Connect-Games

Shawn Salm

Wentworth Institute of Technology

Capstone

Professor Durga Suresh-Menon

April 16th, 2018

Contents

1.0 Glossary

Term	Definition
Tic-Tac-Toe	“Tic-tac-toe (also known as noughts and crosses or Xs and Os) is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game” [1]
Connect Four	“Connect Four (also known as Captain's Mistress, Four Up, Plot Four, Find Four, Four in a Row, Four in a Line and Gravitrips (in Soviet Union)) is a two-player connection game in which the players first choose a color and then take turns dropping colored discs from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. Connect Four is a solved game. The first player can always win by playing the right moves.” [2]
User	A person that registers and logs into Connect-Games.
Computer Student	Each user will have one computer student to train. A computer student is a combination of an algorithm (described in the requirements section below) along with a history of the sequence of moves played by the computer.
Game	A game is either Tic-Tac-Toe or Connect Four and is played between a user and a computer student.
Match	A match is a game played between two computer students.
Login	In order to play Connect-Games, a user must provide a valid user name and password.
User Name	The name in which the user is known within Connect-Games. Each user name must be unique.
Password	A text string that authenticates that the user is who they say they are.
Computer Student State	A computer student may be in one of two states. Either the computer student is available to play against other computer students or the computer student is not available to play.
Sequence of Moves	All games within Connect-Games are played in what can be represented as an array or sequence of numbers that represent the place that the user moved to on a board.
Computer Student Behavior	The algorithm that the computer student follows in order to pick the next move. See the requirements section below for details.
System Availability	A time period in which a user can log into, register, and play Connect-Games.
System Design	The patterns in code that allow the Connect-Games program to support other games besides Tic-Tac-Toe and Connect Four in the future.
System Performance	How fast the Connect-Games responses to the user’s actions.

System Behaviors	Determine if a move is valid or not.
System Rules	Determine if a move results in the player winner a game.
Priority Weight	A number between 1 and 10 with 10 being the highest priority or a must have requirement.
Requirement ID	Unique identifier for a requirement.
MVC	Model–view–controller - architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. [7]
SRP	Single Responsibility Principle (SRP) – a class should only have one, and only one reason to change. [3]
DRY	Don't Repeat Yourself - The DRY principle states that these small pieces of knowledge may only occur exactly once in your entire system. [6]
Abstract Factory	Abstract Factory offers the interface for creating a family of related objects, without explicitly specifying their classes. [5]
Strategy Pattern	Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it. [4]
Program to Interface	The principle that states the designed interactions of classes should be done by way of interfaces or abstractions rather than tightly coupling classes.
LSP	Liskov Substitution Principle (LSP) – build sub-classes that are interchangeable; they must adhere to a contract that allows them to be safely substituted for one another. [3]
ISP	Interface Segregation Principle (ISP) – avoid creating large interfaces that have many unrelated methods so that an implementing class avoids depending on things that it doesn't use. [3]
OCP	Open-Closed Principle (OCP) – for a class to be easy to maintain, it should be designed to allow the behavior to change by adding new code, rather than changing existing code. [3]
DIP	Dependency Inversion Principle (DIP) – the code that implements high-level policy should not depend on the code that implements low-level details. [3]
ADP	Acyclic Dependencies Principle – allow no cycles in the component dependency graph. [3]
SDP	Stable Dependencies Principle – depend in the direction of stability. [3]
SAP	Stable Abstractions Principle – a component should be a abstract as it is stable. [3]
View/Screen/Page	A web page that the user interacts with.
Ajax	Asynchronous JavaScript + XML" is a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications. [8]
Layered Architecture	Partitions the concerns of the application into stacked groups (layers). [9]

Component-Based Architecture	Decomposes application design into reusable functional or logical components that expose well-defined communication interfaces. [9]
Domain model	In software engineering, a domain model is a conceptual model of the domain that incorporates both behavior and data. [10]

2.0 Proposal

21. Problem Statement

Currently, online and mobile games exist that allow one to play games such as Tic-Tac-Toe and Connect Four against other people from around the world or against a computer at different levels of difficulty. Playing against live opponents however, can be problematic in that each person has different schedules as well as different amounts of time in which they can devote to the game. Waiting around for an opponent to make a move may be inconvenient at best and at other times frustrating. Playing a computer solves the asynchronous problem that one has when playing against a live opponent, however, it also may feel mechanical to engage with and unsatisfying when winning. Therefore, there is a need for a game that allows one to have the on-demand convenience of playing a computer, while at the same time have the satisfaction one get when defeating a live opponent or in this case, a computer student.

2.2 How the Connect Games Addresses the Problem

Connect-Games will allow one to teach a computer student how to win at games such as Tic-Tac-Toe or Connect Four simply by playing as an opponent. The user will then be able to see the computer student's progress as it plays against other people's taught computer students. In this way, one will get the convenience of playing against a computer while at the same time, the satisfaction of a computer student winning against another person's computer student. Once a user feels that a student is ready to engage other students, the user will be able to make the computer student available to play any other users' student who has been made available.

2.3 Overview of How the Connect Games Works

Connect-Games will be made up of a C# program run on a server to perform the logic of the system and a SQL database to hold the results of the games played as well as the users' information such as their username and password. The program will be able to manage games that play on a two-dimensional board in which the winner connects n-number of their pieces in a row such as Tic-Tac-Toe or Connect Four. This will require that the games are broken down into simple sequences with the allowed behavior of a player being able to be interchangeable. The rules for winning a game must also be interchangeable. Each games sequence of moves will be stored allowing the computer student to learn by using winning sequences as a model in determining their next move. The game can be hosted on a cloud platform such as Azure with the user interface being a simple combination of ASP.NET MVC microservices with an HTML/JavaScript front end.

3.0 System Requirements

Connect-Games must allow one to teach a simulated computer student how to win at games such as Tic-Tac-Toe and Connect Four simply as playing as an opponent. The user will then be able to measure the computer student's progress as it plays against other user's taught computer students. In this way, one will get the convenience of playing against a computer while at the same time, the satisfaction of a computer student winning against another person's computer student. Once a user feels that a student is ready to engage other students, the user must be able to make the computer student available to play any other users' student who has been made available.

3.1 Enumerated Functional Requirements

User Registration and Login		
Requirement ID	**Priority Weight**	**Description**
REQ-1	10	User must be able to register in order to play the games.
REQ-2	10	User must provide a unique user name.
REQ-3	10	The user's name must be at least one character long for the initial version.
REQ-4	10	User must also provide a password when registering to play the games.
REQ-5	10	Passwords, in the initial version, will have no strict rules as for the password's length or format other than needing to be at least one-character long.
REQ-6	10	User must login using a user name and password already created through registration.
Computer Student's Behavior		
Requirement ID	**Priority Weight**	**Description**
REQ-7	7	The computer student must keep a record of all completed games in which a player won, whether that player was the computer student or not.
REQ-8	10	The computer student must know the basic rules for the games, Tic-Tac-Toe and Connect Four. That is, the computer student must know a valid move and make only valid moves.
REQ-9	10	When making a move, the computer student must first detect if the computer student can win the game with the next move. If so, the computer student must make the winning move.
REQ-10	10	When making a move, the computer student must second detect if the computer student can lose the game with the next move. If so, the computer student must make a move to block the other player from winning.

REQ-11	7	When making a move, the computer student must third test whether or not the current sequence of moves matches a winning game that the computer student has played before. If so the student must follow that sequence using it to decide the next move.
REQ-12	10	When making a move, the computer student must forth and lastly, make a random valid move if REQ-9, REQ10, and REQ-11 are not able to be met.
Computer Student vs Computer Student Matches		
Requirement ID	**Priority Weight**	**Description**
REQ-13	5	A computer student must be able to be available or not available to play against other computer students which setting must be able to be controlled by the user.
REQ-14	5	If the computer student is in a state of available to play other computer students, then another user can challenge that computer student to a match with the user's student.
REQ-15	5	In order to be able to challenge another user's computer student to a match, the user's computer student must be in the state of available to play.
REQ-16	5	When a match is played between computer students, a random pick will be made of who goes first.
User Training Student		
Requirement ID	**Priority Weight**	**Description**
REQ-17	10	A user must have one computer student.
REQ-18	10	The user must be able to play a game against the computer student at any time after logging in.
REQ-19	7	The computer student will store the sequence of moves made during the game if there is a win.
REQ-20	8	The user must be able to decide whether or not to go first when playing against the computer student.
Tic-Tac-Toe Game Requirements		
Requirement ID	**Priority Weight**	**Description**
REQ-21	6	The player that goes first will always be X.
REQ-22	6	The player that moves second will always be O.
REQ-23	6	The default rules as described in the glossary must apply to all games and matches in Tic-Tac-Toe.
REQ-24	6	As soon as there is a winner or there are no more valid moves available to be made, the game must end.
Connect Four Game Requirements		
Requirement ID	**Priority Weight**	**Description**
REQ-25	6	The player that moves first will always be the color black.
REQ-26	6	The player that move second will always be the color red.

REQ-27	6	The default rules as described in the glossary must apply to all games and matches in Connect Four.
REQ-28	6	As soon as there is a winner or there are no more valid moves available to be made, the game must end.

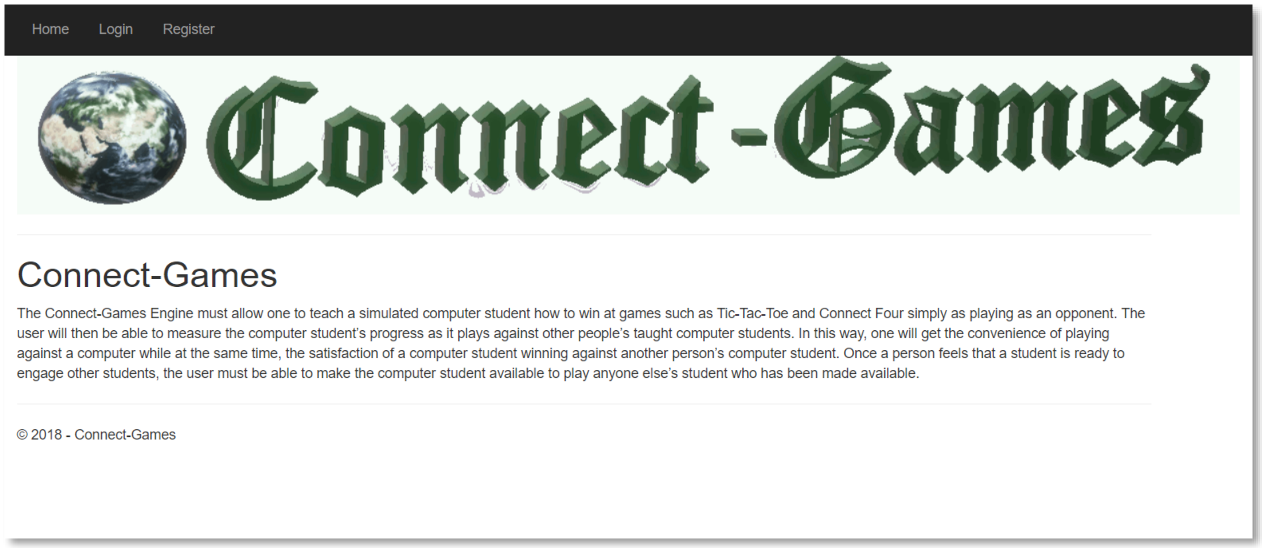
3.2 Enumerated Nonfunctional Requirements

System Design		
Requirement ID	**Priority**	**Description**
FURPS-1	5	The program design must allow different behaviors to be created in order to allow different games to be played. See the glossary for the definition of a behavior in Connect-Games.
FURPS-2	5	The program design must allow different rules to be created in order to allow different games to be played. See the glossary for the definition of a rule in Connect-Games.
System Performance		
Requirement ID	**Priority Weight**	**Description**
FURPS-3	7	The system must response to the user's action within 10 seconds. Actions include registering, logging in, the act of setting up a match between two computer students (not the match itself), and view the home page. Note that the computer student's move is not part of this requirement.
FURPS-4	7	The computer student's move must not take more than a minute to occur. Whether the computer student is playing against another computer student or against a user.
System Availability		
Requirement ID	**Priority Weight**	**Description**
FURPS-5	6	The system should be available to the users 90% of the time.
User's System		
Requirement ID	**Priority Weight**	**Description**
FURPS-6	10	The only browser that will be supported within the first release will be Google Chrome Version 64.0.3282.186 (Official Build) (64-bit) running on a desk top computer. Although it is likely that other, modern browsers will run Connect-Games successfully.
FURPS-7	10	The user's browser must support cookies.
FURPS-8	8	The user's computer must have at least 8 GB of RAM.
FURPS-9	10	The user's computer must have access to the internet.

4.0 On-Screen Appearance Requirements

4.1 Connect-Games Home Page

The Connect-Games home page is the first page the user sees when navigating to the site.



Connect-Games

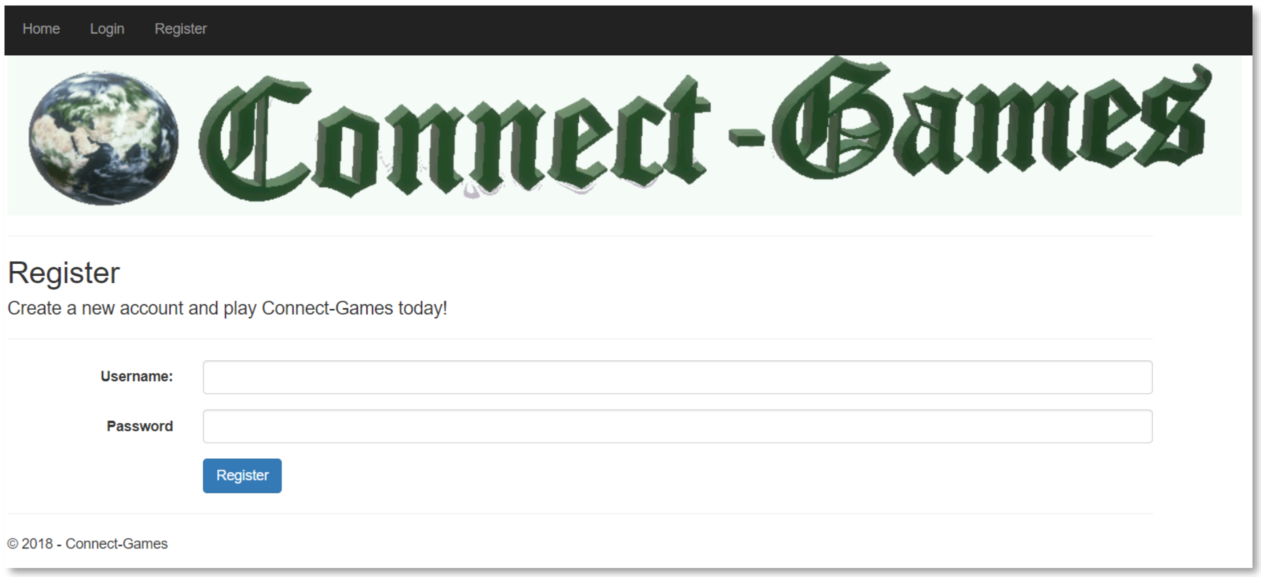
The Connect-Games Engine must allow one to teach a simulated computer student how to win at games such as Tic-Tac-Toe and Connect Four simply as playing as an opponent. The user will then be able to measure the computer student's progress as it plays against other people's taught computer students. In this way, one will get the convenience of playing against a computer while at the same time, the satisfaction of a computer student winning against another person's computer student. Once a person feels that a student is ready to engage other students, the user must be able to make the computer student available to play anyone else's student who has been made available.

© 2018 - Connect-Games

Page 1 – Connect-Games home page for non-logged in users.

4.2 User Registration Page

The Connect-Games registration page must allow the user to register to play.



Register

Create a new account and play Connect-Games today!

Username:

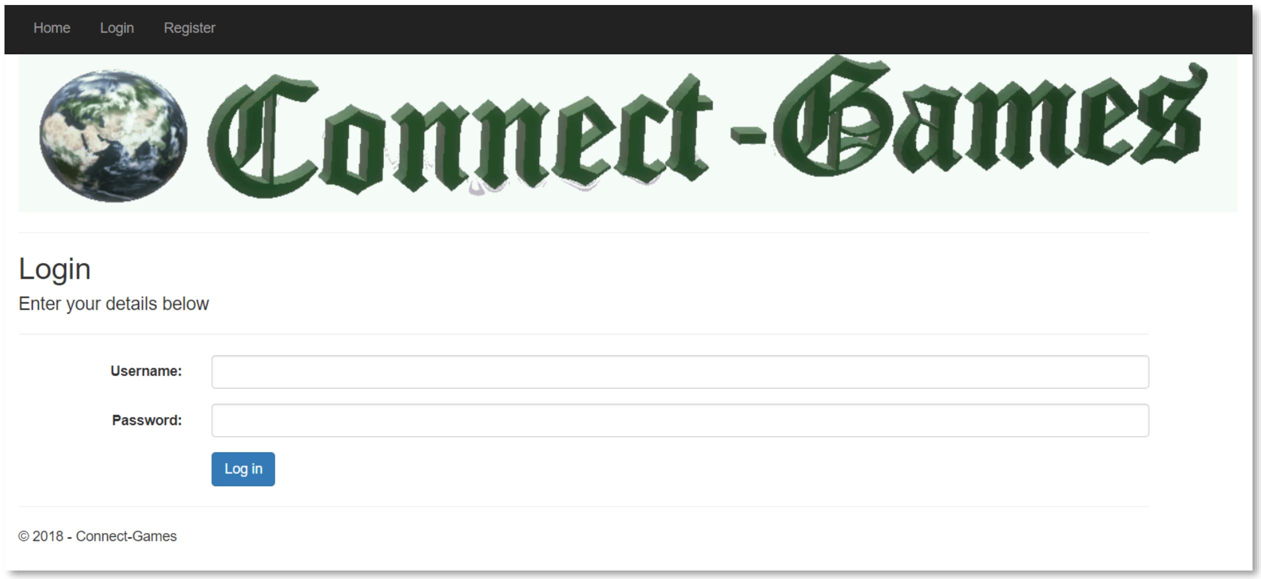
Password

© 2018 - Connect-Games

Page 2 – User registration page.

4.3 User Login Page

The Connect-Games login page must allow an already registered user to log in.



Login

Enter your details below

Username:

Password:

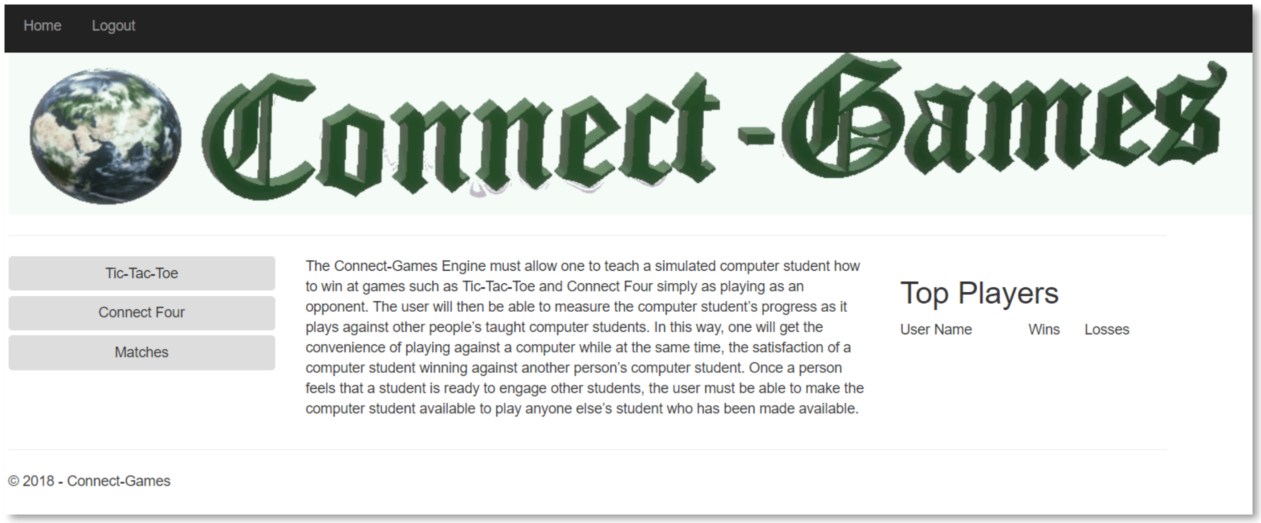
Log in

© 2018 - Connect-Games

Page 3- Connect-Games login page.

4.4 Connect-Games After Logging In

The home screen after logging in must allow the user to play Tic-Tac-Toe or Connect Four against the computer student as well as to setup matches with other users' computer students. Finally, the top three players will be featured on the right side of the page.



The Connect-Games Engine must allow one to teach a simulated computer student how to win at games such as Tic-Tac-Toe and Connect Four simply as playing as an opponent. The user will then be able to measure the computer student's progress as it plays against other people's taught computer students. In this way, one will get the convenience of playing against a computer while at the same time, the satisfaction of a computer student winning against another person's computer student. Once a person feels that a student is ready to engage other students, the user must be able to make the computer student available to play anyone else's student who has been made available.

Top Players

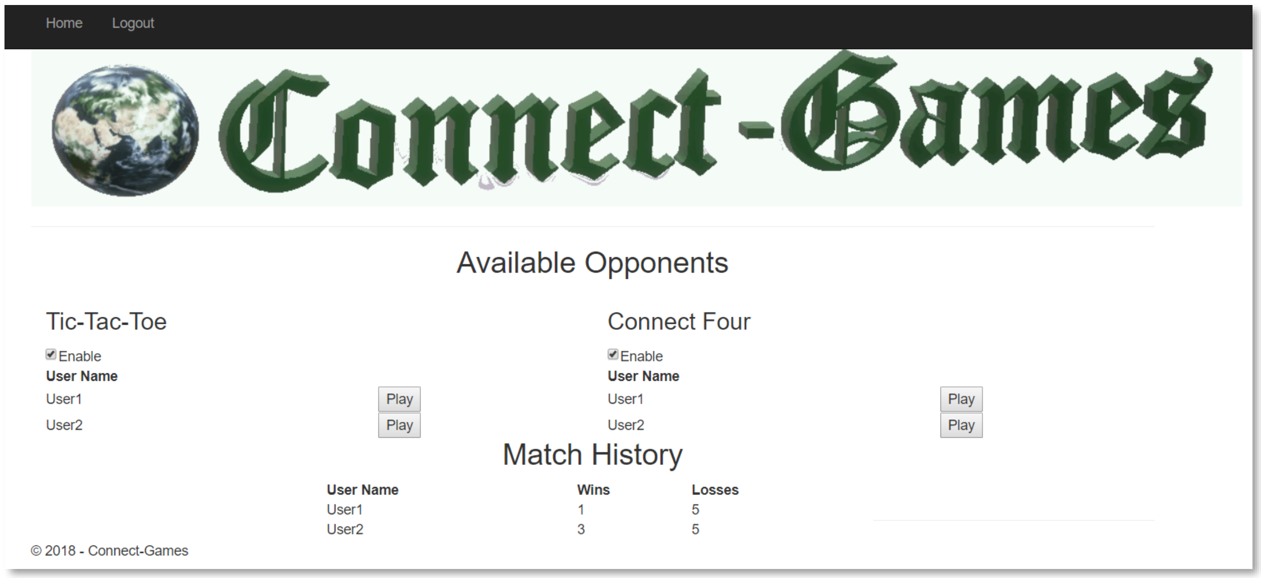
User Name	Wins	Losses

© 2018 - Connect-Games

Page 4 – Home screen after logging in.

4.5 Setting Up Match Between Computer Students

Users must be able to challenge other available opponents to Tic-Tac-Toe or Connect Four. They can also view the computer student's match history.



Available Opponents

Tic-Tac-Toe

☑ Enable

User Name	
User1	Play
User2	Play

Connect Four

☑ Enable

User Name	
User1	Play
User2	Play

Match History

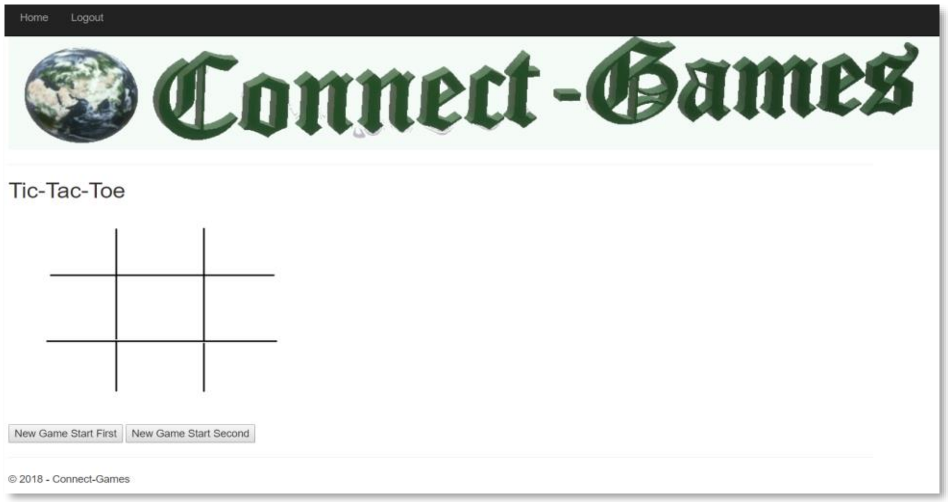
User Name	Wins	Losses
User1	1	5
User2	3	5

© 2018 - Connect-Games

Page 5 – Match screen.

4.6 Tic-Tac-Toe

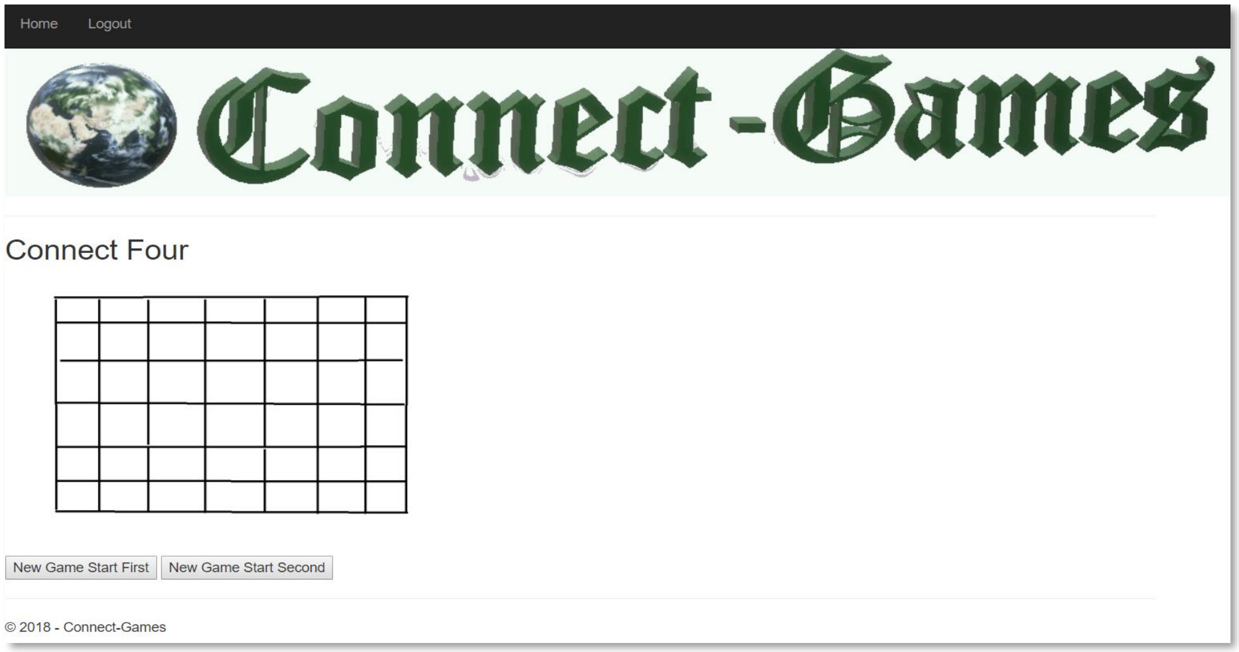
User must be able to play Tic-Tac-Toe against computer student.



Page 6 – Tic-Tac-Toe against computer student.

4.7 Connect Four

User must be able to play Connect Four against computer student.



Page 7 – Connect Four against computer student.

5.0 Functional Requirements Specification

5.1 Stakeholders

There are a couple of stakeholders for Connect-Games. There is the author of Connect-Games, i.e. the student whose grade dependents on it. Second, there is the potential players of the game who will hopefully enjoy playing Connect-Games.

5.2 Actors and Goals

The major actors of Connect-Games will be the users who will take part in playing the games. The players will play against and participate in the training of the computer student as well as initiate matches pinning a computer student against another computer student. The goal of the player is to teach the computer student well enough that the computer student then has the knowledge to win against other computer students.

5.3 Use Cases

i. Casual Description

1. **User Registration** – In order to play Connect-Games, the user must first register with the system using the registration page. The user should enter in a unique user name and password. After successfully registering, the user will then be taken to a login screen to log in.

   Requirements: REQ-1, REQ-2, REQ-3, REQ-4, REQ-5

2. **User Login** – Before playing Connect-Games, the user must login to the system through the Login page. This allows the system to know who the user is and the user's computer student.

   Requirements: REQ-6

3. **User Plays Computer Student Tic-Tac-Toe** – The player can choose to go first or second when playing against the computer student. The player that goes first will always be X with the player moving second being O. The game will continue with each player making a move until either the player wins, the computer student wins, or a tie occurs. Normal Tic-Tac-Toe [1] rules will apply. If there is a winner in the game the sequence of moves will be stored in the system and linked to the computer student for future use when making moves.

   Requirements: REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, REQ-12, REQ-17, REQ-18, REQ-19, REQ-20, REQ-21, REQ-22, REQ-23, REQ-24

4. **User Plays Computer Student Connect Four** – The player can choose to go first or second when playing against the computer student. The player that goes first will always be the color black with the player moving second being red. The game will continue with each player making a move until either the player wins, the computer student wins, or a tie occurs. Normal Connect Four rules will apply [2]. If there is a winner in the game the sequence of moves will be stored in the system and linked to the computer student for future use when making moves.

   Requirements: REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, REQ-12, REQ-17, REQ-18, REQ-19, REQ-20, REQ-25, REQ-26, REQ-27, REQ28

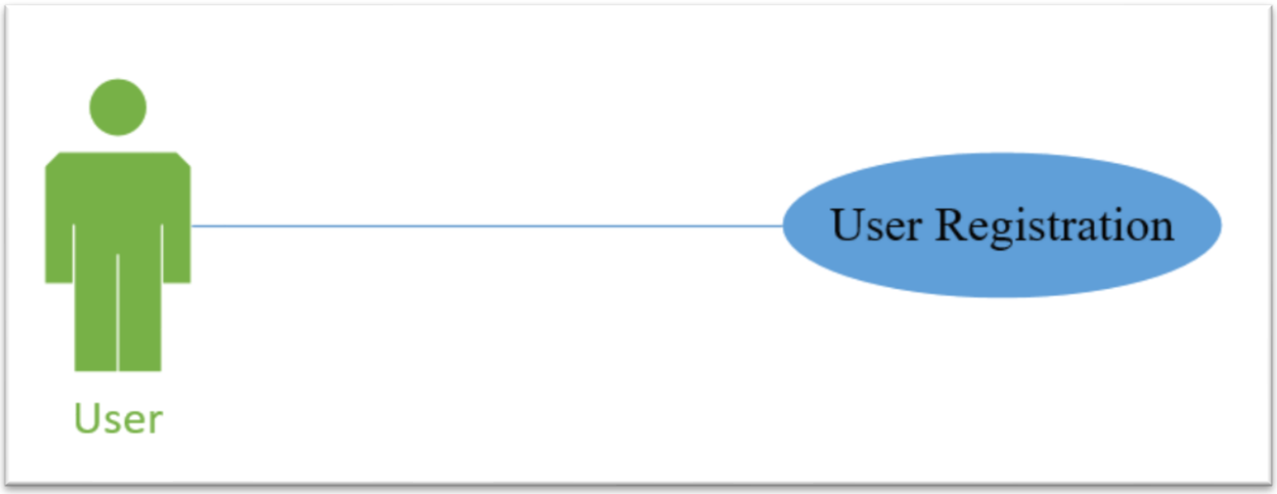
5. **User Sets up Match Between Computer Students** – If the user's computer student and another player's computer student have either Tic-Tac-Toe or Connect Four enabled, which can be done on the Match page, the user will then be able to have the user's computer student play against the other player's computer student with the results being displayed on the Match page. Note for this release the game between the computer students will take place out of sight of the user. In future

releases the user will be able to watch the computer students play in real time or as a replay of past matches.

Requirements: REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, REQ-13, REQ-14, REQ-15, REQ-16

ii. Use Case Diagrams

1. Use Case: User Registration



Use Case 1 – User registration.

Name:	User Registration
Participating Actor:	User (future player)
Entry Condition:	User clicks on the Registration link in the top head of the home page.
Exit Condition:	User has successfully provided a unique user name and password and the system has entered them.
Flow of Events:	1. The user goes to the home page for none registered/logged in users. 2. The user clicks the Registration link at the top menu bar. 3. User enters in a unique user name and password. 4. If the user enters in a user name that already exists they will need to pick another one. 5. If the user does not enter a password of at least one character long (future

	versions will have harder standards) then they will be prompted to enter a new password. 6. Upon completion of the registration form the user will be redirected to the Login screen.
Special Requirements:	None

2. **Use Case: User Login**

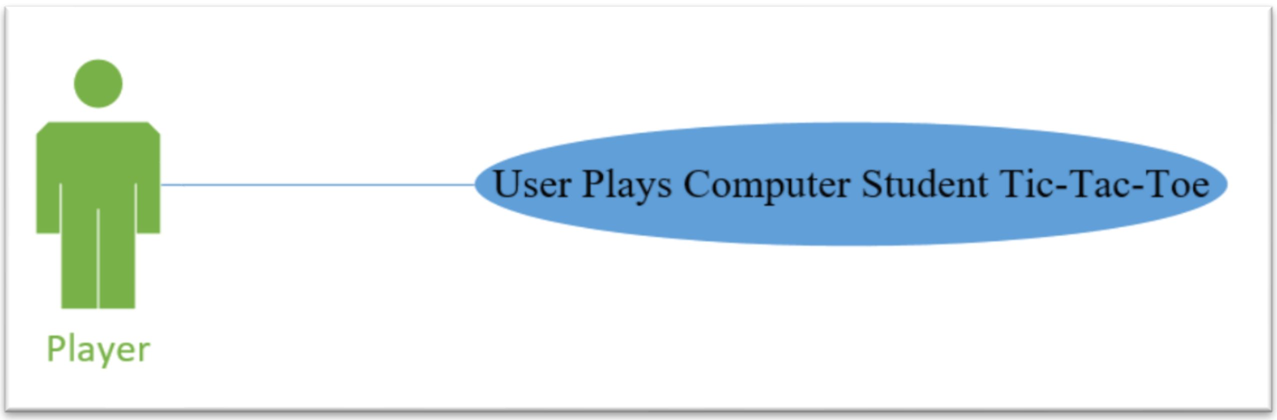
User Login
Player

Use case 2 – User logs in.

Name:	User Login
Participating Actor:	User/player
Entry Condition:	The user enters the login screen by way of the top navigation heading bar or after registration.
Exit Condition:	The user enters in the correct information and is taken to the home page for logged in users which is different than the default home screen in that other actions such as playing Tic-Tac-Toe are now available.
Flow of Events:	1. User arrives at the login screen via the Login link at the top of the page or after successfully filling out the registration form. 2. User enters in their user name and password into the user name and password fields.

	3. If the user name and password does not match any found in the system the user will be prompted to reenter their information (note that this version will not close their user's account if they enter the wrong password 3 times for example, but future versions will.) 4. When the user successfully logs into the application, the user will then be transferred to the home page where they can then play games and set up matches.
Special Requirements:	None

3. Use Case: User Plays Computer Student Tic-Tac-Toe



Use Case 3 – User plays against computer student in Tic-Tac-Toe.

Name:	User Plays Computer Student Tic-Tac-Toe
Participating Actor:	User/player
Entry Condition:	User pushes the button to either start first or start second when playing the game.
Exit Condition:	Either the user or the computer student wins or a tie occurs.
Flow of Events:	1. The user pushes one of two buttons to start the game. If the user goes first, then the application will wait until the user makes the first move by picking open location on the board. If the user goes second, then the computer student will move first, picking an open block to fill.

	2. The interaction will continue with the player and computer student taking turns picking a location until there is a winner or until all available locations on the board have been filled.
Special Requirements:	None

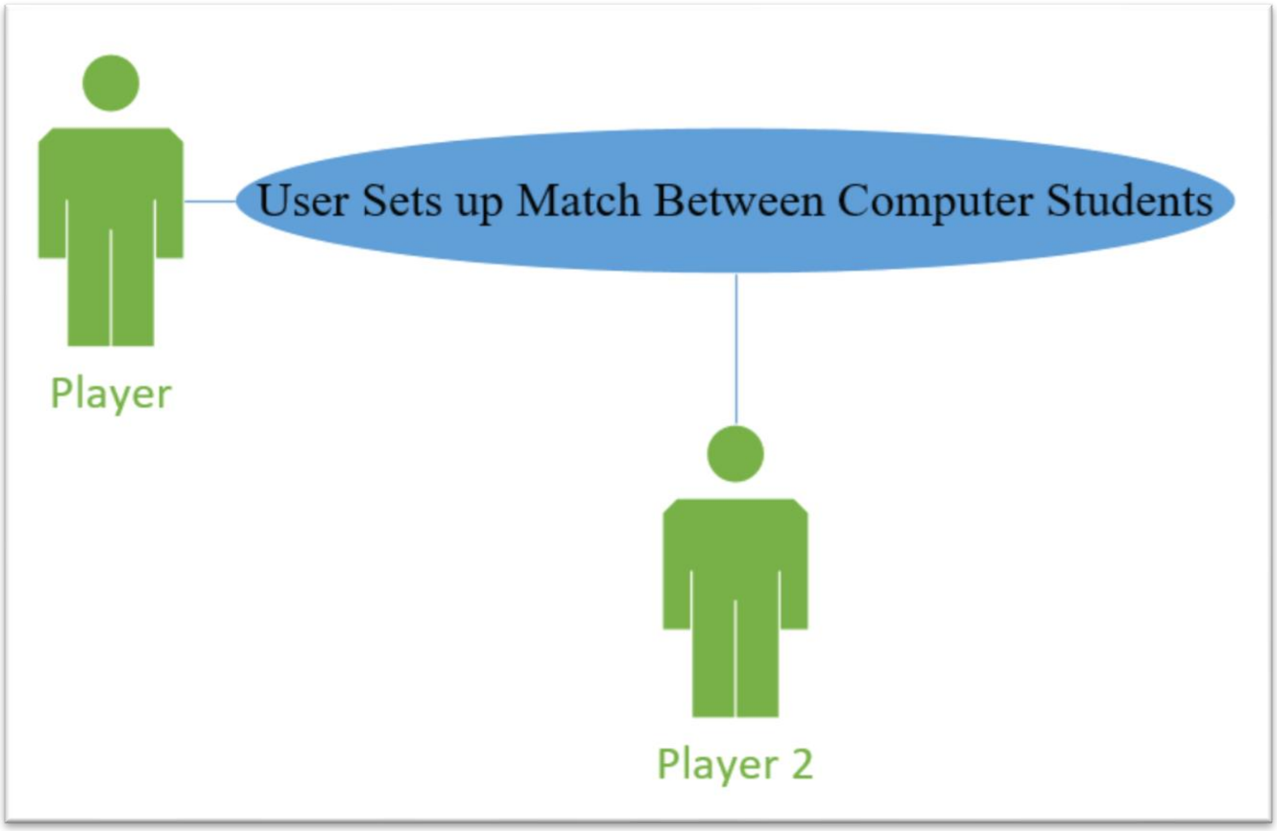
4. **Use Case: User Plays Computer Student Connect Four**

User Plays Computer Student Connect Four
Player

Use Case 4 – User plays computer student in Connect Four.

Name:	User Plays Computer Student Connect Four
Participating Actor:	User/player
Entry Condition:	User pushes the button to either start first or start second when playing the game.
Exit Condition:	Either the user or the computer student wins or a tie occurs.
Flow of Events:	1. The user pushes one of two buttons to start the game. If the user goes first, then the application will wait until the user makes the first move by picking open location on the board. If the user goes second, then the computer student will move first, picking an open block to fill. 2. The interaction will continue with the player and computer student taking turns picking a location until there is a winner or until all available locations on the board have been filled.
Special Requirements:	None

5. Use Case: User Sets up Match Between Computer Students



Use Case 6 – User sets up match between two computer students.

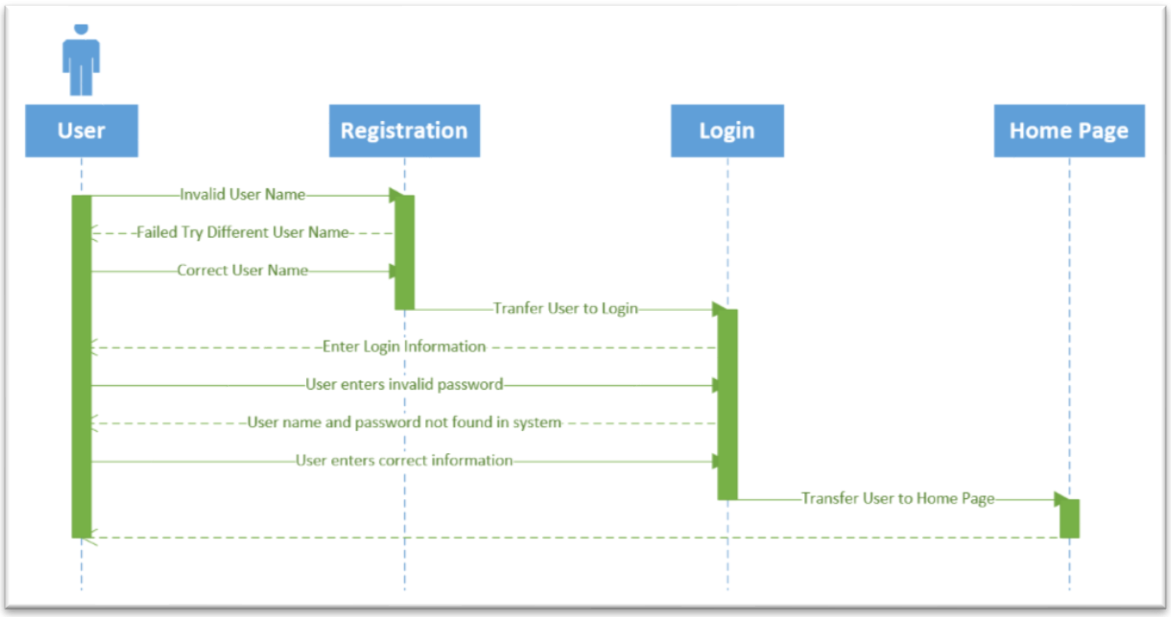
Name:	User Sets up Match Between Computer Students
Participating Actor:	User/player and indirectly another player that has enabled the computer student to player against other computer students.
Entry Condition:	From the Match screen, the user enables the computer student to play other computer students and then pushes the play button beside the list of available computer students to play.
Exit Condition:	The two computer students finish playing the game with one winning or a tie occurring.
Flow of Events:	1. The user enables the computer student to play either Tic-Tac-Toe and/or Connect Four. 2. The user will then see a list of available computer students to play

	against that have also been enabled to play. 3. The user pushes the play button beside the user's name to start the match between the two computer students 4. The results are shown on the Match screen after the match is over.
Special Requirements:	None

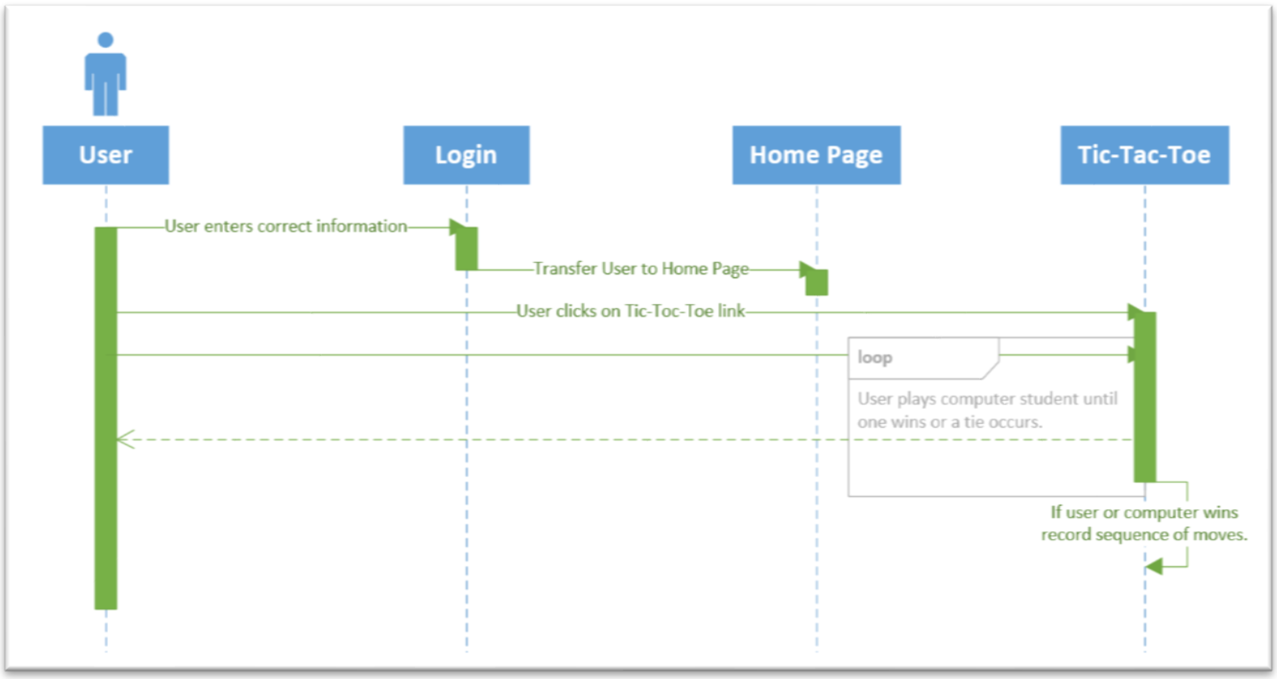
iii. Traceability Matrix

Use Case	**Requirements**
User Registration	REQ-1, REQ-2, REQ-3, REQ-4, REQ-5
User Login	REQ-6
User Plays Computer Student Tic-Tac-Toe	REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, REQ-12, REQ-17, REQ-18, REQ-19, REQ-20, REQ-21, REQ-22, REQ-23, REQ-24
User Plays Computer Student Connect Four	REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, REQ-12, REQ-17, REQ-18, REQ-19, REQ-20, REQ-25, REQ-26, REQ-27, REQ28
User Sets up Match Between Computer Students	REQ-7, REQ-8, REQ-9, REQ-10, REQ-11, REQ-13, REQ-14, REQ-15, REQ-16

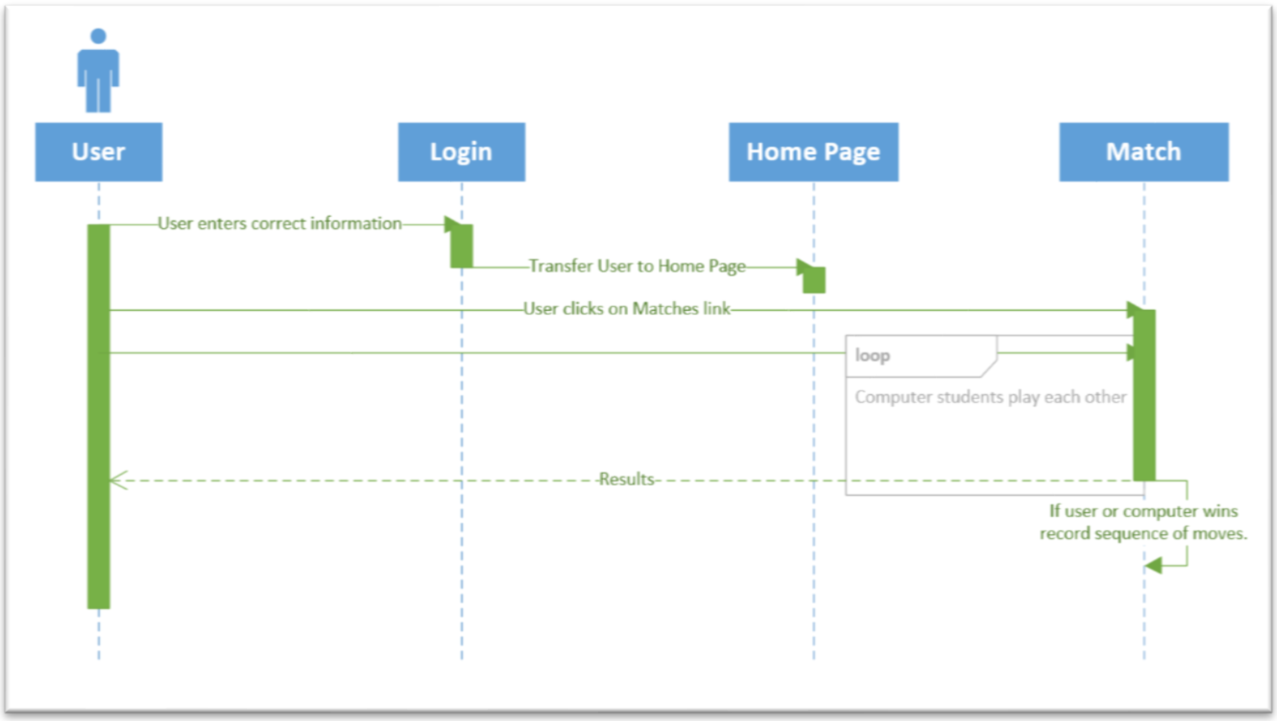
5.4 System Sequence Diagrams


User
Registration
Login
Home Page
Invalid User Name
Failed Try Different User Name
Correct User Name
Tranfer User to Login
Enter Login Information
User enters invalid password
User name and password not found in system
User enters correct information
Transfer User to Home Page

Sequence Diagram 1 - User registration and login.


User
Login
Home Page
Tic-Tac-Toe
User enters correct information
Transfer User to Home Page
User clicks on Tic-Toc-Toe link
loop
User plays computer student until one wins or a tie occurs.
If user or computer wins record sequence of moves.

Sequence Diagram 2 - User logs in and plays Tic-Tac-Toe.

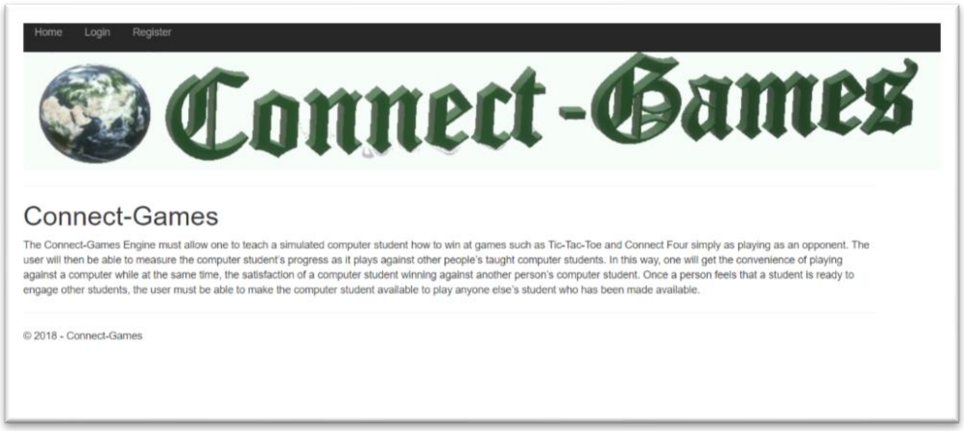

User
Login
Home Page
Match
User enters correct information
Transfer User to Home Page
User clicks on Matches link
loop
Computer students play each other
Results
If user or computer wins record sequence of moves.

Sequence Diagram 3 - User logs in and selects a match between computer students.

6.0 User Interface Specification

6.1 Preliminary Design

The following is a step-by-step flow of how a user registers, logs into, and plays Tic-Tac-Toe with the computer student.

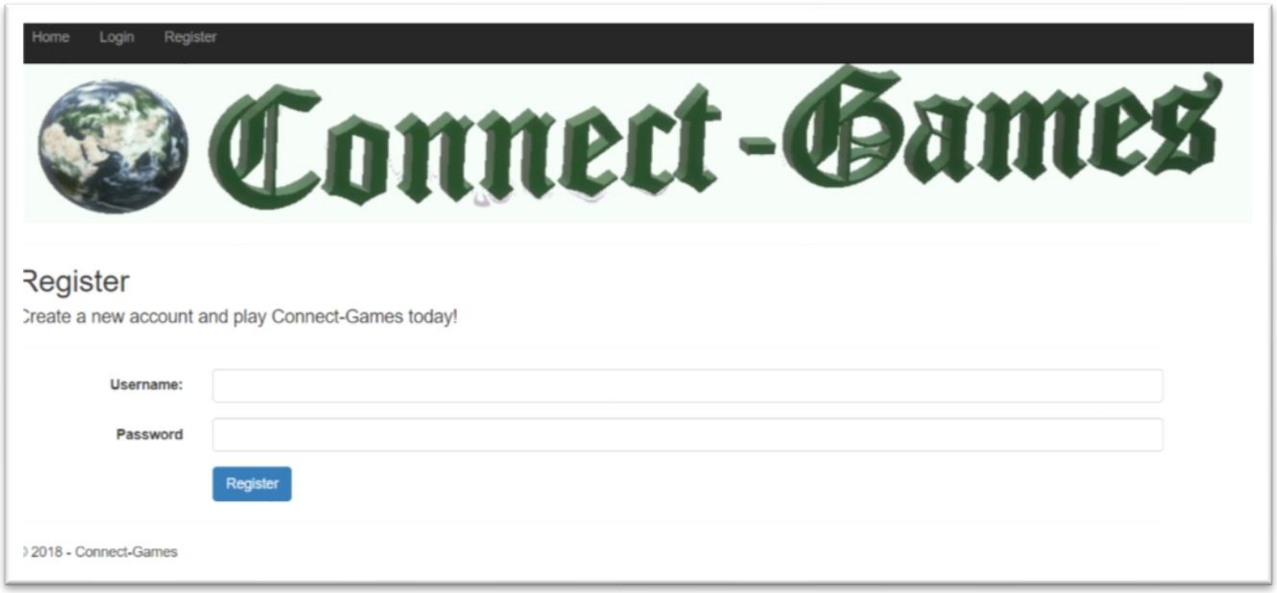


Connect-Games

The Connect-Games Engine must allow one to teach a simulated computer student how to win at games such as Tic-Tac-Toe and Connect Four simply as playing as an opponent. The user will then be able to measure the computer student's progress as it plays against other people's taught computer students. In this way, one will get the convenience of playing against a computer while at the same time, the satisfaction of a computer student winning against another person's computer student. Once a person feels that a student is ready to engage other students, the user must be able to make the computer student available to play anyone else's student who has been made available.

© 2018 - Connect-Games

Page 8 – The user goes to Connect-Games home page for non-logged in users.



Register

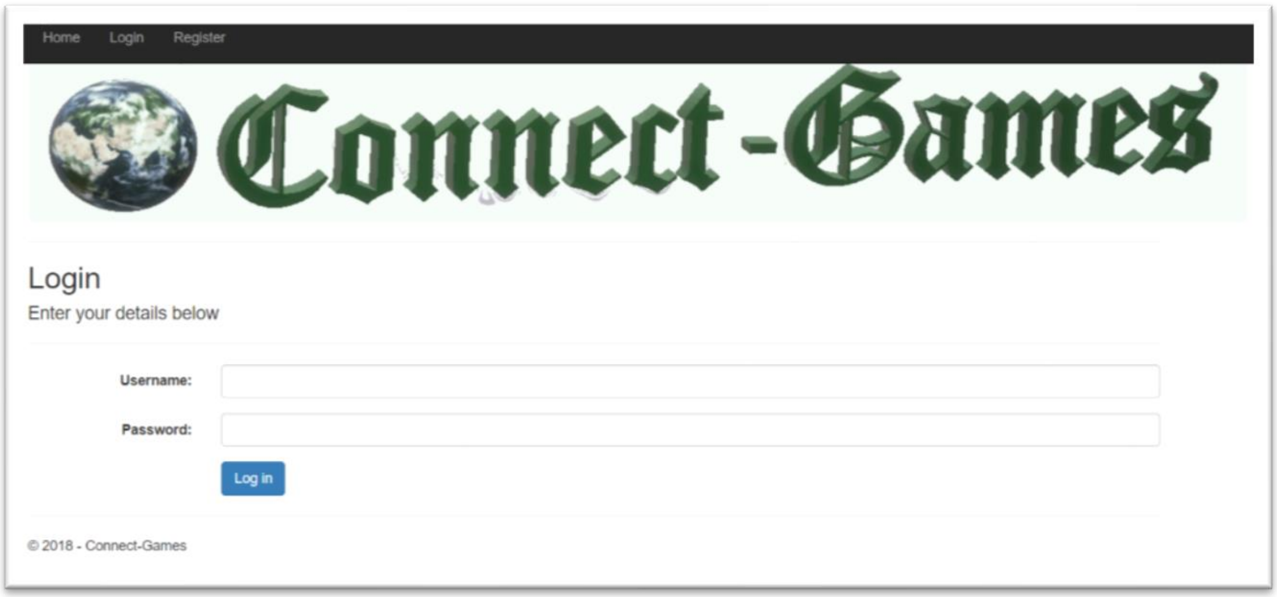
Create a new account and play Connect-Games today!

Username:

Password

© 2018 - Connect-Games

Page 9 – The user clicks on the "Register" link and is take to the Register page. The user then fills in a unique user name and password.



Login

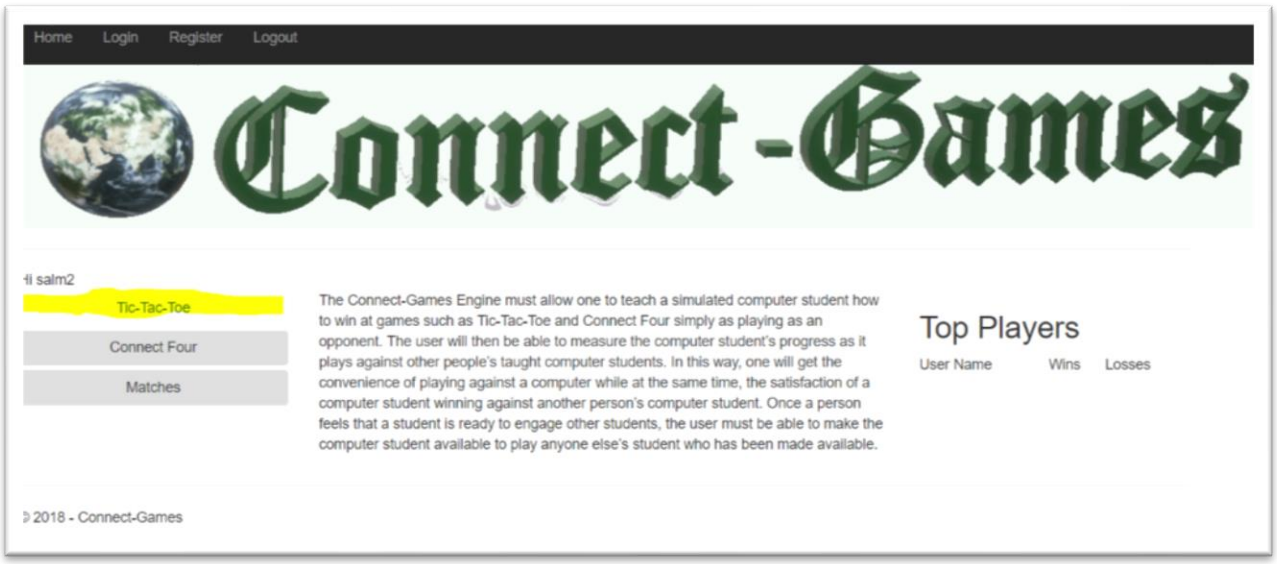
Enter your details below

Username:

Password:

© 2018 - Connect-Games

Page 10- The user is then transferred to the login page. They enter the user name and password used to register.



Hi salm2

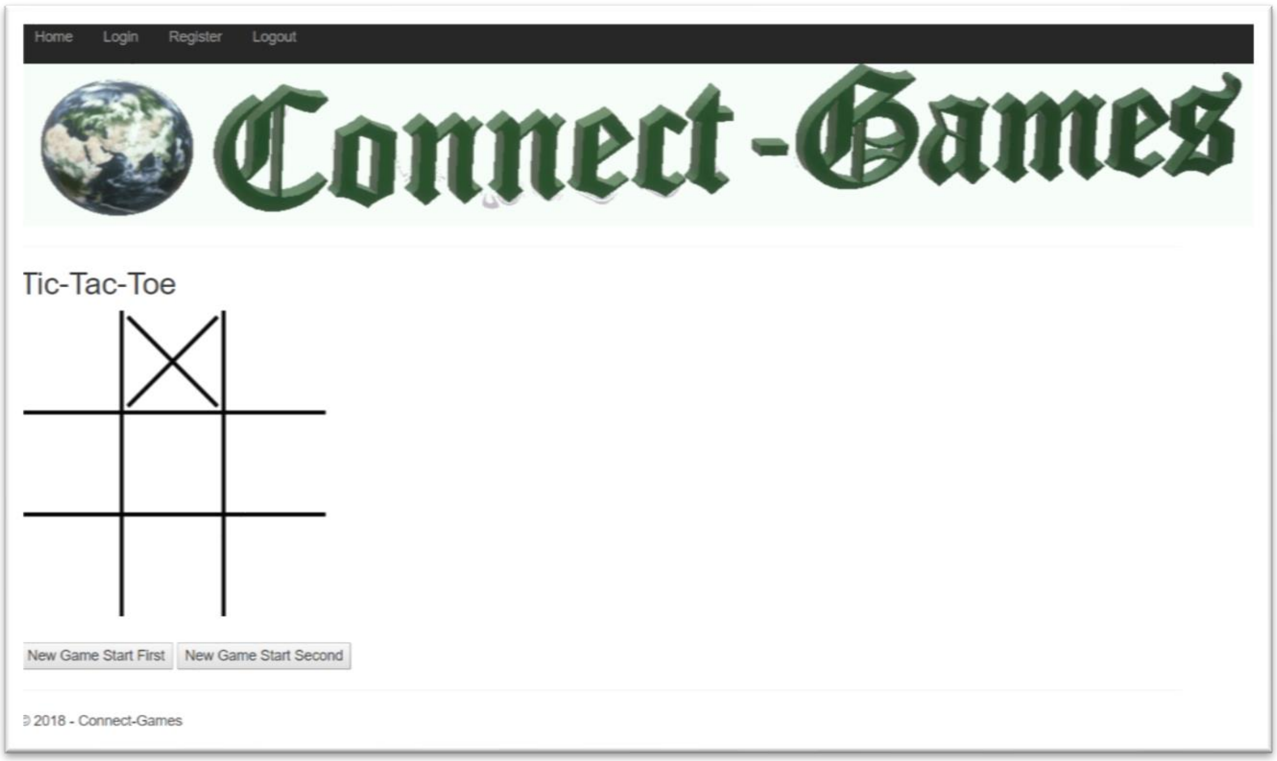
The Connect-Games Engine must allow one to teach a simulated computer student how to win at games such as Tic-Tac-Toe and Connect Four simply as playing as an opponent. The user will then be able to measure the computer student's progress as it plays against other people's taught computer students. In this way, one will get the convenience of playing against a computer while at the same time, the satisfaction of a computer student winning against another person's computer student. Once a person feels that a student is ready to engage other students, the user must be able to make the computer student available to play anyone else's student who has been made available.

Top Players

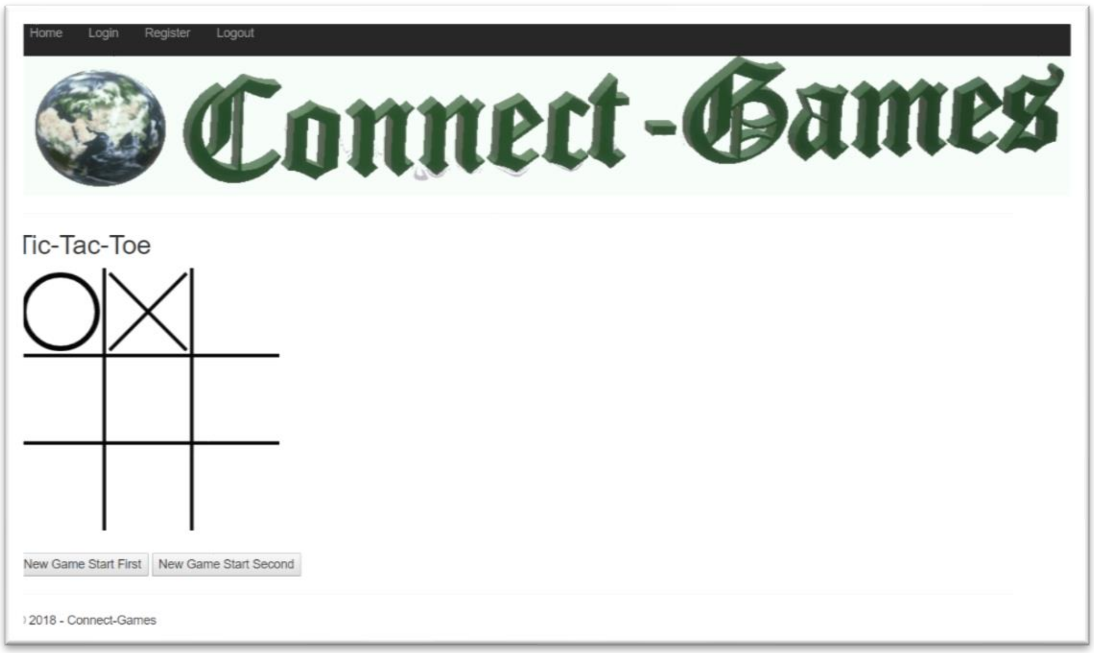
User Name	Wins	Losses

© 2018 - Connect-Games

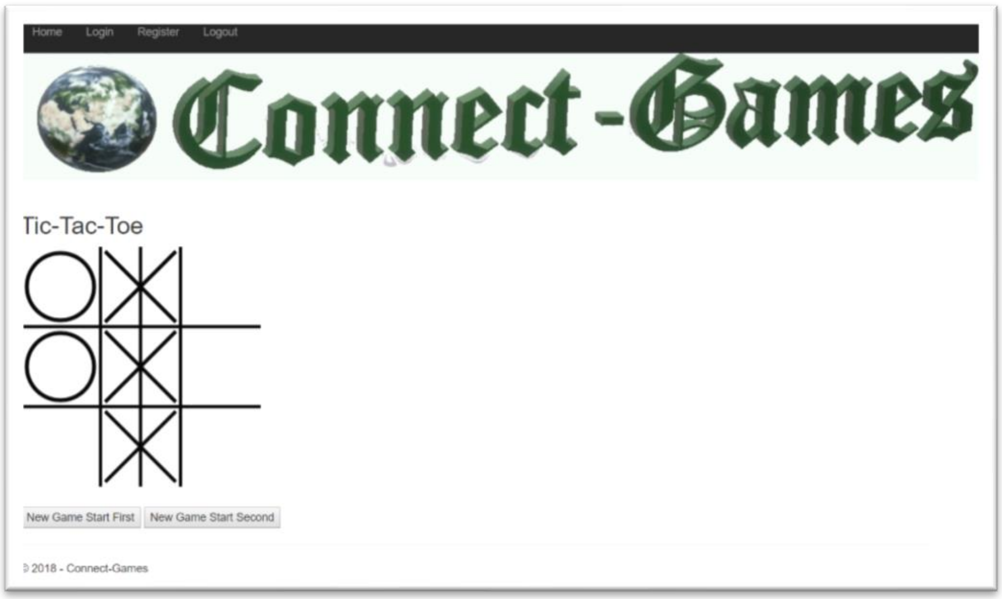
Page 11 – After logging in the user is taken to the Home screen. They then click on the Tic-Tac-Toe link.



Page 12- The user clicks on the "New Game Start Second" button and the computer student makes the first move.



Page 13 - The user clicks in a square that has not been taken yet in order to make a move. The computer student then takes a turn, etc. until someone wins, or a tie occurs.

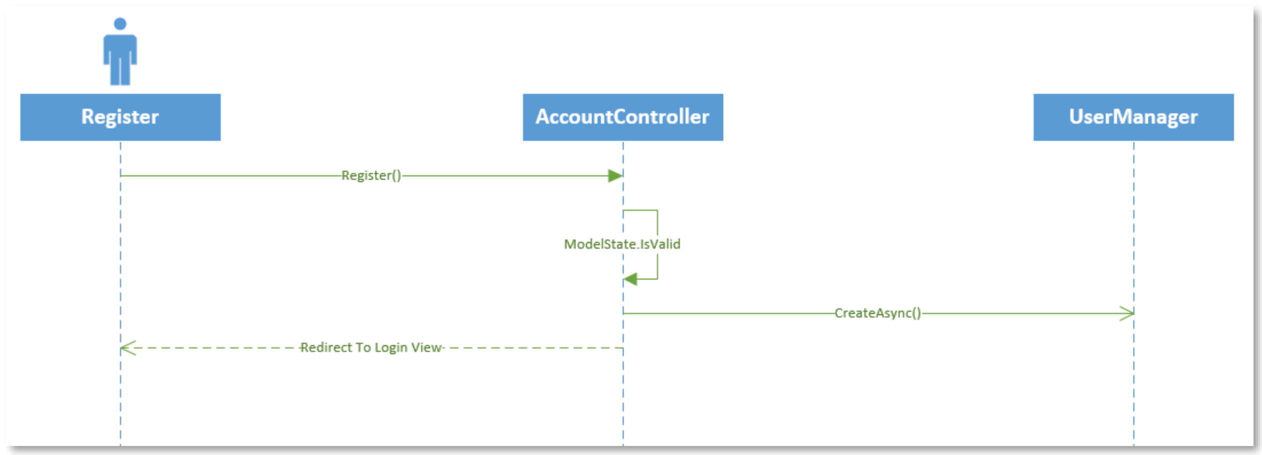


Page 14 - The computer student wins.

7.0 Sequence Diagrams

7.1 User Registration

In order to play Connect-Games, the user must first register with the system using the registration page. The user should enter in a unique user name and password. After successfully registering, the user will then be taken to a login screen to log in.

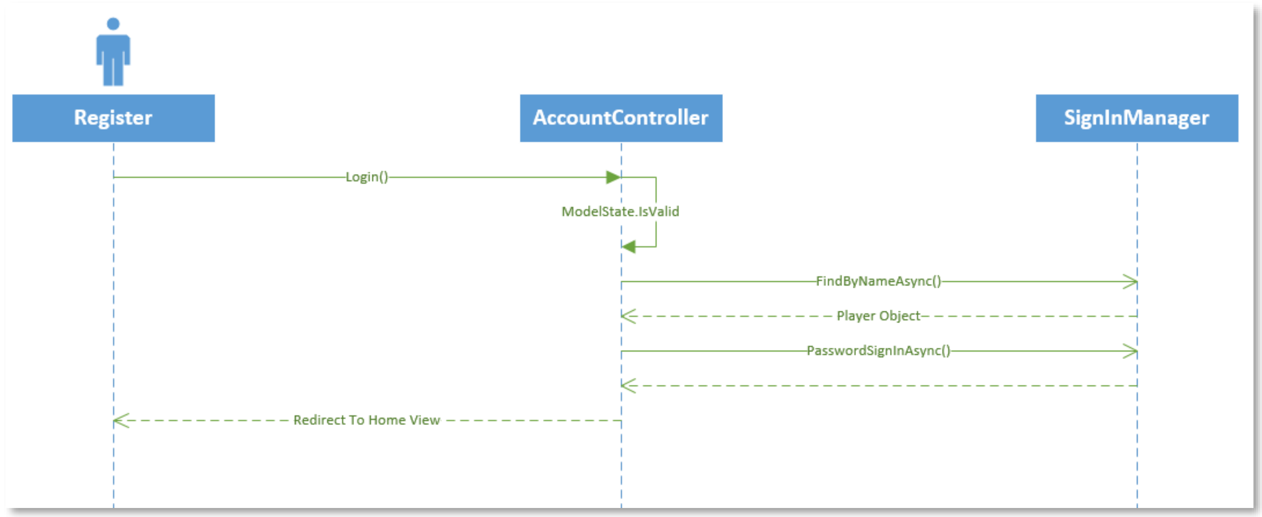


Sequence Diagram 4 – User registers with system.

Class	Design Principles/Responsibilities
AccountController	The AccountController is part of the MVC pattern and handles the interaction with the users regarding the users' registration, login, and logout process.
UserManager	The UserManager class is part of the ASP.NET Core framework 2.0 and handles the registration of the user with the system.

7.2 User Login

Before playing Connect-Games, the user must login to the system through the Login page. This allows the system to know who the user is and the user's computer student.

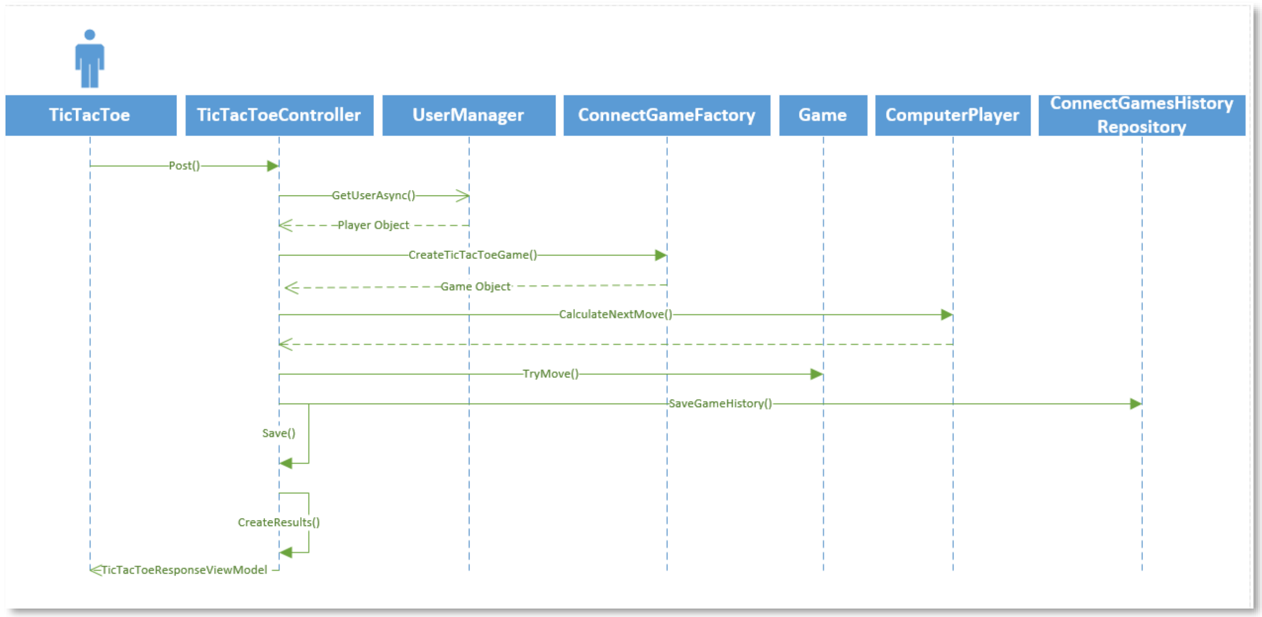


Sequence Diagram 5 – User logins in to system.

Class	Design Principles/Responsibilities
AccountController	The AccountController is part of the MVC pattern and handles the interaction with the users regarding the users' registration, login, and logout process.
SignInManager	The SignInManager class is part of the ASP.NET Core framework 2.0 and handles logging in and out and already registered user.

7.3 User Trains Computer Student at Tic-Tac-Toe

The user may choose to go first or second when training ot playing against the computer student. The player that goes first will always be X with the player moving second being O. The game will continue with each player making a move until either the player wins, the computer student wins, or a tie occurs. Normal Tic-Tac-Toe [1] rules will apply. If there is a winner in the game the sequence of moves will be stored in the system and linked to the computer student for future use when making moves.

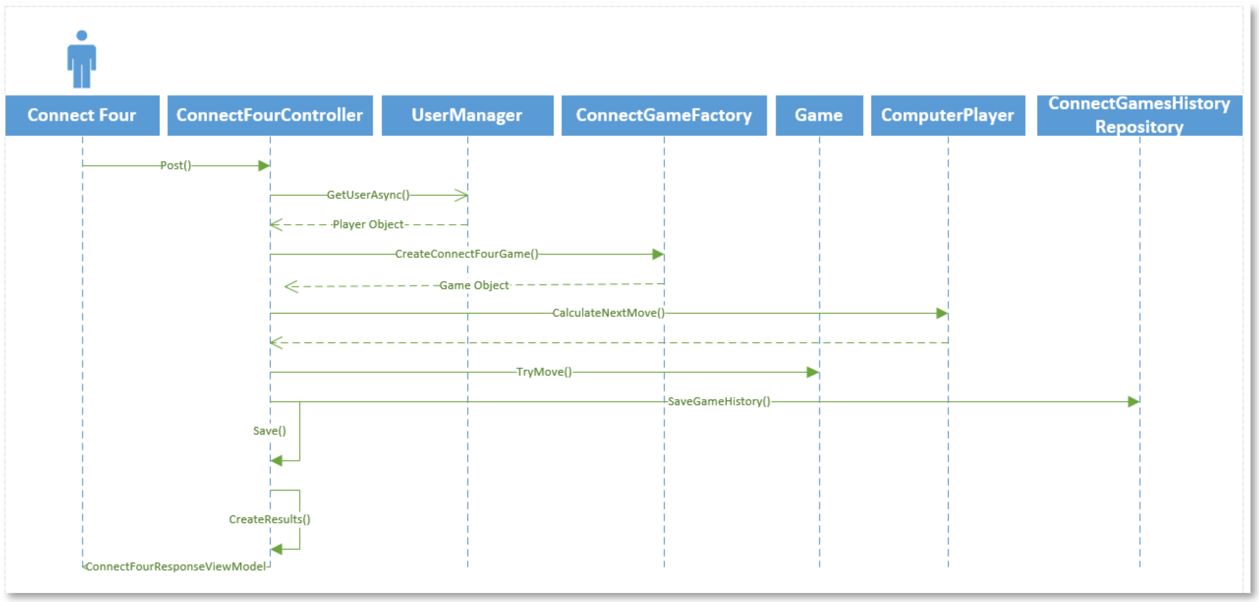


Sequence Diagram 6 – User trains computer student in Tic-Tac-Toe.

Class	**Design Principles/Responsibilities**
TicTacToeController	The TicTacToeController is part of the MVC pattern and handles the interaction with the users regarding playing the computer student. It also follows the SRP.
UserManager	The UserManager class is part of the ASP.NET Core framework 2.0 and handles the registration of the user with the system.
ConnectGameFactory	Based on the Abstract Factory pattern, creates a game object based on the parameter passed in.
Game	A Domain Object that controls and manages the state of a game.
ComputerPlayer	The AI part of the program whose algorithm determines the computer student's next move.
ConnectGamesHistoryRepository	Based on the Repository pattern, it handles the interaction with the database.

7.4 User Trains Computer Student at Connect Four

The user can choose to go first or second when playing against the computer student. The player that goes first will always be the color black with the player moving second being red. The game will continue with each player making a move until either the player wins, the computer student wins, or a tie occurs. Normal Connect Four rules will apply [2]. If there is a winner in the game the sequence of moves will be stored in the system and linked to the computer student for future use when making moves.

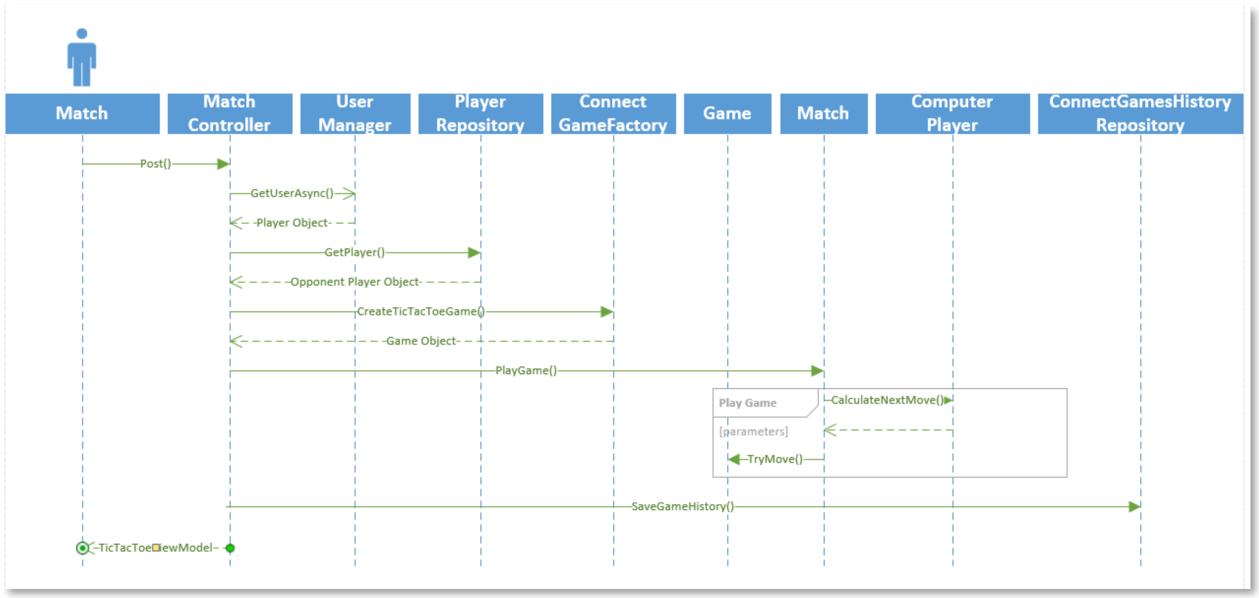


Sequence Diagram 7 – User trains computer student on how to play Connect Four.

Class	**Design Principles/Responsibilities**
ConnectFourController	The ConnectFourController is part of the MVC pattern and handles the interaction with the users regarding playing the computer student. It also follows the SRP.
UserManager	The UserManager class is part of the ASP.NET Core framework 2.0 and handles the registration of the user with the system.
ConnectGameFactory	Based on the Abstract Factory pattern, creates a game object based on the parameter passed in.
Game	A Domain Object that controls and manages the state of a game.
ComputerPlayer	The AI part of the program whose algorithm determines the computer student's next move.
ConnectGamesHistoryRepository	Based on the Repository pattern, it handles the interaction with the database.

7.5 User Sets Match Between Computer Students

The user can have the user’s computer student play against the other player’s computer student with the results being displayed on the Match page.



Sequence Diagram 8 – Match between computer students.

Class	**Design Principles/Responsibilities**
MatchController	The MatchController is part of the MVC pattern and sets up the match.
UserManager	The UserManager class is part of the ASP.NET Core framework 2.0 and handles the registration of the user with the system.
PlayerRepository	Based on the Repository pattern, it handles the interaction with the database
ConnectGameFactory	Based on the Abstract Factory pattern, creates a game object based on the parameter passed in.
Game	A Domain Object that controls and manages the state of a game.
Match	Handles running a game played by two computer students.
ComputerPlayer	The AI part of the program whose algorithm determines the computer student’s next move.
ConnectGamesHistoryRepository	Based on the Repository pattern, it handles the interaction with the database.

8.0 Summary mapping of Design Principles to Classes

Many of the design principles are applied to multiple classes the full picture of which will be shown in future class diagrams and other design documentation which is not featured in this document. For example, whenever possible, Connect-Games uses interfaces to interact with other domain models and uses dependency injection along with the Strategy pattern and DIP.

Class	**Design Principle Used**
AccountController	MVC, ISP, ADP, SDP, SAP
TicTacToeController	MVC, SRP, ISP, ADP, SDP, SAP
MatchController	MVC, ISP, ADP, SDP, SAP
ConnectFourController	MVC, SRP, ISP, ADP, SAP
PlayerRepository	Program to Interface, ISP
ConnectGameFactory	Abstract Factory, Program to Interface, ISP
Game	DRY, Program to Interface, ISP, OCP, DIP
Match	Program to Interface, ISP
ComputerPlayer	Strategy Pattern, Program to Interface, ISP, OCP, DIP
ConnectGamesHistoryRepository	Program to Interface, ISP

Table 1 – Mapping of classes used in sequence diagrams to design principles.

9.0 Class Diagram and Interface Specification

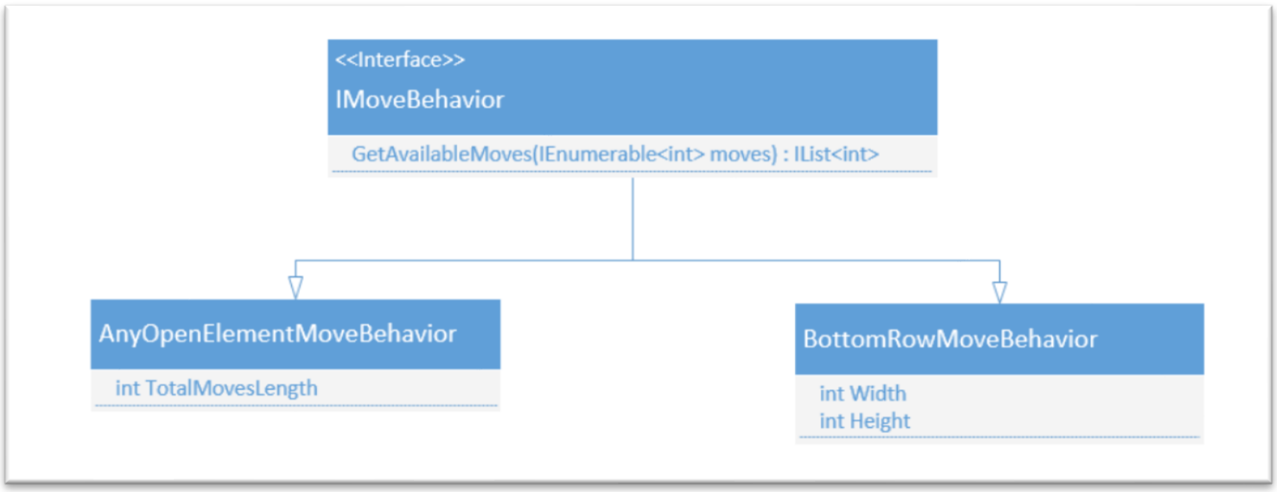
9.1 Class Diagrams

The following class diagrams are displayed at the component level and sometimes further broken down in order to best describe the class diagrams.

Connect-Games.DomainModels

The Connect-Games.DomainModels component is broken down into the following four class diagrams: Move Behaviors, Rules, AI, and Game with the game being the main class diagram that ties the others together.

Move Behaviors



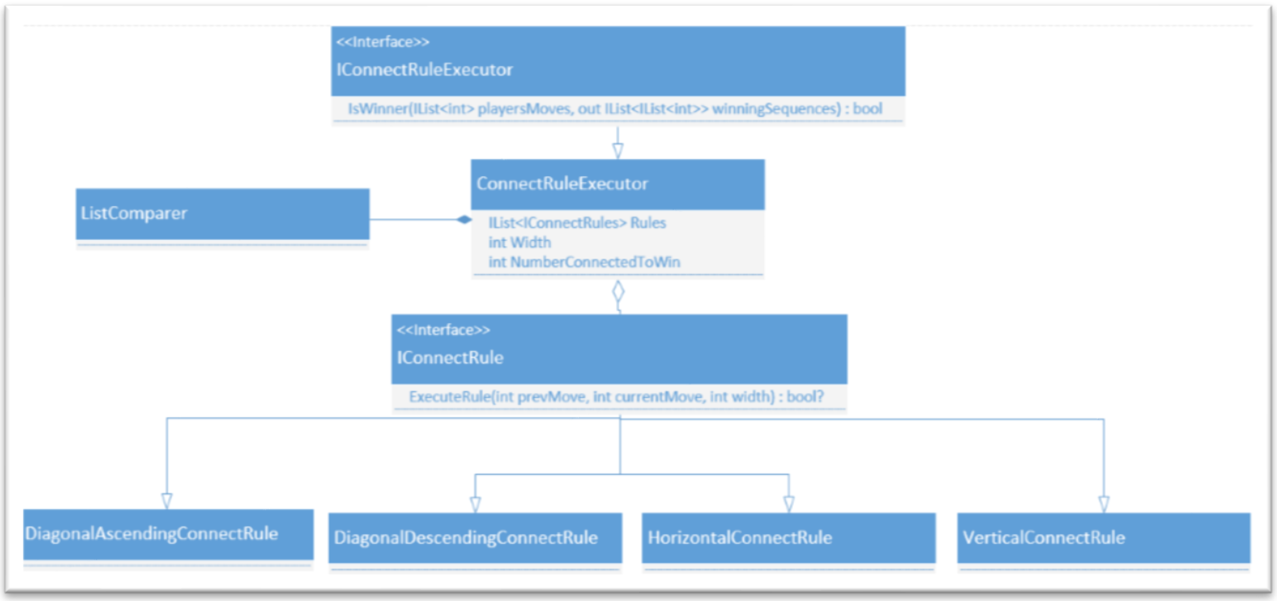
Class Diagram 1 – Move behaviors.

- **IMoveBehavior** - Interface defining the possible moves for a given behavior.
    - **IList<int> GetAvailableMoves(IEnumerable<int> moves)** - Returns the available moves given the moves that have already been made.

- **AnyOpenElementMoveBehavior** – Class that models the available moves when the player can move anywhere on the board. Example would be Tic-Tac-Toe.

- **BottomRowMoveBehavior** – Class that models the behavior allowed by a game such as Connect Four where the pieces are stacked on top of one another.

Rules


<<Interface>>
IConnectRuleExecutor
IsWinner(IList<int> playersMoves, out IList<IList<int>> winningSequences) : bool
ConnectRuleExecutor
IList<IConnectRules> Rules
int Width
int NumberConnectedToWin
ListComparer
<<Interface>>
IConnectRule
ExecuteRule(int prevMove, int currentMove, int width) : bool?
DiagonalAscendingConnectRule
DiagonalDescendingConnectRule
HorizontalConnectRule
VerticalConnectRule

Class Diagram 2 – Rules

- **IConnectRule** – Interface used to test if a connection is made.
    - **bool? ExecuteRule(int prevMove, int currentMove, int width) -** Will return true if the move results in matching the rule's connection rule.

- **DiagonalAscendingConnectRule -** Test for diagonal Right to Left connection (two elements in a row).

- **DiagonalDescendingConnectRule -** Test for diagonal Left to Right connection (two elements in a row).

- **HorizontalConnectRule -** Test for horizontal connection (two elements in a row).

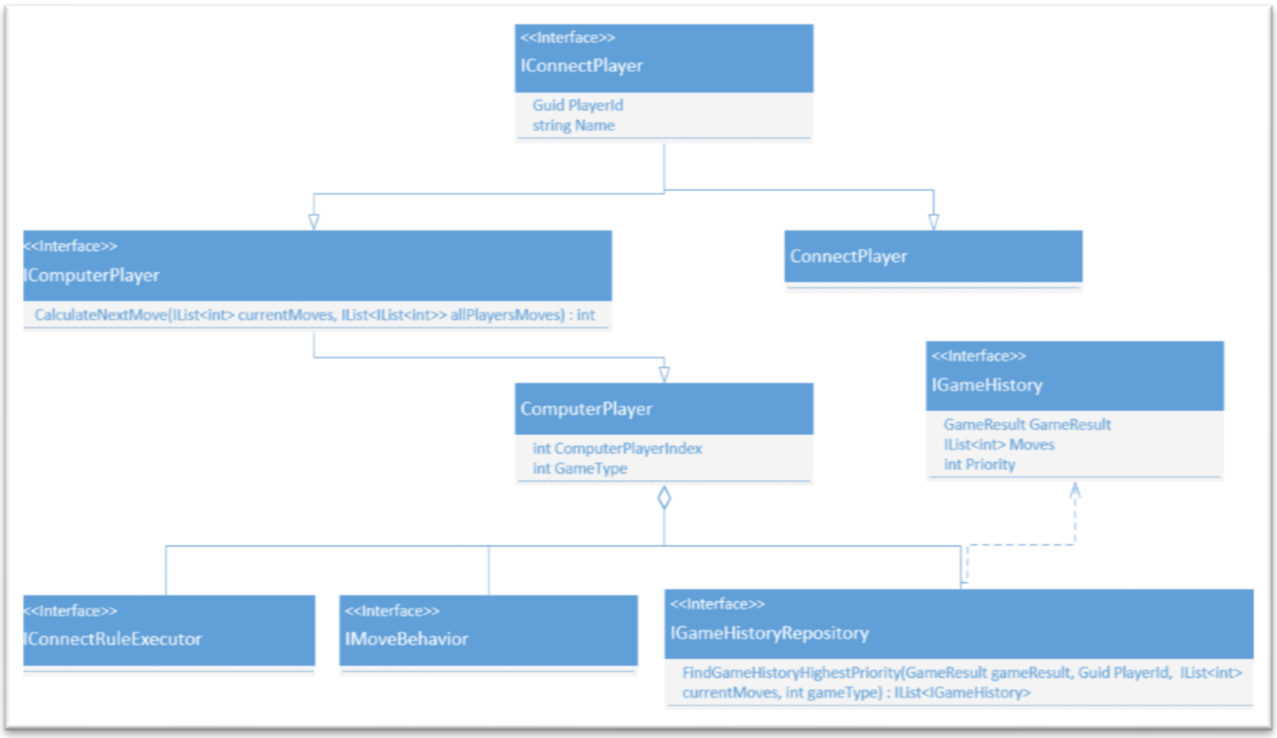
- **VerticalConnectRule -** Test for vertical connection (two elements in a column).

- **IConnectRuleExecutor -** Excutes connection rules to determine if there is a winner.
    - **bool IsWinner(IList<int> playersMoves, out IList<IList<int>> winningSequences)** – Determines if there is a winner.

- **ConnectRuleExecutor** – default implementation of IConnectRuleExecutor.

- **ListComparer -** Helper class for ConnectRuleExecutor determines if a sequence is equal.

AI



Class Diagram 3 – AI design

1. **IConnectPlayer** – Interface that represents a player in the game.

2. **ConnectPlayer** – Default implementation of IConnectPlayer.

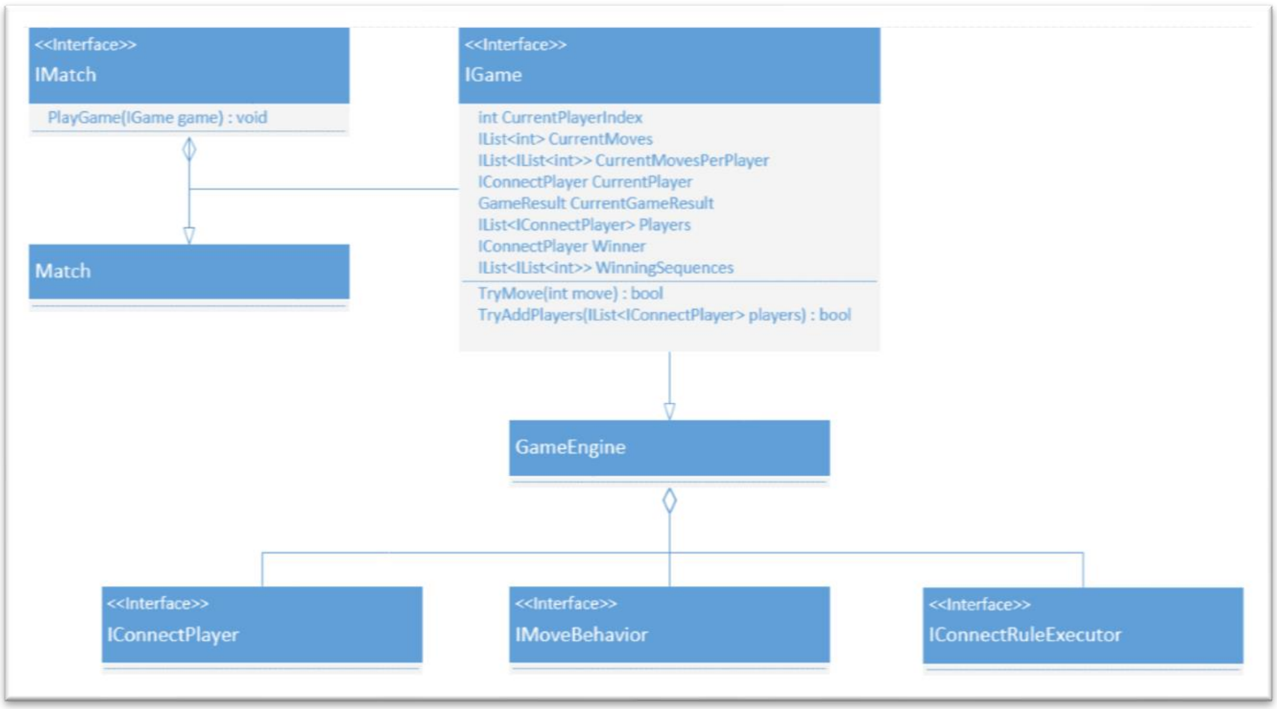
3. **IComputerPlayer** – Interface that represents a computer player in the game.
    - **int CalculateNextMove(IList<int> currentMoves, IList<IList<int>> allPlayersMoves)** – Calculates the best next move for the computer.

4. **ComputerPlayer** – Default implementation of IComputerPlayer.

5. **IGameHistory** – Interface that represents a game played.

6. **IGameHistoryRepository** – Interface that represents the needed storage methods for the computer player.
    - **IList<IGameHistory> FindGameHistoryHighestPriority(GameResult gameResult, Guid PlayerId, IList<int> currentMoves, int gameType)** – Finds a list of existing games that the computer has already played ordered by most used. Note that games are winning games.

Game


<<Interface>>
IMatch
PlayGame(IGame game) : void
Match
<<Interface>>
IGame
int CurrentPlayerIndex
IList<int> CurrentMoves
IList<IList<int>> CurrentMovesPerPlayer
IConnectPlayer CurrentPlayer
GameResult CurrentGameResult
IList<IConnectPlayer> Players
IConnectPlayer Winner
IList<IList<int>> WinningSequences
TryMove(int move) : bool
TryAddPlayers(IList<IConnectPlayer> players) : bool
GameEngine
<<Interface>>
IConnectPlayer
<<Interface>>
IMoveBehavior
<<Interface>>
IConnectRuleExecutor

Class Diagram 4 – The game engine and match.

- **IGame** – Interface that represents a connect game engine.
  - **bool TryMove(int move)** – tries to make the move passed in, will return false if the move is invalid.

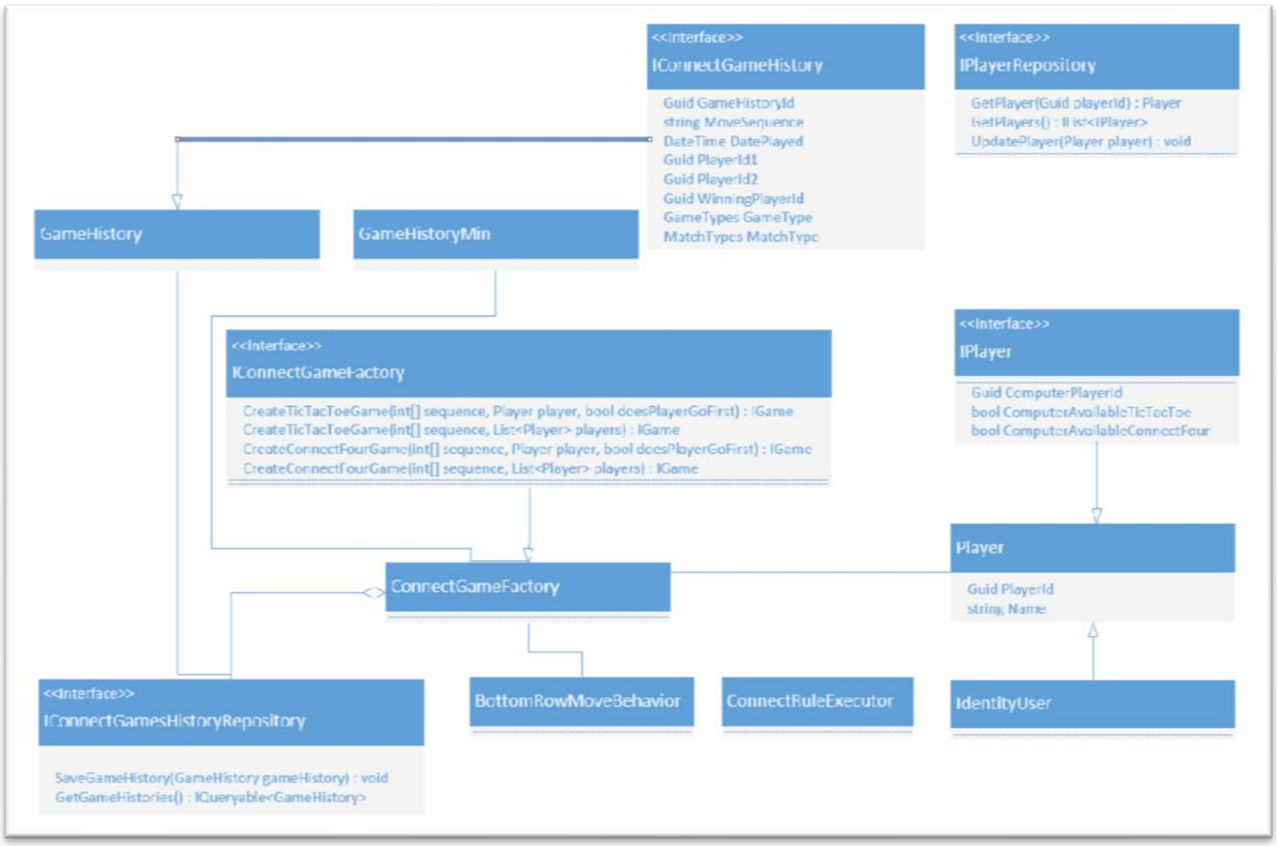
- **GameEngine** – Default implementation of an IGame.

- **IMatch** – Interface that represents a match between two computer players.
  - **void PlayGame(IGame game)** – plays the game between two computer players until one wins or a tie occurs.

- **Match –** Default implementation of IMatch.

Connect-Games.Games

The Connect-Games.Games component implements some of the interfaces in Connect-Games.DomainObjects component as well as contains factory classes and other game specific classes for Tic-Tac-Toe and Connect Four.



<<Interface>>
IConnectGameHistory
Guid GameHistoryId
string MoveSequence
DateTime DatePlayed
Guid PlayerId1
Guid PlayerId2
Guid WinningPlayerId
GameTypes GameType
MatchTypes MatchType
<<Interface>>
IPlayerRepository
GetPlayer(Guid playerId) : Player
GetPlayers() : IList<IPlayer>
UpdatePlayer(Player player) : void
GameHistory
GameHistoryMin
<<Interface>>
IConnectGameFactory
CreateTicTacToeGame(int[] sequence, Player player, bool doesPlayerGoFirst) : IGame
CreateTicTacToeGame(int[] sequence, List<Player> players) : IGame
CreateConnectFourGame(int[] sequence, Player player, bool doesPlayerGoFirst) : IGame
CreateConnectFourGame(int[] sequence, List<Player> players) : IGame
<<Interface>>
IPlayer
Guid ComputerPlayerId
bool ComputerAvailableTicTacToe
bool ComputerAvailableConnectFour
ConnectGameFactory
Player
Guid PlayerId
string Name
<<Interface>>
IConnectGamesHistoryRepository
SaveGameHistory(GameHistory gameHistory) : void
GetGameHistories() : IQueryable<GameHistory>
BottomRowMoveBehavior
ConnectRuleExecutor
IdentityUser

Class Diagram 5 – Connect-Games.Games component.

- **IConnectGameFactory** - Creates Tic-Tac-Toe and Connect Four IGame objects for player training computer student and computer student playing another computer student.
  - **IGame CreateTicTacToeGame(int[] sequence, Player player, bool doesPlayerGoFirst)** – Creates a Tic-Tac-Toe game between player and computer student.
  - **IGame CreateTicTacToeGame(int[] sequence, List<Player> players)** – Creates a Tic-Tac-Toe game between two computer students.
  - **IGame CreateConnectFourGame(int[] sequence, Player player, bool doesPlayerGoFirst)** – Creates a Connect Four game between a player and a computer student.
  - **IGame CreateConnectFourGame(int[] sequence, List<Player> players)** – Creates a Connect Four game between two computer students.

- **ConnectGameFactory** – Default implementation of IConnectGameFactory.

- **IConnectGamesHistoryRepository** – Interface that defines the database methods for the games history records.
    - **void SaveGameHistory(GameHistory gameHistory)** – Defines a method that knows how to save a game history record.
    - **IQueryable<GameHistory> GetGameHistories()** – Defines a method that knows how to query the database for GameHistory records.

- **IPlayerRepository** – Interface that defines methods for managing player records.
    - **Player GetPlayer(Guid playerId)** – Returns a player record.
    - **IList<IPlayer> GetPlayers()** – Returns a list of all the player records.
    - **void UpdatePlayer(Player player)** – Updates information in the player record.

- **IConnectGameHistory** – Interface that represents an abstract version of a game history record.

- **GameHistory** – Class that represents a record in the database.

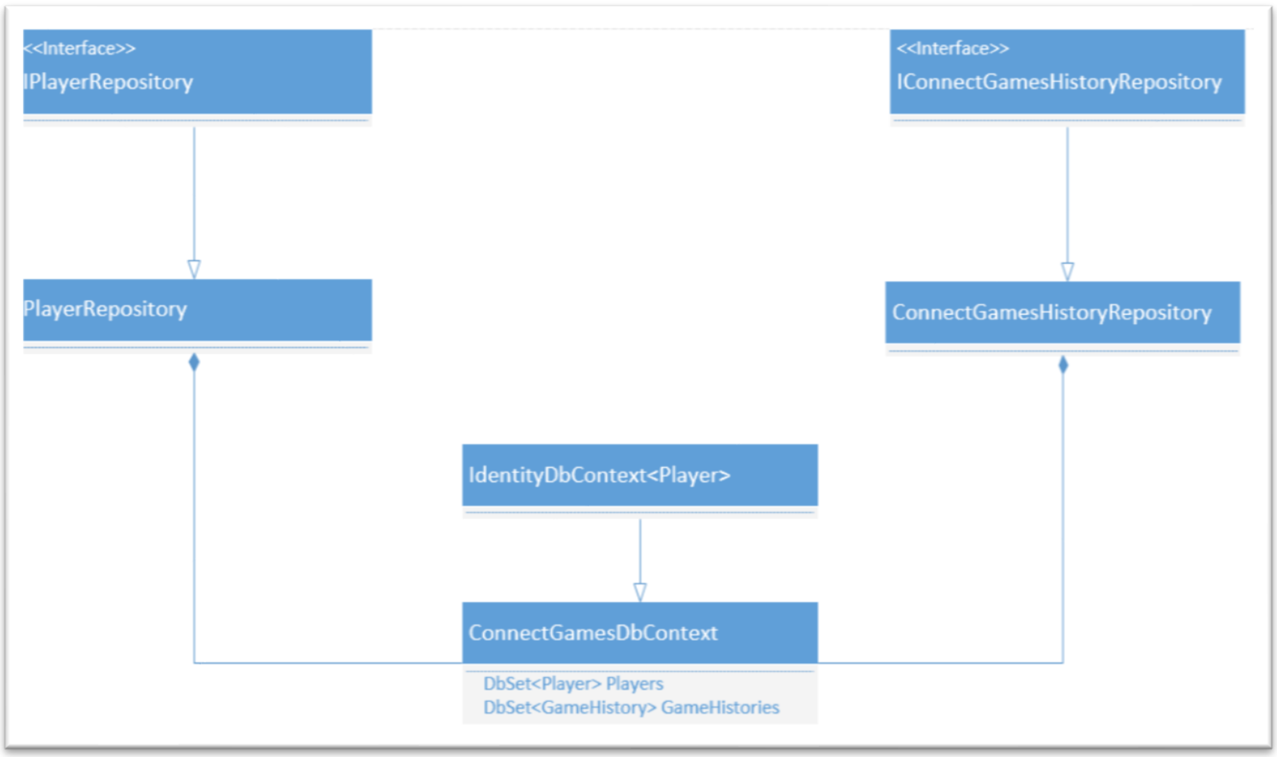
- **GameHistoryMin** – Class that represents a minimum version of IGameHistory.

- **IPlayer** – Interface that defines an abstract version of a player record.

- **Player** – Class the represents a record in the database.

Connect-Games.DAL

The Connect-Games.DAL component contains classes for implementing the IRepository classes such as IPlayerRepository and IConnectGamesHistoryRepository and for interacting with the database.

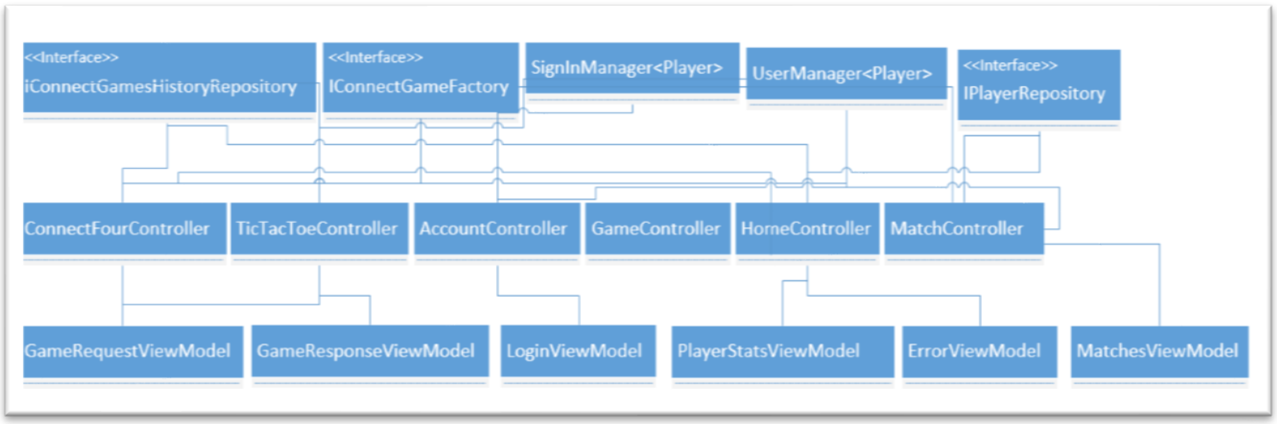

<<Interface>>
IPlayerRepository
<<Interface>>
IConnectGamesHistoryRepository
PlayerRepository
ConnectGamesHistoryRepository
IdentityDbContext<Player>
ConnectGamesDbContext
DbSet<Player> Players
DbSet<GameHistory> GameHistories

Class Diagram 6 – The Connect-Games.DAL component.

- **IdentityDbContext<Player>** - Part of the ASP.NET Core framework for managing users.
- **ConnectGamesDbContext** - Controls access to the database.
- **ConnectGamesHistoryRepository** – Implements IConnectGamesHistoryRepository.
- **PlayerRepository** – Implements IPlayerRepository.

Connect-Games

The Connect-Games component is the outer most component containing the ASP.NET Core MVC framework, JavaScript functions, and view models. It provides the interaction with the users.



Class Diagram 7 – Connect-Games component.

- **ConnectFourController** – Controller that providers a web API interface for training the computer student in Connect Four.
    - **Task<GameResponseViewModel> Post([FromBody] GameRequestViewModel connectFourRequestViewModel) –** Controls the management of a Connect Four game between the user and the computer student.

- **TicTacToeController** – Controller that providers a web API interface for training the computer student in Tic-Tac-Toe.
    - **Task<GameResponseViewModel> Post([FromBody] GameRequestViewModel ticTacToeRequestViewModel) -** Controls the management of a Tic-Tac-Toe game between the user and the computer student.

- **AccountController** – Controller that manages the user's login, logout, and registration processes.
    - **IActionResult Login(string returnUrl)** – Manages the view for the beginning Login page (before the user submits their information.)
    - **Task<IActionResult> Login(LoginViewModel loginViewModel) –** Manages the post back of the login screen to log the user into the game.
    - **IActionResult Register() –** Manages the view for the beginning of the Registration page (before the user enters their information.)
    - **Task<IActionResult> Register(LoginViewModel loginViewModel) –** Manages the post back of the registration page, registering the user.

    - **Task<IActionResult> Logout()** Manages logging out the user.

- **GameController** – Controller for displaying the Tic-Tac-Toe and Connect Four pages.
    - **IActionResult TicTacToe() –** Displays the view for the Tic-Tac-Toe game.
    - **IActionResult ConnectFour() –** Displays the view for the Connect Four game.

- **HomeController** – Controller for displaying the home pages when the user is logged in and not logged in.
    - **IActionResult Index() –** Displays the default home screen when the user is not logged on.
    - **Task<IActionResult> Home()** – Displays the home screen after the user has logged on.
    - **IActionResult Error()** – Displays a default unhandled error page.

- **MatchController** – Controller for displaying and managing the matches between computer students.
    - **Task<IActionResult> Index()** – Displays the default matches screen.
    - **Task<IActionResult> ToggleTicTacToeEnable() –** Handles toggling the user's computer students ability to play Tic-Tac-Toe matches.
    - **Task<IActionResult> ToggleConnectFourEnable() -** Handles toggling the user's computer students ability to play Connect Four matches.
    - **Task<IActionResult> PlayTicTacToeMatch(Guid playerId)** Manages the match between two computer students playing Tic-Tac-Toe.
    - **Task<IActionResult> PlayConnectFourMatch(Guid playerId) -** Manages the match between two computer students playing Connect Four.

- **ErrorViewModel** – View model for displaying unhandled errors.
- **GameRequestViewModel** – used as an input message for the training of computer students.
- **GameResponseViewModel** – used as an output message for the training of computer students.
- **LoginViewModel** – Container used to hold the user's login/registration information.
- **MatchesViewModel** – Container used to hold information about a match between computer students.
- **PlayerStatsViewModel** – Container used to hold information for the player's statistics.

10.0 Traceability Matrix

Classes	User Cases and Domain	Reason
UserManager<Player>, SignInManager<Player>, Player, AccountController, HomeController, LoginViewModel	User Log in, Log out, and Registration	The main classes for this functionality involves the use of ASP.NET MVC Core framework built in user management. The controller classes and view models here are to facilitate collecting information from the user regarding login and registration.
TicTacToeController, GameController, GameRequestViewModel, GameResponseViewModel, IMoveBehavior, AnyOpenElementMoveBehavior, IConnectRule, DiagonalAscendingConnectRule, DiagonalDescendingConnectRule, HorizontalConnectRule, VerticalConnectRule, IConnectRuleExecutor, IConnectPlayer, ConnectPlayer, IComputerPlayer, ComputerPlayer, IGameHistory IGameHistoryRepository, IGame GameEngine, IConnectGameFactory, ConnectGameFactory, IConnectGamesHistoryRepository, IPlayerRepository, IConnectGameHistory, GameHistory, GameHistoryMin, IPlayer Player, IdentityDbContext<Player>, ConnectGamesDbContext, ConnectGamesHistoryRepository, IConnectGamesHistoryRepository,	User trains computer student in Tic-Tac-Toe.	Core domain classes and interfaces are used throughout the game. These include ones related to moving, rules for available valid moves, computer student related classes to determine its next move, the game engine, and the game history related database classes. The TicTacToeController class is used to interact by way of Ajax calls to/from the user in order to manage the game.

PlayerRepository,		
ConnectFourController, GameController, GameRequestViewModel, GameResponseViewModel, IMoveBehavior, BottomRowMoveBehavior, IConnectRule, DiagonalAscendingConnectRule, DiagonalDescendingConnectRule, HorizontalConnectRule, VerticalConnectRule, IConnectRuleExecutor, IConnectPlayer, ConnectPlayer, IComputerPlayer, ComputerPlayer, IGameHistory IGameHistoryRepository, IGame8 GameEngine, IConnectGameFactory, ConnectGameFactory, IConnectGamesHistoryRepository, IPlayerRepository, IConnectGameHistory, GameHistory, GameHistoryMin, IPlayer Player, IdentityDbContext<Player>, ConnectGamesDbContext, ConnectGamesHistoryRepository, IConnectGamesHistoryRepository, PlayerRepository,	User trains computer student in Connect Four.	Core domain classes and interfaces are used throughout the game. These include ones related to moving, rules for available valid moves, computer student related classes to determine its next move, the game engine, and the game history related database classes. The ConnectFourController class is used to interact by way of Ajax calls to/from the user in order to manage the game.
MatchController, MatchesViewModel, IMoveBehavior, AnyOpenElementMoveBehavior, IConnectRule, DiagonalAscendingConnectRule, DiagonalDescendingConnectRule, HorizontalConnectRule, VerticalConnectRule	User sets up match between two computer students at Tic-Tac-Toe.	Core domain classes and interfaces are used throughout the game. These include ones related to moving, rules for available valid moves, computer student related classes to determine its next move, the game engine, and the game

IConnectRuleExecutor, IComputerPlayer, ComputerPlayer, IGameHistory IGameHistoryRepository, IGame GameEngine, IMatch, Match, IConnectGameFactory, ConnectGameFactory, IConnectGamesHistoryRepository, IPlayerRepository, IConnectGameHistory, GameHistory, GameHistoryMin, IPlayer Player, IdentityDbContext<Player>, ConnectGamesDbContext, ConnectGamesHistoryRepository, IConnectGamesHistoryRepository, PlayerRepository,		history related database classes. The match related controller and domain classes cause the computer students to play against each other.
MatchController, MatchesViewModel, IMoveBehavior, BottomRowMoveBehavior, IConnectRule, DiagonalAscendingConnectRule, DiagonalDescendingConnectRule, HorizontalConnectRule, VerticalConnectRule, IConnectRuleExecutor, IComputerPlayer, ComputerPlayer, IGameHistory IGameHistoryRepository, IGame GameEngine, IMatch, Match, IConnectGameFactory, ConnectGameFactory, IConnectGamesHistoryRepository, IPlayerRepository,	User sets up match between two computer students at Connect Four.	Core domain classes and interfaces are used throughout the game. These include ones related to moving, rules for available valid moves, computer student related classes to determine its next move, the game engine, and the game history related database classes. The match related controller and domain classes cause the computer students to play against each other.

IConnectGameHistory, GameHistory, GameHistoryMin, IPlayer Player, IdentityDbContext<Player>, ConnectGamesDbContext, ConnectGamesHistoryRepository, IConnectGamesHistoryRepository, PlayerRepository,		

11.0 System Architecture and System Design

11.1 Architectural Styles

Connect-Games is made from a multiple of architectural styles that are layered in such a way that dependency of the components flow inwards as viewed in the following figure 1.

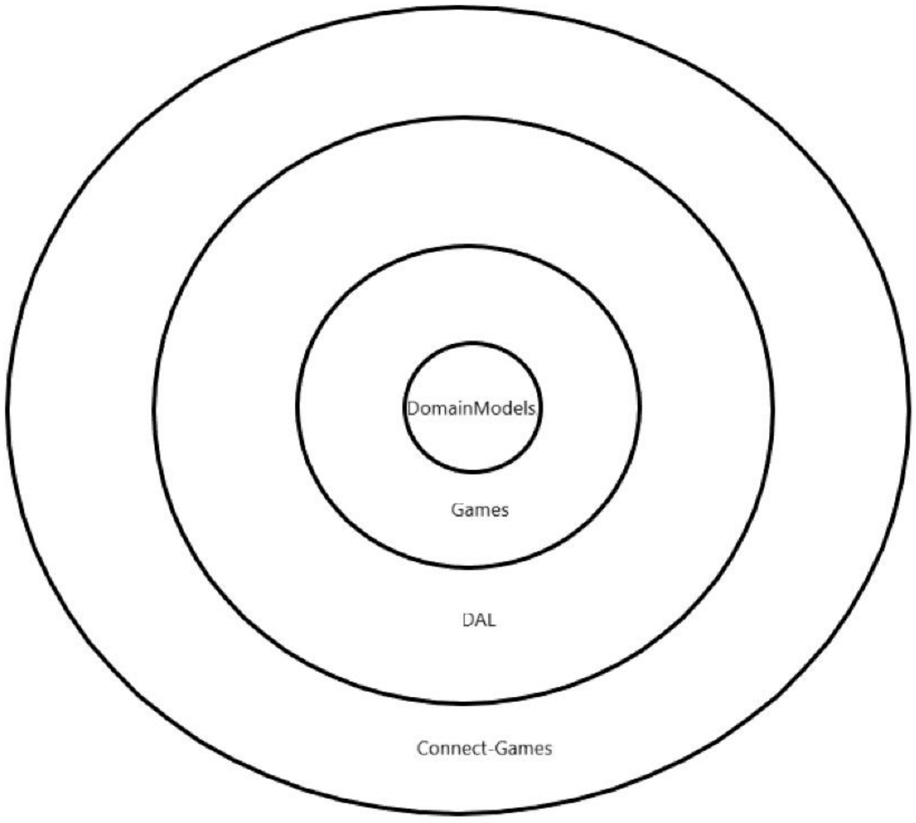


Figure 1 – Dependency layout of for the four components: Connect-Games.DomainModels, Connect-Games.Games, Connect-Games.DAL, and ConnectGames.

The domain models are the center of the design and have no dependencies with other code or frameworks. This component or layer contains the move behaviors, game rules, and game engine needed by Connect-Games and is fully unit testable. The architectural styles therefore include a layered approach, component structure, domain models, and MVC for the UI. These styles enable a testable, independent components and classes that allow maximum flexibility in design and coding.

11.2 Identifying Subsystems/Components

The classes and interfaces contained in the Connect-Games.DomainModels make up the core design of Connect-Games and depends on no frameworks or components other than the .NET Core framework 2.0 in which it is written in. The style is a program to an interface style where as much as possible classes work with abstractions or interfaces. This allows for framework independent, testable, UI independent, and database independent design and code. The Connect-Games.DomainModels can be broken up into four main areas of functionality: move behaviors, rules, AL, and game.

The move behaviors and rules allow for a mixing and matching of available moves and the rules that govern a game. Therefore, more games than Connect Four and Tic-Tac-Toe can be created simply by creating new types of move behaviors and rules, for example, 3D-Tic-Tac-Toe or new game never played before could be invented. The AI section of the component allows for the computer student to learn to play a game. It is also designed using interfaces that allow for different machine learning algorithms to be used in place of the simple default version described later in this paper. Finally, the games section has classes that model a game engine and provide the functionality for a match between to AI computer players. Each of these classes can be reused or replaced simply by providing a different implementation for the given interfaces.

The next layer, Connect-Games.Games, defines classes and interfaces to directly create Tic-Tac-Toe and Connect Four games. This layer, like the previous one, defines interfaces so as to cause dependency inversion defining what database methods are needed but not how to implement them. This layer also defines abstract factory methods for games.

The third layer, Connect-Games.DAL component, provides classes that access a database, in this case a SQL Server database. However, if needed, this layer could be replaced with a component that implements a non-SQL system for example, and no other code in the code would have to change.

The final layer, called simply Connect-Games, is an MVC architectural styles and is the user interface for the game. It includes Ajax calls as well and uses JavaScript on the client's browser in order to draw the game.

11.3 Hardware Requirements

A device capable of running the latest Chrome browser (most newer browsers will also due) with at least 500MB of RAM, with a color display and minimum resolution of 640 × 480 pixels and a network bandwidth of 56 Kbps as Connect-Games accesses the Internet. Connect-Games can run on most newer smart phones and tablets as well as PCs.

12.0 Algorithms and Data Structures

12.1 Algorithms

Connect-Games comes with a default algorithm for the computer student to pick its next move as follows:

7. The computer tests to see if any of its available moves will win the game. If so, that will be the computer's move.
8. The computer tests to see if any of the available moves will cause it to lose the game. If so, the computer will block the opponent from winning by making that move.
9. The computer checks to see if the current sequence of moves equals a sequence of moves so far in which the player won by retrieving records from the GameHistory records where it has played the game. If there is more than one sequence, then a random one is selected. If the computer finds that, based on its game history records, that it is current playing a sequence in which it will lose, then the computer if it is able to, will try a different move than the one in which it lost before.
10. Finally, if none of the above steps caused the computer to make a move, then a random move is selected from the available list of valid moves.

12.2 Data Structures

Connect-Games uses an array of numbers to track the players moves. These are then stored as a comma separated string in the database.

13.0 User Interface Design and Implementation

The only change to the interface was the removal of the player's game history section on the home page. This was due to space requirements and thought to be not very useful.

14.0 Test Cases

User Registration and Login			
Test Case Identifier	**Requirement to be Tested**	**Oracle**	**Expected Test Results**
TEST1	REQ-1 - User must be able to register in order to play the games.	• Success: If user tries to go to a page that is only allowed by users that are logged in such as http://www.connect-	The user cannot login without registering first and the user cannot play Connect-Games

		games.com/Match and they are redirected to the login screen. • Failure: If the user can play Connect-Games without registering and logging in.	without first logging in.
TEST2	REQ-2 - User must provide a unique user name.	• Success: The user provides a duplicate user name (a user name that already has been registered) and upon registering, is given an error messages telling them that they need to pick another user name. • Failure: The user can register with the same user name twice.	The user can not register the same username twice.
TEST3	REQ-3 - The user's name must be at least one character long for the initial version.	• Success: The user tries to register without providing a user name. A message is displayed saying that the user name is required. • Failure: The user is able to register without providing a user name.	The application displays a message that username is required when they don't enter one during registration.
TEST4	REQ-4 - User must also provide a password when registering to play the games.	• Success: When the user tries to register without providing a password the application displays a message that the password is required.	The user sees a password is required message when trying to register without providing a password.

		• Failure: The user is able to create a username and register without a password.	
TEST5	REQ-5 - Passwords, in the initial version, will have no strict rules as for the password's length or format other than needing to be at least one-character long.	• Success: The user enters a simple password, even as simple as one letter, and is able to register and login to Connect-Games. • Failure: The user cannot enter in a simple password to register.	The user enters in a simple password during registration and is successful redirected to the login page.
TEST6	REQ-6 - User must login using a user name and password already created through registration.	• Success: The user tries to login with a username that hasn't been registered yet and is given a username/password not found message. • Failure: The user tries to login with a username that hasn't been registered yet and is able to play the games.	The user enters a non-registered name and gets an error message that the username/password not found.
Computer Student's Behavior			
Test Case Identifier	**Requirement to be Tested**	**Oracle**	**Expected Test Results**
TEST7	REQ-7 - The computer student must keep a record of all completed games in which a player won, whether that player was the computer student or not.	• Success: The user plays a game against their computer student where one player wins. Then the user checks the GameHistories	After playing a winning game there is a record found of the game inside the GameHistories table of the ConnectGames database.

		table in the ConnectGames database to make sure that the record of the game was recorded. • Failure: The user plays a game against their computer student where one player wins. Then the user checks the GameHistories table in the ConnectGames database but finds no record of the game.	
TEST8	REQ-8 - The computer student must know the basic rules for the games, Tic-Tac-Toe and Connect Four. That is, the computer student must know a valid move and make only valid moves.	• Success: The user plays the computer student in Tic-Tac-Toc and Connect Four many times and sees that the computer student never makes an invalid move. • Failure: The user plays the computer student in Tic-Tac-Toe and/or Connect Four and see the computer student make an invalid move.	The user plays the computer student in Tic-Tac-Toe and Connect Four as they normally would play the game.
TEST9	REQ-9 - When making a move, the computer student must first detect if the computer student can win the game with the next move. If so, the computer student must make the winning move.	• Success: The user lets the computer student get two three in a row (depending on if the game is Tic-Tac-Toe or Connect Four) and sees the	The computer wins the game when it is possible for it to do so.

		computer student pick the square that will cause them to win the game the next time it is available. • Failure: The computer student has an opportunity to win but makes a different move instead.	
TEST10	REQ-10 - When making a move, the computer student must second detect if the computer student can lose the game with the next move. If so, the computer student must make a move to block the other player from winning.	• Success: The user makes two or three in a row (depending on if they are playing Tic-Tac-Toe or Connect Four). The next move of the computer student should be to block the user unless the computer can win. • Failure: The computer student fails to block the user from winning when the user only has one way to win.	The computer student blocks the user from winning when the computer student's next move cannot be a winning move and the user can only win one way.
TEST11	REQ-11 - When making a move, the computer student must third test whether or not the current sequence of moves matches a winning game that the computer student has played before. If so the student must follow that sequence using it to decide the next move.	• Success: The computer student uses the existing GameHistories records to decide its next move when the game sequence matches and existing one. • Failure: The computer student fails to follow an existing GameHistories	The computer student follows existing winning sequences to decide its next move when one exists and either player cannot currently win.

		record when it is available.	
TEST12	REQ-12 - When making a move, the computer student must forth and lastly, make a random valid move if REQ-9, REQ10, and REQ-11 are not able to be met.	• Success: If the computer student cannot win, loss, and there is no existing GameHistories records, then it will make a random move. • Failure: When first creating a user and their computer student if the computer student cannot make a move when starting a game.	When the user and computer student are first created via registration the computer student's first move should be random.
Computer Student vs Computer Student Matches			
Test Case Identifier	**Requirement to be Tested**	**Oracle**	**Expected Test Results**
TEST13	REQ-13 - A computer student must be able to be available or not available to play against other computer students which setting must be able to be controlled by the user.	• Success: When the user disables the computer student's ability to play a match on the match page, other users cannot play that student in the game. • Failure: Another user is able to play the computer student even though they are disabled.	The user name of a disabled computer student does not get displayed on the match page.
TEST14	REQ-14 - If the computer student is in a state of available to play other computer students, then another user can challenge that computer student to a match with the user's student.	• Success: A user can play an available (enabled) computer student. • Failure: A user cannot play an available (enabled) computer student.	The user pushes the play button beside an available (enabled) user name and the two computer students play a game.

TEST15	REQ-15 - In order to be able to challenge another user's computer student to a match, the user's computer student must be in the state of available to play.	• Success: A user can play an available (enabled) computer student. • Failure: A user cannot play an available (enabled) computer student.	The user pushes the play button beside an available (enabled) user name and the two computer students play a game.
TEST16	REQ-16 - When a match is played between computer students, a random pick will be made of who goes first.	• Success: When playing a game between two computer students, the application randomly picks one of them to go first. • Failure: When playing many games in a row between the same two students, one of the students always goes first.	When playing ten or more games in a row between the same two students, the application randomly picks one of them to start the game. Note that this could be a 70/30% split just as long as it's not 100% one student.
User Training Student			
Test Case Identifier	**Requirement to be Tested**	**Oracle**	**Expected Test Results**
TEST17	REQ-17 - A user must have one computer student.	• Success: After registering and logging in, the user is able to play their computer student by clicking on the "Tic-Tac-Toe" and "Connect Four" links on the home page and play their computer student. • Failure: After registering and logging in, the user is not able to play their computer student at Tic-Tac-Toe and Connect Four.	After registering and logging in, the user is taken to the Home page where they can see links to play Tic-Tac-Toe and Connect Four. After clicking on the link, they can then pick to go first or not and play their computer student.

TEST18	REQ-18 - The user must be able to play a game against the computer student at any time after logging in.	• Success: After registering and logging in, the user is able to play their computer student by clicking on the "Tic-Tac-Toe" and "Connect Four" links on the home page and play their computer student. • Failure: After registering and logging in, the user is not able to play their computer student at Tic-Tac-Toe and Connect Four.	After registering and logging in, the user is taken to the Home page where they can see links to play Tic-Tac-Toe and Connect Four. After clicking on the link, they can then pick to go first or not and play their computer student.
TEST19	REQ-19 - The computer student will store the sequence of moves made during the game if there is a win.	• Success: The user plays a game against their computer student where one player wins. Then the user checks the GameHistories table in the ConnectGames database to make sure that the record of the game was recorded. • Failure: The user plays a game against their computer student where one player wins. Then the user checks the GameHistories table in the ConnectGames database but finds	After playing a winning game there is a record found of the game inside the GameHistories table of the ConnectGames database.

		no record of the game.	
TEST20	REQ-20 - The user must be able to decide whether or not to go first when playing against the computer student.	• Success: When playing the computer student at Tic-Tac-Toe or Connect Four the user is able to push from one of two buttons that let the user go first or the computer. • Failure: When playing the computer student at Tic-Tac-Toe or Connect Four there is no buttons available to play for the user to choose to go first or second.	When playing the computer student at Tic-Tac-Toe or Connect Four the user is able to push from one of two buttons that let the user go first or the computer. When the user clicks on the button to go first the program waits for the user to make a move. Else, if the user clicks on the button allowing the computer student to make the first move then the program makes the first move.
Tic-Tac-Toe Game Requirements			
Test Case Identifier	**Requirement to be Tested**	**Oracle**	**Expected Test Results**
TEST21	REQ-21 - The player that goes first will always be X.	• Success: Whoever goes first in a game, the user or computer student, their piece will be an X. • Failure: Whoever goes first in a game, the user or computer student, their piece is an O.	The first player to move has an X for a piece.
TEST22	REQ-22 - The player that moves second will always be O.	• Success: Whoever goes second in a game, the user or computer student, their piece will be an O.	The second player to move has an O for a piece.

		• Failure: Whoever goes second in a game, the user or computer student, their piece is an X.	
TEST23	REQ-23 - The default rules as described in the glossary must apply to all games and matches in Tic-Tac-Toe.	• Success: The user plays the computer student in Tic-Tac-Toc many times and sees that the computer student never makes an invalid move. • Failure: The user plays the computer student in Tic-Tac-Toe and see the computer student make an invalid move.	The user plays the computer student in Tic-Tac-Toe as they normally would play the game.
TEST24	REQ-24 - As soon as there is a winner or there are no more valid moves available to be made, the game must end.	• Success: The game ends when either a player wins or there are no more available moves to make. • Failure: The game continues even though a player won the game.	The player is not able to make a move after the game ends.
Connect Four Game Requirements			
Test Case Identifier	**Requirement to be Tested**	**Oracle**	**Expected Test Results**
TEST25	REQ-25 - The player that moves first will always be the color black.	• Success: Whoever goes first in a game, the user or computer student, their piece will be black. • Failure: Whoever goes first in a game, the user or	The first player to move has black for a piece.

		computer student, their piece is an red.	
TEST26	REQ-26 - The player that move second will always be the color red.	• Success: Whoever goes second in a game, the user or computer student, their piece will be red. • Failure: Whoever goes second in a game, the user or computer student, their piece is black.	The second player to move has red for a piece.
TEST27	REQ-27 - The default rules as described in the glossary must apply to all games and matches in Connect Four.	• Success: The user plays the computer student in Connect Four many times and sees that the computer student never makes an invalid move. • Failure: The user plays the computer student in Connect Four and see the computer student make an invalid move.	The user plays the computer student in Connect Four as they normally would play the game.
TEST28	REQ-28 - As soon as there is a winner or there are no more valid moves available to be made, the game must end.	• Success: The game ends when either a player wins or there are no more available moves to make. • Failure: The game continues even though a player won the game.	The player is not able to make a move after the game ends.

15.0 Test Cases Steps

TEST1

Step #	Step Description	Input (data)	Results
1	User goes to www.connect-games.com in a Chrome browser.	URL	The user sees the home page.
2	User clicks on the Register ink at the top menu bar.		The user goes to the Registration page.
3	The user enters a unique username and password and pushes the Register button.	Username and password.	The user is registered with the game and is taken to a logon screen.

Result Summary

The test passed without any problems.

TEST2

Step #	Step Description	Input (data)	Results
1	User goes to www.connect-games.com in a Chrome browser.	URL	The user sees the home page.
2	User clicks on the Register ink at the top menu bar.		The user goes to the Registration page.
3	The user enters a duplicate username and pushes the Register button.	Username and password.	The user sees an error message displayed "Username already exists. Please pick a different name."

Result Summary

The test passed without any problems.

TEST3

Step #	Step Description	Input (data)	Results

1	At the Registration screen, the user does not enter any data for user name and password.		The user sees an error message saying that Username and Password are required.

Result Summary

The test passed without any problems.

TEST4

Step #	Step Description	Input (data)	Results
1	At the Registration screen, the user does not enter any data for password.	Username	The user sees an error message saying that password is required.

Result Summary

The test passed without any problems.

TEST5

Step #	Step Description	Input (data)	Results
1	At the Registration screen, the user enters a unique username and a one letter password.	Username and password.	The user is registered with the game and is taken to a logon screen.

Result Summary

The test passed without any problems.

TEST6

Step #	Step Description	Input (data)	Results

1	At the login screen, the user enters in a username and password that has already been registered and pushes the Login button.	Username and password.	The user is taken to the logged in Home page.

Result Summary

The test passed without any problems.

TEST7

Step #	Step Description	Input (data)	Results
1	The user creates a new username and password and logs in.	Username, password.	User taken to the Home screen.
2	The user clicks on the Tic-Tac-Toe link.		The user is taken to the Tic-Tac-Toe screen.
3	The user chooses to go first by pushing the “New Game Start First” button and clicks on the location where they want to put their X.	X	The user sees their X and the computer student picks a random place to move.
4	The user continues until they win.	X’s	The game ends and the record are written to the GameHistories table.
5	The user can see that the record was stored by clicking on the “New Game Start Second” button allowing the computer student to go first.	O’s	Because there is only one winning record in the table, the computer student’s first move is the same first move that the user made in the previous game.

Result Summary

This test is harder as the easiest way to confirm that the records are being saved is to check the database. However, this test passed as written.

TEST8

Step #	Step Description	Input (data)	Results
1	The user plays the computer student in multiple games.	X’s and O’s	The computer student never makes an invalid move.

Result Summary

The test passed without any problems.

TEST9

Step #	Step Description	Input (data)	Results
1	The user allows the computer student to get two in a row for Tic-Tac-Toe or three in a row for Connect Four	X’s	The computer student picks the winning move and wins the game.

Result Summary

The test passed without any problems.

TEST10

Step #	Step Description	Input (data)	Results
1	The user places two in a row for Tic-Tac-Toe and three in a row for Connect Four.	X’s	The computer student’s next move is to block the user from winning.

Result Summary

The test passed without any problems.

TEST11

Step #	Step Description	Input (data)	Results
1	After playing the first game with the computer student with the user winning. The user then lets the computer student move first. The user makes all the same moves that the computer student made before.	O's	The computer student should make the same moves as the user previously made and win the game.

Result Summary

This test is harder as the easiest way to confirm that the records are being saved is to check the database. However, this test passed as written.

TEST12

Step #	Step Description	Input (data)	Results
1	The user makes a first move that they have never made before while playing the new computer student.	X's	The computer student randomly picks its next move.

Result Summary

This test passed without any problems.

TEST13

Step #	Step Description	Input (data)	Results
1	From the Home page the user clicks on the Matches link.		The user is taken to the matches screen. The default is to have the computer student disabled for both Tic-Tac-Toe and Connect Four.
2	The user pushes the Enable button for Tic-Tac-Toe.		If there are other players' computer students available a

			list of them will become visible with a play button beside each one.
3	The user should now log out by pushing the Logout link.		The user is taken to the non-logged in Home page.
4	The user registers as a different a different user and/or logs in as a different user and navigates to the Matches page. If the current computer student should have enabled both Tic-Tac-Toe and Connect Four.		The user sees that the first user is available for Tic-Tac-Toe but not Connect Four.

Result Summary

This test passed without any problems.

TEST14

Step #	Step Description	Input (data)	Results
1	The user pushes the Play button beside on of the other users on the Matches page.		The match between the two computer students is played and the user is taken to another page to view the results of the match.

Result Summary

This test passed without any problems.

TEST15

Step #	Step Description	Input (data)	Results
1	The user goes to the Matches page.		Only users that have enabled their student to play are visible.

Result Summary

This test passed without any problems.

TEST16

Step #	Step Description	Input (data)	Results
1	The user's computer student plays the other computer student ten or more times in a row.		Sometime the user's computer student should go first other times the other user's computer student should go first.

Result Summary

This test requires random chooses by the application. However, it passed with no problems.

TEST17

Step #	Step Description	Input (data)	Results
1	Each time the user plays their computer student a record is created. This can be seen by making the some moves in the same order as a losing game so that the computer student can be the winner.		The moves match the winning games' proving that there is one and only one computer student per-user.

Result Summary

This is a bit harder to test, however it passed without any problems.

TEST18

Step #	Step Description	Input (data)	Results

1	The user logs in.		At the Home page after logging in, the Tic-Tac-Toe link and the Connect Four link should always be available.

Result Summary

This test passed without any problems.

TEST19

Step #	Step Description	Input (data)	Results
1	Each time the user plays their computer student a record is created. This can be seen by making the some moves in the same order as a losing game so that the computer student can be the winner.		The moves so match the winning games' proving that there is one and only one computer student per-user.

Result Summary

This test is harder to confirm from the application level and is easier to confirm by looking at the database. However, the test passed without a problem.

TEST20

Step #	Step Description	Input (data)	Results
1	User goes to the Tic-Tac-Toe or Connect Four page.		There are always two buttons. One which allows the user to go first and one that allows the computer student to go first.

Result Summary

This test passed without any problems.

TEST21

Step #	Step Description	Input (data)	Results
1	From the Tic-Tac-Toe screen, the user pushes the “New Game Start First” button.		Every time the user will be X.

Result Summary

This test passed without any problems.

TEST22

Step #	Step Description	Input (data)	Results
1	From the Tic-Tac-Toe screen, the user pushes the “New Game Start Second” button.		Every time the user will be O.

Result Summary

This test passed without any problems.

TEST23

Step #	Step Description	Input (data)	Results
1	The user plays the computer student many times.		Each time, the computer student never makes an invalid move.

Result Summary

This test is harder than most as it requires the tester to play many games in order to look for invalid moves. However, the test passed without any problems.

TEST24

Step #	Step Description	Input (data)	Results
1	The user finishes a game by either winning, losing, or tie.		Clicking on the board no longer has any effect.

Result Summary

This test passed without any problems.

TEST25

Step #	Step Description	Input (data)	Results
1	From the Connect Four screen, the user pushes the "New Game Start First" button.		Every time the user will be black.

Result Summary

This test passed without any problems.

TEST26

Step #	Step Description	Input (data)	Results
1	From the Connect Four screen, the user pushes the "New Game Start Second" button.		Every time the user will be red.

Result Summary

This test passed without any problems.

TEST27

Step #	Step Description	Input (data)	Results
1	The user plays the computer student many times.		Each time, the computer student never makes an invalid move.

Result Summary

This test is harder than most as it requires the tester to play many games in order to look for invalid moves. However, the test passed without any problems.

TEST28

Step #	Step Description	Input (data)	Results
1	The user finishes a game by either winning, losing, or tie.		Clicking on the board no longer has any effect.

Result Summary

This test passed without any problems.

16.0 Test Plan Summary

Connect-Games is for the most part an easy application to test as the amount of free text and moves are limited. The hardest part is testing that the application stores the sequence of a game after a player wins and is actually using these sequences when a game is played. However, by tracking the games that a computer student is involved in, this is possible. During test, the application passed all tests without any problems.

17.0 References

[1] Tic-tac-toe. Retrieved on March 17, 2018, https://en.wikipedia.org/wiki/Tic-tac-toe

[2] Connect Four. Retrieved on March 17, 2018, https://en.wikipedia.org/wiki/Connect_Four

[3] Martin, R. C. (2017). Clean architecture: a craftsman's guide to software structure and design. Prentice Hall Press.

[4] Strategy. Retrieved on March 31, 2018, http://www.oodesign.com/strategy-pattern.html

[5] Abstract Factory. Retrieved on March 31, 2018, http://www.oodesign.com/abstract-factory-pattern.html

[6] 3 Key Software Principles You Must Understand. Retrieved on March 31, 2018, https://code.tutsplus.com/tutorials/3-key-software-principles-you-must-understand--net-25161

[7] Model–view–controller. Retrieved on March 31, 2018, https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[8] Ajax (programming). Retrieved on April 7, 2018, https://en.wikipedia.org/wiki/Ajax_(programming)

[9] Chapter 3: Architectural Patterns and Styles. Retrieved on April 2nd, 2018, https://msdn.microsoft.com/en-us/library/ee658117.aspx

[10] Domain Model. Retrieved on April 7th, 2018, https://en.wikipedia.org/wiki/Domain_model

Git Repository

https://github.com/shawnsalm/Connect-Games/tree/master