Web Checkers

OO Design Specification Document

The Avengers

Rochester Institute of Technology

Golisano College of Computing and Information Sciences

Date: 04-11-2019

Version #1: WebCheckers-DSD-30092019

Version History

VERSION	AUTHOR(S)	CHANGE DESCRIPTION	DATE	NOTES
1	Leonardo A. Deema S. Pratyush K. Paola P. Rana A.	Initial DSD.	04/11/2019	

Table of Content

1 Overview

Web Checkers is a web-based application that allow players to play checkers. Players could play with other online players registered on the system or choose to play with the computer. Web Checkers user interface is using drag-and-drop browser capabilities for making the moves. This application is based on American rules.

1.1 Goals and Objectives

Goal:

The goal of the project is to develop an overall intuition for software development through learning various modelling techniques and foundation concepts that capture business requirements put forward by the client and transform these requirements into a workable software product. Our goal is to not only implement the requirements into an application but mainly understand how different design views and structures come into play when working on a software project. In this way, we can build software in a much more efficient and intelligent way.

Objectives:

1. To understand the domain concepts of the Web Checkers game that need to be studied by domain modelling.
2. To understand different software architecture views such as the static view through modelling Class Diagram, the interaction view between Class Objects through the Design Sequence Diagrams and Business Requirements view using Use-Case Modelling.
3. To develop the game in a way that it is most efficiently deployable and portable. We also study its architecture from the view of deployment.
4. To capture the data view using the Entity-Relation Model that would enable us to determine how the data entities are related to each other in our problem domain of Web Checkers. This also helps us to understand the database fundamental associated with the project.
5. To implement all the business requirement of the client and ensure that the end user is fully satisfied with the software product.

1.2 Software Context

This product is will operate in the eGaming context allowing different users to share the gaming experience through this web application.

1.3 Terminology & Definitions

- **Player:** Is any user that can play checkers in the Web Checkers application.
- **Tournament organizer:** It’s the person that creates the tournament. A tournament organizer is not allowed to play in the tournaments created by itself.

2 Functional Descriptions

A brief description of the major functionalities supported by the application is presented [Reference the SRS & Use-case document for details].

FR 1. The player should be able to register in WebCheckers system and create an account to be able to access games and other system functionalities.

- FR 1.1. Player should provide a valid registration information. be able to set the registrations information like desired username, password, valid email for the application.
  - FR 1.1.1. Password should have a minimum length of 8.
  - FR 1.1.2. Password should be alphanumerical and contain at least one of the three Uppercase(A-Z), lowercase(a-z) and number (0-9).
  - FR 1.1.3. Username should be unique and has not been registered for another player.
  - FR 1.1.4. The player should provide a valid and unique email address for verification purposes.

FR 2. Player should be able to login into the WebCheckers application and the system should be able to match the login credentials and authenticate the player into the system.

FR 3. The Player should be able to play a Checkers match against another registered Player or computer.

- FR 3.1. Each Player will have a piece color in a match.
- FR 3.2. Each Player has 60 seconds to make a move.
- FR 3.3. The system will consider a player as a loser if no move conducted in 60 seconds.
- FR 3.4. The system will show warning message when a player made illegal move and will not approve that move.
- FR 3.5. The system will show a message when a game is completed and accordingly declare the winner.
- FR 3.6. The player can resign from playing a game.
  - FR 3.6.1. If a Player resigns the other Player is declared winner and the match ends.
- FR 3.7. A player may be able to play more than one game at a time.

FR 4. The player can choose difficulty level of a game to be "easy" or "hard" for both "human-to-human" and "human-to-computer" games.

- FR 4.1. If the easy level is selected the match should normal based on basic American Rules.
- FR 4.2. If the hard level is selected the match will not allow multiple jumps.

FR 5. A player can create a gaming tournament and is named as tournament organizer.

- FR 5.1. Tournament organizer cannot join this tournament.
- FR 5.2. All players should be humans.
- FR 5.3. Tournament organizer should determine the time and date of the tournament.
- FR 5.4. Tournament organizer should determine the maximum and minimum number of players.
  - FR 5.4.1. Range of players should be even number.
  - FR 5.4.2. If the number of players joining the tournament is odd, the last joining player should be omitted.
  - FR 5.4.3. The system will cancel the tournament and display “Not enough players” if the minimum number of players has not been reached.
- FR 5.5. The system randomly selects players pairs from the list of players joined the tournament.
- FR 5.6. The system should display the list of players before starting the tournament.

- FR 5.7. The system will consider a player as a loser if no move conducted in 60 seconds or not being available.
- FR 5.8. The system should identify winning players to compete for the next consecutive phase.
- FR 5.9. The system randomly selects players pairs for the next phase from the winning players of the previous phase.
- FR 5.10. The system should declare the winner by the end of the tournament.

FR 6. A player can join a tournament.

FR 7. Players should be able to view a replay of previously played matches.

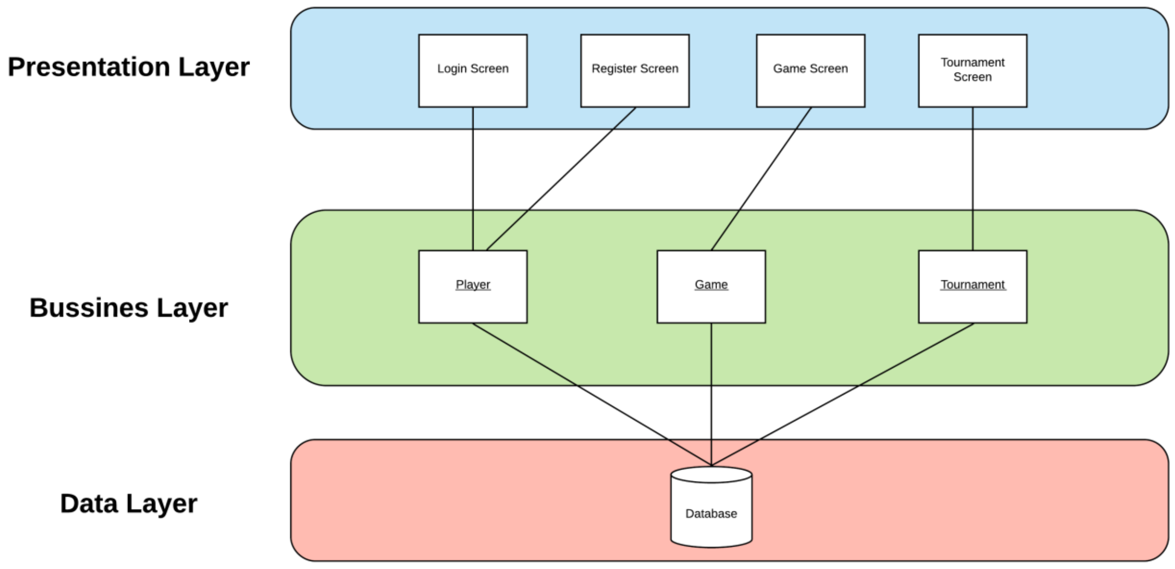
- FR 7.1. Matches should be stored for later review.

FR 8. Players may be able to live view an ongoing game that they are not playing.

3 Non-Functional Descriptions

- **Response Time:** The application needs to allow the flow of information in real time, in order to bring the best gaming experience to the user.
- **Security:** The personal information of the users should be treated as a sensitive element and keep It secure.
- **Efficient resource utilization:** The application should maintain a stable resource consumption.
- **Reliability:** The system should guarantee the availability of the application to maintain the QoS.

4 System Architecture



4.1 Node Architecture

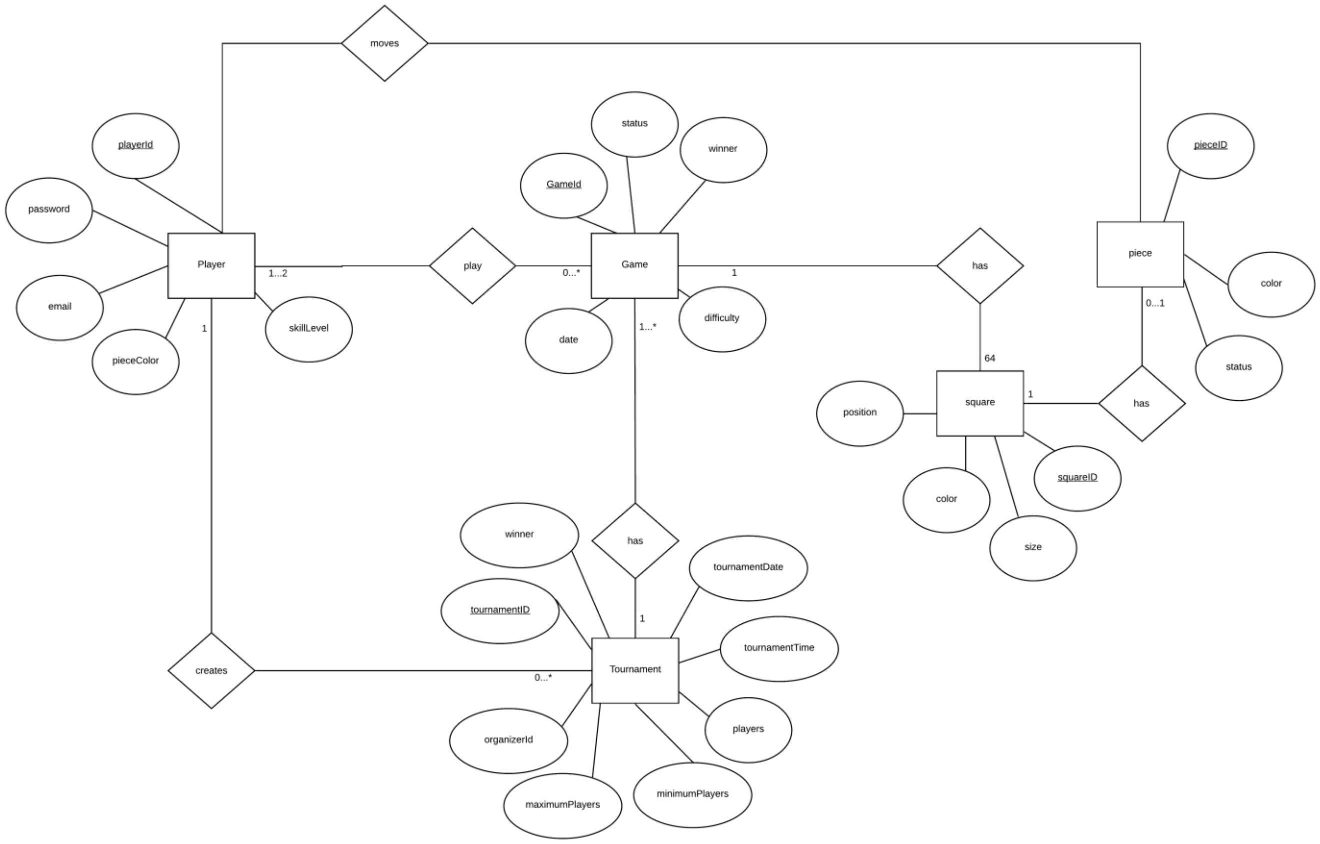
N/A.

4.2 Component Architecture

N/A.

4.3 Data Architecture

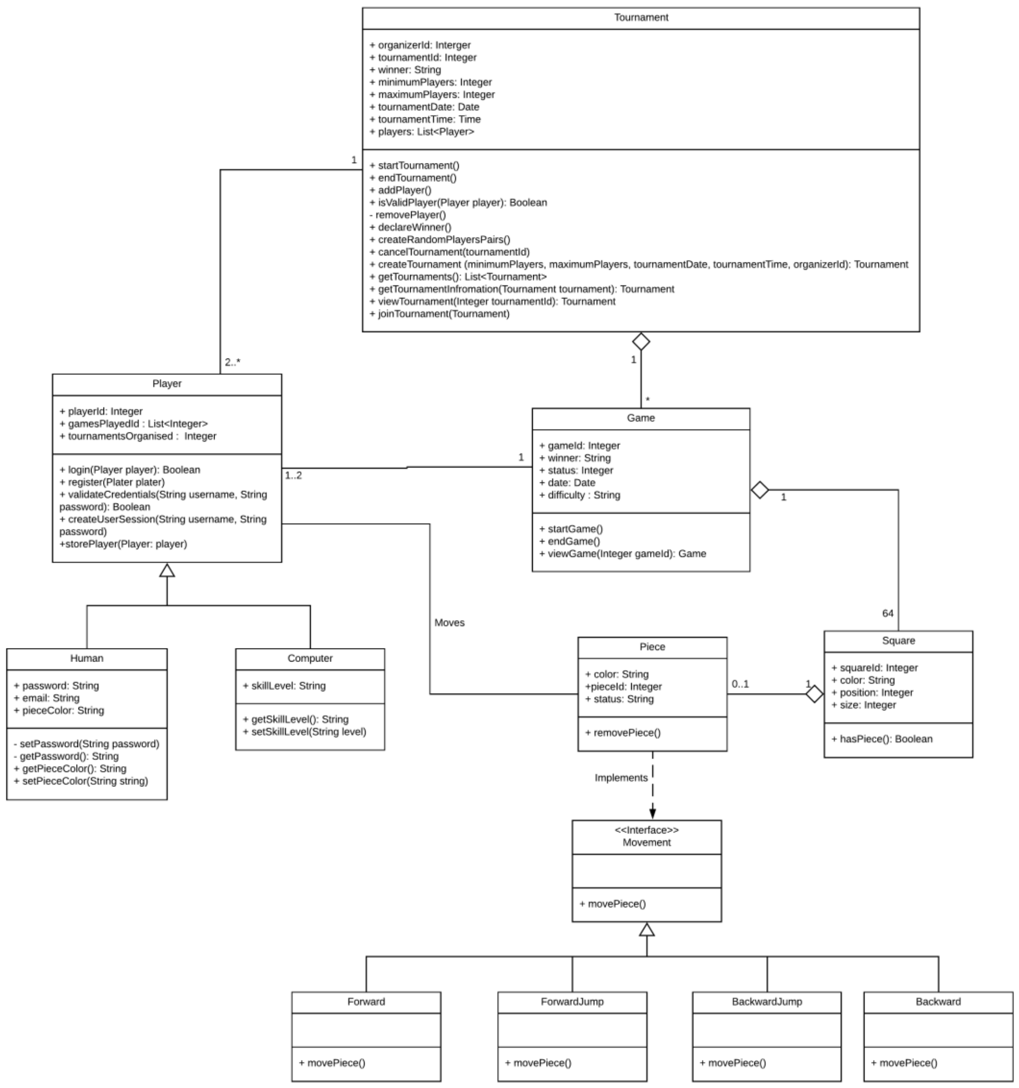
N/A.



5 Detailed Architecture

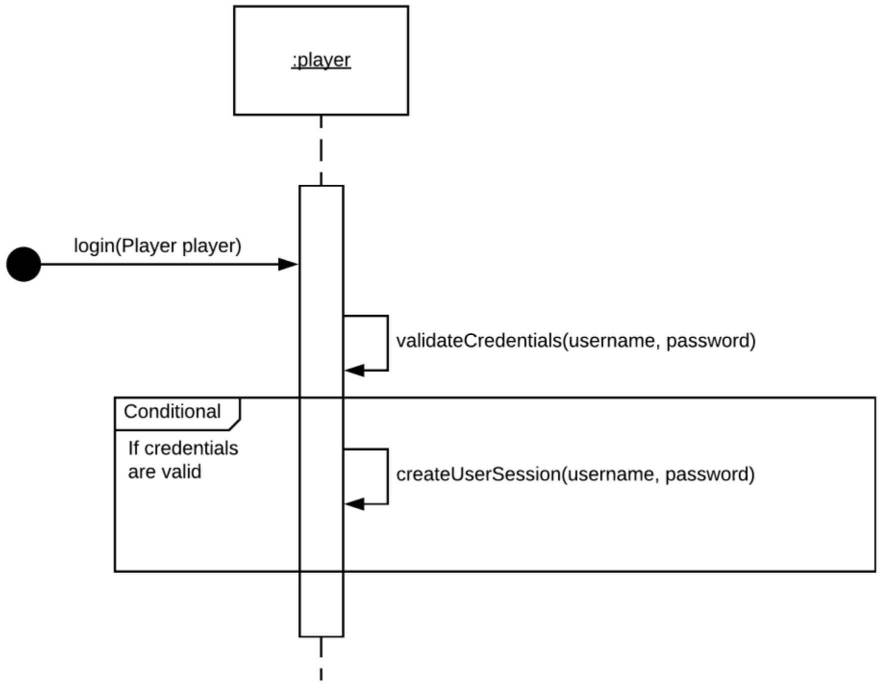
5.1 Module # 1

5.1.1 Class Diagram

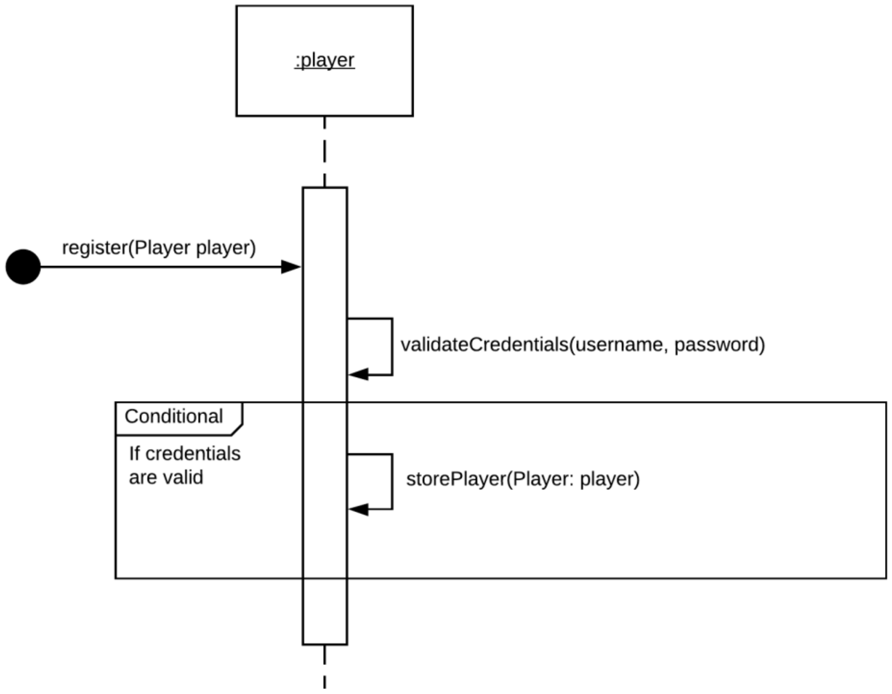

Tournament
+ organizerId: Interger
+ tournamentId: Integer
+ winner: String
+ minimumPlayers: Integer
+ maximumPlayers: Integer
+ tournamentDate: Date
+ tournamentTime: Time
+ players: List<Player>
+ startTournament()
+ endTournament()
+ addPlayer()
+ isValidPlayer(Player player): Boolean
- removePlayer()
+ declareWinner()
+ createRandomPlayersPairs()
+ cancelTournament(tournamentId)
+ createTournament (minimumPlayers, maximumPlayers, tournamentDate, tournamentTime, organizerId): Tournament
+ getTournaments(): List<Tournament>
+ getTournamentInfromation(Tournament tournament): Tournament
+ viewTournament(Integer tournamentId): Tournament
+ joinTournament(Tournament)
1
2..*
1
*
Player
+ playerId: Integer
+ gamesPlayedId : List<Integer>
+ tournamentsOrganised : Integer
+ login(Player player): Boolean
+ register(Plater plater)
+ validateCredentials(String username, String password): Boolean
+ createUserSession(String username, String password)
+storePlayer(Player: player)
1..2
1
Game
+ gameId: Integer
+ winner: String
+ status: Integer
+ date: Date
+ difficulty : String
+ startGame()
+ endGame()
+ viewGame(Integer gameId): Game
1
64
Moves
Human
+ password: String
+ email: String
+ pieceColor: String
- setPassword(String password)
- getPassword(): String
+ getPieceColor(): String
+ setPieceColor(String string)
Computer
+ skillLevel: String
+ getSkillLevel(): String
+ setSkillLevel(String level)
Piece
+ color: String
+pieceId: Integer
+ status: String
+ removePiece()
0..1
1
Square
+ squareId: Integer
+ color: String
+ position: Integer
+ size: Integer
+ hasPiece(): Boolean
Implements
<<Interface>>
Movement
+ movePiece()
Forward
+ movePiece()
ForwardJump
+ movePiece()
BackwardJump
+ movePiece()
Backward
+ movePiece()

5.1.2 Interaction Diagram

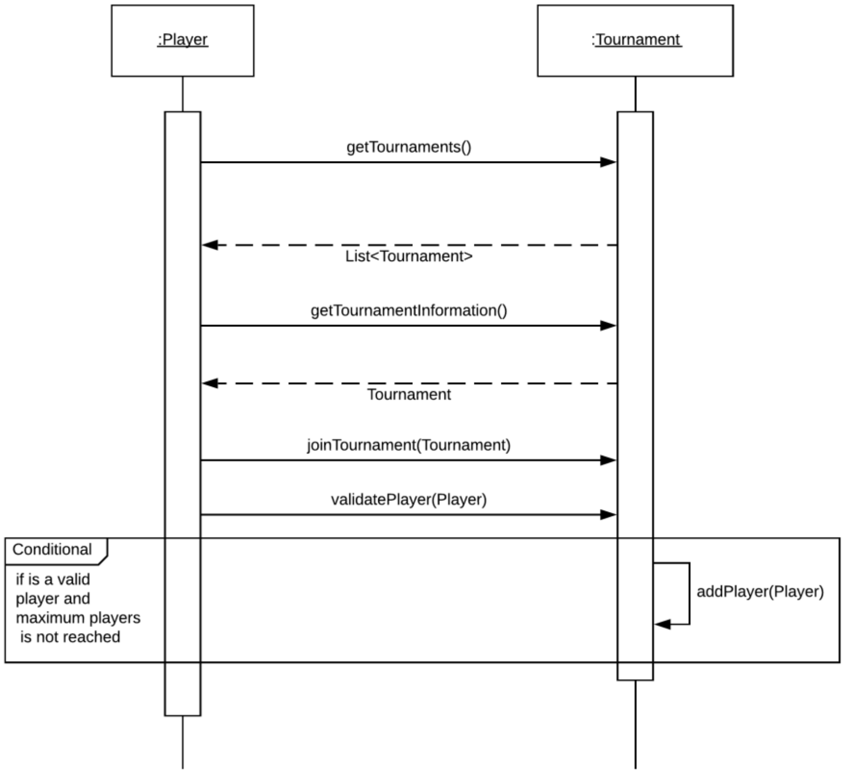
5.1.2.1 Login


:player
login(Player player)
validateCredentials(username, password)
Conditional
If credentials are valid
createUserSession(username, password)

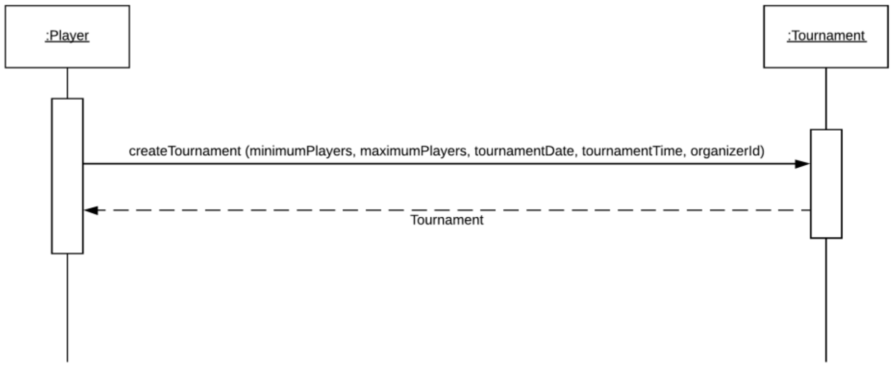
5.1.2.2 Register


:player
register(Player player)
validateCredentials(username, password)
Conditional
If credentials are valid
storePlayer(Player: player)

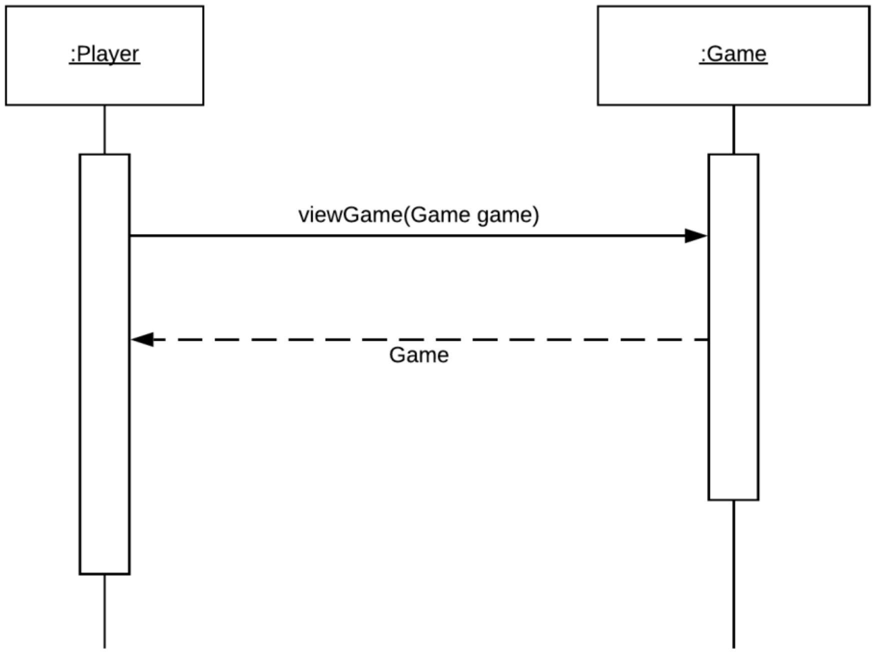
5.1.2.3 Join Tournament



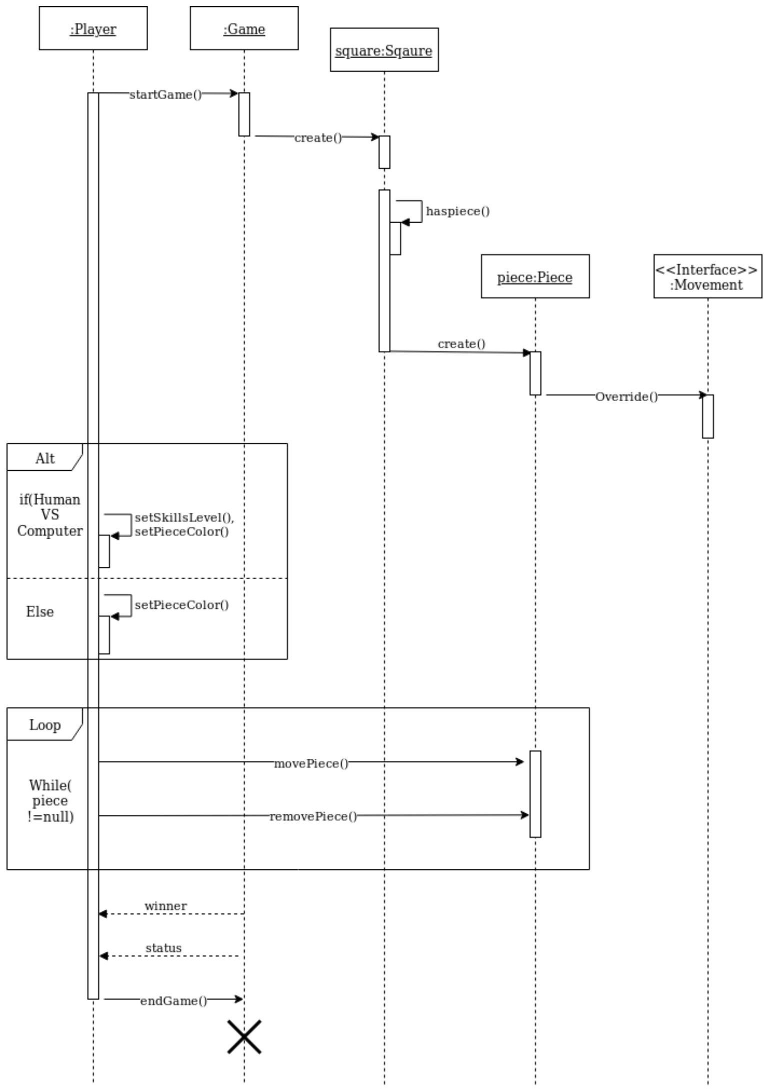
5.1.2.4 Create Tournament


:Player
:Tournament
createTournament (minimumPlayers, maximumPlayers, tournamentDate, tournamentTime, organizerId)
Tournament

5.1.2.5 View Game



5.1.2.6 Play Game

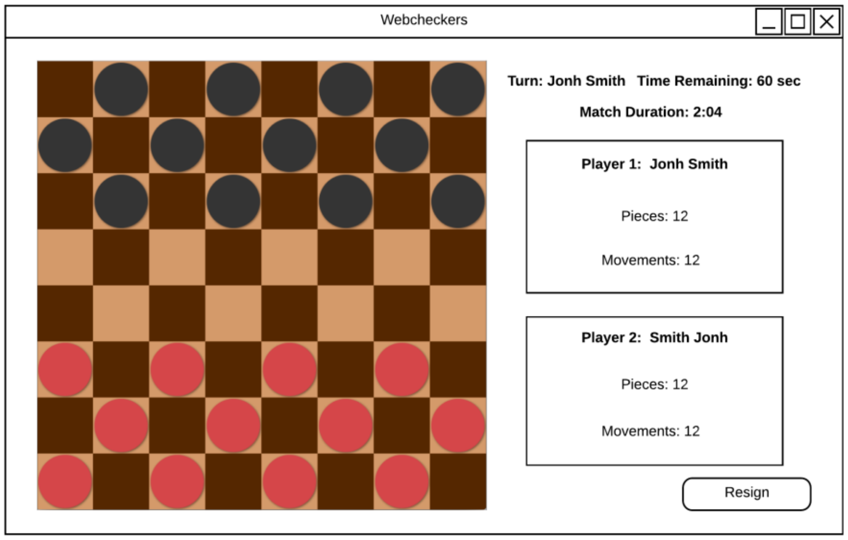


5.1.3 Detailed Design

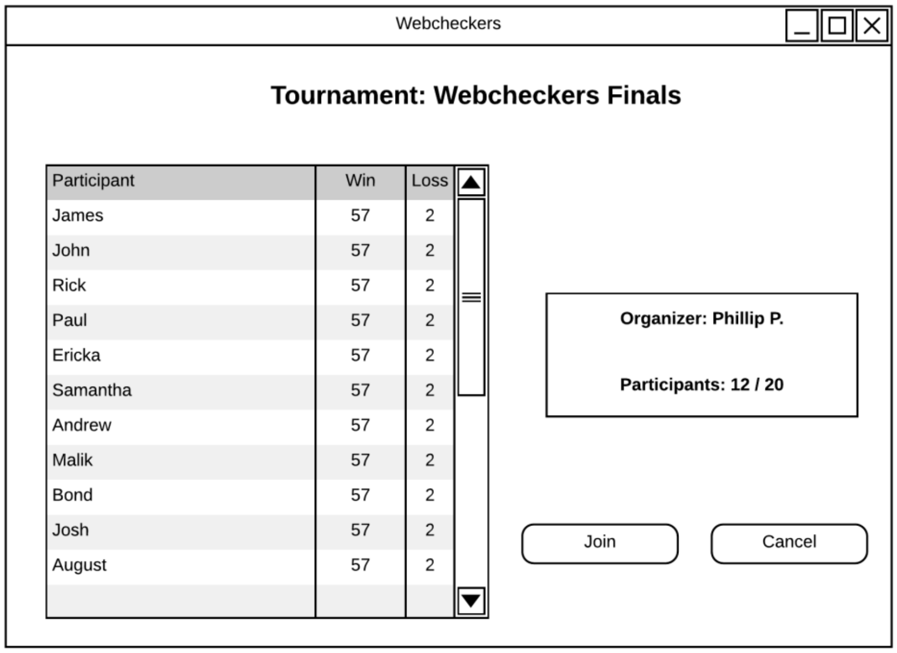
- **Player**: This class has all the information related to the user that will be playing in the system. This has two subclasses: Human and Computer.
- **Tournament**: This class contains all the information related to the tournaments that are available in the system.
- **Match**: This class captures all the information related to a match between two players or a player and the computer.
- **Piece**: This class the information of the pieces that of the checkers game. It implements and interface with the Movement class to get the piece movement type.
- **Square**: This class contains the information of the squares that contain the pieces in the board.

5.2 Human interface

5.2.1 User interface design



Play screen: This interface allows the users to play a web checkers match. It shows relevant information like the players information, match duration and actual turn.



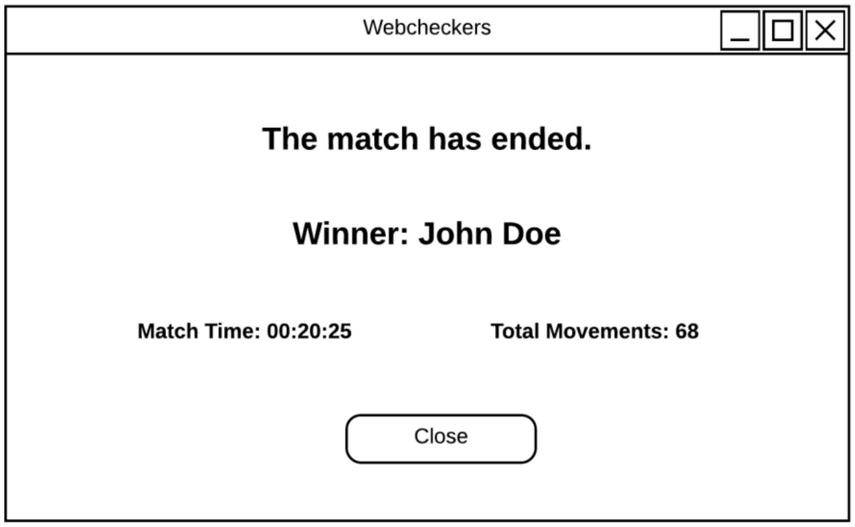
Tournament: Webcheckers Finals

Participant	Win	Loss
James	57	2
John	57	2
Rick	57	2
Paul	57	2
Ericka	57	2
Samantha	57	2
Andrew	57	2
Malik	57	2
Bond	57	2
Josh	57	2
August	57	2

Organizer: Phillip P.

Participants: 12 / 20

Tournament details screen: This interface shows the information of a specific tournament. It contains information like the tournament name, player list and the option to join it.



Match Winner Modal: This element will be present when a match finish. It displays the winner name, match time and total movements.

6 Implementation Issues & Challenges

- **Technology knowledge:** It's a challenge because most of the team haven't worked with the technologies required by the project's stakeholders.
- **Piece movement:** It is a challenge to move a piece in the board from the front end and have that movement check if it is an allowed move and integrate that part of the front end of the application with the back end.
- **View a Match:** Start a game and have multiple players as viewers, were all the movements of the play needs to be broadcasted in real time.
- **Store and display a game history:** Keep track of every move of a game from start to end and being able to reproduce it for a player.

7 Supporting Systems Requirements

7.1 Design Hardware Requirements

In order to properly develop this application, the team needs:

Computer: The team need computers in order to develop this application. This application is meant not meant to work on mobile devices like cellphones or tablets.

Internet Access: The computer needs to have internet access in order to use the application.

Web Browser: The computer needs a browser to access the application.

Mouse or trackpad: Since the interaction in the game are based on drag and drop functionality, this is a crucial requirement for the development and test process.

RAM: 1GB minimum, 2 GB recommended.

Storage: 1 GB (IDE + Project files)

7.2 Design Software Requirements

Operating System (OS)

- Windows
- MacOS

IDE

- **IntelliJ IDEA by JetBrains (https://www.jetbrains.com/idea/):** IDE used by the team to develop the application.

Libraries

- **Spark (http://sparkjava.com/):** Web framework used to create Java Based Web Server.
- **Free Marker (https://freemarker.apache.org/):** Template engine library used to display the elements to the end user.

8 Conclusions & Future Extensions

In this document were described the details related to the development of an object-oriented web checkers application. Most of the elements were presented using unified modelling language (UML) to allow the easy understanding of the information flow in the application.

This document contains all the necessary information to start the development process and should serve as the main guide to answer any question related to the development details.

9 Related Material

- **Checkers:** is a group of strategy board games for two players which involve diagonal moves of uniform game pieces and mandatory captures by jumping over opponent pieces. Ref: https://en.wikipedia.org/wiki/Draughts
- **American rules:** Is a set of rules that defines a specific way of playing the Checkers game. Ref: http://www.se.rit.edu/~swen-261/projects/WebCheckers/American%20Rules.html
- **Drag and drop:** It's a pointing device gesture that allow the movement of a virtual object to another position. Ref: https://en.wikipedia.org/wiki/Drag_and_drop