Table 1: Revision History

| Date | Developer(s) | Change |
|---|---|---|
| Oct $1^{th}$ | All | Initial division of labour |
| Oct $3^{rd}$ | All | Rough draft complete |
| Oct $5^{th}$ | All | Revision and proof-reading |
| Jan $23^{rd}$ | Jeffrey Dang | Changed placement of terms and terminology |
| April $2^{nd}$ | Michael Ilao | Rev1 updated document from instructor feedback + peer reviews |

# Software Requirements Specification for SE 4G06 An AI-based Approach to Designing Board Games

Team #6, Board Gamers
Ilao Michael, ilaom
Bedi Hargun, bedih
Dang Jeffrey, dangj12
Ada Jonah, karaatan
Mai Tianzheng, mait6

April 6, 2023

# Contents

# 1 Reference Material

## 1.1 Abbreviations and Acronyms

| symbol | description |
|---|---|
| SRS | Software Requirements Specification |
| AI | Artificial Intelligence |
| A | Assumption |
| LC | Likely Change |
| FR | Functional Requirement |
| NFR | Non Functional Requirement |
| FSM | Finite State Machine |
| TA | Teaching Assistant |

## 1.2 Terminology and Definitions

This section is expressed in words, not with equations. It provides the meaning of the different words and phrases used in the domain of the problem. The terminology is used to introduce concepts from the world outside of the mathematical model The terminology provides a real-world connection to give the mathematical model meaning.

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- **AI Agent**: Refers to the subsystem that has an AI model trained to play the game at hand acting as one of the players of the game.
- **Game Engine**: Refers to the subsystem that is an abstract representation of the actual game as software.
- **Data Visualization**: Refers to the subsystem that visualizes Game Engine and AI Agent logs.
- **Game State**: Refers to the state of the game, which could include player attributes, player score, and game layout. All attributes and characteristics that change throughout a game simulation are stored here.
- **Observation Space**: Refers to the state of the game that is observable to a Game Agent. (Not all information is available to the Game Agents and can vary from agent to agent)

- **Action Space**: Refers to the set of moves an AI Agent can take in a particular Game Engine.
- **Game Designer**: The game designer is the person who is creating the game by coming up with the rules and writing a scenario for the game. The game designer will be the end user of the system to balance the game and improve the design of the game.
- **Finite State Machine**: A finite state machine is a computation model that can be used to simulate sequential logic by using state, actions and transition functions that decide how a given action and state will transition to a new state. bri

# 2 Introduction

## 2.1 Purpose of Document

The purpose of this document is to define the needed features and intended behaviors behaviours of using an AI-based approach to designing board games. Both the users and software developers use this document as an agreement to involve important information such as system specification, goals of implementation, project constraints, functional and non-functional requirements. This document help helps software developers understand what the game users need, minimize the cost of time and money, split work into multiple smaller pieces, and get project development started on the right path.

## 2.2 Scope of Requirements

The scope of our project is to create an AI-based, game simulation engine that can help identify pitfalls in a game's mechanics and imbalances in the game's scoring system. Our intended users are game developers who wish to use our application to develop a robust game. The AI engine, Game Engine, and the simulator will work together to create output for a decision tree visualization tool that will help the game developers identify various paths of decisions that led to specific end states.
We have initially limited our scope to only have our application work with 2 games- : Tic-Tac-Toe, and Bellow Intent's Age Contrived gameBellow Intent's Age Contrived game, AnA . As a secondary goal, we will work towards having our application be an open-source framework such that it is possible for any game developer to use our system.

## 2.3 Characteristics of Intended Reader

The intended readers of this document are Dr. Spencer Smith, Christopher Schankula (TA), Dr. Sebastien Mosser, and Dr. Vladimir Reinharz. This document is thus intended for an audience that has at minimum a high-level a technical understanding of game development, artificial intelligence, game simulation, and interfacing between low-coupled systems. As a result, the reader is expected to know the basic terminologies either through their own research or from an undergraduate level undergraduate-level course in all of the fields specified. This document will also be used by the developers of this project (members of Group 6), therefore, it is also intended as a guide for all future documents and milestones during the development of this project.

## 2.4 Stakeholders

Table 2: Stakeholders

| | | |
|---|---|---|
| Bellows Intent | Publisher of The Game"An Age Contrived" , *An Age Contrived* | Primary |
| Chris Matthew | Principal Author of "An Age Contrived" and Designer of *An Age Contrived* and The Game Designer | Primary |
| Dr. Sebastien Mosser | Supervisor | Secondary |
| Dr Vladimir Reinharz | Supervisor | Secondary |
| Other Game Designers | Potential Users | Tertiary |
| Group # 6 | Developers | Tertiary |
| Players | | Tertiary |
| Society | | Tertiary |

## 2.5 Organization of Document

This SRS document is intended to be read in the order that the document is presented. The main sections are – scope, general system description, specific system description, functional and non-functional requirements, likely and unlikely changes, and traceability matrices.

The template of this document follows the 4G06 SRS template provided by Dr. Smith Smith. Below are the changes to this template

- **Table of Units**: Removed
- **Table of Symbols**: Removed
- **Mathematical Notation**: Removed
- **Terminology and Definitions**: Moved to Section 1
- **Stakeholders**: Added
- **Normal Operation and Undesired Event Handling**: Added

- **Solution Characteristics Specification**: Replaced by Assumptions and Required Behaviour Overview
- **Likely Changes**: Combined into Likelihood of Changes for Functional Requirements
- **Unlikely Changes**: Combined into Likelihood of Changes for Functional Requirements
- **Impact Analysis**: Added
- **Values of Constants**: Added
- **Development Plan**: Removed
- **Reflection**: Added

A table of contents is explicitly stated on page 3 with a table of contents to prime the reader on what is available in the document.

# 3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

## 3.1 System Context

Figure 1

### 3.1.1 System Architecture

Figure 1 shows the system architecture and how the different agents and components interact with their main inputs and outputs. Exact information is abstracted away as the architecture should allow for "swappable" change and swap of modules. Meaning if that different Game Engines and different AI Agents can be interchanged and the functionality of the system will stay the same, with different outcomes and performance. In this system user input is not given and the AI Agents act as "users " users that interact with the Game Engine giving commands and receiving data to make those decisions. There are two outputs from the Game Engine, first the new game state from a given move and then a log of the actions and state. This log is then read by a data visualizer that the real end user can view.

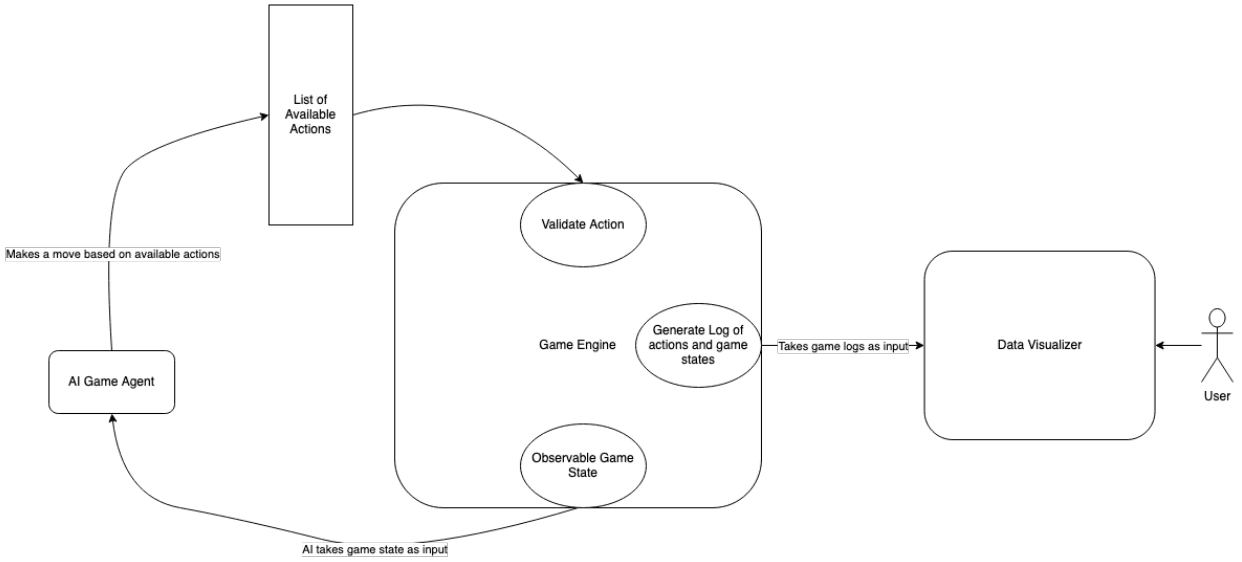


Figure 1: System Context

### 3.1.2 Game Engine Subsystem

Inside the Game Engine, there will be a finite state machine an FSM that controls the main game loop. The finite state automata FSM will be comprised of variables $< Q, \sum, \delta, q_0, F >$

- $Q$: The set of states, for many systems will be almost infinite as there are many different states the game can get to. Tic-Tac-Toe for example has only 23 unique winning states, while a game like chess has almost infinitely many, .
- $\sum$: The alphabet is a list of all possible moves from one state to another.
- $\delta : (Q, \sum) \rightarrow Q$: The transition function that will take the current state and a possible move and output the new updated state.
- $q_0$: The starting state, for Tic-Tac-Toe would be an empty board, for Monopoly would be an empty board, and all players with their starting money.
- $q_1$ The state after the game has began and will be looping through continuously until the game has concluded.
- $q_2$ The final state afterwards the winning move is made in the game and concludes the system.
- $F \subset Q$: The set of final states, this will be all the winning completed end game states for a player. , in this case $q_2$

To simplify the set of states, only three will be used in this diagram: start, finish and in progress. These will be denoted as $q_0, q_1, q_2$ respectively. Although $q_1$ and $q_2$, can be comprised of many sub-states.

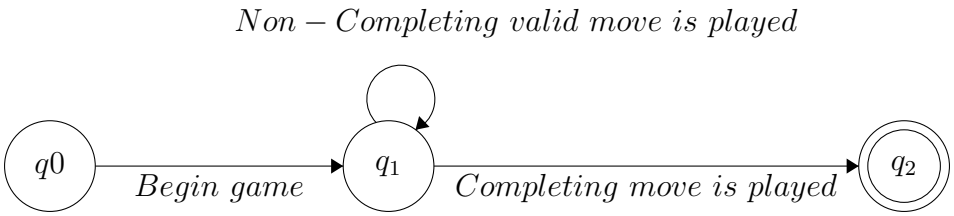


Figure 2: Finite State Machine for Game Loop

The game will continue until a winning game completing move is played and loops after each non-winning non-completing move, and all non-valid moves are rejected. Based on the game non-valid moves could result in the game being forfeited and lost, for this simple example, the Game Engine will reject the AI Agent's move and wait for a valid one. ones.

Since our architecture will support multiple game engines, the FSM is generalized. As the FSM can be as simple as tic-tac-toe which will not have as many states as our complex game, *An Age Contrived*. It is not feasible to fully describe either game in detail here as the diagram for the FSM will be too large too fit. As seen here Erikras, a tic-tac-toe game has a possible 5,478 game states to track. The way $q_1$ will work is it will contain all possible states that the game can be in, in which it is not completed and the connections between each state are the game actions taken to travel between states. Now take this simpler version of tic-tac-toe in Figure 3 where a player needs to get 2 in a row. (Some final states are repeated, due to simplicity and easier to visualize). This simple game can be described as.

- Q: - - -, X - -, - X -, O X -, etc... (All combinations of 1 X, 1 X and 1 O, and 2 X's and 1 O).
- $\sum$: L, M, R, where to play the next character: left, middle or right
- $\delta$: Transition function will take the state and action, and place an X or O, depending who's turn it is in that space.
- $q_0$ our initial state is - - -
- $F$: are all the states with 3 characters (X,X,0). For example XXO, OXX, etc...

Even with this simple game the state space grows large, but it is an efficient way to track all possible states and how different actions take a player from state to state. Our AI will need to keep track of state and actions being taken as that is what our AI will learn from. Our AI will take each state as input and according to its decision making process, it will output an action, as seen in Figure 2.

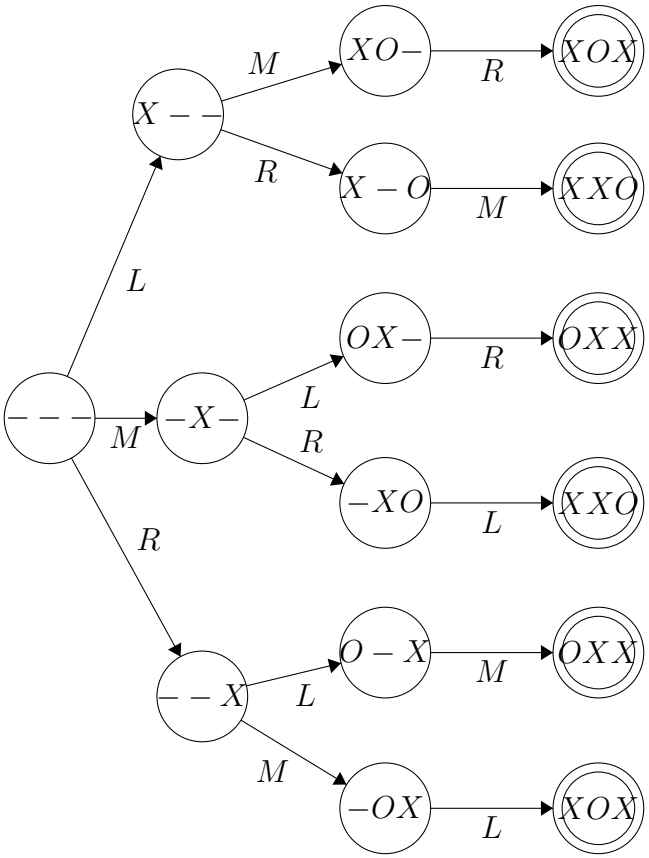


Figure 3: Finite State Machine for Simple Tic Tac Toe

### 3.1.3 Responsibilities

**User Responsibilities:**

- The Users will observe the data visualizer of game simulationsthat will display game simulation metrics.

**System Responsibilities:**

- AI Agent: Receive input of a game state and output a decisionan action.
- Available Actions: List all available moves to the AI Agents (Not an entity itself and is a part of the Game Engine)
- Game Engine: Receives inputs input of decisions made by the AI Agents and updates its update the game state accordingly. Outputs a log file of the game state and actionsand the updated , and the output the new state back to the Game AI Agents.

## 3.2 User Characteristics

The intended users of the system will be AI research professors and game designers. These users will be observing the same output of the system, but analyzing it in a different way. The AI research professors will be using the output to analyze how different kinds of AI play a specific Game Engine. Meaning these users should have a deep an understanding of AI, Neural Networks, Machine Learning, and other AI topics. The game designers will be using the output to check for any "over-powered " strategies or game-breaking rules, these users should have a deep understanding of the game rules and mechanics (For the specific Game Engine in use). For all users of the system, they should have an understanding of statistics up to Stats 3Y03, to read and understand the data output.

## 3.3 System Constraints

The main constraint of the system is it the AI and Game Engine must be developed in Python, as AI libraries are in Python are the easiest to implement and highly supported, the Game Engine and all other components . The Game Engine must be implemented in Python for simple compatibility and integration. Potential libraries to be used for AI component of the system are Gymnasium gym, Simple sim or Tianshou wel.

While Python must be used for the Game Engine and AI, the data visualizer can be developed in any language or library as coupling between the Data visualizer and the Game Engine is very low. Potential libraries for data visualizing are:

- Apache ECharts apa
- Plotly plo
- react-vis rea

## 3.4 Normal Operation & Undesired Event Handling

The system will have a main loop where all the operations will take place after the initialization until the win game completion condition is reached, then the system will exit the loop and end the current game to provide data visualization. This initialization, main loop, and win game completion condition can be modeled and think as a finite state machineas an FSM. The process is explained in detail in Section 2.1 3.1.2 and visualized in Figure 2. In the event of an unexpected outcome, since the games to be simulated are extremely complex and each move changes the state 2.

To minimize undesired outcomes, where a game reaches an illegal state, ex: tic-tac-toe, where O has more spaces than X at the end of the gamesignificantly, , whenever an AI takes an illegal action that would proceed to this state, the simulation will start over instead of any error handling. This decision is justified by the fact that each simulation should take less than 5 minutes The Speed Benchmark 7, and no undesired event will have a harmful effect on the system. Also a large negative reward will be assigned to the AI, which in turn

will make the AI less likely to take this move which causes an undesirable behaviour for the system. Thus, the cost of restarting the simulation is cheaper than any extensive undesired event handling since we would have to handle thousands of different scenarios if we were to decide to do undesired event handling rather than restarting that particular simulation. As well as training the AI to only take legal actions given a certain game state.

Actions are also only given to the AI if they are legal to take in that specific game state, this will minimize undesired outcomes as the action that needs to be taken to get to an illegal state will not be provided to the AI. For example for tic-tac-toe if an X is placed in the center, the O player will be given all the actions except the center tile to place its O. Referring back to Figure 3, the only moves supplied after taking an action are the ones that haven't been taken. If L is taken first, the only moves available are M and R. Our system will be designed in a similar fashion where actions provided to the AI will be decided on the current state, in order to prevent illegal states.

When an undesired event occurs and the simulation must restart, the output will be omitted from the game log, to ensure the data visualizer is only showing legal games that are played. Ensures all games visualized contain complete and un-corrupted data.

# 4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by definitions and assumptions.

## 4.1 Problem Description

Designing a game is not an easy task. Especially, if the game is not a softwarethat software. That can be updated with correction and balancing patches later on, but a physical board game where an incorrect game design can result in significant financial and reputation costs as it is expensive or sometimes impossible to patch a physical board game. In complex games, there are multiple players, dozens of variables and actions that can be taken, and millions of possible scenarios that each decision and variable can cause. A good game should be a balanced one and should not contain any definitive way to win the game every single time to make it more competitive and fun. However, it is also humanly not possible to go through each scenario and balance all the variables to make sure there are no loopholes in the game. Our project is trying to create a process where game designers can see most of the scenarios and balance the game before even publishing by utilizing the power of computers.

### 4.1.1 Goal Statements

Given the inputs, the goal statements are:

- **Game Simulation**: The system must simulate good quality board games containing rules and strategieswith no serious consequences of mistakes, that are not trivial ex: (Snakes and Ladders).

- **AI Learning**: The system must implement artificial intelligent players to play thousands of games efficiently to detect patterns and learn winning strategies.
- **Complex Problem Solving**: The AI game players of the system must be able to analyze complex decision paths. Each different AI game player should have a different decision-making process.
- **Quality Data**: The system must output readable and correct data to track and save decisions, game states, and patterns throughout the simulations, as well as compare simulations against each other to find winning strategies.
- **Data Visualization Analysis**: The system must provide a variety of charts to visualize the data for users.
- **Live User Interface**: The system should have a live user interface where users can view the game states in real-time and the actions that AI game players are making.

### 4.2 Assumptions

A1: The game to be analyzed will have a clear reward function that AI Agent can optimize its learning towards. This reward function will be similar to the scoring/points system in the game.
**Rationale**: The AI needs a reward to choose which moves are optimal in certain states.

A2: The game to be analyzed will have a clear win condition where the game ends.
**Rationale**: It shouldn't be an open-ended, infinite game where the goal is to just increase your score, but rather there should be a specific condition(s) that players are playing towards to win and end the game. Otherwise, AI agents won't be able to find the optimal actions to win the game but rather keep outputting sub-optimal decisions which will have little to no value for game design improvements.

A3: The game to be analyzed will have concrete rules that a computer can represent and enforce.
**Rationale**: If the game cannot be simulated digitally, then AI Agent cannot be trained on the game and the system wouldn't be able to generate any meaningful data to aid the game design process.

## 5 Required Behaviour Overview

Our project consists of three main subsystems– Game Engine, AI Agent, and a data visualizer. These subsystems will work together to help game developers find imbalances in the game mechanics as well as find strategies that result in higher-than-expected winning percentages.

**AI Agent** – An AI model trained to play the gameat hand , acting as one of the players of the game to give insights into the possible paths to be taken. It would analyze all possible paths that a player can take at any point in the game. It would take as input the game state information (current state and available moves for the player) and output the next move that the player will take. AI and all learning algorithms will be abstracted away by the libraries used so exact techniques and formulas for learning will not be required by the developers to implement.
**Game Engine** – Abstract representation of the actual game as software. Contains all the game rules, observation space (state of the game), and action space (possible moves and commands that a player can choose from). It is also responsible for validating input from the AI Agent and outputting the game state with the moves available for the player.
**Data Visualizer** - Read the Game Engine and AI Agent log, and provide diverse graphic representations of data and information of moves to game state transitions. It will take the output from both the AI Agent and Game Engine to provide a way for the user to decipher imbalances in the game. The input and output of the system by the end-user (Game Designers) are only seen here as the data will be generated before hand and will be shown by the data visualizer, where the user can select different charts and graphs to visualize the data.

# 6 Functional Requirements

## 6.1 AI Agent's Functional Requirements

| **FR1** | AI Agent should must take the state as input. |
|---|---|
| **Rationale** | AI Agent will not store the state information. Rather it will get it from the Game Engine as an input. |
| **Verify** | AI will output the same result given the same input after resetting the system. |
| **Priority** | High |

| | |
|---|---|
| **FR2** | AI Agent can be swapped with another AI Agentor a human player. |
| **Rationale** | The project aims to provide a tool for all game designers to test their games. Every game will require a different AI model trained specifically on that game. Thus, AI Agent Agents needs to be modular enough to swap with other AI Agents. |
| **Verify** | Change the AI and the system will still complete game simulations. |
| **Priority** | Medium |

| | |
|---|---|
| **FR3** | AI Agent should must be observable. |
| **Rationale** | The decisions of the AI Agent should be recorded so that it can be reviewed by the game designer to guide the design of the board game. |
| **Verify** | After each simulation an output of the game log will be generated. |
| **Priority** | High |

| | |
|---|---|
| **FR4** | AI Agent should must only output an action that is in the action space provided by the Game Engine. |
| **Rationale** | By only selecting the actions in the action space provided by the Game Engine, errors will be reduced and the game can actually progress. |
| **Verify** | Given a state check all illegal moves, these must be omitted from the action space |
| **Priority** | High |

| | |
|---|---|
| **FR5** | AI Agent shall not see the state information that a human player cannot see. |
| **Rationale** | In multiplayer games, there will be information specific to the user (e.g. hidden cards) which the AI Agent should not see to have the same level of visibility as a human player would have. |
| **Verify** | Check the game state each AI Agent observes and ensure they are only see hidden states that apply to them. |
| **Priority** | High |

| | |
|---|---|
| **FR6** | AI Agent shall always respond with an action. |
| **Rationale** | Even if there is no optimal move, AI Agent should output an action so that the game can progress. |
| **Verify** | Game simulation must complete even if the AI will lose. |
| **Priority** | High |

## 6.2 Game Engine Functional Requirements

| | |
|---|---|
| **FR7** | The Game Engine must validate the input from the AI Agent |
| **Rationale** | One of the core functionality of our system is the interaction between the AI Agent and the Game Engine. The Game Engine must be responsible for validating the input (state) from the AI engine, to ensure the action taken is a valid action. |
| **Verify** | After receiving an action from the AI check if it is legal given the current game state. |
| **Priority** | High |

| | |
|---|---|
| **FR8** | The Game Engine must have an end/winning state to end the game. |
| **Rationale** | A Game Engine's main role is to transition from the start state to the end state using the player's moves. The Game Engine must validate if a player(s) has reached the end state at any point of the gameplay. |
| **Verify** | The game simulation must end. |
| **Priority** | High |

| | |
|---|---|
| **FR9** | The Game Engine must output the current state of the game, if not in the end state. |
| **Rationale** | As the AI Agent will be using the Game Engine's output to perform different moves, it is essential for the Game Engine to provide the current state of the game in order for the AI to make appropriate decisions. |
| **Verify** | Ensure after each action taken a new state is generated and provided to the AI |
| **Priority** | High |
| **FR10** | The Game Engine must follow all the playing and business rules of the game |
| **Rationale** | It is necessary for the Game Engine to implement all the game rules such that it can validate moves and transition to the end state. |
| **Verify** | Given a state and an action compare and ensure the new state that is generated is the same as described in the game rules. |
| **Priority** | High |
| **FR11** | The Game Engine must supply a list of game moves that an AI Agent can take. |
| **Rationale** | It is necessary for the AI Agent to know what moves it can take. This move list does not ensure all moves are legal, only that they can be executed. |
| **Verify** | Ensure AI is given a list of moves when it is there turn to take an action |
| **Priority** | High |

## 6.3 Game Data Visualization Analysis Requirement

| **FR12** | The game system must generate a variety of data visualization charts |
|---|---|
| **Rationale** | The functionality of data visualization charts is to provide different graphic representations of data and information such as pie charts, bar graphs, and line charts to the users. It can efficiently show insights that may be missed in traditional reports and translates the data into a visual context that the users can understand easily. |
| **Verify** | Generate more than Graph Number Benchmark 1 graphs and charts that represent different data points. |
| **Priority** | High |

| **FR13** | The game system must read data from the Game Engine and AI Agent logs. |
|---|---|
| **Rationale** | The functionality of reading data from the Game Engine and AI Agent allows the system to proceed with deeper data analysis. |
| **Verify** | Ensure the data from the game engine matches what is visualized on the data visualizer. |
| **Priority** | High |

| **FR14** | The game system must allow users to select different data points |
|---|---|
| **Rationale** | It is necessary for users to select a variety of data points such as win percentages of different first moves taken, game state changes in different decision-making processes, and actions having the highest win rates. |
| **Verify** | Ensure data visualized changes between different data points |
| **Priority** | High |

| | |
|---|---|
| **FR15** | The game system must compare data between different Game Engine and AI Agent logsAI Agents. |
| **Rationale** | The functionality of data comparison evaluates and compares selected data points. For example, it can compare multiple AI players' winning paths and output the most optimal option. |
| **Verify** | Ensure the visualizer can view data from different AIs at once. |
| **Priority** | Low |

# 7 Nonfunctional Requirements

This section will describe the non-functional requirements which will be the qualities the system has and why they are important and how they can be verified.

## 7.1 Accuracy

| | |
|---|---|
| **NFR1:** | The system, specifically the AI Agent should must get more and more accurate after each simulation in choosing winning moves. After (Learning Benchmark ) 1000 simulations reaching the Learning Benchmark 2, an AI Agent should must be able to win against its previous version. |
| **Rationale** | Since some of the AI Agent's Agents will be using reinforcement learning they must get more intelligent at playing the game after a duration of simulations. |

## 7.2 Usability

| | |
|---|---|
| **NFR2:** | The system should must be understood and navigable by the technical stakeholders (Dr. Mosser, Dr. Reinharz and Game Designers) after a (Understanding Benchmark ) 30-minute explanation an explanation length of the Understanding Benchmark 6, from the developers |
| **Rationale** | Usability of the system is not of high importance for to the general public. The main users will be AI Research Professors and Game Designers, these two users will only be interacting with one part of the system, that is the data visualizer. Users familiar with data visualization software should must have an easy time navigating and understanding the data if they have a Usability Benchmark 3 of data visualizing experience. |

## 7.3 Modularity

| | |
|---|---|
| **NFR3:** | The system should must be modular enough to where additional AI Agents can be implemented in two days integrated in Modular Benchmark 1 4 of development time. A Game Engine can be implemented in 7 days integrated in Modular Benchmark 2 5 of development time. |
| **Rationale** | Modularity is a very important quality the system needs to have. All the components need to be easily swappable, that is AI Agents should must be able to be changed with other implementations of AI, with minimal effort and the structure should must support different Game Engines being integrated. These benchmarks are only for integration time within our system and not the actual development time it takes to create these components. |

## 7.4 Portability

| | |
|---|---|
| **NFR4:** | The system should must run on macOS Monterey and newer, Windows10 and newer, and on Aliance Alliance Canada's Computer Cluster. The simulation of the Game Engine with the AI Agent's must be able to run on each OS for them to be verified as working. |
| **Rationale** | These operating systems will be used by the developers and the Aliance Alliance Canada's Computer Cluster will be used in case additional computing power is needed. |

## 7.5 Performance

| | |
|---|---|
| **NFR5:** | The system should must complete a single simulation in under (Speed Benchmark ) 5 minutesthe Speed Benchmark 7. |
| **Rationale** | If a simulation is taking too long it will slow down the learning of the AI Agents as well as take up resources from the developers' computers. |

## 7.6 Extensibility

| | |
|---|---|
| **NFR6:** | The system should must be able to be extended to include additional rules and mechanics in 2 days Extensible Benchmark 8 of development time from a developer within the team. |
| **Rationale** | With more complex games and games still in development rules and mechanics will be added iteratively rather than all at once. |

### 7.7 Look and Feel

| | |
|---|---|
| **NFR7:** | The system specifically the Data Visualizer should must clearly indicate data using different colours, lines and shapes. Should Must appeal to more than 75% (the Look Feel Benchmark ) 9 of game designers |
| **Rationale** | With a clear and clean look the data will be easier to visualize. |

### 7.8 Security

N/A

### 7.9 Cultural and Political

N/A

### 7.10 Legal

N/A

## 8 Phase-In Plan

All requirements with *High* priority must be implemented prior to the Revision 0 demonstration (February 6-17, 2023) as they represent the core functionality of our system.

- FR1 1
- FR3 3
- FR4 4
- FR5 5
- FR6 6
- FR7 7
- FR8 8
- FR9 9
- FR10 10
- FR11 11
- FR12 12

- FR13 13
- FR14 14

All requirements with *Medium* priority will be implemented before the Revision 1 demonstration (March 20-31, 2023).

- FR2 2

All requirements with *Low* priority will be added after all *High* and *Medium* requirements are implemented, time permitting.

- FR15 15

# 9 Likelihood of Changes for Functional Requirements

| Requirement | Likelihood of Change | Rationale | Ways to Change |
|---|---|---|---|
| **FR1** | Very Unlikely | Core implementation aspect | N/A |
| **FR2** | Very Unlikely | Core implementation aspect | N/A |
| **FR3** | Very Unlikely | Core implementation aspect | N/A |
| **FR4** | Very Unlikely | Must produce valid actions within the space or the system could loop indefinitely | N/A |
| **FR5** | Very Unlikely | AI Agent should not have an advantage over a human player | N/A |
| **FR6** | Very Unlikely | Must produce an action or the system cannot continue function | N/A |
| **FR7** | Very Unlikely | Core implementation aspect | N/A |
| **FR8** | Very Unlikely | Core implementation aspect | N/A |
| **FR9** | Very Unlikely | Important to visualize the state game for data analysis and allows for a human player to understand the state of game | N/A |
| **FR10** | Very Unlikely | Core implementation aspect | N/A |
| **FR11** | Very Unlikely | Allows for the AI Agent to generate a valid action to prevent invalid actions | N/A |
| **FR12** | Unlikely | May produce only a specific visualization data charts | Produce only specific data visualization charts |
| **FR13** | Very Unlikely | Required for the stakeholder to understand the data | N/A |
| **FR14** | Unlikely | Subject to time constraints, is not required in the minimal viable product but highly valued for the stakeholder | Requirement is removed and function is not implemented in final product |
| **FR15** | Likely | Subject to time constraints, is not required in the minimal viable product | Requirement is removed and function is not implemented in final product |

# 10 Traceability Matrices and Graphs

The table below shows the traceability matrix in how functional requirements depend on each other as well as depend on non-functional requirements and assumptions.

| ITEM | FR1 | FR2 | FR3 | FR4 | FR5 | FR6 | FR7 | FR8 | FR9 | FR10 | FR11 | FR12 | FR13 | FR14 | FR15 | NFR1 | NFR2 | NFR3 | NFR4 | NFR5 | NFR6 | NFR7 | A1 | A2 | A3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FR1 | | | | | | | | | X | | | | | | | X | | X | | X | X | | | | |
| FR2 | | | | | X | | X | | | | X | | X | | X | X | | X | | X | X | | | | |
| FR3 | | | | | | | | | | | | | X | | X | | | | | | | | | | |
| FR4 | | | | | | | | | | | X | | | | | X | | X | | X | X | | | | X |
| FR5 | | X | | | | | | | | | | | | | | | | | | | | | | | |
| FR6 | | | | | | | X | | | X | | | | | | X | | | | X | | | X | | |
| FR7 | | X | | | | X | | | | | | | | | | | | | | | | | | | X |
| FR8 | | | | | | | | | | | | | | | | X | | | | X | | | X | X | |
| FR9 | X | | | | | | | | | | | | | | | | | | | X | | | | | |
| FR10 | | | | | | X | | | | | | | | | | | | | | | | | | | X |
| FR11 | | X | | X | | | | | | | | | | | | X | | | | | | | | | |
| FR12 | | | | | | | | | | | | | | | | | X | | | | | X | | | |
| FR13 | | X | X | | | | | | | | | | | | | | | | | | | | | | |
| FR14 | | | | | | | | | | | | | | | | | X | | | | | X | | | |
| FR15 | | X | X | | | | | | | | | | | | | | X | | | | | X | | | |
| NFR1 | X | X | | X | | X | | X | | | X | | | | | | | | | | | | | | |
| NFR2 | | | X | | | | | | | | | X | | X | X | | | | | | | | | | |
| NFR3 | X | X | | X | | | | | | | | | | | | | | | | | | | | | |
| NFR4 | | | | | | | | | | | | | | | | | | | | | | | | | |
| NFR5 | X | X | | X | | X | | X | X | | | | | | | | | | | | | | | | |
| NFR6 | X | X | | X | | | | | | | | | | | | | | | | | | | | | |
| NFR7 | | | | | | | | | | | | X | | X | X | | | | | | | | | | |
| A1 | | | | | | X | | X | | | | | | | | | | | | | | | | | |
| A2 | | | | | | | | X | X | | | | | | | | | | | | | | | | |
| A3 | | | | X | | | X | | | X | | | | | | | | | | | | | | | |

# 11 Impact Analysis

This system will impact society as a stakeholder through the means of cultural factors as it is dealing with managing the quality of entertainment or specifically, the quality of board games. Completing this system will provide board game designers and game designers as a whole with a different and quicker means of balancing their turn-based games with AI. Being able to simulate many games in parallel to find unbalanced strategies in their games will allow them to produce these higher quality games for society to enjoy. In terms of health, safety, and other factors, this system will have a minimal impact.

Cultural bias in this system will not be an issue as the data that the AI will learn from comes explicitly from the game engine which is a replication of the game rules and not from an outside source. There is no outside impact from society on the AI as it is bias-free and trains from a blank slate, we are not using any pre-trained models, but doing all the training from within our own system.

# 12 Values of Constants

1. Graph Benchmark: 5
2. Learning Benchmark: 1000 simulations

3. Usability Benchmark: 1 year
4. Modular Benchmark 1: 2 days
5. Modular Benchmark 2: 7 days
6. Understanding Benchmark: 30 minutes
7. Speed Benchmark: 5 minutes on an average laptop with 8GB of RAM and at least i5 processor.
8. Extensible Benchmark: 2 days
9. Look and Feel Benchmark: 75%

# 13 Reflection Appendix

## 13.1 Game mechanics and rules

Assigned Team Members: *All*
All team members will be required to thoroughly understand the game rules for all the games we will be working on. This knowledge and expertise will be acquired through playing the game, reading game rule books, and any online resources that explain the rules of the games.

Every team member will be using all the discussed methods to gain the expertise in this domain as it is easy for everyone to understand the game rules by playing the game and then reading the game rule books to further enhance their knowledge.

## 13.2 AI Agent

Assigned Team Members: *Jeffrey and Jonah*
Learning how to develop and implement an AI Agent will require the knowledge and learning of specific AI APIs and libraries. The skills acquired from learning how these APIs and libraries work will allow the team to create feasible enough agents to generate viable moves for the Game Engine to accept. The team members assigned for to developing the AI Agents will begin with reinforcement learning libraries, Gym, PettingZoo, and SIMPLE, as a starting point to see if any viable AI Agents can be implemented with them and see if the team can further progress to other libraries. The team members were chosen for this topic for their previous experience in the machine learning course, COMPSCI 4ML3: Introduction to Machine Learning, and it should assist them in having to learn these libraries.
Approaches to learn learning these libraries will involve dealing with tutorials, working with the starter code, and reading through the documentation.

Jonah will use a combination of tutorials and documentation as he is able to understand

better through tutorials and then he will be able to continue enforcing his knowledge further with the documentation given for the APIs and libraries.

Jeffrey will begin by working on the starter code and continue will changing parameters in the starter code to see what output results as a consequence. Additionally, the use of the documentation will help change the starter code successfully. The reason for choosing this approach is because he particularly enjoys implementing first and then debugging towards the desired result afterward.

## 13.3 Game Engine

Assigned Team Members: *Michael and Hargun*
Knowledge of Game Engine development and object-oriented programming is expected for this subsystem. The group members responsible for this subsystem will need to gain an understanding of how a Game Engine works and the level of abstraction it requires for it to be compatible with the other subsystems. The team members are also required to have experience with Python and have the thorough understanding principles of object-oriented programming. It is expected team members have a basic understanding of all libraries and frameworks that are used during the development of this subsystem.
Knowledge and expertise will be gained through online tutorials, online articles, documentations of libraries/frameworks that are used, and through our supervisors.

Hargun will be using online tutorials and documentation to gain expertise in this domain area as it he finds it easier to learn from practical examples and then apply it to the desired problem.

Michael will be using online articles, documentation and our supervisors' expertise to gain knowledge in this domain area. Michael likes to research thoroughly before trying to solve a problem which is why it is beneficial for him to use these methods.

## 13.4 Data Visualization Analysis

Assigned Team Members: *Tianzheng* Data Visualization Analysis will require developers to understand users' expected data points, graphic representations, and what types of data need to compare. The group members are required to have sufficient knowledge of python data science libraries such as Matplotlib, Seaborn, NumPy, and pandas to create informative graphics and present accurate information to the users efficiently.

Tianzheng will be using online tutorials and documentation to gain expertise in data visualization and data analysis so as to find the most effective way to analyze the logs from the Game Engine and AI Agent.

# References

A board game of engine building and fantasy gods. URL https://www.anagecontrived.com/.

Apache echarts. URL https://echarts.apache.org/en/index.html.

Finite state machines. URL https://brilliant.org/wiki/finite-state-machines/.

Gymnasium documentation. URL https://gymnasium.farama.org/.

Plotly. URL https://plotly.com/python/.

react-vis. URL https://uber.github.io/react-vis/.

Simple ai. URL http://www.simple-ai.com/.

Welcome to tianshou! URL https://tianshou.readthedocs.io/en/master/.

Erikras. Tic-tac-toe fsm. URL https://github.com/erikras/tic-tac-toe-finite-state-machine.

Dr. Spencer Smith. Srs template. URL https://github.com/smiths/capTemplate/tree/main/docs/SRS.