**CSE 350**

**Team #9**

**Mason Mechlin, Sitan Subedi, Yamuna Rizal, Dylan Pike, Zane Peckenpaugh**

**Connect Four Game**

**Software Requirements Specifications Document**

**Version 1.0 Date: 06/19/2022**

**Table of Contents**

1. **Introduction**
   1. *Purpose*
   2. *Scope*
   3. *Definitions, Acronyms, and Abbreviations*
   4. *References*
   5. *Overview*
2. **The Overall Description**

*2.1 Product Perspective*

*2.1.1 Concept of Operations*

*2.1.2 User Interface Concepts*

*2.1.3 Hardware Interfaces*

*2.1.4 Software Interfaces*

*2.1.5 Communications Interfaces*

*2.1.6 Memory Constraints*

*2.1.7 Operations*

*2.1.8 Site Adaptation Requirements*

*2.2 Product Functions*

*2.2.1 Load Game*

*2.2.2 Player Moves*

*2.2.3 Rewind Mode*

*2.3 User Characteristics*

*2.4 Constraints*

*2.5 Assumptions and Dependencies*

*2.6 Apportioning of Requirements*

1. **Specific Requirements**

*3.1 External Interface Requirements*

*3.1.1 User Interface*

*3.2 Classes/Objects*

*3.2.1 Player Class*

*3.2.2 Computer Player Class*

*3.2.3 Human Player Class*

*3.2.4 Remote Player Class*

*3.2.5 Board Class*

*3.2.6 Player Create Menu*

*3.2.7 Board Interface*

*3.3 Performance Requirements*

*3.4 Design Constraints*

*3.5 Standardization Requirements*

*3.6 Software System Attributes*

1. **Supporting Information**
2. **Task 2**
3. **Task 3**
4. **Introduction**
   1. **Purpose**

This document is meant to be a reference for the requirements of a Connect Four Computer game. The first two sections will give a high-level overview of what customers expect from the game and the third section will be a detailed checklist that the developers must complete to meet the expectations of the customer. Nevertheless, the third section will be understandable to customers.

* 1. **Scope**

This document will cover the totality of the development requirements for the Connect Four game. Ideally, all features mentioned in this document will be implemented, but time constraints may not allow some to come to fruition. These features will be prioritized in a later section. At least all the features deemed “essential” will be implemented.

* 1. **Definitions, Acronyms, and Abbreviations**

System: A software application running the Connect Four game.

Game-Instance: The System, discluding the functions that generate commands such as computer player’s AI, the events and event handling functions that translate the user’s input into commands, and the infrastructure that handles sending a remote player’s commands.

Computer: The software and hardware that excludes the system.

User: The person or people controlling the system directly.

Remote User: The person sending commands to the system through the internet.

Player: A construct of the system that controls input and represents the participants to the game.

Input-type: The mode through which commands are passed to the system’s player construct. Three types are specified for the purpose of this application.

* Human: A player of this type receives commands through inputs from the user through the computer’s hardware interfaces (such as keyboard and mouse).
* Computer: A player of this type receives commands from an AI in the system.
* Remote: A player of this type receives commands from a remote user through an internet connection.
  1. **References**

H. Wexler, N. Strongin “Connect Four Instructions Manual” Milton-Bradley, 1974. [Online]. Available: <https://www.manualslib.com/manual/580224/Milton-Bradley-Connect-Four.html> [Accessed: 13 June, 2022]

* 1. **Overview**

The connect four game is intended to simulate a modified game of connect four between two players.

1. **Overall Description**

This application will simulate a game of connect-four between two players. The simulation will be capable of determining the conditions of a finished game and will also be able to detect illegal moves and prevent players from doing them. The players will also be able to rewind games to review previous moves.

There may also be time for the development of extra features such as the above-mentioned computer AI player and online play. These features are “desired” as per the customers’ wish-list but may end up being cut if there is no time to finish them before the deadline.

* 1. **Product Perspective**

This connect four application is simply meant to meet the demands of the customer as a tool for demonstrating the game of connect four and teaching its rules and strategies. The rewinding system stands out as a particular function for fulfilling this role, allowing players to review their strategies and simulate alternative paths a given game could have taken. If time allows, a computer vs computer mode will also be implemented. This should give onlookers an easily accessible demonstration of the game’s rules.

* + 1. **Concept of Operations**

The system can be in three states:

* Setup: Setting up the game; creating the player characters that will interface with the software.
* In-game: Players take turns placing checkers on the game board until the conditions for victory are met.
* Rewind: After a game is finished, the user can rewind the moves played throughout the game.

Use cases will be made further describing these functions.

* + 1. **User Interface Concepts**

Qr code

Description automatically generated

*Figure 2.1 “Player Creation” Figure 2.2 “Game board”*

* + - 1. **Player Creation**

The start menu will have a simple press any button to start. Pressing any button will bring the user into a menu for setting up the next game. The player creation menu will have two button lists that have the input types that can be selected for each respective player. A text input box below both lists will be used to input the names of each player.

**2.1.2.2** **Main Game Screen**

The main game interface will show the connect-four gameboard. To the right of the board is the name of player one and the input-type of player one. To the left of the board is the name of player two and the input-type of player two. The bottom center of the screen will display the turn number of the current turn. The player who is currently moving will have their name highlighted by a brightly colored border.

* + 1. **Hardware Interfaces**

None.

* + 1. **Software Interfaces**

None.

* + 1. **Communications Interfaces**

Should time allow, the game will be playable over the internet. A player of the remote input type shall retrieve commands from another system running the same application.

* + 1. **Memory Constraints**

The game shall only require a maximum storage of 42 moves as a game can only have 42 moves, each one taking one of the 42 spaces available on the board. There should also be enough space to keep track of the state of all 42 of the spaces.

* + 1. **Operations**

The user is intended to be present for all operations performed by the system. When two players with the “human” input type are playing, both players shall receive their commands from the user’s mouse and keyboard; two human players implies that two people are using the same mouse and keyboard to input commands. On the other hand, when one of the players is remote and the other is human, only one player will retrieve commands from the user. The other player will retrieve its commands from the remote user. From the perspective of the remote user’s system, the player the remote user controls is a human input-type player that retrieves commands directly from the remote user’s mouse and keyboard, while the user is a remote input type player.

* + 1. **Site Adaptation Requirements**

None.

* 1. **Product Functions**

The following are the ideal set of use cases for the full version of the game, should all desired features be implemented. The user controlling the menus options will be referred to as “User”. “Player” is a generic name that can be used to refer to a computer, human, or remote human player. “System” refers to the computer when it is acting automatically.

* + 1. Load Game

1. User selects “new game” in the start-up menu
2. User selects one of four game types:
3. Human vs. Human
4. Human vs. Computer
5. Computer vs. Computer
6. Human vs. Remote Human
7. Both players choose or are given in-game personas that will be displayed in the ensuing game and will be used to indicate which player is expected to act.
   * 1. Player moves
8. System notifies player by their in-game persona to select their move.
9. Player chooses a column to drop their piece.
10. System verifies the move’s legality and displays the outcome of the player’s choice.
11. System determines if conditions are sufficient to declare victory. If so, game ends and declares the currently active player as the victor, else repeat from step one with the other player.
    * 1. Rewind mode
12. System asks if the user would like to rewind once the game has finished. If the user chooses to exit, they are sent to start menu. Else they move to step 2.
13. User chooses either to rewind by one turn, move forward by one turn, rewind to beginning, move to the last turn or exit rewind mode. Should exit be chosen User is sent to the start menu. Else repeat step 2.

**2.3 User Characteristics**

This application is intended for use by people of all ages.

**2.4 Constraints**

The game will be developed in python and will thus be available on Windows and Linux systems.

**2.5 Assumptions and Dependencies**

None.

**2.6 Apportioning of Requirements**

The requirements for this application are partitioned into three groups: essential, desired, and optional. Essential requirements are deemed necessary and will be exclusively targeted for the development of the prototype. “Desired” requirements are those mentioned in our customer’s wish-list. Should all the essential requirements pass the implementation phase, the “desired” requirements will be targeted next. “Optional” requirements will be implemented at the discretion of the developers once essential requirements have passed testing and desired requirements have passed implementation. These requirements will be small quality-of-life upgrades that will not interfere or tamper, in any way, with the functioning of the essential or desired requirements.

**3. Specific Requirements**

**3.1 External Interface Requirements**

**3.1.1 User Interface**

**3.1.1.1 Player Creation**

Both players choose one input option (human, computer, remote) and write their name in a text input box. Should a player be remote, the input box is greyed, and the name of the remote player is sent over the internet. If there is no other user connected via the internet, the remote option for that player is greyed.

**3.1.1.2 Game Board**

Besides the connect-four board in the center of the screen, there are some details that are not immediately obvious. Both players have a color assigned to them; player 1 is yellow, player 2 is red. When it is player x’s turn, the box containing the name of player x is highlighted with a border of the player x’s assigned color. While player x is choosing a location to drop their checker, the player’s checker can be seen floating above the column that player’s mouse is hovering over or closest to.

When the game is finished, the names of both players become faded and left and right direction arrows appear at the bottom center of the screen. These buttons may be clicked to either advance(right) or rewind(left) to by one move. There are also double arrows adjacent to both single arrow buttons that either send the user to the beginning of the move sequence (far left) or to the end (far right).

* 1. **Classes/Objects**

**Diagram

Description automatically generated**

* + 1. **Player Class**

This is the class for an entity that directly interacts with the board.

* + - 1. **Player Attributes**
         1. **Name**

Players shall have a name used to refer to them. This name will be highlighted when the player’s turn has arrived.

* + - * 1. **Color**

The checkers the players place will be colored according to the color chosen for that player.

* + - 1. **Player Entities**
         1. **Player 1**

When the game first starts, this player goes first.

* + - * 1. **Player 2**

When the game starts, this player moves after player 1.

* + - 1. **Player Functionality**
         1. **Option Pick**

A player object shall be able to send the instructions of a move to the board.

* + 1. **Computer Player Class**
       1. **Computer Player Attributes**

The computer class shall have all the attributes of the player class (refer to 3.2.1.1).

* + - * 1. **Difficulty Level**

The computer class shall also have a difficulty level attribute to determine the sophistication of the AI that will decide the moves it will take.

* + - 1. **Computer Player Entities**

A computer player can correspond to either player 1 or player 2 or both players.

* + - 1. **Computer Player Functionality**
         1. **Option Pick**

A computer player generates commands to the board through an AI.

* + 1. **Human Player Class**
       1. **Human Player Attributes**

The computer class shall have all the attributes of the player class (refer to 3.2.1.1).

* + - 1. **Human Player Entities**

A human player can correspond to either player 1 or player 2 or both players.

* + - 1. **Human Player Functionality**
         1. **Option Pick**

A human player generates commands to the board by taking input from the mouse and keyboard of the computer that runs the system.

* + 1. **Remote Player Class**

Instantiations of this class from the user’s perspective are instantiations of a human player from the perspective of another user connected to the first user through the internet.

* + - 1. **Remote Player Attributes**

Refer to attributes for Player (3.2.1.1).

* + - 1. **Remote Player Entities**

A remote player can be player 1 or player 2 but not both.

* + - 1. **Remote Player Functionality**
         1. **Option Pick**

A remote player generates commands by receiving them from a system operated by another user.

* + 1. **Board Class**
       1. **Board Attributes**
          1. **Checker Arrangement**

The board has an arrangement of checkers that belong to either player 1 or player 2 and empty spaces.

* + - * 1. **Move Log**

The board shall have a log of all previous moves played in the game.

* + - * 1. **Players**

The board shall keep track of the players that are playing on it.

* + - 1. **Board Entities**

There shall be only one instance of the board class for the game in progress

* + - 1. **Board Functionality**

**3.2.5.3.1 Input Command**

The board shall be able to receive commands from the player and respond by updating to a state that is legal in terms of the rules of connect four and intended by the player. The board then logs that move in its move log. Finally, the board changes the current player to the player who moves next.

**3.2.5.3.2 Rewind**

The board should be able to change its arrangement to replicate a state it had taken previously in the current game.

**3.2.5.3.3 Game Check**

After every move played, the board shall check its checker arrangement to check for the condition of a connect-four and end the game should it meet those conditions.

* + 1. **Player Create Menu**
       1. **Player Create Menu Attributes**
          1. **Input Name Field**

The player create menu shall, in all versions of the game, have a player name entry field for both players used to assign names to the player entities. Should the user choose the remote input type for either player entity the remote input type selection button for the other player entity will be greyed out and non-selectable. The input field for the player entity chosen to be remote shall also be greyed out and be made to only respond to a remote player.

* + - * 1. **Human Input type selection buttons (Desired)**

Should other input types than “human” be accessible, there will be a human type-selection button corresponding to both players. Otherwise, the player’s input type will be assumed to be human by default. Should the user select a different input type button (for the same player) after selecting this button, this button shall become unselected.

* + - * 1. **Computer Input type selection buttons (Desired)**

Should the computer player input type be implemented, there shall be a computer player button corresponding to both players. Should the user select a different input type button (for the same player) after selecting this button, this button shall become unselected.

* + - * 1. **Remote Input type selection buttons (Desired)**

Should the remote player input type be implemented, there shall be a computer player button corresponding to both players.

* + - 1. **Player Create Menu Functionality**
         1. **Button functionality**

For each player, the user can only select one input type. If the user selects another input type button for the same player, the previously selected button shall become unselected.

* + - * 1. **Input box functionality**

The input box for a player shall be inaccessible to the user if they choose the remote input type for that player.

* + 1. **Board interface**

The board interface is simply the collection of visual elements the player will see to allow them to keep track of the progress of the game they are playing.

* + - 1. **Board interface Attributes**
         1. **Board**

The board interface shall represent a board datatype whose contents inform the appearance of the board to the player.

* + - * 1. **Board Space**

For every data element in the board’s checker arrangement, there shall be a sprite corresponding to it. This sprite shall always be a square with a circle enclosed; colored either black(empty) or some color corresponding to the player who placed checker at the corresponding position.

* + - * 1. **Columns**

When a player places a checker, they will have to choose a column in which the checker will be placed. The board interface shall have spaces allocated that respond when the mouse is hovering over them.

* + - * 1. **Player Nametags**

The players shall have their names displayed on either side of the board space. When their turn has arrived the current players’ name tag shall be highlighted with the color that corresponds to the player’s color property.

* + - * 1. **Rewind Buttons**

The board interface shall also have rewind buttons that activate when the game is over. These buttons shall display the previously played moves that the user intends.

**3.2.7.2** **Board Interface Events**

**3.2.7.2.1 Column Hover**

When the current player’s mouse is hovering over one of the board interfaces column spaces, a checker with a color corresponding to the color assigned to that player shall be visible above the column; indicating the player is considering dropping said checker on top of the column’s stack.

**3.2.7.2.2 Click Column**

When the current player clicks on one of the board interfaces column spaces, the board interface shall respond by adding a checker with a color corresponding to that player’s color property.

* 1. **Performance Requirements**

In a remote game, the time it should take from submitting a command and the other players game-instance retrieving that command followed by the opponent’s system sending an acknowledgement should be no longer than 10 seconds.

The computer player should take no longer than 2 seconds to generate its next move in a form that can be read by the game-instance.

For a human player, the time it should take for the game-instance to receive a mouse-click command from the OS and translate it to system readable data should be no longer than one second.

For all types of players, the time it should take for the game-instance to process and display a system readable command it receives should be no longer than one second.

The success rate for the opponent retrieving an accurate command on first attempt shall be at least 90%

* 1. **Design Constraints**

**3.4.1 Hard Drive Usage**

The files of the fully installed game shall take no more than 10Mb of hard drive.

**3.4.2 Memory Usage**

The structures needed for a game instance (players, board, board interface) shall take no more than 10Mb of memory.

**3.4.3 Reliability**

If it takes longer than 10 seconds for opponent’s game-instance to retrieve a command and send the acknowledgement back, the opponent shall send another command.

If the remote opponent’s game-instance retrieves a command that it cannot read, it shall request a resend.

Should the opponent’s system retrieve a command that can be read by the game instance, it shall confirm the accuracy of this command by sending that command back as an acknowledgement.

**3.4.4 Expandability**

The considerations for expandability past the full version of the game (all desired features have been implemented) shall only pertain to customizability features (player profiles pictures, more options for player colors or more characters represented in names). The design process of the full version shall consider the potential for such features to be implemented later.

**3.4.5 Portability**

The application shall be installable on at least all Windows 10 and 11 computers.

* 1. **Standardization Requirements**

The system will not need to comply to any pre-established standards.

* 1. **Software System Attributes**

**3.6.1 Reliability**

The game-instance shall never fail to process a command that is formatted correctly, and it shall be able to tell if a command is formatted correctly.

**3.6.2 Availability**

The system shall be installable by running a single installer executable. It shall be a major goal to make the remote connection functionality as simple as possible; requiring the player to input a single code to connect with an opponent.

**3.6.3 Security**

The game is not intended to handle any sensitive information or keep track of data on its users. There will be no considerations in terms of security.

**3.6.4 Maintainability**

The design of the program shall be particularly modular in sections of the code where computer characters’ AI and remote connections are concerned. These will be the most technically complicated and, in the case of the remote connection functionality, subject to future changes.

**3.6.5 Portability**

The system shall be written in Python so it should be available on any machine that has Python installed.

1. **Supporting information**

Not Applicable

1. **Task 2 – Traceability Matrix**

**Table

Description automatically generated**

1. **Task 3 – State Diagram/Class Model**

**Diagram, engineering drawing

Description automatically generated**

**Diagram

Description automatically generated**