28-3-2018

CS372 Project

The Chess Game

Melanie MacDonald

Iyanu Akinyemi

Jose Lagrosa

Ulises Oviedo

Ashir Judah

Ryan Leadbetter

Contents

1. Problem definition.

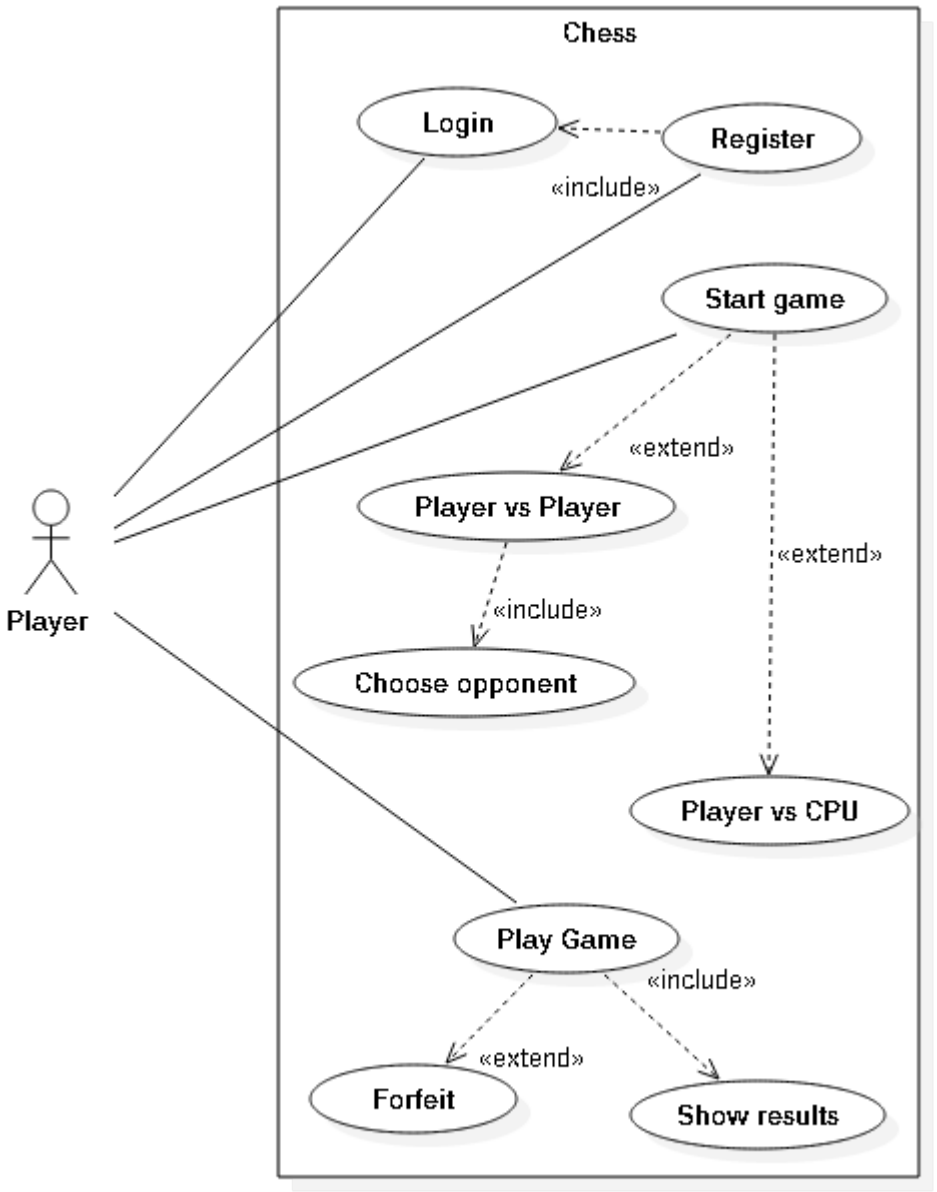
Computers have significantly increased access to multiple forms of entertainment, one such form being video games. Many online games are slow or complicated to use. Users want a game that is responsive and easy to use. We are aiming to create a game a simple chess game that is geared towards those that want to start a quick game against other players or a computer player.

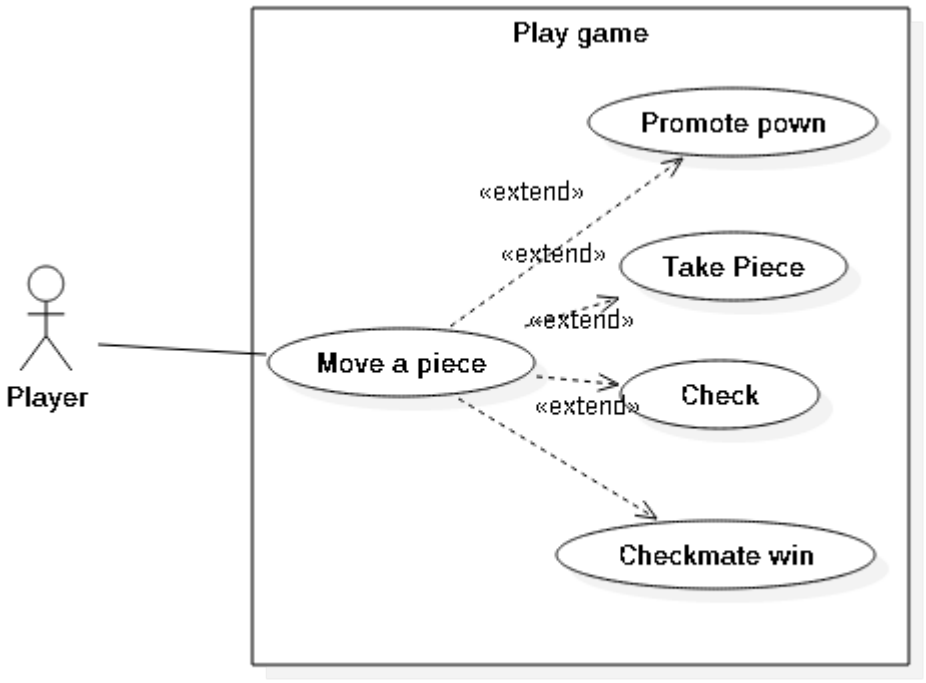
2. Software requirements specification document:

Functional Requirements for Player

- Start Game: The user can begin a game against a computer.
- Display Leaderboard: Look at the chess rankings.
- Find Game: The user can challenge another player or accept a game request.

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




Play game
Promote pown
«extend»
«extend»
Take Piece
«extend»
Move a piece
Check
«extend»
Checkmate win
Player

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

Use case: “Start Game”

Initiating actor: Player

Pre-conditions: Player is logged in

Primary Scenario:

1. Player selects the “Start PvP Match” or “Start AI Match” option from the UI
2. If Player selected to play against another player, they are entered into a lobby. If they have selected to play against the AI, they are taken to a difficulty options screen for the AI.
3. If the Player has selected to play against the AI, upon selecting a difficulty the game starts.

Exceptions:

1. The Player’s login cookie has expired during the process and they are no longer a verified user.
    - *The user will be then redirected to the login page.*

2. The Player no longer wishes to start a game or wants to reset their decisions. Upon clicking “Return”:
    - *The user is redirected to the main menu*

3. The Player somehow loses connection to the webpage during an AI match.
    - *The game will not count as a loss or a win.*

Post-conditions: The Player is entered into a matchmaking lobby if they have chosen to play against another player, or if they have chosen to play against the AI the game starts with the specified settings.

Benefiting Actor: Player

Use case: “Find Opponent”

Initiating actor: Player

Pre-conditions: Player is logged in and has selected to start a game against another Player

Primary Scenario:

1. Player is entered into the lobby
2. Player is given a choice of opponents to challenge
3. On clicking an opponent the player sends an invite to their opponent
4. If the player being challenged agrees to start the match, both players are entered into a game and the player who starts the game will be the white pieces

Exceptions:

1. The Player’s login cookie has expired during the process and they are no longer a verified user.
    - *The user will be then redirected to the login page.*
2. A Player is challenged to a match, and the following message displays:
    - *You are being challenged! Would you like to accept? Yes No*

    While the Player sending the challenge is shown the following message:
    - *Challenge sent. Response pending… If you are rejected you will return to the lobby*

    If the Player rejects the invite, both players are returned to the lobby.
3. The Player somehow loses connection to the webpage while in a game.
    - *The game will not count as a loss or a win.*
4. The Player wishes to exit the lobby. Upon selecting “Return”:
    - *The user is removed from the lobby and redirected to the main menu*

Post-conditions: Player is in a game against their matched opponent

Benefiting Actor: Player

Use case: “Forfeit”

Initiating actor: Player

Pre-conditions: Player is logged in and currently in a game

Primary Scenario:

1. A Player or the AI wins the game
2. The result of the match is displayed, and the appropriate game statistics are updated
3. The Player selects to “Return to main menu” in a match against the AI, or “Return to lobby” if playing player versus player and is redirected appropriately, ending the game

Exceptions:

1. Player willingly leaves a match partway. System ends game and presents a message to remaining Player:
    - *Your opponent has forfeited!*
    - *You win by default Return to lobby*

    The Player who is leaving is presented the following message on exit:

    - *Forfeit?*
    - *This match will be recorded as a loss Yes.*
2. A fatal connectivity error occurs during the match.
    - *The game will not count as a loss or a win.*

Post-conditions: Player is no longer in a game

Benefiting Actor: Player

C. 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 leaderboard. 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.

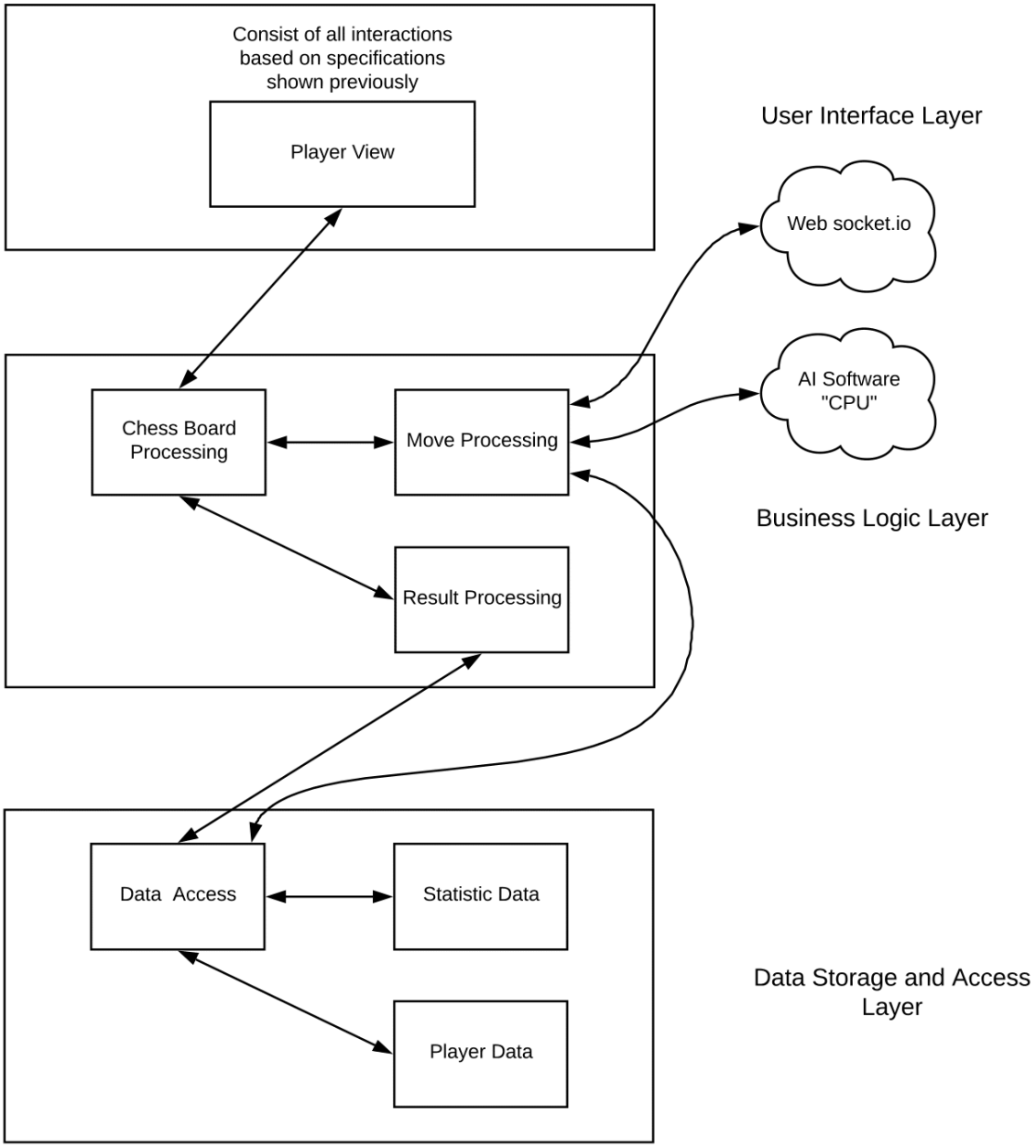
User Friendliness: A simple menu-based design will be used to allow users to quickly and efficiently move about the various game types. Finding games will be clear and concise. The available players for online games should displayed in a simple way in order for players to find specific opponents. The 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 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.

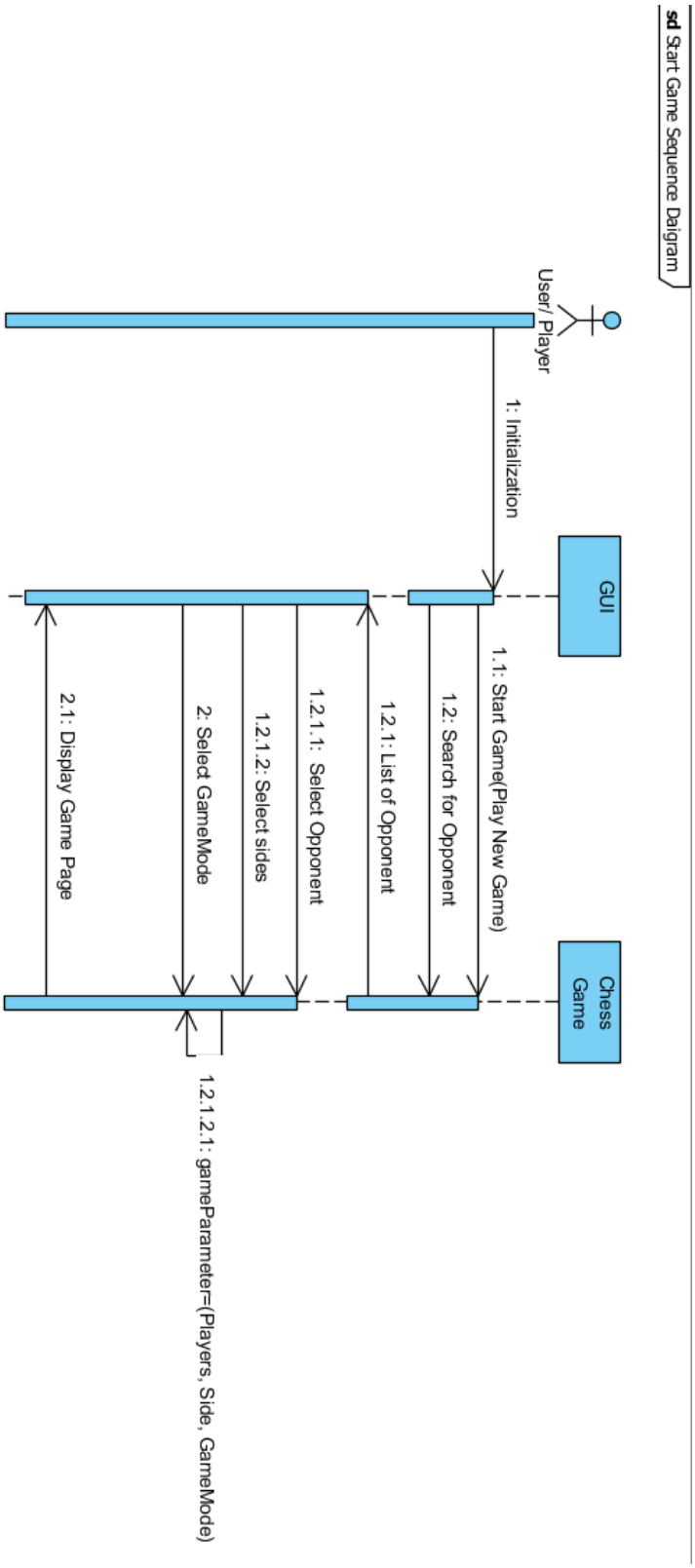
3. Design specification document

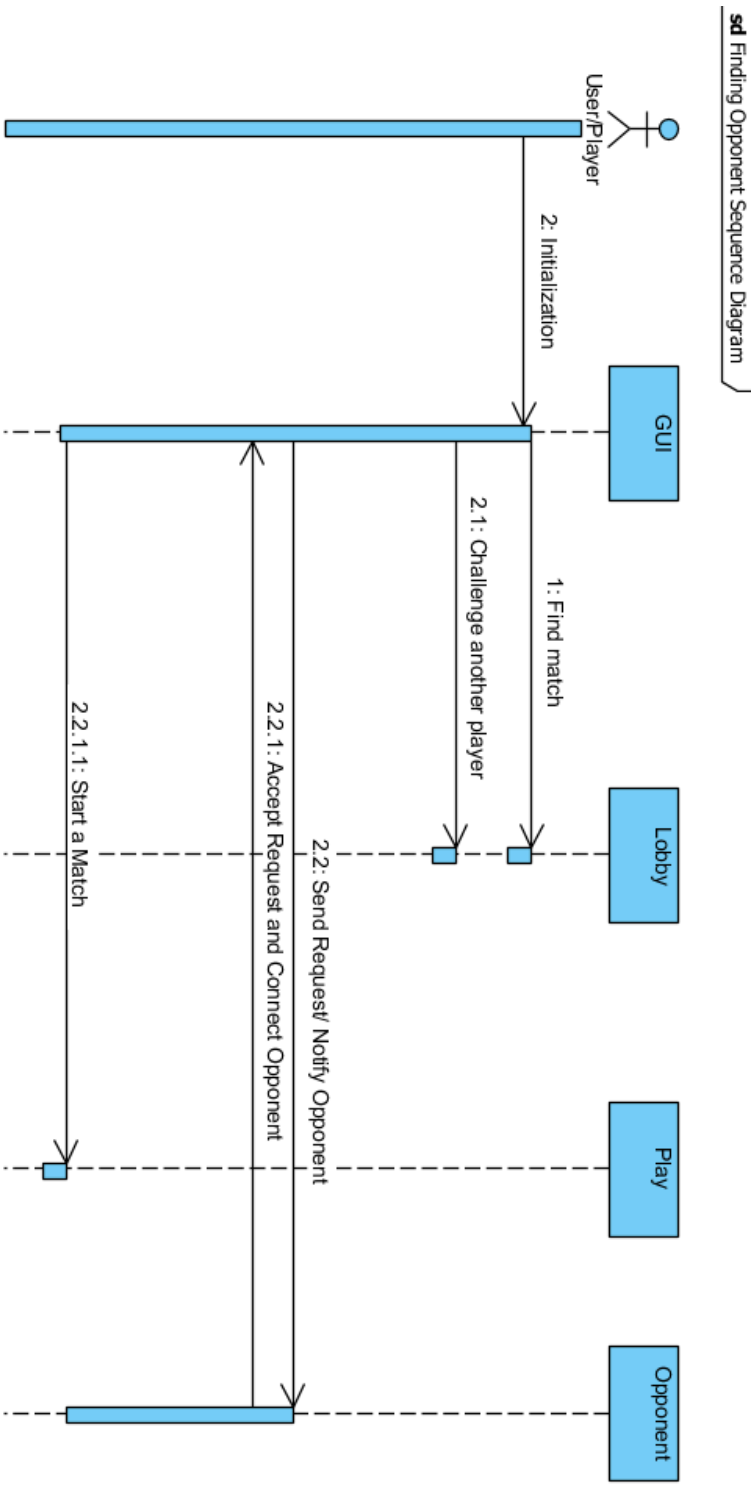
A. Software architecture (multi-layer).

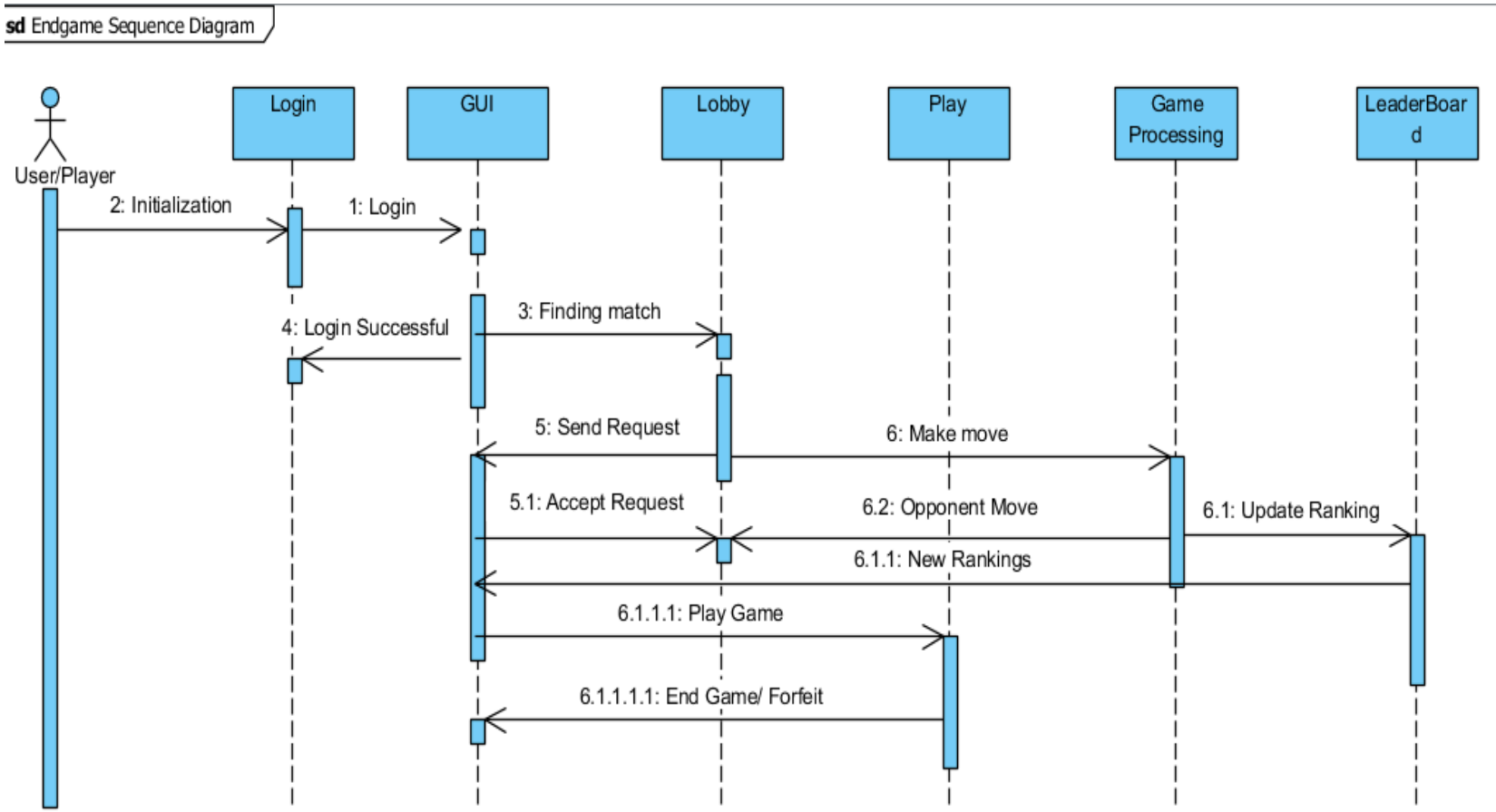
3 - Layer Architecture
(Chess Application)



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

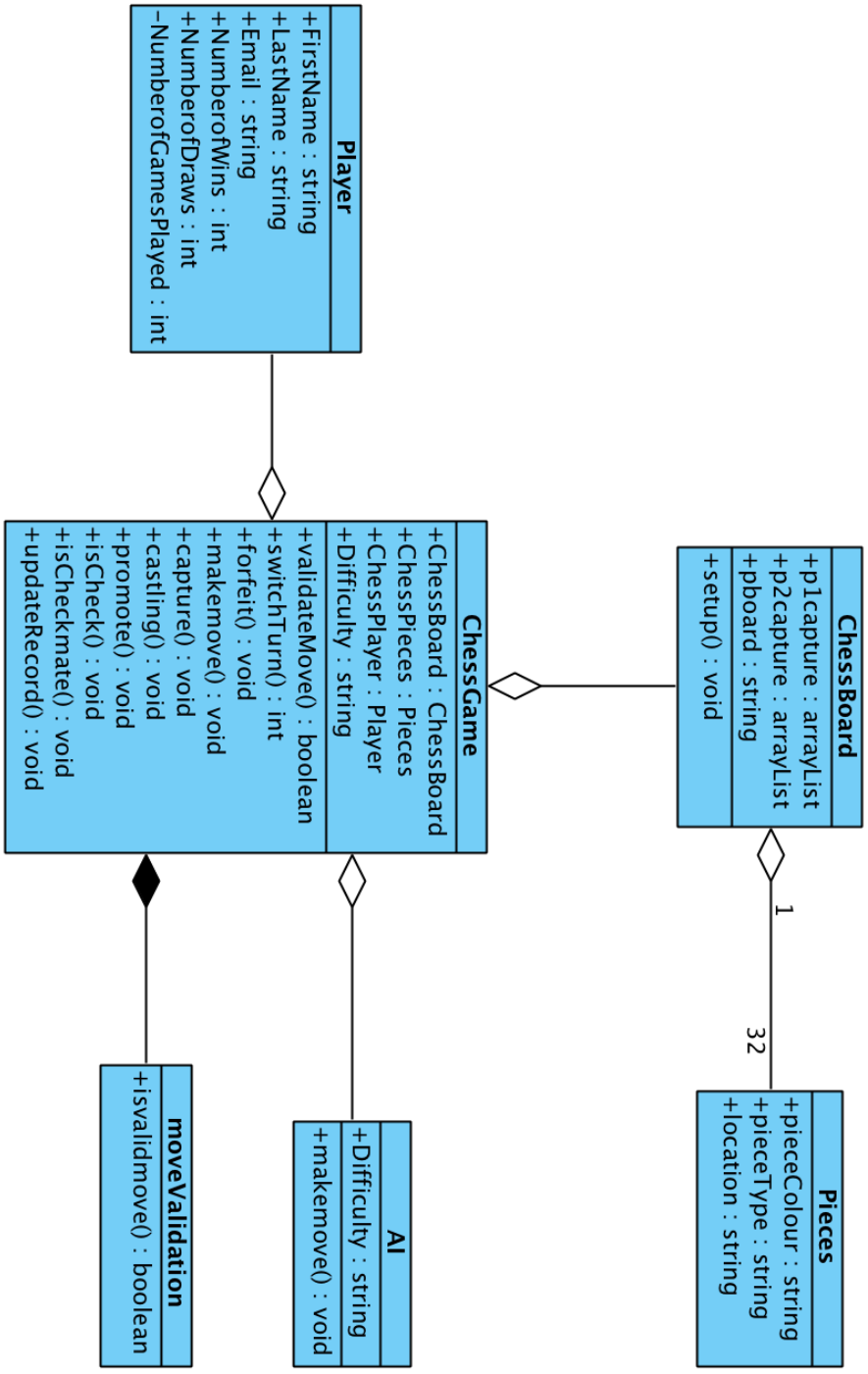



sd Finding Opponent Sequence Diagram
User/Player
GUI
Lobby
Play
Opponent
2: Initialization
1: Find match
2.1: Challenge another player
2.2: Send Request/ Notify Opponent
2.2.1: Accept Request and Connect Opponent
2.2.1.1: Start a Match

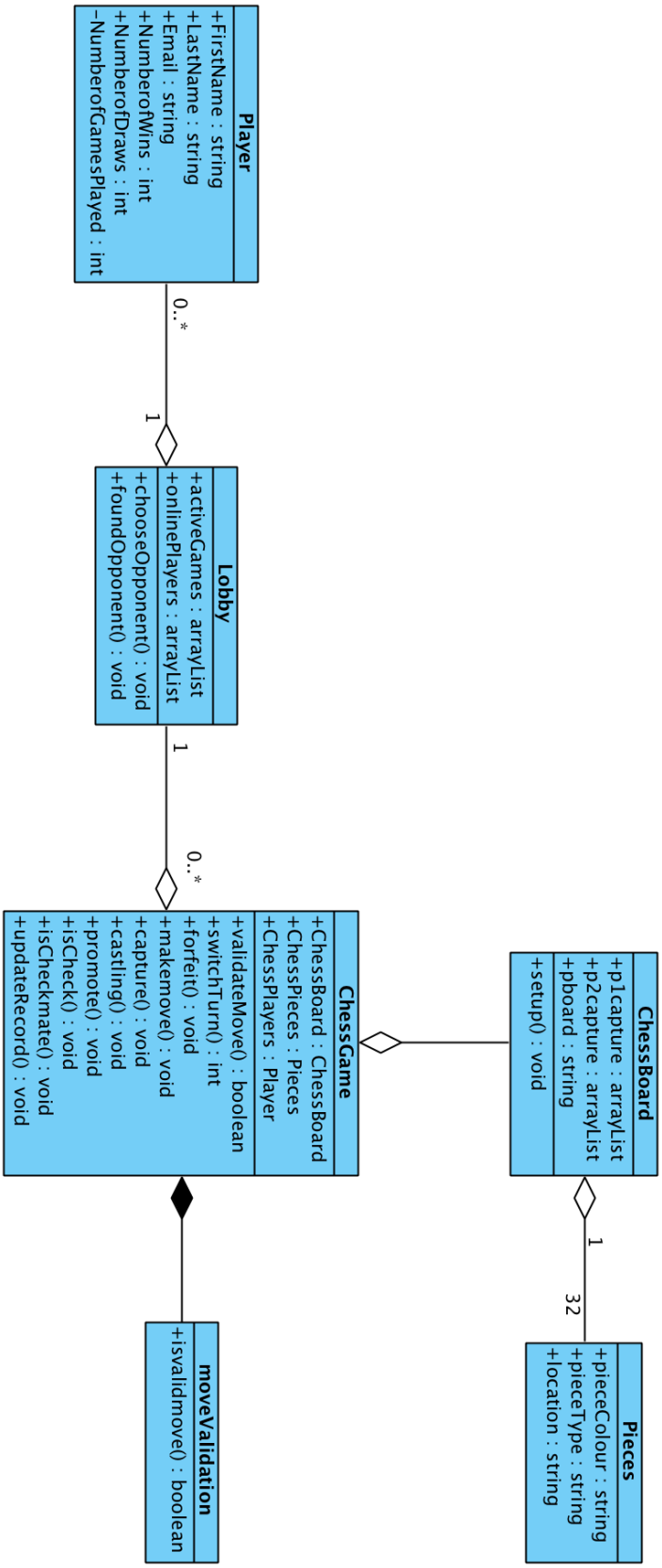

sd Endgame Sequence Diagram
User/Player
Login
GUI
Lobby
Play
Game Processing
LeaderBoard
2: Initialization
1: Login
4: Login Successful
3: Finding match
5: Send Request
6: Make move
5.1: Accept Request
6.2: Opponent Move
6.1: Update Ranking
6.1.1: New Rankings
6.1.1.1: Play Game
6.1.1.1.1: End Game/ Forfeit

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

PlayerVsCPU Class diagram

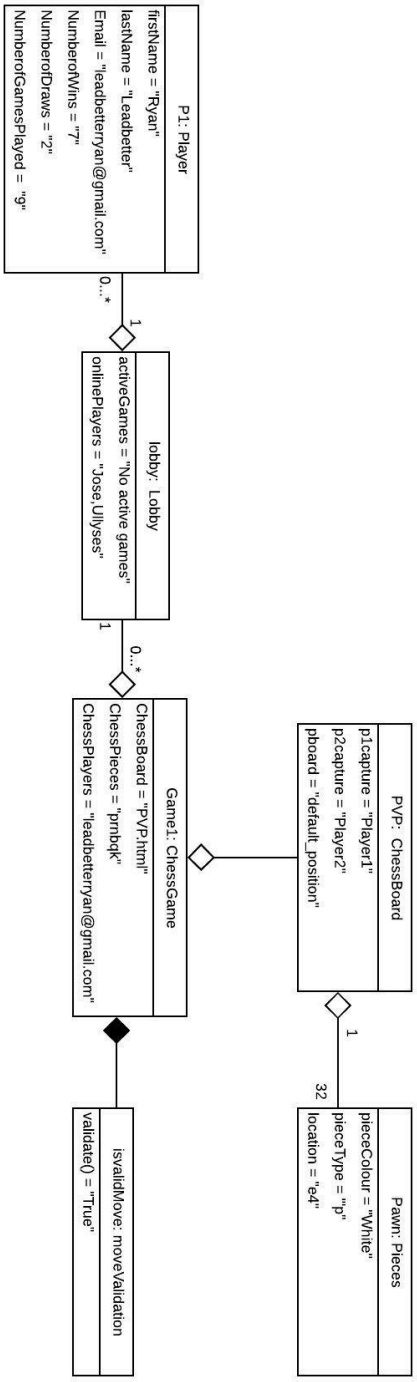


PlayerVsPlayer class diagram

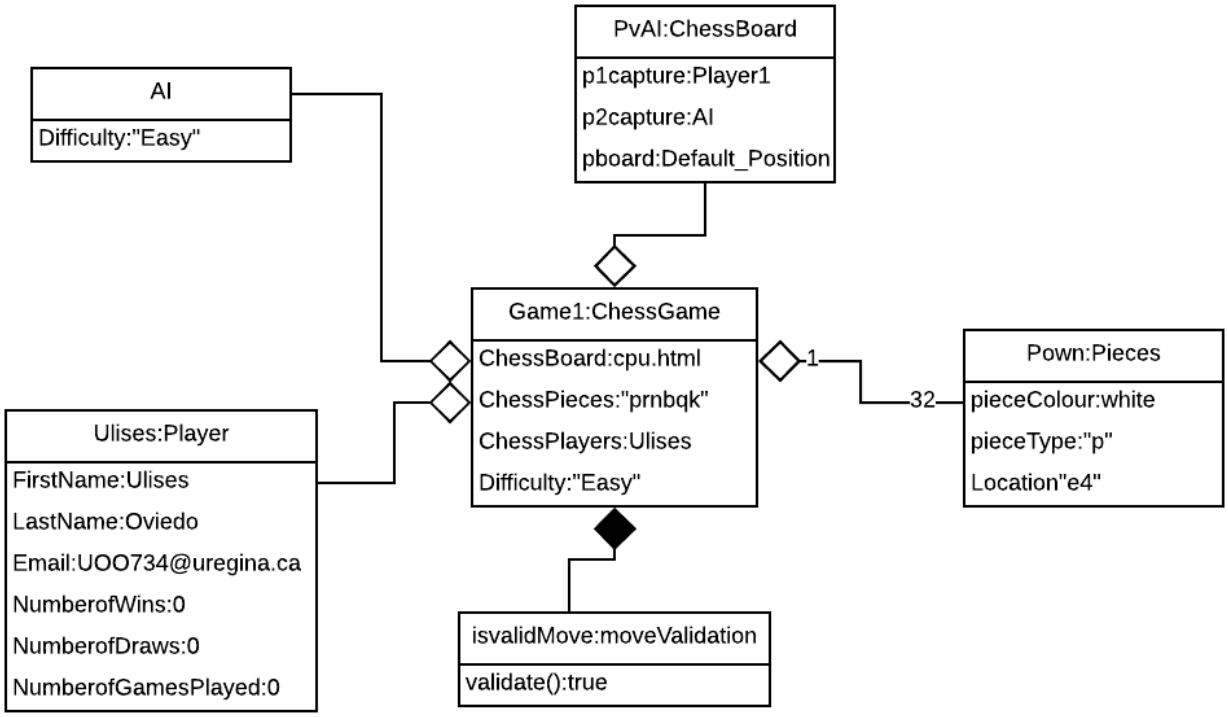


D. Two examples of object diagrams.

PvP Object Diagram



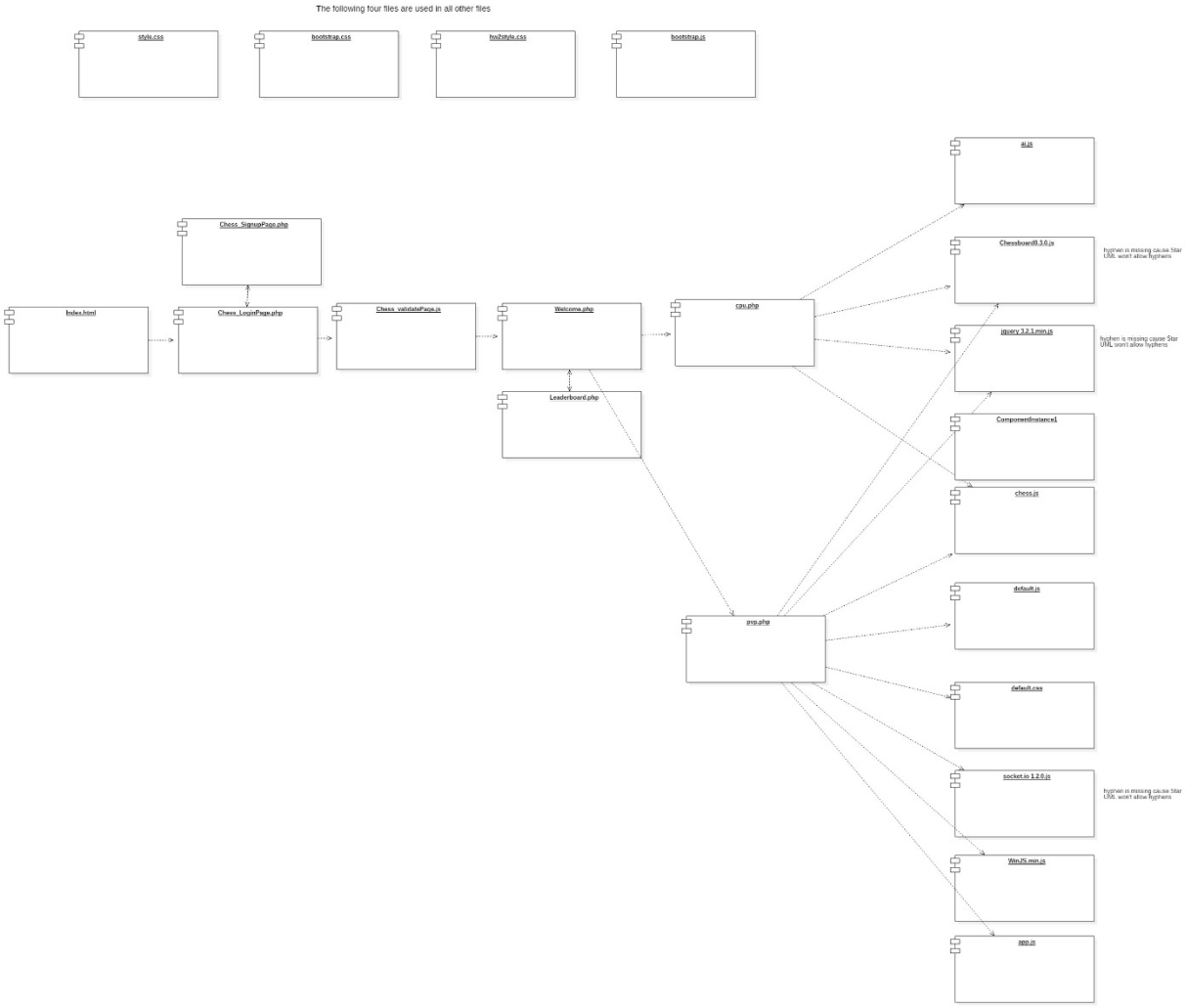
PlayerVsCPU Object Diagram



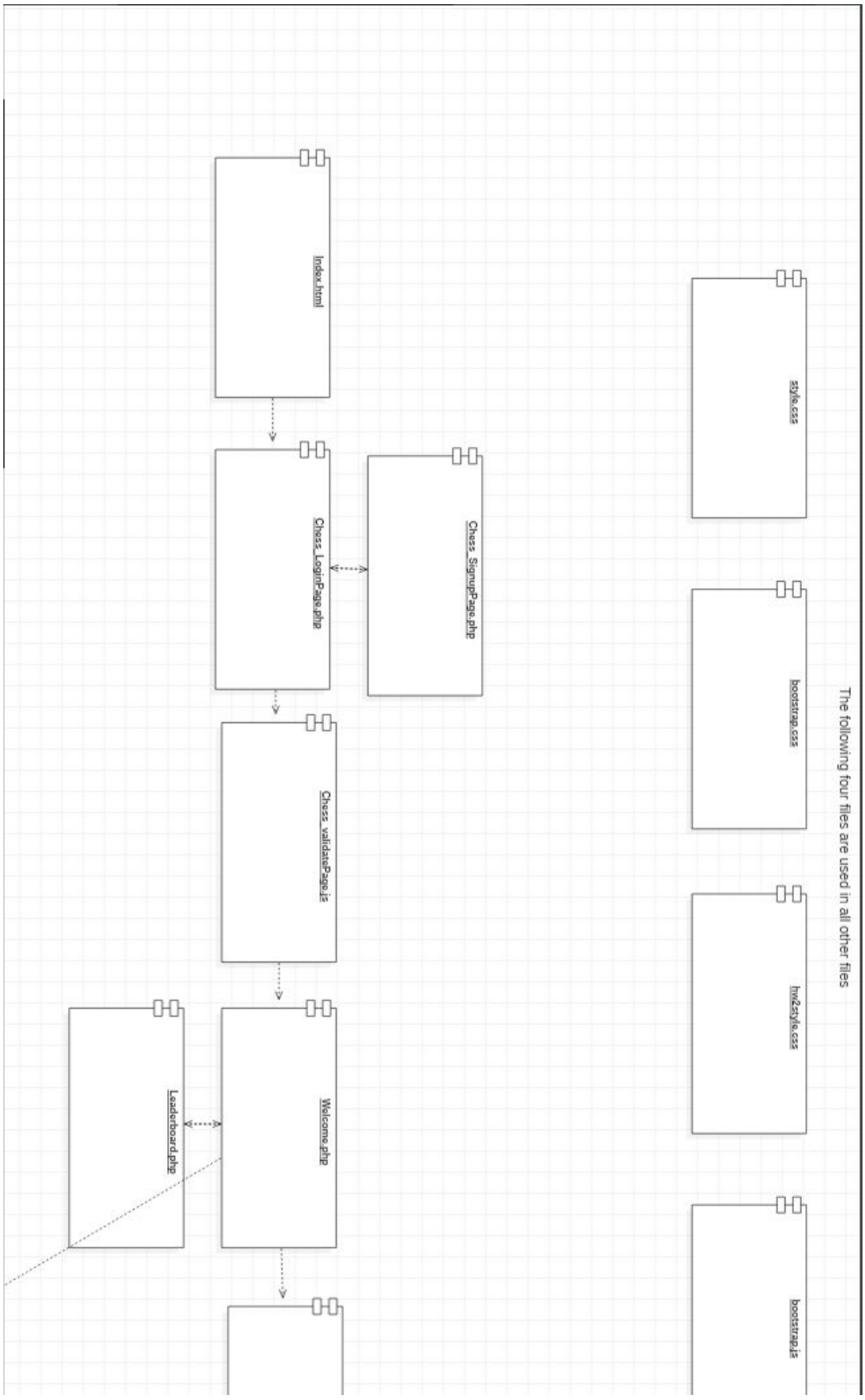
4. Code description, including:

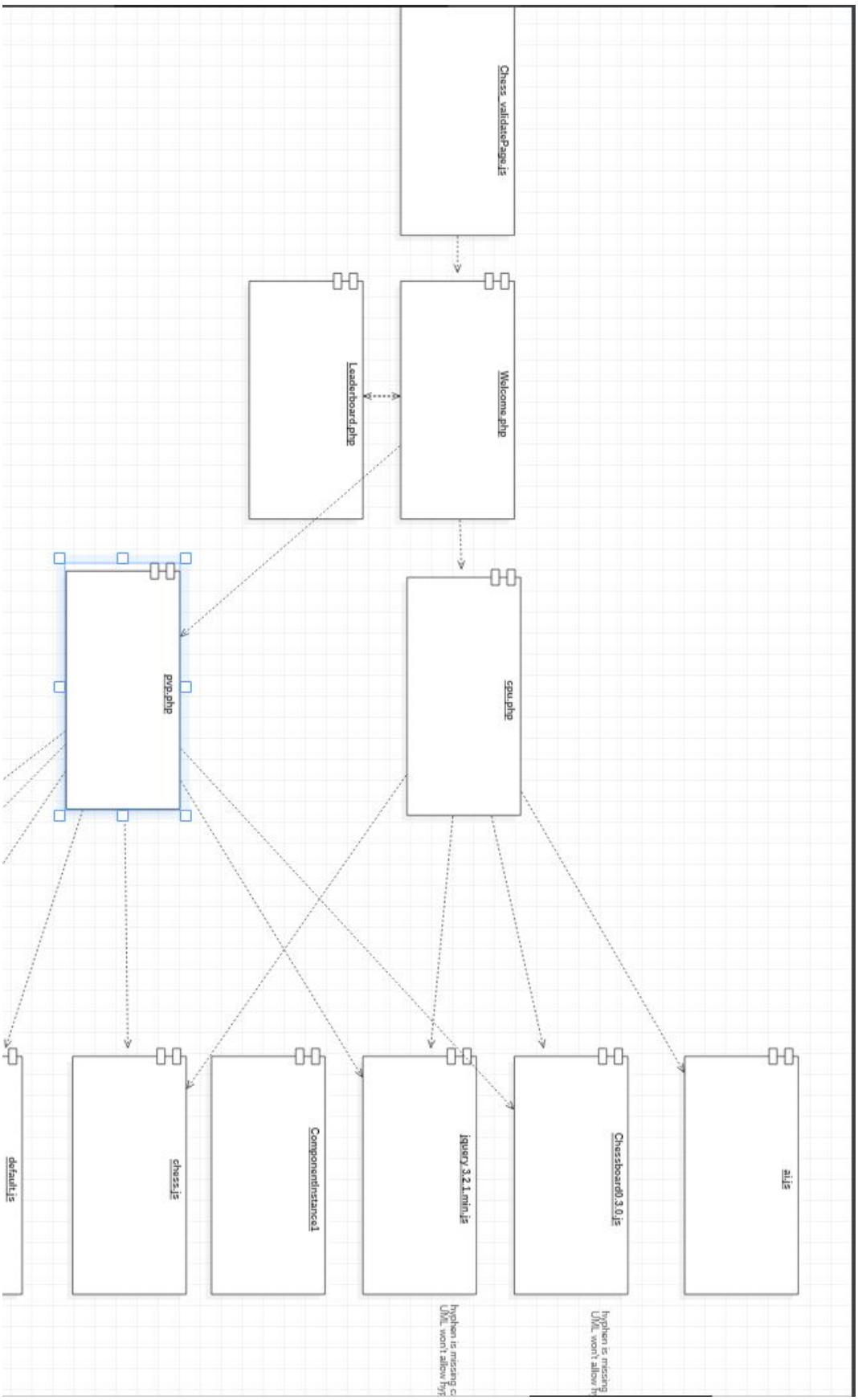
A. Component diagram regarding the organization of the source code.

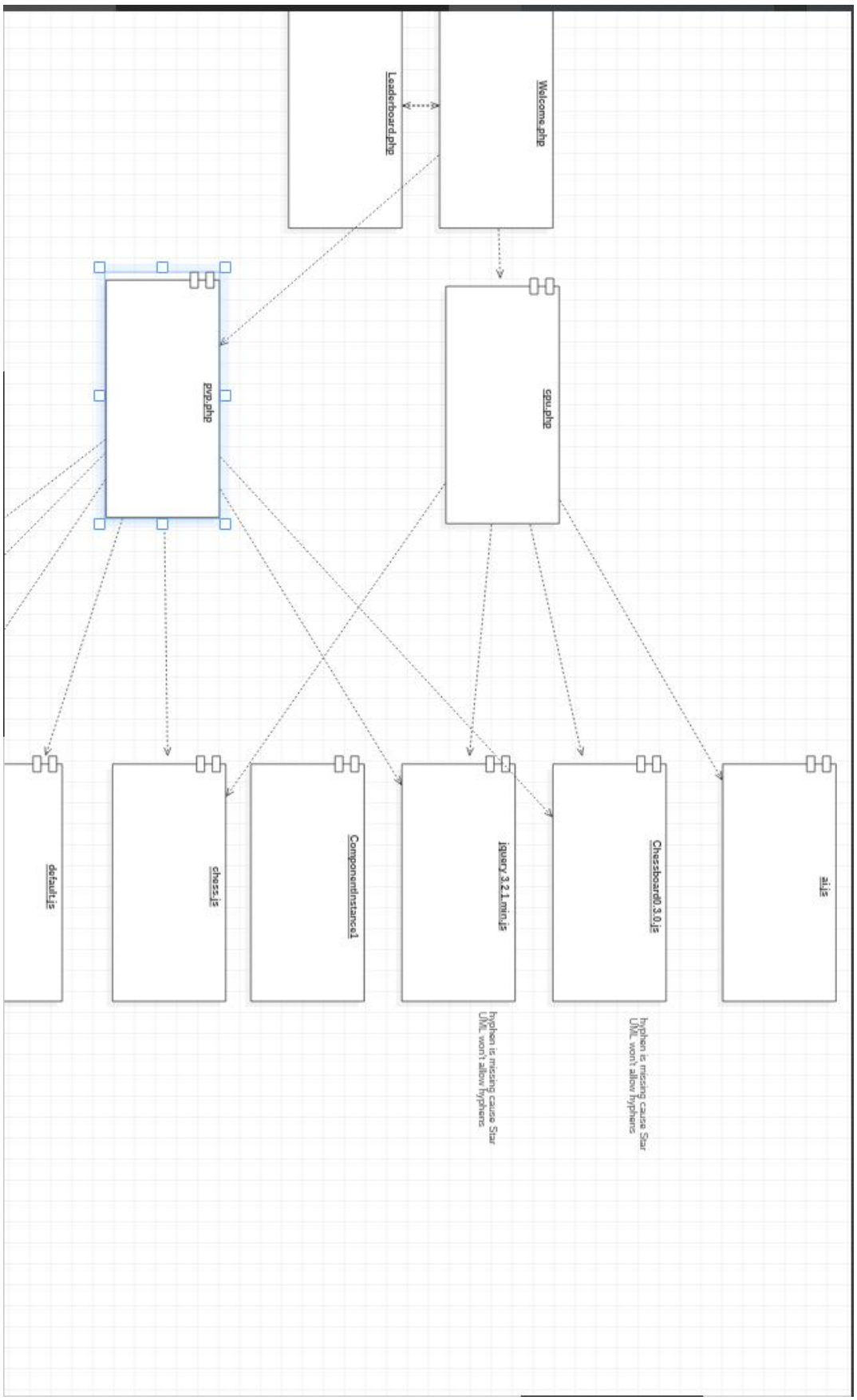
Complete Component diagram

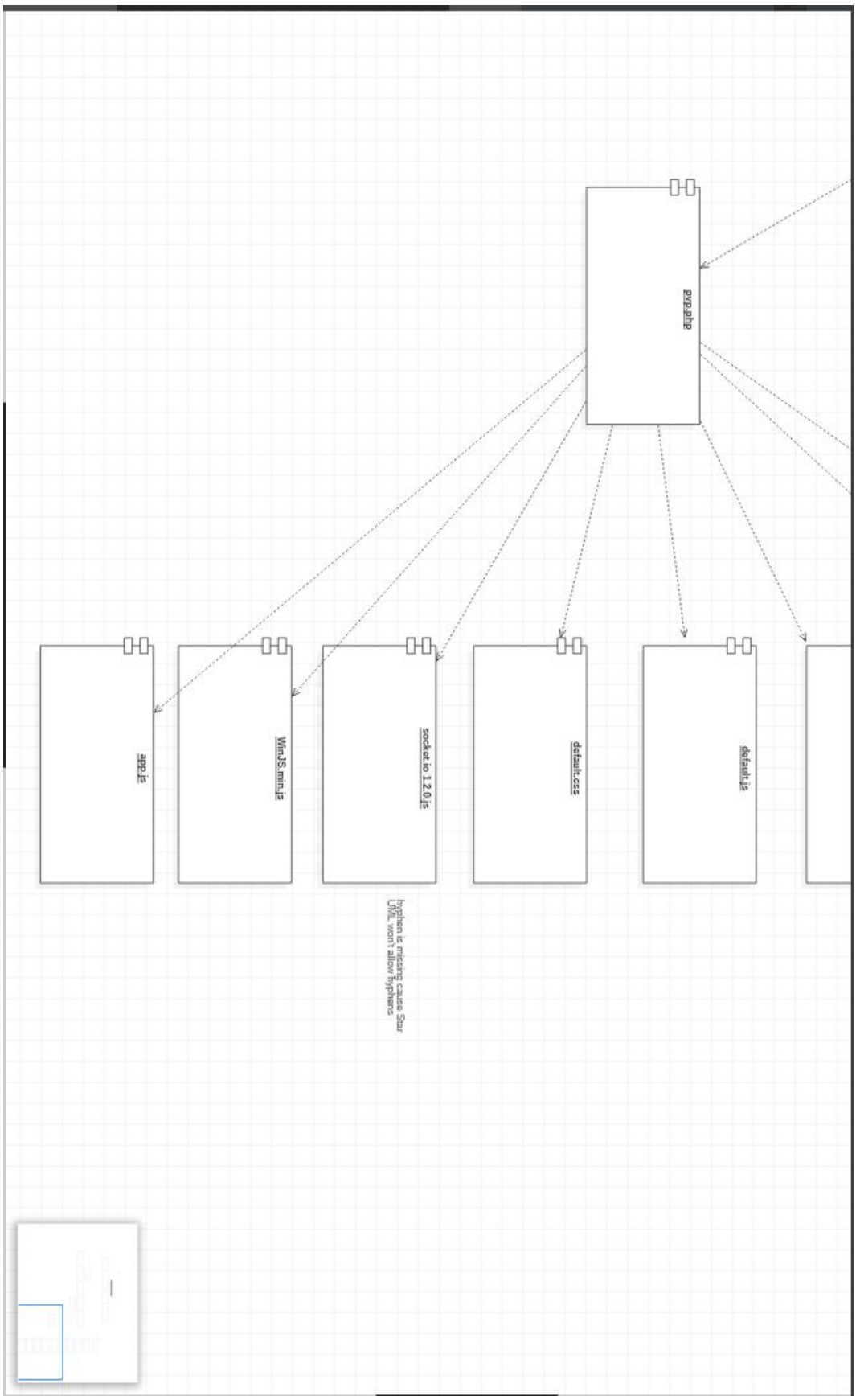


Zoom to the Component Diagram

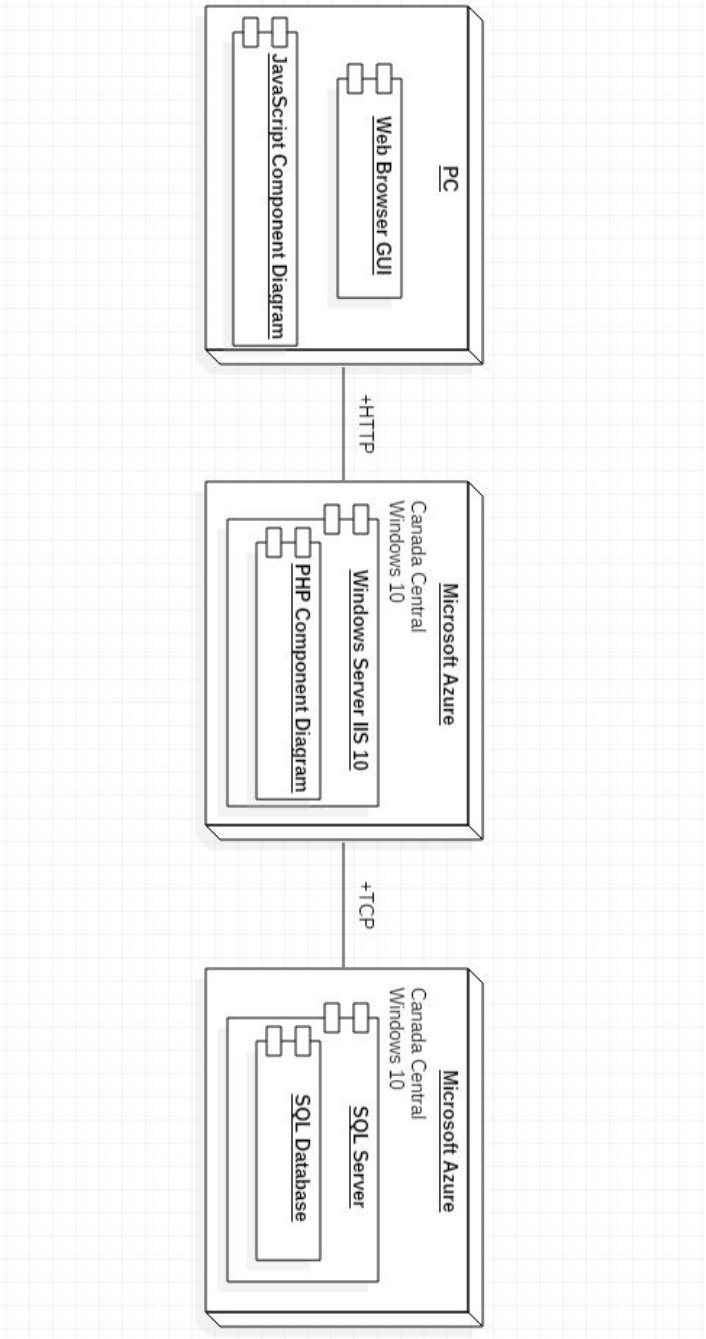



Chess_validatePage.js
Leaderboard.php
Welcome.php
pvp.php
cpu.php
default.js
chess.js
Componentinstance1
jquery-3.2.1.min.js
Chessboard0.3.0.js
ajax


Leaderboard.php
Welcome.php
pvp.php
spu.php
default.js
chess.js
ComponentInstance1
jquery-3.2.1.min.js
hyphen is missing cause Star UML won't allow hyphens
Chessboard0.3.0.js
hyphen is missing cause Star UML won't allow hyphens
mis


pvp.php
app.js
WinJS.min.js
socket.io 1.2.0.js
default.css
default.js
hyphen is missing cause Star
UML won't allow hyphens

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



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

Functions implemented:

- **displayMessage()**
  - exists within ai.js and default.js, used to display popup lightbox message with pertinent information to the user and buttons as necessary

- **displayUpdate(header, message, buttonType)**
  - exists in default.js, used to display a more customized popup lightbox message with multiple options for the types of buttons displayed

- **$('#button1').click(function() {…}**
  - exists in both ai.js and default.js, handles the PHP POST request that increments the appropriate game statistics upon the completion or forfeiting of a game

- **$('#button2').on('click', function() {…}**
  - exists in default.js, handles the event that a player accept a game invite

- **$('#button3').on('click', function() {…}**
  - exists in default.js, handles the event that a player rejects a game invite

- **displayResults(game)**
  - exists in ai.js, displays and updates the appropriate result of the match

- **socket.on('gameAccepted', function(msg) {…}**
  - exists in app.js, used in PvP matches when a player accepts a game match request from another player as server-side handling to signal to the player that sent the invite that their opponent has accepted. Also exists in default.js as client-side handling to initialize the game on acceptance for both players

- **socket.on('gameRejected', function(msg) {…}**
  - exists in app.js, used in PvP matches when a player rejects a game match request, used for server-side handling to signal to the requesting player that their opponent has declined. Also exists in default.js as client-side handling to enter the players back into the lobby and hide the lightbox popup

- **socket.on('opponentLeft', function(msg) {…}**
  - exists in app.js, used in PvP matches when one player forfeits the match, signals to the remaining player that their opponent has left. Also exists in default.js as client-side handling to trigger a PHP POST request that increments the remaining players' win count after their opponent has forfeited

- **selectDifficulty()**
  - exists in cpu.php, handles the selection of the difficulty for the AI

- **promptUserLeave()**
  - exists in cpu.php and pvp.php, displays message to the user in the event they select to forfeit, asking them to confirm they would like to leave

- **cancel()**
  - exists in cpu.php and pvp.php, controls visibility of the “x” button on the displaying popup lightbox, allowing the user to cancel if input is not required or they change their mind. In pvp.php it also sets a flag for whether the match was forfeited

Additionally, custom PHP code was included in all PHP files to handle all queries and session variables to do with the user, in terms of their information and statistics

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

http://chess372.azurewebsites.net/

5. Technical documentation, including:

A. 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.

B. 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 is used with socket.io and PHP for the connection to the database

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

C. 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 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".

Simple-chess-ai: Is a simple chess algorithm with alpha-beta pruning and board evaluation with piece-square tables created by Lauri Hartikka. https://github.com/lhartikk/simple-chess-ai Hartikka does not have a license for this library.

D. 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

Google Chrome: This web browser will be used for the performance testing for the five performance tests.

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

CONSTRAINT_TYPE	CONSTRAINT_NAME	DELETE_ACTION	UPDATE_ACTION	STATUS_ENABLED	STATUS_FOR_REPLICATION	CONSTRAINT_KEYS			
DEFAULT on column aiDraws	DF__user_info__aiDra__4CA06362	(n/a)	(n/a)	(n/a)	(n/a)	((0))			
DEFAULT on column aiGamesPlayed	DF__user_info__aiGam__4D94879B	(n/a)	(n/a)	(n/a)	(n/a)	((0))			
DEFAULT on column aiWins	DF__user_info__aiWin__4BAC3F29	(n/a)	(n/a)	(n/a)	(n/a)	((0))			
DEFAULT on column pvpDraws	DF__user_info__pvpDr__49C3F6B7	(n/a)	(n/a)	(n/a)	(n/a)	((0))			
DEFAULT on column pvpGamesPlayed	DF__user_info__pvpGa__4AB81AF0	(n/a)	(n/a)	(n/a)	(n/a)	((0))			
DEFAULT on column pvpWins	DF__user_info__pvpWi__48CFD27E	(n/a)	(n/a)	(n/a)	(n/a)	((0))			
FIRSTNAME	LASTNAME	PASSWORD	EMAIL	PVPWINS	PVPDRAWS	PVPGAMESPLAYED	AIWINS	AIDRAWS	AIGAMESPLA
dummy1	dummy1	12345678	email@uregina.ca	3	0	21	5	0	10
dummy2	dummy2	12345678	email@uregina.ca	1	10	12	1	3	4
dummy3	dummy3	12345678	email@uregina.ca	0	0	1	10	0	15
ryan	admin	12345678	email1@uregina.(	5	2	11	2	0	7
Melanie	McDougall	welcome4	macdon6m@ureg	0	0	0	0	0	0
Ryan	Leadbetter	12345678	ryan.leadbetter@	0	0	0	0	0	0
Why	SoMean	12345678	test@test.ca	1	0	4	6	0	16
Ryan	Leadbetter	Leadbetter19	leadbetterryan@g	0	0	0	0	0	5
temp	one	12345678	temp@temp.ca	0	0	0	0	0	0
Test	Testing	12345678	test2@test.ca	3	0	3	0	0	1
iyanu	abby	babydivaa01	abbyakinyemii@g	0	0	0	0	0	0
Z	ultimo	12345678	andjks@dummy.(	0	0	2	1	0	1
Jose	Lagrosa	12345	joselagrosa@ureg	0	0	0	1	0	3
Ulises	Oviedo	12345678	ulises.oviedo27@	0	0	0	0	0	0
ChessMaster	ChessMaster	12345678	chess@master.ca	1000001	1	1000003	1000000	0	1000000
Test	Case	12345678	testcase@uregina	1					

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

The project only has 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 wins 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:

A. 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:

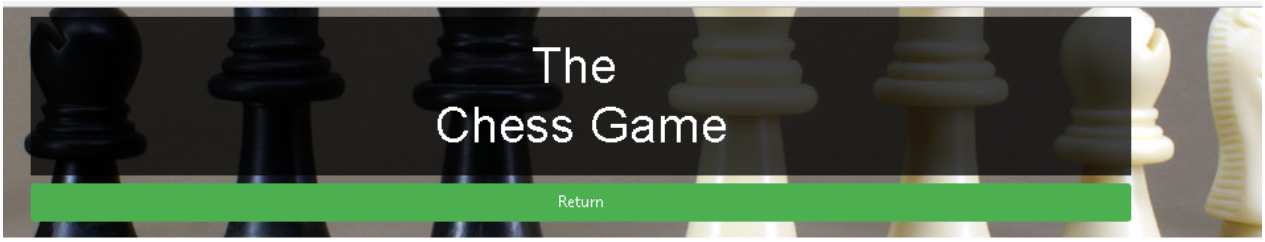
Control number	Action	objective or Explanation	Expected results	Pass	Fail
1	User sign up and login	The user must be able to create an account and be able to enter the game with the same account created	Account correctly created and used to access the game	X	
2	Player enter an online room and play against 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	

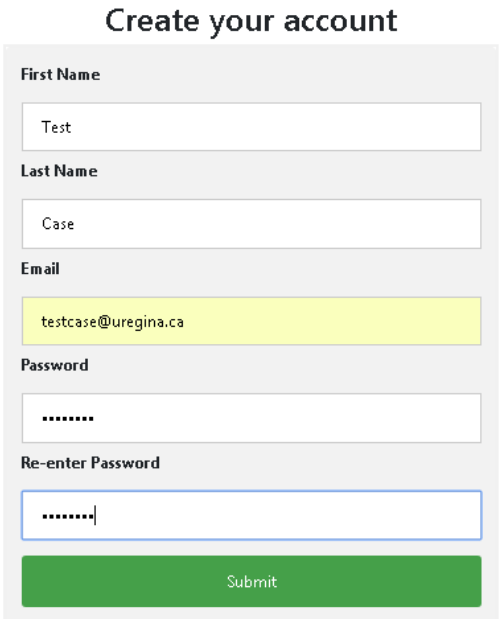
User sign up and login Test case

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

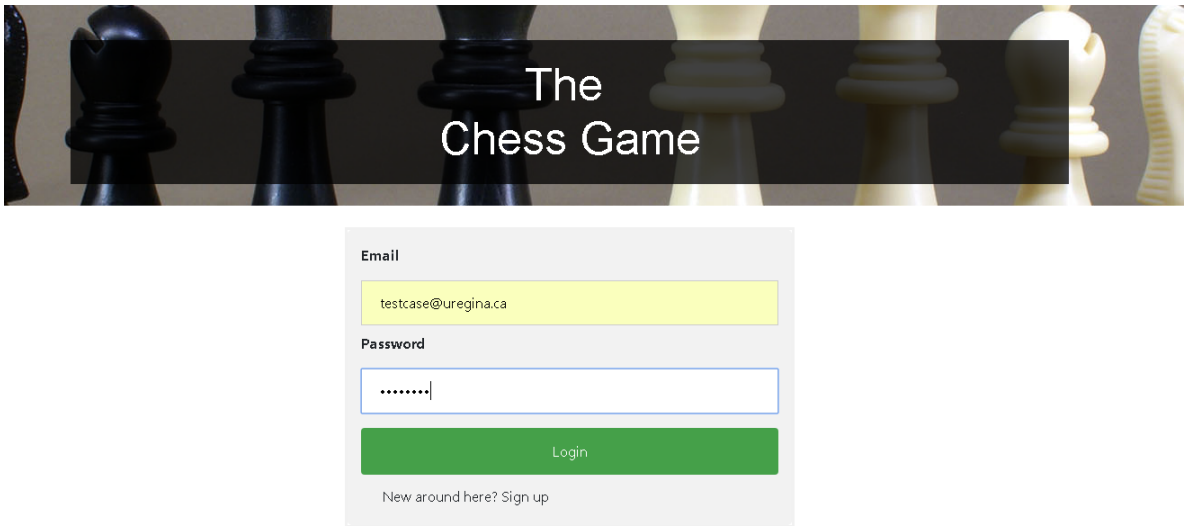
Test steps:

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:
    a. First Name: Test
    b. Last Name: Case
    c. Email: testcase@uregina.ca
    d. Password:12345678
    e. Re-enter Password:12345678

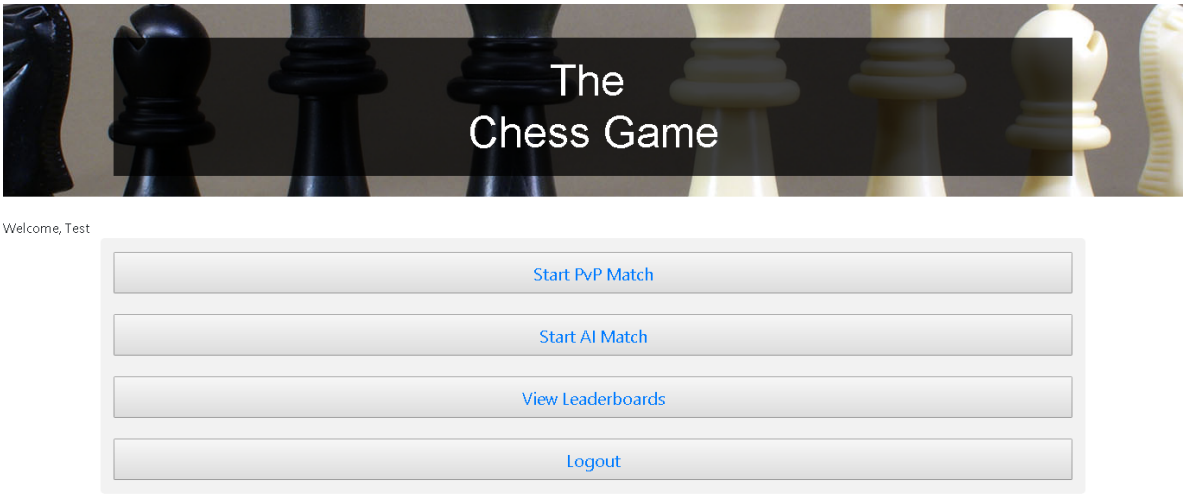




4. Click button submit
5. Enter email and password



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



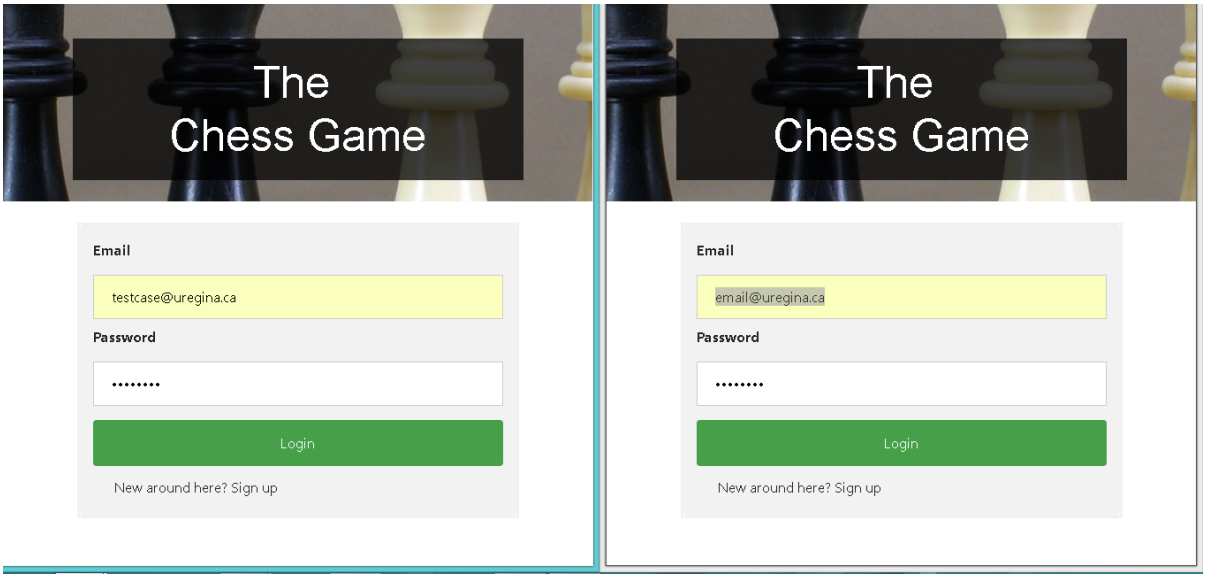
Expected results: Account correctly created and used to access the game

Player enter an online room and play against another player

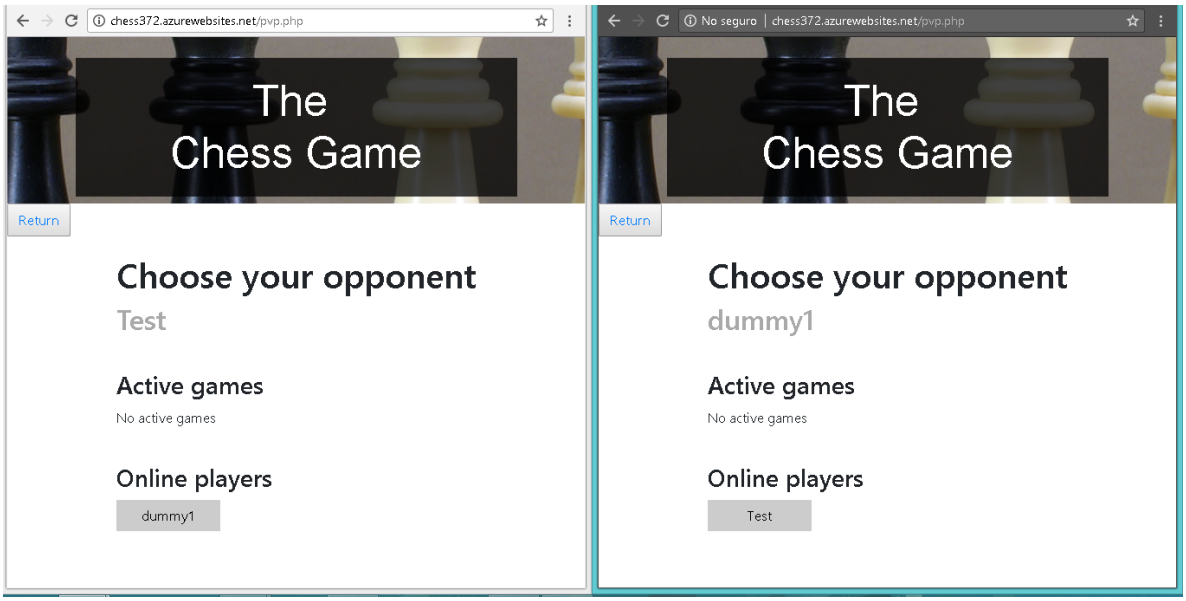
Objective: The player joins online games against other players

Test steps:

1. Login to the webpage: http://chess372.azurewebsites.net/Chess_LoginPage.php
2. open a new tab in another browser and login with another account
    a. First account: testcase@uregina.ca password: 12345678
    b. Second account: email@uregina.ca password:12345678



3. Click Start PvP Match in both Tabs
4. Click in find opponent in both tabs



5. Select in one tab the user from the other tab.
6. Accept the challenge in both Tabs

7. The board will then be see in both tabs:

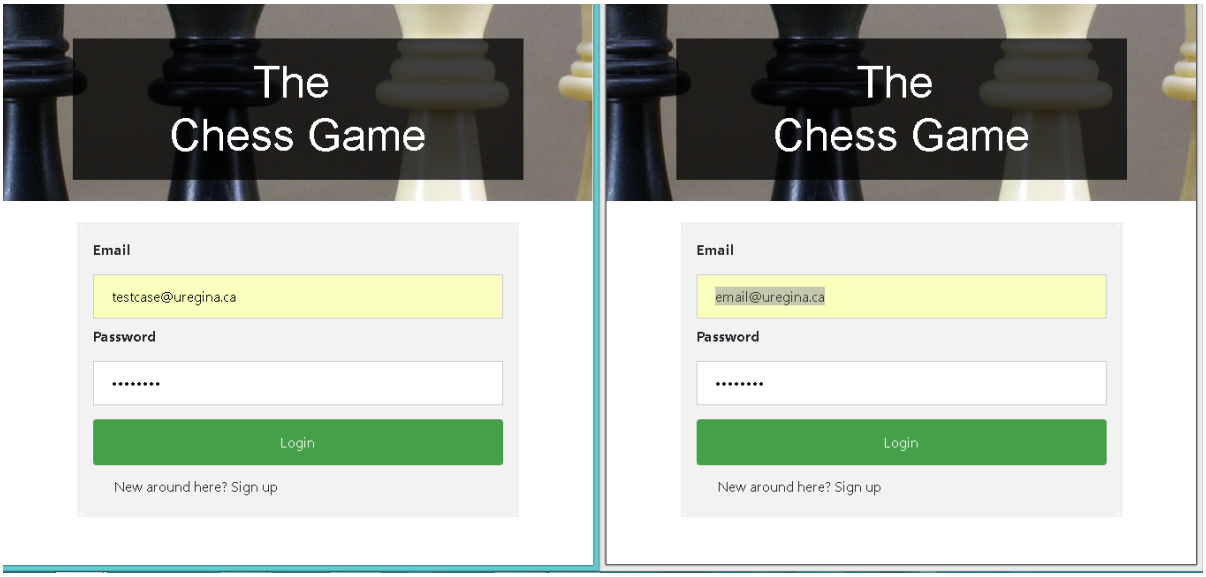
Expected results: online players playing with each other

Real time chess

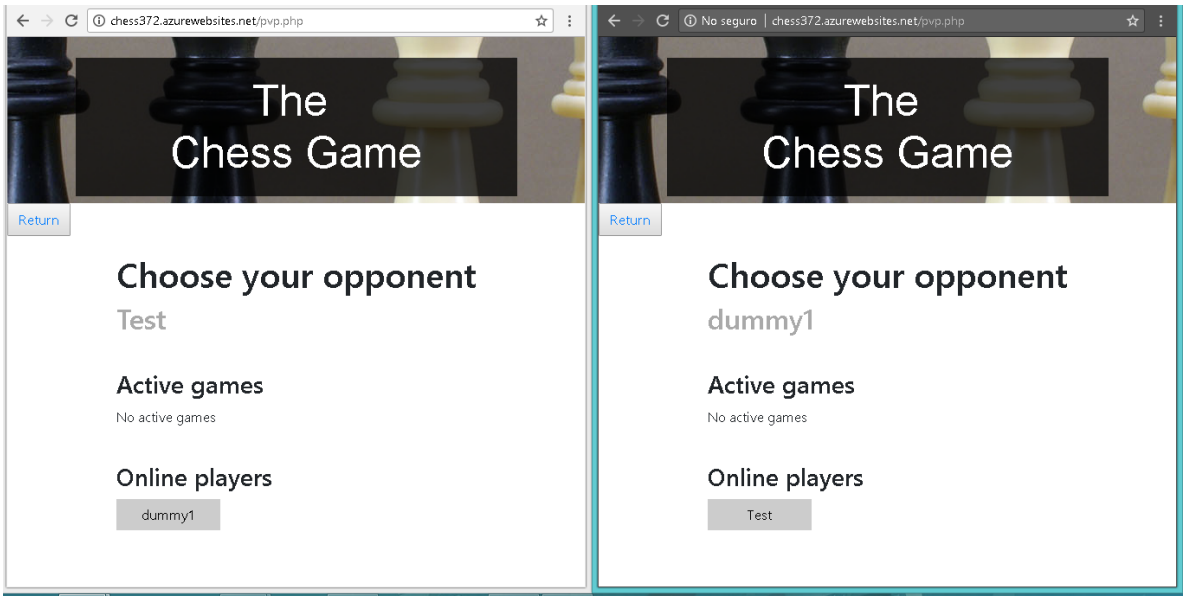
Objective: The movement of the game should be able to be viewed in real time in online games

Test steps:

1. Login to the webpage: http://chess372.azurewebsites.net/Chess_LoginPage.php
2. open a new tab in another browser and login with another account
    a. First account: testcase@uregina.ca password: 12345678
    b. Second account: email@uregina.ca password:12345678



3. Click Start PvP Match in both Tabs
4. Click in find opponent in both tabs



5. Select in one tab the user from the other tab.
6. Accept the challenge in both Tabs

7. Star playing in both tabs and the pieces will move in real time.

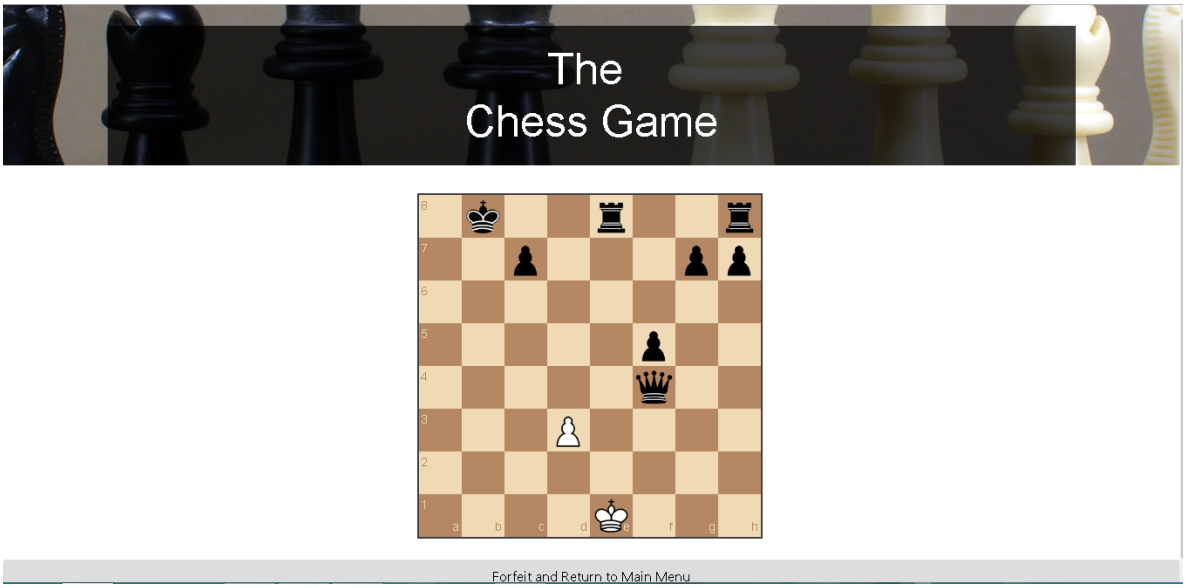
Expected results: chess online with real-time movements

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



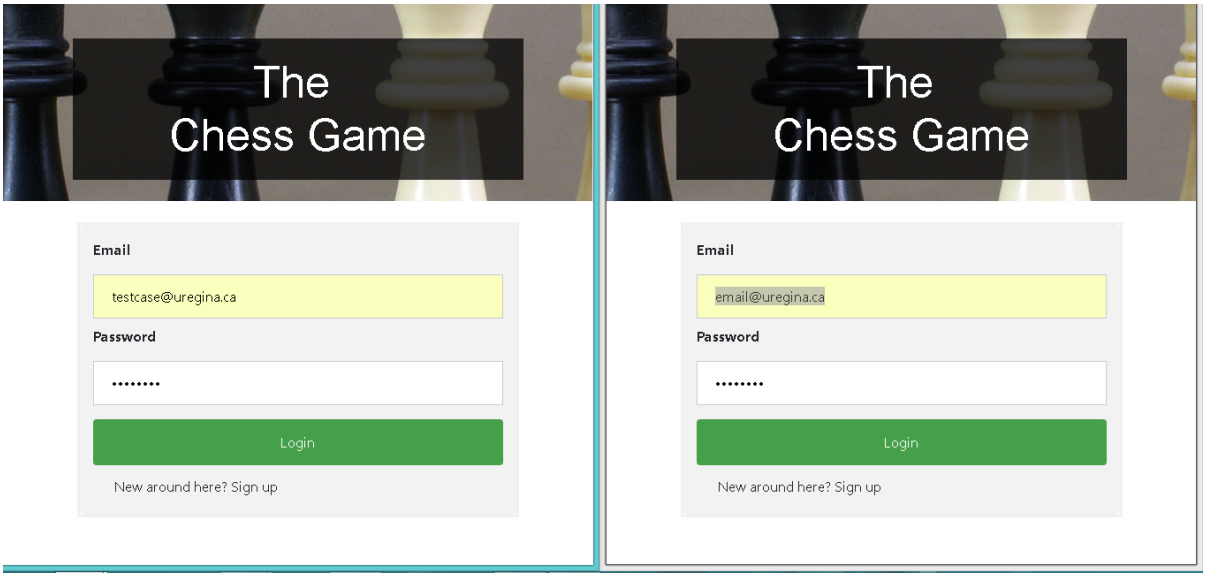
Expected Results: Computer plays against the player according to the difficulty

Leaderboard correctly updates with each win

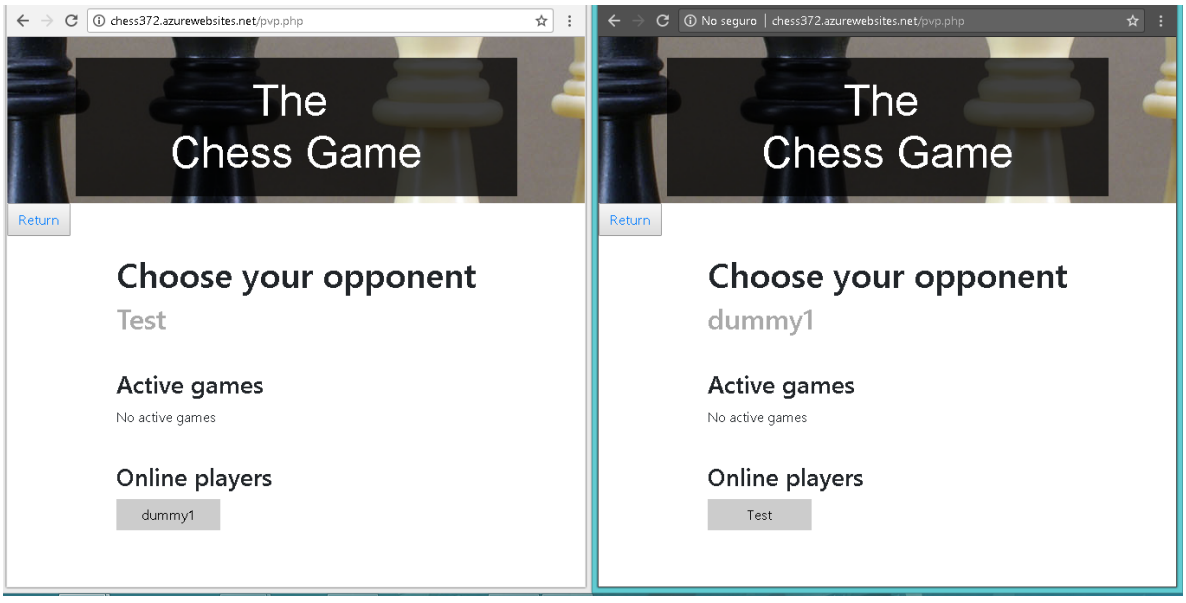
Objective: The movement of the game should be able to be viewed in real time in online games

Test steps:

1. Login to the webpage: http://chess372.azurewebsites.net/Chess_LoginPage.php
2. open a new tab in another browser and login with another account
    a. First account: testcase@uregina.ca password: 12345678
    b. Second account: email@uregina.ca password:12345678

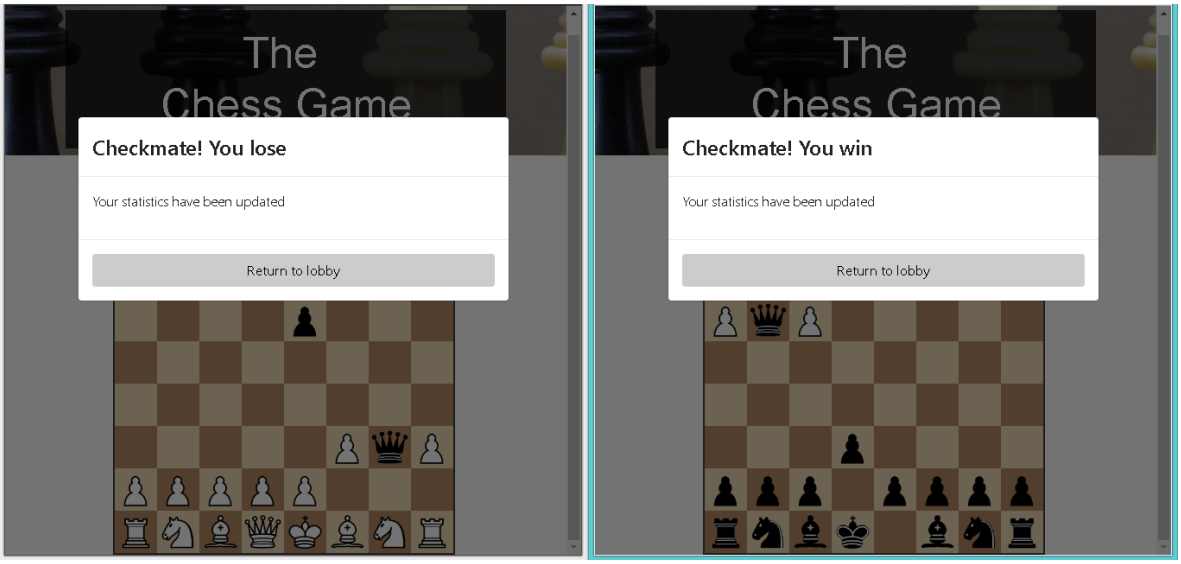


3. Click Start PvP Match in both Tabs
4. Click in find opponent in both tabs



5. Select in one tab the user from the other tab.
6. Select in one tab the user from the other tab.
7. Accept the challenge in both Tabs

8. Star playing in both tabs
9. Win the match with the first user



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

Return

PvP Rankings

Ranking	First Name	Last Name	Wins	Draws	Games Played
1	ryan	admin	5	2	11
2	Test	Testing	3	0	3
3	dummy1	dummy1	3	0	20
4	dummy2	dummy2	1	10	11
5	Why	SoMean	1	0	4
6	Test	Case	1	0	1
7	iyanu	abby	0	0	0
8	temp	one	0	0	0
9	Ryan	Leadbetter	0	0	0
10	Ryan	Leadbetter	0	0	0

Expected results: Leaderboard correctly update

B. 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:

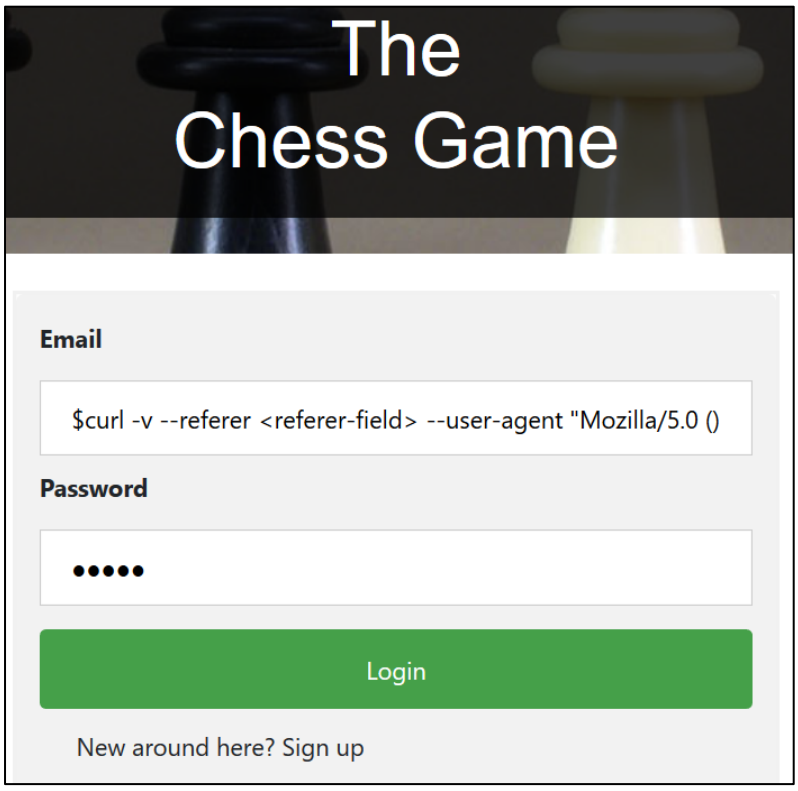
Control number	Action	Objective or Explanation	Expected results	Pass	Fail
1	Form input validation	The user must not be able to insert invalid characters or data. Hence, preventing the injection of malicious codes. Having said that, a range of valid characters are implemented within the system.	User cannot 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

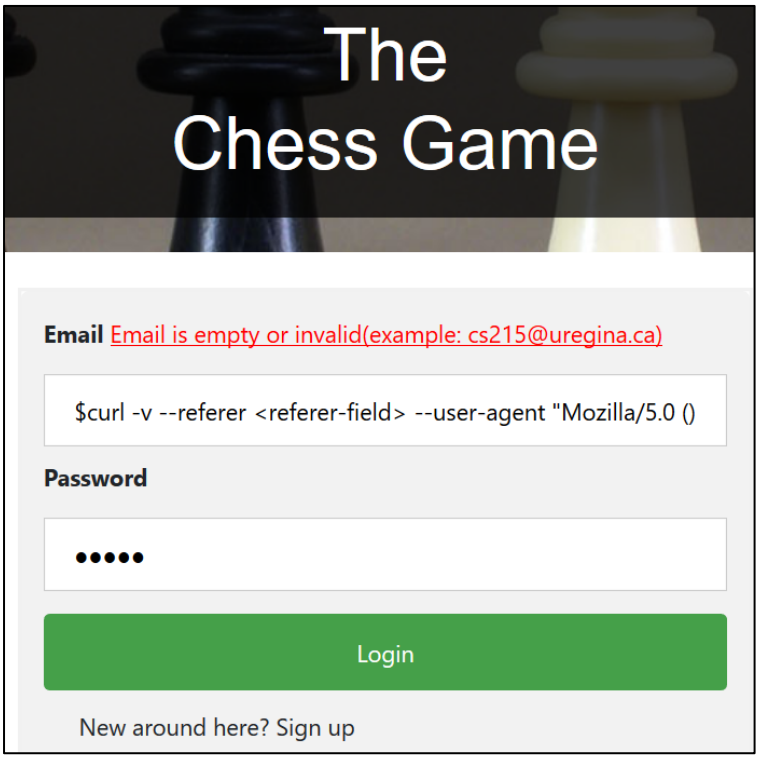
Objective: The user must not be able to insert invalid characters or data.

Test steps:

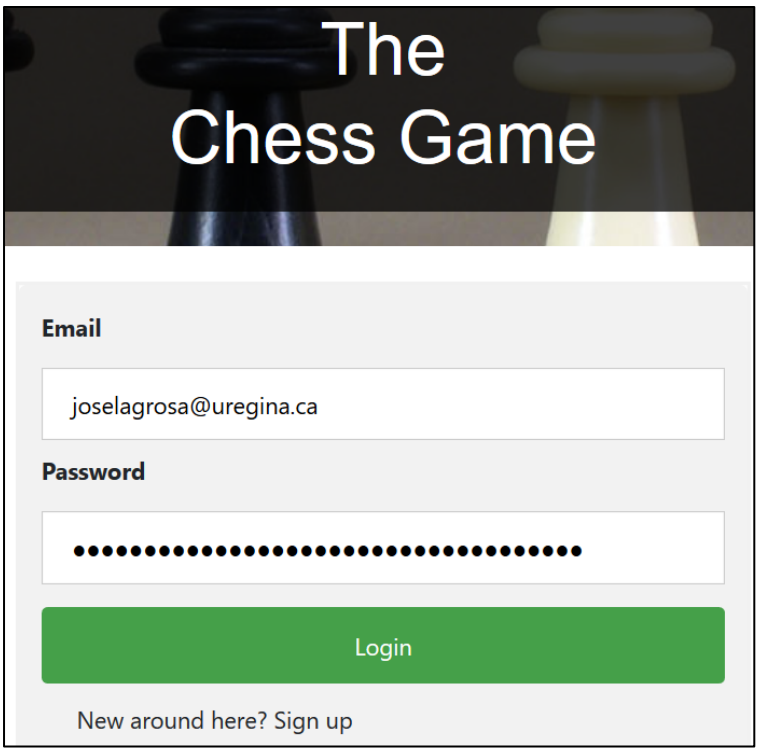
7. Enter the login webpage: http://chess372.azurewebsites.net/Chess_LoginPage.php
8. Fill email with invalid characters but with a valid and registered password:
    a. Email: $curl -v --referer <referer-field> --user-agent "Mozilla/5.0 ()
    b. Password: 12345



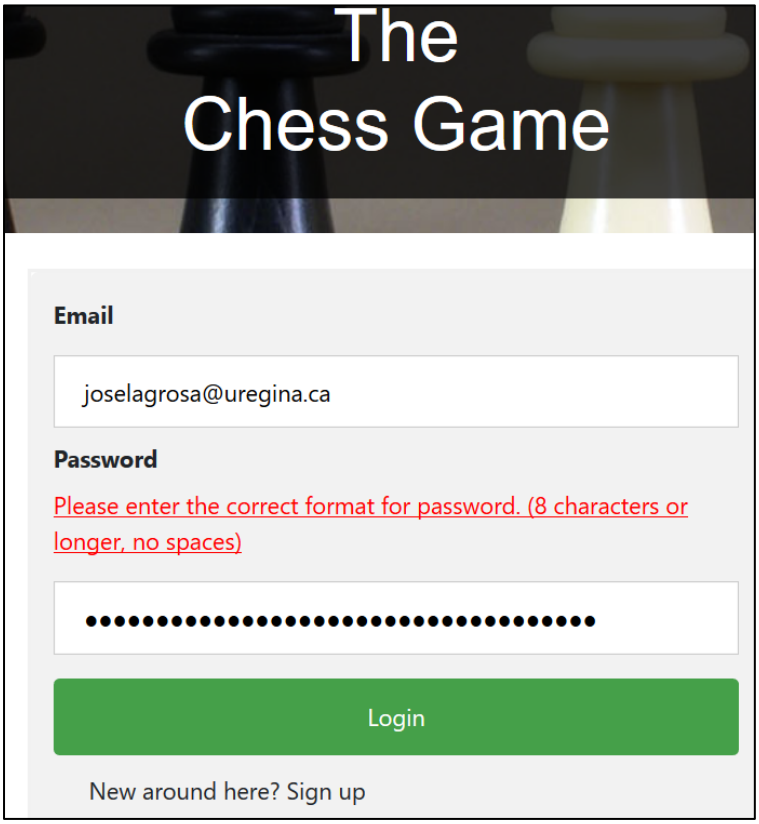
9. Click button submit



10. Fill password with invalid characters but with a valid and registered email:
    a. Email: joselagrosa@uregina.ca
    b. Password: <_type=password>" with "<_type=text>



11. Click button submit



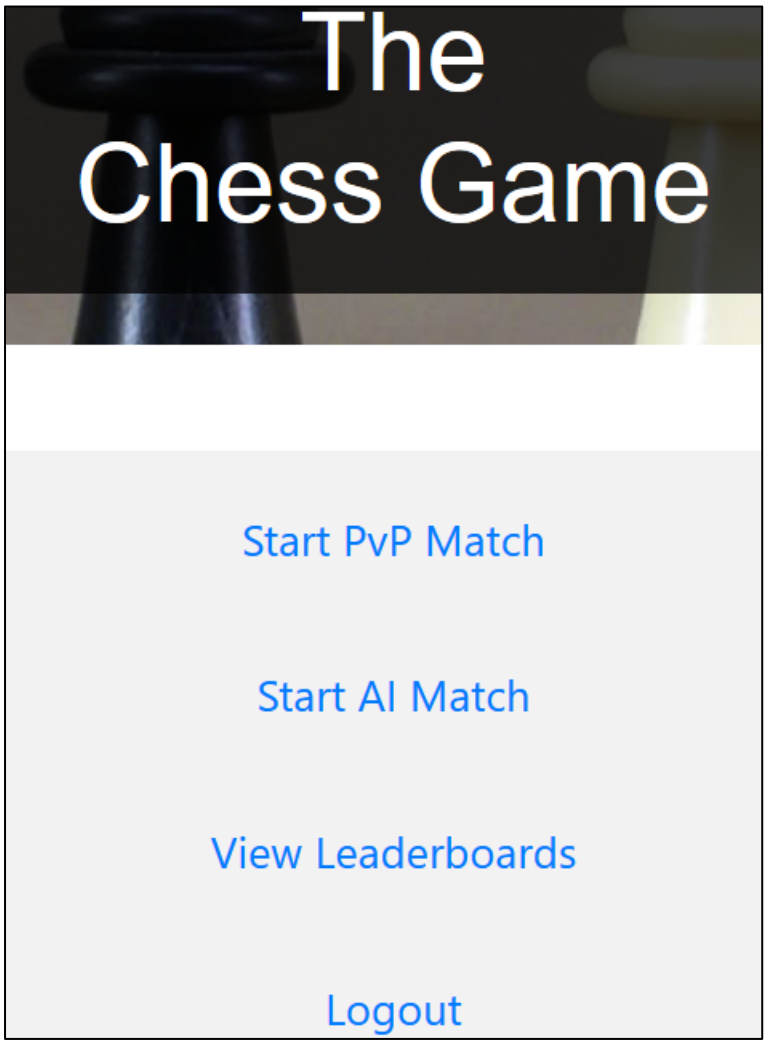
Expected results: User failed to insert invalid characters and was prevented from entering the game.

Reject incorrect moves

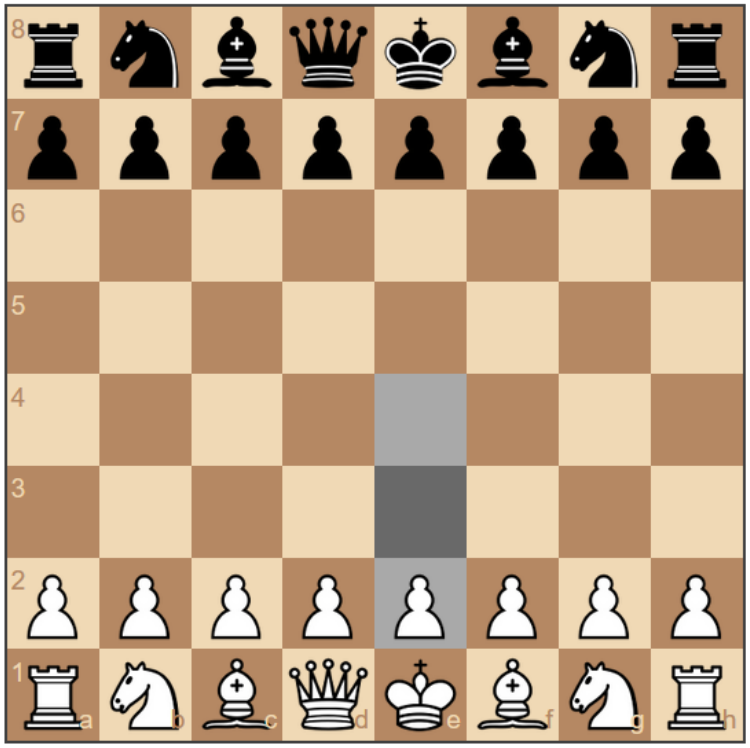
Objective: The user must only be allowed to move chess pieces with legal moves.

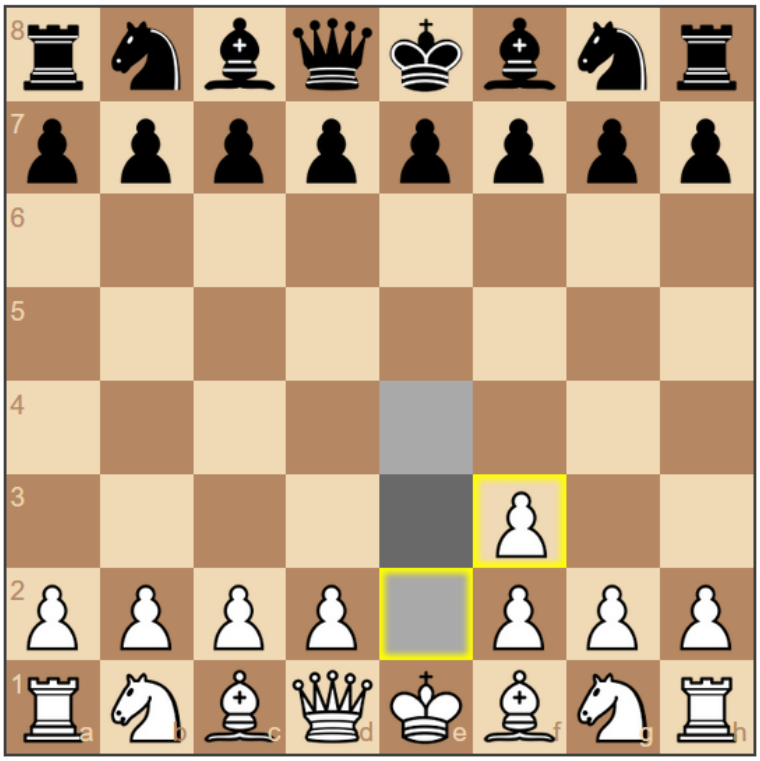
Test steps:

1. Successfully login user



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





Expected results: User unable to execute illegal move.

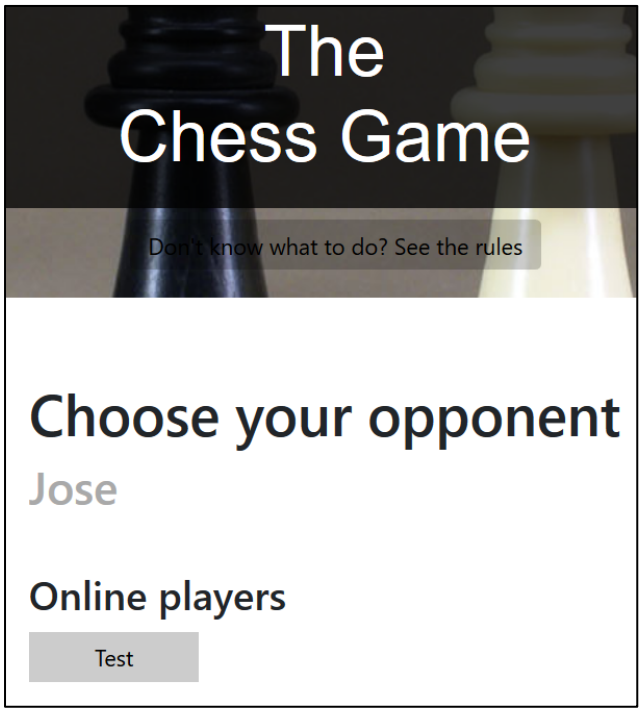
Multiple moves in one turn

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

Test steps:

1. Enter PVP lobby

2. Find opponent and Start a PVP game



3. 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.

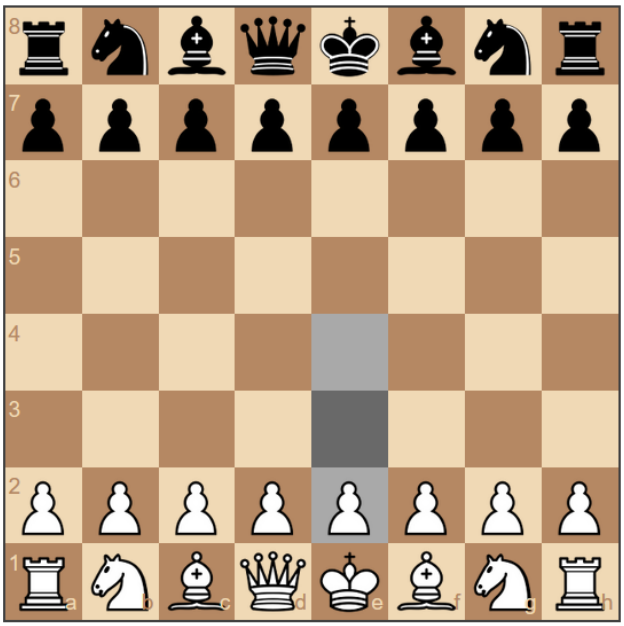
Expected results: Each player will only be allowed one move for each turn.

Player cannot manipulate opponent's chess pieces

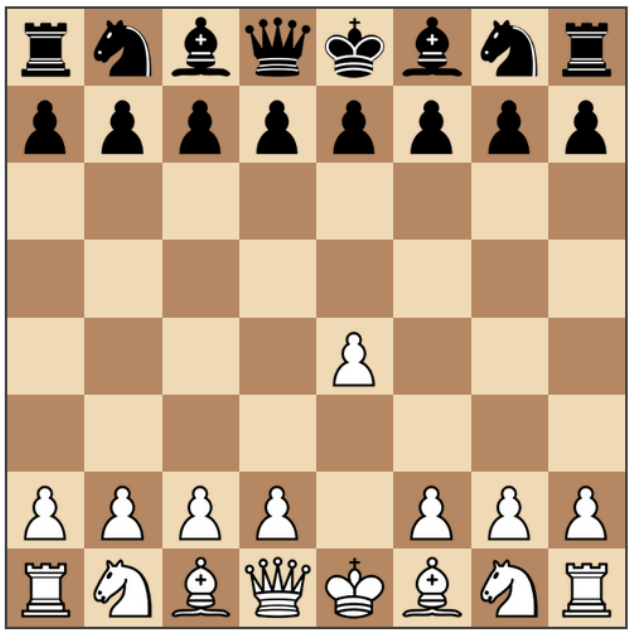
Objective: The user must control the piece with the color designated to him/her.

Test steps:

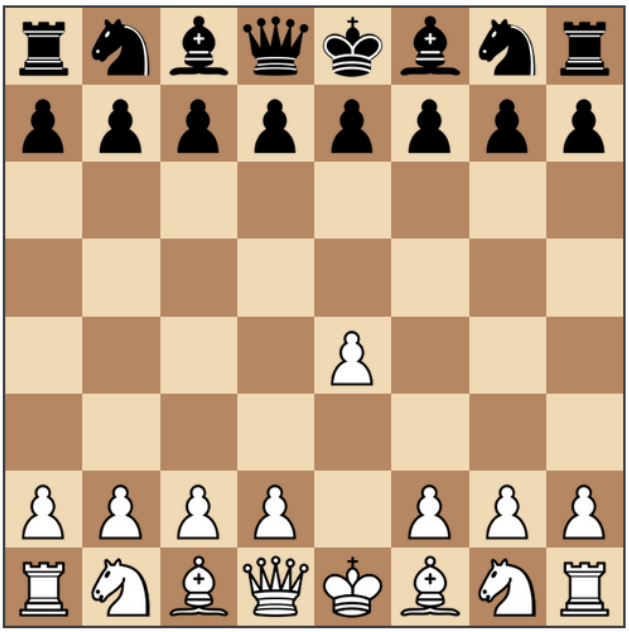
1. Start a game with control over the white pieces



2. Attempt to move a black pawn



3. 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.

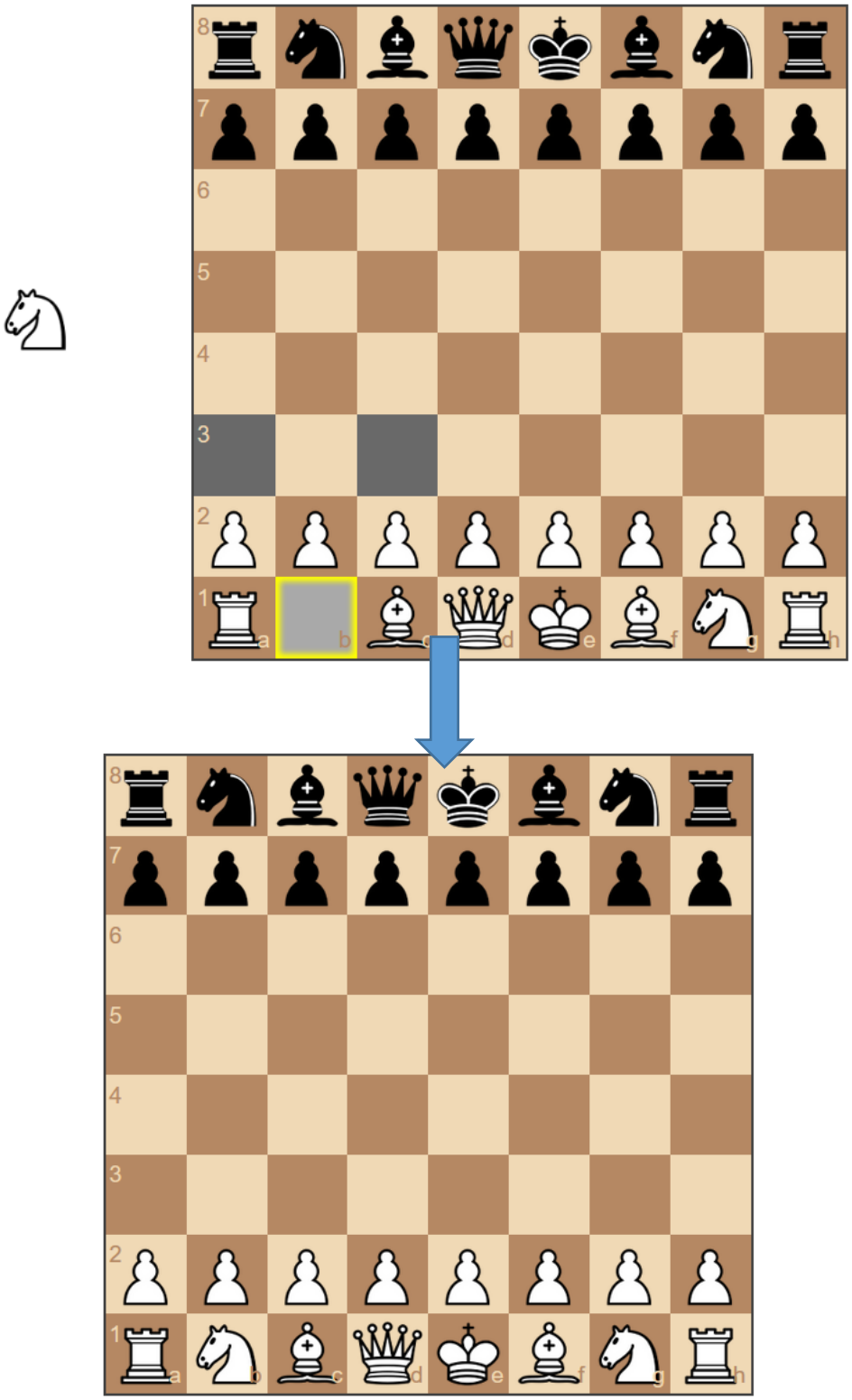
Player cannot manipulate opponent's chess pieces

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

Test steps:

1. Start a game

2. 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.

C. Performance testing of five functions.

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

2) 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

3) 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

4) 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

5) 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.