Connect AI

Ren Luca
Member
Fantastic Four
Aubervilliers, France
renluca.pro@gmail.com

Bouillette Nicolas
Member
Fantastic Four
Garges-les-Gonesse, France
bouillette.nicolas@outlook.fr

Robidou Lucas
Member
Fantastic Four
Lanester, France
luc.robidou@gmail.com

Schirinzi Francesco
Member
Fantastic Four
Geroldswil, Switzerland
schirfr1@student.zhaw.ch

Abstract—**During the second semester at Hanyang University, we were asked to make a project that involved both Software Engineering and Artificial Intelligence. Our team decided to make a game, and we chose to implement "4 wins" (also known as "Connect Four"). This game is a turn-based game, which involves two players trying to connect four of their checkers in a row while preventing their opponent (the other player) from doing the same. For our project, we will only be able to play against the AI. The goal of this project is to obtain a Connect 4 AI as good as possible inside the timeframe and scope of this project to obtain experience working with multiple algorithms with different levels of difficulty.**

I. INTRODUCTION

Machine learning (ML) and Artificial Intelligence (AI) are becoming more and more advanced and are widely used in many different aspects. The applications of ML and AI are becoming a bigger part of our everyday lives. For example, when you're typing on a smartphone and it gives you a suggestion on the next word based on the first part of a sentence, or when your camera auto adjusts its settings based on the image that it is currently seeing.

Only a decade ago, it was thought nearly impossible that a machine would be able to beat world champions in advanced strategy games such as Go, but now Go has now been convincingly defeated by AIs without human inputs. Reinforcement learning AIs can now start from random strategies playing against itself and achieve superhuman play tactics in less than 24 hours without any information from domain expects only using the game rules.

As the world is opening up for the usefulness of ML and AI, a need for more people understanding and working within this area is also essential. We are a group of students currently studying at Hanyang University that wants to extend our knowledge within this field. We are therefore building an AI playing Connect 4. This game is a turn-based game, which involves two players trying to connect four of their checkers in a row while preventing their opponent (the other player) from doing the same.

This game will be playable through a web interface, and the outcomes will be saved on the server. Thus, we will be able to let the AI play against itself and train itself until a certain time has passed. Players will then be able to face the AI in the game.

II. REQUIREMENTS

The requirements are splitted in two main parts. The ID's from 0-49 are reserved for the game start and the options one can set only at the start. Further, the ID's from 50-100 are designated for the requirements during a game.

- [ID 1] The user can start a game with option for the difficulty
- [ID 2] The user can choose a difficulty
- [ID 3] The player can load an old game with an ID
- [ID 50] The player can click on the QUIT button to go back to the main menu.
- [ID 51] The player can click on the go back button to revert the play he and the computer made for the previous turn.
- [ID 52] The player can click on a column to place a token

III. DEVELOPMENT ENVIRONMENT

A. Choice of software development platform

For the platform, we will be using Linux OS because we're accustomed to it and it's a open source OS, it is also easier to install certain programming software on it.

For the Frontend, the team decided to use VueJs, a Javascript Framework, for creating a SPA (Single Page Application). VueJs has a good learning curve, because it doesn't have a large footprint but can provide important functionalities. Besides VueJs itself and the styling package BootstrapV4, two very popular libraries - Vue-Router and Axios - are used in this project.

For the back-end part, it has been decided to use Python3 along with Bottle, a web framework, to provide an API for the front-end. Python has been chosen for its simplicity, while Bottle is used because it does not require any dependency and is fast and simple to use.

To resume, we will be using Javascript 1.8.5, python3.7, sklearn (scikit) version 0.21.3. Our OS are mainly debian 10 and ubuntu 18.04. We chose those version because they're either the latest stable version or because we are accustomed to it.

Apart from an (optional) server, our project is totally free of charge, since most of our tools are free. Thus, there will only be labor cost. Estimations:

- Front-End - 80h
- Back-End - 100h
- Document - 50h

B. Software in use

Two different methodologies are implemented and analysed in this project - Minimax and AlphaZero. The Minimax algorithm is a heuristic that is fine tuned to perform well in the Connect 4 game, whereas AlphaZero is a more advanced AI that trains Neural Network using a search heuristic. The two methods are explained below starting with the Minimax algorithm.

Minimax Algorithm:

Minimax (MM) is a backtracking decision rule that is implemented for outcome optimization of decisional problems in various fields, such as decision theory, game theory, statistics, but also (and more importantly for us) in artificial intelligence. MM is based on the assumption that each player always plays optimally in his own interest and in perfect rationality ("homo economicus").

This algorithm is applicable to the Connect 4 game because it is a turn-based two players zero-sum game. Zero sum game represents a situation in which each participant's (player) gain or loss (in this case the score of the board state, and eventually the win or loss outcome) is exactly balanced by the losses or gains of the other participants (player). In "poor words", the choices a player can make are directly dependant from the previous choice(s) of the other player, and vice versa.

In Minimax the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible. The maximin value of a player is the highest value that the player can be sure to get without knowing the actions of the other players; equivalently, it is the lowest value the other players can force the player to receive when they know the player's action.

The minimax value of a player is the smallest value that the other players can force the player to receive, without knowing the player's actions; equivalently, it is the largest value the player can be sure to get when they know the actions of the other players.

The intrinsic perfect information of this game, allows to evaluate (assign a score) each possible board state and, more importantly, all possible consequent board states derived from all possible future states! If we represent this as a tree, the root would represent the first board state (only one piece from one player), and from there, for each turn (piece) we would generate seven children, each representing the board state for each column where the other player could drop the next piece. Each of these seven children, would have further seven children evaluating the possible moves derived from them.

With this technique, the "computer" is able to "pre-play" all possible moves on the board and choose the score maximising strategy (path from root - current state- to leaf representing winning the game) for each of the opponent's moves. The superior computational power of the computer allows it to think ahead of all possible moves, and the opponent (human being) is left with the task of minimizing his loss.

AlphaZero:

AlphaZero is a single system that can teach itself from scratch how to master the games of chess, shogi, and Go at superhuman level. AlphaZero was developed by DeepMind in 2017 and it gained a lot of attention because of the revolutionary results it obtained within the reinforcement learning (RL) area. In just 4 hours of training the AlphaZero AI was able to beat Stockfish (the former chess master AI) in chess starting from a random strategy only playing against itself. The AlphaZero system is better at aiming its search for the best possible strategy than former systems. The AlphaZero system contains three key components: Deep convolutional residual neural network (ResNet) Monte Carlo Tree Search (MCTS) Reinforcement Learning (RL) evaluation to improve the performance of the neural network (NN)

IV. SPECIFICATIONS

- [ID 1] The user can start a new game, the option for the difficulty currently selected will be chosen for the difficulty of the AI.

```
in cases difficulty is:
        easy then ai.load_easy()
        normal then ai.load_normal()
        hard then ai.load_hard()
game.start()
```

- [ID 2] The user can choose a difficulty in the main menu.

```
select option_difficulty:
        difficulty is easy
        difficulty is normal
        difficulty is hard
```

- [ID 3] The player can load an old game with an ID, if doesn't exist, create new game.

```
if id_valid(id) then:
        game.load(id)
else:
        game.start(id)
```

- [ID 50] The player can click on the QUIT button to go back to the main menu.

```
If player.quit():
    game.exit()
    game.showMainMenu()
```

- [ID 51] The player can click on the go back button to revert the play he and the computer made for the previous turn.

```
if player.resetTurn():
    ai.cancelLastMove()
    player.cancelLastMove()
```

- [ID 52] The player can click on a column to place a token

```
if game.valid_move():
        game.update()
else
        game.show_error()
```

V. Architecture Design & Implementation

In this section one can find every information to work on this project.

We described how the files are split up in the directories, how the different application communicate together and how the applications were designed.

A. Overall architecture

Both applications are run locally. But the apps can also be deployed without much effort to a server.

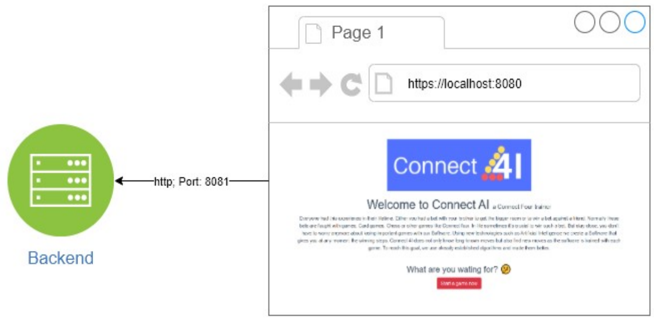


Fig. 1. Overview of architecture.

B. File Organization

TABLE I
Overview of the GitHub Repository root folder.

File/Folder	Description
README.md	Documentation
REST-API.md	Documentation
doc	The document as Latex version
ressources	Assets (Diagrams, ...) that have to be versioned.
src_backend	See file organization in the corresponding section.
src_frontend	See file organization in the corresponding section.

C. Backend

Design Overview

Bottle functions
Provides Bottle functionalities (launch the web server, handle the request from the front, etc.).

Game logic functions
Provides functions related to the logic of the game, the rules, the goal of each player. . . .

Ai modules
Translate the game logic into something the two AIs can understand.

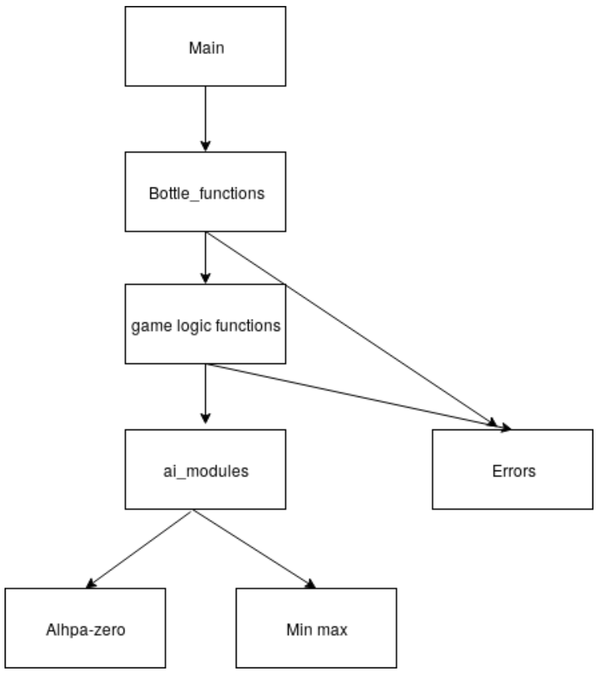


Fig. 2. Overview of Back-End

As the projects we used for the AIs use different coding conventions, we had to translate our variables into something those projects could use. That is why the main module do not import directly the AIs, but actually uses ai_modules that do it partially.

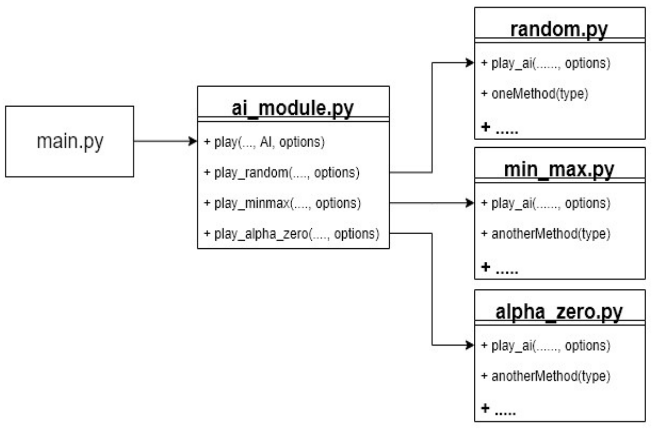

main.py
ai_module.py
+ play(..., AI, options)
+ play_random(...., options)
+ play_minmax(...., options)
+ play_alpha_zero(...., options)
random.py
+ play_ai(......, options)
+ oneMethod(type)
+ .....
min_max.py
+ play_ai(......, options)
+ anotherMethod(type)
+ .....
alpha_zero.py
+ play_ai(......, options)
+ anotherMethod(type)
+ .....

Fig. 3. Adapter paradigm

TABLE II
FILE ORGANIZATION BACK-END

File/Folder	Description
src_backend/ src/bottles_functions.py	The function that handle the web requests
src_backend/ src/errors.py	A module containing the definitions of errors classes
src_backend/ src/game_logic_functions.py	Functions related to the logic of the game
src_backend/ src/main.py	The main file to execute in order to launch the server

Ais classes :

- AI_test : an ai that is used to test the game
- AI_easy : an AI that play randomly
- AI_medium : an AI that uses Minimax algorithm
- AI_hard : an AI that uses AlphaZero algorithm

Bottle functions :

- from_list_to_json : take a list and turn it into a dictionary that can be parsed by the front
- start_game : gives an uuid (unique identifier) to the front
- load_game : Load a game asked by the front and send it. If the game asked by the front does not exists, it is created and saved on the disk.
- play_game : registers the move of the player on the disk and send back the move of the ai
- undo_move : undo the last move of the player. Should the last player be the AI, undo the AI's move and the last player move.

Errors :

- moveNotValidError : error raised when the player made an illegal movement
- aiCantMoveError : error raised when the AI can't find a way to play (eg: the board is full)

Functions related to the logic of the game :

- is_move_valid : check if a move is valide (ie: the chosen position is empty and all position below are already taken)
- place_token : Place the token in the grid and return the grid. This function can raise raise moveNotValidError if the move is not valid.
- horizontal_check, vertical_check, diagonal_left_check, diagonal_right_check : as their names indicate, check if there is a winner horizontally, vertically and diagonally
- victory : checks if someone won the game

D. Frontend

We use Vue-Router to provides the functionality to create a SPA (Single Page Application).

Axios is a HTTP request library that contains more features besides the default Javascript Fetch-Functionality. For development purposes, the possibility to mock HTTP-Request (Catch and return custom offline data) was implemented. Therefore, the frontend could be tested independently from the backend.

Concerning the deployment and access, For the project we used to functionality of VueJs. It implements through a packe a standard HTTP-Server.

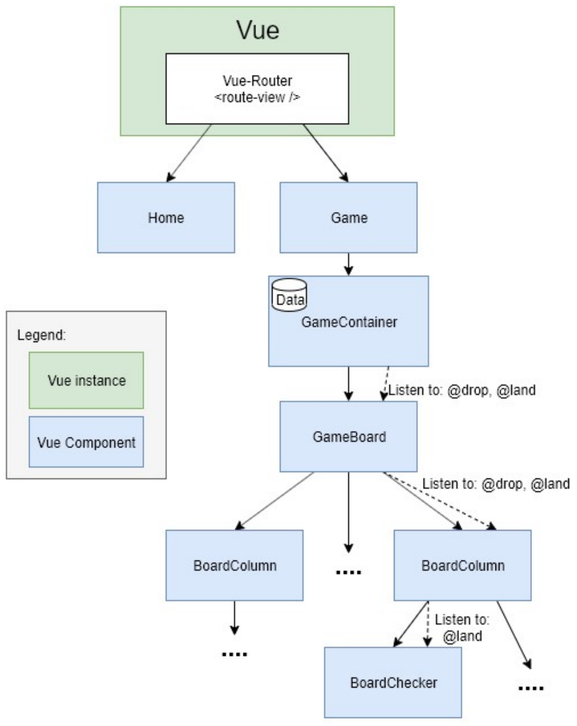
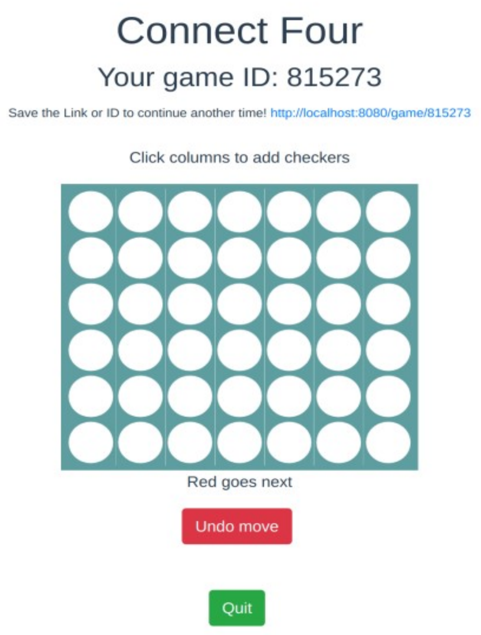


Fig. 4. Component Design Overview

TABLE III
FILE ORGANIZATION BACK-END

File/Folder	Description
node_modules	Auto generated, when the package dependencies are installed
public	Contains all assets that are included in the build. This assets are publicly accessible
src	Contains all the JavaScript files
src/*/*.vue	Vue Components
src/*/main.js	Entry point of the VueJs app
srv/*/*.js	Helper function files
.env	File to set environment variables for the app
babble.config.js	Configuration for the JavaScript compiler
package.json	List of all required packages
vue.config.js	Configuration file for VueJs

E. Game run example


Connect Four

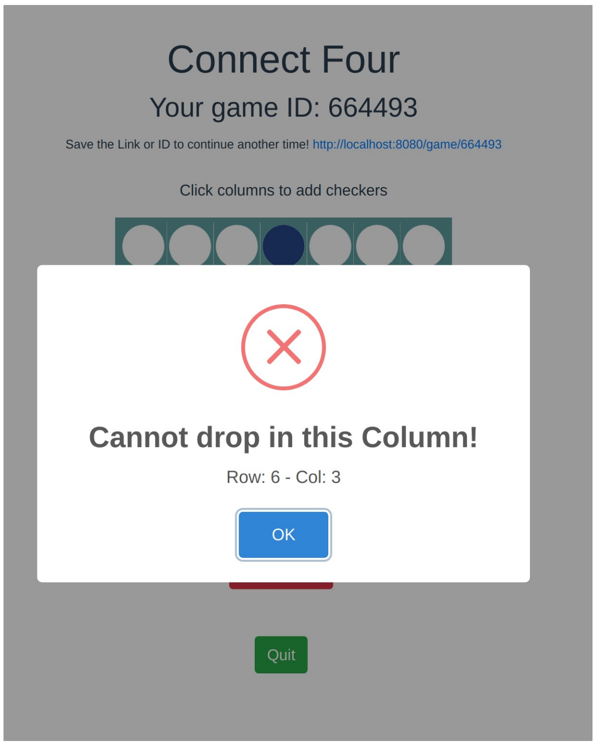
Your game ID: 815273

Save the Link or ID to continue another time! http://localhost:8080/game/815273

Click columns to add checkers

Red goes next

Fig. 5. Start of game


Connect Four

Your game ID: 664493

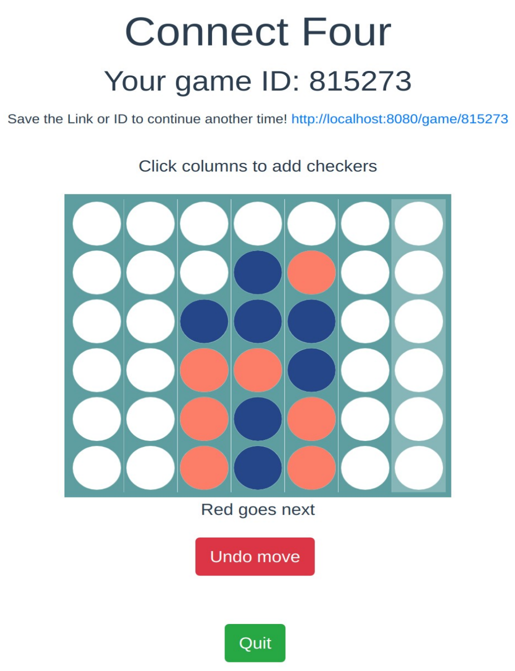
Save the Link or ID to continue another time! http://localhost:8080/game/664493

Click columns to add checkers

Cannot drop in this Column!

Row: 6 - Col: 3

Fig. 7. Impossible move error


Connect Four

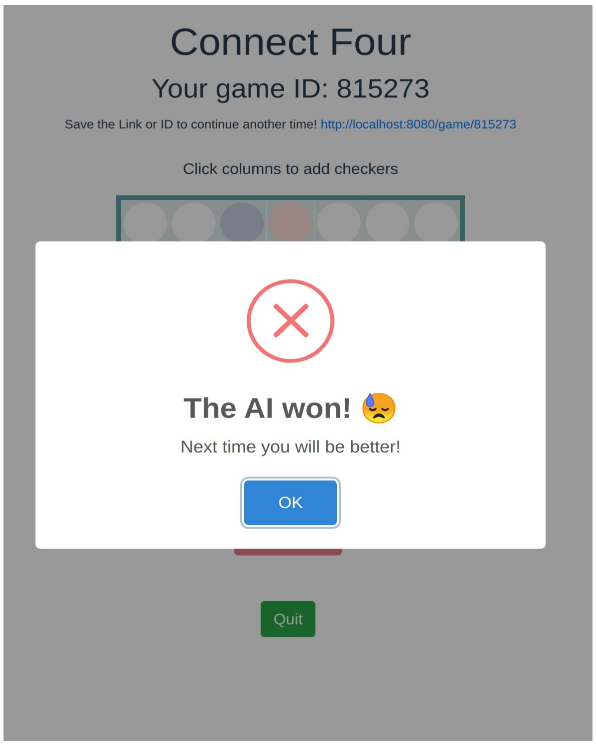
Your game ID: 815273

Save the Link or ID to continue another time! http://localhost:8080/game/815273

Click columns to add checkers

Red goes next

Fig. 6. Game in progress


Connect Four

Your game ID: 815273

Save the Link or ID to continue another time! http://localhost:8080/game/815273

Click columns to add checkers

The AI won! 😓

Next time you will be better!

Fig. 8. Game lost

VI. CASES

A. *Use case*

- [ID 1] The user can start a game with option for the difficulty
- [ID 2] The user can choose a difficulty
- [ID 3] The player can load an old game with an ID
- [ID 50] The player can click on the QUIT button to go back to the main menu.
- [ID 51] The player can click on the go back button to revert the play he and the computer made for the previous turn.
- [ID 52] The player can click on a column to place a token

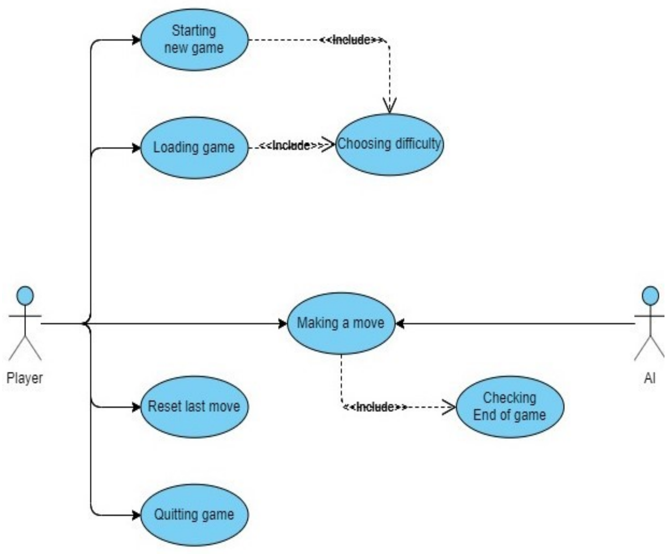
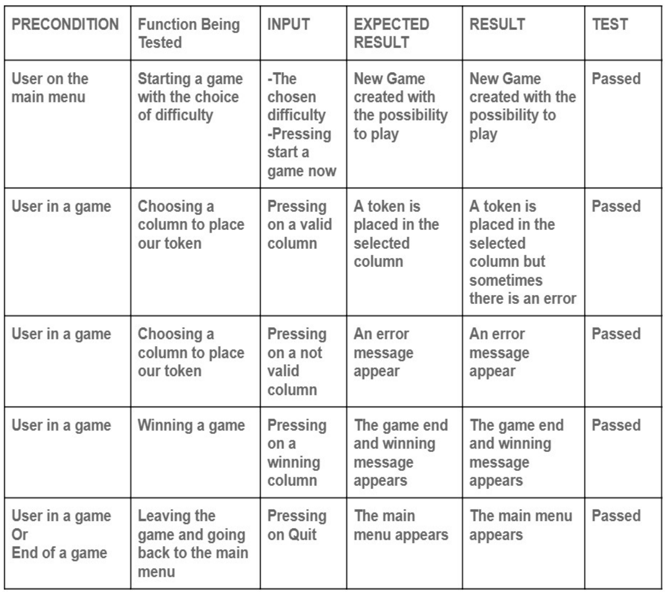


Fig. 9. Use Case Diagram

B. *Test case*



PRECONDITION	Function Being Tested	INPUT	EXPECTED RESULT	RESULT	TEST
User on the main menu	Starting a game with the choice of difficulty	-The chosen difficulty -Pressing start a game now	New Game created with the possibility to play	New Game created with the possibility to play	Passed
User in a game	Choosing a column to place our token	Pressing on a valid column	A token is placed in the selected column	A token is placed in the selected column but sometimes there is an error	Passed
User in a game	Choosing a column to place our token	Pressing on a not valid column	An error message appear	An error message appear	Passed
User in a game	Winning a game	Pressing on a winning column	The game end and winning message appears	The game end and winning message appears	Passed
User in a game Or End of a game	Leaving the game and going back to the main menu	Pressing on Quit	The main menu appears	The main menu appears	Passed

Fig. 10. Test Case Tab

VII. DISCUSSION

During this project the work was divided between each member, 1 on the front-end, 2 on the back-end and 1 for the document. For communication, we had a WhatsApp group and a Discord group, although we had multiple means to contacts each other, it could take multiples hours or even days to get an answer because we all had different schedule.

We found that playing multiples roles: the client, developer, etc. was pretty hard. Indeed, creating a project, defining its limits and coding, doing all of those was quite difficult and sometimes we felt like we didn't know where to go.

One of the big problems was coding a computable script for everyone. Not everyone was working on the same distributions or with the same version of languages, some functions could cause errors for one and work for others.

Also, the logic approach for the use case diagram was hard: in fact, being careful not to show temporality is for us counterintuitive and this sometimes blocked us in our reasoning.

For the front-end we used Vue.js, a framework that some member never used before, so obviously they didn't know how it works. Each time they had a question about the front-end, they needed to wait for someone who knew the answer. For other projects, they were able to look into the code themselves, but here, it was impossible. It slowed them down quite a bit. Furthermore, the fact that some people were only attending one course (AI) gave us a hard time, as their viewpoint on the project was slightly different from us.

To conclude, the project wasn't too easy but also not too difficult. Developing the frontend was fairly easy as we had someone who was used to it. In the late project lifetime, we had to do a lot in the backend and it was a little time-consuming because some member normally don't use python. In addition, we also had some tricky problems. One was, that we couldn't start the AlphaZero AI because of incompatibility with the GPU-Drivers in combination with the Python version which had a conflict with a required package. Nonetheless, we could figure out how to solve the problem and make everything run.