A MAJOR PROJECT REPORT ON

CHESS WITH ARTIFICIAL INTELLIGENCE (A.I.)

Submitted in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

To

Guru Gobind Singh Indraprastha University, Delhi

Under the Guidance of:
Dr. Barkha Bahl

Submitted by:
N. Krishna Khanth
BCA-VI Sem
02220602018

Session 2018 – 2021

TRINITY INSTITUTE OF PROFESSIONAL STUDIES

(Affiliated to Guru Gobind Singh Indraprastha University, Delhi)
Ranked "A+" Institution by SFRC, Govt. of NCT of India
Recognized under section 2(f) of the UGC Act, 1956
NAAC Accredited "B++" Grade Institution

TRINITY INSTITUTE OF PROFESSIONAL STUDIES

(Affiliated to Guru Gobind Singh Indraprastha University, Delhi)
Ranked "A+" Institution by SFRC, Govt. of NCT of India
Recognized under section 2(f) of the UGC Act, 1956
NAAC Accredited "B++" Grade Institution

To Whom It May Concern

I **N. Krishna Khanth**, Enrolment No. **02220602018** from BCA-VI Sem of the Trinity Institute of Professional Studies, Delhi hereby declare that the Major Project Report entitled **"Chess with Artificial Intelligence (A.I.)"** at **Trinity Institute of Professional Studies** is an original work and the same has not been submitted to any other Institute for the award of any other degree.

Date: Signature of the Student

Certified that the Project Report submitted in partial fulfilment of Bachelor of Computer Applications (BCA) to be awarded by G.G.S.I.P. University, Delhi by **N. Krishna Khanth**, Enrolment No. **02220602018** has been completed under my guidance and is Satisfactory.

Date: Signature of the Guide

Name of the Guide:
Dr. Barkha Bahl
Designation:
Professor & Director

Acknowledgement

We would like to take this opportunity to thank The Guru Gobind Singh Indraprastha University (GGSIPU) for having projects as a part of the BCA curriculum. Many people at the university have influenced the shape and content of this project and supported us through it.

We express our sincere gratitude to Dr. Barkha Bahl for mentoring us in this project. She has been an inspiration and role model for this topic. Her guidance and active support have made it possible for us to work on the assignment.

We would like to extend our thanks to the members involved in making this project successful and without whom this project would not have been possible. Also, we would like to express sincere gratitude to our parents and a load of thanks to our friends for helping us throughout the course of the project.

N. Krishna Khanth (02220602018)

Abstract

Chess is a game of strategy and tactics for two players, played on an 8x8 chequered board. Chess is fun and leisure activity that develops perspective, deepens focus, boosts planning skills, etc. Our project implements the chess game with graphical user interface. The chess game follows the basic rules of chess and all the chess pieces only move according to valid moves for that piece. We propose an application which will allow users to play chess against other players as well as computer AI.

Our motive is to make computer think the possible moves and decide a suitable move depending on the difficulty level in the constraints of the game. Moreover, it will provide all the features with a user-friendly interface. The GUI will be understandable and straightforward.

We have compared our project work with existing system on the following parameters:

- UI: User Interface
- UX: User Experience
- Performance

This report discusses the project developed and implemented by us along with the challenges we faced during the development period of the project.

Table of Contents

Table of Figures

List of Tables

List of Abbreviations

S. No.	Abbreviated Name	Full Name
1.	A.I.	Artificial Intelligence
2.	P.G.N.	Post-Game Notation
3.	F.E.N.	Forsyth–Edwards Notation
4.	G.U.I.	Graphical User Interface
5.	U.I.	User Interface
6.	G.I.L.	Global Interpreter Lock
7.	F.I.D.E.	The International Chess Federation
8.	I.D.E.	Integrated Development Environment
9.	D.F.D.	Data Flow Diagram
10.	O.S.	Operating System
11.	T.C.	Test Cases
12.	Q.A.	Quality Assurance

1. Introduction

Chess is a game of strategy and tactics for two players, played on an 8x8 chequered board. Chess is fun and leisure activity that develops perspective, deepens focus, boosts planning skills, etc. Our project implements the chess game with graphical user interface.

The chess game follows the basic rules of chess and all the chess pieces only move according to valid moves for that piece. We propose an application which will allow users to play chess against other players as well as computer AI.

Our motive is to make computer think the possible moves and decide a suitable move depending on the difficulty level in the constraints of the game. Moreover, it will provide all the features with a user-friendly interface. The collection will be understandable and straightforward.

This report discusses the project developed and implemented by us along with the challenges we faced during the development period of the project.

1.1 Idea and Purpose

We suggest an application with GUI which will allow users to play chess with other players as well as a computer AI. This application will provide premium features like high level AI free of cost and minimal computational loads. AI will use several complex algorithms to decide moves and provide a challenging experience to users. This project will allow us to explore field of artificial intelligence and its scope.

1.2 Problem Definition

In general, in today's world every industry is set out to make money. Even in noble sports such as chess, the world's leading chess platforms like chess.com, lichess.com, chessbase.com and chess24.com provide only basic functionality (less powerful A.I. to practice against and for analysis) to average (free) user, lack of U.I. Customization options in terms of colour scheme, setting up matches with time odds and require premium subscription to be able to overcome these limitations.

People who cannot afford to pay for premium subscriptions will miss out on technical advancement in such areas.

1.3 Existing Methods

Presently it has been observed that, the following methods are used in chess application:

1. Limited or no UI customisation – Present applications for chess provide almost no or minimal customisation to its UI like Theme changes, colour scheme changes.
2. Expensive features – Advanced features like high level AI are not free and expensive, most effective A.I. are locked under premium memberships along with features such as setting up match with time odds, saving PGN or FEN Notations of a game.
3. Poor AI – Many free of cost AI provide no challenge for even an average player. They are designed in a way so that they calculate their best moves ineffectively.

1.4 Motivation

This application's motivation arose due to the current money-making trend in game industry:

1. Players are stuck with provided UI for a given application.
2. Game becomes tedious if it does not provide enough of a challenge.
3. Post-game notation is not available in many applications.
4. Players cannot set maximum duration of game.

As players, we feel that these problems cause discomfort in gaming experience.

1.5 Suggested Solutions

We suggest the following solutions to the above-discussed problems:

1. A free of cost chess application with AI functionality.
2. Several option to customise UI of the application.
3. Multiple levels of difficulty for AI.
4. Time control in the hands of the user.

2. System Requirement Analysis

2.1 Project Plan

The project management plan has been broken down in the following parts:

2.1.1. Stakeholders and Expectations

Technical Team: Have ready access to individuals with the authority to make decisions regarding events.

Project Manager: Have an application in the form of a desktop application.

Client: Gain an application which anybody can use easily and play chess.

2.1.2. Project Priorities and Degrees of Freedom

The main focus of the project is to develop a chess application with A.I. integrated in it, on time and within budget. Further we aim to conduct a comprehensive testing of the system to detect bugs and defects. Another priority area is proper allocation of resources and team members to the various roles and responsibilities that arise during the software development.

2.1.3. Approach

We will be following an Incremental approach for this project. The first iteration will focus on basic functionality of the application and subsequent iterations will depend upon that and incorporate more features as time allows based on their priority and importance to the whole application.

2.1.4. Assumptions

- Fair play among players.
- Feasibility of interfacing the chess application with the other applications.
- Players are present physically on same computer.

2.1.5. Success Criteria

The project will be considered a success if the team delivers an operational prototype at the end of the semester with the earlier mentioned features.

2.1.6. Risks and Obstacles to Success

The risks and hurdles that might occur during the development of the G.U.I and implementation of operational chess rules. The fact that python has a Global Interpreter Lock (GIL) which limits the resource allocated to the program, tkinter the G.U.I used for developments issue with memory management and other risks include power failures etc.

2.1.7. Scope

This project will consist of creating an application for playing chess against A.I. and other players.

The project will provide functionality of:

- Up to two players can play chess in this application.
- Single player can play against an A.I.
- Player can obtain PGN or FEN notation of the match.
- Player can play timed matches.

2.2 Software Requirement Specifications

Software Requirement Specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide. It establishes the basis for an agreement between customers and the software providers on what the software product is to do and what it is not expected to do so that there is no room for confusion in the future. If used appropriately, SRS can help prevent software project failure.

2.2.1. Functional Requirements

Our proposed system has the following requirements:

Proposed system has the following requirements:

1. Game will follow rules of chess as provided by International Chess Federation FIDE.
2. Players should be able to move chess pieces using mouse.
3. PGN notation of each move should be displayed at time of gameplay.
4. A file should be saved after each game containing PGN (Post Game Notation) of that game.
5. Players should be able to play with other players in local machine.
6. Players should be able to play with computer AI.
7. Players should be able to choose from at least 3 difficulty levels.
8. Proper message should be displayed for every special event like victory, checkmate, stalemate and draw.
9. A clock should be displayed for total time remaining since beginning of the game.
10. Players should be able to set a time limit of the game.

2.2.2. Non-Functional Requirements

Performance Requirements:

- The system needs to be reliable.
- If unable to process the request, then appropriate error message.
- Windows are loaded within few seconds.

2.3 Tools and Technologies

The following tools and technologies are expected to be used in development. Further may be added as the operations are implemented.

2.3.1. Development Language

2.3.1.1. Python 3.9

Python is a general-purpose programming language that can be used on any modern computer operating system. It can be used for processing text, numbers, images, scientific data and just about anything else you might save on a computer. It is used daily in the operations of the Google search engine, the video-sharing website YouTube, NASA and the New York Stock Exchange. These are but a few of the places where Python plays important roles in the success of the business, government, and non-profit organizations; there are many others.

2.3.1.2. Pip 21.1.2

pip is a de facto standard package-management system used to install and manage software packages written in Python. Many packages can be found in the default source for packages and their dependencies — Python Package Index. Most distributions of Python come with pip preinstalled.

2.3.2. Development Platforms

2.3.2.1. PyCharm 2021.1.2

PyCharm is an Integrated Development Environment (IDE) used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

2.3.2.2. Atom 1.52.0 x64

Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies. Most of the extending packages have free software licenses and are community-built and maintained. Atom is based on Electron (formerly known as Atom Shell), a framework that enables cross-platform desktop applications using Chromium and Node.js. It is written in CoffeeScript and Less.

2.3.2.3. Jupyter Notebook 6.3.0

Jupyter Notebook also called IPython Notebooks is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

2.3.3. Libraries

Library	Version
certifi	2021.5.30
chardet	4.0.0
chess	1.5.0
docopt	0.6.2
idna	2.10
MouseInfo	0.1.3
Pillow	8.2.0
pip	21.1.2
PyAutoGUI	0.9.52
PyGetWindow	0.0.9
PyMsgBox	1.0.9
pyperclip	1.8.0
PyRect	0.1.4
PyScreeze	0.1.27
python-dateutil	2.8.1
PyTweening	1.0.3
pytz	2021.1
requests	2.25.1
setuptools	57.0.0
six	1.16.0
Tkinter	8.6
tktimer	0.1.0
urllib3	1.26.5
yarg	0.1.9

Table 1: List of Libraries

2.4 Hardware Requirements

The following are the minimum hardware requirements that are recommended to be fulfilled in order to run this software.

1. CPU: Intel core i3 3rd Generation / AMD FX-6100
2. RAM: 2 GB
3. GPU: Integrated Graphics
4. Storage: 1 GB

2.5 Software Requirements

The following software requirements are recommended to be fulfilled in order to run this software.

1. OS: Any Operating System
2. Programming Language: Python

3. System Feasibility Study

3.1 Feasibility Study

This is an Assessment of the feasibility of our project.

3.1.1. Economic Feasibility

The project is economically feasible as it works with functions with low-cost services such as laptops and desktops.

3.1.2. Technical Feasibility

The current project is technically feasible as the application requires:

1. Any python supported IDE
2. GUI development tools

All these are readily available and can be successfully deployed on any available computer.

3.1.3. Behavioural Feasibility

The application is behaviourally feasible since it requires no technical guidance; all the modules are user friendly.

3.1.4. Operational Feasibility

The application is operationally feasible as:

1. Complete GUI-Base, which is user friendly.
2. Inputs to be taken are self-explanatory.
3. The system cuts down the cost of clients.

4. System Design

4.1 Use Case

Use Case Model

The Use Case Model describes the proposed functionality of our system. The diagram represents a discrete unit of interaction between the player (user) and the application.

The project's use case model consists of the following type of actors:

- Main Actors – Player
- Supporting Actors – A.I.

The primary modules (different functions) are:

- Match Configuration
- U.I. Configuration
- Clock Management
- Chess Board

The model is given below:

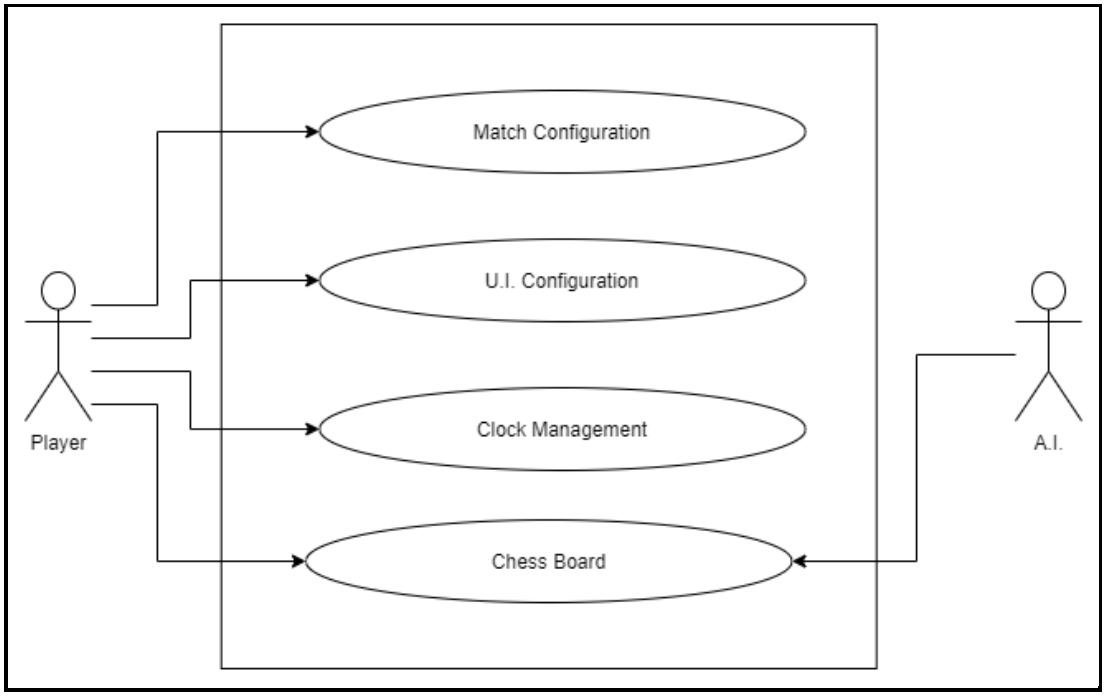


Figure 1: Use Case Diagram

Our project has the following use case specifications:

4.1.1. Use Case: Match Configurations

Description: This use case takes care of determining who the opponent is. Whether it's against a human in the same local machine or if it is human vs AI. If the match is against ai it also determines which colour the human and AI will get.

Primary Flow:

1. User enters who the match is against (default is human).

Alternate Flow at: Opponent selection.

1. The user selects ai as opponent.
2. The user selects colour of choice.
3. The user selects AI strength.
4. These inputs are sent to backend.

Alternate Flow E1: Player colour not selected.

1. The user is prompted to select colour.
2. User may select colour or change opponent.
3. This data is sent to the backend.

Alternate Flow E2: AI strength not set.

1. The user is prompted to select AI strength.
2. User may select AI strength or change opponent.
3. This data is sent to the backend.

4.1.2. Use Case: Clock Management

Description: This use case takes care of setting the clock timer for both players, may it be human or ai.

Note: to change the clock timers the game has to be restarted.

Primary Flow:

1. User provides clock time in minutes for white (default is 1).
2. User provides clock time in minutes for black (default is 1).
3. These inputs are provided to the backend.

4.1.3. Use Case: UI Configuration

Description: This use case takes care of setting the colour scheme of the board.

Note: board colour scheme maybe changed while in-game.

Primary Flow:

1. User provides primary colour scheme (default is brown).
2. User provides secondary colour scheme (default is white).
3. These inputs are provided to the backend.

4.1.4. Use Case: Chess Board

Description: This use case takes care of creation of the board and setting its characteristics as provided by the backend. This acts as the front end of the app.

Primary Flow:

1. The user input is in the form of mouse click.
2. The user clicks on the piece to be moved.
3. The front end contacts the backend via an API and provides all moves available to that piece.
4. The user then provides another input is in the form of mouse click.
5. This input is the destination of the earlier selected piece.
6. This is updated in the backend.

Alternate Flow 1: Different piece of same colour selected.

1. Previously selected piece is dropped.
2. New input click is accepted.
3. The front end contacts the backend via an API and provides all moves available to that piece.
4. The user then provides another input is in the form of mouse click.
5. This input is the destination of the earlier selected piece.
6. This is updated in the backend.

Error Flow E1: Destination square is invalid.

1. Move isn’t made.
2. User has to provide new input within valid destinations.

Error Flow E2: Destination is illegal.

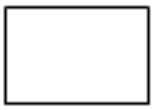
1. Move isn’t made.
2. User has to provide new input in legal destinations.

4.2 Flow Chart

A flowchart is a picture of the separate steps of a process in sequential order. This flow chart depicts the control flow in our project. How the flow of control in our project changes with every non-similar actions. Given below are the component description of flow chart objects used in project.

Flow Chart Components:

1. Process

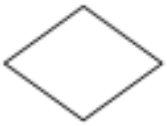


Rectangle - One step in the process. The step is written inside the box. Usually, only one arrow goes out of the box.

2. Flowline (Arrowhead)

Arrow - Direction of flow from one step or decision to another.

3. Decision



Diamond - Decision based on a question. The question is written in the diamond. More than one arrow goes out of the diamond, each one showing the direction the process takes for a given answer to the question. (Often the answers are "yes" and "no.")

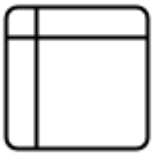
4. Terminal

Circle or oval - Indicates the beginning and ending of a program or sub-process. Represented as a stadium, oval or rounded (fillet) rectangle. They usually contain the word "Start" or "End", or another phrase signalling the start or end of a process, such as "submit inquiry" or "receive product".

5. Input / Output

Shows a conditional operation that determines which one of the two paths the program will take. The operation is commonly a yes/no question or true/false test. Represented as a diamond.

6. Internal Storage



File represents storage of a file in internal storage.

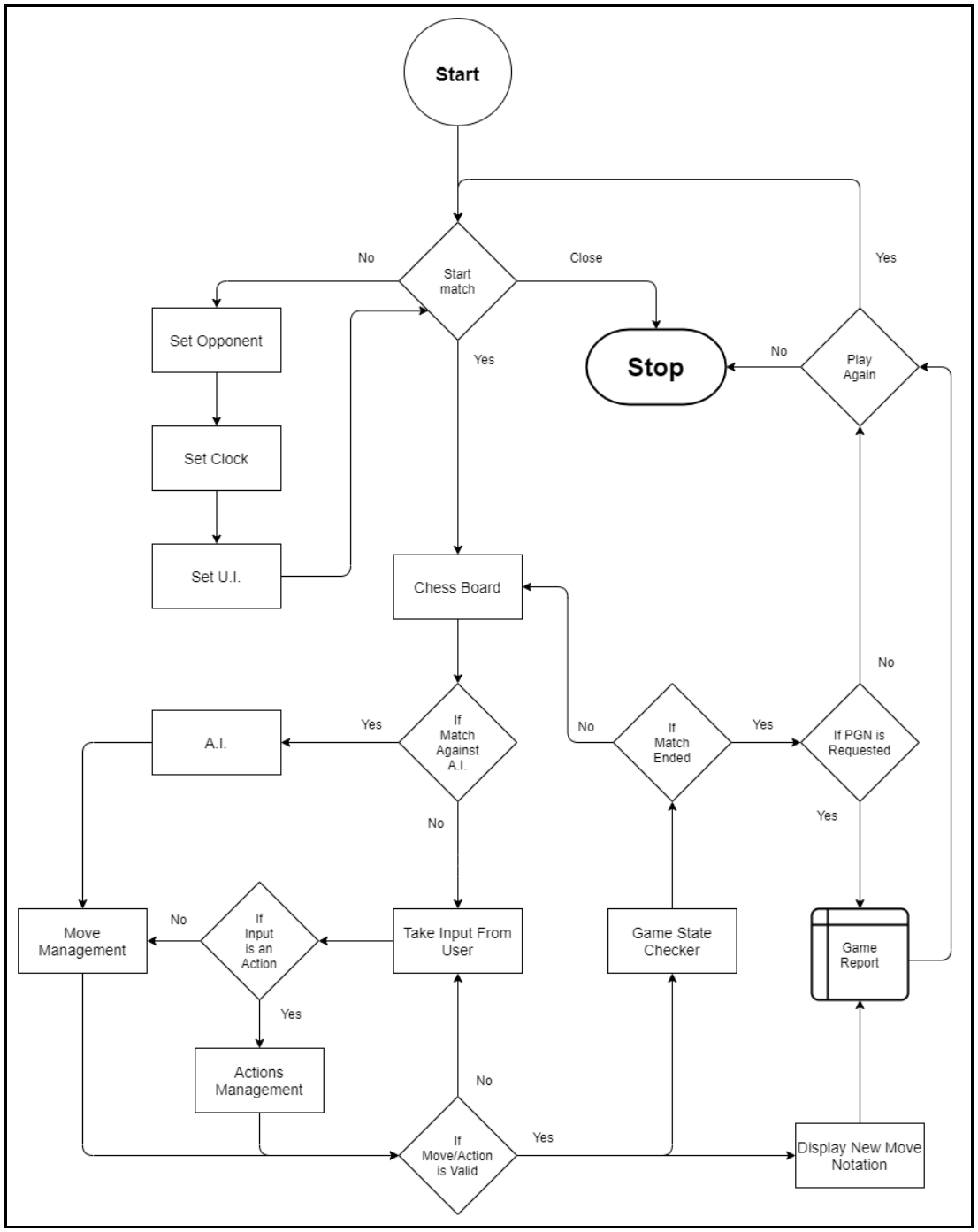


Figure 2: Flow Chart

4.3 Data Flow Diagram (DFD):

Following are the various levels of data flow diagram of our project to better explain its internal structure and flow of data within various modules, how they are processed and how they affect the control flow of the project.

4.3.1 Level 0:

Inputs:

1. Match Data

   User chooses opponent, set clock time and UI of board.

2. Actions

   Actions performed by user to resign, draw offer, save PGN, save FEN.

3. Moves

   Chess moves played by a user like moving a piece and capture moves.

Outputs:

1. Game State

   Different states in during or at end of game like checkmate, timeout, stalemates and draws.

2. Chess Board

   Graphical representation of chess board so users can see the board and the pieces.

3. Notation

   Notation in form of PGN & FEN for moves made by user during a game.

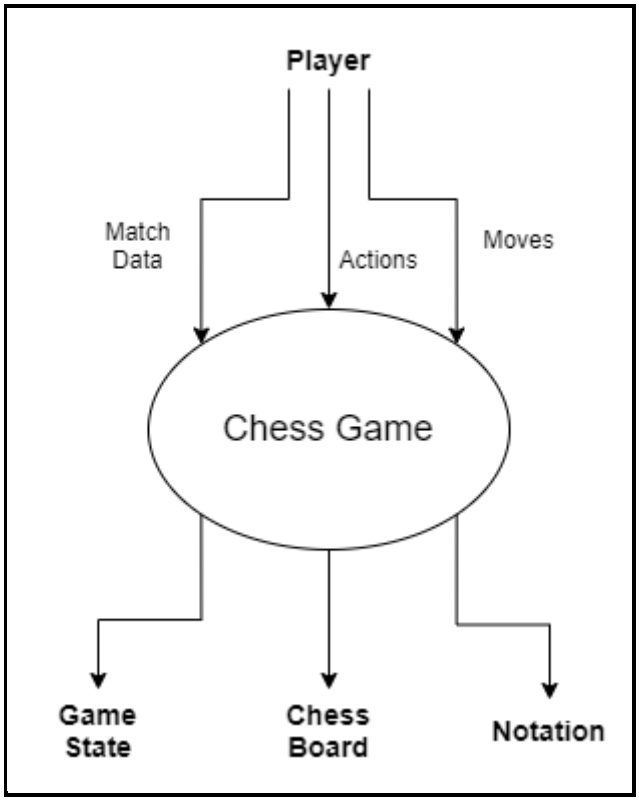


Figure 3: DFD Level 0

4.3.2 Level 1:

1. Game configuration system

   Takes opponent, clock and UI as input to be applied on board.

2. Move management system

   Takes player move as input and checks if it is valid then checks if the move is legal.

3. Chess board management system

   Observes and computes the current state of the board based on previous actions, moves and configurations as input.

4. Chess board

   Reflects the result of managing the board in the board.

5. AI

   Takes the move made by opponent (human) as input to compute best move for itself and returns its best moves notation.

6. Report

   Takes the move made by players as input and converts its notation into PGN & FEN formats that can be saved by user.

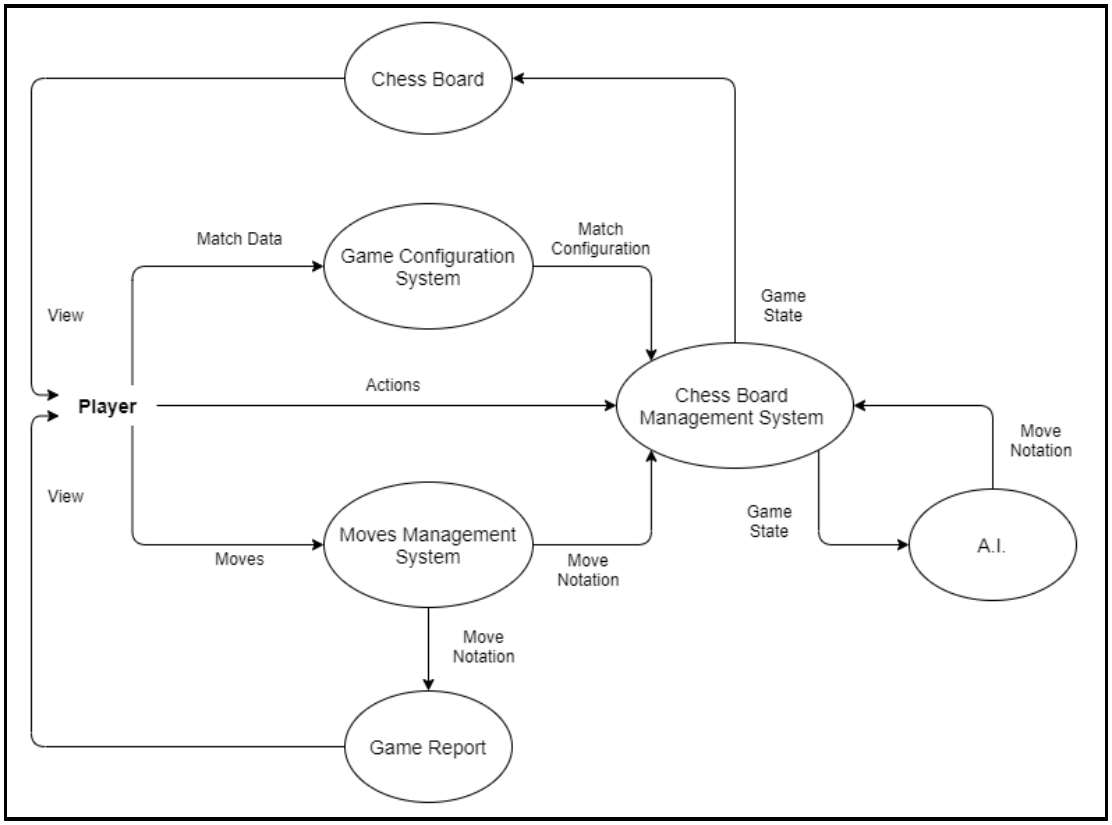


Figure 4: DFD Level 1

4.3.3 Level 2:

4.3.3.1 Game configuration system:

- Clock settings
- UI settings
- Opponent selection
- Loading the board state

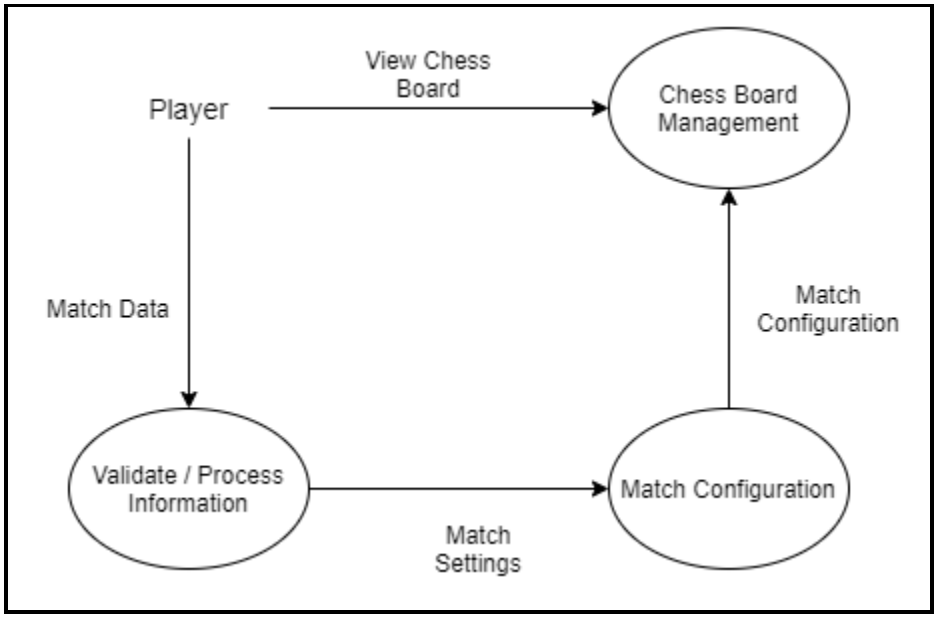


Figure 5: DFD Level 2 - Game Configuration System

4.3.3.2 Chess board management:

It validates piece movement according to rules of chess:

- Making a move
- Move notation generation
- Ending the game
- Updating the board state
- Loads the new layout

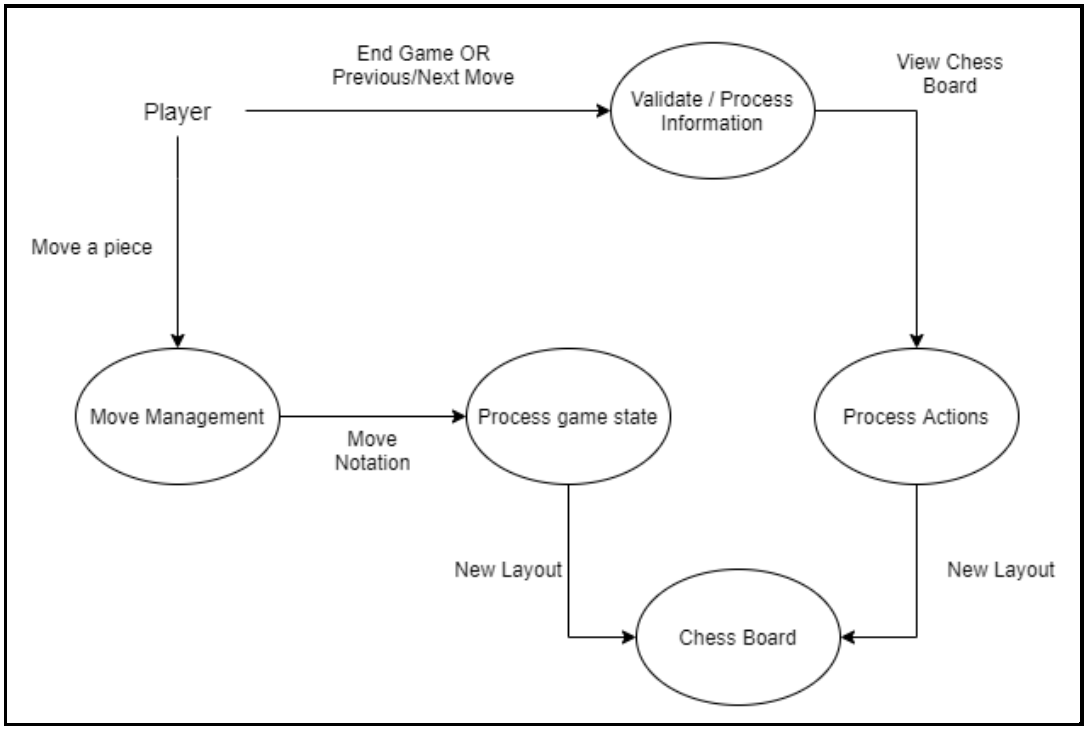


Figure 6: DFD Level 2 - Chess Board Management System

4.3.3.3 Move management system:

- Moving a piece
- Checking validity of move
- Checking legality of move
- Generate move notation
- Updating the board state

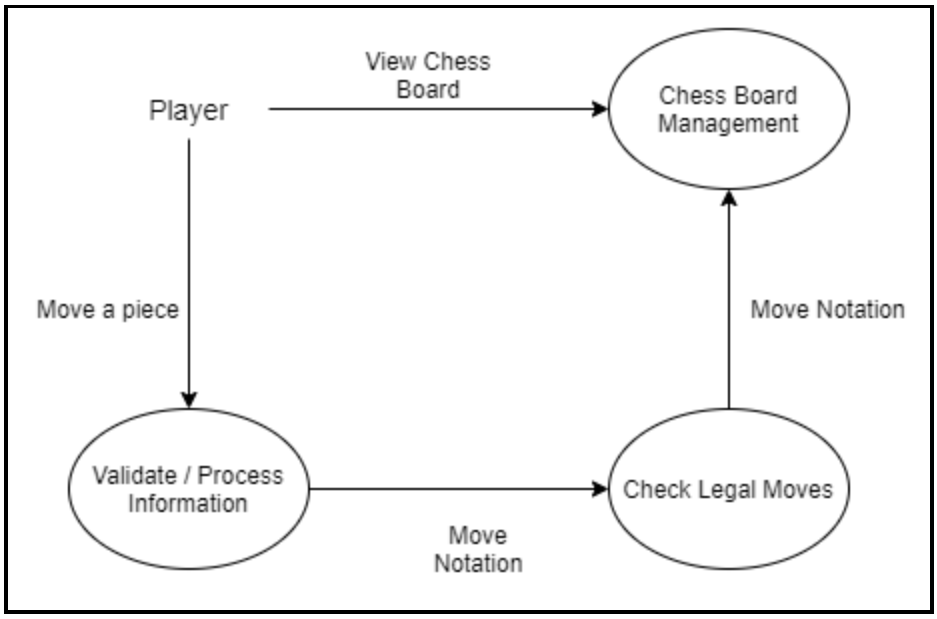


Figure 7: DFD Level 2 - Moves Management System

4.3.3.4 A.I.:

- Analysis of human made move
- Computing of best move
- Generating move notation
- Updating board state

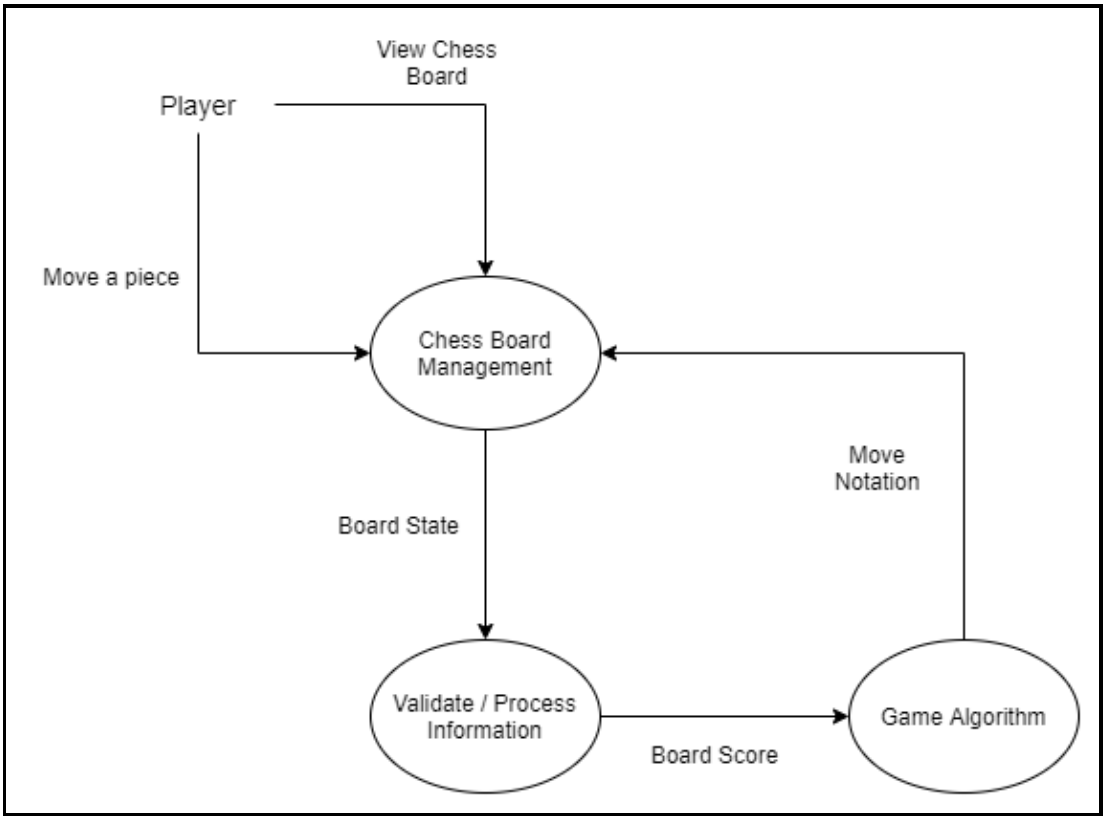


Figure 8: DFD Level 2 - AI

4.3.3.5 Game Report management:

- Generate human comprehensible move notation
- Updating move log
- Updating game state

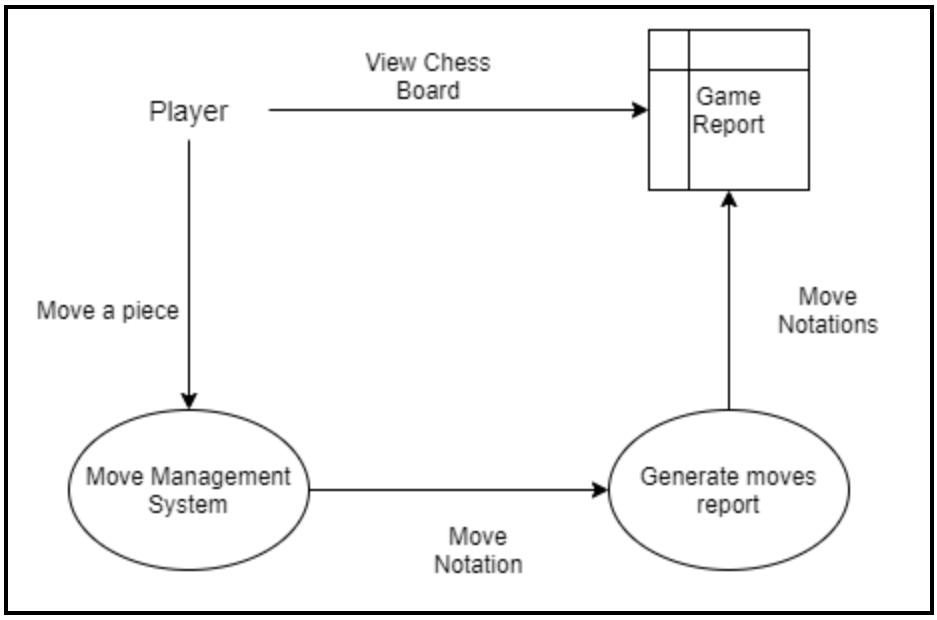


Figure 9: DFD Level 2 - Game Report

4.3.3.6 Chess board:

- Graphical representation of pieces
- Graphically making the move after move is made
- Representing human comprehensible move notation
- Updating board layout
- Prompting game states

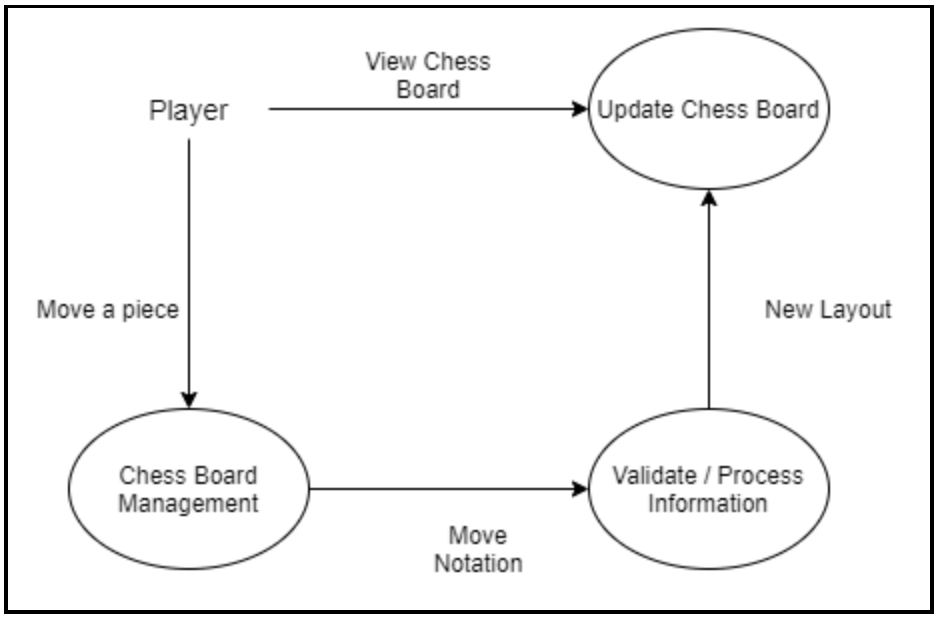


Figure 10: DFD Level 2 - Chess Board

5. System Development

5.1 Foreground Process

5.1.1: Script snippet view.py

```
42          def countdown(self, wc, bc):
43              """
44              Decrement Clock timer based on move made by player
45              :param wc: white timer at present
46              :param bc: black timer at present
47              :return: victor based on time
48              """
49              global pt, clkstp
50              while wc and bc:
51                  if not clkstp:
52                      ai_move_made = False
53                      if self.stp == True:
54                          break
55
56                      # black won on time
57                      if wc <= 0:
58                          View.time_out_win(vobj, 1)
59                          break
60
61                      # white won on time
62                      if bc <= 0:
63                          View.time_out_win(vobj, 2)
64                          break
65
66                      # decrement white timer
67                      if pt == "white":
68                          wm = int((wc / 60) % 60)
69                          ws = int((wc) % 60)
70                          wt = int((wc * 10) % 10)
71                          wht = '{:02d}:{:02d}:{:0d}'.format(wm, ws, wt)
72                          wc -= 0.01
73                          c2.config(text=wht)
74                          configurations.wctime = wht
75
76                      # decrement black timer
77                      if pt == "black":
78                          bm = int((bc / 60) % 60)
79                          bs = int((bc) % 60)
80                          bt = int((bc * 10) % 10)
81                          blk = '{:02d}:{:02d}:{:0d}'.format(bm, bs, bt)
82                          bc -= 0.01
83                          c1.config(text=blk)
84                          configurations.bctime = blk
85
86                  time.sleep(0.01)
87
88          def pause(self):
89              """
90              Pause timer for both players
91              """
92              global clkstp
93              clkstp = not clkstp
94
95          def resume(self):
96              """
97              Resume timer for both players
98              """
99              global clkstp
100             clkstp = not clkstp
101
102         def stop(self):
103             """
104             Stops timer for both players
105             """
106             self.stp = True
107
108     except:
109         pass
```

Figure 11: Script snippet view.py - Timer

```
368     def on_square_clicked(self, event):
369         """
370         Registers mouse clicks and determines piece on square
371         :param event: mouse click event
372         """
373         config = ConfigParser()
374         config.read('../data/chess_options.ini')
375         mclr = guiconfig.main_color
376         root.config(bg=mclr)
377         clicked_row, clicked_column = self.get_clicked_row_column(event)
378         global flipflg
379         if flipflg == 1:
380             clicked_row, clicked_column = self.flip_loc(clicked_row, clicked_column)
381 
382         position_of_click = self.controller.get_alphanumeric_position((clicked_row, 
383         # on second click
384         if self.selected_piece_position:
385             try:
386                 pmv = self.controller.pre_move_validation(self.selected_piece_positi
387                 if pmv == 1:
388                     self.auto_flip_board()
389 
390             except:
391                 self.shift(self.selected_piece_position, position_of_click)
392             finally:
393                 self.selected_piece_position = None
394 
395         self.update_highlight_list(position_of_click)
396         self.board_view_check()
397         self.can_state_log()
398         pt = self.controller.player_turn()
399         isai = config.get('ai_stats', 'is_ai')
400         ai_color = config.get('ai_stats', 'ai_color')
401         # if match against AI
402         if isai == "True" and pt == ai_color and not model.Model.gameOver:
403             global click_button, send_to_ai, rec_form_ai
404             root.update()
405             click_button = self.canvas.unbind("<Button-1>", click_button)
406             bd = self.controller.get_board()
407             mc = self.controller.get_lastmove()
408             try:
409                 lm = bd.peek()
410             except:
411                 lm = 0
412 
413             # start thread for AI
414             Tai = threading.Thread(target=self.ai_update)
415             Tai.setDaemon(True)
416             Tai.start()
417             root.configure(cursor="exchange #0000FF")
418             send_to_ai.send([bd, mc, lm, False])
```

Figure 12: Script snippet view.py – Mouse Click on Board

```
327     def ai_update(self):
328         """
329         Call AI & recieve AI's move
330         """
331         global rec_form_ai
332         mv = rec_form_ai.recv()
333         if mv == "00":
334             pass
335         else:
336             # for AI move of pawn promotion
337             if len(mv) == 5:
338                 sp = mv[0] + mv[1]
339                 fp = mv[2] + mv[3]
340                 up = mv[4]
341                 if mv[3] == "8":
342                     up = mv[4]
343                     up = up.upper()
344                 else:
345                     up = mv[4]
346 
347                 config.set('ai_stats', 'promo', up)
348                 with open('../data/chess_options.ini', 'w') as config_file:
349                     config.write(config_file)
350 
351                 pmv = self.controller.pre_move_validation(sp, fp)
352 
353             # if AI move is normal
354             else:
355                 sp = mv[0] + mv[1]
356                 fp = mv[2] + mv[3]
357                 pmv = self.controller.pre_move_validation(sp, fp)
358 
359             root.after(100, self.board_view_check)
360             root.after(100, self.can_state_log)
361             root.after(100, root.update())
362             self.last_move_highlight(sp, fp)
363 
364         root.config(cursor='')
365         global click_button
366         click_button = self.canvas.bind("<Button-1>", self.on_square_clicked)
```

Figure 13: Script snippet view.py – Call A.I. Process

5.1.2: Script snippet model.py

```python
    def move(self, start_pos, final_pos):
        """
        Makes the move in the board and updates dictionary accordingly
        :param start_pos: start position of piece
        :param final_pos: end position of piece
        """
        # move count
        config = ConfigParser()
        config.read('../data/chess_options.ini')
        mcnt = int(config.get('chess_moves', 'move_count'))
        mcnt += 1
        config.set('chess_moves', 'move_count', str(mcnt))
        with open('../data/chess_options.ini', 'w') as config_file:
            config.write(config_file)

        # check for enpassant possiblity
        try:
            mt = self.moveType[-1]
            if mt == "Enpassant Move":
                self.enpass_possible = 0
            else:
                pass
        except:
            pass

        # if normal move
        if self.enpass_possible == 0:
            self[final_pos] = self.pop(start_pos, None)

        # if enpassant move
        elif self.enpass_possible > 0:
            try:
                self[final_pos] = self.pop(start_pos, None)
                at = self.enpass_square[-1]
                ta = self.tmp_attack_square[-1]
                sp = self.enpass_start_square[-1]
                if final_pos == at:
                    if mt == "Double Forward Move":
                        self.pop(ta, None)
                        self.enpass_move_made = True
                        self.keep_reference(self)

            except:
                pass

        self.enpass_possible = 0
```

Figure 14: Script snippet model.py - Update Moves

5.1.3: Script snippet piece.py

```
55      def moves_available(self, current_position, directions, distance):
56          """
57          Collection of all moves from all possible pieces
58          :param current_position: start position of piece
59          :param directions: direction of move
60          :param distance: distance to move
61          :return: all moves available
62          """
63          piece = self
64          model = self.model
65          allowed_moves = []
66          next_allowed_moves = []
67          start_row, start_column = get_numeric_notation(current_position)
68          for x, y in directions:
69              collision = False
70              for step in range(1, distance + 1):
71                  # is path blocked
72                  if collision:
73                      break
74
75                  destination = start_row + step * x, start_column + step * y
76                  # allowed move
77                  if self.possible_position(destination) not in model.all_occupied_posit
78                      allowed_moves.append(destination)
79                  # capture move
80                  elif self.possible_position(destination) in model.all_positions_occupi
81                      collision = True
82                  # normal move
83                  else:
84                      allowed_moves.append(destination)
85                      collision = True
86
87          allowed_moves = filter(model.is_on_board, allowed_moves)
88          return map(model.get_alphanumeric_position, allowed_moves)
```

Figure 15: Script snippet piece.py – Legal Moves

5.2 Background Process

5.2.1: Script snippet aimanager.py

```python
def start_process(ai_child_conn, ui_parent_conn):
    """
    This function initializes the chess board for AI.
    :param ai_child_conn: AI will receive message through this connection.
    :param ui_parent_conn: AI will send response through this connection.
    """
    ai_rec_conn = ai_child_conn
    ai_send_conn = ui_parent_conn
    # Creating object of ChessAI class in aimodel
    cai = aimodel.ChessAI()
    ai_manager(cai, ai_rec_conn, ai_send_conn)

```

Figure 16: Script snippet aimanager.py - A.I. Process

```python
def ai_manager(cai, ai_rec_conn, ai_send_conn):
    """
    This function keeps AI active during the course of the application
    :param cai: Object of ChessAI class of aimodel.
    :param ai_rec_conn: Connection object to receive message from.
    :param ai_send_conn: Connection object to send message to.
    """
    while True:
        # Waiting for a message from Frontend.
        brd, mov_count, human_move, stop_ai = ai_rec_conn.recv()
        if stop_ai == "new":
            # Start a new game
            cai.new_board()
            continue
        if stop_ai:
            # Application closed
            break

        # Start AI processing
        move = cai.send_bd(brd, mov_count, human_move)

        # Send response back to frontend
        ai_send_conn.send(str(move))

    # Closing connection
    ai_send_conn.close()

```

Figure 17: Script snippet aimanager.py - G.U.I - A.I. Communication

5.2.2: Script snippet aimodel.py

```python
# Evaluate Score of Board
def eval_scores(pos=True, cas=True):
    """
    This function evaluates board score.
    If both parameters are false this function will only do material evaluation.
    :param pos: True if need to do positional evaluation, default = True.
    :param cas: True if need to do castling evaluation, default = True.
    :return: Score of board based on whose turn it is.
    """
    global endgame, infinity, board
    # Check for End Of Game
    if board.is_checkmate():
        return -infinity
    elif board.is_stalemate() or board.is_insufficient_material():
        return 0
    elif board.is_fivefold_repetition():
        return 0

    # Count total number of each type of pieces of each player
    wp = len(board.pieces(chess.PAWN, chess.WHITE))
    bp = len(board.pieces(chess.PAWN, chess.BLACK))
    wn = len(board.pieces(chess.KNIGHT, chess.WHITE))
    bn = len(board.pieces(chess.KNIGHT, chess.BLACK))
    wb = len(board.pieces(chess.BISHOP, chess.WHITE))
    bb = len(board.pieces(chess.BISHOP, chess.BLACK))
    wr = len(board.pieces(chess.ROOK, chess.WHITE))
    br = len(board.pieces(chess.ROOK, chess.BLACK))
    wq = len(board.pieces(chess.QUEEN, chess.WHITE))
    bq = len(board.pieces(chess.QUEEN, chess.BLACK))

    # Material score
    beval = pawn * (wp - bp) + night * (wn - bn) + bishop * (wb - bb) + rook * (wr -
    # Piece Position
    if pos:
        beval = eval_pos(beval)

    # Castling Evaluation
    if cas:
        beval = eval_castle(beval)

    if board.turn:
        # If Turn is of White side
        return beval
    else:
        # If Turn is of Black side
        return -beval
```

Figure 18: Script snippet aimodel.py - Board Evaluation

```
336     # Quiescence Search Algorithm
337     def quiesce(self, alpha, beta):
338         """
339         This function implements Quiescence Search.
340         The purpose of this search is to only evaluate "quiet" positions, or positior
341         tactical moves to be made.
342         This search is needed to avoid the horizon effect.
343         :param alpha: Alpha for alpha-beta pruning.
344         :param beta: Beta for alpha-beta pruning.
345         :return: Board score evaluated at end of current Quiescence Search.
346         """
347         global board
348         stand_pat = eval_scores()
349 
350         if stand_pat >= beta:
351             return beta
352 
353         if alpha < stand_pat:
354             alpha = stand_pat
355 
356         for move in board.legal_moves:
357             if board.is_capture(move):
358                 # Check castling move
359                 wc_flag, bc_flag = castling_check(move)
360                 board.push(move)
361                 score = -self.quiesce(-beta, -alpha)
362                 board.pop()
363                 # Undoing effects of castling check
364                 castling_check_undo(wc_flag, bc_flag)
365                 if score >= beta:
366                     return beta
367 
368                 if score > alpha:
369                     alpha = score
370 
371         return alpha
```

Figure 19: Script snippet aimodel.py - Quiescence Search

```python
    # Negated Minimax (Negamax) Algorithm
    def pvnegamax(self, depth, table):
        """
        This function is similar to pvSearch function.
        It first searches for move in book (i.e. pre-calculated moves).
        This function uses pvSearch function to find best move.

        Algorithms:
            Negated Minimax Search
            Principal Variation Search (PVS) with Aspiration
            Move Ordering
            Alpha-Beta Pruning
            Singular Reply Extension

        :param depth: Depth to which search is to be performed.
        :param table: Tree / Transposition table to use during search.
        :return: Move, Table (if move is from book move than table = "book move").
        """
        global board, infinity, move_count
        try:
            if move_count <= 15:
                # Book Move
                path = "../books/human.bin"
                book_move = chess.polyglot.MemoryMappedReader(path).weighted_choice(bc
                return book_move, "book move"
            else:
                raise Exception("No more book move")

        except:
            # Negamax if not in book

            # aplha - Current best score
            # beta - Best possible score

            lMoves = list(board.legal_moves)
            table.alpha = -infinity
            table.beta = infinity

            # Singular Reply Extension
            if len(list(lMoves)) == 1:
                depth = depth + 1

            TTNeeded = len(table.nodes) == 0
            # Dont make new TT if already exists
            if TTNeeded:
                # make custom move tree
                for move in lMoves:
                    newNode = Tree(-infinity)
                    table.add_node(move, newNode, -infinity)

            # using fail-soft with negamax
            # For first move of every depth
            move = list(table.nodes.keys())[0]

            # Check castling move
            wc_flag, bc_flag = castling_check(move)
            board.push(move)
            bValue = self.pvSearch(-table.beta, -table.alpha, depth - 1, table.nodes[m
            board.pop()
            # Undoing effects of castling check
            castling_check_undo(wc_flag, bc_flag)
            bValue = -bValue

            table.nodes[move][1] = bValue
            table.val = bValue
            # bMove = move
            if bValue < table.beta:
                if bValue > table.alpha:
                    table.alpha = bValue

                # First depth is handled here
                for move in list(table.nodes.keys())[1:]:
                    # Check castling move
```

Figure 20: Script snippet aimodel.py - Negamax Part 1

```
373     def pvSearch(self, alpha, beta, depthleft, subtree, do_null, check_ext):
374         """
375         Principal Variation Search (PVS), an enhancement to Alpha-Beta, based on null-
376         none PV-nodes, to prove a move is worse or not than an already safe score fro
377 
378         Pruning Applied:
379             Alpha-Beta Pruning
380             Extended Null Move Pruning
381 
382         Extensions Applied:
383             Check Extension
384             Singular Reply Extension
385 
386         Other Algorithms:
387             Negated Minimax Search
388             Principal Variation Search (PVS) with Aspiration
389             Quiescence Search
390             Move Ordering
391 
392         :param alpha: Alpha for alpha-beta pruning.
393         :param beta: Beta for alpha-beta pruning.
394         :param depthleft: Depth that is left to be searched.
395         :param subtree: Current active node that needs to be searched.
396         :param do_null: True if Extended Null-move pruning is to be performed else Fal
397         :param check_ext: Number of times check extension has been performed.
398         :return: Score of current subtree.
399         """
400         global board, infinity, endgame
401         if board.is_checkmate():
402             subtree.val = -infinity
403             return -infinity
404         elif board.is_stalemate() or board.is_insufficient_material():
405             subtree.val = 0
406             return 0
407         elif board.is_fivefold_repetition():
408             subtree.val = 0
409             return 0
410 
411         lMoves = board.legal_moves
412         subtree.alpha = alpha
413         subtree.beta = beta
414 
415         # Check extension
416         if board.is_check() and depthleft <= 3 and check_ext < 4:
417             depthleft = depthleft + 1
418             check_ext = check_ext + 1
419         # Singular Reply Extension
420         elif len(list(lMoves)) == 1 and depthleft <= 3:
421             depthleft = depthleft + 1
422 
423         # Board Evaluation
424         if depthleft <= 0:
425             result = self.quiesce(alpha, beta)
426             subtree.val = result
427             return result
428 
429         TTNeeded = len(subtree.nodes) == 0
430         # Dont make new TT if already exists
431         if TTNeeded:
432             # Make custom move tree
433             for move in lMoves:
434                 newNode = Tree(-infinity)
435                 subtree.add_node(move, newNode, -infinity)
436 
437         # Extended Null Move Pruning
438         # conduct a null-move search if it is legal and desired
439         if (not board.is_check()) and do_null:
440             move = chess.Move.null()
441             board.push(move)
442             newNode = Tree(-infinity)
443             subtree.add_node(move, newNode, -infinity)
```

Figure 21: Script snippet aimodel.py - Negamax Part 2

6. System Implementation

- Main screen:

Figure 22: Main Screen

- Main screen with default settings:

Figure 23: Main screen with default settings

- Main screen with AI selected

Figure 24: Main screen with AI selected

- Changing clock time of users / players:

Figure 25: Changing clock time of users / players

- Starting game against human:

Figure 26: Starting game against human

- Black Pawn move:

Figure 27: Black Pawn move

- White Pawn move:

Figure 28: White Pawn move

- Black En-passant move available:

Figure 29: Black En-passant move available

- Black En-passant capture move made:

Figure 30: Black En-passant capture move made

- White Queen move:

Figure 31: White Queen move

- Black Bishop move:

Figure 32: Black Bishop move

- Black Queen move:

Figure 33: Black Queen move

- Black King Long Castle move available:

Figure 34: Black King Long Castle move available

- Black King Long Castle move made:

Figure 35: Black King Long Castle move made

- White King Castle move available:

Figure 36: White King Castle move available

- White King Short Castle move made:

Figure 37: White King Short Castle move made

- Voiding En-passant move:

Figure 38: Voiding En-passant move - 1

Figure 39: Voiding En-passant move - 2

Figure 40: Voiding En-passant move - 3

- Rook move:

Figure 41: Rook move

- Pawn capture blocked due to pin:

Figure 42: Pawn capture blocked due to pin

- White delivering check:

Figure 43: White delivering check

- King moves:

Figure 44: King moves

- Pawn Promotion move:

Figure 45: Pawn Promotion move

- Options to Promote Pawn:

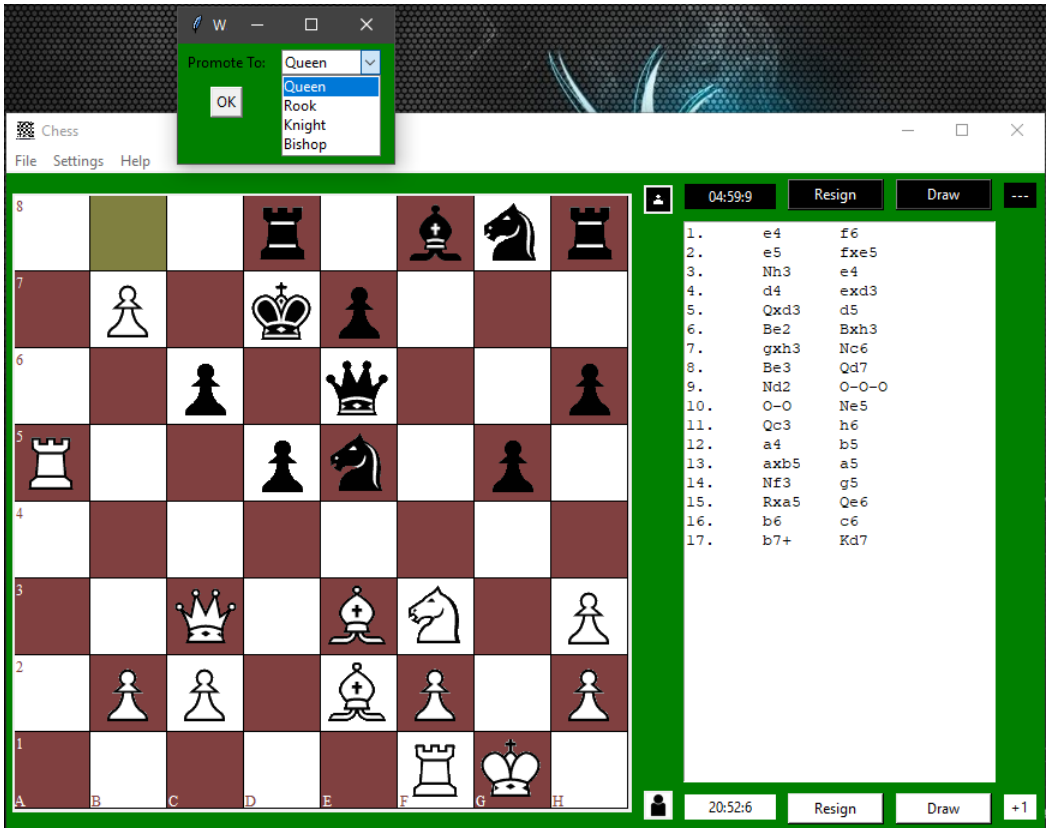


Figure 46: Options to Promote Pawn

- Pawn Selected to be Promoted as Queen:

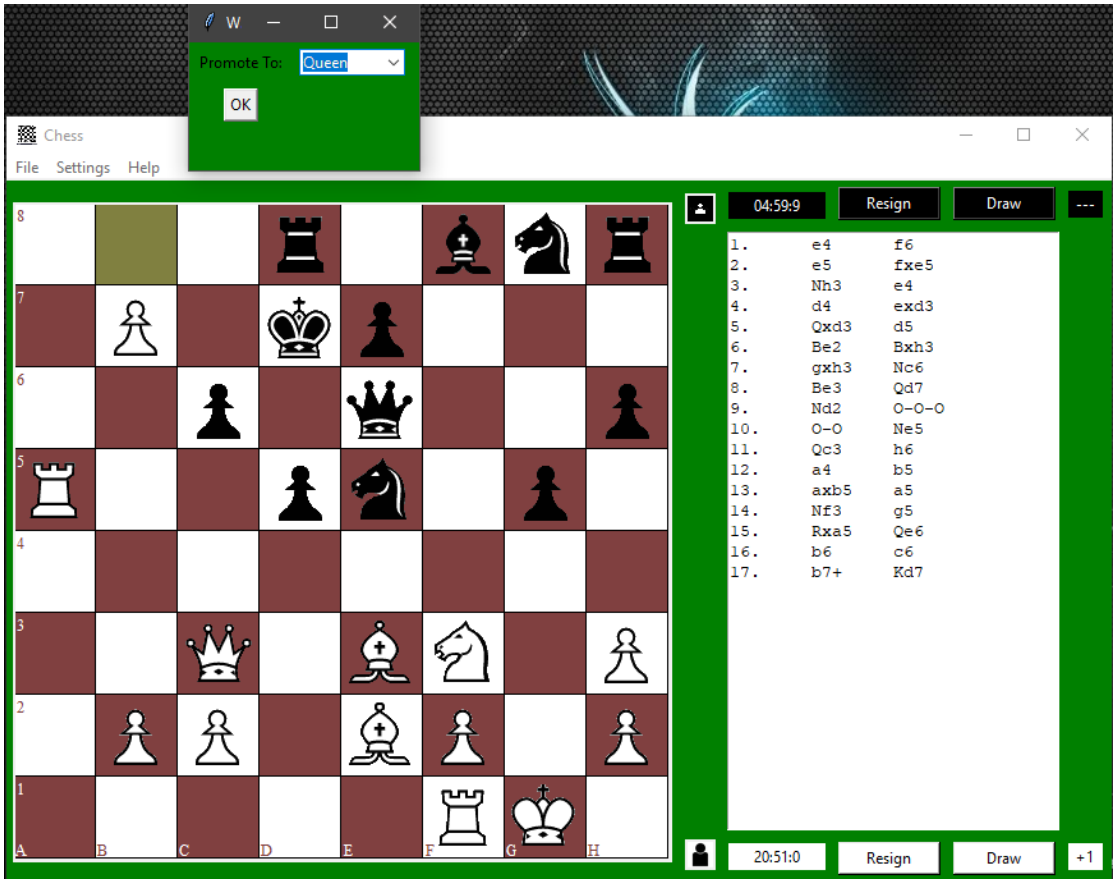


Figure 47: Pawn Selected to be Promoted as Queen

- Pawn Promoted to Queen:

Figure 48: Pawn Promoted to Queen

- Move log keeping track of moves:

Figure 49: Move log keeping track of moves

- New Queen:

Figure 50: New Queen

- Black Offered Draw:

Figure 51: Black Offered Draw

- White Rejects Draw Offer:

Figure 52: White Rejects Draw Offer

- Black Time runs out:

Figure 53: Black Time runs out

- After Game End:

Figure 54: After Game End - 1

Figure 55: After Game End – 2

Figure 56: After Game End - 3

- Chess Board Closing:

Figure 57: Chess Board Closing

- Save PGN Window:

Figure 58: Save PGN Window

- On Save PGN Clicked:

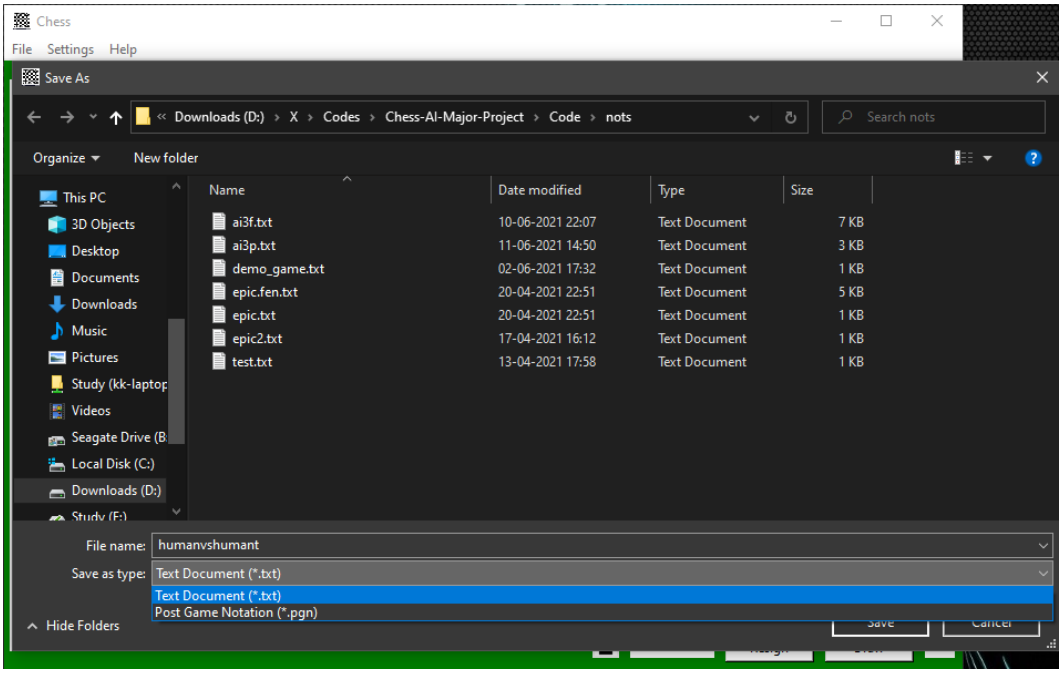


Figure 59: On Save PGN Clicked

- PGN Saved:

Figure 60: PGN Saved

- Save FEN Window:

Figure 61: Save FEN Window

- On Save FEN Clicked:

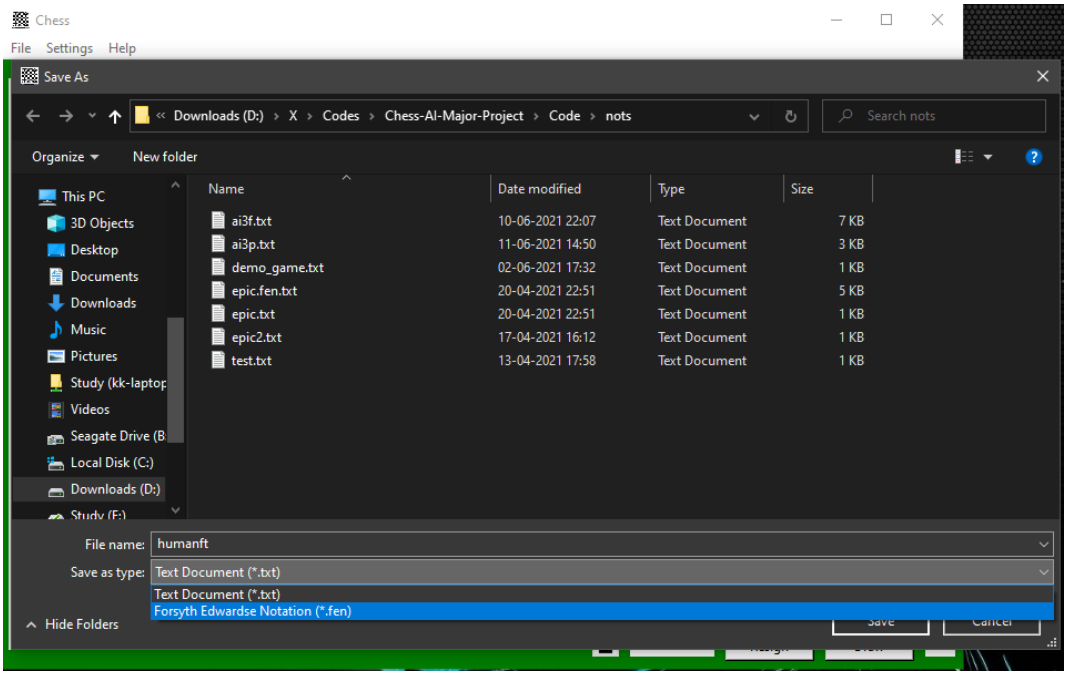


Figure 62: On Save FEN Clicked

- FEN Saved:

Figure 63: FEN Saved

- Saved Files in Directory:

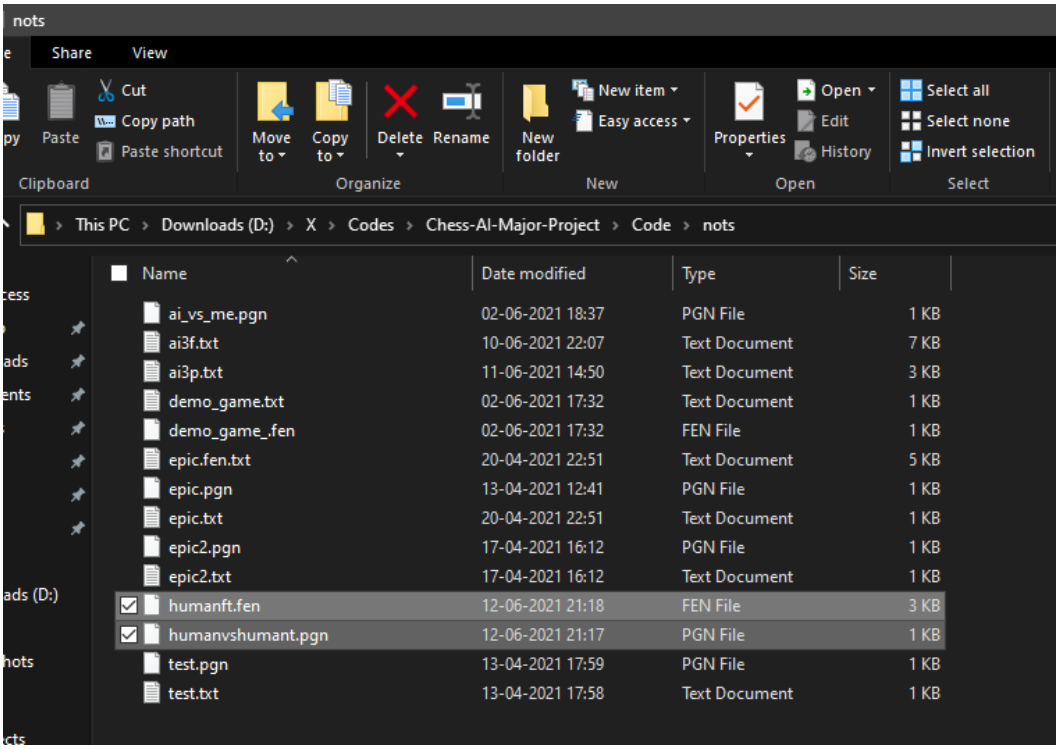


Figure 64: Saved Files in Directory

- Instruction Menu:

Figure 65: Instruction Menu

- On Instruction Menu Clicked:

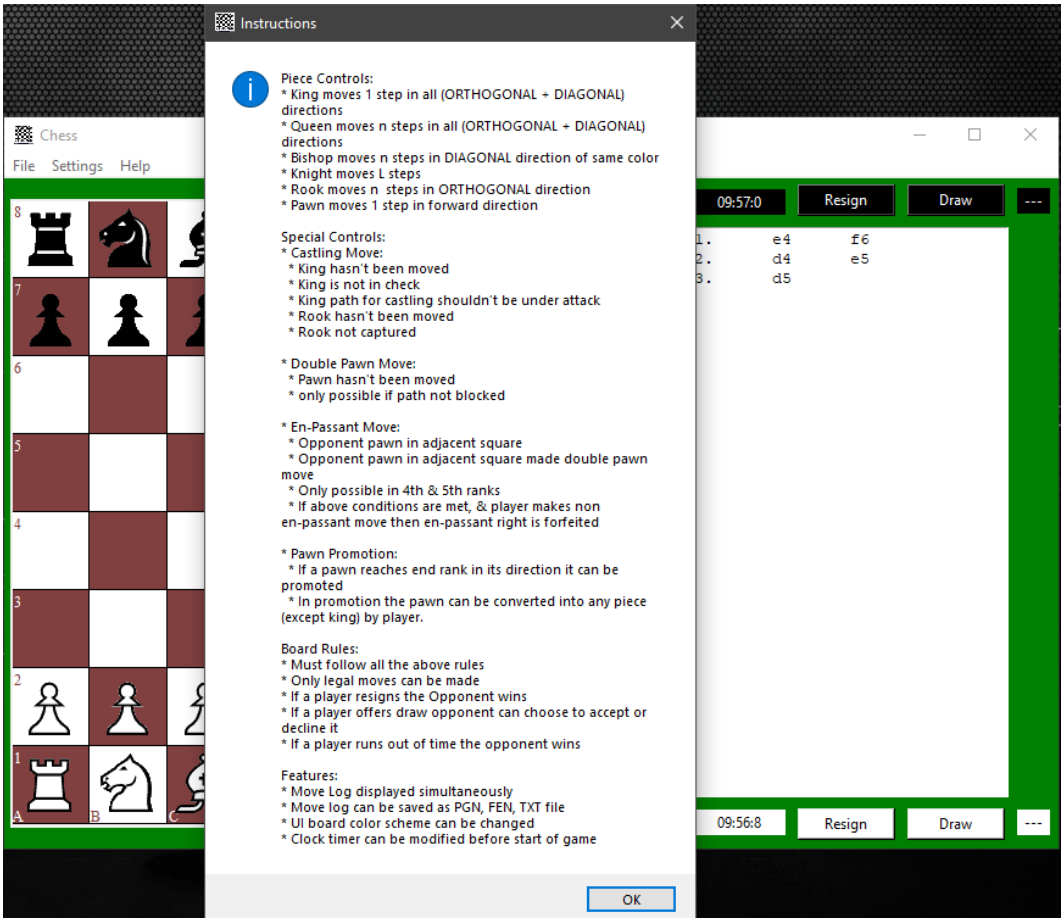


Figure 66: On Instruction Menu Clicked

- Pausing Clock of a Game after certain moves:

Figure 67: Pausing Clock of a Game after certain moves

- Clock unaffected after moves:

Figure 68: Clock unaffected after moves

- Flip Board:

Figure 69: Flip Board

- On Flip Board Clicked:

Figure 70: On Flip Board Clicked

- Auto-Flip Board:

Figure 71: Auto-Flip Board

- On Auto-Flip Board Clocked:

Figure 72: On Auto-Flip Board Clocked

- Board Auto-Flips after Black Move:

Figure 73: Board Auto-Flips after Black Move

- Board Auto-Flips after White Move:

Figure 74: Board Auto-Flips after White Move

- Turning Off Auto-Flip:

Figure 75: Turning Off Auto-Flip

- Auto-Flip Turned Off:

Figure 76: Auto-Flip Turned Off

- UI Settings:

Figure 77: UI Settings

- UI Settings Window:

Figure 78: UI Settings Window

- Primary Colour Selection:

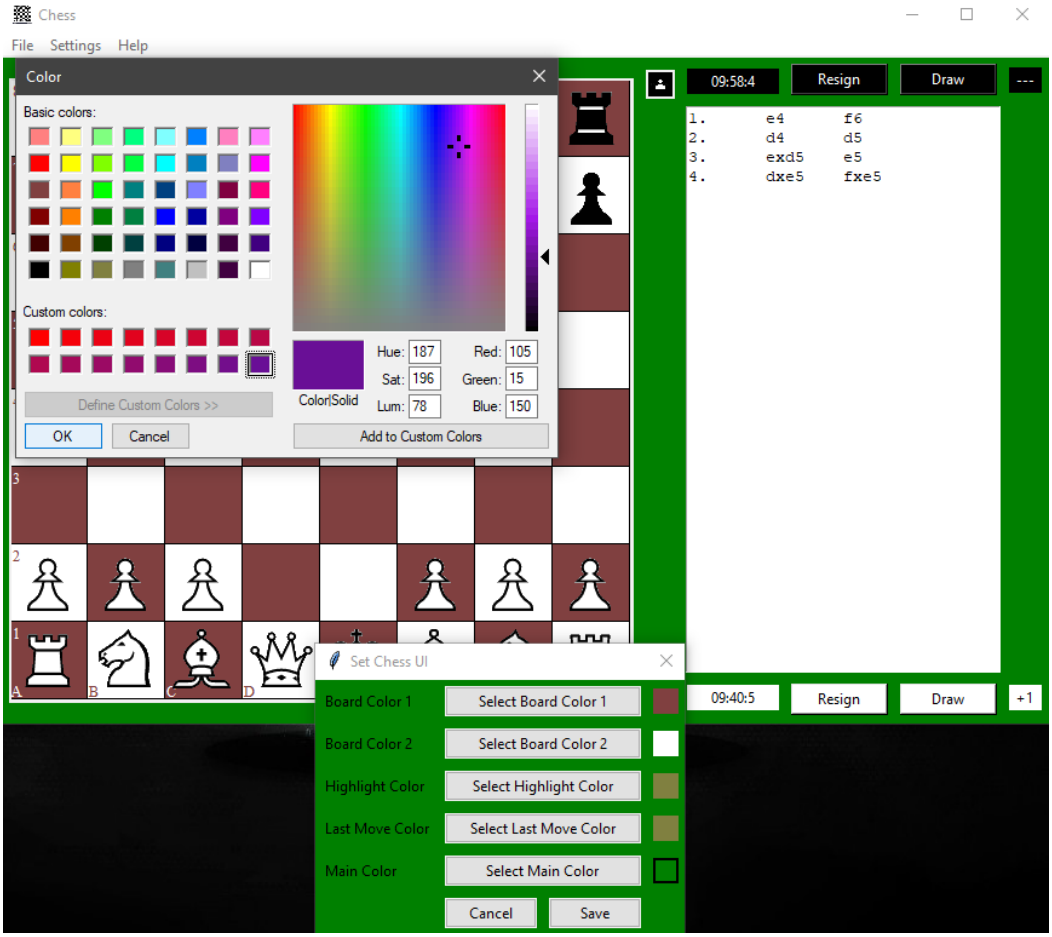


Figure 79: Primary Color Selection

- Secondary Colour Selection:

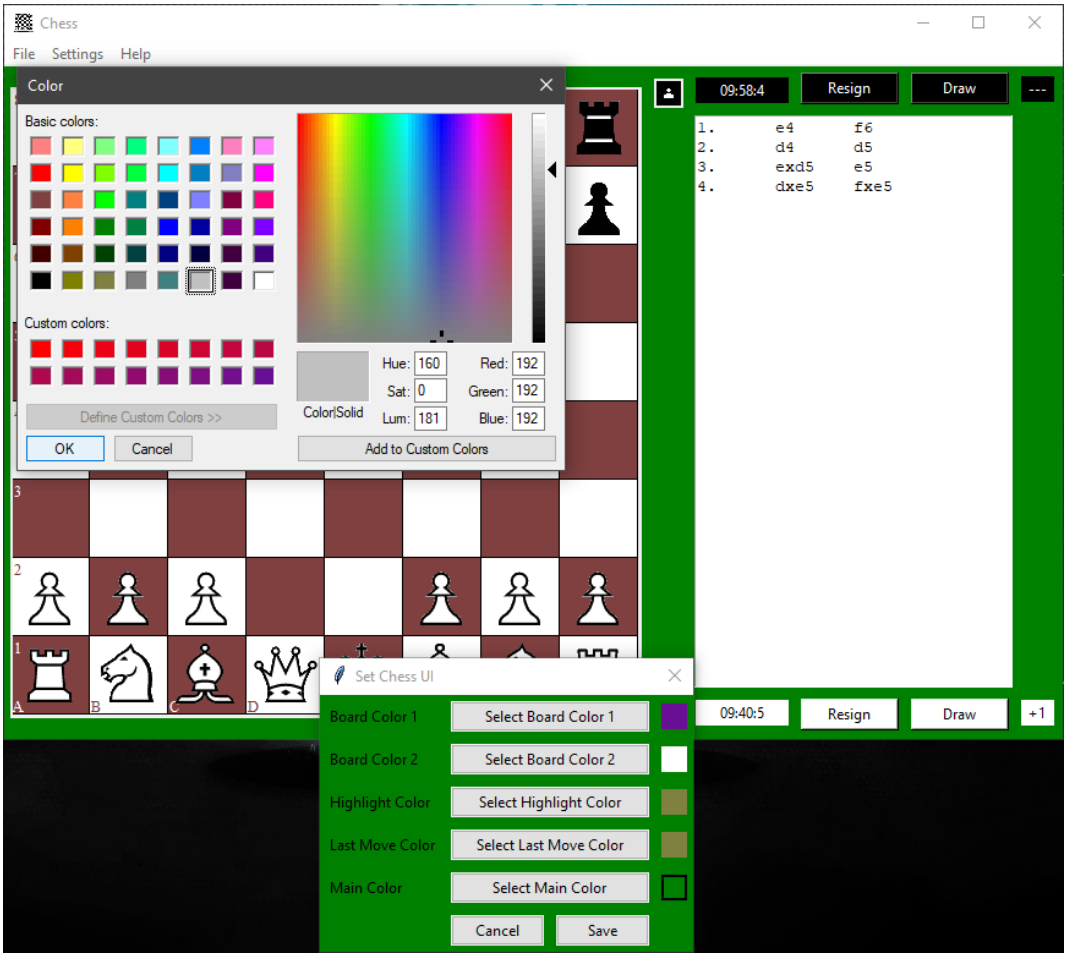


Figure 80: Secondary Color Selection

- Applying Selected Colours:

Figure 81: Applying Selected Colors – 1

Figure 82: Applying Selected Colors - 2

- Available Moves Highlight Colour:

Figure 83: Available Moves Highlight Color

- Changing Available Moves Highlight Colour:

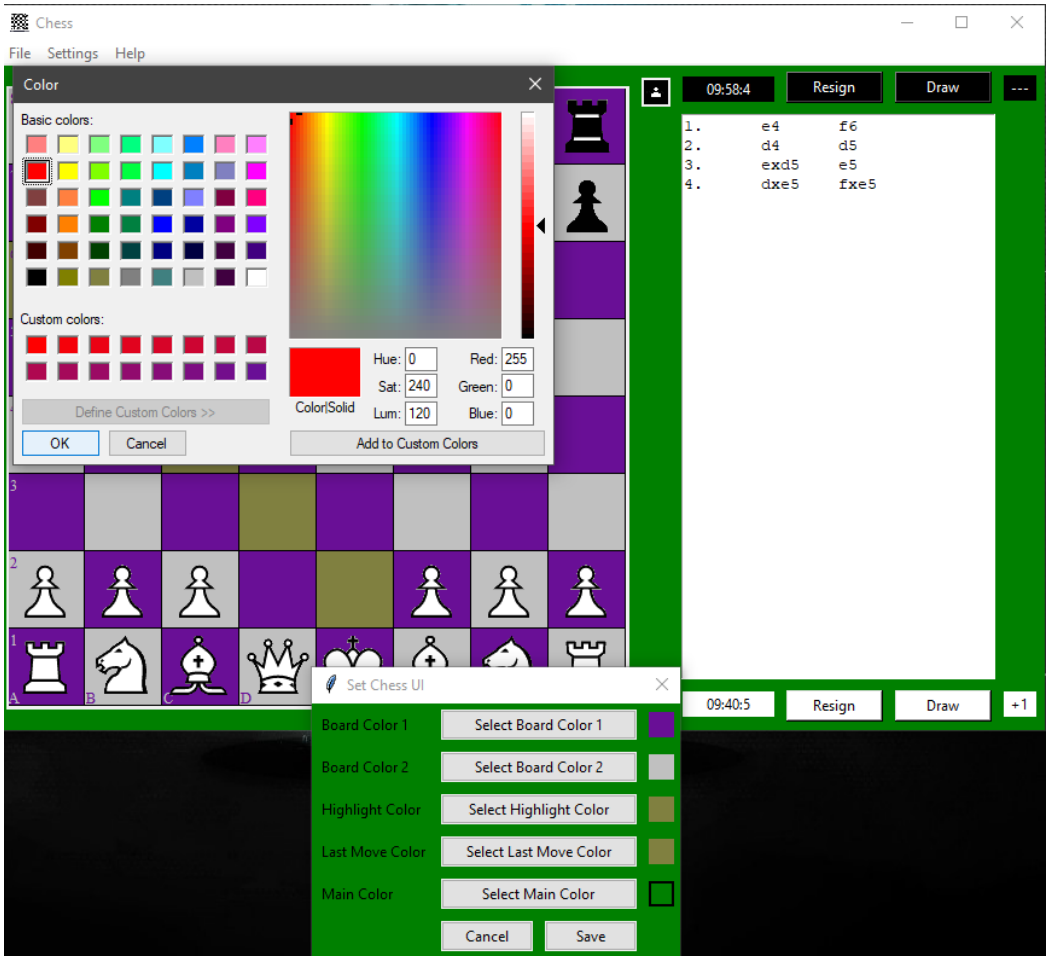


Figure 84: Changing Available Moves Highlight Color

- Applying Selected Colours:

Figure 85: Applying Selected Colors

- Highlight Colour Changed:

Figure 86: Highlight Colour Changed

- Changing Theme Colour:

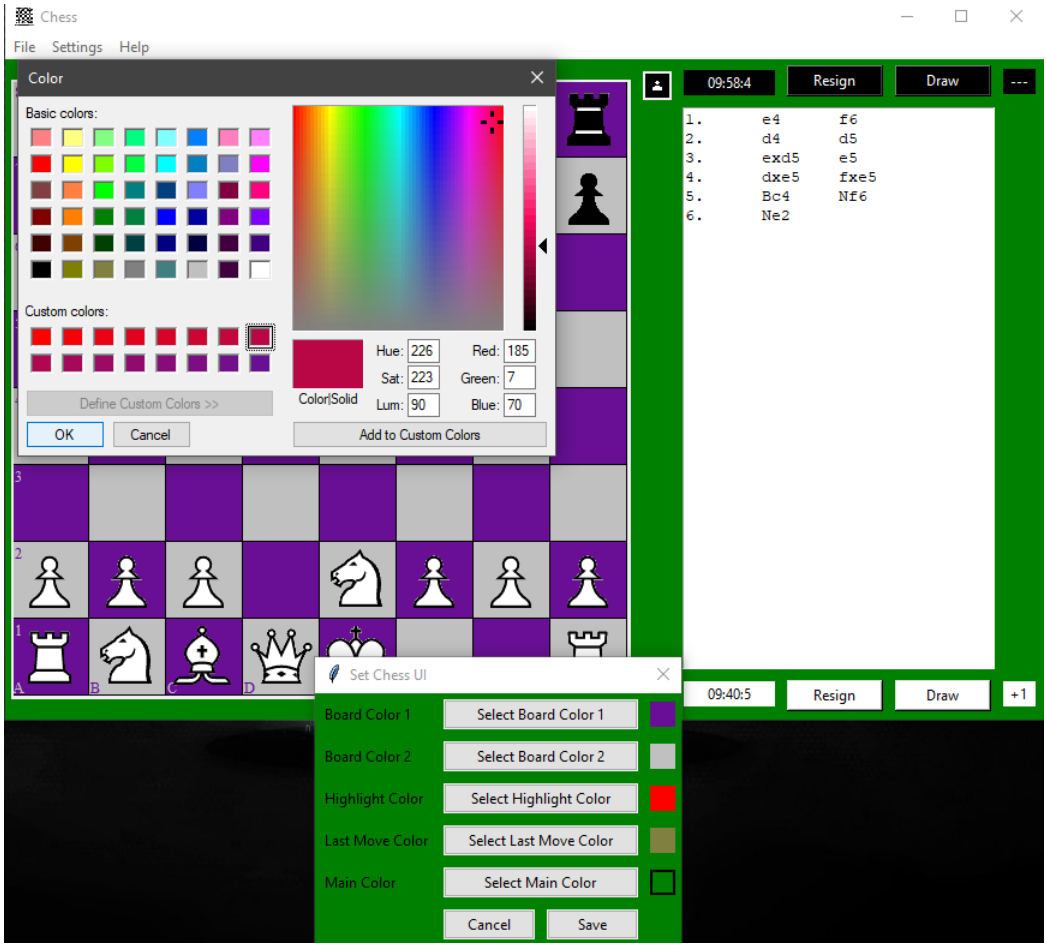


Figure 87: Changing Theme Color

- Theme Colour Preview:

Figure 88: Theme Colour Preview

- Theme Colour Changed:

Figure 89: Theme Colour Changed

- Save PGN via Menu:

Figure 90: Save PGN via Menu

- Saving PGN in txt format:

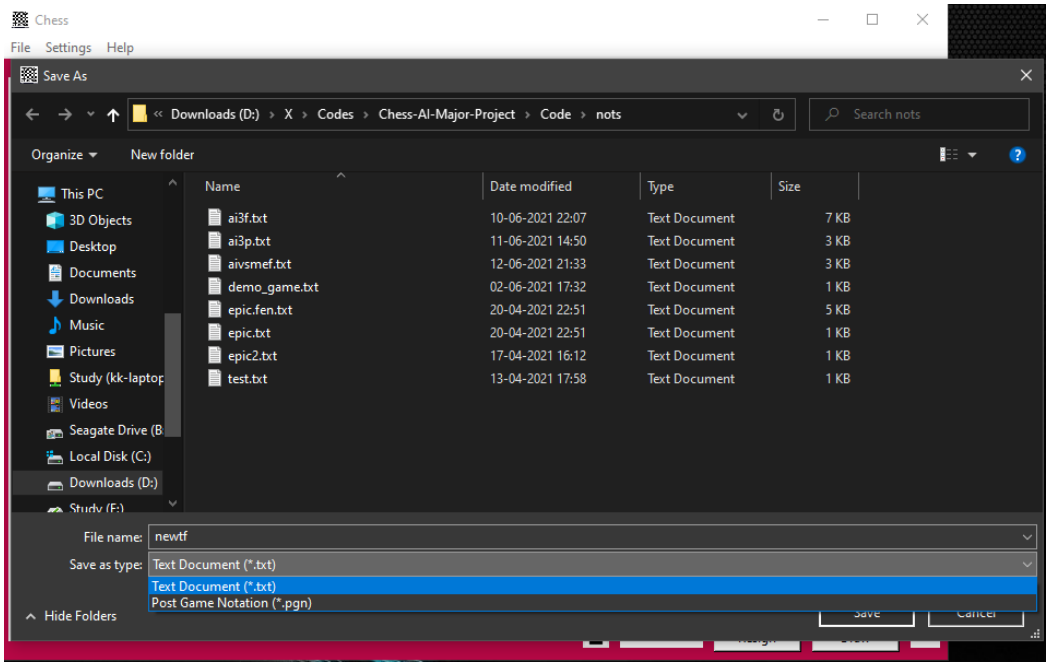


Figure 91: Saving PGN in txt format

- PGN Saved as TXT:

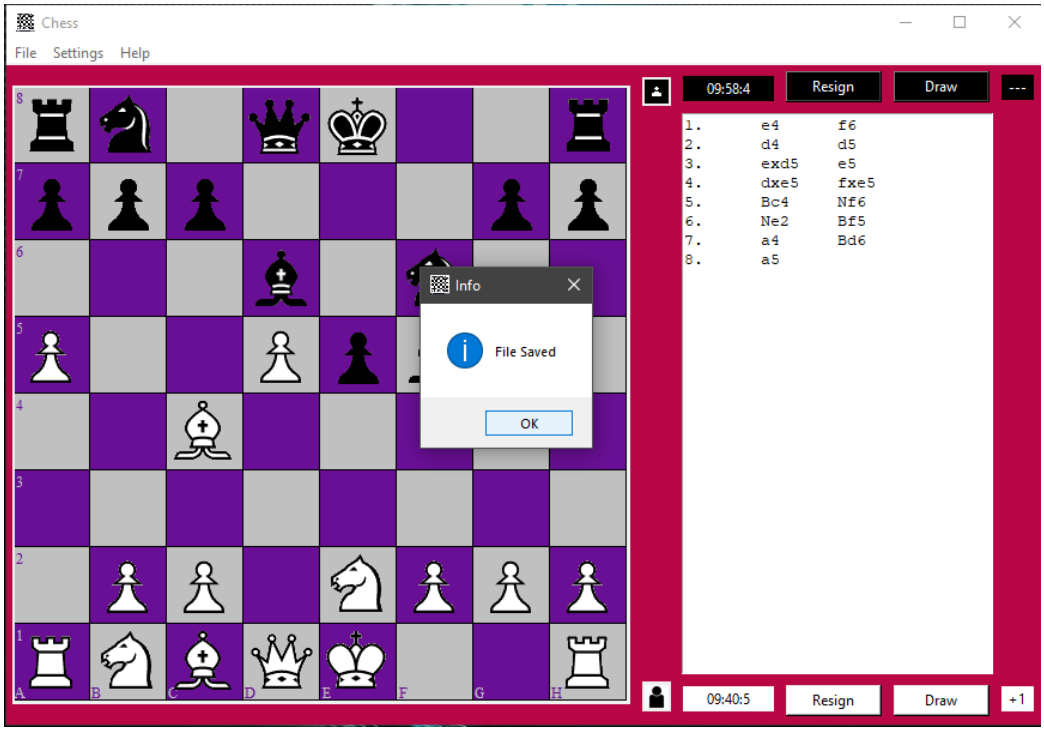


Figure 92: PGN Saved as TXT

- PGN in Selected Directory:

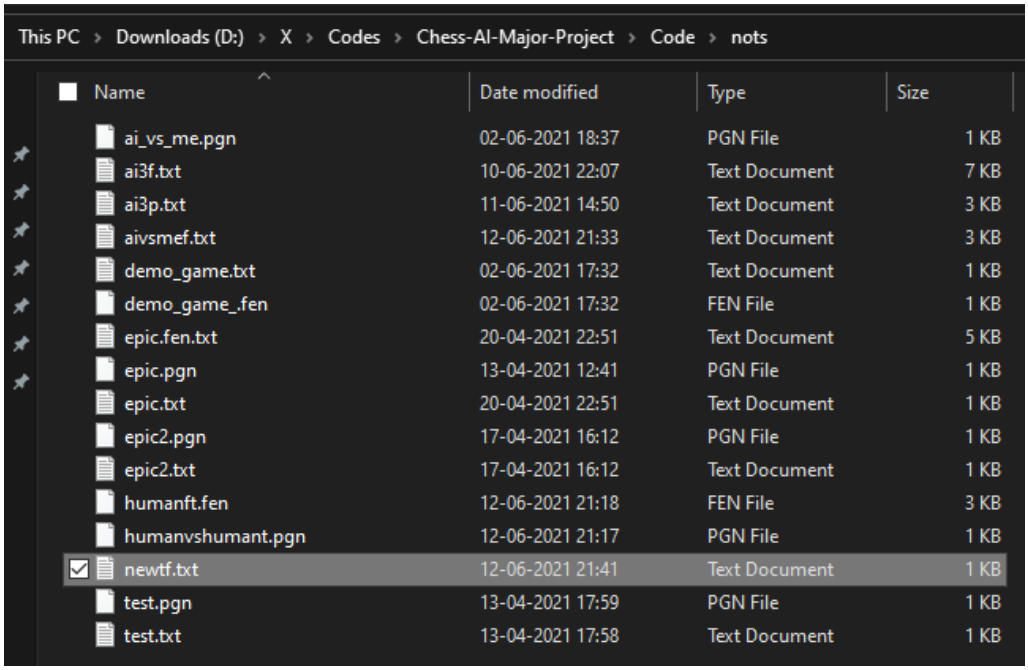


Figure 93: PGN in Selected Directory

- Contents of newtf.txt:

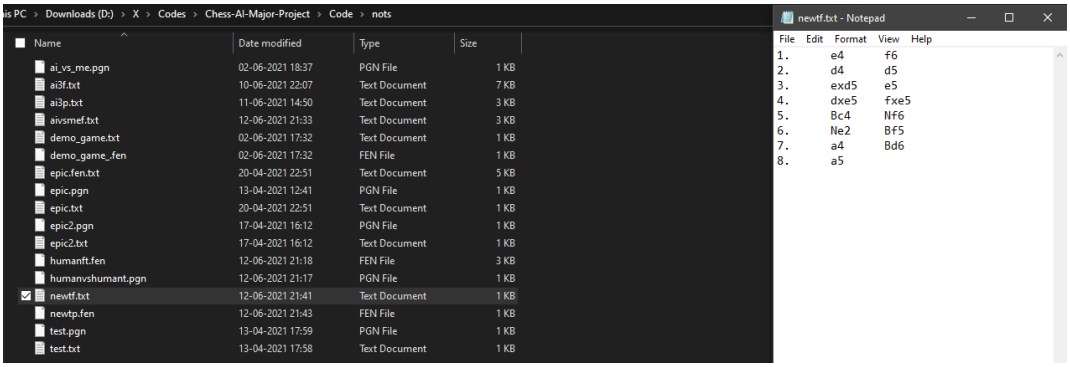


Figure 94: Contents of newtf.txt

- Save FEN via Menu:

Figure 95: Save FEN via Menu

- Saving FEN:

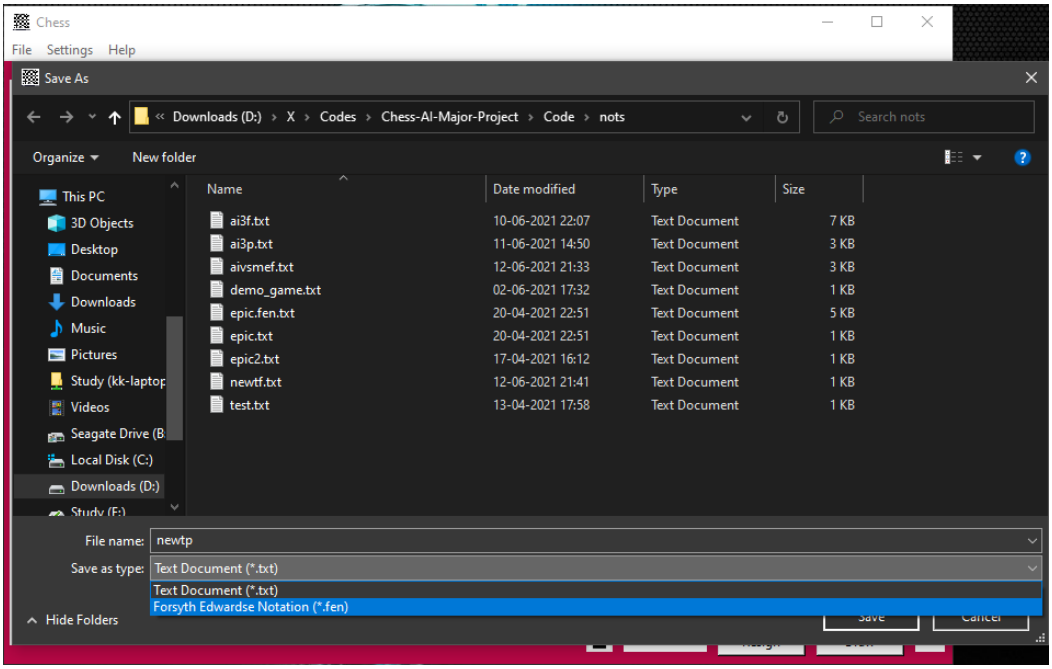


Figure 96: Saving FEN

- FEN Saved:

Figure 97: FEN Saved

- Contents of newtp.fen:

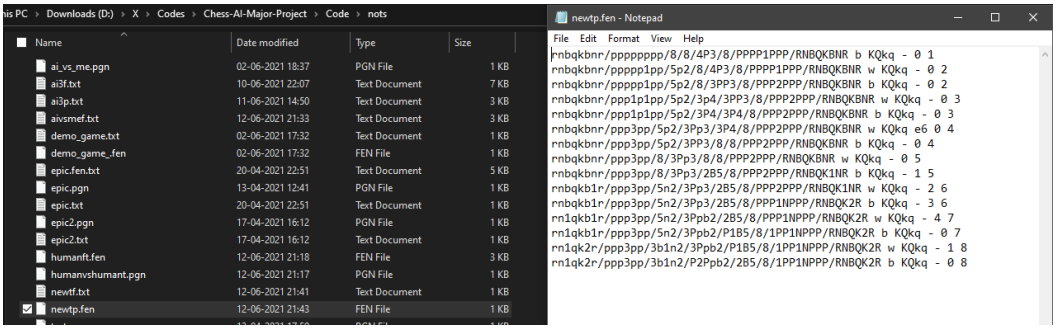


Figure 98: Contents of newtp.fen

- New Game Menu:

Figure 99: New Game Menu

- On New Game Menu Clicked:

Figure 100: On New Game Menu Clicked

- Exit Menu:

Figure 101: Exit Menu

- On Exit Menu Clicked:

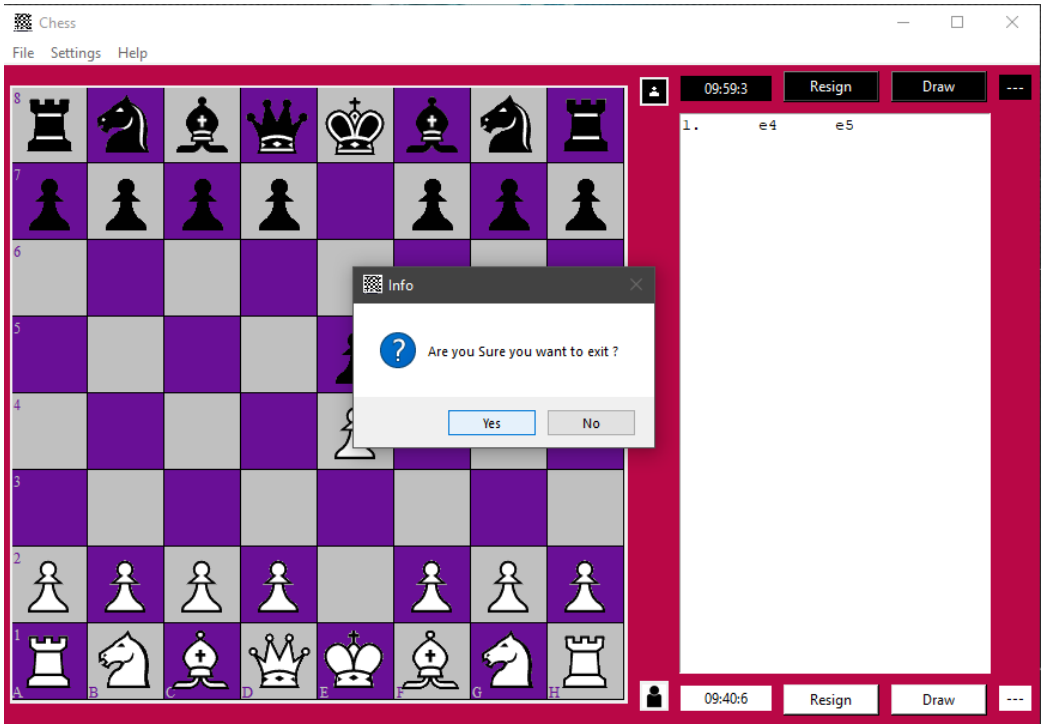


Figure 102: On Exit Menu Clicked - 1

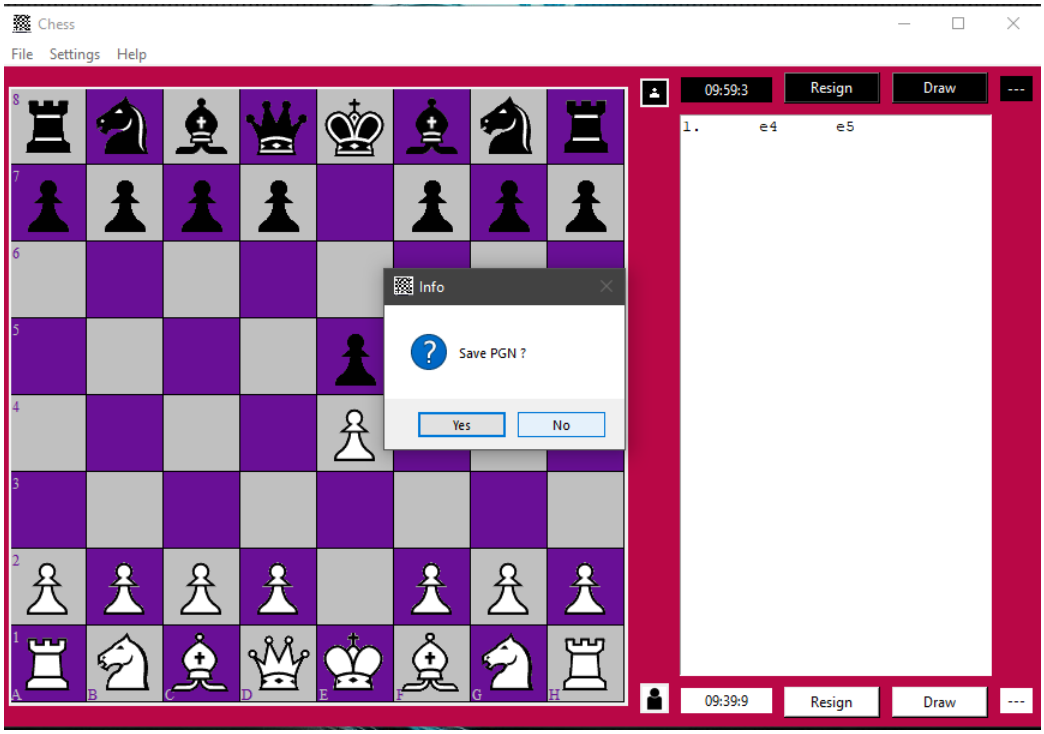


Figure 103: On Exit Menu Clicked - 2

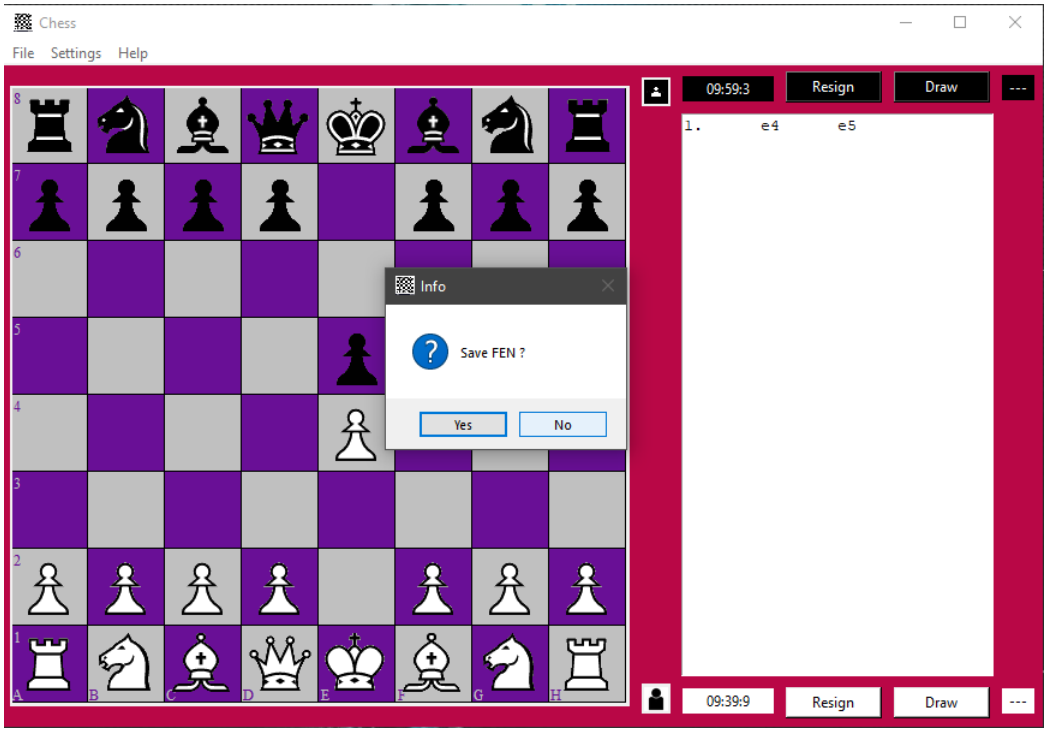


Figure 104: On Exit Menu Clicked - 3

- Draw by Insufficient Material:

Figure 105: Draw by Insufficient Material - 1

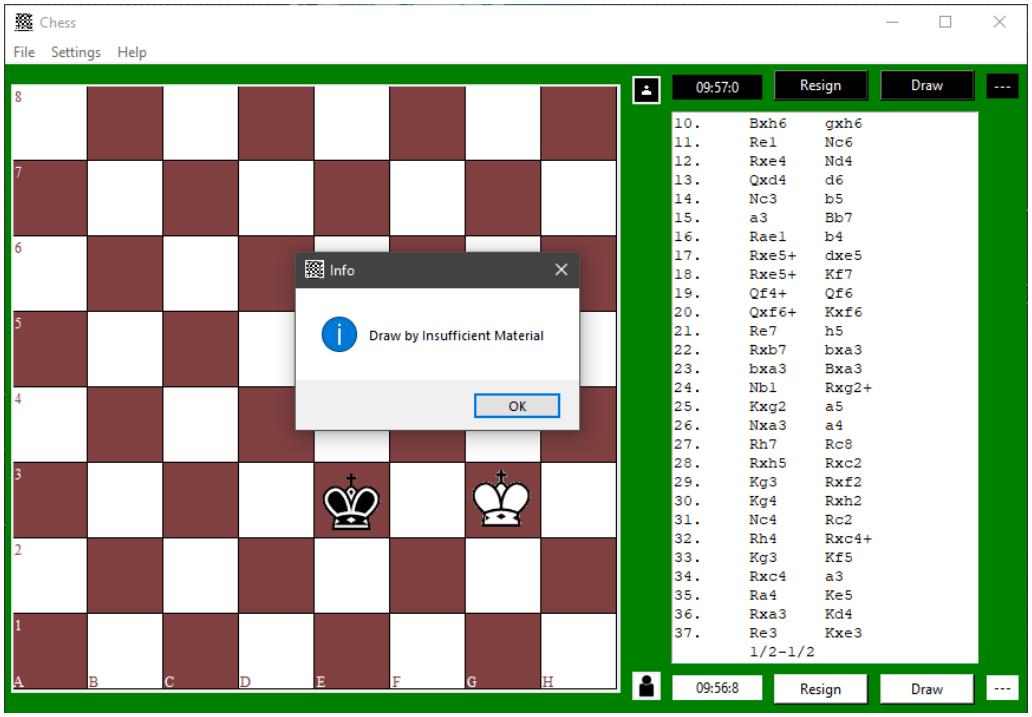


Figure 106: Draw by Insufficient Material - 2

- Draw by Agreement:

Figure 107: Draw by Agreement - 1

Figure 108: Draw by Agreement - 2

- Draw by Repetition:

Figure 109: Draw by Repetition - 1

Figure 110: Draw by Repetition - 2

- Stalemate:

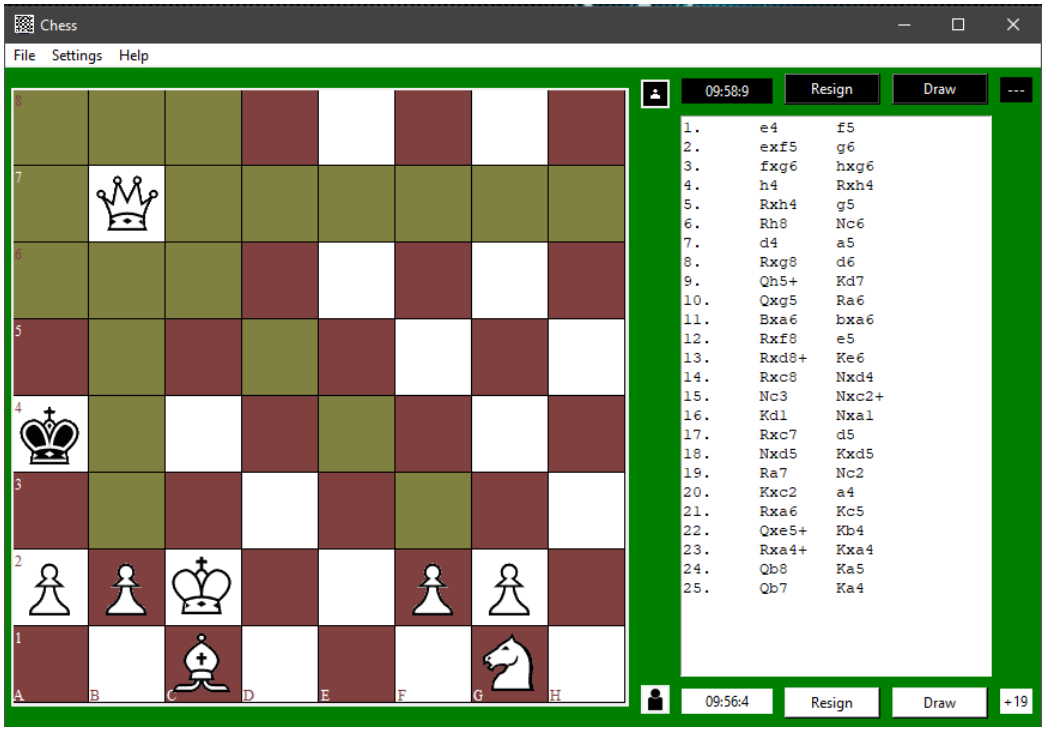


Figure 111: Stalemate - 1

Figure 112: Stalemate - 2

- Selecting AI as Opponent:

Figure 113: Selecting AI as Opponent

- Default AI Settings:

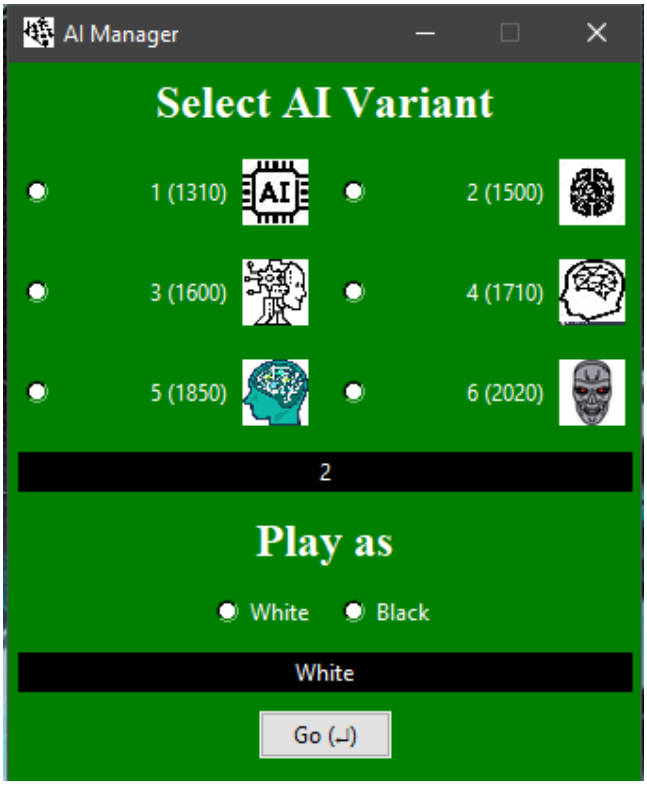


Figure 114: Default AI Settings

- Selecting Different AI Strength:

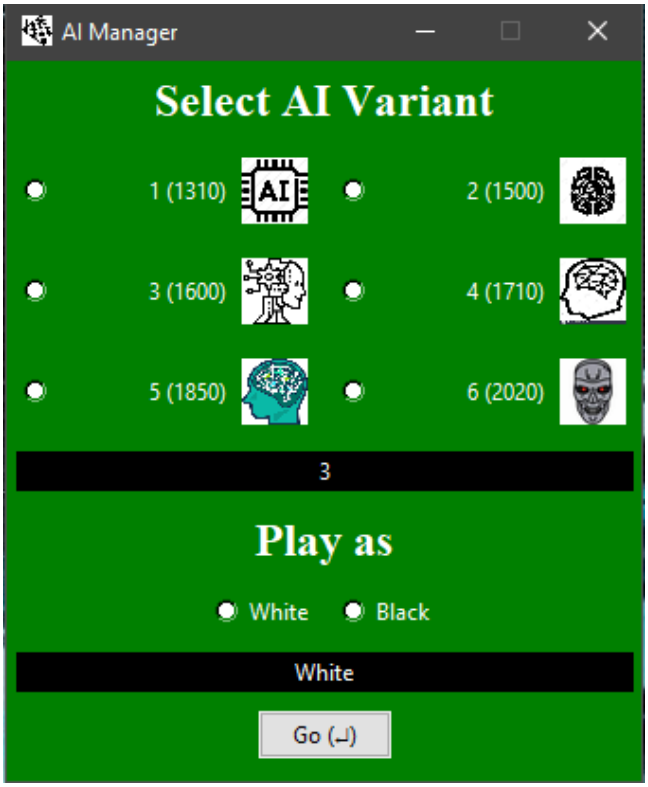


Figure 115: Selecting Different AI Strength

- Match Against AI Started with Human as White:

Figure 116: Match Against AI Started with Human as White

- AI Last Move Highlighted:

Figure 117: AI Last Move Highlighted

- AI Delivering Check to Human:

Figure 118: AI Delivering Check to Human

- Preparing Checkmate Sequence Against AI:

Figure 119: Preparing Checkmate Sequence Against AI - 1

Figure 120: Preparing Checkmate Sequence Against AI - 2

Figure 121: Preparing Checkmate Sequence Against AI - 3

Figure 122: Preparing Checkmate Sequence Against AI - 4

- Save AI vs Human Game FEN:

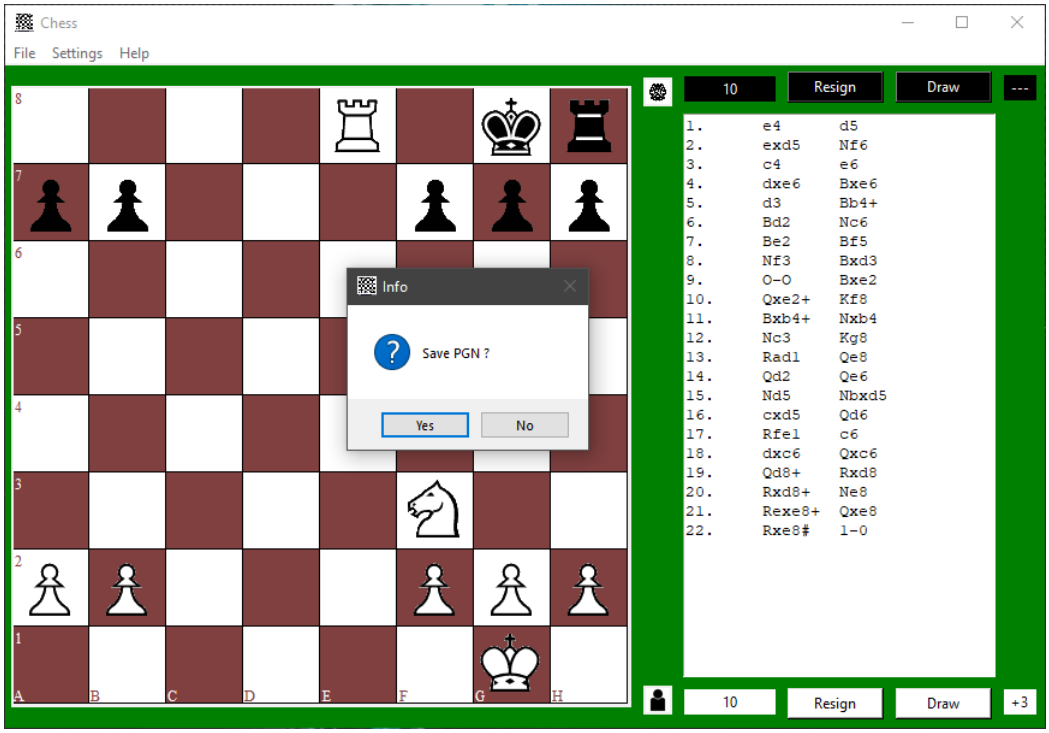


Figure 123: Save AI vs Human Game FEN - 1

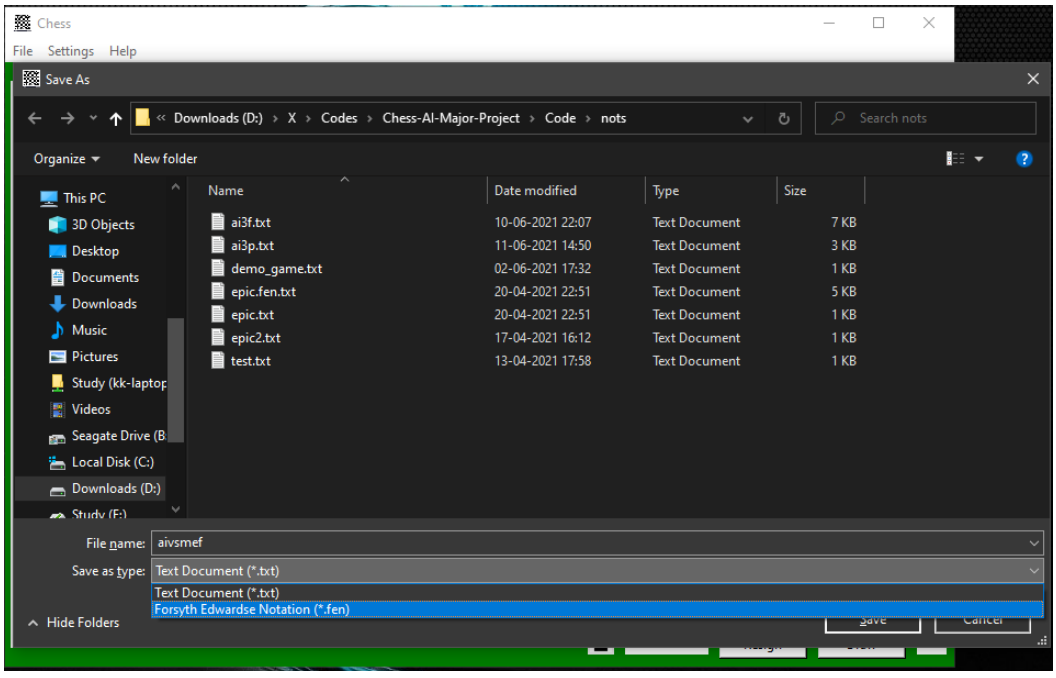


Figure 124: Save AI vs Human Game FEN - 2

- Exiting After Victory:

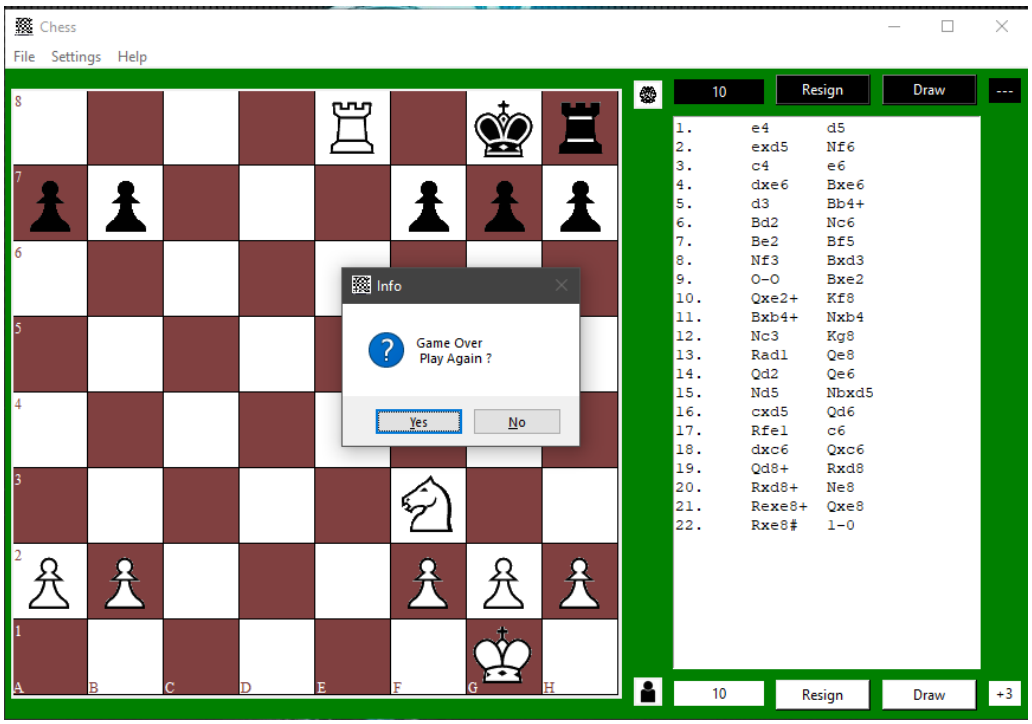


Figure 125: Exiting After Victory

7. System Testing

Testing is the final stage of the application development which plays a vital role in the process of creating high-quality software.

The aspects taken into consideration while carrying out testing of our project involve functional and non-functional areas (the basic functionality of the application), security of the application, usability and performance monitoring, ability to adapt to the multitude of desktops, devices, and operating systems.

Figure 126: Testing Phases

The following different types of testing that have been performed to check the functionality, performance and overall efficiency of our application:

7.1 System Testing

We performed system (compatibility) testing to test our application with the supported software and hardware configurations.
We performed the following:

- OS Configuration

7.1.1. Cross-platform testing:

It allows evaluating the work of the application in different OS:

- Windows
- Mac OS
- Linux

7.2 Module Testing

7.2.1 Modules

We checked each module in our application separately to test its functionality, connection and inter-dependence on other modules.

Modules	Description	%TCs Executed	%TCs Passed	TCs Pending	Priority
Move Generator	Makes user selected move.	100%	100%	0	HIGH
Move Authentication	Ensures only legal moves can be selected from the valid moves.	100%	100%	0	HIGH
Piece / Pawn Promotion	Promote pawn to user desired piece.	100%	100%	0	MEDIUM
Castling	Ensures legal castling moves.	100%	100%	0	MEDIUM
Board Manager	Provides GUI of board.	100%	100%	0	HIGH
Board State Manager	Monitors & Acts for & upon game conclusive actions.	100%	100%	0	HIGH
Clock Manager	Provides features of a chess clock and updates board manager.	100%	100%	0	MEDIUM
G.U.I. Manager	Provides features to modify look & feel of the app.	100%	100%	0	MEDIUM
Report Generation	Generate report in form of PGN & FEN for user.	100%	100%	0	MEDIUM
A.I. Manager	Manages communication between A.I. and G.U.I.	100%	100%	0	HIGH
A.I.	Creates & traverses a tree of moves given situation.	100%	100%	0	HIGH

Table 2: Module Testing

7.2.1 A.I.

A.I.'s accuracy and time have been improved by use of several algorithms.

Figure 127: Match of our A.I. against bot of chess.com

In above match our A.I. won a match against a 1300 rated bot from chess.com with accuracy of 93.7%.

A.I. accuracy range = 80 – 95%

Average Accuracy = 87%

A.I. Algorithms:

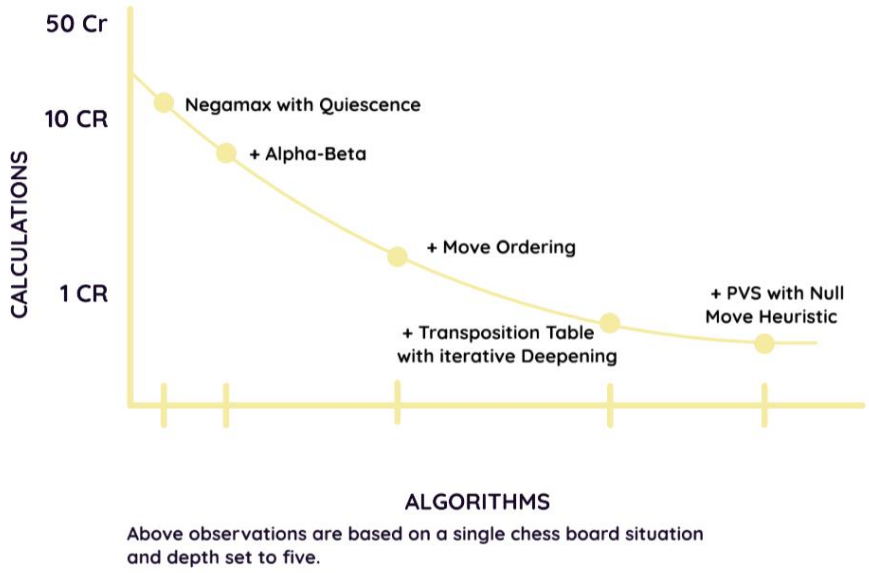


Figure 128: A.I. Algorithms

1. Search Algorithms
    a. Negamax (Negated Minimax) Search
    b. PVS (Principal Variation Search) with Aspiration
    c. Quiescense Search
2. Pruning Algorithms
    a. Alpha-Beta Pruning
    b. Extended Null Move Pruning
3. Extensions
    a. Check Extension
    b. Singular Reply Extension
4. Evaluation Techniques
    a. Material Evaluation
    b. Positional Evaluation
    c. Castling Evaluation
5. Other Techniques
    a. Move Ordering
    b. Iterative Deepening
    c. Transposition Table

7.3 User Testing

We have tested our system with some students and people unfamiliar with software.

7.4 Functionality Testing

We performed functional testing to ensure that each function of our application operated in conformance with the requirement specification. Basically, functionality testing is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (unlike white-box testing).

1. Buttons and Textboxes Testing, where we verified
    - Correctness of buttons.
    - Checking buttons lead to the desired windows.
    - If there are windows that are not referenced.
    - There are no broken buttons.
    - Inputs are being read correctly from input boxes.

2. Testing for all windows, where we verified
    - The input data validity.
    - Allowed values for the data field.
    - Invalid input values for the data field.

3. Code Validation, where we verified
    - Code syntax errors.

7.5 Interface Testing

We performed interface testing to verify that the graphical user interface of our application meets the specifications. Basically, Interface testing ensures that all interactions between the A.I. & Backend and application interfaces are running smoothly.

This includes checking the communication processes as well as making sure that error messages are displayed correctly. Further, interruptions by the user as well as the A.I. & Backend also needs to be tested for correct handling. Some of the verifications we performed are:

- Compliance with the standards of graphical interfaces.
- Design elements evaluation: layout, colors, fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons
- Testing with different screen resolutions.
- Testing the graphical user interface on target devices i.e. desktops, laptops.

8. Conclusion and Future Scope

The application would provide a simple chess playing experience to users. A free application with good A.I. for chess enthusiast. We hope it would provide a satisfactory experience playing against provided A.I. player.

8.1 Future Scope

As of now, this project is a desktop application. The players have to be on the same computer to play against each other.

In the near future:

1. This application could be redesigned as an internet connected or web-based application to make online matches possible.
2. Login and scoring functionality can be added. Friend list functionality can be added so players can easily play with specific players. Functionality to maintain overall score of players can be added.
3. This will make the match making process easier for the players and encourage more people to participate.
4. Functionality to analyse a chess match can be added.
5. Functionality to load from a certain point of a past game can be added.
6. AI can be further improved.

References

[1] https://pypi.org/project/tkintertable/

[2] https://pypi.org/project/python-chess/

[3] https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/

[4] https://www.fide.com/FIDE/handbook/LawsOfChess.pdf

[5] https://www.w3schools.com/python/python_modules.asp

[6] https://www.chessprogramming.org/Minimax

[7] https://www.chessprogramming.org/Negamax

[8] https://pypi.org/project/chess/

[9] https://medium.com/the-innovation/the-anatomy-of-a-chess-ai-2087d0d565#:~:text=The%20minimax%20algorithm%20takes%20advantage,other%20tries%20to%20minimize%20it.

[10] https://www.chessprogramming.org/NegaScout

[11] https://www.chessprogramming.org/Alpha-Beta

[12] https://www.chessprogramming.org/Quiescence_Search

[13] https://medium.com/dscvitpune/lets-create-a-chess-ai-8542a12afef

[14] David-Tabibi, O., & Netanyahu, N. S. (2008, September). Extended null-move reductions. In International Conference on Computers and Games (pp. 205-216). Springer, Berlin, Heidelberg.

[15] https://adamberent.com/2019/03/02/chess-board-evaluation/

[16] https://readthedocs.org/projects/python-chess/downloads/pdf/stable/

[17] https://medium.com/analytics-vidhya/python-tips-multithreading-vs-multiprocessing-data-sharing-tutorial-52743ed48825

[18] https://www.chessprogramming.org/PolyGlot

[19] https://www.chessprogramming.org/Piece-Square_Tables