28-3-2018

CS372 Project

The Chess Game

Melanie MacDonald

Iyanu Akinyemi

Jose Lagrosa

Ulises Oviedo

Ashir Judah

Ryan Leadbetter

Contenido

[1. Problem definition 3](#_Toc510027957)

[2. Software requirements specification document: 4](#_Toc510027958)

[A. For each type of users, the use case diagram with use case and actors. 4](#_Toc510027959)

[B. Define in detail three use cases (the most complex ones). 6](#_Toc510027960)

[C. Software qualities (correctness, efficiency, robustness and user friendliness 7](#_Toc510027961)

[3. Design specification document 8](#_Toc510027962)

[A. Software architecture (multi-layer). 8](#_Toc510027963)

[B. Three sequence diagrams (for the three defined use cases). 9](#_Toc510027964)

[C. Class diagrams with all the classes and their relationships. 12](#_Toc510027965)

[PvP Class diagram 12](#_Toc510027966)

[PvsCPU class diagram 13](#_Toc510027967)

[D. Two examples of object diagrams. 14](#_Toc510027968)

[PvP Object Diagram 14](#_Toc510027969)

[PvsCPU Object Diagram 15](#_Toc510027970)

[4. Code description, including: 16](#_Toc510027971)

[A. Component diagram regarding the organization of the source code. 16](#_Toc510027972)

[Zoom to the Component Diagram 17](#_Toc510027973)

[B. Deployment diagram regarding the hardware configuration of the software system. 22](#_Toc510027974)

[C. List the classes and functions that you have implemented. 23](#_Toc510027975)

[D. Include the link of your Web-based application. 24](#_Toc510027976)

[5. Technical documentation, including: 25](#_Toc510027977)

[A. UML supporting tool. 25](#_Toc510027978)

[B. Programming languages. 26](#_Toc510027979)

[C. Reused algorithms and programs. Include their links/sources. 27](#_Toc510027980)

[D. Software tools and environments. Indicate which software parts are using which tools. 28](#_Toc510027981)

[E. Database management system. Provide a screenshot of the table contents. 29](#_Toc510027982)

[6. Code testing, including: 30](#_Toc510027983)

[A. Correctness testing with five test cases. For each test case, submit the screenshots of the inputs and outputs. 30](#_Toc510027984)

[User sign up and login Test case 31](#_Toc510027985)

[Real time chess 35](#_Toc510027986)

[Player plays against the CPU 37](#_Toc510027987)

[Leaderboard correctly updates with each win 38](#_Toc510027988)

[B. Robustness testing with five test cases. For each test case, submit the screenshots of the inputs and outputs. 40](#_Toc510027989)

[Form input validation 41](#_Toc510027990)

[Reject incorrect moves 44](#_Toc510027991)

[Multiple moves in one turn 47](#_Toc510027992)

[Player cannot manipulate opponent’s chess pieces 50](#_Toc510027993)

[Player cannot manipulate opponent’s chess pieces 52](#_Toc510027994)

[C. Performance testing of five functions. 55](#_Toc510027995)

[7. A document that clearly indicates the completed tasks of each member. 56](#_Toc510027996)

1. Problem definition.

Computers have significantly increased access to multiple forms of entertainment, one such form being video games. We are aiming to create a game that is geared towards those that do not have much time in their day to play games, or who are looking for a brief distraction. The game will be short but addicting so that it has replay value without boring the user. This game allows the player to start a play session that lasts for the span of 5 minutes, be able to save their game, and upload their high score.

# 2. Software requirements specification document:

Functional Requirements for Player

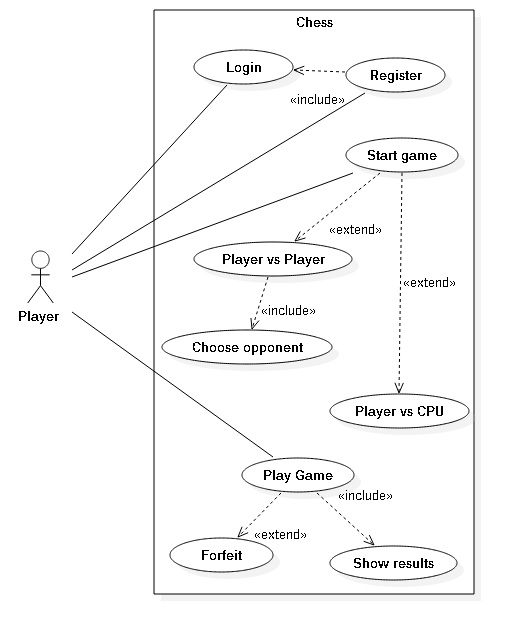
- Start Game: The user can begin a game against a computer.

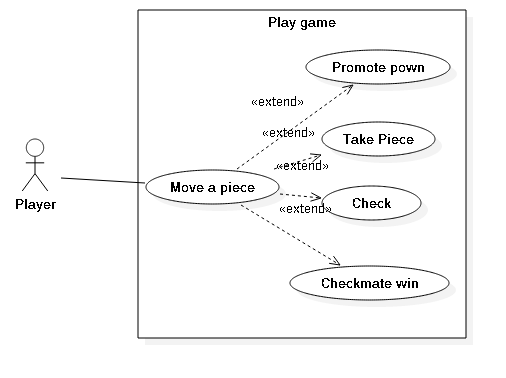
- Display Statistics: Look at the statistics

- Host Game: The user can begin a game against another player.

- Join Game: The user can join a game session created by another player.

## For each type of users, the use case diagram with use case and actors.





## Define in detail three use cases (the most complex ones).

## Software qualities (correctness, efficiency, robustness and user friendliness

Correctness – The application should be able to successfully log the user into the application. If the user does not have an account, the software should be able to successfully create an account and add it to the database. The program should be able to send a user to the type of game they have chosen. The code should be able to find other players as well as be able to create a session between two players. The application must follow the rules of a regular chess game from movement of the individual pieces to the advanced techniques such as castling and promotion. The program must be able to switch turns after a player is done with their turn. The application should also recognize when a match has ended and be able to push the player back to the home screen. The game should be able to recognize when a match has been won/tied. The game should also allow the user to forfeit a match. The application should correctly increment a player’s win score/draw score in the leader board. The software should not allow user to access anything without signing in. The program should be able to correctly log the user out of a session.

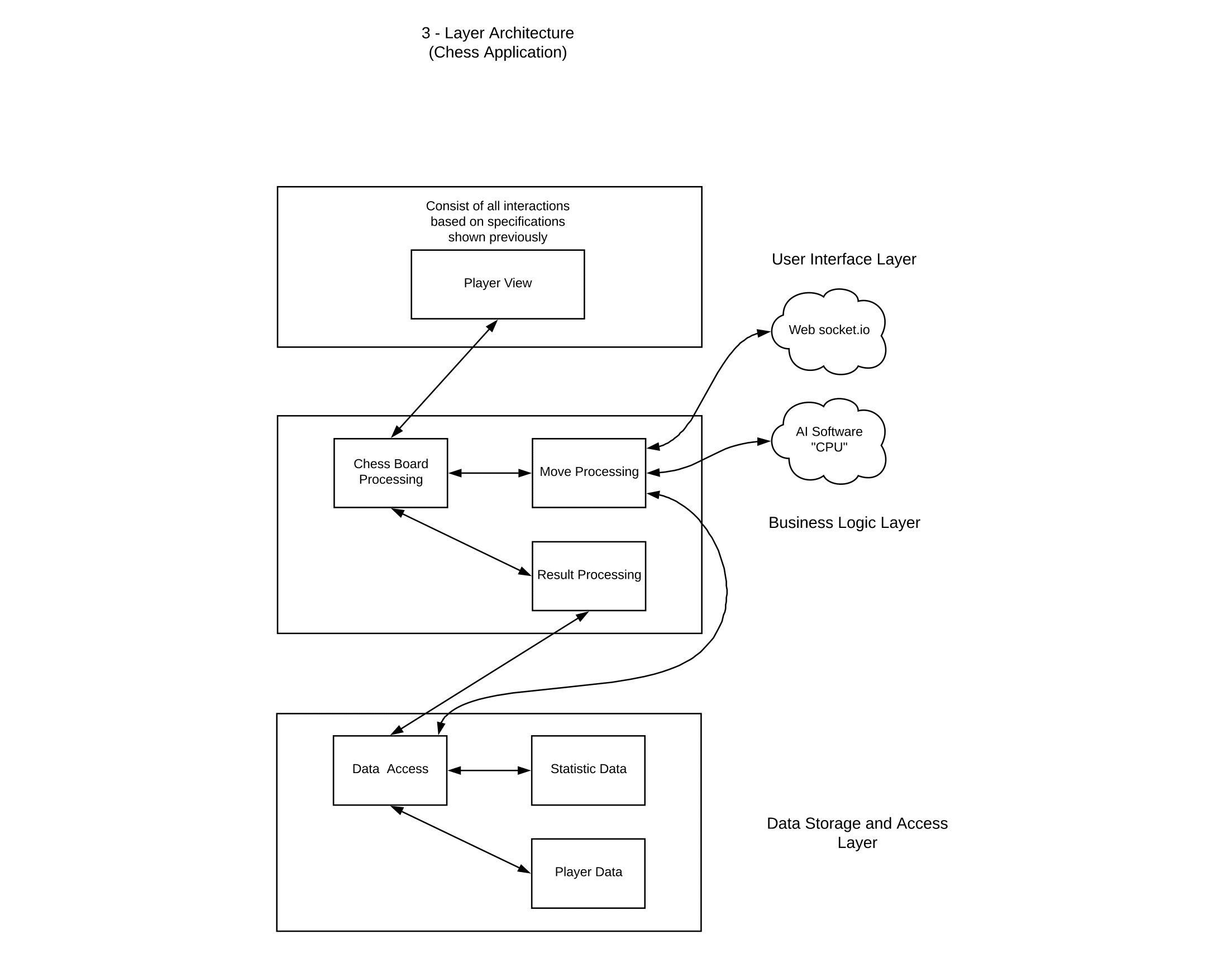
Robustness – The application should be able to handle incorrect moves and clearly show legal moves a player can make. The program should be able to handle incorrect data being entered into the login and sign up pages. The application should not allow a user to move the other player’s pieces. The system should not be able to enter another player’s active game. The player cannot forfeit a game without it being recorded as a loss. The user should not be able to affect their rankings on the leaderboards. The user should not be able to access games without having logged in.

User Friendliness – A simple menu based design will be used to allow users to quickly and efficiently move about the various game types. Finding games will be clear and concise. The available players for online games should displayed in a simple way in order for player’s to find specific opponents. The leaderboards should be informative and allow users to quickly find their rankings amongst other players. A helpful link to the rules of chess should be placed for players who may not know them. Signing in and logging should not be confuse or dissuade the user from making/signing-into their account

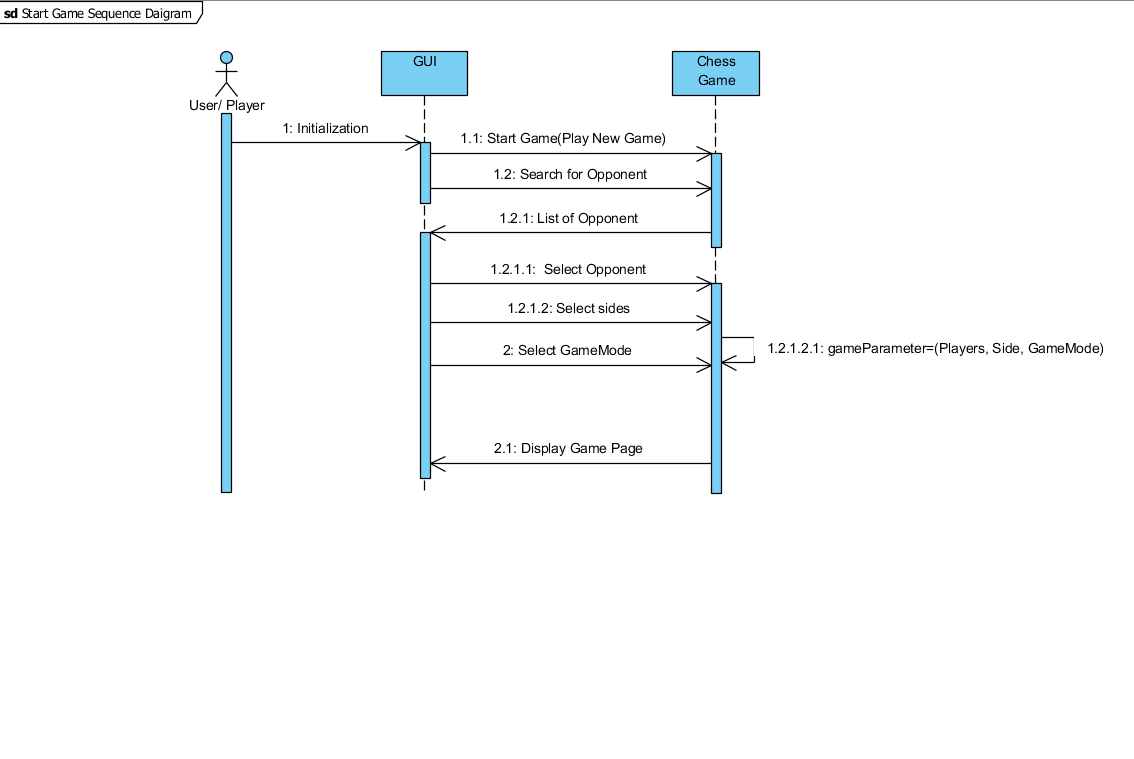
Performance and Time-Efficiency - The built-in commands should function at the right time to allow user to exert minimal effort to quickly navigate the system and eventually play a game. The AI should be fast to allow the user to have a fun game and not cause frustration. The program should also evaluate a player’s moves quickly to not disrupt the flow of the game. The leaderboards should be able to immediately update a player’s score and ranking to allow for easy comparisons.

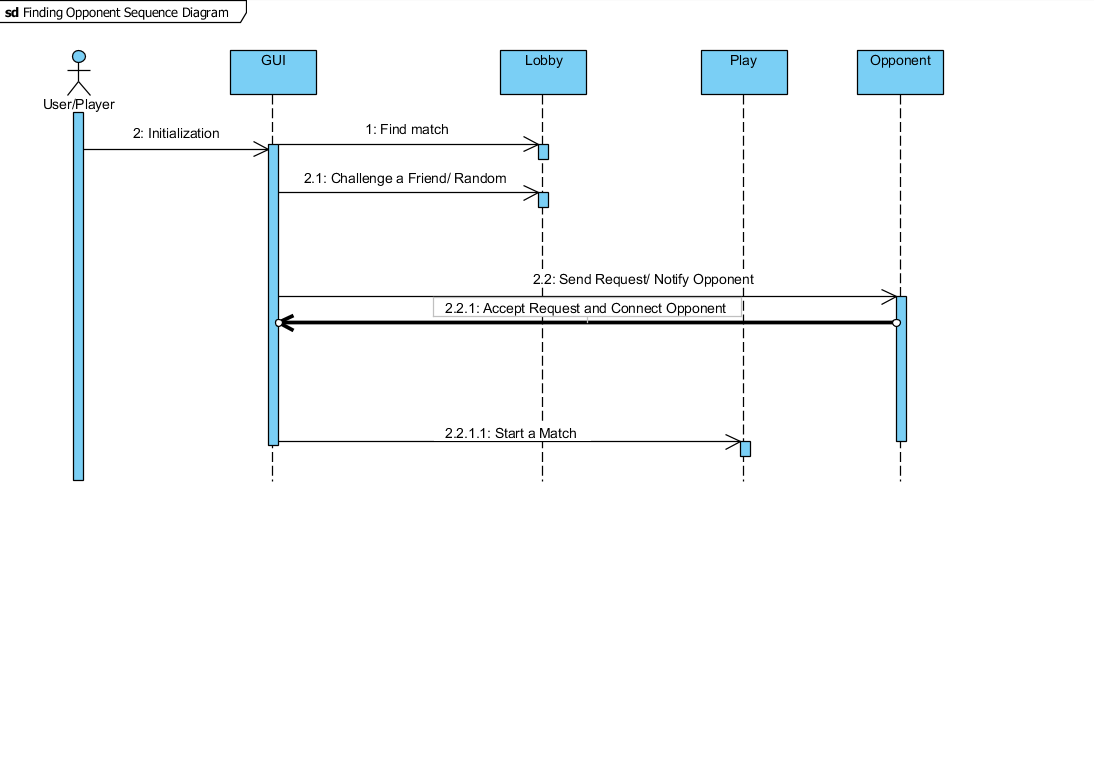
# 3. Design specification document

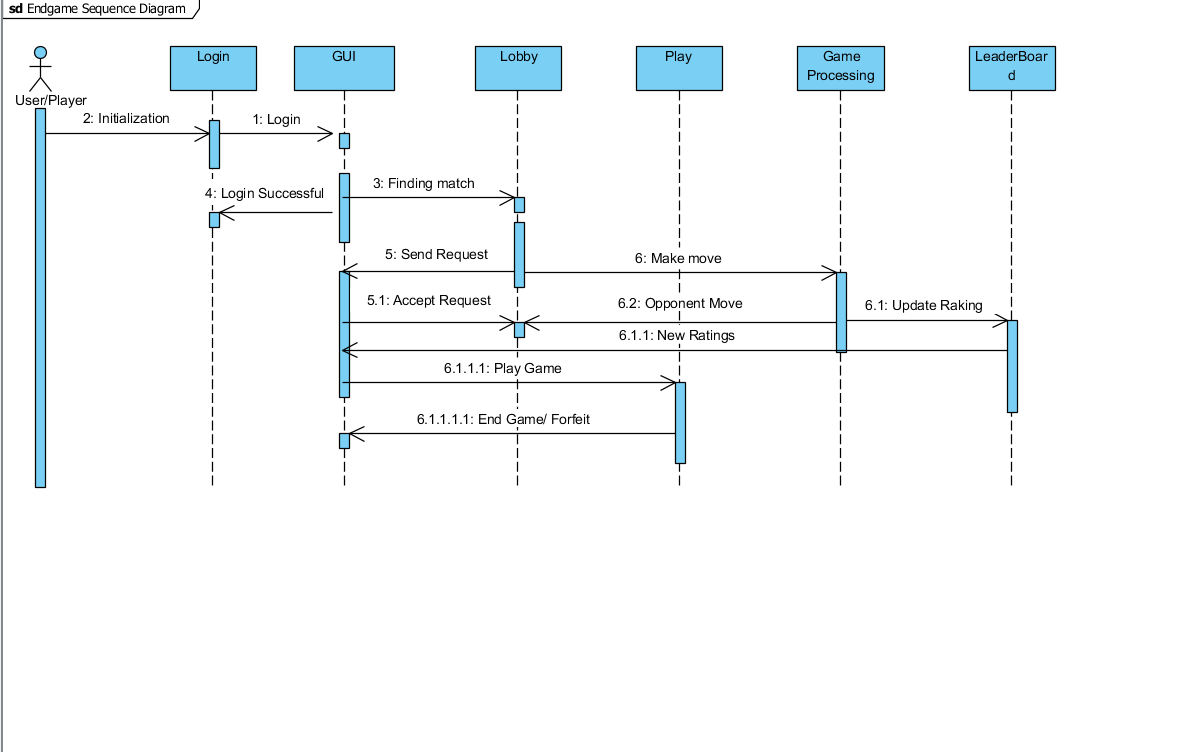
## Software architecture (multi-layer).



## Three sequence diagrams (for the three defined use cases).

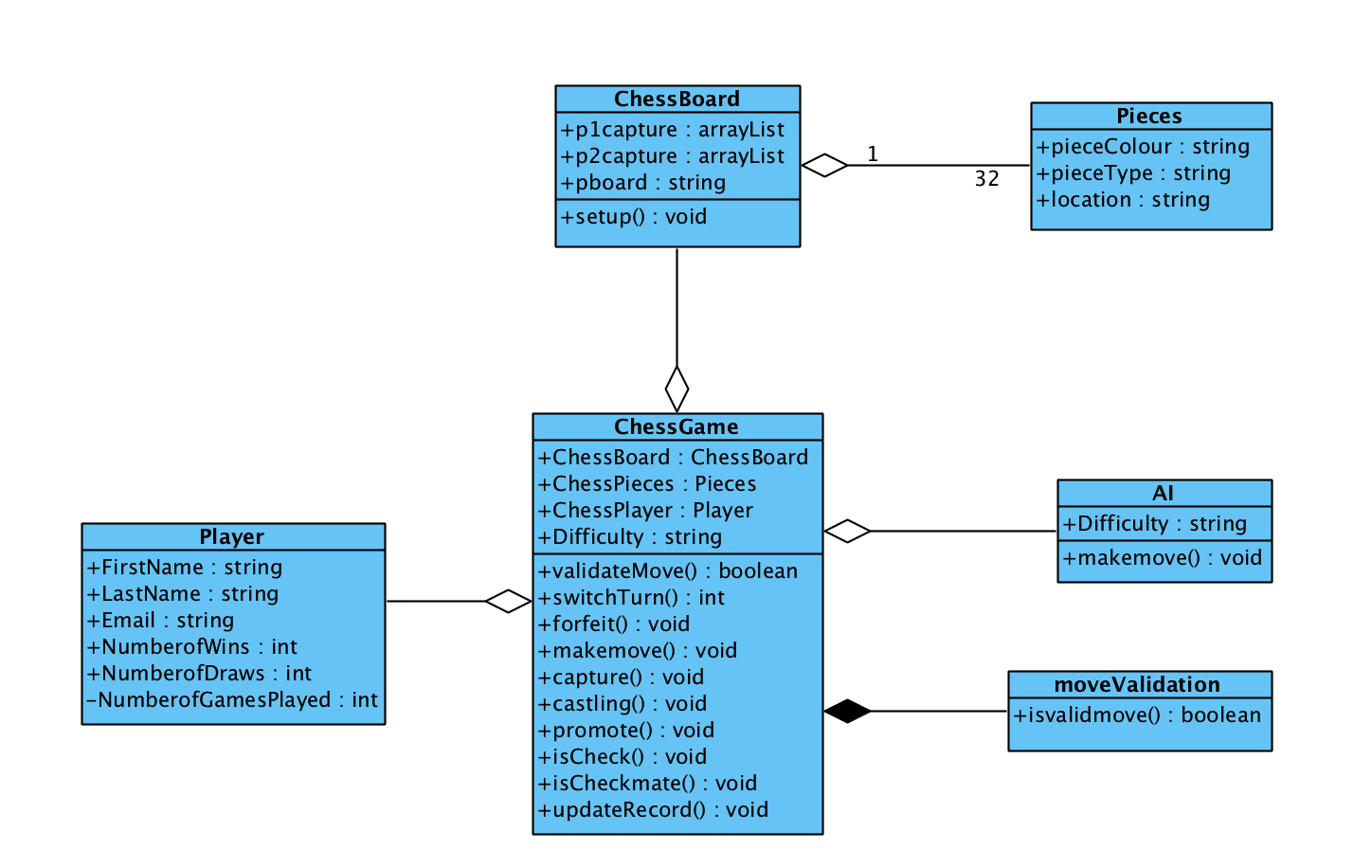




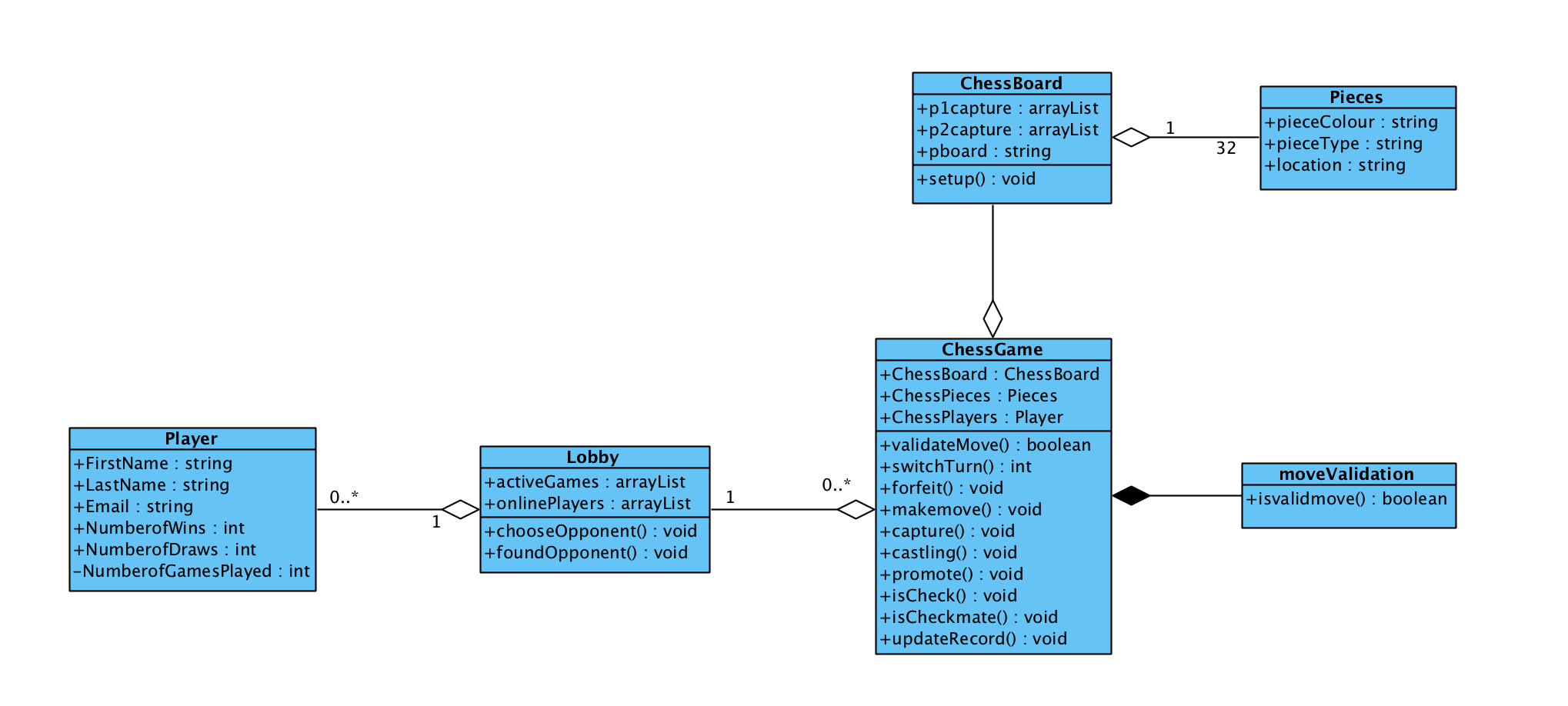


## Class diagrams with all the classes and their relationships.

### PvP Class diagram

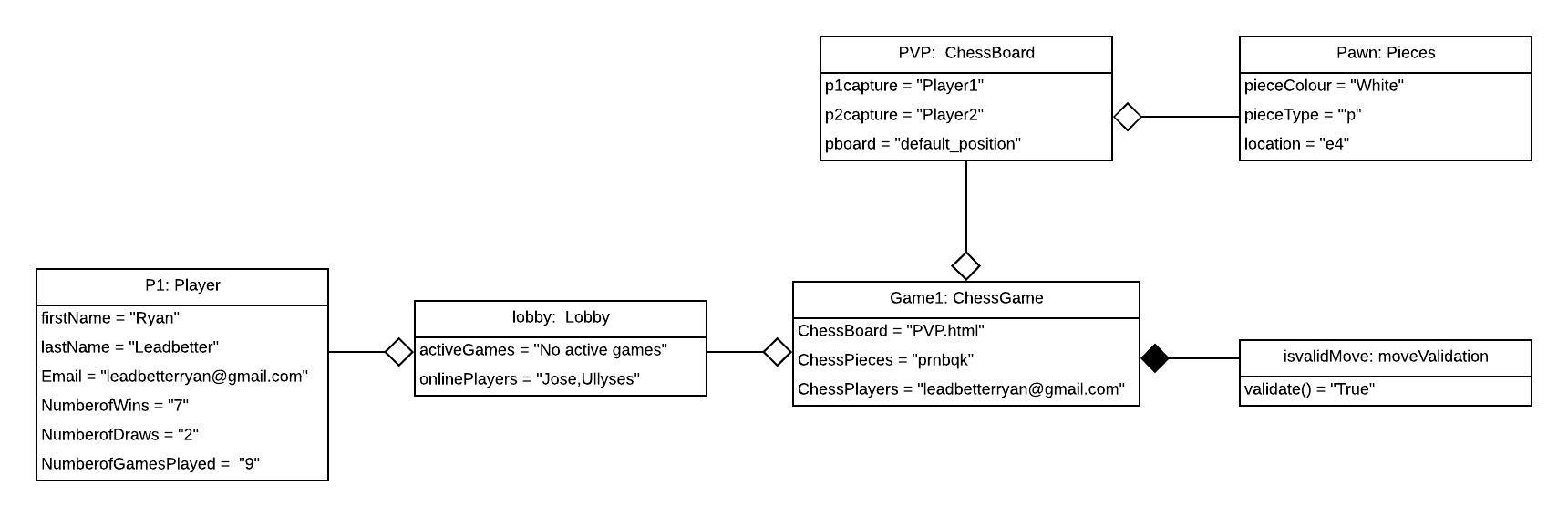


### PvsCPU class diagram

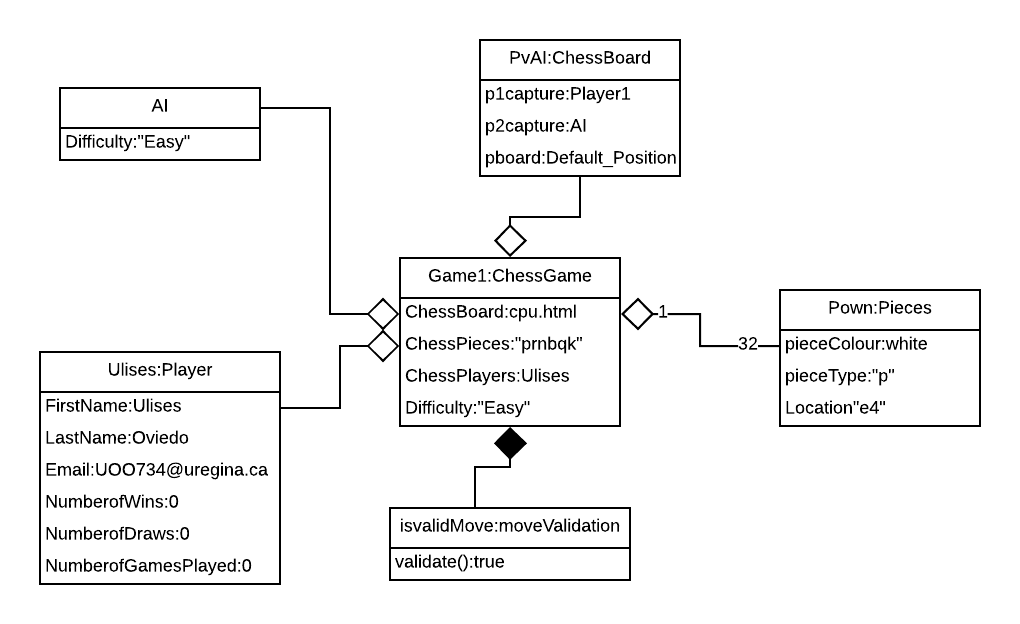


## Two examples of object diagrams.

### PvP Object Diagram



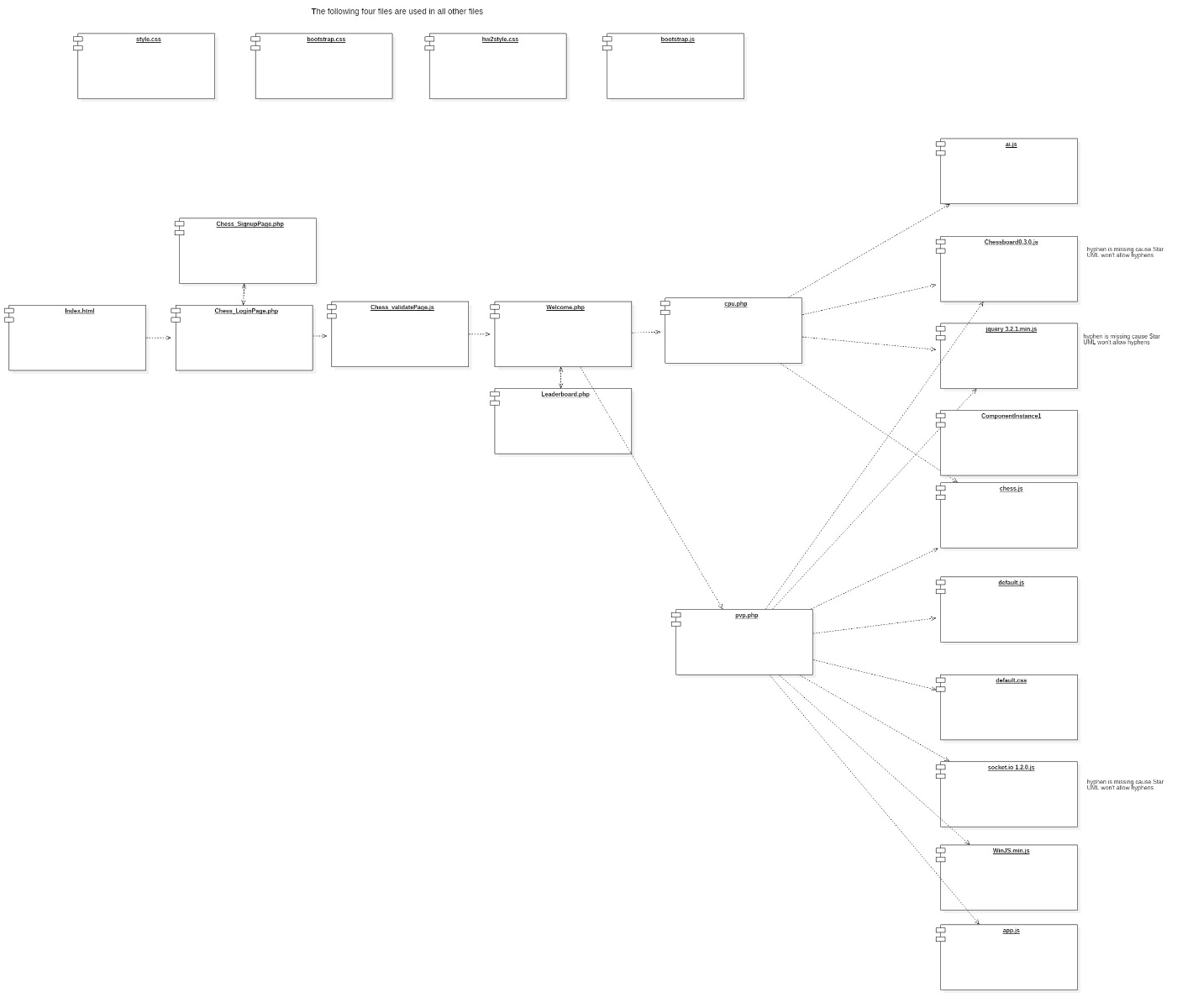
### PvsCPU Object Diagram



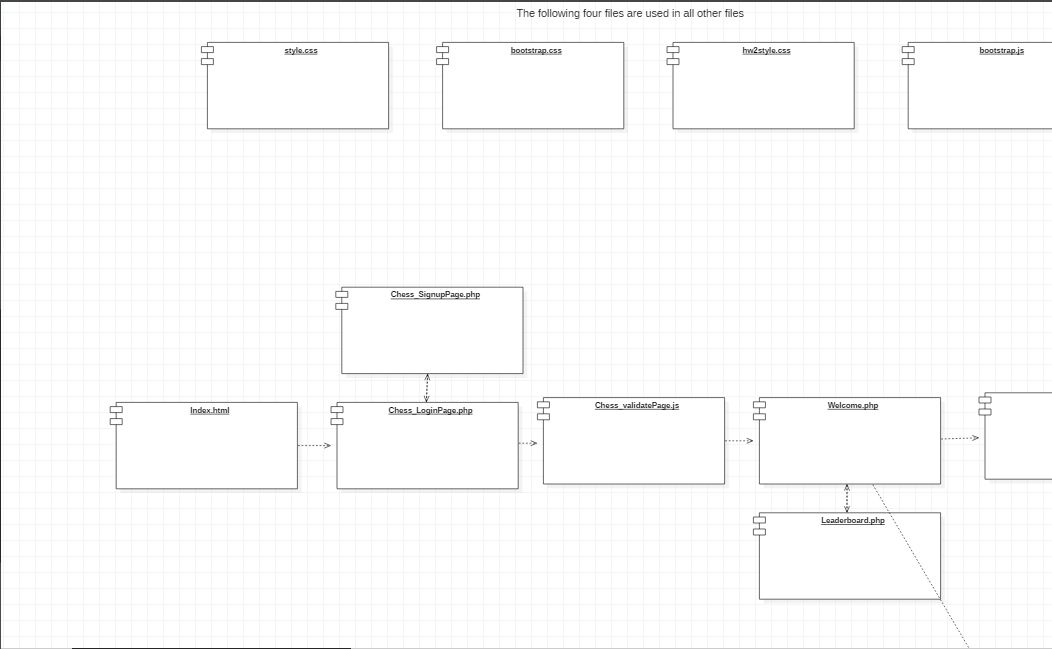
# 4. Code description, including:

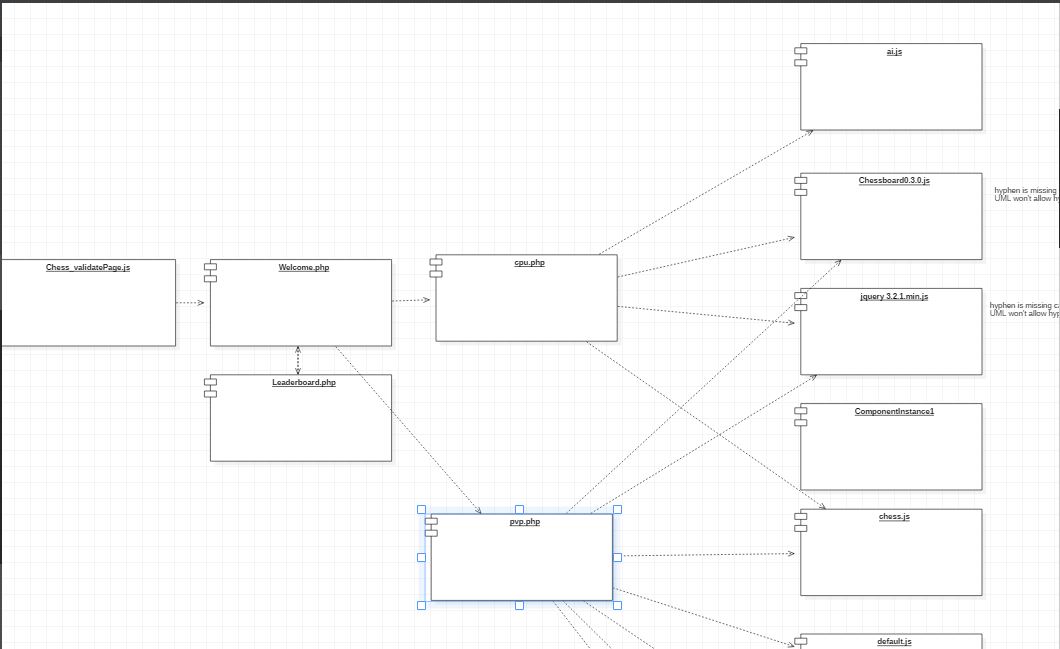
## Component diagram regarding the organization of the source code.

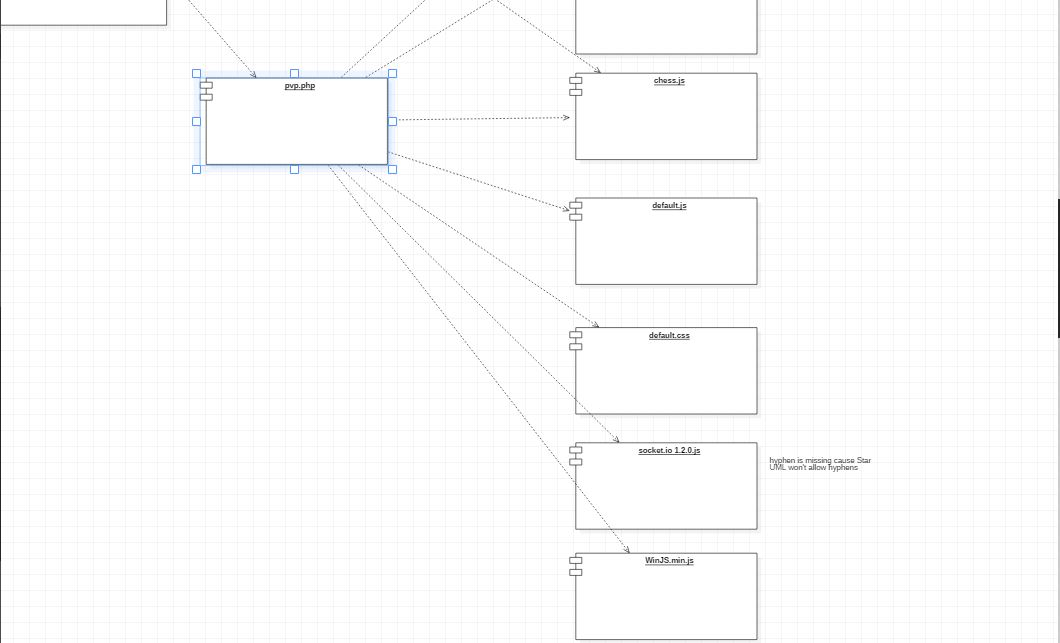
Complete Component diagram

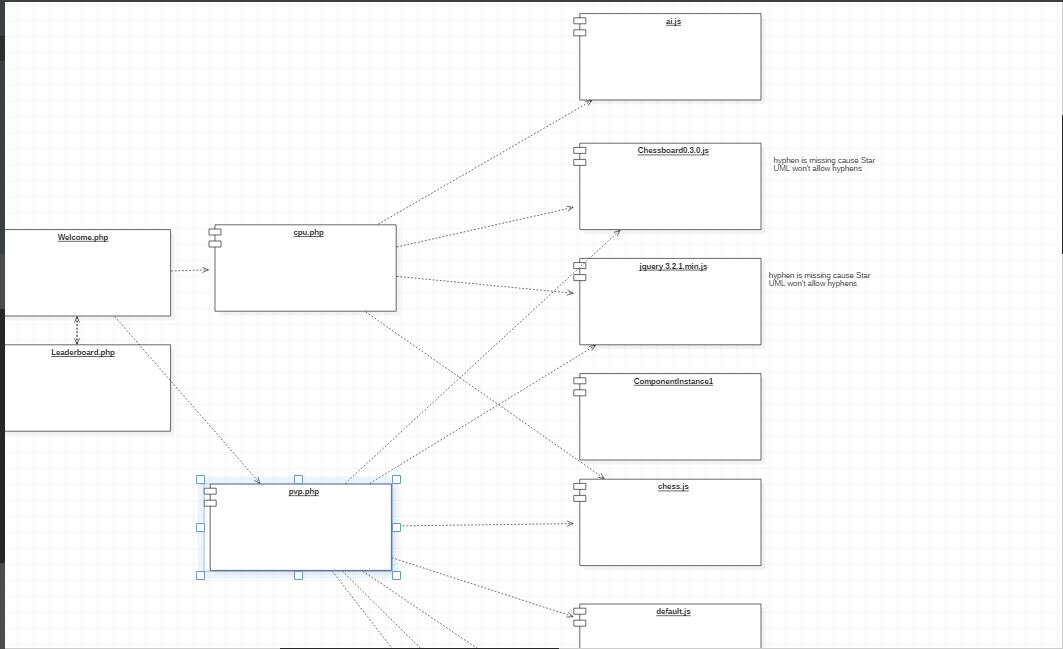


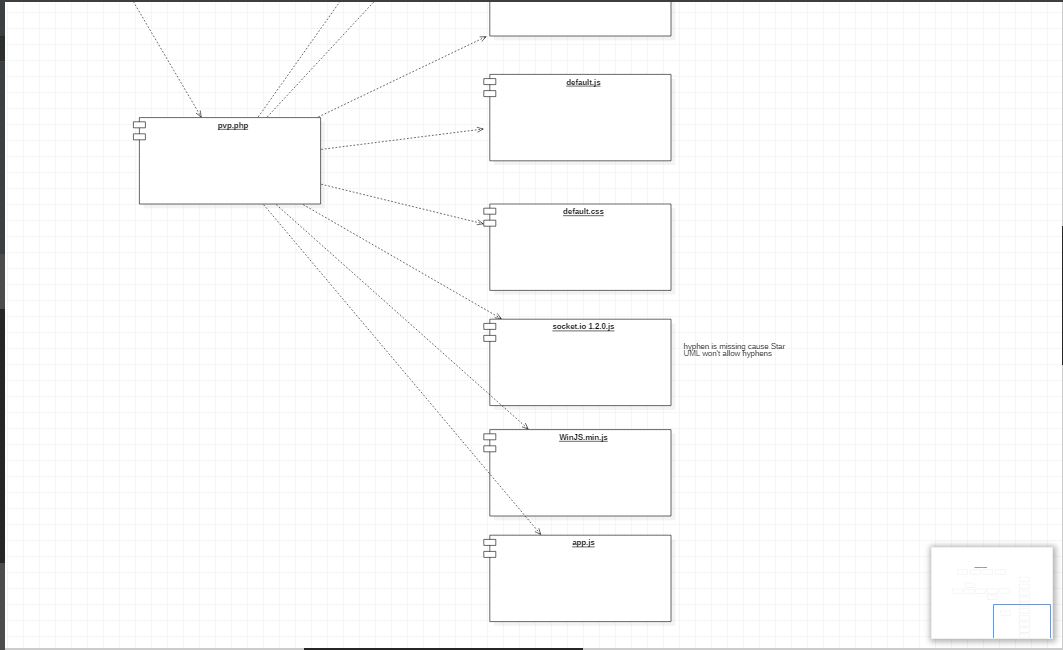
### Zoom to the Component Diagram



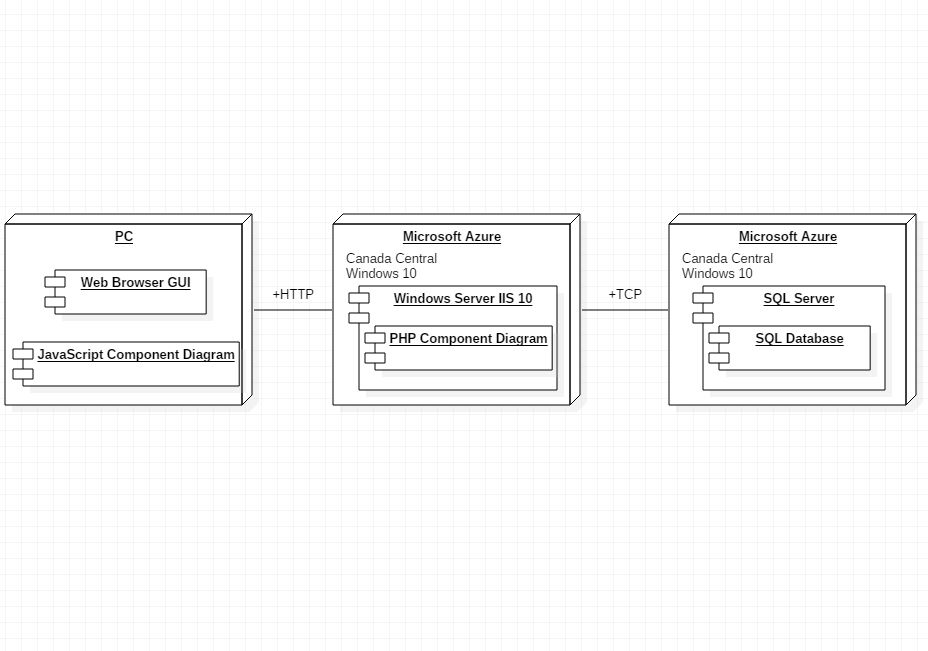








## Deployment diagram regarding the hardware configuration of the software system.



## List the classes and functions that you have implemented.

## Include the link of your Web-based application.

<http://chess372.azurewebsites.net/>

# 5. Technical documentation, including:

## UML supporting tool.

Lucidchart: This is used in designing the object diagram which expresses the object combinations of the class diagram.

Visual Paradigm: This is used to design the class diagram that describes the structure of the chess game system and the sequence diagram that shows the sequence of object interactions of the use cases.

StarUML: This is used for designing use cases diagram which illustrates the relationship between the use cases and component diagram describing the component within the chess game software system. It was also used to design the deployment the diagram which describes the hardware within the overall architecture of the system.

## Programming languages.

GUI: PHP, Html, and CSS was used to implement the graphical user interface of the chess game system.

Business layer: Javascript is used for the behavior of the chess Board and the pieces, Jquery it is used with socket.io and PHP for the connection to the database

Database: SQL was used to implement the database because the cloud server (Azure) database only use SQL language

## Reused algorithms and programs. Include their links/sources.

We reused codes and program from "David Washington" to create a real-time multiplayer chess game using socket.io. . <https://github.com/dwcares/realchess> David Washington doesn’t have any license.

Chess: This program was used in developing and implement the chess game from Jeff Hlywa (jhlywa@gmail.com) Copyright (c) 2017, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

"THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE".

ChessBoard: This program was used to implement the chessboard and the pieces from Chris Oakman copyright 2013.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

"THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE".

## Software tools and environments. Indicate which software parts are using which tools.

Node.js: This is an open-source library that uses an event-driven and non-blocking Input/Output model which makes it efficient and effective. The application used Javascript and asynchronous programming on the server.

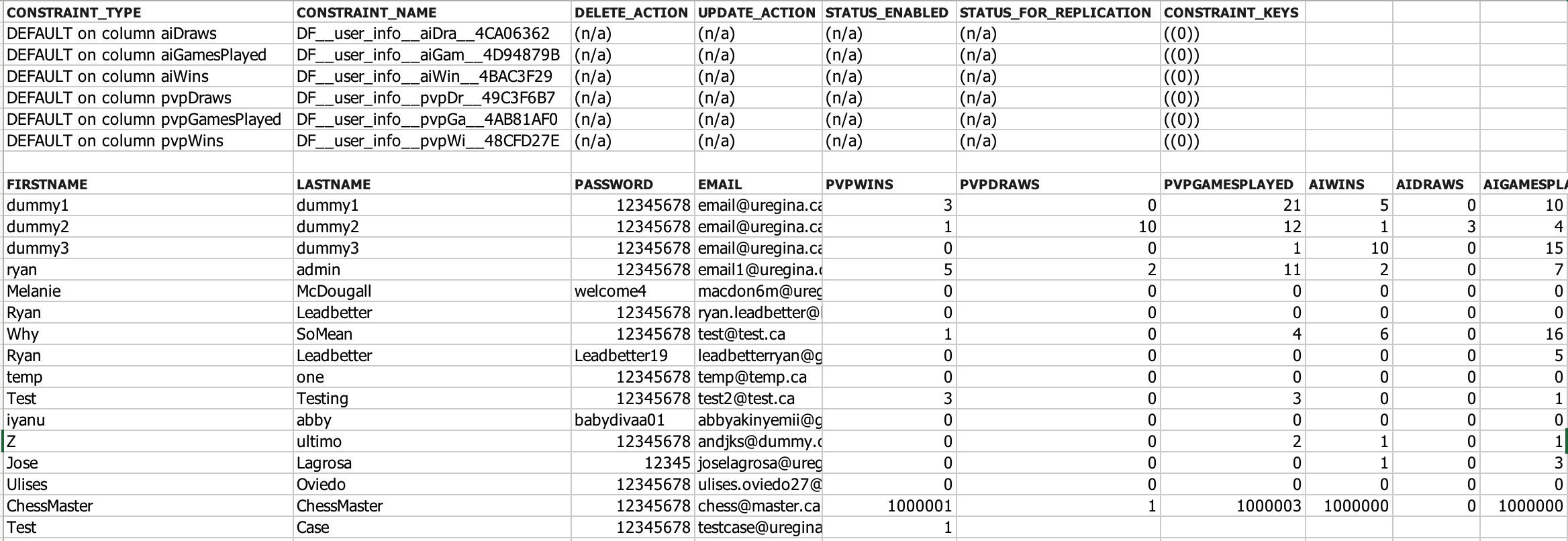
Visual Studio Code: It is used for editing codes and optimizing it for building and debugging the web-based application.

Bracket: It is a modern text editing application that was used for writing and editing codes which made designing easier and lightweight.

Microsoft Azure: This is a cloud computing used to build, test, deploy and manage applications. We published our server to Azure using GitHub. Azure acts like our localhost

Github: The application was used for communicating, discovering, sharing and building the web-based chess game

## Database management system. Provide a screenshot of the table contents.



The Data engine used for the game is SQL server. The data layer is hosted also in azure server.

The project only have a table where all user information is read and updated. So every time a user login to the application the email and password are compared to the ones in the table user\_information and when a player win against other player or the CPU(AI) the score is updated in the PVPWINS or AIWINS according to the opponent.

Azure SQL Database shares the SQL Server 2016 codebase.

# 6. Code testing, including:

## Correctness testing with five test cases. For each test case, submit the screenshots of the inputs and outputs.

The resume of test cases for correctness are the following with their respective results:

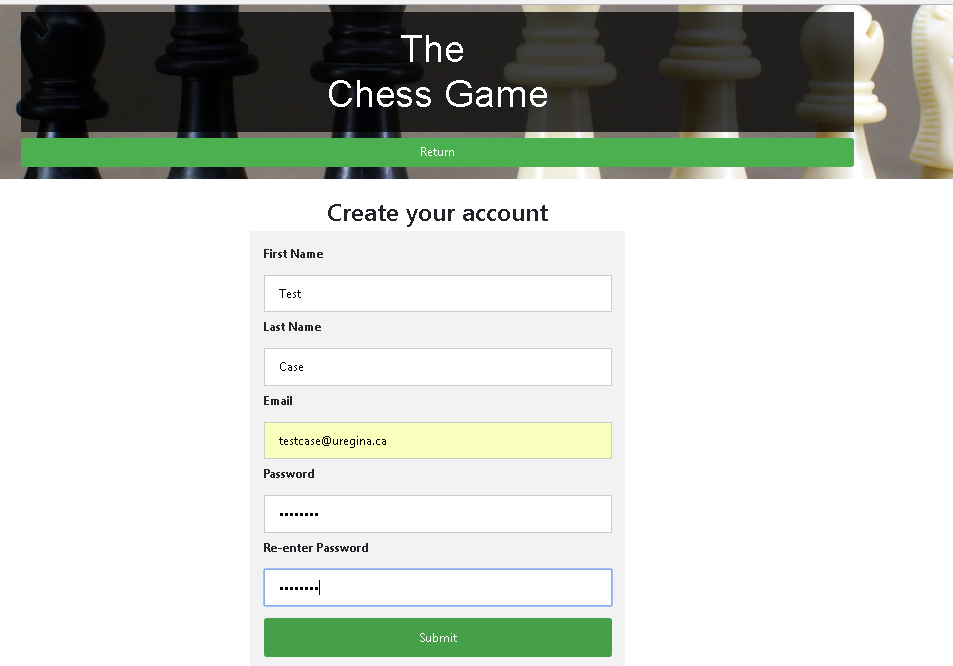
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Control number** | **Action** | **objective or Explanation** | **Expected results** | **Pass** | **Fail** |
| 1 | User sign up and login | The user must be able to create an account and be able to enter the game with the same account created | Account correctly created and used to access the game | X |  |
| 2 | Player enter an online room and play against another player | The player joins online games against other players | online players playing with each other | X |  |
| 3 | Real time chess | The movement of the game should be able to be viewed in real time in online games | chess online with real-time movements | X |  |
| 4 | Player plays against the CPU | The player can play against a computer | Computer plays against the player according to the difficulty | X |  |
| 5 | Leaderboard correctly updates with each win | According to the player's wins, the leaderboard is updated | Leaderboard correctly update | X |  |

## User sign up and login Test case

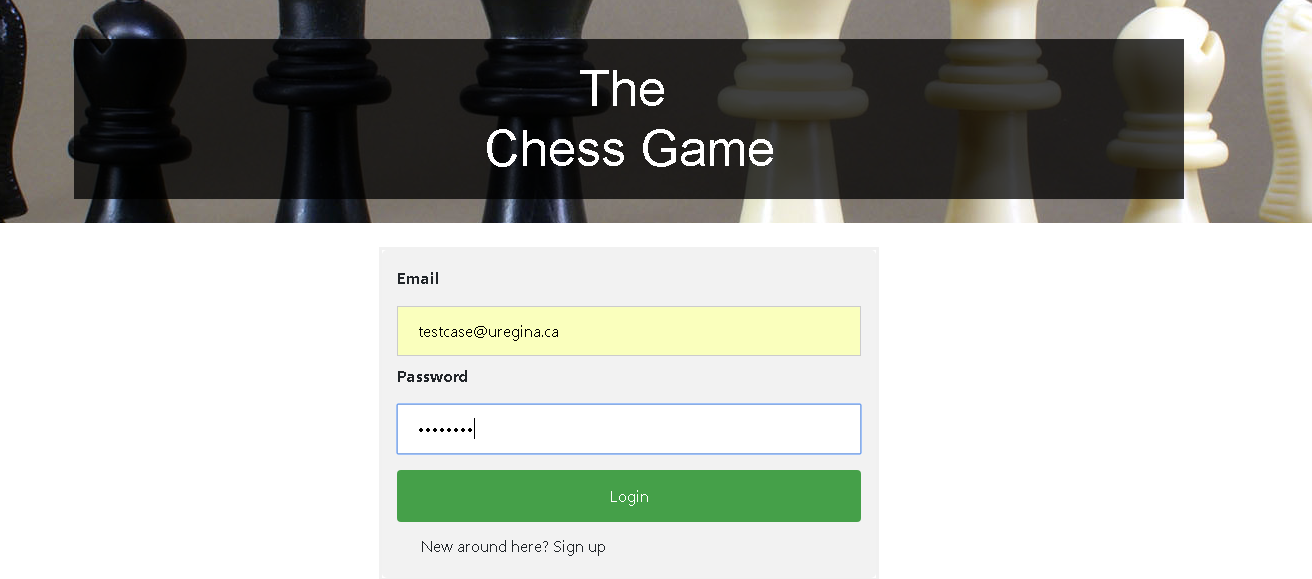
**Objective:** The user must be able to create an account and be able to enter the game with the same account created.

Test steps:

1. Enter the webpage: <http://chess372.azurewebsites.net/Chess_LoginPage.php>
2. Select the option “New around here? Sign up”
3. Fill all the text boxes with the user data:
   1. First Name: Test
   2. Last Name: Case
   3. Email: testcase@uregina.ca
   4. Password:12345678
   5. Re-enter Password:12345678



1. Click button submit
2. Enter email and password



1. The welcome page will show up with the name Test:



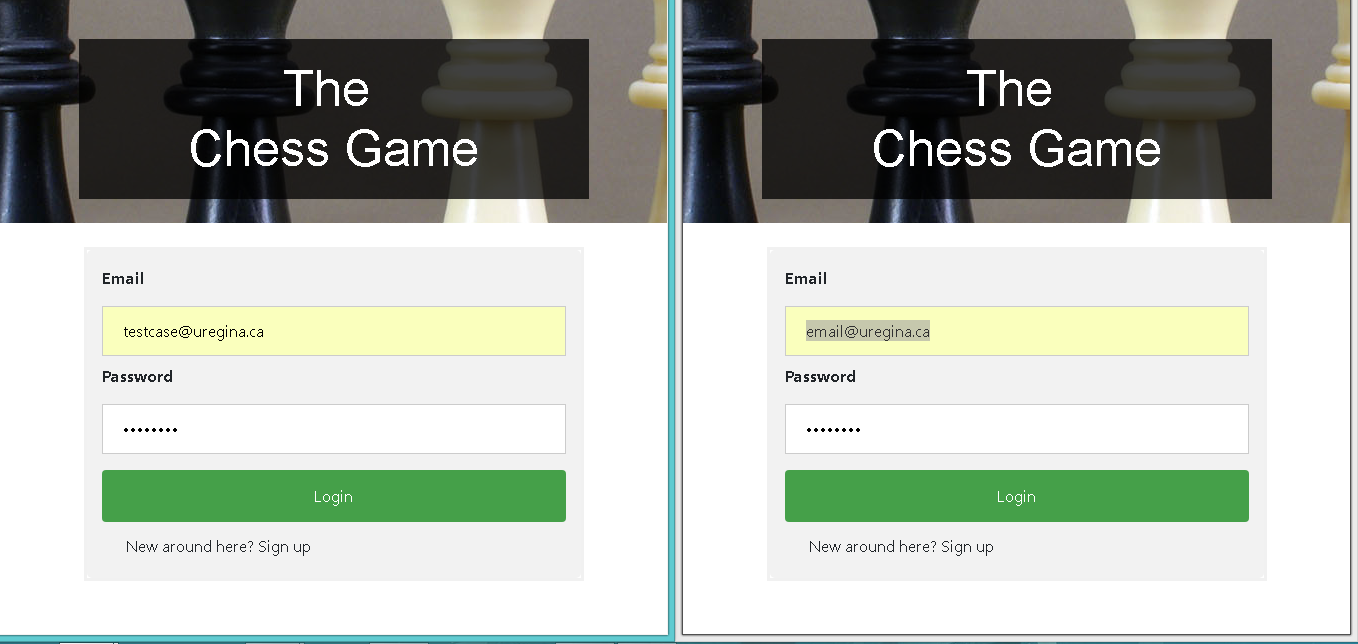
**Expected results**: Account correctly created and used to access the game

Player enter an online room and play against another player

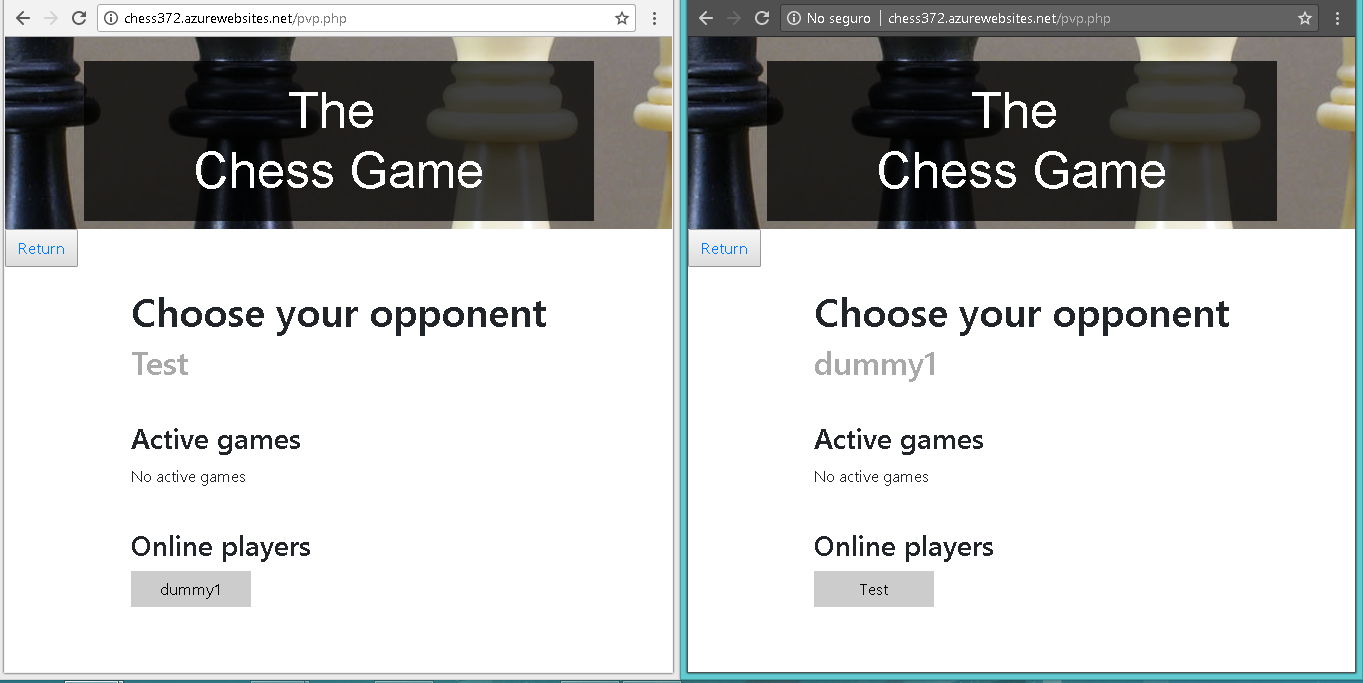
**Objective:** The player joins online games against other players

Test steps:

1. Login to the webpage: <http://chess372.azurewebsites.net/Chess_LoginPage.php>
2. open a new incognito tab and login with another account
   1. First account: [testcase@uregina.ca](mailto:testcase@uregina.ca) password: 12345678
   2. Second account: [email@uregina.ca](mailto:email@uregina.ca) password:12345678



1. Click Start Pvp Match in both Tabs
2. Click in find opponent in both tabs



1. Select in one tab the user from the other tab.
2. Accept the challenge in both Tabs
3. The board will then be see in both tabs:



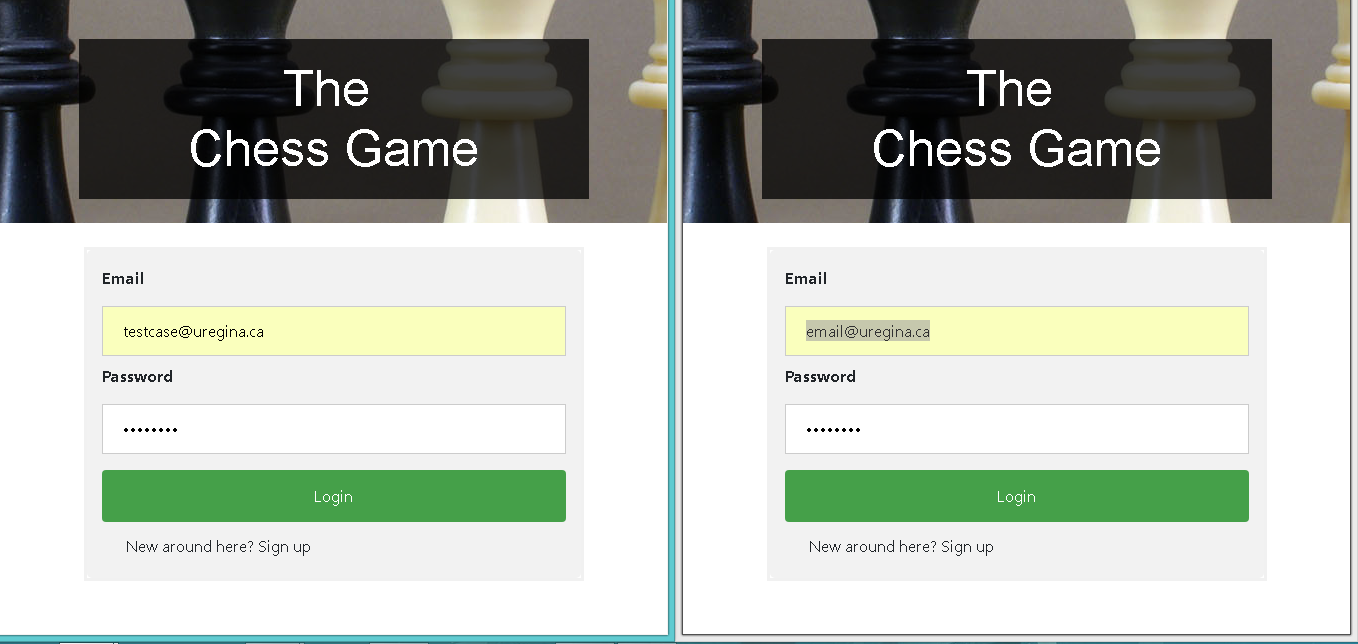
**Expected results**: online players playing with each other

## Real time chess

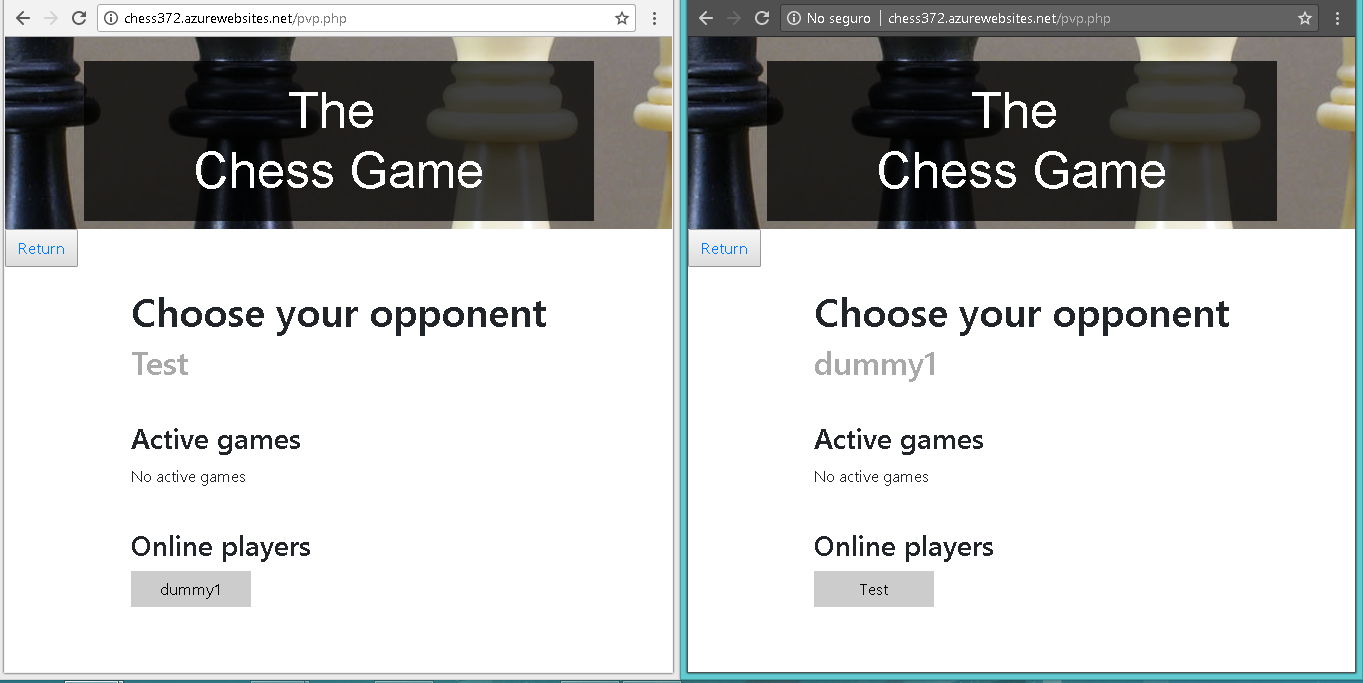
**Objective:** The movement of the game should be able to be viewed in real time in online games

Test steps:

1. Login to the webpage: <http://chess372.azurewebsites.net/Chess_LoginPage.php>
2. open a new incognito tab and login with another account
   1. First account: [testcase@uregina.ca](mailto:testcase@uregina.ca) password: 12345678
   2. Second account: [email@uregina.ca](mailto:email@uregina.ca) password:12345678



1. Click Start Pvp Match in both Tabs
2. Click in find opponent in both tabs



1. Select in one tab the user from the other tab..
2. Accept the challenge in both Tabs
3. Star playing in both tabs and the pieces will move in real time.



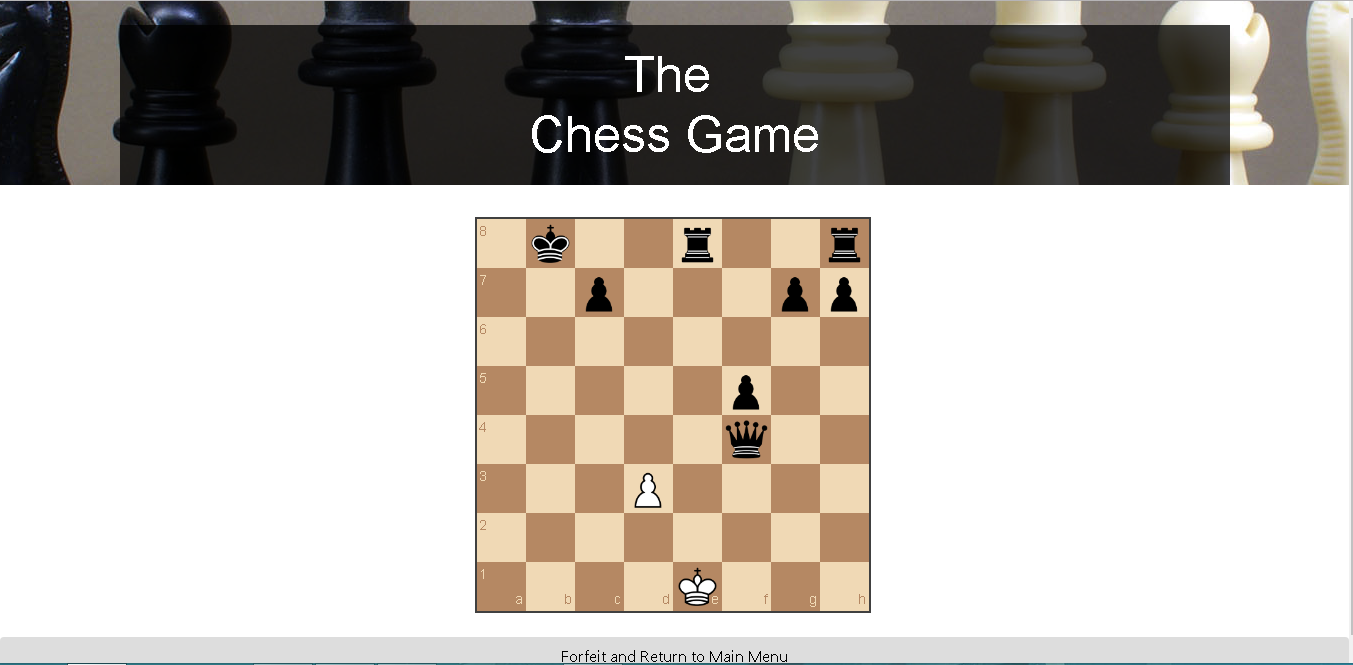
**Expected results**: chess online with real-time movements

## Player plays against the CPU

**Objective:** The movement of the game should be able to be viewed in real time in online games

Test steps:

1. Login to the webpage: <http://chess372.azurewebsites.net/Chess_LoginPage.php>
2. Click Start AI Match
3. Start playing



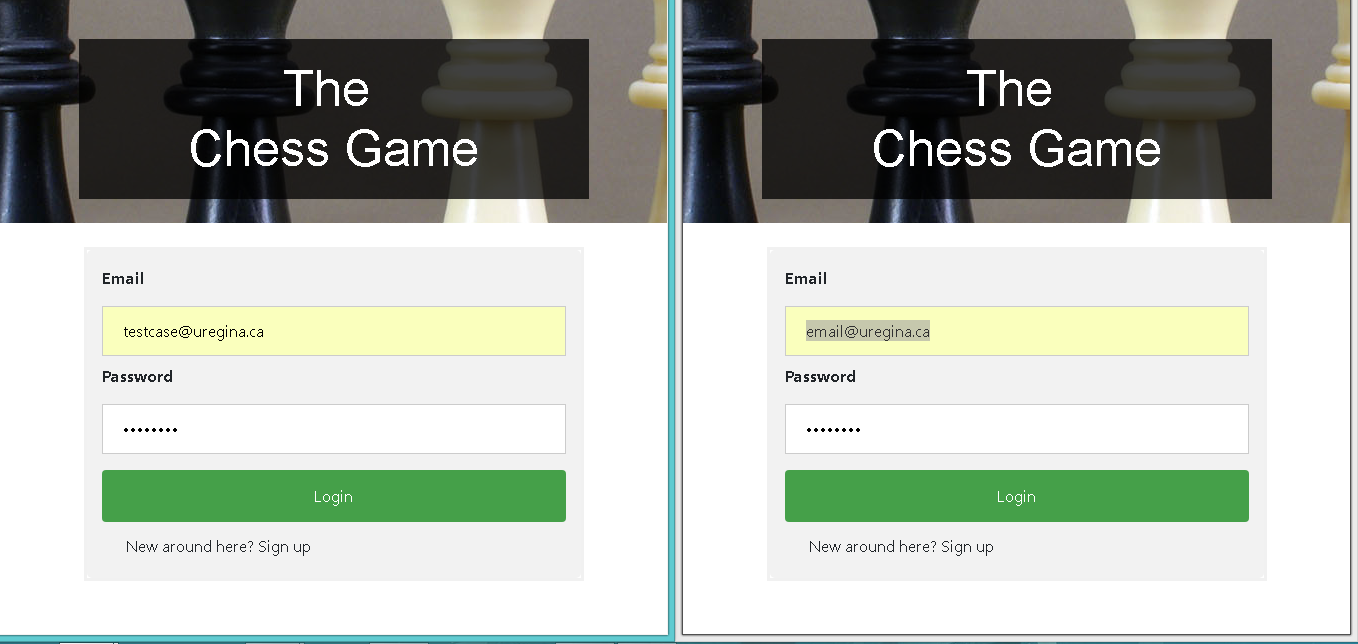
**Expected Results**: Computer plays against the player according to the difficulty

## Leaderboard correctly updates with each win

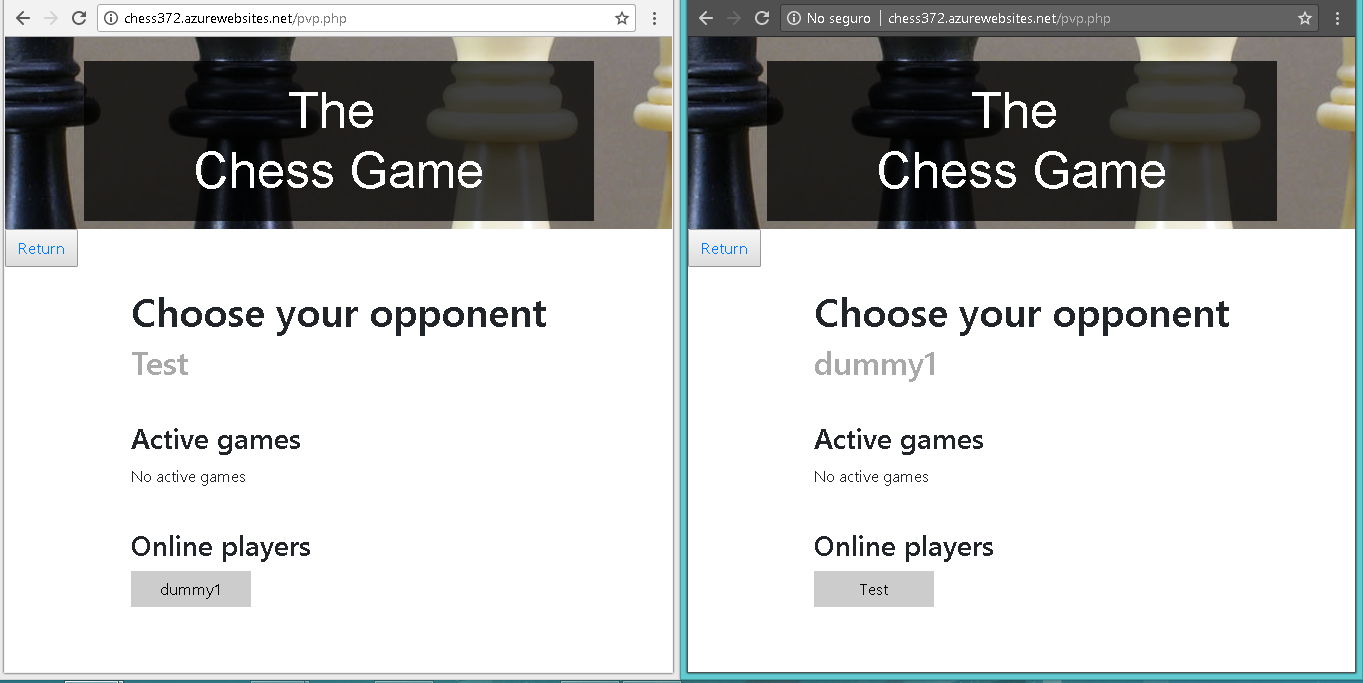
**Objective:** The movement of the game should be able to be viewed in real time in online games

Test steps:

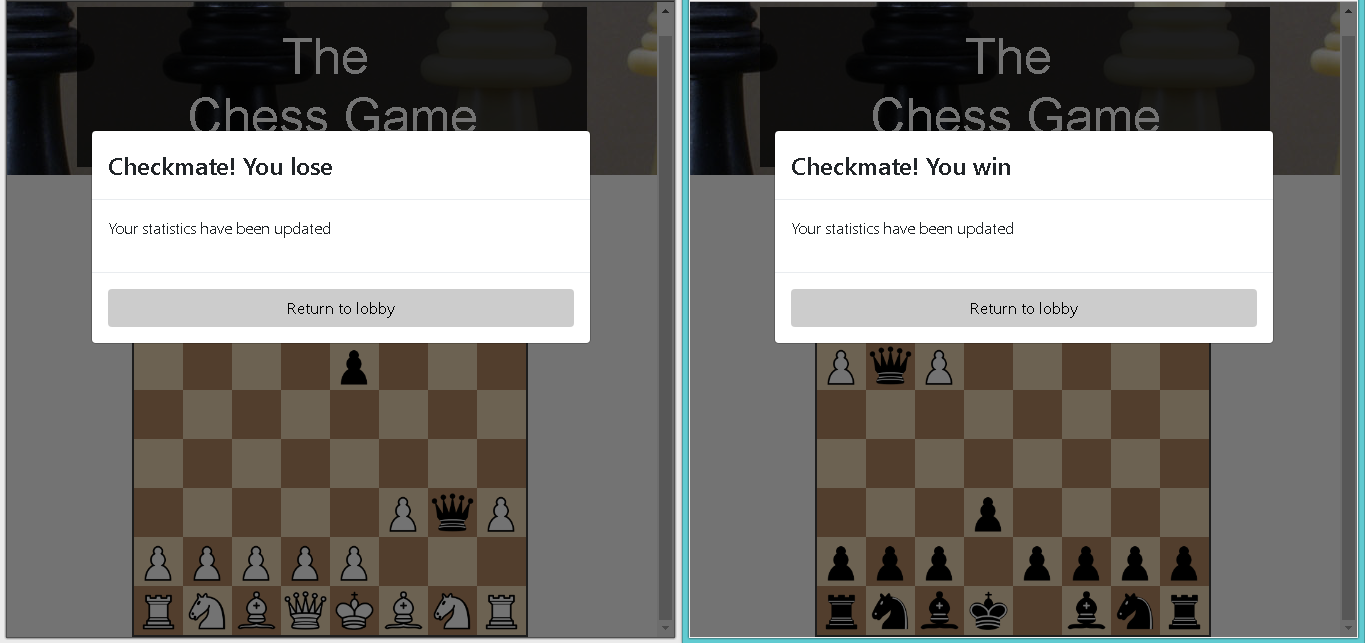
1. Login to the webpage: <http://chess372.azurewebsites.net/Chess_LoginPage.php>
2. open a new incognito tab and login with another account
   1. First account: [testcase@uregina.ca](mailto:testcase@uregina.ca) password: 12345678
   2. Second account: [email@uregina.ca](mailto:email@uregina.ca) password:12345678



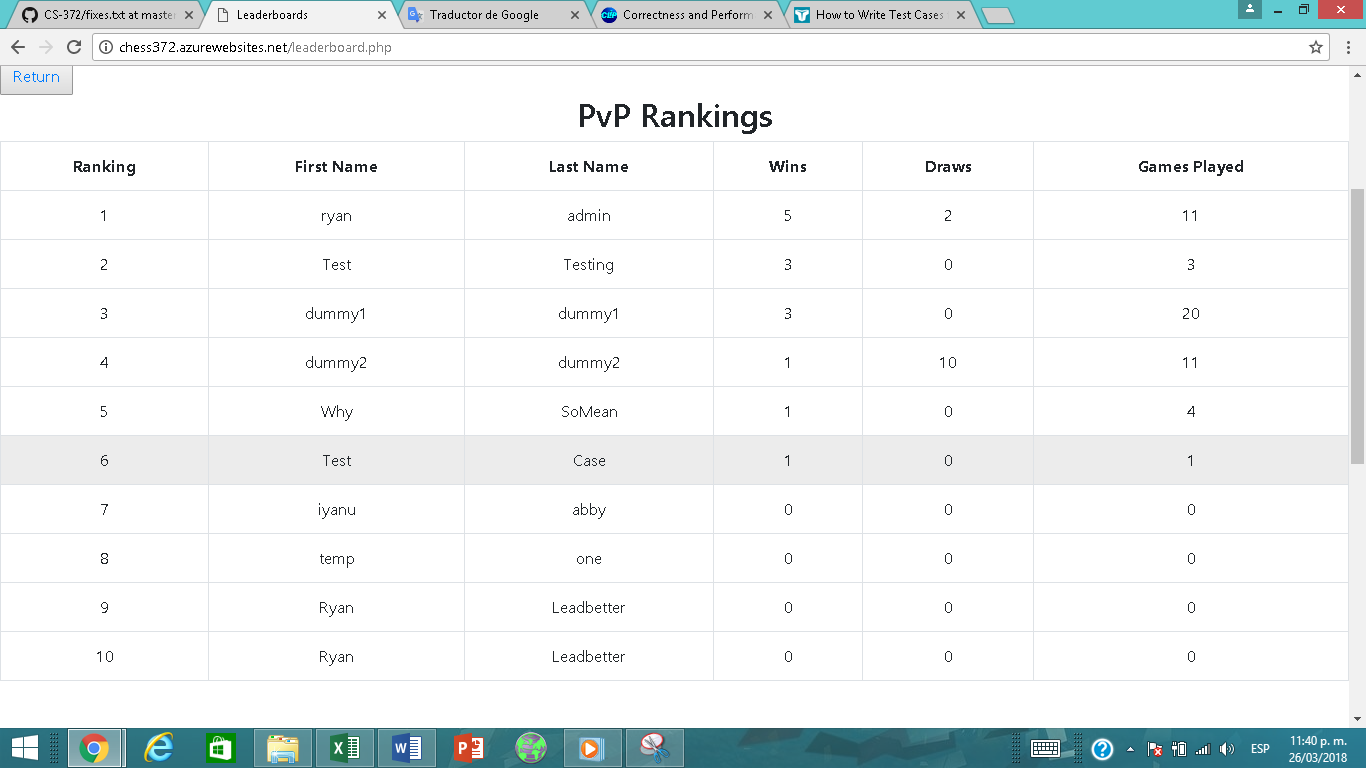
1. Click Start Pvp Match in both Tabs
2. Click in find opponent in both tabs



1. Select in one tab the user from the other tab.
2. Select in one tab the user from the other tab.
3. Accept the challenge in both Tabs
4. Star playing in both tabs
5. Win the match with the first user



1. Go back to the leaderboard and see the wins column of the user:



**Expected results:** Leaderboard correctly update

## Robustness testing with five test cases. For each test case, submit the screenshots of the inputs and outputs.

The resume of test cases for robustness are the following with their respective results:

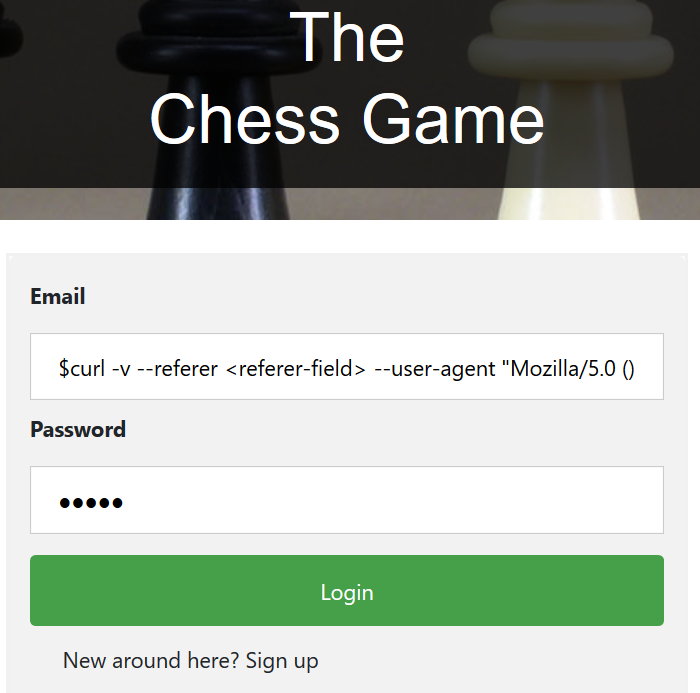
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Control number** | **Action** | **Objective or Explanation** | **Expected results** | **Pass** | **Fail** |
| 1 | Form input validation | The user must not be able to insert invalid characters or data. Hence, preventing the injection of malicious codes. Having said that, a range of valid characters are implemented within the system. | User cannot input invalid characters and will not be allowed to enter the game. | X |  |
| 2 | Reject incorrect moves | The program will prevent user from committing invalid moves of the chess pieces | Program will highlight valid moves for each chess pieces. Moreover, the incorrect move will not be executed and the piece will return to its previous valid location. | X |  |
| 3 | Multiple moves in one turn | According to the rules of chess, each player is only allowed one move at a time. Hence, we will test if numerous move is possible before switching turns with the opposite player. | Each player will only be allowed one move for each turn. | X |  |
| 4 | Player cannot manipulate opponent’s chess pieces | Allowing control over the opponent’s pieces is prohibited within the chess game rules and regulations. Therefore, manipulation of the opponent’s move should not be allowed. | If White was assigned to a player, then he/she can only move the white chess pieces can be moved. And if Black was assigned to a player, then he/she can only move the black chess pieces. | X |  |
| 5 | Chess pieces cannot be moved outside the chessboard | Chess pieces should only be allowed to move towards destination tiles that is within the boundary of an 8x8 matrix board. | The system will not allow user to allocate pieces outside the boundaries of a legitimate chess board. | X |  |

## Form input validation

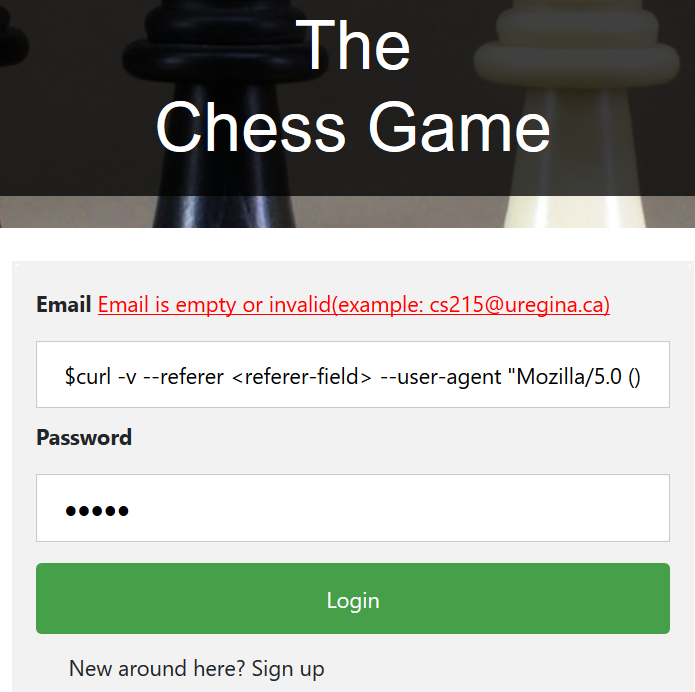
**Objective:** The user must not be able to insert invalid characters or data.

Test steps:

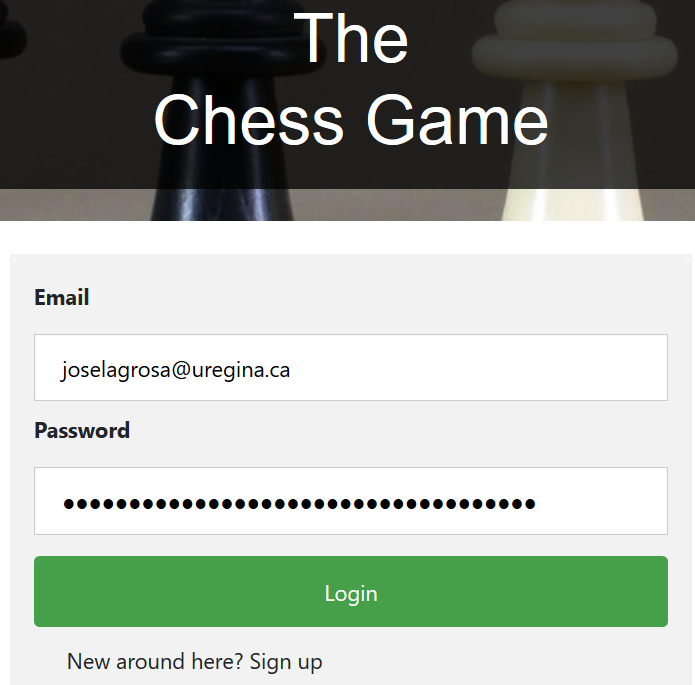
1. Enter the login webpage: <http://chess372.azurewebsites.net/Chess_LoginPage.php>
2. Fill email with invalid characters but with a valid and registered password:
   1. Email: $curl -v --referer <referer-field> --user-agent "Mozilla/5.0 ()
   2. Password: 12345



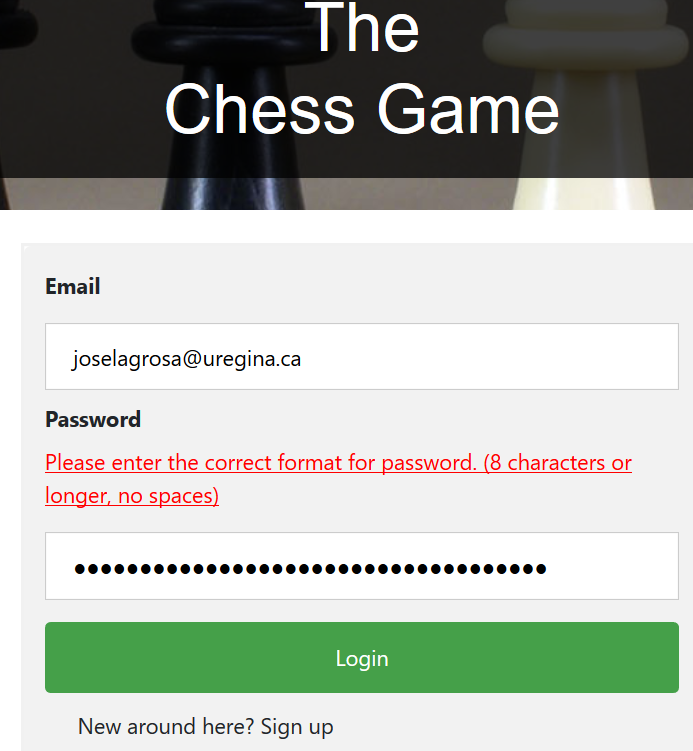
1. Click button submit



1. Fill password with invalid characters but with a valid and registered email:
   1. Email: joselagrosa@uregina.ca
   2. Password: <\_type=password>” with “<\_type=text>



1. Click button submit



**Expected results**: User failed to insert invalid characters and was prevented from entering the game.

## Reject incorrect moves

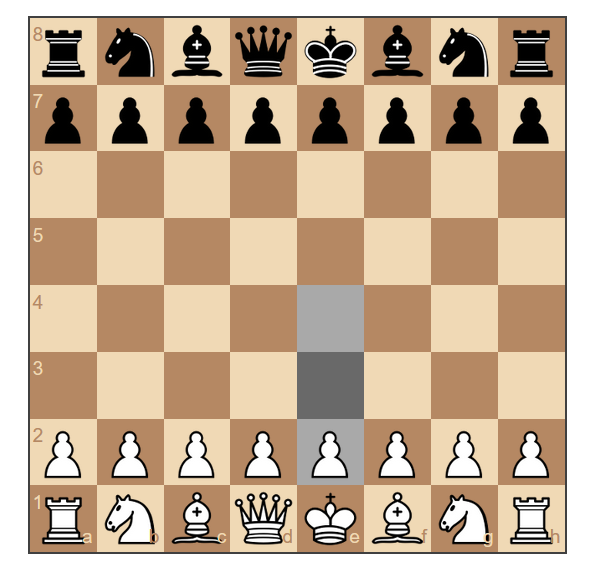
**Objective:** The user must only be allowed to move chess pieces with legal moves.

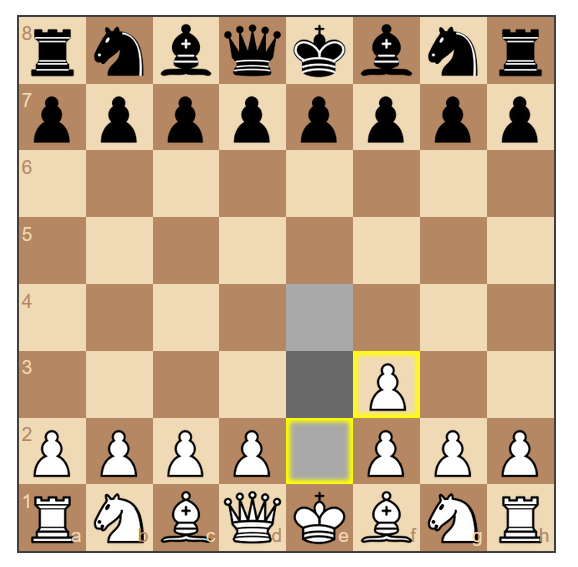
Test steps:

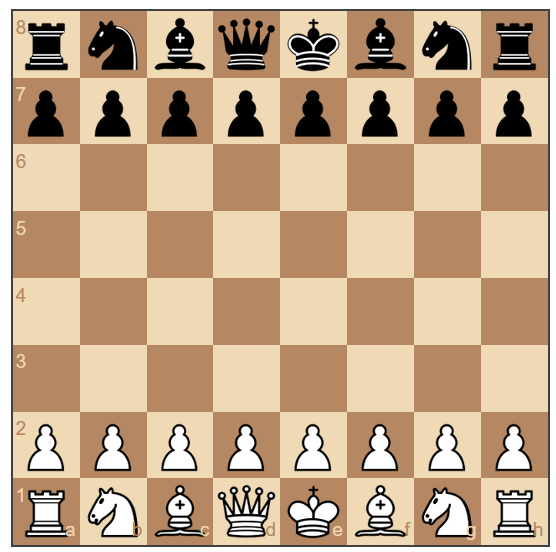
1. Successfully login user



1. Start a match
   1. Play AI Match
   2. Attempt to move a piece with an illegal move by putting it into the highlighted tile.







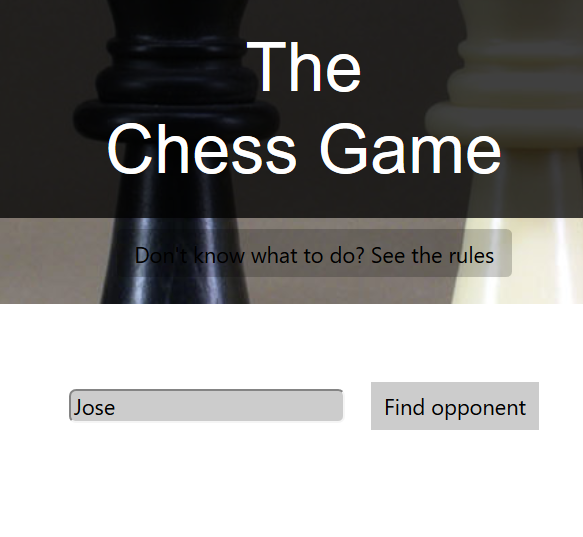
**Expected results**: User unable to execute illegal move.

## Multiple moves in one turn

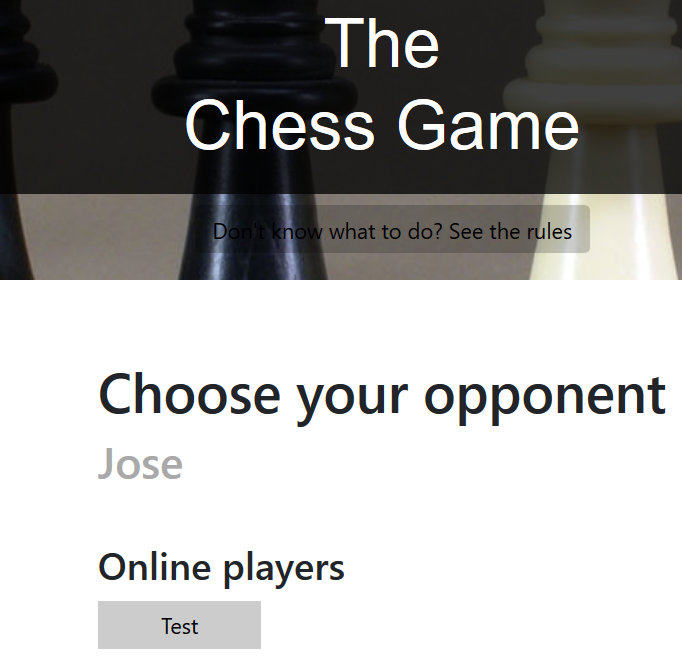
**Objective:** Each player can only move one piece per turn.

Test steps:

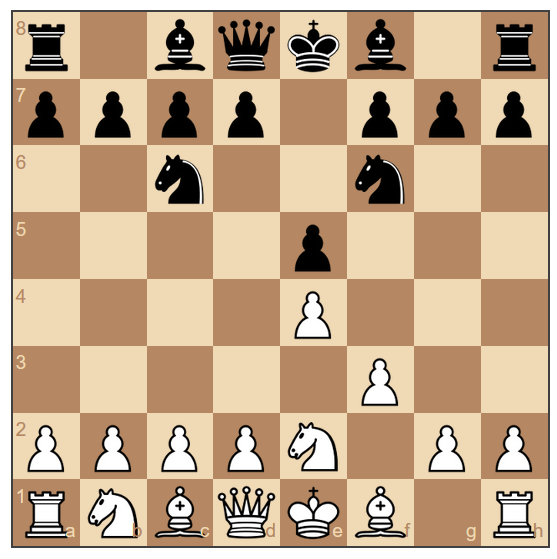
1. Enter PVP lobby



1. Find opponent and Start a PVP game



1. Attempt to move more than one piece in a single turn



|  |  |  |
| --- | --- | --- |
|  | Player’s move | Opponent’s move |
| 1st turn | Pe4 | Pe5 |
| 2nd turn | Pf3 | Nf6 |
| 3rd turn | Ne2 | Nc6 |

* Looking at the figure and table above, we can surmise that each player is only given one move per turn and is not permitted to execute multiple moves in one turn.

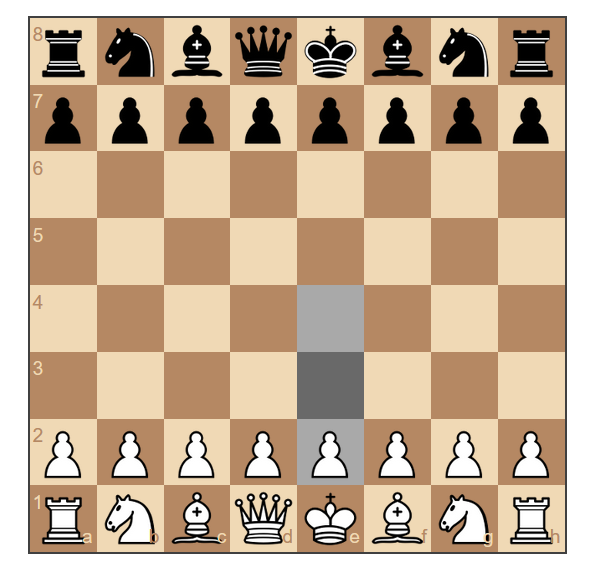
**Expected results**: Each player will only be allowed one move for each turn.

## Player cannot manipulate opponent’s chess pieces

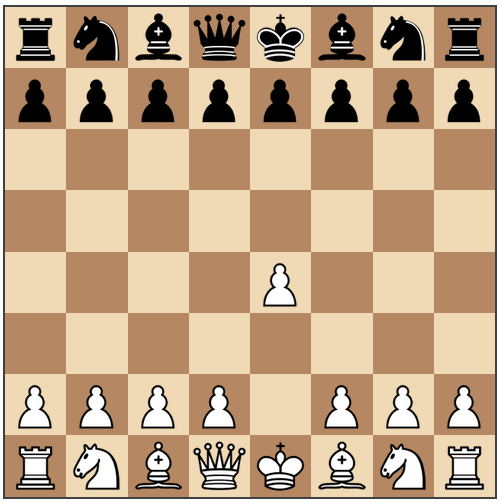
**Objective:** The user must control the piece with the color designated to him/her.

Test steps:

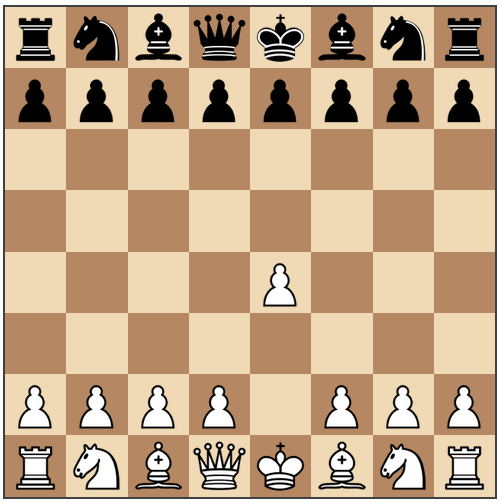
1. Start a game with control over the white pieces



1. Attempt to move a black pawn



1. Attempt to move black horse



* As we can see, the system will not even let the user to click the black pieces, even more forcing it to move.

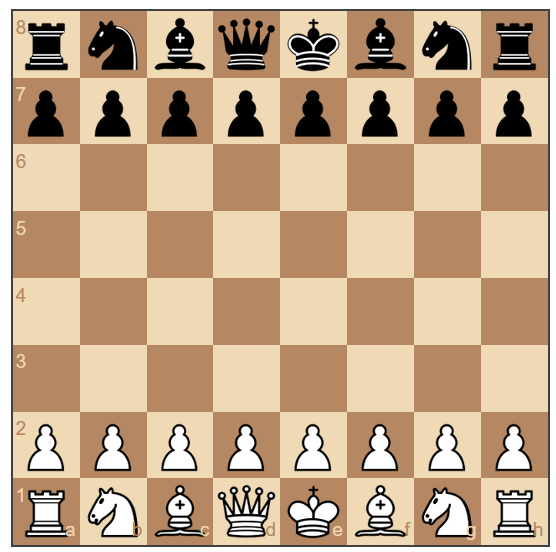
**Expected results**: If White was assigned to a player, then he/she can only move the white chess pieces can be moved. And if Black was assigned to a player, then he/she can only move the black chess pieces.

## Player cannot manipulate opponent’s chess pieces

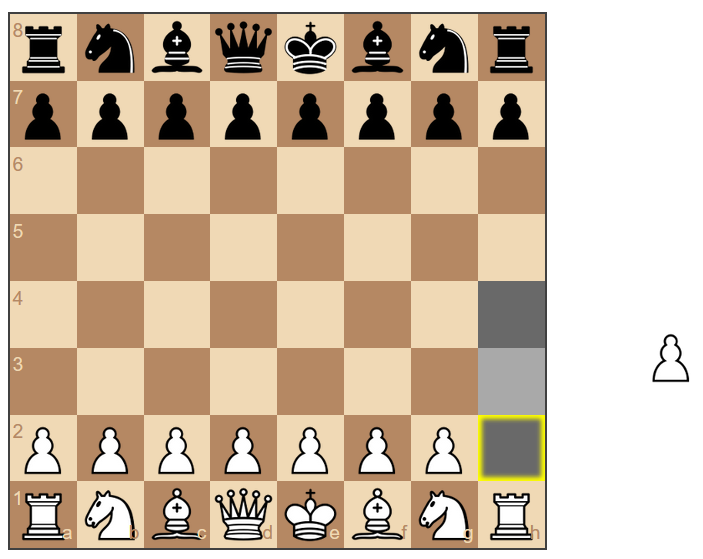
**Objective:** The user must control the piece with the color designated to him/her.

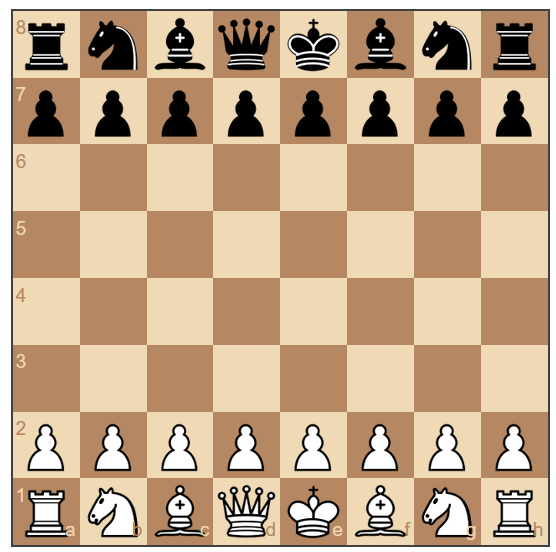
Test steps:

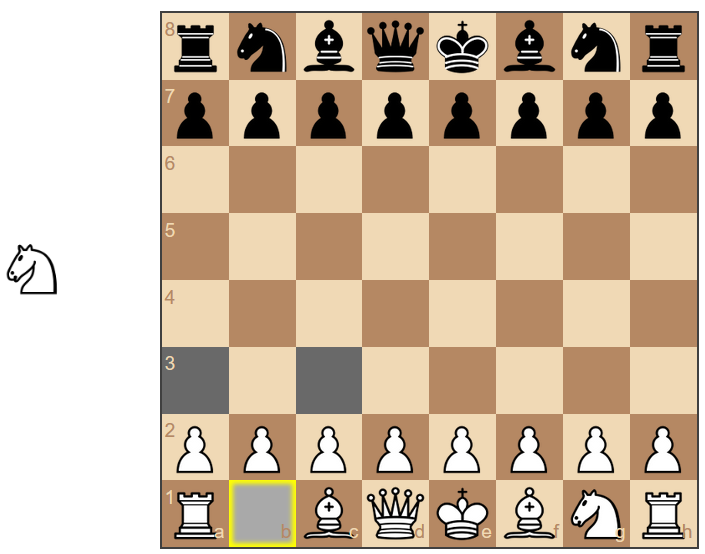
1. Start a game

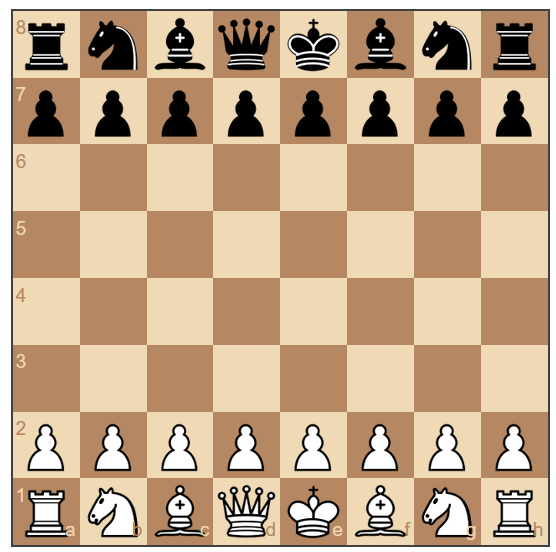


1. Try to move pieces outside the board









**Expected results**: The system will not allow user to allocate pieces outside the boundaries of a legitimate chess board.

## Performance testing of five functions.

Performance testing will be tested using the performance function built into the Google Chrome Web Browser.

1. AI generating a move in AI chess match:

This performance test will determine the time that it will take for the AI to make a move after a player has made a move in a game of chess. The function will be tested for each difficulty as well as 20 separate moves over the course of the game of chess taking the average time of each move.

Results:

* Easy: 215.23ms
* Medium: 197.17ms
* Hard: 3138.23ms

1. Joining a game with another player in PVP chess match:

This performance test will determine the time that it takes for a match to begin once a player has challenged another player in the lobby for a PVP chess match. This will be tested 5 different time taking the average times for the function.

Results: 82.4ms

1. Alerting a player that the opponent has left the match:

This performance test will determine the time that it takes for the player to be notified that the opponent has left the match. This will be tested 5 different time taking the average times for the function.

Results: 11.06ms

1. Alerting a player that the game has ended:

This performance test will determine the amount of time that it takes for a player to be notified that the game has ended and display if they have won or lost the game. This will be tested 5 different time taking the average times for the function.

Results: 77.6ms

1. Wait time for a player move to generate on opponent’s board:

This performance test will determine the amount of time that it takes for a move to be made on the opponent’s GUI, after a player has made a move. This will be tested 5 different time taking the average times for the function.

Results: 61.88ms

# 7. A document that clearly indicates the completed tasks of each member.

***All work was done equally by all team members.***