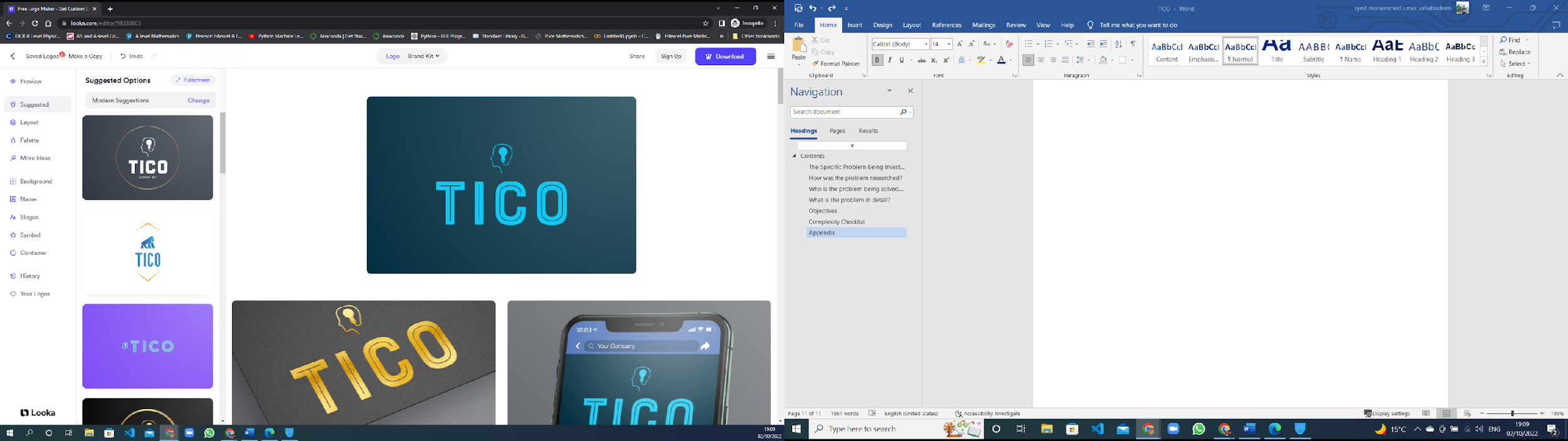


|  |
| --- |
| The Ideal Chess Opponent  text divider |
| Candidate Number: 6139  Centre Number: 52221 |
| TICO  text divider  Candidate name:  *Syed Mohammed Umar Sahabudeen* |



# Contents

Contents

[Contents 2](#_Toc119981890)

[The Specific Problem Being Investigated 4](#_Toc119981891)

[How was the problem researched? 5](#_Toc119981892)

[Who is the problem being solved for? 6](#_Toc119981893)

[What is the problem in detail? 7](#_Toc119981894)

[Objectives: 9](#_Toc119981895)

[Modelling of problem/system 14](#_Toc119981896)

[SQL commands: 16](#_Toc119981897)

[Overall System Design: 17](#_Toc119981898)

[IPSO 18](#_Toc119981899)

[Tensorflow model structure: 20](#_Toc119981900)

[Data dictionary: 21](#_Toc119981901)

[Class diagram: 28](#_Toc119981902)

[Interface screenshots: 29](#_Toc119981903)

[30](#_Toc119981904)

[32](#_Toc119981905)

[33](#_Toc119981906)

[Algorithms: 34](#_Toc119981907)

[Technical solution 38](#_Toc119981909)

[Evidence of build classes: 39](#_Toc119981911)

[Code listings: 42](#_Toc119981912)

[42](#_Toc119981913)

[Testing: 84](#_Toc119981917)

[Static javascript files: 85](#_Toc119981918)

[References: 99](#_Toc119981919)

[Appendix A: 100](#_Toc119981920)

[Appendix B: 101](#_Toc119981921)

## The Specific Problem Being Investigated

The Ideal Chess Opponent is a platform for weak chess players to play against weak chess bots. The point of my project is to build up the confidence of weak or beginner chess players by creating an AI that starts off very weak (*and thus is very easy to beat, but does get better the more games it plays against a specific player*) for the player to play against.

This Ai is what I am calling The Ideal Chess Opponent or TICO. The data to train the model (acting as a heuristic function for the minimax) comes from the games the players play against it and thus the players need to login to the TICO website to play against their specifically trained Ai.

I have attempted to solve this problem by designed an Ai that uses the very same minimax algorithm as the top chess engines but with a convolutional neural network acting as the heuristic function.

My project is a website with a powerful backend doing all the logic processing and getting the chess engines I train to play.

## How was the problem researched?

I choose to make observations, read documents and conduct informal focus groups with my end users to research my problem.

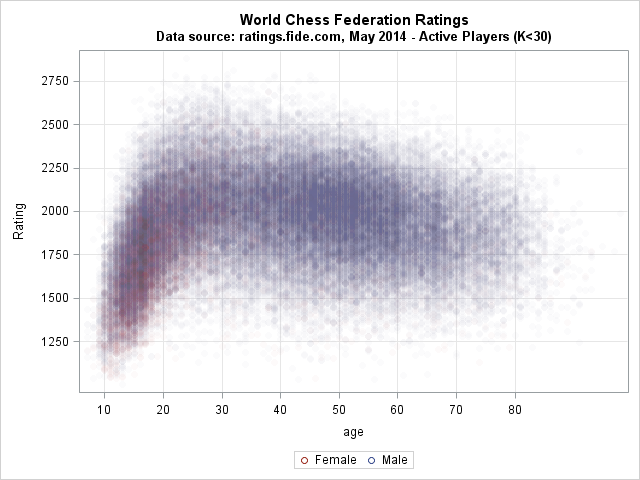
To make the kind of observations I needed: I played chess against players who I knew to be not very confident, or at least less confident than I was. I repeatedly played against them over and over (beating them every time) to see if they would get demotivated. I was trying to see if I could find a common pattern of behavior amongst all the weak chess players I played against.

From my observations: I found that the chess players I played against and defeated, did want to rematch and continue playing for up to a week of constant defeats before they begin not wanting to play against me anymore or talk about the subject of chess in general. I believed these observations were grounds enough to prompt the second stage of my research.

I went and read an article on [chesspulse.com](https://chesspulse.com/are-chess-computers-unbeatable/) [1] that describes how chess engines are designed to be simply too powerful for humans to beat. I also read an article on [blog.rebellionresearch.com](https://blog.rebellionresearch.com/blog/is-ai-the-end-of-chess) [2] which went so far as to say Ai will be the end of chess itself as engines get powerful enough to map every game start to finish. I gathered from these articles that the current chess engines out there were evidently too perfect for my disheartened clients to play against.

So, I decided to trial an earlier version of my project with a weak Ai and have my clients play against it (in informal focus groups). I found that they found it rather amusing and enjoyable to watch a chess engine play chess so terribly and evidently gained some confidence back in their own chess abilities.

In conclusion, I found that my clients (weak chess players) could neither play against existing chess engines or would find it too disheartening to play against many chess players until they came across chess players who play at their level and thus my idea for weak engines to play against came to fruition.



## Who is the problem being solved for?

My clients are my end users. My clients are weak or beginner chess players who are not confident in their own chess abilities. My clients are those players who do not play chess often as they keep constantly getting beaten as they play against me or a perfect engine. My clients also include some of my friends. Typically, the longer a person plays chess: the better they tend to be so I expect my clientele to be between 8 to 25.

## What is the problem in detail?

Beginner or weak players quickly lose confidence when playing against regular chess engines as these have been trained with terabytes of boards and logic and never make any blunders; even at supposedly low difficulty settings. Finding players who are also at the same level is also not convenient for most people.

Whilst it is commonly believed that the more times you do anything; the better you will eventually become at it, in order to want to keep pushing yourself despite almost always losing takes monumental motivation. This motivation cannot be found in people who just causally play chess and is mostly found in those who want to play competitively -> these players are not my target users.

Beginner or weak chess players would still want to feel the dopamine rush of winning a chess game but are not likely to want to have to read chess books on strategy or play hundreds of matches to get easy wins. This is unlikely to happen if they continue to play against other humans who also want to win but have the knowledge and experience to have much higher chance of winning all the matches they play against weaker players.

These beginner/weak players need a neutral opponent designed to make mistakes as they do and does not really care about winning. There do exist chess engines (that technically don’t care about winning) out there but all of them are designed to play perfectly and are nearly impossible to beat. There needs to be such an opponent that is as neutral as existing chess engines but that also makes mistakes like human players.

## Objectives:

1. The board page:

* Any users accessing the website should be directed to a main page when they enter the website’s URL
* The user should have the option to be redirected to the login page or register page or the leaderboards page
* Once a user has logged in, they should now be able to play against their engine
* The chess board should be set up properly:

> The rooks are placed on the outside corners, right and left edge

> The knights are placed immediately inside of the rooks.

> The bishops are placed immediately inside of the knights.

> The queen is placed on the central square of the same color as that of the player: white queen on the white square and black queen on the black square.

> The king takes the vacant spot next to the queen.

> The pawns are placed one square in front of all of the other pieces.

* The user should be able to click on a piece and click on a destination square and the piece should move there if the move is legal
* The currently selected square should be displayed
* The last legal move should be displayed
* The player should be able to perform all the legal moves of chess:
* Moving a piece in accordance with how that piece can move
* Players should be able to castle only if it is a legal
* En passant captures should happen if the player decides to perform an En passant capture
* Players should be prompted by a suitable dialog to promote a pawn if it reaches the eighth rank (white pawn) or first rank (black pawn)
* The player should be notified if the game has been won or lost.

1. The register page:

* Should allow a user to input their first name, last name, email, username and their chosen password should be able to be entered twice
* Should sanitize user inputs before manipulating the users database with input data
* Whether the user has entered appropriate details to register must be determined before their data is added to the database of users
* The user may only be able to register if they have entered a character or a string of characters for their first name, last name, username and password (spaces do not count as a valid character)
* The user may only be able to register if their email address is valid; it contains an “@” symbol at the very least
* A user cannot be registered if the username they wish to register with already exists in the database (that user name has been taken)
* A user cannot be registered if their passwords do not match
* Whenever a user attempts to register but fails to register once they press submit to have their data verified: an appropriate message detailing the nature of the rejection of that registration attempt should be displayed:
* For example, if the passwords do not match: then the message may be something like “Failed to register as passwords do not match!”
* If a user successfully registers then their details need to be inserted into the database of users
* An appropriate successful registration message should be displayed along with the newly registered username if the registration was successful
* If the registration was successful then the must be redirected back to the main page
* A model is created for them and initialized with a lightly trained model

1. The login page:

-Has a link to send an email with password to use to reset password

* Should allow the user to enter their username and password
* Should sanitize user inputs before manipulating the user database with input data
* User’s username and password should be feed into a hashing algorithm to compute the stored value associated with that username
* The user should be notified with appropriate error messages if they get their password wrong or if they have not inputted any data
* The user should be redirected back to the main page with the option of playing the game
* The user should get a suitable message if their login was successful
* A User object should be created

1. User Profile page:

-The user should have the ability to change their password, first name, last name and email if they want to and the database should be updated with these new details

1. The main page:

* Should provide a user the options to click on to the login page, register page or the leaderboard page
* Should be the first page the user sees when they enter the websites URL
* A logged in user should have the options to play the game, see the leaderboard or manage their account details

1. The Django back end:

* Should handle all the get and post requests being sent by the register and login pages
* Should direct users to the appropriate pages they click on
* Should create a single board object that will be manipulated and used for move validation throughout a chess game:
* Should return True for valid moves posted to it or false if not
* Should return Boolean values for the board states such as whether white has castling rights or if there are any legal en passant
* Should allow valid promotions as well

1. The logic behind the move validation and Ai game play:

* A board object should be created using the python chess library
* This board object is what is used to validate moves
* A minimax algorithm with alpha beta pruning is to be used as the algorithm to determine a very good move for the Ai to play
* The minimax algorithm’s heuristic function is going to be a neural network trained with the process described below
* This heuristic function is to be unique for every player as each player will play differently and generate unique boards and scores for the Ai to analyze
* When a player has logged on and is going to play:
* Their corresponding model is loaded (you can think of this like using a default robot body with a simple/empty brain)
* This is like restoring the model’s progress ^
* A function that converts the current board object into a NumPy array of dimensions [14,8,8] should be used to store each board in a form suitable for the heuristic function to analyze
* After every move, the board is to be converted into a 3d NumPy matrix of zeroes and ones
* This board picture ^ is to then be appended to a list of board pictures of moves throughout the current game
* Stockfish should be imported and used to give an approximate score at each board position
* This score is to be appended to list of scores that have the same indexes as their corresponding board pictures in the board pictures list mentioned in the objective before the last objective
* These arrays are to then be converted to NumPy arrays and to be saved in that format
* When a game is over:
* The board pictures .npy file is to be loaded
* The board scores .npy file is to be loaded
* The player’s model is feed the board pictures numpy array and the board scores numpy array and trained to map the board pictures to its corresponding score
* The model is then saved
* The model’s progress is evaluated by testing with the testboards.npy and testscores.npy files
* This score is then modified in the database

1. The leaderboard page:

* Should be able to be viewed by anyone visiting the site
* Should display all the users in the format of something akin to an excel spreadsheet with the data mentioned below
* Should display a player’s username alongside their relative increase in score since starting off with 0%
* These relative scores should be ordered from largest to smallest

## Modelling of problem/system

Login page:

|  |  |
| --- | --- |
| IPSO: | Information: |
| Input | -user name -password |
| Process | Clean input data to remove SQL injections or empty fields |
| Process | Hash the username and password to calculate the hashed password stored for that user name in the database |
| Process | Look up to see if a matching username or password can be found |
| Output | If the user logged in or output an error message |

Register page:

|  |  |
| --- | --- |
| IPSO: | Information: |
| Input | -user name -passwords -first name -last name - email |
| Process | Clean input data to remove SQL injections or empty fields |
| Process | Check if passwords match |
| Process | Check if any existing users exist with that username |
| Store | Store the user’s details if the previous process yields false |
| Output | If the user successfully registered or inputted invalid data or if the username has been taken |

Current engines:

|  |  |
| --- | --- |
| IPSO: | Information: |
| Process | Check whether it is the players turn to play or the engine’s turn |
| Process | If it is the engines turn then a get request is sent to the server to get the engines move |
| Input | The player enters their move by clicking the piece they want to move and where they want to move it on the board |
| Process | The players move is sent via a post request to the server |
| Process | The server validates the move and sends a Boolean true or false |
| Output | If the previous process was true then the board is updated with the player’s move |
| Process | Check if game is over |
| Output | Whether the player won or the engine or if it was a draw |

Design:

Flat file database called users.db

This is the single table inside called “UserInfo”:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column: | Datatype: | Example: | Use: | Validation: |
| Username | string | Rock | To uniquely identify users | Cannot be the same as another users |
| Password | string | 8732726372 | A hashed value calculated from the user’s input | Needs to not be null |
| FirstName | string | Jhon | To address the user in  an email | Not null |
| LastName | string | smith | To address the user in an email | Not null |
| Email | string | example@gmail.com | To send the user a recovery password | Needs to have an @ symbol |
| relative | float | 0.000723726 | A measure of the accuracy of a model | none |

## SQL commands:

SQL command to create the UserInfo table:



SQL command to insert a new record into the UserInfo table:



SQL command to return all users with the username: Username:



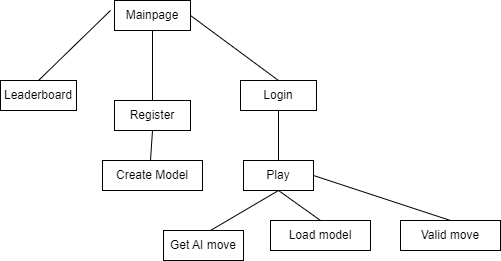
SQL command to select a record based on a username:



SQL command to return the relative progresses with the greatest values first:



# Overall System Design:



## IPSO

|  |  |  |
| --- | --- | --- |
| IPSO | Program Section | Item |
| Input | Logging in | Username  password |
|  | registering | Username, Password, first name, last name, email, repeat password again |
|  | Playing against AI | Player’s move |
|  | Leader board | Press link |
|  | Managing account | New password  New first name  New last name  New email  New password  Repeat new password |
| Processing | Logging in | See if username and password match with existing username and password in database |
|  | registering | See if the entered username has been taken and if not update the UserInfo table with entered details |
|  | Playing against AI | AI analyses the board and makes its next move |
|  | Leader board | The leaderboard table is returned |
|  | Managing account | If the passwords entered match: then details updated in table |
| storage | Saving the model, saving the board snaps and scores | All board positions and board scores are saved in separate files. The model is also saved. |
| output | On screen moving for player | If valid move then piece will move on the screen |
|  | On screen moving for AI | The piece will move on the screen |
|  | Successful login, register | A message displaying a success login or register is displayed |
|  | Successful details update | A message saying details for player with username has been updated successfully. |
|  |  |  |
|  |  |  |

## Tensorflow model structure:

## Data dictionary:

|  |  |  |  |
| --- | --- | --- | --- |
| Data item | Data type | Validation | Sample data |
| move | string |  | “e2e4” |
| targetsquare | String |  | “e2” |
| currentsquare | string |  | “e4” |
| Boardsnaps | 4 d array |  | [[[[0 0 0 0 0 0 0 0]    [1 1 1 1 1 1 1 1]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [1 1 1 1 1 1 1 1]    [0 0 0 0 0 0 0 0]]   [[0 1 0 0 0 0 1 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 1 0 0 0 0 1 0]]   [[0 0 1 0 0 1 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 1 0 0 1 0 0]]   [[1 0 0 0 0 0 0 1]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [1 0 0 0 0 0 0 1]]   [[0 0 0 1 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 1 0 0 0 0]]   [[0 0 0 0 1 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 1 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]], [[[0 0 0 0 0 0 0 0]    [1 1 1 1 1 1 1 1]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [1 1 1 1 1 1 1 1]    [0 0 0 0 0 0 0 0]]   [[0 1 0 0 0 0 1 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 1 0 0 0 0 1 0]]   [[0 0 1 0 0 1 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 1 0 0 1 0 0]]   [[1 0 0 0 0 0 0 1]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [1 0 0 0 0 0 0 1]]   [[0 0 0 1 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 1 0 0 0 0]]   [[0 0 0 0 1 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 1 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]   [[0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]    [0 0 0 0 0 0 0 0]]]  ] |
| boardscores | 1 d array |  | [32,-67,45,900] |
| letterToCoordinate | dictionary |  |  |
| model | Tensor flow model |  |  |
| board | Chess.Board() this is an instance of chess.Board() class. |  | board |

## Class diagram:



## Interface screenshots:

Mainpage:



## 

## 

## 

# Algorithms:

## 

^The above minimax algorithm is used to find the best possible move for the AI assuming that the player is also playing the best that they can.



^The above algorithm is used to hash the user’s entered password



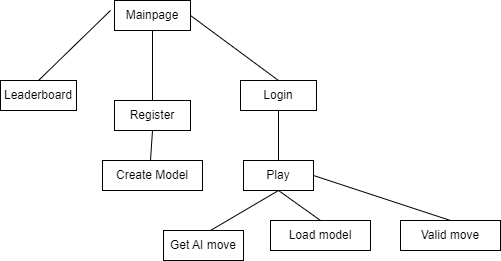
^The convert to AI board function is what turns a regular board with pieces on it, into an array of ones and zeroes that the AI can analyze.

## Technical solution

## 

^Evidence of creating a flat file database with one table UserInfo inside

System design:



## Evidence of build classes:

class Piece {

    constructor(name, color,value,symbol) {

      this.name = name;

      this.color = color;

      this.value = value;

      this.symbol = symbol;

    }

    getSymbol(){

        return this.symbol;

    }

    getName(){

        return this.name;

    }

    getColor(){

        return this.color;

    }

    getValue(){

        return this.value;

    }

  }

class Pawn extends Piece {

constructor(color,symbol) {

    super("pawn", color , 10, symbol);

}

}

class Rook extends Piece {

constructor(color,symbol) {

    super("rook", color , 50, symbol);

}

}

class Knight extends Piece {

constructor(color,symbol) {

    super("knight", color , 30, symbol);

}

}

class Bishop extends Piece {

constructor(color,symbol) {

    super("bishop", color , 30, symbol);

}

}

class King extends Piece {

constructor(color,symbol) {

    super("king", color , 900, symbol);

}

}

class Queen extends Piece {

constructor(color,symbol) {

    super("queen", color , 90, symbol);

}

}

^Javascript classes

class User:

    def \_\_init\_\_(self,Username, Password,Firstname, Lastname, Email, relative):

        self.Username = Username

        self.Password = Password

        self.Firstname = Firstname

        self.Lastname = Lastname

        self.Email = Email

        self.relative = relative

    def getUsername(self):

        return self.Username

    def getFirstname(self):

        return self.Firstname

    def getLastname(self):

        return self.Lastname

    def getEmail(self):

        return self.Email

    def getrelative(self):

        return self.relative

    def resetPassword(self):

        conn = sqlite3.connect('users.db')

        c = conn.cursor()

        words = ["Chess", "Board", "Pawn", "Knights", "king"]

        code = random.choice(words)

        count = 0

        for i in code:

            count+= ord(i)

        tempPass = self.Password - count

        with conn:

            c.execute("UPDATE UserInfo SET Password = (:tempPass)  WHERE Username = (:Username)", {'Username': self.Username, 'tempPass': tempPass})

    def updateRelative(self, newRelative):

        conn = sqlite3.connect('users.db')

        c = conn.cursor()

        with conn:

            c.execute("""UPDATE UserInfo SET relative = (:Relative) WHERE Username = (:Username)""",

            {'Username': self.Username, 'Relative':newRelative})

    def updateDetails(self,Password,Firstname, Lastname, Email):

        conn = sqlite3.connect('users.db')

        c = conn.cursor()

        with conn:

            c.execute("""UPDATE UserInfo SET Password = (:Password),

            Firstname = (:Firstname), Lastname = (:Lastname),

            Email = (:Email)  WHERE Username = (:Username)""",

            {'Username': self.Username, 'Password': Password, 'Firstname': Firstname, 'Lastname': Lastname, 'Email': Email})

^Python class

## Code listings:

## 

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>MainPage</title>

  </head>

  <style>

    a {

      display: block;

      margin-top: auto;

      margin-left: auto;

      margin-right: auto;

      background-color: rgb(13, 28, 232);

      height: 15%;

      width: 50%;

      margin-top: 1%;

      font-size: 100px;

      text-align: center;

    }

    .messages {

      cursor: pointer;

      float: left;

      width: 100%;

      height: 160px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      display: table-cell;

      vertical-align: middle;

    }

  </style>

  <body>

    {% if Username %}

    <h1>You are logged in as: {{Username}}</h1>

    {% endif %}

    <a type="a" href="/templates/profile.html">My Account</a>

    <a type="a" href="/templates/board.html">Play!</a>

    <a type="a" href="/templates/login.html">Login!</a>

    <a type="a" href="/templates/register.html">Register!</a>

    <a type="a" href="/templates/leaderboard.html">Leaderboard!</a>

    <div class="messages">

      {% if messages %}

        {% for message in messages %}

          <div class="alert alert-{{ message.tags }}">

            {{ message }}

          </div>

        {% endfor %}

      {% endif %}

    </div>

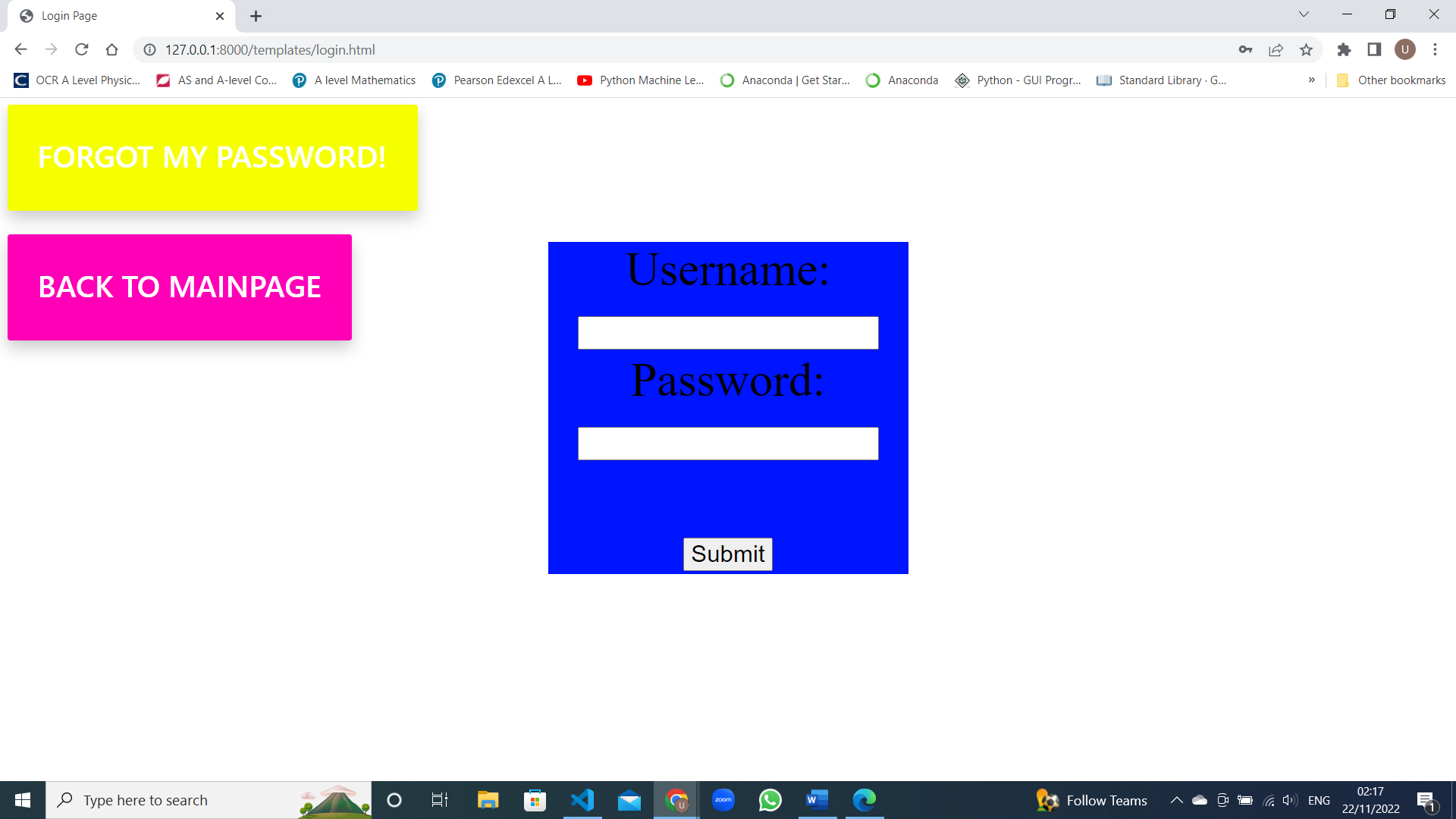
    <script></script>

  </body>

</html>

## 





<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Login Page</title>

  </head>

  <style>

    form {

      display: block;

      margin-top: auto;

      margin-left: auto;

      margin-right: auto;

      background-color: rgb(0, 21, 255);

      height: 20%;

      width: 25%;

      margin-top: 10%;

      font-size: 50px;

      text-align: center;

    }

    input {

      font-size: 25px;

    }

    .messages {

      cursor: pointer;

      float: left;

      width: 100%;

      height: 160px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      display: table-cell;

      vertical-align: middle;

    }

    .forgottenlink{

    left: 0.5%;

    top: 1%;

    color: rgb(255, 255, 255);

    font-family: "Inter UI", "SF Pro Display", -apple-system, BlinkMacSystemFont,

    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",

    sans-serif;

    font-weight: 500;

    padding: 1em 1em 1.5em;

    text-decoration: none;

    text-transform: uppercase;

    cursor: pointer;

    display: block;

    justify-content: center;

    position: absolute;

    border-color: black;

    border-radius: 3px;

    font-size: xx-large;

    background-color: #f6ff00;

    min-width: 160px;

    height: 2rem;

    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

    z-index: 1;

    }

    .mainpagelink{

    left: 0.5%;

    top: 20%;

    color: rgb(255, 255, 255);

    font-family: "Inter UI", "SF Pro Display", -apple-system, BlinkMacSystemFont,

    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",

    sans-serif;

    font-weight: 500;

    padding: 1em 1em 1.5em;

    text-decoration: none;

    text-transform: uppercase;

    cursor: pointer;

    display: block;

    justify-content: center;

    position: absolute;

    border-color: black;

    border-radius: 3px;

    font-size: xx-large;

    background-color: #ff00b7;

    min-width: 160px;

    height: 2rem;

    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

    z-index: 1;

    }

  </style>

  <body>

    <a type="a" class= "mainpagelink" href="mainpage.html">Back to Mainpage</a>

    <a type="a" class = "forgottenlink" href="/templates/forgotten.html"  onclick="alert('Check your email for instructions!')">Forgot my password!</a>

    <form action="authenticate" method="post">

      {% csrf\_token %}

      <label class="big" for="Username">Username:</label><br />

      <input type="text" id="Username" name="Username" value="" /><br />

      <label for="Password">Password:</label><br />

      <input class="big" type="password" id="pass" name="Password" value="" /><br /><br />

      <input type="submit" value="Submit" text = "Login"/>

    </form>

    <div class="messages">

      {% if messages %}

        {% for message in messages %}

          <div class="alert alert-{{ message.tags }}" role="alert">

            {{ message }}

          </div>

        {% endfor %}

      {% endif %}

    </div>

  </body>

</html>





<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Register Page</title>

  </head>

  <style>

    form {

      display: block;

      margin-top: auto;

      margin-left: auto;

      margin-right: auto;

      background-color: rgb(25, 0, 255);

      height: 20%;

      width: 25%;

      margin-top: 10%;

      font-size: 50px;

      text-align: center;

    }

    input {

      font-size: 25px;

    }

    .messages {

      cursor: pointer;

      float: left;

      width: 100%;

      height: 160px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      display: table-cell;

      vertical-align: middle;

    }

    .mainpagelink{

    left: 0.5%;

    top: 20%;

    color: rgb(255, 255, 255);

    font-family: "Inter UI", "SF Pro Display", -apple-system, BlinkMacSystemFont,

    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",

    sans-serif;

    font-weight: 500;

    padding: 1em 1em 1.5em;

    text-decoration: none;

    text-transform: uppercase;

    cursor: pointer;

    display: block;

    justify-content: center;

    position: absolute;

    border-color: black;

    border-radius: 3px;

    font-size: xx-large;

    background-color: #ff00b7;

    min-width: 160px;

    height: 2rem;

    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

    z-index: 1;

    }

  </style>

  <body>

    <a type="a" class= "mainpagelink" href="mainpage.html">Back to Mainpage</a>

    <div class="messages">

      {% if messages %}

        {% for message in messages %}

          <div class="alert alert-{{ message.tags }}" role="alert">

            {{ message }}

          </div>

        {% endfor %}

      {% endif %}

    </div>

    <form action="addUser" method="post">

      {% csrf\_token %}

      <label for="FirstName">First name:</label><br />

      <input type="text" id="FirstName" name="FirstName" value="" /><br />

      <label for="LastName">Last name:</label><br />

      <input type="text" id="LastName" name="LastName" value="" /><br />

      <label for="Email">Email:</label><br />

      <input type="text" id="Email" name="Email" value="" /><br />

      <label for="Username">Username:</label><br />

      <input type="text" id="Username" name="Username" value="" /><br />

      <label for="Password">Password:</label><br />

      <input type="password" id="pass" name="Password" value="" /><br /><br />

      <label for="PasswordRepeat">Repeat Password:</label><br />

      <input type="password" id="pass" name="PasswordRepeat" value="" /><br /><br />

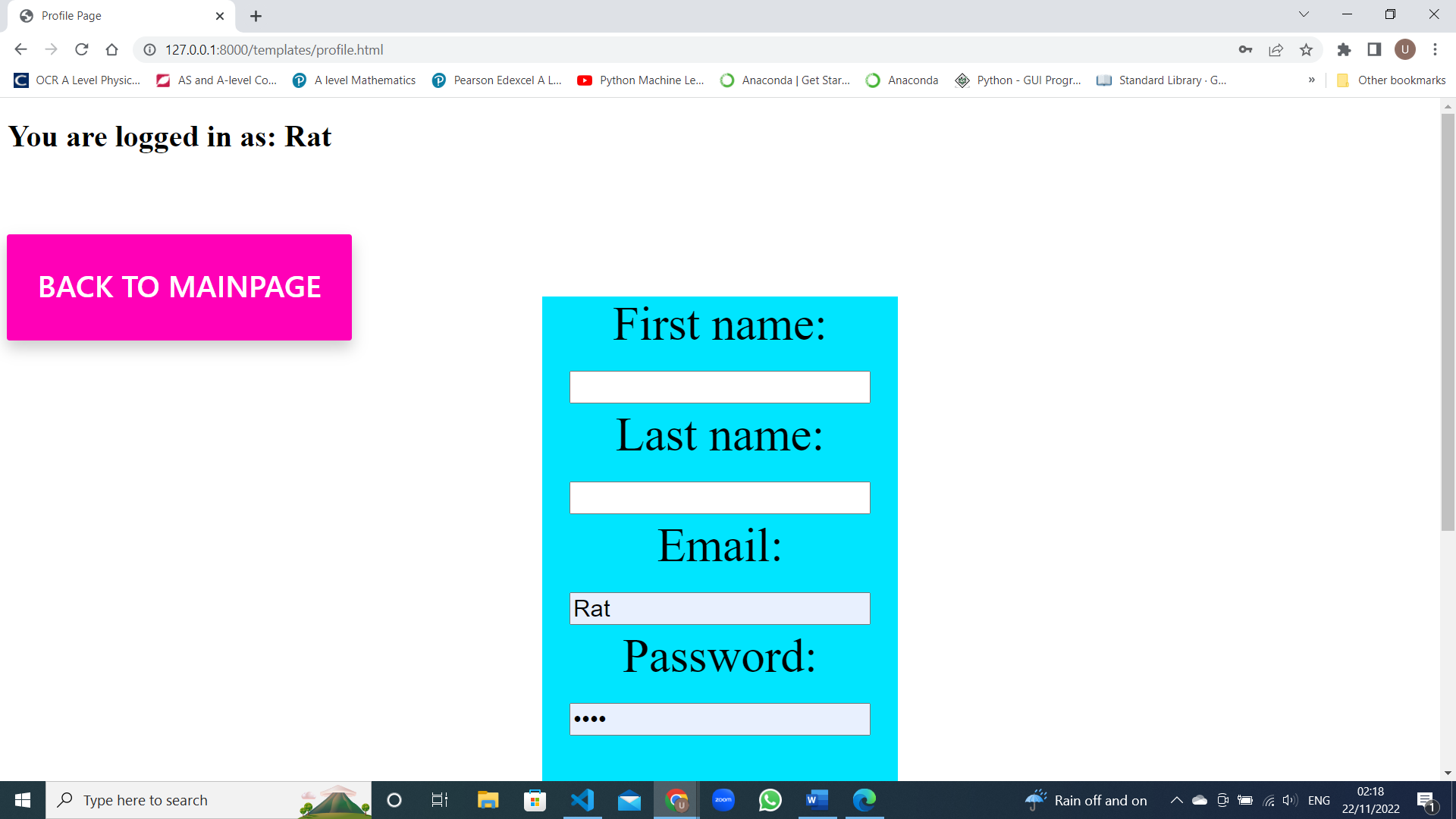
      <input type="submit" value="Submit" />

    </form>

  </body>

</html>

## 



<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Profile Page</title>

  </head>

  <style>

    form {

      display: block;

      margin-top: auto;

      margin-left: auto;

      margin-right: auto;

      background-color: rgb(0, 229, 255);

      height: 20%;

      width: 25%;

      margin-top: 10%;

      font-size: 50px;

      text-align: center;

    }

    input {

      font-size: 25px;

    }

    .messages {

      cursor: pointer;

      float: left;

      width: 100%;

      height: 160px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      display: table-cell;

      vertical-align: middle;

    }

    .mainpagelink{

    left: 0.5%;

    top: 20%;

    color: rgb(255, 255, 255);

    font-family: "Inter UI", "SF Pro Display", -apple-system, BlinkMacSystemFont,

    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",

    sans-serif;

    font-weight: 500;

    padding: 1em 1em 1.5em;

    text-decoration: none;

    text-transform: uppercase;

    cursor: pointer;

    display: block;

    justify-content: center;

    position: absolute;

    border-color: black;

    border-radius: 3px;

    font-size: xx-large;

    background-color: #ff00b7;

    min-width: 160px;

    height: 2rem;

    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

    z-index: 1;

    }

  </style>

  <body>

    <a type="a" class= "mainpagelink" href="mainpage.html">Back to Mainpage</a>

    {% if Username %}

    <h1>You are logged in as: {{Username}}</h1>

    {% endif %}

    <form action="updateDetails" method="post">

      {% csrf\_token %}

      <label for="FirstName">First name:</label><br />

      <input type="text" id="FirstName" name="FirstName" value="" /><br />

      <label for="LastName">Last name:</label><br />

      <input type="text" id="LastName" name="LastName" value="" /><br />

      <label for="Email">Email:</label><br />

      <input type="text" id="Email" name="Email" value="" /><br />

      <label for="Password">Password:</label><br />

      <input type="password" id="pass" name="Password" value="" /><br /><br />

      <label for="PasswordRepeat">Repeat Password:</label><br />

      <input type="password" id="pass" name="PasswordRepeat" value="" /><br /><br />

      <input type="submit" value="Submit"/>

    </form>

    <div class="messages">

      {% if messages %}

        {% for message in messages %}

          <div class="alert alert-{{ message.tags }}" role="alert">

            {{ message }}

          </div>

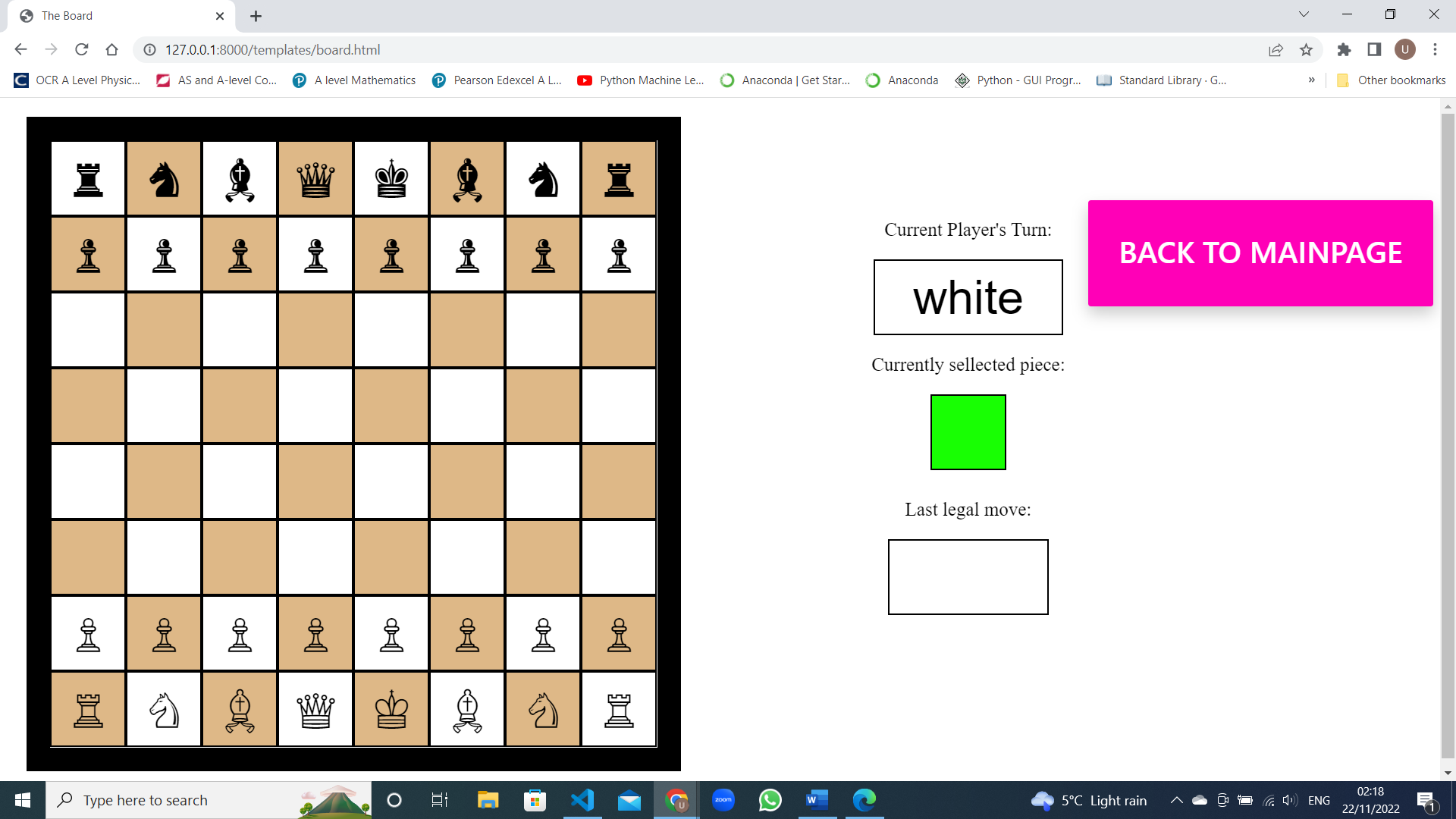
        {% endfor %}

      {% endif %}

    </div>

  </body>

</html>

{% load static %}

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>The Board</title>

  </head>

  <style>

    .chessboard {

      width: 640px;

      height: 640px;

      margin: 20px;

      border: 25px solid rgb(0,0,0);

    }

    .chessboardDisabled {

      pointer-events: none;

      width: 640px;

      height: 640px;

      margin: 20px;

      border: 25px solid rgb(0,0,0);

    }

    .hidePromotionDiv {

      position: absolute;

      width: 640px;

      height: 640px;

      margin: 20px;

      display: none;

      background-color:white;

      border: 25px solid rgb(255, 255, 255);

    }

    .showPromotionDiv {

      position: absolute;

      width: 640px;

      height: 640px;

      margin: 20px;

      background-color:white;

      border: 25px solid rgb(255, 255, 255);

    }

    .black {

      cursor: pointer;

      float: left;

      width: 80px;

      height: 80px;

      background-color: burlywood;

      font-size: 50px;

      text-align: center;

      display: table-cell;

      vertical-align: middle;

    }

    .white {

      cursor: pointer;

      float: left;

      width: 80px;

      height: 80px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      display: table-cell;

      vertical-align: middle;

    }

    .green {

      width: 80px;

      height: 80px;

      background-color: rgb(24, 255, 3);

      font-size: 40px;

      text-align: center;

      left: 60%;

    }

    .playerTurn{

      width: 200px;

      height: 80px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      left: 60%;

    }

    .promotionButton{

      cursor: pointer;

      width: 200px;

      height: 80px;

      background-color: beige;

      font-size: 50px;

      text-align: center;

      left: 60%;

      margin: 5%;

    }

    .biege {

      width: 170px;

      height: 80px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      left: 70%;

    }

    .optionsDiv{

      position: fixed;

      width: 220px;

      height: 200px;

      background-color: rgb(255, 255, 255);

      font-size: 50px;

      text-align: center;

      left: 60%;

      top: 15%;

    }

    .p{

      font-size: 20px;

    }

    .hidePromotionTitle{

      display: none;

    }

    .showPromotionTitle{

      display: block;

    }

    .hidePromotionSelection{

      display: none;

    }

    .showPromotionSelection{

      display: block;

    }

    .mainpagelink{

    right: 0.5%;

    top: 15%;

    color: rgb(255, 255, 255);

    font-family: "Inter UI", "SF Pro Display", -apple-system, BlinkMacSystemFont,

    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",

    sans-serif;

    font-weight: 500;

    padding: 1em 1em 1.5em;

    text-decoration: none;

    text-transform: uppercase;

    cursor: pointer;

    display: block;

    justify-content: center;

    position: absolute;

    border-color: black;

    border-radius: 3px;

    font-size: xx-large;

    background-color: #ff00b7;

    min-width: 160px;

    height: 2rem;

    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

    z-index: 1;

    }

  </style>

  <body>

    <div class = "optionsDiv" id="optionsDiv">

      <p class = "p">Current Player's Turn:</p>

      <button id = "playerTurn" class = "playerTurn"></button>

      <p id = "pieceSelect" class = "p">Currently sellected piece:</p>

      <button class = "green" id = "current"></button>

      <p class = "p">Last legal move:</p>

      <button class = "biege" id = "move"></button>

    </div>

    <div id = "promotionDiv" class = "hidePromotionDiv">

      <h4 class="hidePromotionTitle" id = "promotionTitle">Choose what piece to promote to:</h4>

      <select name="promotionPieces" id="promotionOptions" class = "hidePromotionSelection">

        <option value="none" selected disabled>-- Select Piece --</option>

        <option value="q">Queen</option>

        <option value="r">Rook</option>

        <option value="b">Bishop</option>

        <option value="n">Knight</option>

      </select>

    </div>

    <a type="a" class= "mainpagelink" href="mainpage.html">Back to Mainpage</a>

    <div id= "board" class="chessboard">

      <!-- 1st -->

      <button rank="1" file="1" player="" piece ="" class="white" onClick = "action(event)" id="a8"></button>

      <button rank="1" file="2" player="" piece ="" class="black" onClick = "action(event)" id="b8"></button>

      <button rank="1" file="3" player="" piece ="" class="white" onClick = "action(event)" id="c8"></button>

      <button rank="1" file="4" player="" piece ="" class="black" onClick = "action(event)" id="d8"></button>

      <button rank="1" file="5" player="" piece ="" class="white" onClick = "action(event)" id="e8"></button>

      <button rank="1" file="6" player="" piece =""class="black" onClick = "action(event)" id="f8"></button>

      <button rank="1" file="7" player="" piece =""class="white" onClick = "action(event)" id="g8"></button>

      <button rank="1" file="8" player="" piece =""class="black" onClick = "action(event)" id="h8"></button>

      <!-- 2nd -->

      <button rank="2" file="1" player="" piece ="" class="black" onClick = "action(event)" id="a7"></button>

      <button rank="2" file="2" player="" piece =""class="white" onClick = "action(event)" id="b7"></button>

      <button rank="2" file="3" player="" piece =""class="black" onClick = "action(event)" id="c7"></button>

      <button rank="2" file="4" player="" piece =""class="white" onClick = "action(event)" id="d7"></button>

      <button rank="2" file="5" player="" piece =""class="black" onClick = "action(event)" id="e7"></button>

      <button rank="2" file="6" player="" piece =""class="white" onClick = "action(event)" id="f7"></button>

      <button rank="2" file="7" player="" piece =""class="black" onClick = "action(event)" id="g7"></button>

      <button rank="2" file="8" player="" piece =""class="white" onClick = "action(event)" id="h7"></button>

      <!-- 3th -->

      <button rank="3" file="1" player="" piece =""class="white" onClick = "action(event)" id="a6"></button>

      <button rank="3" file="2" player="" piece =""class="black" onClick = "action(event)" id="b6"></button>

      <button rank="3" file="3" player="" piece =""class="white" onClick = "action(event)" id="c6"></button>

      <button rank="3" file="4" player="" piece =""class="black" onClick = "action(event)" id="d6"></button>

      <button rank="3" file="5" player="" piece =""class="white" onClick = "action(event)" id="e6"></button>

      <button rank="3" file="6" player="" piece =""class="black" onClick = "action(event)" id="f6"></button>

      <button rank="3" file="7" player="" piece =""class="white" onClick = "action(event)" id="g6"></button>

      <button rank="3" file="8" player="" piece =""class="black" onClick = "action(event)" id="h6"></button>

      <!-- 4st -->

      <button rank="4" file="1" player="" piece ="" class="black" onClick = "action(event)" id="a5"></button>

      <button rank="4" file="2" player="" piece ="" class="white" onClick = "action(event)" id="b5"></button>

      <button rank="4" file="3" player="" piece =""class="black" onClick = "action(event)" id="c5"></button>

      <button rank="4" file="4" player="" piece =""class="white" onClick = "action(event)" id="d5"></button>

      <button rank="4" file="5" player="" piece =""class="black" onClick = "action(event)" id="e5"></button>

      <button rank="4" file="6" player="" piece =""class="white" onClick = "action(event)" id="f5"></button>

      <button rank="4" file="7" player="" piece =""class="black" onClick = "action(event)" id="g5"></button>

      <button rank="4" file="8" player="" piece =""class="white" onClick = "action(event)" id="h5"></button>

      <!-- 5th -->

      <button rank="5" file="1" player="" piece ="" class="white" onClick = "action(event)" id="a4"></button>

      <button rank="5" file="2" player="" piece =""class="black" onClick = "action(event)" id="b4"></button>

      <button rank="5" file="3" player="" piece =""class="white" onClick = "action(event)" id="c4"></button>

      <button rank="5" file="4" player="" piece =""class="black" onClick = "action(event)" id="d4"></button>

      <button rank="5" file="5" player="" piece =""class="white" onClick = "action(event)" id="e4"></button>

      <button rank="5" file="6" player="" piece =""class="black" onClick = "action(event)" id="f4"></button>

      <button rank="5" file="7" player="" piece =""class="white" onClick = "action(event)" id="g4"></button>

      <button rank="5" file="8" player="" piece =""class="black" onClick = "action(event)" id="h4"></button>

      <!-- 6th -->

      <button rank="6" file="1" player="" piece =""class="black" onClick = "action(event)" id="a3"></button>

      <button rank="6" file="2" player=""piece ="" class="white" onClick = "action(event)" id="b3"></button>

      <button rank="6" file="3" player="" piece =""class="black" onClick = "action(event)" id="c3"></button>

      <button rank="6" file="4" player=""piece ="" class="white" onClick = "action(event)" id="d3"></button>

      <button rank="6" file="5" player="" piece =""class="black" onClick = "action(event)" id="e3"></button>

      <button rank="6" file="6" player="" piece =""class="white" onClick = "action(event)" id="f3"></button>

      <button rank="6" file="7" player="" piece =""class="black" onClick = "action(event)" id="g3"></button>

      <button rank="6" file="8" player="" piece =""class="white" onClick = "action(event)" id="h3"></button>

      <!-- 7th -->

      <button rank="7" file="1" player="" piece =""class="white" onClick = "action(event)" id="a2"></button>

      <button rank="7" file="2" player="" piece =""class="black" onClick = "action(event)" id="b2"></button>

      <button rank="7" file="3" player="" piece =""class="white" onClick = "action(event)" id="c2"></button>

      <button rank="7" file="4" player="" piece =""class="black" onClick = "action(event)" id="d2"></button>

      <button rank="7" file="5" player="" piece =""class="white" onClick = "action(event)" id="e2"></button>

      <button rank="7" file="6" player="" piece =""class="black" onClick = "action(event)" id="f2"></button>

      <button rank="7" file="7" player="" piece =""class="white" onClick = "action(event)" id="g2"></button>

      <button rank="7" file="8" player="" piece =""class="black" onClick = "action(event)" id="h2"></button>

      <!-- 8th -->

      <button rank="8" file="1" player="" piece =""class="black" onClick = "action(event)" id="a1"></button>

      <button rank="8" file="2" player="" piece =""class="white" onClick = "action(event)" id="b1"></button>

      <button rank="8" file="3" player="" piece =""class="black" onClick = "action(event)" id="c1"></button>

      <button rank="8" file="4" player="" piece =""class="white" onClick = "action(event)" id="d1"></button>

      <button rank="8" file="5" player="" piece =""class="black" onClick = "action(event)" id="e1"></button>

      <button rank="8" file="6" player="" piece =""class="white" onClick = "action(event)" id="f1"></button>

      <button rank="8" file="7" player="" piece =""class="black" onClick = "action(event)" id="g1"></button>

      <button rank="8" file="8" player="" piece =""class="white" onClick = "action(event)" id="h1"></button>

    </div>

    {% csrf\_token %}

    <script>

let legalEnPassant = false

let whiteKingSide = false

let whiteQueenSide = false

let blackQueenSide = false

let blackKingSide = false

let moveValid = false

function getStates(){

  return $.ajax({

    url: '/boardStates',

    type: 'get',

    async: false,

    data: {

        hide : "getting Boardstates",

    }});

}

function validMove(moveMade){

  let csrf = $('input[name=csrfmiddlewaretoken]').val();

  return $.ajax({

    url: '/validMove',

    type: 'post',

    async: false,

    data: {

        move : moveMade,

        csrfmiddlewaretoken: csrf

    }});

}

function action(event){

  const promotionPlaces = ["a8","b8","c8","d8","e8","f8","g8","h8","a1","b1","c1","d1","e1","f1","g1","h1"];

  const enPassantPositions = ["a6","b6","c6","d6","e6","f6","g6","h6","a3","b3","c3","d3","e3","f3","g3","h3"];

  let csrf = $('input[name=csrfmiddlewaretoken]').val();

  if (currentSquare == ""){

  currentSquare = String(event.target.id);

  document.getElementById("current").innerHTML = currentSquare

  //If there was no current square selected then the id of the clicked on element becomes the current square

  }

  else{

    targetSquare = String(event.target.id);

    move = currentSquare + targetSquare

    //If there was a current square than the target square bceomes the cliked on element

$.when(getStates()).then(function successHandler(response){

    legalEnPassant = response.legalEnPassant

    whiteKingSide = response.whiteKingSide

    whiteQueenSide = response.whiteQueenSide

    blackQueenSide = response.blackQueenSide

    blackKingSide = response.blackKingSide

 },

 function errorHandler(){

   console.log("Error has occurred")

 })

//Getting whether each side has casling rights

$.when(validMove(move)).then(function successHandler(response){

  moveValid = response.valid

  gameOver = response.gameOver

 },

 function errorHandler(){

   console.log("Error has occurred")

 })

//An ajax call to get whether the move is valid from the django side

if (moveValid == true){

  document.getElementById("move").innerHTML = move

  if (document.getElementById(currentSquare).getAttribute("piece") == "pawn" && document.getElementById(targetSquare).getAttribute("piece") == "" && enPassantPositions.includes(move.slice(2,4)) && legalEnPassant){

    console.log("enpassanting")

             if (document.getElementById(currentSquare).getAttribute("player") == "white"){

                 document.getElementById(move.slice(2,3) + String((Number(move.slice(3,4))-1))).innerHTML = ""

                 document.getElementById(move.slice(2,3) + String((Number(move.slice(3,4))-1))).setAttribute("piece","")

                 document.getElementById(move.slice(2,3) + String((Number(move.slice(3,4))-1))).setAttribute("player","")

             }

             else if (document.getElementById(currentSquare).getAttribute("player") == "black"){

                 document.getElementById(move.slice(2,3) + String((Number(move.slice(3,4))+1))).innerHTML = ""

                 document.getElementById(move.slice(2,3) + String((Number(move.slice(3,4))+1))).setAttribute("piece","")

                 document.getElementById(move.slice(2,3) + String((Number(move.slice(3,4))+1))).setAttribute("player","")

                }

                moveOnScreen(currentSquare,targetSquare)

  }

  //The above if statement deals with en passent captures on the GUI side

  else if (document.getElementById(currentSquare).getAttribute("piece") == "king" && document.getElementById(currentSquare).getAttribute("player") == "white" && targetSquare == "g1" && whiteKingSide){

  moveOnScreen("h1","f1")

  moveOnScreen(currentSquare,targetSquare)

  document.getElementById("move").innerHTML = move

  resetAll()

  }

  else if (document.getElementById(currentSquare).getAttribute("piece") == "king" && document.getElementById(currentSquare).getAttribute("player") == "white" && targetSquare == "c1" && whiteQueenSide){

  moveOnScreen("a1","d1")

  moveOnScreen(currentSquare,targetSquare)

  document.getElementById("move").innerHTML = move

  resetAll()

  }

  else if (document.getElementById(currentSquare).getAttribute("piece") == "king" && document.getElementById(currentSquare).getAttribute("player") == "black" && targetSquare == "c8" && blackQueenSide){

  moveOnScreen("a8","d8")

  moveOnScreen(currentSquare,targetSquare)

  document.getElementById("move").innerHTML = move

  resetAll()

  }

  else if (document.getElementById(currentSquare).getAttribute("piece") == "king" && document.getElementById(currentSquare).getAttribute("player") == "black" && targetSquare == "g8" && blackKingSide){

  moveOnScreen("h8","f8")

  moveOnScreen(currentSquare,targetSquare)

  document.getElementById("move").innerHTML = move

  resetAll()

}

else{

  moveOnScreen(currentSquare,targetSquare)

}

if (playerTurn == "white"){

  playerTurn = "black"

  document.getElementById("playerTurn").innerHTML = playerTurn

  }

  else{

      playerTurn = "white"

      document.getElementById("playerTurn").innerHTML = playerTurn

  }

// ^Switching which side plays next

  document.getElementById("board").setAttribute("class", "chessboardDisabled")

  if (!(gameOver)){

  resetAll()

  AIPlay()

  }

  else{

    console.log(gameOver)

    gameIsOver()

    trainModel()

  }

// ^To let the AI make its move

}

else  if (promotionPlaces.includes(move.slice(2,4)) && document.getElementById(currentSquare).getAttribute("piece") == "pawn"){

  document.getElementById("promotionDiv").setAttribute("class", "showPromotionDiv")

  document.getElementById("promotionTitle").setAttribute("class", "showPromotionTitle")

  document.getElementById("promotionOptions").setAttribute("class", "showPromotionSelection")

}

//^This opens the option panel to promote a pawn

else{

  console.log("Invalid move")

  resetAll()

}

}

}

    </script>

    <script src="{% static 'AJAX.js' %}"></script>

    <script src="{% static 'classesAndInstances.js' %}"></script>

    <script src="{% static 'startBoard.js' %}"></script>

    <script src="{% static 'AIplaying.js' %}"></script>

    <script src="{% static 'gameOver.js' %}"></script>

    <script src="{% static 'moveOnScreen.js' %}"></script>

    <script src="{% static 'promoter.js' %}"></script>

    <script src="{% static 'resetAllGlobals.js' %}"></script>

    <script src="{% static 'resetCurrentSquare.js' %}"></script>

  </body>

</html>

from django.shortcuts import render

from django.http import  JsonResponse

import chess

import chess.engine

import random

import numpy

import tensorflow as tf

from keras import models

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2' #Tells program to ignore an unimportant warning

Boardsnaps = []

Boardscores = []

board = chess.Board()

model = ""

User = ""

letterToCoordinate = {

  'a': 0,

  'b': 1,

  'c': 2,

  'd': 3,

  'e': 4,

  'f': 5,

  'g': 6,

  'h': 7

}

def model\_set(user):

    global model

    model = models.load\_model(f'./models/{user}')

def userSet(user):

    global User

    User = user

@tf.function #This line creates an optimized graph temporarily in memory to quickly make predictions

def get\_score(x):

    return model(x)

def SaveBoardData(Boardsnaps,Boardscores):

    numpy.save("ManyBoards.npy",Boardsnaps, True)

    numpy.save("ManyScores.npy",Boardscores, True)

def SaveBoard(board): #Creates and stores a picture of the chess board in this instant and records the score as well.

        AIboard = ConvertToAIboard(board)

        Score = getScore(board,not(board.turn))

        Boardsnaps.append(AIboard)

        Boardscores.append(Score)

        SaveBoardData(Boardsnaps,Boardscores)

def coordinateToIndex(square):

  letter = chess.square\_name(square) #This function call converts the square index of a square such as 28 to its name; which for '28' would be 'e4'

  return 8 - int(letter[1]), letterToCoordinate[letter[0]] #This part takes that name, e.g. 'e4' and convert this position to its coordinate of 4 4 using the letterToCoordinate dictionary to convert the letter

def ConvertToAIboard(board):

    #This program turns the chess board into something the AI can analyse better

    AIboard = numpy.zeros((14, 8, 8), dtype=numpy.int8)

    pawn\_number = 1

    knight\_number = 2

    bishop\_number = 3

    rook\_number = 4

    queen\_number = 5

    king\_number = 6

    #unravel\_index converts an index such as 9 into its corresponding co ordinate for an array of given dimensions.

    #for example: in an 8 by 8 dimensional array: the co ordinates corresponding to the index 9 would be (1,1)

    #The index (1,1) was calculated by doing 1 x 8 + 1 to get 9

    for index in board.pieces(pawn\_number, True):

        position = numpy.unravel\_index(index, (8, 8))  #In a board of 64, each coordinate is represented by an index numberered from 0 to 63, in order to convert this index to a coordinate: I have unravelled it

        AIboard[pawn\_number - 1][position[0]][position[1]] = 1 #The first coordinate is the number of the array (out of 12) the second coordinate and the other two are y and x coordinates respectively

    for index in board.pieces(pawn\_number, False): #First the indexes of all the white pieces of the piece number are added to the numpy array: AIboard

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[pawn\_number][position[0]][position[1]] = 1

    for index in board.pieces(knight\_number, True):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[knight\_number][position[0]][position[1]] = 1

    for index in board.pieces(knight\_number, False):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[knight\_number + 1][position[0]][position[1]] = 1

    for index in board.pieces(bishop\_number, True):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[bishop\_number + 1][position[0]][position[1]] = 1

    for index in board.pieces(bishop\_number, False):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[bishop\_number + 2][position[0]][position[1]] = 1

    for index in board.pieces(rook\_number, True):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[rook\_number + 2][position[0]][position[1]] = 1

    for index in board.pieces(rook\_number, False):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[rook\_number + 3][position[0]][position[1]] = 1

    for index in board.pieces(queen\_number, True):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[queen\_number + 3][position[0]][position[1]] = 1

    for index in board.pieces(queen\_number, False):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[queen\_number + 4][position[0]][position[1]] = 1

    for index in board.pieces(king\_number, True):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[king\_number + 4][position[0]][position[1]] = 1

    for index in board.pieces(king\_number, False):

        position = numpy.unravel\_index(index, (8, 8))

        AIboard[king\_number + 5][position[0]][position[1]] = 1

    #The pairs of for loops create 2, two dimensional numpy arrays with ones representing the case where a piece of the

    #corresponding number and color is positioned on it

    RealTurn = board.turn

    board.turn = chess.WHITE

    for move in board.generate\_legal\_captures():

        y , x = coordinateToIndex(move.to\_square)

        AIboard[12][y][x] = 1

    board.turn = chess.BLACK

    for move in board.generate\_legal\_captures():

        y , x = coordinateToIndex(move.to\_square)

        AIboard[13][y][x] = 1

    board.turn = RealTurn

    #The above chunk of code turns all the squares that black and white can capture a piece on and converts it into into a square index

    #The chunck of code saves the current player turn in a temporray variable to retore to once its simulated the board from the perspective of both players

    return(AIboard)

def getScore(board,side):

    with chess.engine.SimpleEngine.popen\_uci("templates\stockfish.exe") as stockfish: #Calling a well known chess AI to give an estimated board score at a certain position for a given board position

        info = stockfish.analyse(board, chess.engine.Limit(time=0.01))

        if info["score"] == "None":

            return 0

        if str(info["score"].relative) == "#-0" and side:

            return 100000

        if str(info["score"].relative) == "#-0"and (not(side)):

            return -100000

        if "#" in str(info["score"].relative):

            return int(str(info["score"].relative)[1:])

        else:

            return int(str(info["score"].white().score()))

def miniMax(user,board, depth, alpha, beta, maxscorer):

    if depth == 0 or board.is\_game\_over():

        x = ConvertToAIboard(board) #Turns the board object into 2d array of arrays of 1s and 0s

        x = x.reshape(1,14,8,8) #Adds an extra dimension to the now array of arrays so it is like an item in a list with only one item

        return get\_score(x)

    child\_nodes = list(board.legal\_moves)

    if maxscorer == False: #If we want the best move for black

        worstScore = numpy.inf #The worst score is infinity

        for child in child\_nodes:

            board.push(child) #Play the first move

            position\_evaluation = float(miniMax(user,board,depth-1,alpha, beta, True)[0]) #Get the score at the position at the end of the tree

            board.pop() #Restore the board to before playing the move

            if position\_evaluation < worstScore: #Replace the worst score as you go along to get the lowest possible score

                worstScore = position\_evaluation

                best\_move = child

            beta = min(beta,position\_evaluation) #Pruning off the section of the tree which the oppononet is not likely to go down

            if beta <= alpha:

                break

        return worstScore, best\_move

    elif maxscorer == True: #If we want the best move for white

        bestScore = -numpy.inf #The worst score is negative infinity

        for child in child\_nodes:

            board.push(child)

            position\_evaluation = float(miniMax(user,board,depth-1,alpha, beta, False)[0]) #Get the score at the position at the end of the tree

            board.pop() #Restore the board to before playing the move

            if position\_evaluation > bestScore: #Replace the worst score as you go along to get the highest possible score

                bestScore = position\_evaluation

                best\_move = child

            alpha = max(alpha,position\_evaluation)#Pruning off the section of the tree which the oppononet is not likely to go down

            if beta <= alpha:

                break

        return bestScore, best\_move

def train\_model(request):

    global User

    temp = request.GET.get('hide')

    x = numpy.load("Boards.npy")

    y = numpy.load("Scores.npy")

    y = numpy.asarray(y / abs(y).max() / 2 + 0.5, dtype=numpy.float32) # normalization (0 - 1)

    x\_test = numpy.load("TestBoards.npy")

    y\_test = numpy.load("TestScores.npy")

    y\_test = numpy.asarray(y\_test / abs(y\_test).max() / 2 + 0.5, dtype=numpy.float32) # normalization (0 - 1)

    model.fit(x,y,epochs=100)

    results = model.evaluate(x\_test, y\_test, batch\_size=128)

    relative = (( 0.0006020597647875547 - results ) / 0.0006020597647875547) \*100

    User.updateRelative(relative)

    model.save(f'./models/{User.getUsername()}')

    return JsonResponse({"trained": True},status = 200)

def isValid(move):

    global board

    try:

        board.push\_san(move)

        SaveBoard(board)

        return True

    except ValueError:

        return False

def boardStates(request):

    temp = request.GET.get('hide')

    legalEnPassant =  board.has\_legal\_en\_passant()

    whiteKingSide = bool(chess.BB\_H1)

    whiteQueenSide = bool(chess.BB\_A1)

    blackQueenSide = bool(chess.BB\_A8)

    blackKingSide = bool(chess.BB\_H8)

    return JsonResponse({"legalEnPassant": legalEnPassant,

    "whiteKingSide": whiteKingSide,

    "whiteQueenSide": whiteQueenSide,

    "blackQueenSide": blackQueenSide,

    "blackKingSide": blackKingSide},status = 200)

def resetBoard(request):

    temp = request.GET.get('hide')

    board.reset()

    return JsonResponse({"reset":True}, status = 200)

def getWinner(request):

    temp = request.GET.get('hide')

    winner = board.outcome().winner

    if winner == True:

        winner = "White Wins!"

    else:

        winner = "Black Wins!"

    return JsonResponse({"winner":winner}, status = 200)

def validMove(request):

    move = str(request.POST.get('move'))

    m = isValid(move)

    gameOver = board.is\_game\_over()

    return JsonResponse({

    "gameOver": gameOver,

    "valid": m }, status = 200)

def load\_board(request):

    global User

    from users.views import LoggedInUser

    if LoggedInUser == "":

        return render(request, 'mainpage.html')

    else:

        User = LoggedInUser

        model\_set(User.getUsername())

        return render(request, 'board.html')

def home(request):

    if User == "":

        return render(request, 'mainpage.html')

    else:

        return render(request, 'mainpage.html',{'Username' : User.getUsername()})

# count = 0

def AiMove(request):

    global count

    temp = request.GET.get('stuff')

    legalEnPassant =  board.has\_legal\_en\_passant()

    whiteKingSide = bool(chess.BB\_H1)

    whiteQueenSide = bool(chess.BB\_A1)

    blackQueenSide = bool(chess.BB\_A8)

    blackKingSide = bool(chess.BB\_H8)

    moveAI = ""

    # moves = ['d7d5','d5c4','c4b3','b3a2','a2b1n'] #A specififc set of moves to test the AI can promote pawns as expected

    # moves = ['d7d5','d5e4','e4f3','f3g2','g2h1q'] #A specififc set of moves to test the AI can promote pawns as expected

    # moves = ['g7g5','g5g4','g4h3']  #A black side enpassant capture

    # moves = ['g7g5','g5g4','g4f3']  #A black side enpassant capture

    # moves = ['h7h6','a7a5'] #A white side enpassant capture

    # moves = ['h7h6','d7d5'] #A white side enpassant capture

    # moves = ['e7e6','f8e7','g8f6','e8g8'] #Castling for white and black kingside

    # moves = ['d7d5','c8e6','b8c6','d8d7','e8c8'] #Castling for white and black queenside

    # board.push\_uci(moves[count])

    # moveAI = moves[count]

    # count += 1

    # try: #Playing against a random AI which is easy to beat for testing what happens when the player wins

    #     child\_nodes = list(board.legal\_moves)

    #     move = random.choice(child\_nodes)

    #     board.push(move)

    #     SaveBoard(board)

    #     moveAI = str(move)

    # except IndexError: #What to do if the the game is over (AI lost)

    #     moveAI = ""

    gameOver = board.is\_game\_over()

    # try: #Playing against the most power chess AI at this time to get beaten by the AI to see what happens when the AI wins

    #     Stockfish = chess.engine.SimpleEngine.popen\_uci("templates\stockfish.exe")

    #     moveAI = Stockfish.play(board,chess.engine.Limit(time=0.1))

    #     Stockfish.quit()

    #     board.push(moveAI.move)

    #     SaveBoard(board)

    #     moveAI = str(moveAI.move)

    # except AttributeError: #What to do if the the game is over (AI lost)

    #     moveAI = ""

    # gameOver = board.is\_game\_over()

    # try: #This is my actual AI code

    #     moveAI = (miniMax("TICO",board, 2, -numpy.inf, numpy.inf, False)[1])

    #     board.push(moveAI)

    #     SaveBoard(board)

    # except Exception as e: #If the game is over and the AI lost then no AI move should be returned

    #     pass

    # gameOver = board.is\_game\_over()

    # moveAI = str(moveAI)

    return JsonResponse({"legalEnPassant": legalEnPassant,

    "whiteKingSide": whiteKingSide,

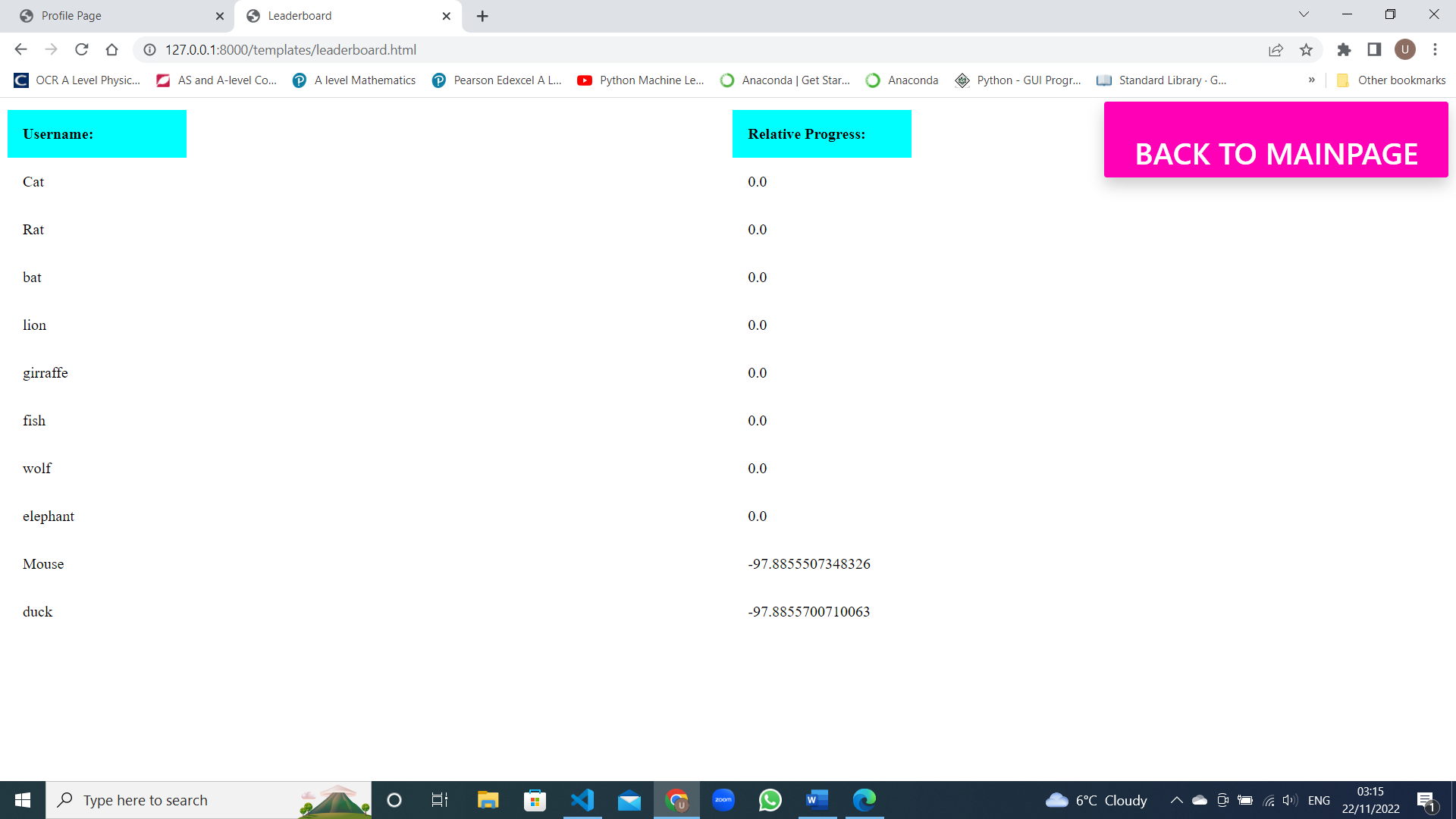
    "whiteQueenSide": whiteQueenSide,

    "blackQueenSide": blackQueenSide,

    "blackKingSide": blackKingSide,

    "moveAI": moveAI,

    "gameOver":gameOver }, status = 200)



<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>Leaderboard</title>

  </head>

  <style>

    .mainpagelink{

    right: 0.5%;

    top: 0.5%;

    color: rgb(255, 255, 255);

    font-family: "Inter UI", "SF Pro Display", -apple-system, BlinkMacSystemFont,

    "Segoe UI", Roboto, Oxygen, Ubuntu, Cantarell, "Open Sans", "Helvetica Neue",

    sans-serif;

    font-weight: 500;

    padding: 1em 1em 1.5em;

    text-decoration: none;

    text-transform: uppercase;

    cursor: pointer;

    display: block;

    justify-content: center;

    position: absolute;

    border-color: black;

    border-radius: 3px;

    font-size: xx-large;

    background-color: #ff00b7;

    min-width: 160px;

    height: 2rem;

    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);

    z-index: 1;

    }

    \* {

  box-sizing: border-box;

}

.row {

  display: flex;

  margin-left:-5px;

  margin-right:-5px;

}

.column {

  flex: 50%;

  padding: 5px;

}

table {

  border-collapse: collapse;

  border-spacing: 0;

  width: 25%;

}

td {

  text-align: left;

  padding: 16px;

}

th {

  text-align: left;

  padding: 16px;

  background-color: aqua;

}

  </style>

  <body>

    <a type="a" class= "mainpagelink" href="mainpage.html">Back to Mainpage</a>

    <div class="row">

      <div class="column">

    <table>

        <tr>

            <th>Username:</th>

        </tr>

      <tr>

        {% for username in users %}

        <tr>

        <td> {{ username.0 }}</td>

      </tr>

        {% endfor %}

      </table>

    </div>

    <div class="column">

    <table>

      <tr>

        <th>Relative Progress:</th>

      </tr>

        {% for score in scores %}

        <tr>

        <td> {{ score.0 }}</td>

      </tr>

        {% endfor %}

      </tr>

    </table>

  </div>

</div>

  </body>

</html>



## 

^This is the URLS.py file that maps URLS to functions e.g. that render html webpages

## Testing:

<https://youtu.be/RvRVKf-xdVw>

^Video includes test plan

Improvements:



^SQL Querry did not return relative progress with the largest values first in the tests. So I changed the code and the out put can be see in the code listings

## Static javascript files:

AIplaying.js:

function getAImove(){

    return $.ajax({

      url: '/AiMove',

      type: 'get',

      async: true,

      data: {

          stuff : "getting Aimove",

      }});

  }

function AIPlay(){

    $.when(getAImove()).then(function successHandler(response){

      legalEnPassant = response.legalEnPassant

      whiteKingSide = response.whiteKingSide

      whiteQueenSide = response.whiteQueenSide

      blackQueenSide = response.blackQueenSide

      blackKingSide = response.blackKingSide

      AImove = response.moveAI

      gameOver = response.gameOver

      const promotionPlaces = ["a8","b8","c8","d8","e8","f8","g8","h8","a1","b1","c1","d1","e1","f1","g1","h1"];

      const enPassantPositions = ["a6","b6","c6","d6","e6","f6","g6","h6","a3","b3","c3","d3","e3","f3","g3","h3"];

  AIsquare = AImove.substring(0, 2)

  AIsquareTarget = AImove.substring(2, 4)

  if (AImove.length == 5){

    piece = AImove.substr(4,5)

  if (piece == "q"){

      document.getElementById(AIsquareTarget).setAttribute("piece","queen")

      document.getElementById(AIsquareTarget).setAttribute("player","black")

      document.getElementById(AIsquareTarget).innerHTML = "&#9819;"

    }

  else if (piece == "r"){

      document.getElementById(AIsquareTarget).setAttribute("piece","rook")

      document.getElementById(AIsquareTarget).setAttribute("player","black")

      document.getElementById(AIsquareTarget).innerHTML = "&#9820;"

    }

  else if (piece == "b"){

      document.getElementById(AIsquareTarget).setAttribute("piece","bishop")

      document.getElementById(AIsquareTarget).setAttribute("player","black")

      document.getElementById(AIsquareTarget).innerHTML = "&#9821;"

    }

  else if (piece == "n"){

      document.getElementById(AIsquareTarget).setAttribute("piece","knight")

      document.getElementById(AIsquareTarget).setAttribute("player","black")

      document.getElementById(AIsquareTarget).innerHTML = "&#9822;"

    }

    resetCurrentSquare(AIsquare)

  }

  else{

  if (document.getElementById(AIsquare).getAttribute("piece") == "pawn" && document.getElementById(AIsquareTarget).getAttribute("piece") == "" && enPassantPositions.includes(AImove.slice(2,4)) && legalEnPassant){

             if (document.getElementById(AIsquare).getAttribute("player") == "white"){

                 document.getElementById(AImove.slice(2,3) + String((Number(AImove.slice(3,4))-1))).innerHTML = ""

                 document.getElementById(AImove.slice(2,3) + String((Number(AImove.slice(3,4))-1))).setAttribute("piece","")

                 document.getElementById(AImove.slice(2,3) + String((Number(AImove.slice(3,4))-1))).setAttribute("player","")

             }

             else if (document.getElementById(AIsquare).getAttribute("player") == "black"){

                 document.getElementById(AImove.slice(2,3) + String((Number(AImove.slice(3,4))+1))).innerHTML = ""

                 document.getElementById(AImove.slice(2,3) + String((Number(AImove.slice(3,4))+1))).setAttribute("piece","")

                 document.getElementById(AImove.slice(2,3) + String((Number(AImove.slice(3,4))+1))).setAttribute("player","")

                }

                moveOnScreen(AIsquare,AIsquareTarget)

  }

  //The above if statement deals with en passent captures on the GUI side

  else if (document.getElementById(AIsquare).getAttribute("piece") == "king" && document.getElementById(AIsquare).getAttribute("player") == "white" && AIsquareTarget == "g1" && whiteKingSide){

  moveOnScreen("h1","f1")

  moveOnScreen(AIsquare,AIsquareTarget)

  document.getElementById("move").innerHTML = AImove

  }

  else if (document.getElementById(AIsquare).getAttribute("piece") == "king" && document.getElementById(AIsquare).getAttribute("player") == "white" && AIsquareTarget == "c1" && whiteQueenSide){

  moveOnScreen("a1","d1")

  moveOnScreen(AIsquare,AIsquareTarget)

  document.getElementById("move").innerHTML = AImove

  }

  else if (document.getElementById(AIsquare).getAttribute("piece") == "king" && document.getElementById(AIsquare).getAttribute("player") == "black" && AIsquareTarget == "c8" && blackQueenSide){

  moveOnScreen("a8","d8")

  moveOnScreen(AIsquare,AIsquareTarget)

  document.getElementById("move").innerHTML = AImove

  }

  else if (document.getElementById(AIsquare).getAttribute("piece") == "king" && document.getElementById(AIsquare).getAttribute("player") == "black" && AIsquareTarget == "g8" && blackKingSide){

  moveOnScreen("h8","f8")

  moveOnScreen(AIsquare,AIsquareTarget)

  document.getElementById("move").innerHTML = AImove

  }

  else{

    moveOnScreen(AIsquare,AIsquareTarget)

  }

}

  if (playerTurn == "white"){

    playerTurn = "black"

  }

  else{

      playerTurn = "white"

  }

  if (!(gameOver)){

    document.getElementById("playerTurn").innerHTML = playerTurn

    document.getElementById("move").innerHTML = AImove

    document.getElementById("board").setAttribute("class", "chessboard")

    }

    else{

      console.log(gameOver)

        gameIsOver()

        trainModel()

    }

    },

    function errorHandler(){

      console.log("Error has occurred")

    })

  }

classesAndInstances.js:

class Piece {

    constructor(name, color,value,symbol) {

      this.name = name;

      this.color = color;

      this.value = value;

      this.symbol = symbol;

    }

    getSymbol(){

        return this.symbol;

    }

    getName(){

        return this.name;

    }

    getColor(){

        return this.color;

    }

    getValue(){

        return this.value;

    }

  }

class Pawn extends Piece {

constructor(color,symbol) {

    super("pawn", color , 10, symbol);

}

}

class Rook extends Piece {

constructor(color,symbol) {

    super("rook", color , 50, symbol);

}

}

class Knight extends Piece {

constructor(color,symbol) {

    super("knight", color , 30, symbol);

}

}

class Bishop extends Piece {

constructor(color,symbol) {

    super("bishop", color , 30, symbol);

}

}

class King extends Piece {

constructor(color,symbol) {

    super("king", color , 900, symbol);

}

}

class Queen extends Piece {

constructor(color,symbol) {

    super("queen", color , 90, symbol);

}

}

BlackPawn = new Pawn("black","&#9823");

BlackRook = new Rook("black","&#9820;");

BlackKnight = new Knight("black","&#9822;");

BlackBishop = new Bishop("black","&#9821;");

BlackQueen = new Queen("black","&#9819;");

BlackKing = new King("black","&#9818;");

WhitePawn = new Pawn("white","&#9817;");

WhiteRook = new Rook("white","&#9814;");

WhiteKnight = new Knight("white","&#9816;");

WhiteBishop = new Bishop("white","&#9815;");

WhiteQueen = new Queen("white","&#9813;");

WhiteKing = new King("white","&#9812;");

gameOver.js:

class Piece {

    constructor(name, color,value,symbol) {

      this.name = name;

      this.color = color;

      this.value = value;

      this.symbol = symbol;

    }

    getSymbol(){

        return this.symbol;

    }

    getName(){

        return this.name;

    }

    getColor(){

        return this.color;

    }

    getValue(){

        return this.value;

    }

  }

class Pawn extends Piece {

constructor(color,symbol) {

    super("pawn", color , 10, symbol);

}

}

class Rook extends Piece {

constructor(color,symbol) {

    super("rook", color , 50, symbol);

}

}

class Knight extends Piece {

constructor(color,symbol) {

    super("knight", color , 30, symbol);

}

}

class Bishop extends Piece {

constructor(color,symbol) {

    super("bishop", color , 30, symbol);

}

}

class King extends Piece {

constructor(color,symbol) {

    super("king", color , 900, symbol);

}

}

class Queen extends Piece {

constructor(color,symbol) {

    super("queen", color , 90, symbol);

}

}

BlackPawn = new Pawn("black","&#9823");

BlackRook = new Rook("black","&#9820;");

BlackKnight = new Knight("black","&#9822;");

BlackBishop = new Bishop("black","&#9821;");

BlackQueen = new Queen("black","&#9819;");

BlackKing = new King("black","&#9818;");

WhitePawn = new Pawn("white","&#9817;");

WhiteRook = new Rook("white","&#9814;");

WhiteKnight = new Knight("white","&#9816;");

WhiteBishop = new Bishop("white","&#9815;");

WhiteQueen = new Queen("white","&#9813;");

WhiteKing = new King("white","&#9812;");

moveOnScreen.js:

function moveOnScreen(currentSquare, targetSquare){

    document.getElementById(targetSquare).setAttribute("piece",document.getElementById(currentSquare).getAttribute("piece"))

    document.getElementById(targetSquare).setAttribute("player",document.getElementById(currentSquare).getAttribute("player"))

    document.getElementById(targetSquare).innerHTML = document.getElementById(currentSquare).innerHTML

  document.getElementById(currentSquare).setAttribute("piece","")

  document.getElementById(currentSquare).setAttribute("player","")

  document.getElementById(currentSquare).innerHTML = ""

}

Promoter.js:

let selector = document.getElementById("promotionOptions")

// Adding an event listener now that the promotions menu is visible.

selector.addEventListener("change", () => {

    let pieces = document.getElementById("promotionOptions")

    let piece = pieces.options[pieces.selectedIndex].value

    move =  move + piece

    $("#promotionOptions").val("none")

    $.when(validMove(move)).then(function successHandler(response){

      moveValid = response.valid

      gameOver = response.gameOver

     },

     function errorHandler(){

       console.log("Error has occurred")

     })

    if (moveValid==true) {

    document.getElementById("move").innerHTML = move

    color = document.getElementById(currentSquare).getAttribute("player")

    if (color == "white"){

        if (piece == "q"){

            document.getElementById(targetSquare).setAttribute("piece","queen")

            document.getElementById(targetSquare).setAttribute("player","white")

            document.getElementById(targetSquare).innerHTML = "&#9813;"

        }

        else if (piece == "r"){

            document.getElementById(targetSquare).setAttribute("piece","rook")

            document.getElementById(targetSquare).setAttribute("player","white")

            document.getElementById(targetSquare).innerHTML = "&#9814;"

        }

        else if (piece == "b"){

            document.getElementById(targetSquare).setAttribute("piece","bishop")

            document.getElementById(targetSquare).setAttribute("player","white")

            document.getElementById(targetSquare).innerHTML = "&#9815;"

        }

        else if (piece == "n"){

            document.getElementById(targetSquare).setAttribute("piece","knight")

            document.getElementById(targetSquare).setAttribute("player","white")

            document.getElementById(targetSquare).innerHTML ="&#9816;"

          }

      }

    else if (color == "black"){

        if (piece == "q"){

            document.getElementById(targetSquare).setAttribute("piece","queen")

            document.getElementById(targetSquare).setAttribute("player","black")

            document.getElementById(targetSquare).innerHTML = "&#9819;"

          }

        else if (piece == "r"){

            document.getElementById(targetSquare).setAttribute("piece","rook")

            document.getElementById(targetSquare).setAttribute("player","black")

            document.getElementById(targetSquare).innerHTML = "&#9820;"

          }

        else if (piece == "b"){

            document.getElementById(targetSquare).setAttribute("piece","bishop")

            document.getElementById(targetSquare).setAttribute("player","black")

            document.getElementById(targetSquare).innerHTML = "&#9821;"

          }

        else if (piece == "n"){

            document.getElementById(targetSquare).setAttribute("piece","knight")

            document.getElementById(targetSquare).setAttribute("player","black")

            document.getElementById(targetSquare).innerHTML = "&#9822;"

          }

        }

    resetCurrentSquare(currentSquare)

    document.getElementById("promotionDiv").setAttribute("class", "hideromotionDiv")

    document.getElementById("promotionTitle").setAttribute("class", "hidePromotionTitle")

    document.getElementById("promotionOptions").setAttribute("class", "hidePromotionSelection")

    if (playerTurn == "white"){

      playerTurn = "black"

      }

      else{

          playerTurn = "white"

      }

    document.getElementById("playerTurn").innerHTML = playerTurn

    document.getElementById("board").setAttribute("class", "chessboardDisabled")

    if (!(gameOver)){

      resetAll()

      AIPlay()

      }

      else{

        console.log(gameOver)

        gameIsOver()

        trainModel()

      }

}else{

  resetAll()

  document.getElementById("promotionDiv").setAttribute("class", "hideromotionDiv")

  document.getElementById("promotionTitle").setAttribute("class", "hidePromotionTitle")

  document.getElementById("promotionOptions").setAttribute("class", "hidePromotionSelection")

}}

)

ResetAllGlobals:

function resetAll(){

    document.getElementById("current").innerHTML = ""

    currentSquare = ""

    targetSquare = ""

    move = ""

}

ResetCurrentSquare:

function resetCurrentSquare(currentSquare){

    document.getElementById(currentSquare).setAttribute("piece","")

    document.getElementById(currentSquare).setAttribute("player","")

    document.getElementById(currentSquare).innerHTML = ""

}

StartBoard.js:

let currentSquare = ""

let targetSquare = ""

let playerTurn = "white"

let move = ""

let playerSide = ""

function resetBoard(){

  $.ajax({

    url: '/resetBoard',

    type: 'get',

    async: false,

    data: {

        hide : "ResetBoard"},

  })};

function reset\_game(){

    BlackPawns = ["a7","b7","c7","d7","e7","f7","g7","h7"];

    for (let i of BlackPawns) {

        document.getElementById(i).innerHTML = BlackPawn.getSymbol()

        document.getElementById(i).setAttribute("piece",BlackPawn.getName())

        document.getElementById(i).setAttribute("player",BlackPawn.getColor())

    };

    BlackRooks = ["a8","h8"];

    for (let i of BlackRooks) {

        document.getElementById(i).innerHTML = BlackRook.getSymbol()

        document.getElementById(i).setAttribute("piece",BlackRook.getName())

        document.getElementById(i).setAttribute("player",BlackRook.getColor())

    };

    BlackKnights = ["b8","g8"];

    for (let i of BlackKnights) {

        document.getElementById(i).innerHTML = BlackKnight.getSymbol()

        document.getElementById(i).setAttribute("piece",BlackKnight.getName())

        document.getElementById(i).setAttribute("player",BlackKnight.getColor())

    };

    BlackBishops = ["c8","f8"];

    for (let i of BlackBishops) {

        document.getElementById(i).innerHTML = BlackBishop.getSymbol()

        document.getElementById(i).setAttribute("piece",BlackBishop.getName())

        document.getElementById(i).setAttribute("player",BlackBishop.getColor())

    };

    document.getElementById("d8").innerHTML = BlackQueen.getSymbol()

    document.getElementById("d8").setAttribute("piece",BlackQueen.getName())

    document.getElementById("d8").setAttribute("player",BlackQueen.getColor())

    document.getElementById("e8").innerHTML = BlackKing.getSymbol()

    document.getElementById("e8").setAttribute("piece",BlackKing.getName())

    document.getElementById("e8").setAttribute("player",BlackKing.getColor())

    WhitePawns = ["a2","b2","c2","d2","e2","f2","g2","h2"];

    for (let i of WhitePawns) {

        document.getElementById(i).innerHTML = WhitePawn.getSymbol()

        document.getElementById(i).setAttribute("piece",WhitePawn.getName())

        document.getElementById(i).setAttribute("player",WhitePawn.getColor())

    };

    WhiteRooks = ["a1","h1"]

    for (let i of WhiteRooks) {

        document.getElementById(i).innerHTML = WhiteRook.getSymbol()

        document.getElementById(i).setAttribute("piece",WhiteRook.getName())

        document.getElementById(i).setAttribute("player",WhiteRook.getColor())

    };

    WhiteKnights = ["b1","g1"]

    for (let i of WhiteKnights) {

        document.getElementById(i).innerHTML = WhiteKnight.getSymbol()

        document.getElementById(i).setAttribute("piece",WhiteKnight.getName())

        document.getElementById(i).setAttribute("player",WhiteKnight.getColor())

    };

    WhiteBishops = ["c1","f1"]

    for (let i of WhiteBishops) {

        document.getElementById(i).innerHTML = WhiteBishop.getSymbol()

        document.getElementById(i).setAttribute("piece",WhiteBishop.getName())

        document.getElementById(i).setAttribute("player",WhiteBishop.getColor())

    };

    document.getElementById("d1").innerHTML = WhiteQueen.getSymbol()

    document.getElementById("d1").setAttribute("piece",WhiteQueen.getName())

    document.getElementById("d1").setAttribute("player",WhiteQueen.getColor())

    document.getElementById("e1").innerHTML = WhiteKing.getSymbol()

    document.getElementById("e1").setAttribute("piece",WhiteKing.getName())

    document.getElementById("e1").setAttribute("player",WhiteKing.getColor())

    document.getElementById("playerTurn").innerHTML = playerTurn

    resetBoard()

}

reset\_game()

## References:

[1] Giovanni Di Luca, 2021, *Are Chess Computers Unbeatable? The Cold Hard Truth*, accessed 3/10/2022, <[https://chesspulse.com/are-chess-computers-unbeatable](https://chesspulse.com/are-chess-computers-unbeatable/)/>

[2] Bryan Xiao, 2020, *Is Ai the End of Chess?* accessed 3/10/2022,

<<https://blog.rebellionresearch.com/blog/is-ai-the-end-of-chess>>

## Appendix A:

Are Chess Computers Unbeatable? The Cold Hard Truth

Chess Trivia / By [Giovanni Di Luca](https://chesspulse.com/author/gio/)

There’s no doubt about it that chess computers have come on leaps and bounds over the years. Chess is a game which lends itself very well to mathematical analysis and thus, it’s a great game to put computers to work on. The question is: can a human play fairly against a computer? And if not, is there any point in humans playing chess anymore?

Are chess computers unbeatable?

Chess computers are now so strong that they are practically unbeatable. It is highly unlikely that even the greatest human players would beat a computer playing at a full capacity. This is because a computer can analyze millions of possibilities and compare them against each other within seconds. No human mind can hope to compete with such analytical powers.

## 

## Appendix B:

Is Ai the End of Chess?

For as long as chess has been around, it has been a way to match two minds against one another, competing to best the other through a combination of analytical skill and intuition.

However, has chess become pointless through the development of increasingly advanced AI chess engines? Is AI the end of chess?

In short, the answer is both yes and no.

It is true that the best chess engines are already unbeatable by humans. Rapid developments are being made to AI chess engines. AlphaZero, currently the strongest engine, utilizes machine learning. AlphaZero is more efficient than Stockfish, the previous AI chess champion, in that AlphaZero searches fewer positions. Furthermore, AlphaZero beat Stockfish after only 4 hours of learning.

In recent years, due to the prevalence of such chess engines, the game has lost much of its uncertainty. In fact, among players rated 2750 and above, the draw rate has increased dramatically over the years. One can see how this increase may seem to be an indicator of chess becoming more and more “solved.” Some argue that the high percentage of draws in high rated chess detracts from the game’s entertainment.

However, at its core, chess is a game that tests the skill of humans. This means even if a game ends in a draw, it doesn’t mean that it was pointless or boring. Chess is about the process and the battle that takes place during the game, not the end result. Yes, it is entirely possible that in the future AI will have optimized every move from start to finish in a game, and then chess will be a matter of pure memorization.