A Mini Project Synopsis on

Chess AI

T.E. - I.T Engineering

Submitted By

Neel Dudheliya	**19104049**
Ayush Jain	**18104005**
Anand Morye	**19104018**

Under The Guidance Of

Prof. Jayshree Jha

DEPARTMENT OF INFORMATION TECHNOLOGY

A.P.SHAH INSTITUTE OF TECHNOLOGY

G.B. Road, Kasarvadavali, Thane (W), Mumbai-400615

UNIVERSITY OF MUMBAI

Academic year : 2021-22

CERTIFICATE

This to certify that the Mini Project report on **Chess AI** has been submitted by Neel Dudheliya (19104049), Ayush Jain (18104005) and Anand Morye (191040) who are a Bonafede students of A. P. Shah Institute of Technology, Thane, Mumbai, as a partial fulfilment of the requirement for the degree in **Information Technology**, during the academic year **2021-2022** in the satisfactory manner as per the curriculum laid down by University of Mumbai.

Prof. Jayshree Jha
Guide

Prof. Kiran Deshpande	Dr. Uttam D.Kolekar
Head Department of Information Technology	Principal

External Examiner(s)
1.
2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

This project would not have come to fruition without the invaluable help of our guide **Prof. Jayshree Jha.** Expressing gratitude towards our HOD, **Prof. Kiran Deshpande,** and the Department of Internet Technology for providing us with the opportunity as well as support required to pursue this project. We would also like to thank our teacher **Prof. Nahid Shaik** who gave us her valuable suggestions and ideas when we were in need of them. We would also like to thank our peers for their helpful suggestions.

TABLE OF CONTENTS

Chapter 1

Introduction

Chess is a two-player strategy board game played on a checkered board with 64 squares arranged in an 8×8 square grid. Played by millions of people worldwide, chess is believed to be derived from the Indian game chaturanga sometime before the 7th century.

Chess has gained tremendous popularity, and it is leading to a revolutionary trend due to more people getting involved with chess at the time of this global pandemic. So, there is no better time to analyze the role of artificial intelligence in improving the quality of chess.

Artificial Intelligence is a revolution in itself with the numerous feats of accomplishments that it has been able to achieve. The use of AI in the real world and real-life scenarios is ample. They have a wide array of use cases to improve the quality of life in general. Another wonderful use of Artificial Intelligence is in the sport of Chess.

1.1 Purpose

Chess, known as the game of "perfect information", since both players are conscious about the entire situation of the game at all time, just by looking at the board, has since been considered as standard of testing intelligence. The kind of victory of a machine over a human grand master has led to make chess one of the most used games to promote artificial intelligence and Computer Science. chess engines are not able to strategically plan moves or to explain why they compute such a combination of moves. Chess is now seeing record numbers of people playing the game even as AI itself continues to get better at playing. These shifts have created a unique testbed for studying the interactions between humans and AI: formidable AI chess-playing ability combined with a large, growing human interest in the game has resulted in a wide variety of playing styles and player skill levels.There's a lot of work out there that attempts to match AI chess play to varying human skill levels, but the result is often AI that makes decisions and plays moves differently than human players at that skill level. The goal for our research is to better bridge the gap between AI and human chess-playing abilities. The question for AI and its ability to learn is: can AI make the same fine-grained decisions that humans do at a specific skill level? This is a good starting point for aligning AI with human behavior in chess.

1.2 Objectives

- To create easy graphical user-interface for user to play chess with AI.
- AI should achieve amateur level gameplay against a human.
- To implement MinMax algorithm for making better decision.
- To implement alpha beta pruning,
- To try and select the best move using AI against the user for upto the depth up to 5.
- To help the user for their training in chess game.
- To help user to increase their problem solving skills.
- To predict what user will do next.

1.3 Scope

The primary goal is to provide a chess game that is intuitive and entertaining for players of all skill levels. User can train himself in single-player offline mode and test its skill against the AI. User can make new strategies and test his pre-game strategies against the AI. Chess can help to increase your problem solving skills. Users can increase their cognitive skills by playing chess.User can select levels according to their difficulty. Chess helps to increase memory power just by visualisation. Chess helps users to increase their concentration power. Users can increase their creativity by playing chess. planning ,reasoning ,and foresight can be improved by playing chess. Users can improve their concentration by playing chess.Chess develops one’s ability to think rationally as well as logically. Players must think ‘outside the box’,in order to increase their odds at winning the game.This type of rational thinking promotes a sense of curiosity and wonder,which is excellent for everybody, especially children. Helps users for pattern recognition. Playing chess not only increases a player's level of mental strength but also imparts a sense of self-esteem and self-confidence when one is able to win at such a challenging game. This skill goes far beyond the chessboard and builds self worth that translates into everyday life,benefitting children,teenagers,and adults. Spatial skills involve the ability to visualize an object in space, picture its fixed position and then mentally manipulate the image. This is key in the game of chess, as players must be able to calculate and analyze variations.

Chapter 2

Problem Definition

Chess, one of the oldest and most popular board games, played by two opponents on a checkered board with specially designed pieces of contrasting colours, commonly white and black. White moves first, after which the players alternate turns in accordance with fixed rules, each player attempting to force the opponent's principal piece, the King, into checkmate—a position where it is unable to avoid capture. The concept of a machine capable of playing chess has been around for hundreds of years. Genuine chess algorithms were available as early as 1948. Turochamp - the name of which is derived from Alan Turing and David Champernowne, the men who invented the algorithm - was implemented by hand, as computer technology at the time was simply incapable of the task. It was not until the late 50s that computer hardware caught up with the ideas of Turing, Champernowne and Shannon and the first programs capable of a full game of chess appeared. These programs were weak players, easily beaten by all but the most inexperienced beginner, but they set in motion a wave of improvements. As computers grew more powerful and available chess techniques became more numerous, the strength of the top programs grew ever greater until eventually even the strongest of grandmasters were considered lucky to hold a draw. What goes into a chess engine? That is, what elements are necessary for a program to play a legal game of chess? All chess engines, however basic, must achieve the following:

- An GUI representation of chessboard that allows for the execution of the given move.
- Finding the legal moves in a given position.
- Accepting (legal) moves from an opponent (user).
- Assessing a position to determine which side is better - evaluation.
- Looking ahead to determine the move that gives it the best position - search.
- Determining when the game has ended.

Chapter 3

Proposed System

The Artificial Intelligence algorithms developed for human play utilize many different kinds of principles. While it is complex to discuss what each engine uses for its functionality and working, we know that a few popular engines like Alpha zero make use of neural networks, deep learning, and neural net-like automation. Leela Chess Zero utilizes an open-source implementation of AlphaZero, which learns chess through self-play games and deep reinforcement learning. In chess, game tree is a directed graph whose nodes are positions in the game and edges are moves the players make. Each node in the game tree has a value and leaf node that the game ends at are labeled with the payoff earned by each player.[4] In the game tree, the player's position in the game is represented by nodes and the edges represent the moves they can make at that position. The Game tree is also important in artificial intelligence because the search algorithm, using the MinMax algorithm or other search algorithms, searches the game tree to find the best move. The complete game tree starts from the initial position and contains all the possible moves in the game. In a complete game tree, the number of leaf nodes is the number of possible different ways the game can be played. Some game trees like tic-tac-toe are easier to search but searching the complete tree in larger games like chess takes a longer time. Instead of searching the complete tree, the chess program searches a partial game tree. It starts from the current position and it searches as many plies as it can in the limited time. Increasing the search depth (number of plies it searches) will result in finding a better move but it takes more time to search.

In computer chess games, the search algorithms are used to find the best move. The search algorithm will look ahead for different moves and evaluate the positions after making each move. Different chess engines use different search algorithms. Some search algorithms include: Minimax algorithm, NegaMax algorithm, NegaScout algorithm, and Alpha-Beta algorithm. We are going to use MinMax algorithm and Alpha-Beta algorithm.

3.1 Features & Functionality

Chess engine is a computer program that decides what move to make during the game. It makes some calculations based on the current position to decide the next move.

- User-friendly GUI created using HTML, CSS & JAVASCRIPT.
- A graphical interface for the chess game has a graphical chess board that allows users to enter moves by clicking on the board or dragging a piece on the board just like a real chess game.
- Chess Engine is build on python using chess.py making it faster.
- GUI and Chess engine are two different programs connected using flask.
- Our game tree has variable depth options ranging from 1 to 5 in turn helping user to decide the difficulty of the AI.
- We have used MinMax algorithm and Alpha-Beta pruning for move evaluation.

Chapter 4

Project Outcomes

This Final Project aimed to create a practical and useable product, which can be considered as software and even an educational tool. A real life situation, which is a chess game in the current case, was modeled without any restrictions and based on its modelers personal understanding. Another aim of this Final Project was to analyze two agent base approaches. It was a fascinating and quite helpful experience for us to observe the differences between the two approaches by producing a practical work rather than conducting just a theoretical research. The users will observe and realize the fact that it is a very open-ended model, allowing users to interact with it using their own imagination, which was the main idea behind all this effort. Hopefully, however, it has managed to lay the groundwork for further study of narrative in video games.

Our Project is very lightweight since most of the heavy lifting is done by the server the client requirements are low i.e Brower only. The client-server architecture helps to achieve this performance and comfort to the user. Since it’s a Web Application user can access the game whenever and wherever he wants providing portability.

Chapter 5

Hardware Requirements

- Operatin System – Windows, Android, IOS
- Processor – Pentium intel or above
- RAM – minimum 256MB
- Storage – 16MB

Software Requirements

- Browser – Chrome or any modern browser
- HTML/CSS – up-to-date browsers for HTML5 & CSS3
- Javascript – ECMAScript 2015 or above
- Python 3+
- Flask (server) framework
- Python libraries -
  - python-chess
- Javascript libraries –
  - Chess.js
  - Chessboard.js

Chapter 6

Project Design

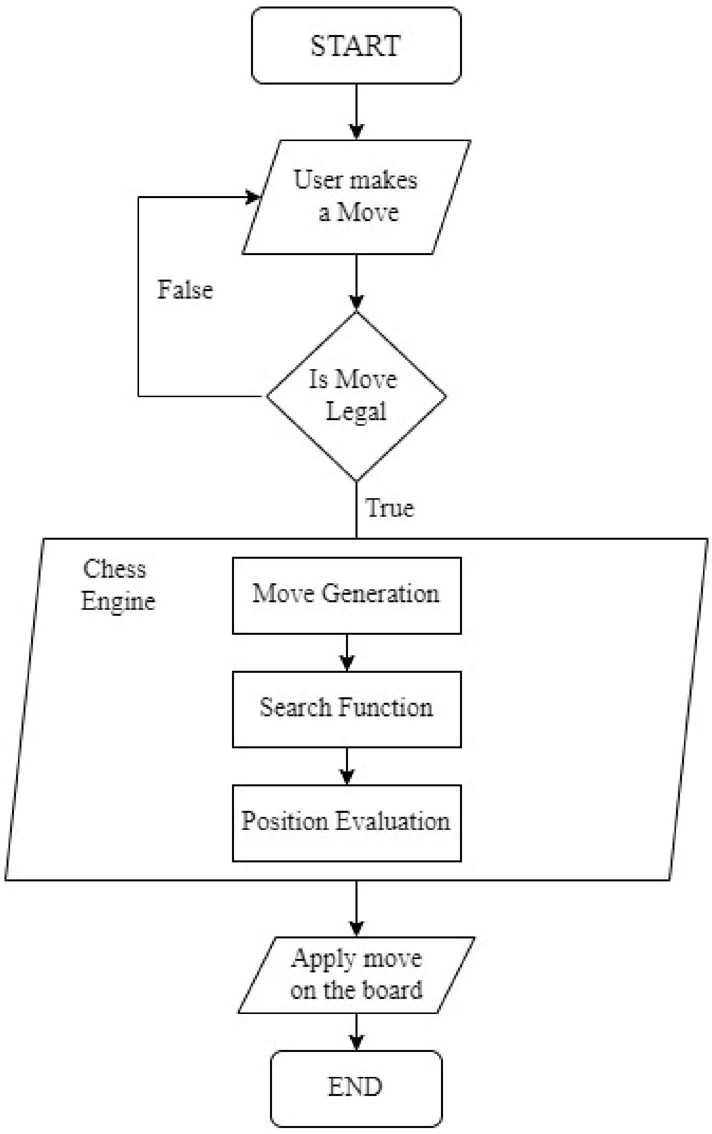


Figure 6.1 Flow Chart of Chess Engine

6.1 Move Generation

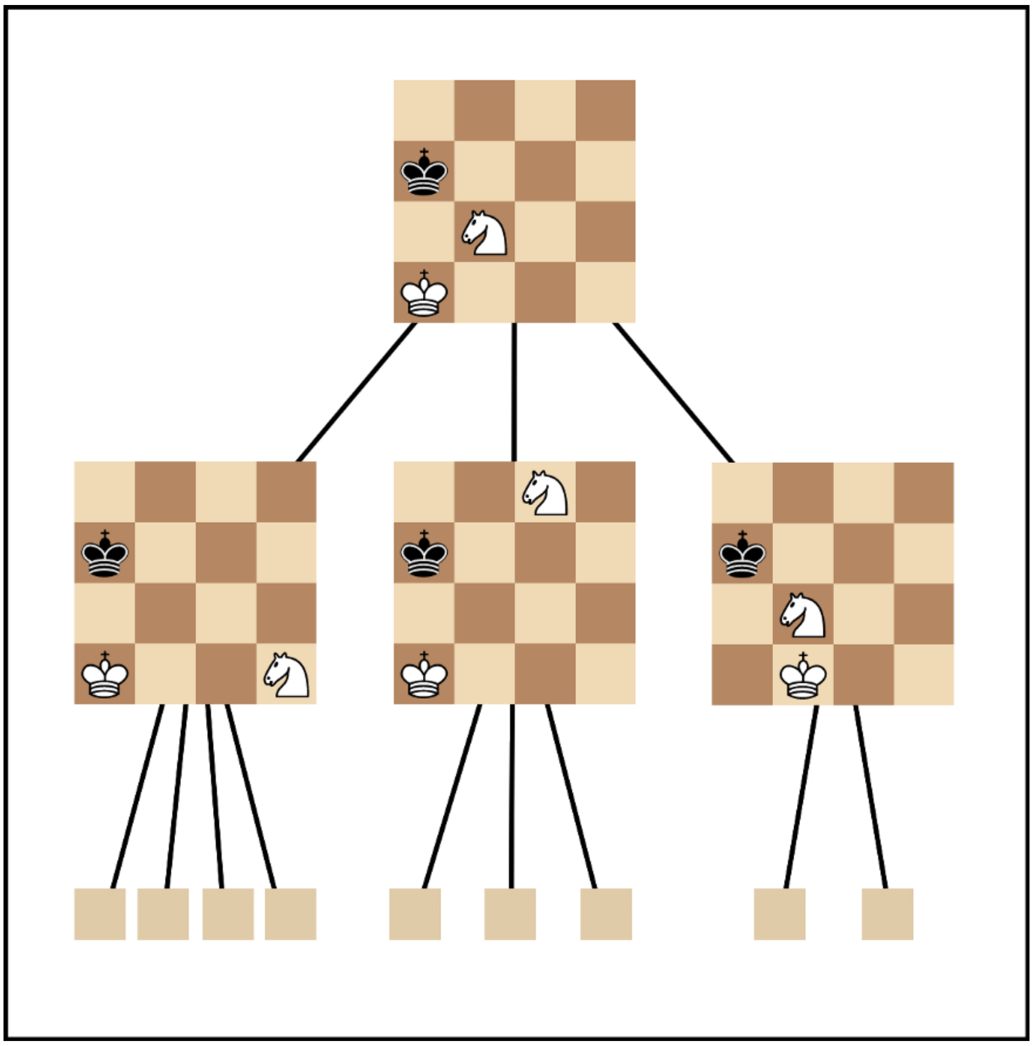


Figure 6.2 Game Tree

We’ll use the python-chess library for move generation, and chessboard.js for visualizing the board. The move generation library basically implements all the rules of chess. Based on this, we can calculate all legal moves for a given board state. Using these libraries will help us focus only on the most interesting task: creating the algorithm that finds the best move.

6.2 Search Function

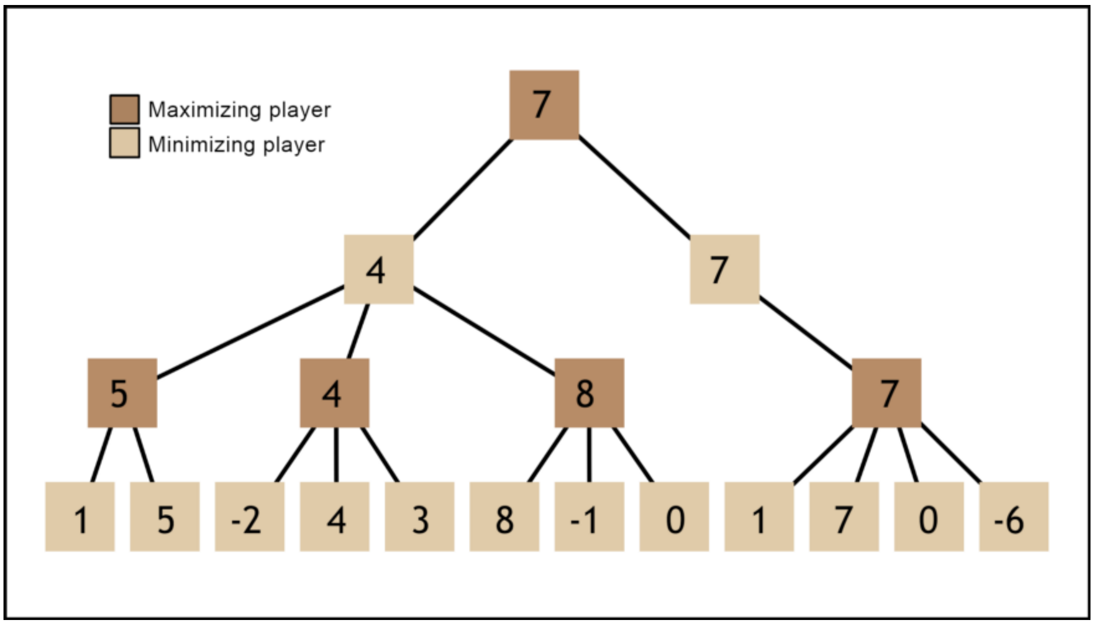


Figure 6.3 Min-Max Algorithm

We’re going to create a search tree from which the algorithm can chose the best move. This is done by using the Min-Max algorithm. In this algorithm, the recursive tree of all possible moves is explored to a given depth, and the position is evaluated at the ending “leaves” of the tree. After that, we return either the smallest or the largest value of the child to the parent node, depending on whether it’s a white or black to move. (That is, we try to either minimize or maximize the outcome at each level.). The effectiveness of the minimax algorithm is heavily based on the search depth we can achieve.

Alpha-Beta pruning is an optimization method to the minimax algorithm that allows us to disregard some branches in the search tree. This helps us evaluate the minimax search tree much deeper, while using the same resources.

The alpha-beta pruning is based on the situation where we can stop evaluating a part of the search tree if we find a move that leads to a worse situation than a previously discovered move. The alpha-beta pruning does not influence the outcome of the minimax algorithm — it only makes it faster. The alpha-beta algorithm also is more efficient if we happen to visit first those paths that lead to good moves.

6.3 Position Evaluation

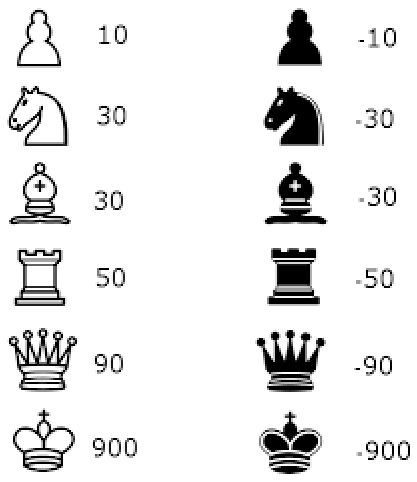


Figure 6.4 Piece value

With the evaluation function, we’re able to create an algorithm that chooses the move that gives the highest evaluation. The only tangible improvement is that our algorithm will now capture a piece if it can. The initial evaluation function is quite naive as we only count the material that is found on the board. To improve this, we add to the evaluation a factor that takes in account the position of the pieces. For example, a knight on the center of the board is better (because it has more options and is thus more active) than a knight on the edge of the board.

Chapter 7

Project Scheduling

Group Member	Time	Work to be done
Ayush Jain	2nd week of February	Literature Review, reading papers, Designing UI and choosing algorithm.
Neel Dudheliya	4rd week of February	Creating the front-end with required options of undoing moves, setting difficulty, etc.
Anand Morye	1th week of March	Learning and testing python-chess library functionalities and properties.
Everyone	3nd week of March	Creating a command line short version of chess engine using python.
Everyone	4th week of March	Connecting the improved chess python engine to the front-end.
Everyone	2nd week of April	Testing and fixing errors and making required changes. Making of Report and Presentation.

Chapter 8

Screenshots of Application

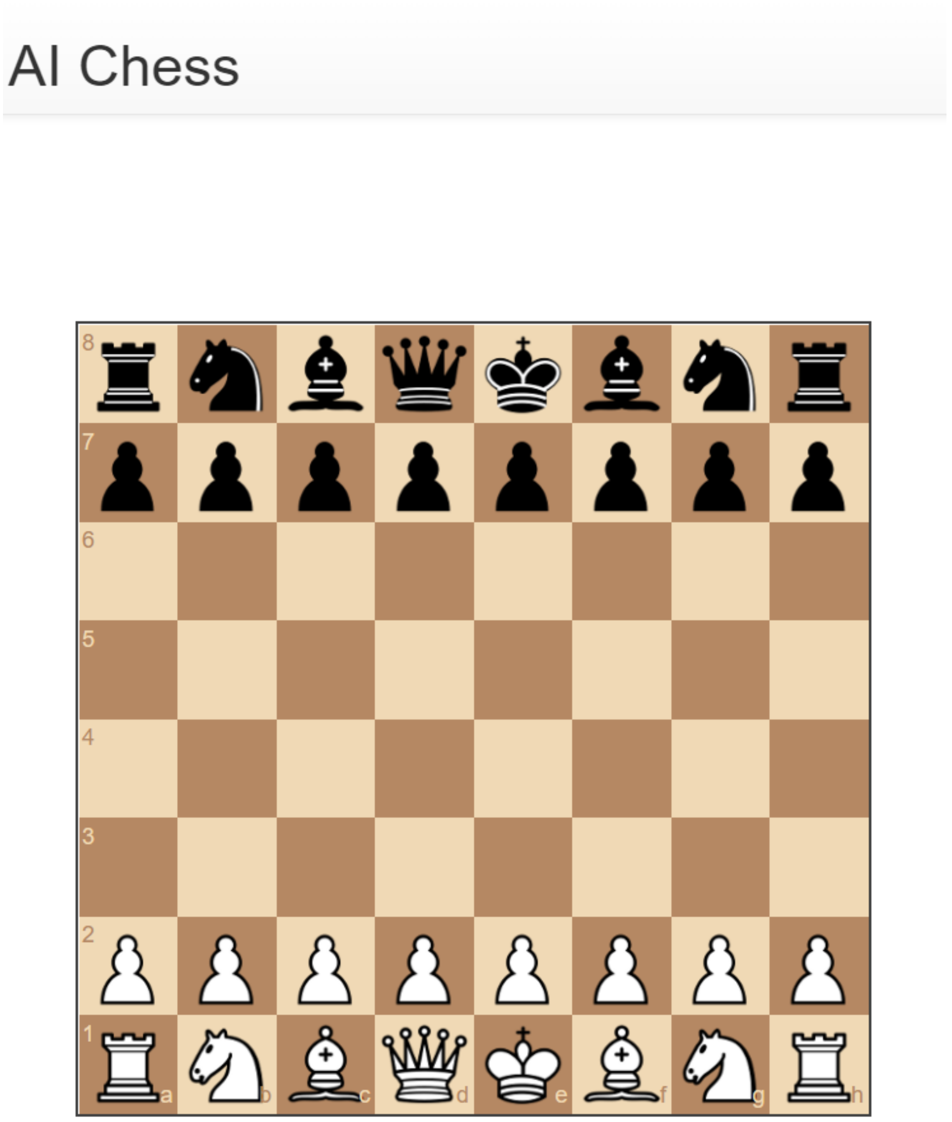


Figure 8.1 User-Interface on browser

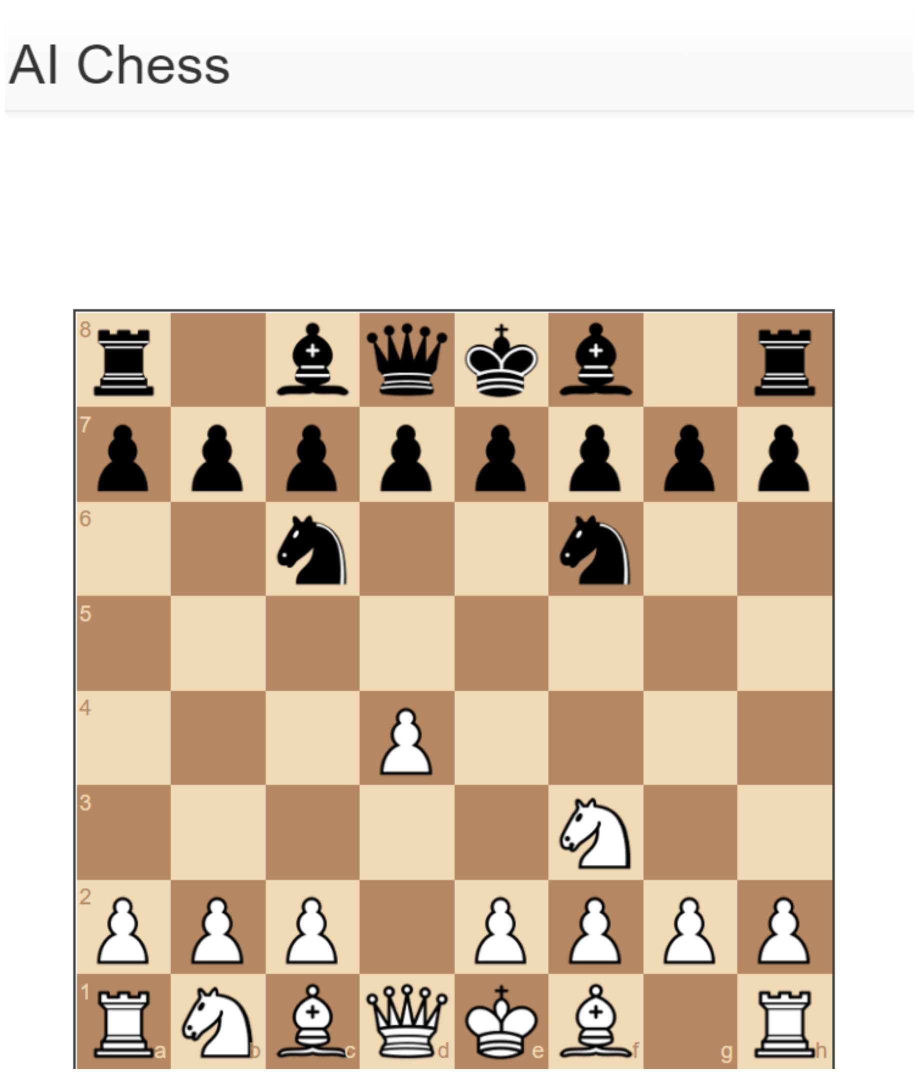
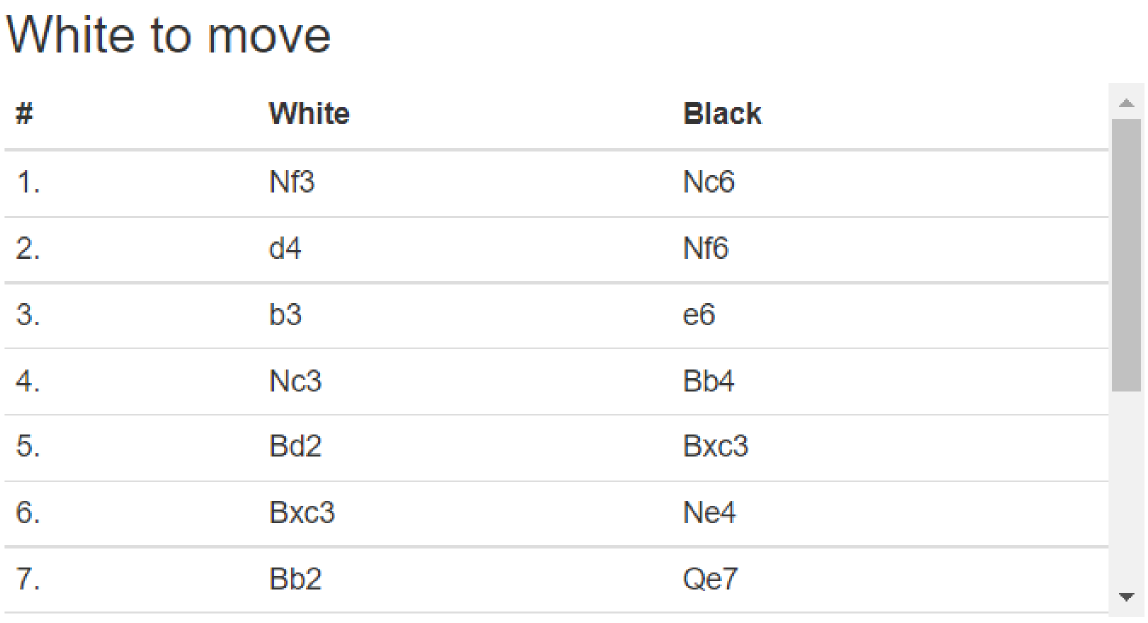


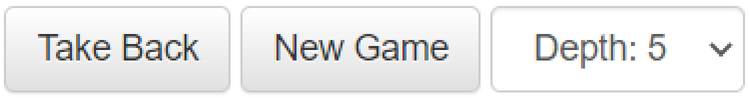
Figure 8.2 User (white) plays a move, AI (black)



White to move

#	White	Black
1.	Nf3	Nc6
2.	d4	Nf6
3.	b3	e6
4.	Nc3	Bb4
5.	Bd2	Bxc3
6.	Bxc3	Ne4
7.	Bb2	Qe7

Figure 8.3 Game state and Move history in tabular format



Depth: 5

Figure 8.4 Game settings with difficulty (depth)

Chapter 9

Conclusion

This Final Project aims to create a functional and usable product, which can be considered as software and become an educational tool. It was a fascinating and quite helpful experience for us to observe the differences between the two approaches by producing practical work rather than conducting just theoretical research. The users will observe and realize the fact that it is a very open-ended model, allowing users to interact with it using their imagination, which was the main idea behind all this effort. Some further improvements are mentioned below:

- Increasing the depth of the algorithm by using Quiescence Search and optimizing it.
- Improving evaluation function so that it does not depend on depth for creating strategies.
- Implementing dynamic depth for higher difficulty levels.
- Improving GUI.

References

- “Iterative deepening depth-first search” Retrieved from: http://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search [May 5, 2015].
- “Board representation (chess)" Retrieved from: http://en.wikipedia.org/wiki/Board_representation_(chess) [December 19, 2014].
- "a-history-of-computer-chess" Retrieved from: http://hightechhistory.com/2011/04/21/ a-history-of-computer-chess-from-the-mechanical-turk-to-Cdeep-blue/ [June 15, 2013].
- Abdelbar, A.M., 2005, Alpha-Beta Pruning and Alth¨ofer’s Pathology-Free Negamax Algorithm, ICCA.
- Putra, Werda & Heryawan, Lukman. (2017). APPLYING ALPHA-BETA ALGORITHM IN A CHESS ENGINE. Jurnal Teknosains. 6. 37. 10.22146/teknosains.11380.