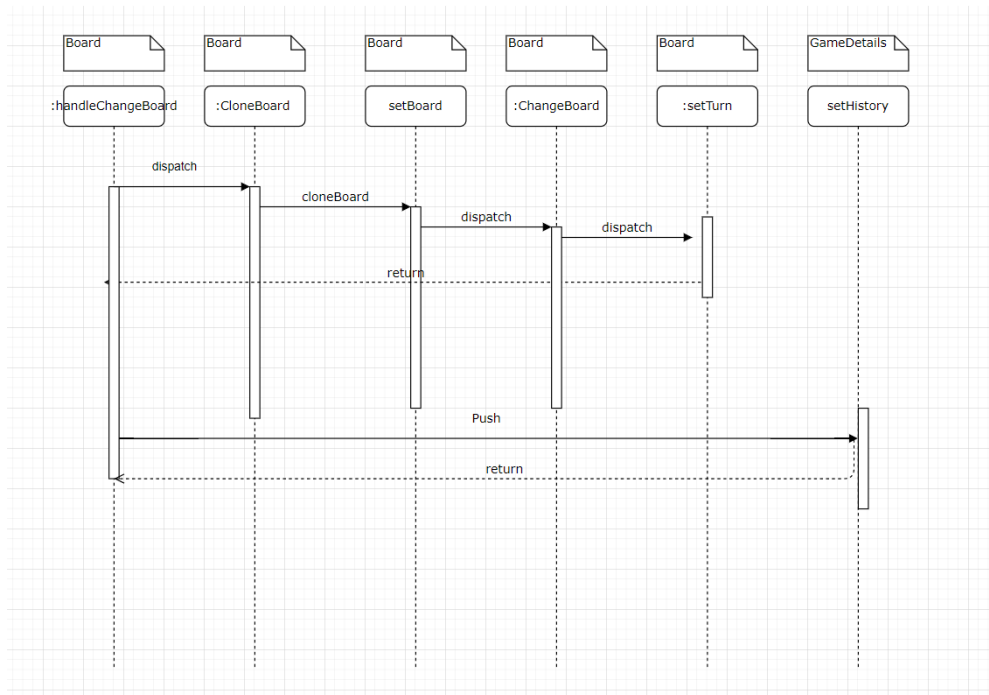
Class Diagram:



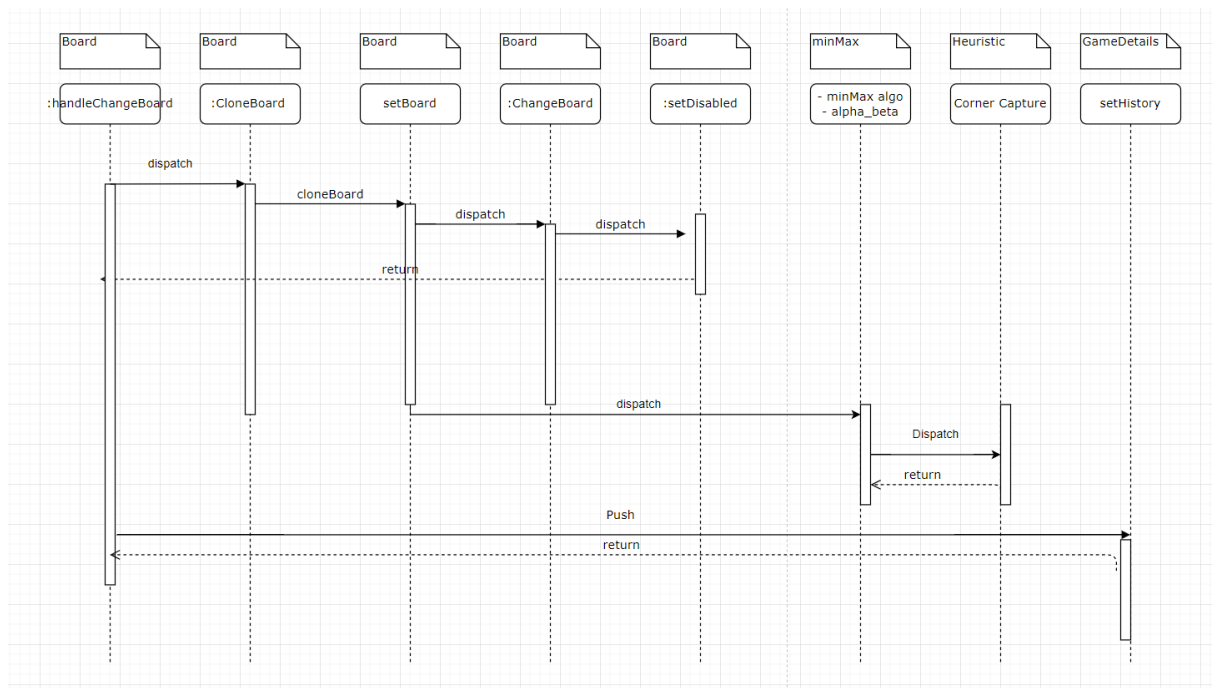


Sequence Diagram:

- For Human Vs Humans :



- For Human Vs AI :



Algorithms:

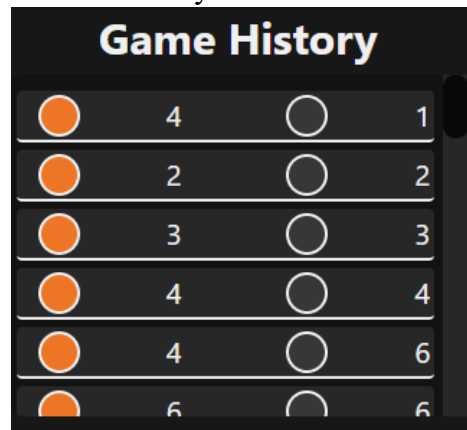
- Alpha-beta pruning:
- Min Max













Difficulty:

- Easy (depth=1)
- Hard (depth=2)

Features of the game:

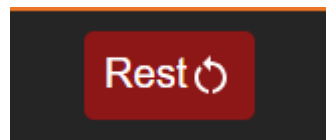
- Provide a history, that shows the history of the game, so the user can get back to, and show the status of this history state.



	4		1
	2		2
	3		3
	4		4
	4		6
	6		6

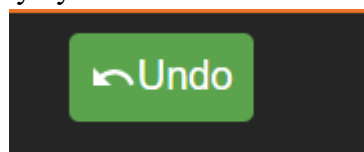
- Rest the game:

This feature rest the game, return to the initial board, erase the history of the game.



- Undo :

This feature get back by just one step back, and if we were in HvH (human vs human), I will reverse the turn, else I will not, beside that it will pop the stack of history array by one.



Heuristic:

- Corners Capture:

It gives the corner higher value than other cells,

```
export const evaluateCorners=(board:BoardType)=> {  
  // this heuristic is being used in HvC only  
  const cornerValue = 100; // here i give a high value for the corners.  
  // Define corners
```

```

const corners = [
  [0, 0], // top left
  [0, 7], // top right
  [7, 0], // bottom left
  [7, 7] // bottom right
];

let playerScore = 0;
let computerScore = 0;
for (let row = 0; row < board.length; row++) {
  for (let col = 0; col < board[row].length; col++) {
    if (board[row][col] === 1) {
      playerScore++;
    } else if (board[row][col] === 0) {
      computerScore++;
    }
  }
}

// Add corner control values
for (const [row, col] of corners) {
  if (board[row][col] === 1) {
    playerScore += cornerValue;
  } else if (board[row][col] === 0) {
    computerScore += cornerValue;
  }
}

return playerScore - computerScore;
}

```

- Difference Pieces : The heuristic that I am using is the difference between the white(orange) and black circles.

```

export const evaluateBoard=(board:BoardType)=> {
  // the heuristic that is being used.
  // Example: evaluate based on the difference in number of pieces
  const playerPieces = board.flat().filter(piece => piece === 1).length;
  const computerPieces = board.flat().filter(piece => piece === 0).length;
  return playerPieces - computerPieces;
}

```

Maximum difficulty:

Maximum difficulty occurs at hard level of the game, in which the deep of the search tree is 3, at this level it is become more difficult to beat the AI version of this.

How to Start the project:

- You need to have node to be downloaded into your machine.
- Make a clone of the repo in your local machine.
- Type this command in the cmd:
 - `cd #ProjectFolderName`
 - `npm run start`