Chess
Artificial Intelligence

Anirudh Saxena- RA1911033010069
Nandhagopan Ramesh- RA1911033010068
Nigel Jonathan Renny- RA1911033010088

Abstract

Chess has been played by the world's masterminds for ages and is the game where the **brain needs a lot of processing**.
Artificial Intelligence has influenced the way in which chess games are played at the top level. Most of the Grandmasters and Super Grandmasters (Rated at a FIDE above 2700) utilize these modern Artificial Intelligence chess engines to analyze their games as well as the games of their competitors.
We have developed a basic interactive chess game using python.

Issues in existing system

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 analyse millions of possibilities and compare them against each other within seconds but Machines are like humans, and everybody makes mistakes; that does not mean they will always make the same mistakes. Technology evolves rapidly, so do machines and artificial intelligence. New tech is developed every now or then; it makes those machines more robust, scalable, making fewer mistakes

Problem Statement

Chess is a game for two players, dubbed White and Black. The goal is to capture your opponent's king. In the game, this is known as a checkmate.Chess is played on a board with 64 squares. Each player begins with 16 pieces, lined up in two rows. The first row is occupied by pieces called pawns. The next row contains: a king, a queen, two rooks, two bishops, and two knights.

Chess is defined as a game of "perfect information", because both players are aware of the entire state of the game world at all times: just by looking at the board, you can see which pieces are alive and where they are located.

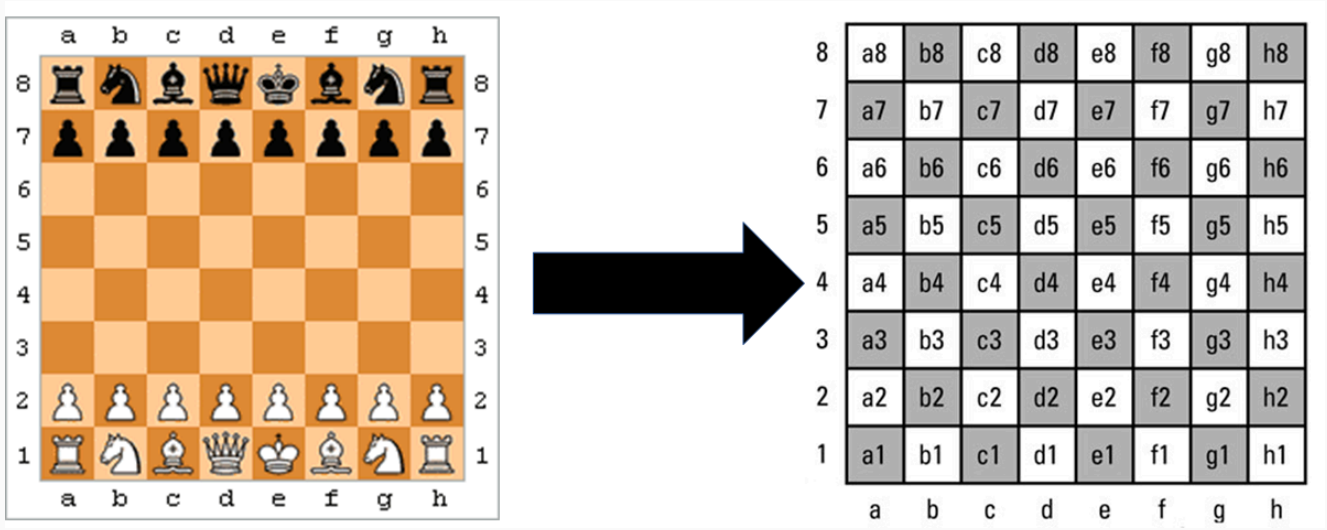
To change chess game from physical form to figurative form fully realistic, Several things are needed to make chess game computerised and intelligent.

Objective

- Some way to represent a chess board in memory, so that it knows what the state of the game is.
- Rules to determine how to generate legal moves, so that it can play without cheating A technique to choose the move to make amongst all legal possibilities, so that it can choose a move instead of being forced to pick one at random.
- A way to compare moves and positions, so that it makes intelligent choices.
- Some sort of user interface

Proposed methodology

Board Representation



Before writing any algorithm for our engine it's important for us to code the logic behind chess pieces i.e. assign every possible legal move to each and every chess piece.Our work has been made a lot simpler by the python-chess library which will provide us with all of the move generation and validation.

Proposed methodology

Board Evaluation

1. Avoid exchanging one minor piece for three pawns.
2. Always have the bishop in pairs.
3. Avoid exchanging two minor pieces for a rook and a pawn.

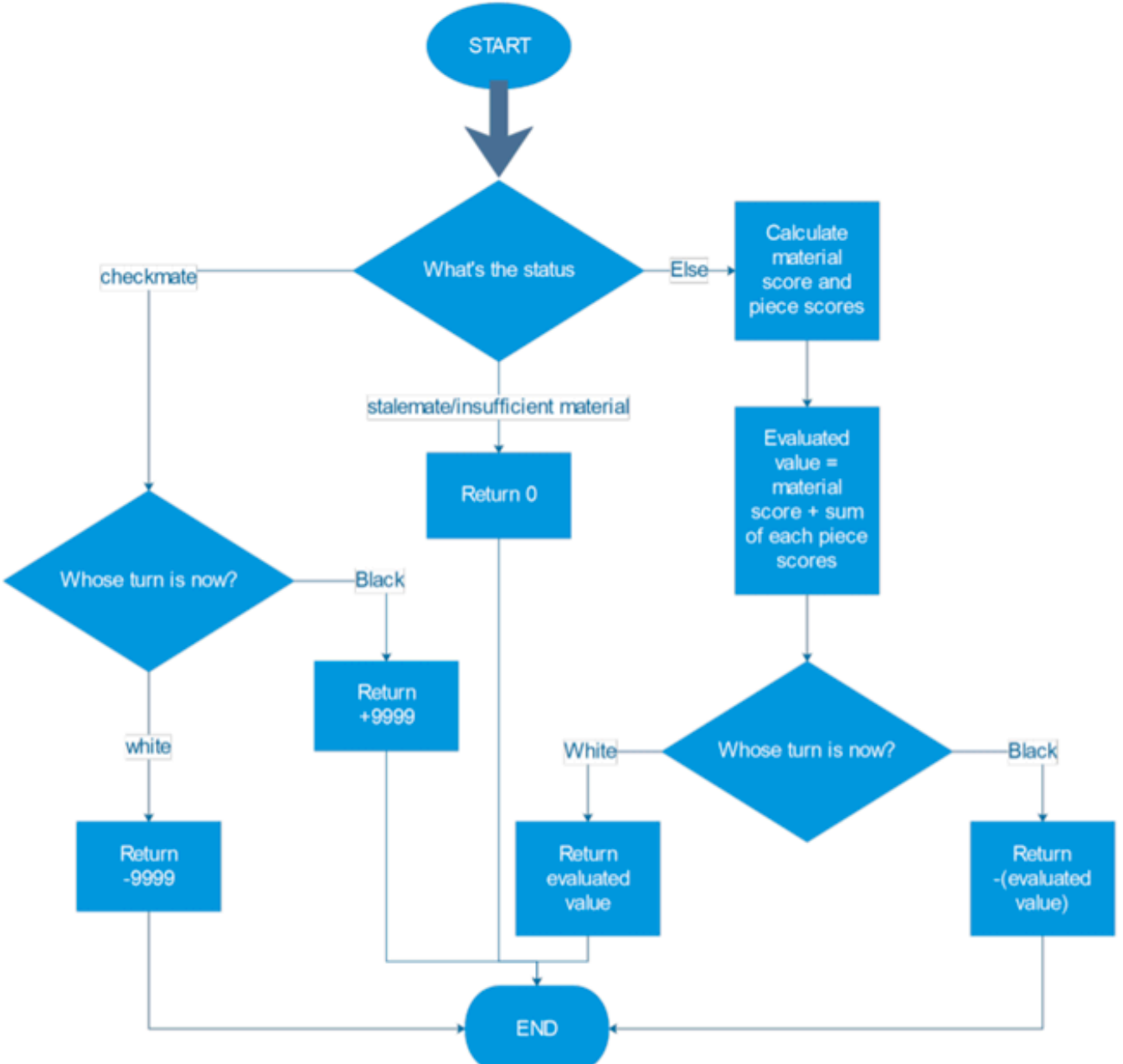
So, the equations from the above insights will be:

1. Bishop > 3 Pawns & Knight > 3 Pawns
2. Bishop > Knight
3. Bishop + Knight > Rook + Pawn

By simplifying the above equations, we get Bishop > Knight > 3 Pawns.
Also, we must convert the third equation as Bishop + Knight = Rook + 1.5 Pawn as two minor pieces are worth a rook and two pawns.

Evaluation Function

Flowchart



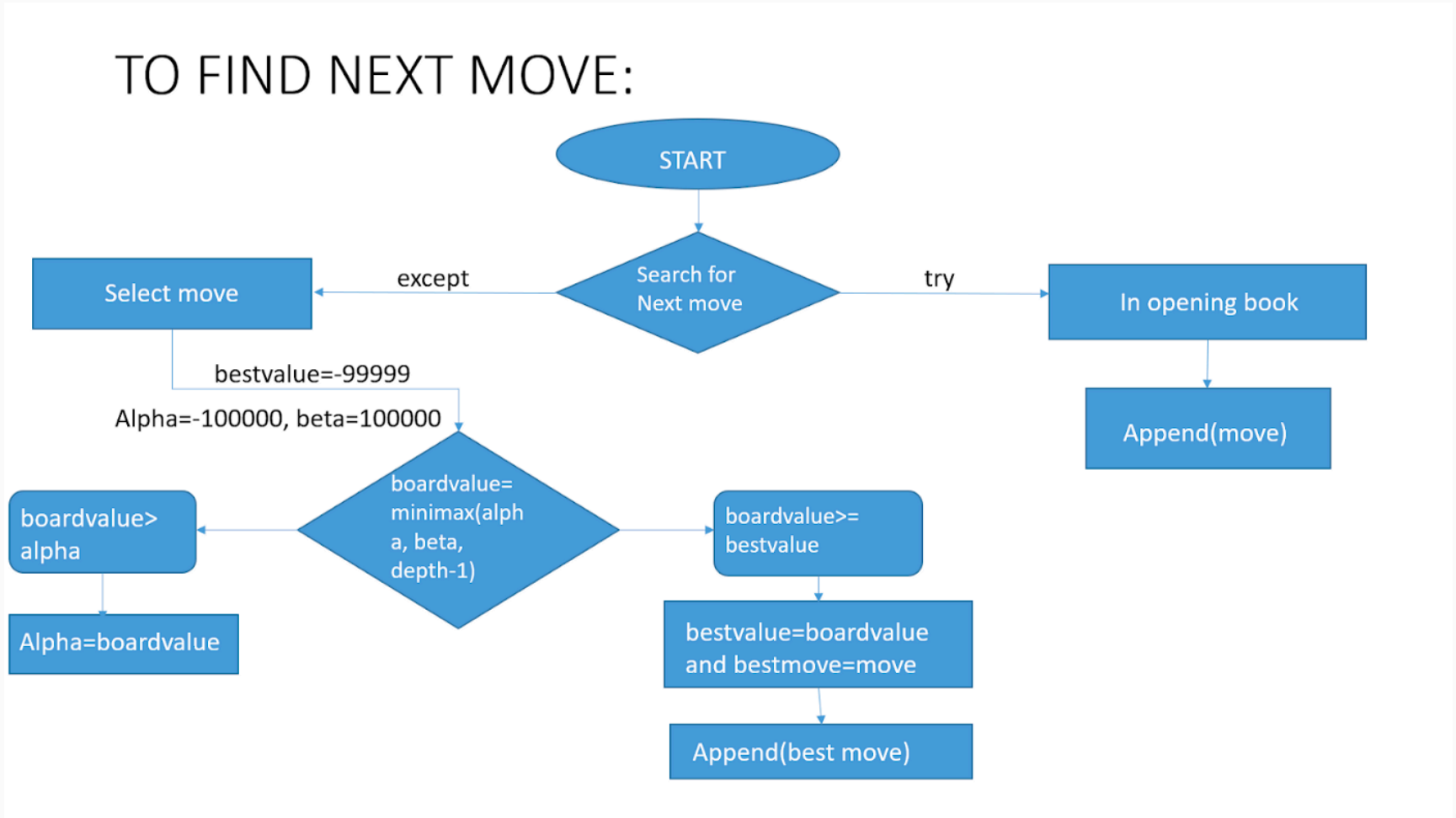
Proposed methodology

Move Selection

This will be the last step for our algorithm where we will use the Negamax Implementation of the Minimax Algorithm which is a mostly used algorithm for any two-player games such as checkers, snake, and ladders, etc. Then will later optimise it using Alpha-Beta pruning which will in turn reduce our execution time.
we will be using the negamax variant of minimax for better results as we will need only one function that is to maximize the utility of both the players. The difference is that here, one player loss is equal to another player's gain and vice versa. In terms of the game, the value of the given position to the first player is the negation of the value to the second player. Initially, we will set alpha as negative infinity and beta is positive infinity so that, both players start with the worst possible score.

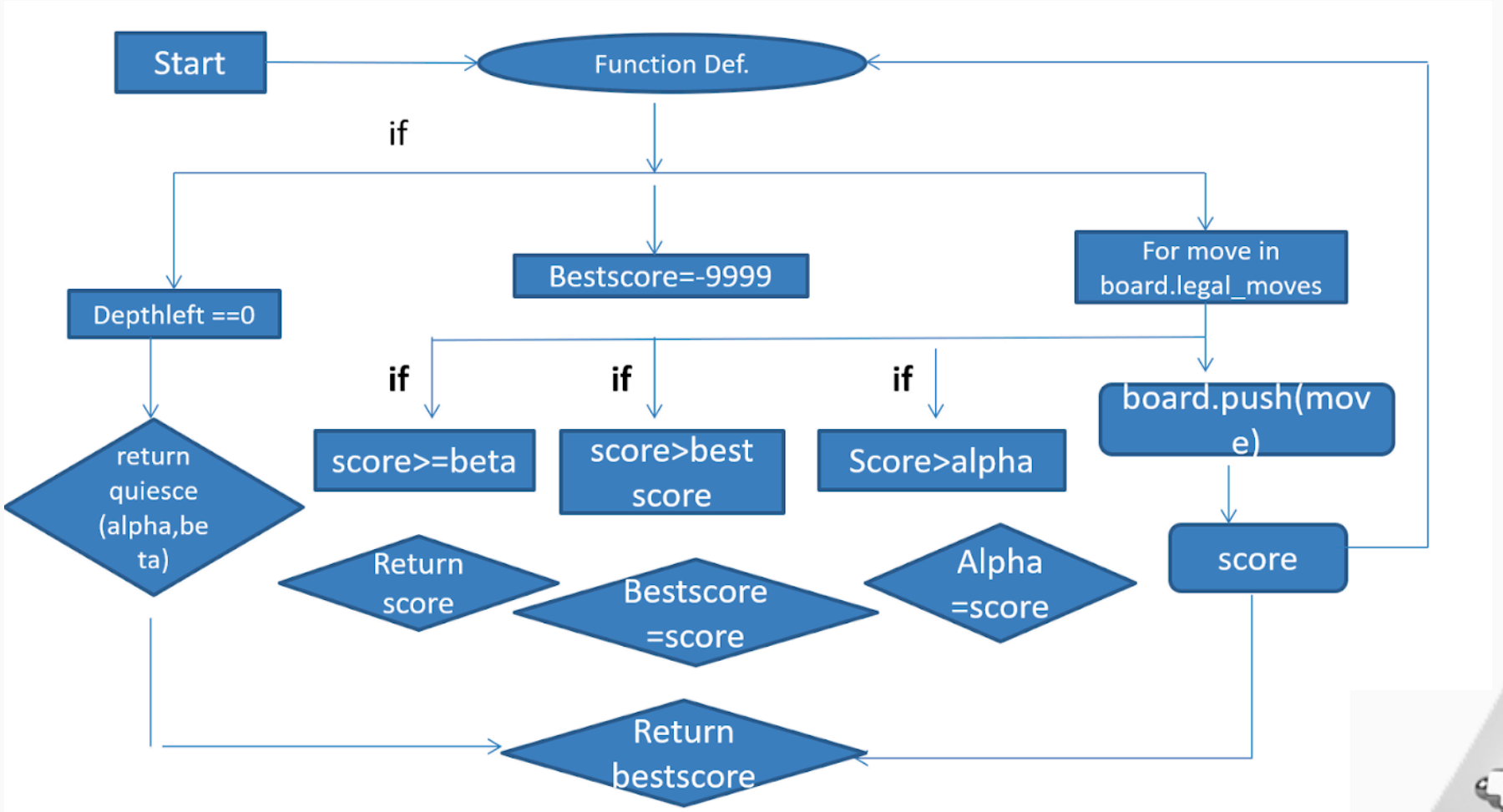
Search Function

Flowchart



AlphaBeta Function

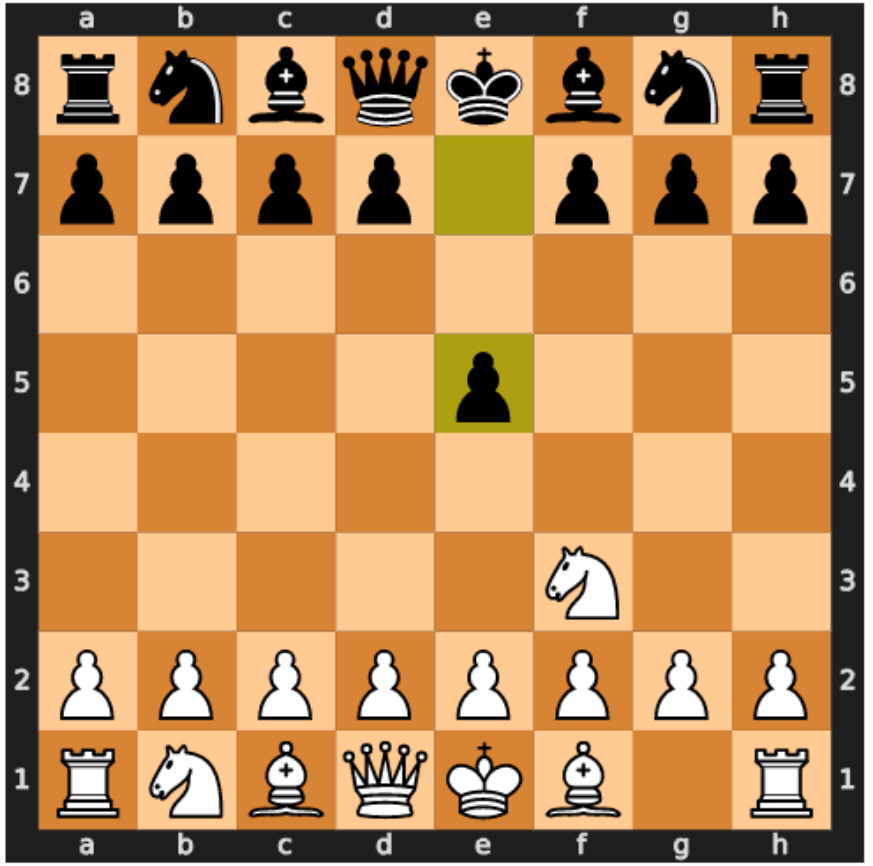
Flowchart



Demo

```
# use this for computer move
mov = selectmove(3)
board.push(mov)
board
```


```
# use this for human move, where e5 is the example move
board.push_san("e5")
board
```

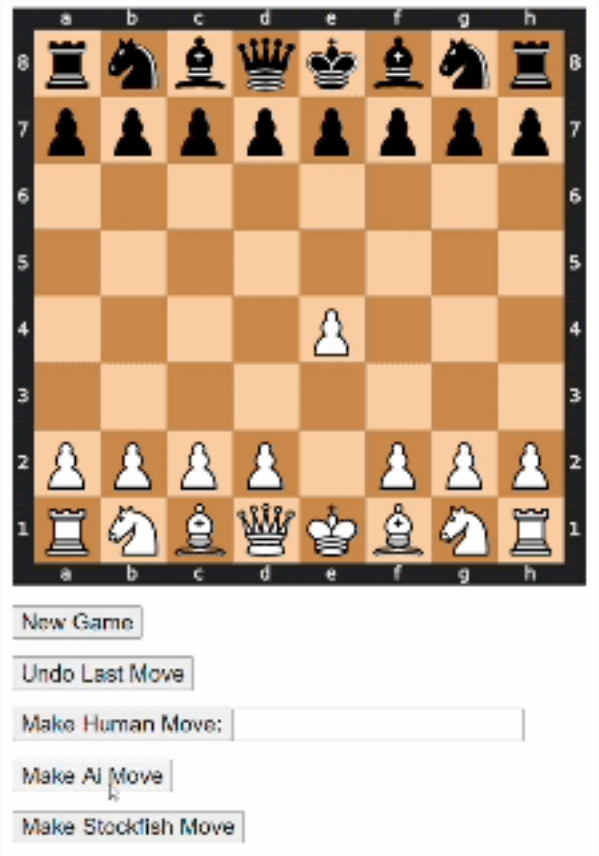


Demo

User input

Move executed

Demo



AI Move

Move executed

Thank you

AI project- Chess