

CHESS GAME

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

MANI SURYA TEJA KOTA (37110369)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND
TECHNOLOGY (DEEMED TO BE
UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by
AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

March 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC | 12B Status by UGC | Approved by
AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of MANI SURYA TEJA (37110369) who carried out the project entitled “CHESS GAME” under my supervision from November 2020 to March 2021.

Internal Guide

Dr. B. BHARATHI, M.E., Ph.D.,

Head of the Department

Dr. S.Vigneshwari M.E., Ph.D.,

Head of the Department

Dr. L.Lakshmanan M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **MANI SURYA TEJA KOTA (37110369)** hereby declare that the Project Report entitled “**CHESS GAME**” is done by me under the guidance of **Dr. B. BHARATHI, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D., and Dr.S.Vigneshwari M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. B. BHARATHI, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Based upon the game chess. Here the requirements are quiet simple, just a python compiler is enough to program this CHESS game. I used a recent latest version of python compiler (python 3.7.2). This game contains as generally a chess game has, all pieces are named in the format of piece color. The „random' package is used in this game, which is used to randomized our chances or player turns to play. We have to enter the inputs to play the game are 1. piece name 2. initial x, y values 3. destination x, y values.

So the key of the game is purely based upon the co-ordinates of x y. And firstly this game is k-sensitive. And a single mistake of entering piece name will not let us to do it again. And the output is presented by printable named function which also represents the chess board along with the value update. And the rules of each piece is the only time taking and also hardest part the game, cause for example piece „rook" 'have to move only either the up-side, down-side, right-side or left-side, so in the co-ordinates of x and y either x value should vary or y value should vary. And here the problem is if in between the initial and destination point if any same side of the player"s piece contain then castle should stop in possible front of the piece. So that every step is must and should counted to check whether any piece contains or not. And the further information is detailed in the introduction part of the game.

TABLE OF CONTENTS

S.NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
1.	CHAPTER 1: INTRODUCTION	
	1.1 INTRODUCTION	1
	1.2 RESEARCH AND SIGNIFICANCE	1
2.	CHAPTER 2: LITERATURE SURVEY	2
3.	CHAPTER 3: AIM AND SCOPE	
	3.1 AIM	3
	3.2 SCOPE	3
4.	CHAPTER 4: SYSTEM DESIGN AND METHODOLOGY	
	4.1 PROPOSED SYSTEM	4
	4.2 REQUIREMENT SPECIFICATION	4
	4.2.1 SOFTWARE REQUIREMENTS	4
	4.2.2 HARDWARE REQUIREMENTS	5
	4.3 EXISTING SYSTEM	5
5.	CHAPTER 5: SYSTEM DEVELOPMENT METHODOLOGY	
	5.1 SYSTEM ARCHITECTURE	6
6.	CHAPTER 6: RESULTS AND DISCUSSIONS	
	6.1 MODULE DESCRIPTION	8
	6.1.1 Game Intro Module	8
	6.1.2 User Input module	9
	6.1.3 Step Validation module	9
	6.1.4 Game Updation module	10
	6.1.5 Winner declaration module	14

7.	CHAPTER 7: CONCLUSION	15
	REFERENCES	16
	APPENDIX	
	1. PAPER ACCEPTANCE MAIL	17
	2. PLAGARISM REPORT	18
	3. SOURCE CODE	19
	4. PAPER ACCEPTANCE	31

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
5.1	CHESS ARCHITECTURE	6
6.1	GAME BOARD	8
6.2	GAME INTRO MODULE	9
6.3	USER INPUT MODULE	9
6.4.1	SIGNAL 0	10
6.4.2	SIGNAL 1	11
6.4.3	SIGNAL 11	12
6.4.4	SIGNAL 22	13
6.4.6	WINNER DECLARATION MODULE	14

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Overview: Based upon the game chess. Here the requirements are quite simple, just a python compiler is enough to program this CHESS game. I used a recent latest version of python compiler (python 3.7.2). This game contains as generally a chess game has, all pieces are named in the format of (Piece name.colour). The „random' package is used in this game, which is used to randomized our chances or player turns to play. We have to enter the inputs to play the game are 1. piece name 2. initial x, y values 3. destination x, y values.

So the key of the game is purely based upon the co-ordinates of x y. And firstly this game is k-sensitive. And a single mistake of entering piece name will not let us to do it again. And the output is presented by printable named function which also represents the chess board along with the value update. And the rules of each piece is the only time taking and also hardest part the game, cause for example piece „rook" 'have to move only either the up-side, down-side, right-side or left-side, so in the co-ordinates of x and y either x value should vary or y value should vary. And here the problem is if in between the initial and destination point if any same side of the player"s piece contain then castle should stop in possible front of the piece. So that every step is must and should counted to check whether any piece contains or not. And the further information is detailed in the introduction part of the game.

1.2 RESEARCH AND SIGNIFICANCE

The main aim of this application I.e. to create a own and customizable chess engine, which is easy to understand and code. And also to provide a user friendly interface for the user while playing.

CHAPTER 2

LITERATURE SURVEY

Literature survey is that the most significant step in code development method. Before developing the tool it's necessary to see the time issue, economy and company strength. Once these items are happy, then consecutive step is to see that OS and language may be used for developing the tool.

Once the programmers begin building the tool the programmers need ton of external support. This support are obtained from senior programmers, from book or from websites.

Before building the system the on prime of thought unit taken into thought for developing the planned system. the main half of the project development sector considers and totally survey all the desired desires for developing the project. for each project Literature survey is that the most significant sector in code development method.

Before developing the tools and therefore the associated coming up with it's necessary to work out and survey the time issue, resource demand, man power, economy, and company strength.

Once these items area unit glad and absolutely surveyed, then consecutive step is to see concerning the software package specifications within the individual system like what form of software system the project would need, and what area unit all the required software package area unit required to proceed with consecutive step like developing the tools, and also the associated operations.

CHAPTER 3

AIM AND SCOPE

3.1 AIM

The objective of this project is to create a own Chess engine which is user friendly and also to make user comfortable. And this project is a two player game. And prediction for game winner is to display for every move.

3.2 SCOPE

The scope of the project is as follows:

Scope of this project is to make a own chess engine which is customizable like to implement the rules like castling in the game. And to make game is for entertainment purpose. To provide a friendly environment where user can improve game skills with more interest.

CHAPTER 4

SYSTEM DESIGN AND METHODOLOGY

4.1 PROPOSED SYSTEM

This system is proposed with certain features which makes user to feel comfortable.

The proposed system is a engine which can detect whether the present move is right or wrong.

Checks the each and every move is going to be the part of winning prediction of the player.

Notifications like player moves and taken opponent's chess piece is displayed.

Illegal moves are not considered and given chance to make a perfect move.

Understandable code for everyone because of co-ordinates geometry concept.

4.2 REQUIREMENT SPECIFICATION

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the user"s machines already. So, the main need is to introduce appropriate equipment for the product. The requirements are split into two categories, namely:

4.2.1 SOFTWARE REQUIREMENTS

The basic software requirements to run the program are:

1. Windows 10
2. Pycharm and Modules like random, pygame are needed to be imported.

4.2.2 HARDWARE REQUIREMENTS

The basic hardware required to run the program are:

1. Hard disk of 5 GB
2. System memory (RAM) of 512 MB
3. I3 processor-based computer or higher.

4.3 EXISTING SYSTEM

In the present scenario, python has its module called chess which can do all the operations on the chess moves. But graphically its not. And it is the most used module for the chess game and for chess study purpose.

CHAPTER 5

SOFTWARE DEVELOPMENT METHODOLOGY

5.1 SYSTEM ARCHITECTURE

The project is implemented by a combination of Pycharm and other modules like pygame and random. It contains many files separate folder for images to represent the pieces and its movements, and another for the source code. The code follows CHESS architecture. Below in Figure 5.1 the architecture diagram is shown.

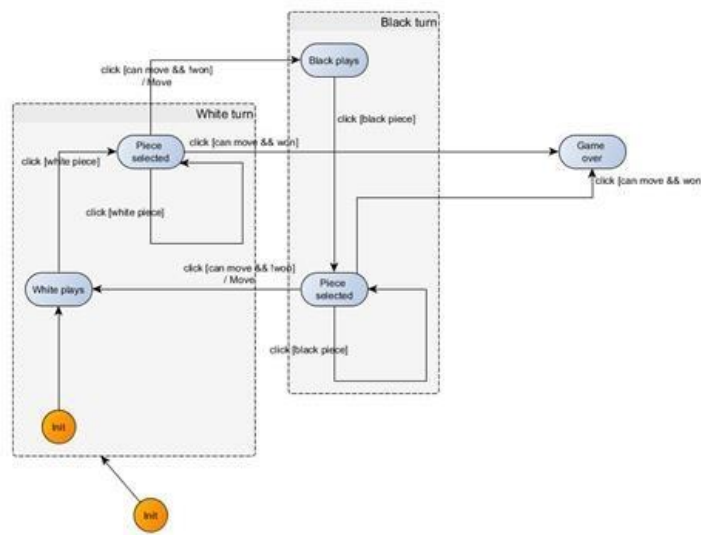


Figure 5.1: CHESS ARCHITECTURE

CHESS game is done by based on 2-Dimensional array. I took a[8][8] array to have 8*8 square box or to represent the board. And placing is done by referencing a normal chess board, pieces like castle, bishop, knight, king, queen and pawn are their entire different rules. And win flag is applied to declare the winner. Finally who kills the king he'll be the winner of the game and announced at last.

KEYS:

CASTLE - The movement of the castle must be either in straight line or horizontal

Line like [x1][y1] is to move. so,if the difference is dx or dy movement must be [x1+dx][y1] or [x1][y1+dy].

KNIGHT - The movement of the knight must be in shape of „L" in any direction and

be like [x1][y1] is to move. So, general movement is [x1+2][y1+1] or [x1+2][y1-1] or [x1+2][y1+1] or [x1-2][y1+1] or [x1-2][y1-1] or [x1+2][y1-1] or [x1-2][y1-1].

BISHOP - The movement of the bishop must be in diagonal in any direction and be

like [x1][y1] is to move. So, general movement if dx and dy are equal [x1+dx][y1+dy].

QUEEN - The movement of queen is a combination of castle and bishop. So, the

general movement if dx and dy are the difference, be like [x1+dx][y1] or [x1][y1+dy] or [x1+dx][y1+dy].

KING - The movement of king must have to move one step in any direction so,

move must be in [x1+1][y] or [x1][y+1] or [x1+1][y-1] or [x1-1][y-1] or [x1+1][y+1] or [x1-1][y+1]

CHAPTER 6

RESULTS AND DISCUSSION



FIGURE 6.1 GAME BOARD

MODULE DESCRIPTION:

- Game Intro Module
- User Input module
- Step Validation module
- Game Updation module

- Winner deaclaration module

Game Intro Module:

It is important to know the players information to specify them during the game. So in this module players information is taken and saved for future purpose. And also to change the chances of players to play.

```
"C:\Users\KOTA SRI SURYA TEJA\PycharmProjects\pythonProject\venv\Scripts\python.exe"  
pygame 2.0.1 (SDL 2.0.14, Python 3.8.6)  
Hello from the pygame community. https://www.pygame.org/contribute.html  
ENTER YOUR NAME PLAYER 1 Mani  
ENTER YOUR NAME PLAYER 2 Surya  
Mani turn (whites)  
|
```

FIGURE 6.2 GAME INTRO MODULE

User Input module:

User Input module is used to ensure the event of every mouse is not for the chess piece movement. And it captures the coordinates whenever it selects on the chess board and sends the coordinates for the validation, to check whether it's a legal move or not. And this project is made with window size of (512 px,512 px).

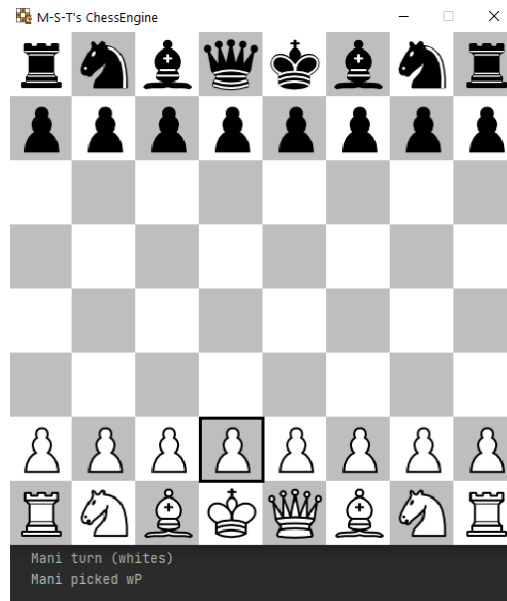


FIGURE 6.3 USER INPUT MODULE

Step Validation module:

This module provides signal whether the move is valid/legal or not. And in this module there are multiple signals given for the input move. There are totally 4 types of signals / flags . They are :-

1. Signal 0:

This signal is for to indicate the input move / step is not valid / legal. And also if the player unwantedly selects the opponent"s chess piece it generates a signal 0 and displays the message that "wrong move".

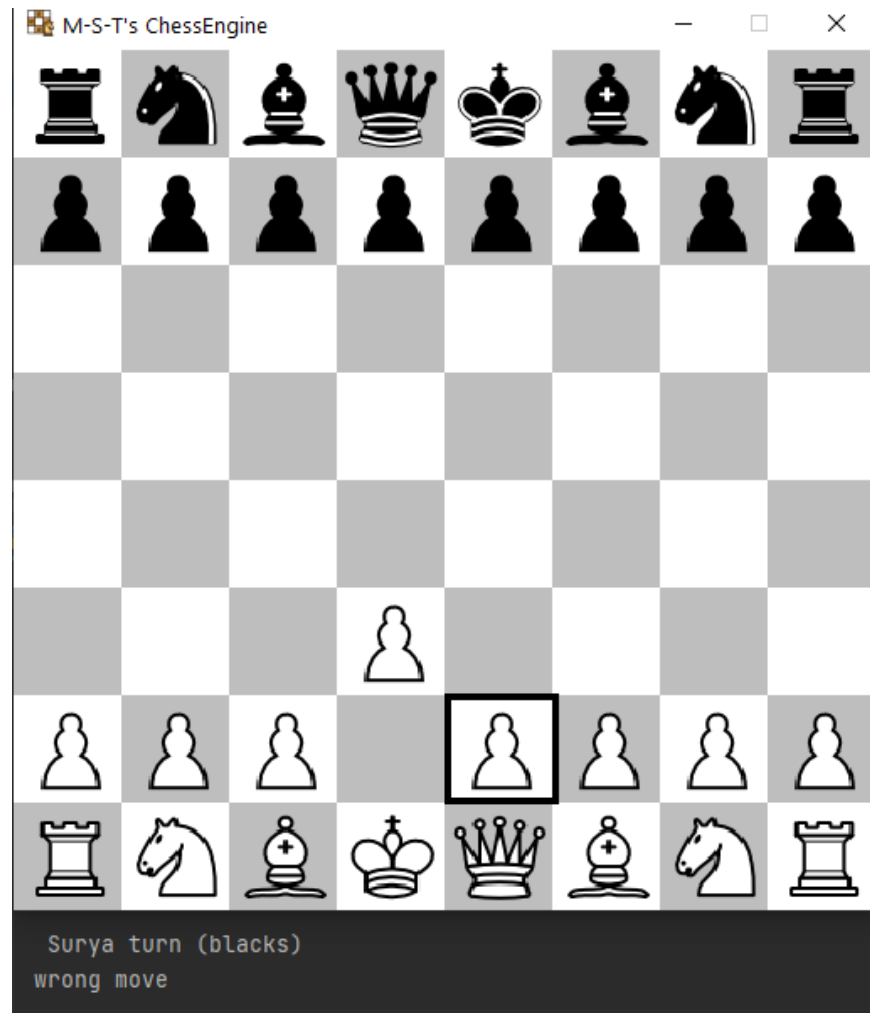


FIGURE 6.4.1 SIGNAL 0

2. Signal 1:

This signal is for the chess piece whether it can move to its destination with no interaction of any chess piece in its way. If it can made the destination, then signal is generated and message is generated that it made a safe move.

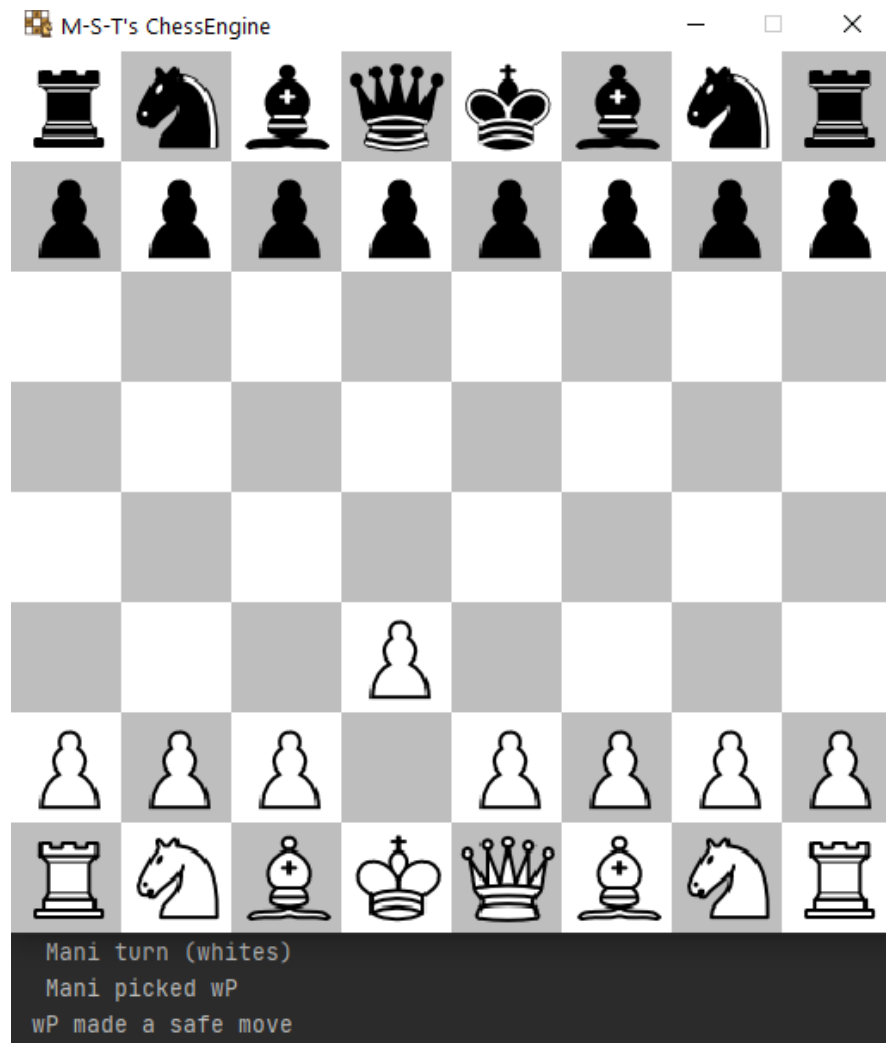


FIGURE 6.4.2 SIGNAL 1

3. *Signal 11:*

This signal is for chess piece whether the player1 piece captured or taken the opponents piece. If this condition is true then system generates a message like “[player1 piece] killed [player2 piece]”.



FIGURE 6.4.3 SIGNAL 11

4. Signal 22:

This signal is for chess piece whether the player1 piece is perfectly captured the opponent's king. If it is done it returns a signal 22. Finally winner declaration will be done and game will be over.



FIGURE 6.4.4 SIGNAL 22

And after completion of checking these validations it goes to the chess board for the next step updation of the chess board. So that it can move chess piece in the code and display over the window.

Game Updation Module:

This module is used to update the chess board which is used to display current state of board for every move. There will be done two things.

- ◆— Either to swap the piece with empty square.
- ◆— Or two replace the opponents piece with the players piece.

Winner declaration module:

This module is used to fetch the game all the time to check whether the opponent's king is been captured or not. If yes, then game is closed and displays the message like "[winning player's name] win"

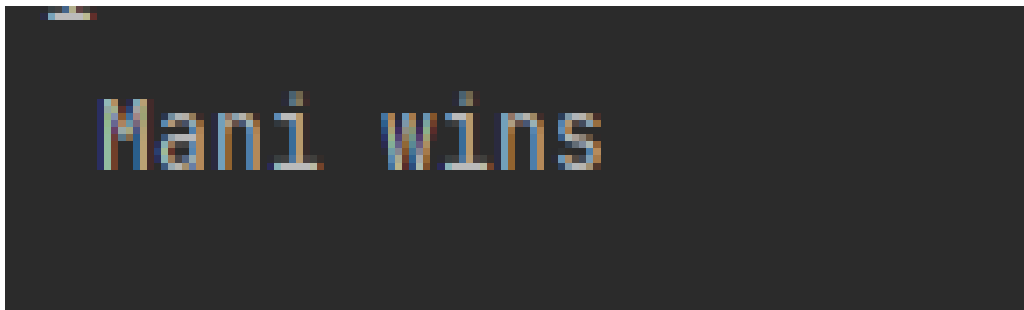


FIGURE 6.4.6 WINNER DECLARATION MODULE

CHAPTER 7

CONCLUSION

By checking all the possible cases to run the chess engine. This project running successfully without any flaws. And also can be done in advanced way.

It is a chess game with more benefits. Firstly, its more benefitable for the ones who wants and tries to study the code and logic behind the code. And its GUI programming of the pygame is more flexible than the “Ktinker module” for gaming interface.

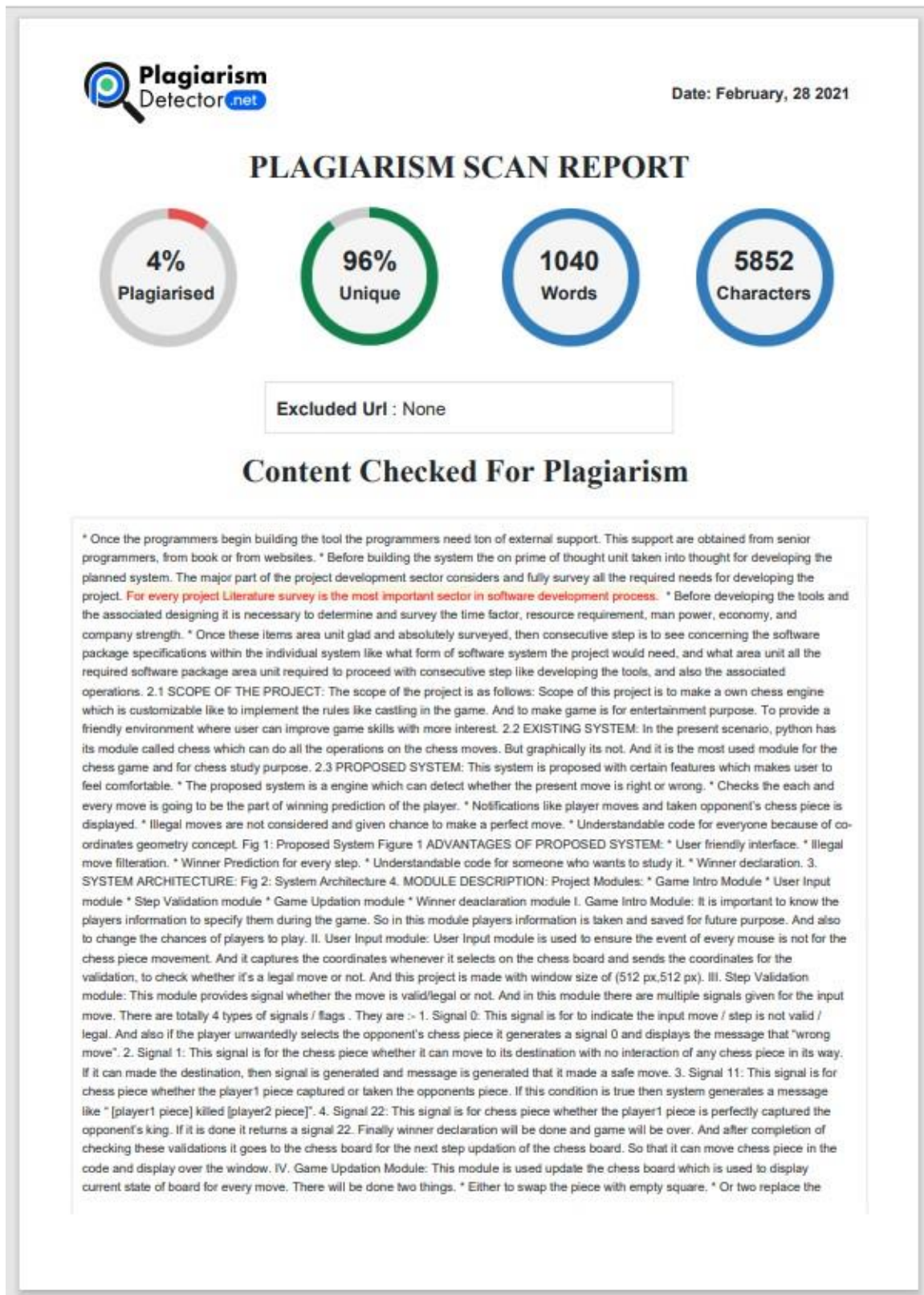
REFERENCES

- [1] Co-ordinates concept in geometry.
- [2] Mathematical logics on co-ordinates.
- [3] Pygame module tutorials on youtube.
<https://youtu.be/FfWpgLFMI7w> (link)

APPENDIX

1.PAPER ACCEPTANCE

2. PLAGIARISM REPORT



SOURCE CODE

```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows,
actions, and settings.

import pygame as p

s1=[]
s1.append(input('ENTER YOUR NAME PLAYER 1'))
s1.append(input('ENTER YOUR NAME PLAYER 2'))
W = [0]
B = [0]

p.init()
h = w = 512
s = p.display.set_mode((512, 512))
p.display.set_caption("M-S-T's ChessEngine")
l=p.image.load('chess-board.png')
p.display.set_icon(l)
c = [p.Color('white'), p.Color('grey')]
sq = h//8
pi = ['bR', 'bN', 'bB', 'bQ', 'bK', 'bP', 'wR', 'wK', 'wB', 'wQ', 'wN',
      'wP']
b_board = [['bR', 'bN', 'bB', 'bQ', 'bK', 'bB', 'bN', 'bR'],
            ['bP', 'bP', 'bP', 'bP', 'bP', 'bP', 'bP', 'bP'],
            [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
            [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
            [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
            [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
            ['wP', 'wP', 'wP', 'wP', 'wP', 'wP', 'wP', 'wP'],
            ['wR', 'wN', 'wB', 'wK', 'wQ', 'wB', 'wN', 'wR']]
pics = {}
for i in pi:
    pics[i] = p.transform.scale(p.image.load(i+'.png'), (64, 64))

def chess_board():
    for i in range(8):
        for j in range(8):
            c1 = c[(i+j) % 2]
            p.draw.rect(s, c1, p.Rect(j*sq, i*sq, sq, sq))

    for i in range(8):
        for j in range(8):
            if b_board[i][j] != ' ':
                s.blit(pics[b_board[i][j]], p.Rect(j*sq, i*sq, 64, 64))

f = 1
prev = [-1, -1]
```

```

def validation(a, b):
    m = b_board[a[0]][a[1]]
    if m[0] == 'b':
        if m[1] == 'P':
            if a[0] == 1 and b[0] == 3 and a[1] == b[1] and
b_board[a[0]+1][a[1]] == ' ' and b_board[b[0]][b[1]] == ' ':
                return 1
            elif a[0]+1 == b[0] and a[1] == b[1] and b_board[b[0]][b[1]] ==
' ':
                return 1
            elif a[0]+1 == b[0] and (a[1]+1 == b[1] or a[1]-1 == b[1]) and
b_board[b[0]][b[1]][0] == 'w':
                if b_board[b[0]][b[1]][1] == 'K':
                    return 22
                return 11
            return 0

        elif m[1] == 'R':
            if a[0] == b[0] or a[1] == b[1]:
                f=0
                c=[0,0,0,0]
                while a[1]!=b[1] or a[0]!=b[0]:
                    if a[0]+(c[0]+1)<b[0] and
b_board[a[0]+(c[0]+1)][a[1]]==' ' and b_board[a[0]+(c[0]+1)][a[1]][0]!='b'
and (f==0 or f==1):
                        c[0]+= 1
                        f = 1
                        elif a[0]+(c[1]-1)>b[0] and b_board[a[0]+(c[1]-
1)][a[1]]==' ' and b_board[a[0]+(c[1]-1)][a[1]][0]!='b' and (f==0 or
f==2):
                            f = 2
                            c[1]-=1
                            elif a[1]+(c[2]+1)<b[1] and
b_board[a[0]][a[1]+(c[2]+1)]==' ' and b_board[a[0]][a[1]+(c[2]+1)][0]!='b'
and (f==0 or f==3):
                                f = 3
                                c[2]+=1
                                elif a[1]+(c[3]-1)>b[1] and b_board[a[0]][a[1]+(c[3]-
1)]==' ' and b_board[a[0]][a[1]+(c[3]-1)][0]!='b' and (f==0 or f==4):
                                    f = 4
                                    c[3]-=1
                                else:
                                    if f<=2 and (a[0]+(c[f-1]+1)==b[0] or a[0]+(c[f-1]-
1)==b[0]) and a[1]==b[1]:
                                        if b_board[b[0]][b[1]] == 'wK':
                                            return 22
                                        elif b_board[b[0]][b[1]][0] == 'w':
                                            return 11
                                        elif b_board[b[0]][b[1]]==' ':
                                            return 1
                                        elif f>2 and a[0]==b[0] and (a[1]+(c[f-1]+1)==b[1]
or a[1]+(c[f-1]-1) == b[1]):
                                            if b_board[b[0]][b[1]] == 'wK':
                                                return 22
                                            elif b_board[b[0]][b[1]][0] == 'w':

```

```

        return 11
    elif b_board[b[0]][b[1]] == ' ':
        return 1
    else:
        if abs(a[0]-b[0])==1 or abs(a[1]-b[1])==1:
            if b_board[b[0]][b[1]] == 'wK':
                return 22
            elif b_board[b[0]][b[1]][0] == 'w':
                return 11
            elif b_board[b[0]][b[1]] == ' ':
                return 1
        return 0
    return 0

elif m[1]=='N':
    d1 = a[0]-b[0]
    d2 = a[1]-b[1]
    if (d1==1 and d2==2) or (d1==2 and d2==1) or (d1==-2 and d2==1)
or (d1==2 and d2==-1) or (d1==-1 and d2==2) or (d1==-2 and d2==-1) or
(d1==-1 and d2==2) or (d1==1 and d2==2):
        if b_board[b[0]][b[1]]==' ':
            return 1
        elif b_board[b[0]][b[1]]=='wK':
            return 22
        elif b_board[b[0]][b[1]][0]=='w':
            return 11
    return 0

elif m[1]=='B':
    d1 = (b[0]-a[0])
    d2 = (b[1]-a[1])
    c=0
    f=0
    while (abs(d1)==abs(d2)) !=0:
        if abs(d1)==abs(d2) and c<abs(d1) and d1>0 and d2>0 and
b_board[a[0]+(c+1)][a[1]+(c+1)]==' ' and (f==0 or f==1):
            c+=1
            f=1
        elif abs(d1)==abs(d2) and c<abs(d1) and d1<0 and d2>0 and
b_board[a[0]-(c+1)][a[1]+(c+1)]==' ' and (f==0 or f==2):
            c+=1
            f=2
        elif abs(d1)==abs(d2) and c<abs(d1) and d1>0 and d2<0 and
b_board[a[0]+(c+1)][a[1]-(c+1)]==' ' and (f==0 or f==3):
            c+=1
            f=3
        elif abs(d1)==abs(d2) and c<abs(d1) and d1<0 and d2<0 and
b_board[a[0]-(c+1)][a[1]-(c+1)]==' ' and (f==0 or f==4):
            c+=1
            f=4
    elif abs(d1)==1:
        if b_board[b[0]][b[1]]==' ':
            return 1
        elif b_board[b[0]][b[1]]=='wK':
            return 22
        elif b_board[b[0]][b[1]][0]=='w':
            return 11

```

```

        else:
            return 0
    else:
        break
    if f>0:
        if b_board[b[0]][b[1]] == ' ':
            return 1
        elif b_board[b[0]][b[1]] == 'wK':
            return 22
        elif b_board[b[0]][b[1]][0] == 'w':
            return 11
    return 0

    if m[1]=='Q':
        d1=(b[0]-a[0])
        d2=(b[1]-a[1])
        if abs(d1)==abs(d2):
            c = 0
            f = 0
            while (abs(d1) == abs(d2)) != 0:
                if abs(d1) == abs(d2) and c < abs(d1) and d1 > 0 and d2
> 0 and b_board[a[0] + (c + 1)][a[1] + (c + 1)] == ' ' and (f == 0 or f ==
1):
                    c += 1
                    print(a, b, c)
                    f = 1
                elif abs(d1) == abs(d2) and c < abs(d1) and d1 < 0 and
d2 > 0 and b_board[a[0] - (c + 1)][
a[1] + (c + 1)] == ' ' and (f == 0 or f == 2):
                    c += 1
                    f = 2
                elif abs(d1) == abs(d2) and c < abs(d1) and d1 > 0 and
d2 < 0 and b_board[a[0] + (c + 1)][
a[1] - (c + 1)] == ' ' and (f == 0 or f == 3):
                    c += 1
                    f = 3
                elif abs(d1) == abs(d2) and c < abs(d1) and d1 < 0 and
d2 < 0 and b_board[a[0] - (c + 1)][
a[1] - (c + 1)] == ' ' and (f == 0 or f == 4):
                    c += 1
                    f = 4
            elif abs(d1) == 1:
                if b_board[b[0]][b[1]] == ' ':
                    return 1
                elif b_board[b[0]][b[1]] == 'wK':
                    return 22
                elif b_board[b[0]][b[1]][0] == 'w':
                    return 11
            else:
                return 0
        else:
            break
    if f > 0:
        if b_board[b[0]][b[1]] == ' ':
            return 1
        elif b_board[b[0]][b[1]] == 'wK':

```

```

        return 22
    elif b_board[b[0]][b[1]][0] == 'w':
        return 11
    elif d1==0 or d2==0:
        f = 0
        c = [0, 0, 0, 0]
        while a[1] != b[1] or a[0] != b[0]:
            if a[0] + (c[0] + 1) < b[0] and b_board[a[0] + (c[0] +
1)][a[1]] == ' ' and \
                b_board[a[0] + (c[0] + 1)][a[1]][0] != 'b' and
(f == 0 or f == 1):
                c[0] += 1
                f = 1
            elif a[0] + (c[1] - 1) > b[0] and b_board[a[0] + (c[1]
- 1)][a[1]] == ' ' and \
                b_board[a[0] + (c[1] - 1)][a[1]][0] != 'b' and
(f == 0 or f == 2):
                f = 2
                c[1] -= 1
            elif a[1] + (c[2] + 1) < b[1] and b_board[a[0]][a[1] +
(c[2] + 1)] == ' ' and \
                b_board[a[0]][a[1] + (c[2] + 1)][0] != 'b' and
(f == 0 or f == 3):
                f = 3
                c[2] += 1
            elif a[1] + (c[3] - 1) > b[1] and b_board[a[0]][a[1] +
(c[3] - 1)] == ' ' and \
                b_board[a[0]][a[1] + (c[3] - 1)][0] != 'b' and
(f == 0 or f == 4):
                f = 4
                c[3] -= 1
        else:
            if f <= 2 and (a[0] + (c[f - 1] + 1) == b[0] or
a[0] + (c[f - 1] - 1) == b[0]) and a[1] == b[1]:
                if b_board[b[0]][b[1]] == 'wK':
                    return 22
                elif b_board[b[0]][b[1]][0] == 'w':
                    return 11
                elif b_board[b[0]][b[1]] == ' ':
                    return 1
            elif f > 2 and a[0] == b[0] and (
a[1] + (c[f - 1] + 1) == b[1] or a[1] +
(c[f - 1] - 1) == b[1]):
                if b_board[b[0]][b[1]] == 'wK':
                    return 22
                elif b_board[b[0]][b[1]][0] == 'w':
                    return 11
                elif b_board[b[0]][b[1]] == ' ':
                    return 1
            else:
                if abs(a[0] - b[0]) == 1 or abs(a[1] - b[1]) ==
1:
                    if b_board[b[0]][b[1]] == 'wK':
                        return 22
                    elif b_board[b[0]][b[1]][0] == 'w':
                        return 1

```

```

        return 0
    return 0

    elif m[1]=='K':
        d1=a[0]-b[0]
        d2=a[1]-b[1]
        if (abs(d1)<2 and abs(d2)<2) and (a[0]!=b[0] or a[1]!=b[1]):
            if b_board[b[0]][b[1]]=='wK':
                return 22
            elif b_board[b[0]][b[1]][0]=='w':
                return 11
            elif b_board[b[0]][b[1]]==' ':
                return 1
        return 0

    if m[0]=='w':
        if m[1]=='P':
            if a[0]==6 and b[0]==4 and a[1]==b[1] and b_board[a[0]-1][a[1]]==' ' and b_board[b[0]][b[1]]==' ':
                return 1
            elif a[0]-1==b[0] and a[1]==b[1] and b_board[b[0]][b[1]]==' ':
                return 1
            elif a[0]-1==b[0] and (a[1]-1==b[1] or a[1]+1==b[1]) and b_board[b[0]][b[1]][0]=='b':
                if b_board[b[0]][b[1]][1]=='K':
                    return 22
                return 11
            return 0

        elif m[1] == 'R':
            if a[0] == b[0] or a[1] == b[1]:
                f=0
                c=[0,0,0,0]
                while a[1]!=b[1] or a[0]!=b[0]:
                    if a[0]+(c[0]+1)<b[0] and b_board[a[0]+(c[0]+1)][a[1]]==' ' and b_board[a[0]+(c[0]+1)][a[1]][0]!='w' and (f==0 or f==1):
                        c[0]+= 1
                        f = 1
                    elif a[0]+(c[1]-1)>b[0] and b_board[a[0]+(c[1]-1)][a[1]]==' ' and b_board[a[0]+(c[1]-1)][a[1]][0]!='w' and (f==0 or f==2):
                        f = 2
                        c[1]-=1
                    elif a[1]+(c[2]+1)<b[1] and b_board[a[0]][a[1]+(c[2]+1)]==' ' and b_board[a[0]][a[1]+(c[2]+1)][0]!='w' and (f==0 or f==3):
                        f = 3
                        c[2]+=1
                    elif a[1]+(c[3]-1)>b[1] and b_board[a[0]][a[1]+(c[3]-1)]==' ' and b_board[a[0]][a[1]+(c[3]-1)][0]!='w' and (f==0 or f==4):
                        f = 4
                        c[3]-=1
                    else:
                        if f<=2 and (a[0]+(c[f-1]+1)==b[0] or a[0]+(c[f-1]-1)==b[0]) and a[1]==b[1]:
                            if b_board[b[0]][b[1]] == 'bK':

```

```

        return 22
        elif b_board[b[0]][b[1]][0] == 'b':
            return 11
        elif b_board[b[0]][b[1]]==' ':
            return 1
        elif f>2 and a[0]==b[0] and (a[1]+(c[f-1]+1)==b[1] or
a[1]+(c[f-1]-1) == b[1]):
            if b_board[b[0]][b[1]] == 'bK':
                return 22
            elif b_board[b[0]][b[1]][0] == 'w':
                return 11
            elif b_board[b[0]][b[1]] == ' ':
                return 1
        else:
            if abs(a[0]-b[0])==1 or abs(a[1]-b[1])==1:
                if b_board[b[0]][b[1]] == 'bK':
                    return 22
                elif b_board[b[0]][b[1]][0] == 'b':
                    return 11
                elif b_board[b[0]][b[1]] == ' ':
                    return 1
            return 0
    return 0

elif m[1]=='N':
    d1 = a[0]-b[0]
    d2 = a[1]-b[1]
    if (d1==1 and d2==2) or (d1==2 and d2==1) or (d1==2 and d2==1) or
(d1==2 and d2==1) or (d1==1 and d2==2) or (d1==2 and d2==1) or (d1==1
and d2==2) or (d1==1 and d2==2):
        if b_board[b[0]][b[1]]==' ':
            return 1
        elif b_board[b[0]][b[1]]=='wK':
            return 22
        elif b_board[b[0]][b[1]][0]=='w':
            return 11
    return 0

elif m[1]=='B':
    d1 = (b[0]-a[0])
    d2 = (b[1]-a[1])
    c=0
    f=0
    while (abs(d1)==abs(d2))!=0:
        if abs(d1)==abs(d2) and c<abs(d1) and d1>0 and d2>0 and
b_board[a[0]+(c+1)][a[1]+(c+1)]==' ' and (f==0 or f==1):
            c+=1
            print(a,b,c)
            f=1
        elif abs(d1)==abs(d2) and c<abs(d1) and d1<0 and d2>0 and
b_board[a[0]-(c+1)][a[1]+(c+1)]==' ' and (f==0 or f==2):
            c+=1
            f=2
        elif abs(d1)==abs(d2) and c<abs(d1) and d1>0 and d2<0 and
b_board[a[0]+(c+1)][a[1]-(c+1)]==' ' and (f==0 or f==3):

```

```

        c+=1
        f=3
        elif abs(d1)==abs(d2) and c<abs(d1) and d1<0 and d2<0 and
b_board[a[0]-(c+1)][a[1]-(c+1)]==' ' and (f==0 or f==4):
            c+=1
            f=4
            elif abs(d1)==1:
                if b_board[b[0]][b[1]]==' ':
                    return 1
                elif b_board[b[0]][b[1]]=='bK':
                    return 22
                elif b_board[b[0]][b[1]][0]=='b':
                    return 11
                else:
                    return 0
            else:
                break
    if f>0:
        if b_board[b[0]][b[1]] == ' ':
            return 1
        elif b_board[b[0]][b[1]] == 'bK':
            return 22
        elif b_board[b[0]][b[1]][0] == 'b':
            return 11

    return 0

elif m[1]=='Q':
    d1 = (b[0] - a[0])
    d2 = (b[1] - a[1])
    if abs(d1) == abs(d2):
        c = 0
        f = 0
        while (abs(d1) == abs(d2)) != 0:
            if abs(d1) == abs(d2) and c < abs(d1) and d1 > 0 and d2 >
0 and b_board[a[0] + (c + 1)][
                a[1] + (c + 1)] == ' ' and (f == 0 or f == 1):
                    c += 1
                    print(a, b, c)
                    f = 1
            elif abs(d1) == abs(d2) and c < abs(d1) and d1 < 0 and d2
> 0 and b_board[a[0] - (c + 1)][
                a[1] + (c + 1)] == ' ' and (f == 0 or f == 2):
                    c += 1
                    f = 2
            elif abs(d1) == abs(d2) and c < abs(d1) and d1 > 0 and d2
< 0 and b_board[a[0] + (c + 1)][
                a[1] - (c + 1)] == ' ' and (f == 0 or f == 3):
                    c += 1
                    f = 3
            elif abs(d1) == abs(d2) and c < abs(d1) and d1 < 0 and d2
< 0 and b_board[a[0] - (c + 1)][
                a[1] - (c + 1)] == ' ' and (f == 0 or f == 4):
                    c += 1
                    f = 4
            elif abs(d1) == 1:
                if b_board[b[0]][b[1]] == ' ':
                    return 1

```

```

        elif b_board[b[0]][b[1]] == 'bK':
            return 22
        elif b_board[b[0]][b[1]][0] == 'b':
            return 11
        else:
            return 0
    else:
        break
if f > 0:
    if b_board[b[0]][b[1]] == ' ':
        return 1
    elif b_board[b[0]][b[1]] == 'bK':
        return 22
    elif b_board[b[0]][b[1]][0] == 'b':
        return 11
if a[0] == b[0] or a[1] == b[1]:
    f=0
    c=[0,0,0,0]
    while a[1]!=b[1] or a[0]!=b[0]:
        if a[0]+(c[0]+1)<b[0] and b_board[a[0]+(c[0]+1)][a[1]]=='
' and b_board[a[0]+(c[0]+1)][a[1]][0]!='w' and (f==0 or f==1):
            c[0]+= 1
            f = 1
            elif a[0]+(c[1]-1)>b[0] and b_board[a[0]+(c[1]-
1)][a[1]]==' ' and b_board[a[0]+(c[1]-1)][a[1]][0]!='w' and (f==0 or f==2):
                f = 2
                c[1]-=1
                elif a[1]+(c[2]+1)<b[1] and
b_board[a[0]][a[1]+(c[2]+1)]==' ' and b_board[a[0]][a[1]+(c[2]+1)][0]!='w'
and (f==0 or f==3):
                    f = 3
                    c[2]+=1
                    elif a[1]+(c[3]-1)>b[1] and b_board[a[0]][a[1]+(c[3]-
1)]==' ' and b_board[a[0]][a[1]+(c[3]-1)][0]!='w' and (f==0 or f==4):
                        f = 4
                        c[3]-=1
            else:
                if f<=2 and (a[0]+(c[f-1]+1)==b[0] or a[0]+(c[f-1]-
1)==b[0]) and a[1]==b[1]:
                    print(1)
                    if b_board[b[0]][b[1]] == 'bK':
                        return 22
                    elif b_board[b[0]][b[1]][0] == 'b':
                        return 11
                    elif b_board[b[0]][b[1]]==' ':
                        return 1
                    elif f>2 and a[0]==b[0] and (a[1]+(c[f-1]+1)==b[1] or
a[1]+(c[f-1]-1) == b[1]):
                        print(2)
                        if b_board[b[0]][b[1]] == 'bK':
                            return 22
                        elif b_board[b[0]][b[1]][0] == 'w':
                            return 11
                        elif b_board[b[0]][b[1]] == ' ':
                            return 1
                    else:
                        if abs(a[0]-b[0])==1 or abs(a[1]-b[1])==1:
                            if b_board[b[0]][b[1]] == 'bK':

```

```

        return 22
    elif b_board[b[0]][b[1]][0] == 'b':
        return 11
    elif b_board[b[0]][b[1]] == ' ':
        return 1

    return 0
print(a,b,c,f)
return 0

elif m[1]=='K':
    d1=a[0]-b[0]
    d2=a[1]-b[1]
    if (abs(d1)<2 and abs(d2)<2) and (a[0]!=b[0] or a[1]!=b[1]):
        print('king w',1)
        if b_board[b[0]][b[1]]=='bK':
            return 22
        elif b_board[b[0]][b[1]][0]=='b':
            return 11
        elif b_board[b[0]][b[1]]==' ':
            return 1
    return 0
return 0

def Prob_to_win():
    print()
    print()
    if W[0] > B[0]:
        print('Looks like game winning is favourable to ' + s1[0])
    elif W[0] < B[0]:
        print('Looks like game winning is favourable to ' + s1[1])
    else:
        print('Looks like game id going to be tie')
    print()
    print()
    print()

print(s1[0] + ' turn (whites)')
while f:
    if f==1:
        for e in p.event.get():
            if e.type == p.QUIT:
                f = 0
            elif e.type == p.MOUSEBUTTONDOWN:
                loc = p.mouse.get_pos()
                y=loc[0]//64
                x=loc[1]//64

                if [x, y]!=prev and prev != [-1, -1] and
b_board[prev[0]][prev[1]][0]=='w':
                    v=validation(prev,[x, y])
                    if v==1:
                        t=b_board[x][y]
                        b_board[x][y]=b_board[prev[0]][prev[1]]
                        b_board[prev[0]][prev[1]]=t
                    print(b_board[x][y]+' made a safe move')

```

```

        prev = [-1, -1]
        f=2
        v=-1
        Prob_to_win()
        print(s1[1]+' turn (blacks)')
    elif [x, y] != prev and prev != [-1, -1] and v>=11:
        if v==11:
            print(b_board[prev[0]][prev[1]]+' killed'
'+b_board[x][y])
            b_board[x][y]=b_board[prev[0]][prev[1]]
            b_board[prev[0]][prev[1]]=' '
            f=2
            prev = [-1, -1]
            v=-1
            W[0]+=1
            Prob_to_win()
            print(s1[1]+' turn (blacks)')
        else:
            print(s1[0]+' wins')
            f=0
    elif v==0:
        prev=[-1,-1]
    elif [x, y]!=prev and prev == [-1, -1] and
b_board[x][y][0]=='w':
        print(s1[0]+' picked '+b_board[x][y])
        prev[0] = x
        prev[1] = y
    else:
        prev=[-1,-1]
        print('Wrong move')

chess_board()
p.display.flip()

if f==2:
    for e in p.event.get():
        if e.type == p.QUIT:
            f = 0
        elif e.type == p.MOUSEBUTTONDOWN:
            loc = p.mouse.get_pos()
            y=loc[0]//64
            x=loc[1]//64

            if [x, y]!=prev and prev != [-1, -1] and
b_board[prev[0]][prev[1]][0]=='b':
                v=validation(prev,[x, y])
                if v==1:
                    t=b_board[x][y]
                    b_board[x][y]=b_board[prev[0]][prev[1]]
                    b_board[prev[0]][prev[1]]=t
                    print(b_board[x][y]+' made a safe move')
                    prev = [-1, -1]
                    f=1
                    v=-1
                    Prob_to_win()
                    print(s1[0]+' turn (whites)')

                elif [x, y] != prev and prev != [-1, -1] and v >= 11:
                    if v == 11:

```

```

        print(b_board[prev[0]][prev[1]]+' killed
'+b_board[x][y])
        b_board[x][y] = b_board[prev[0]][prev[1]]
        b_board[prev[0]][prev[1]] = ' '
        f = 1
        prev = [-1, -1]
        v=-1
        B[0]+=1
        Prob_to_win()
        print(s1[0]+' turn (whites)')
    else:
        print(s1[1]+' wins')
        f = 0
    elif v==0:
        prev=[-1,-1]
    elif [x, y]!=prev and prev == [-1, -1] and
b_board[x][y][0]=='b':
        print(s1[1]+' picked '+b_board[x][y])
        prev[0] = x
        prev[1] = y
    else:
        prev=[-1,-1]
        print('wrong move')

chess_board()
p.display.flip()

```

PAPER ACCEPTANCE

CHESSE GAME

Mani Surya Teja Kota¹

¹ B.E. Computer science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India

ABSTRACT- "CHESSE GAME" is a game based project on Chess engine. Generally, chess is a two player game where each player strives to win on their individual sides. And making with user friendly interface is the 's main theme. And implementing the maximum rules for individual chess piece by mathematical way using co-ordinates concept in geometry.

Keywords: - Python Programming, Prediction, Problem Solving, Pygame, GUI.

1. INTRODUCTION:

The objective of this project is to create a own Chess engine which is user friendly and also to make user comfortable. And this project is a two player game. And prediction for game winner is to display for every move.

2. LITERATURE SURVEY:

Literature survey is that the most significant in code development method. Before developing the tool it's necessary to see the time issue, economy and company strength. Once these items are happy, then consecutive step is to see that OS and language may be used for developing the tool.

Once the programmers begin building the tool the programmers need ton of external support. This support are obtained from senior programmers, from book or from websites.

Before building the system the on the of thought unit taken into thought for developing the planned system. the main half of the project development sector considers and totally survey all the desired desires for developing the project. for each project Literature survey is that the most significant sector in code development method.

Before developing the tools and before the associated coming up with it's necessary to work out and survey the time issue, resource demand, man power, economy, and company strength.

Once these items area unit glad and fully surveyed, then consecutive step is to see concerning the software package specifications within the individual system like what form of software system the project would need, and what area unit all the required software package area unit required to proceed with consecutive step like developing the tools, and also the associated operations.

2.1 SCOPE OF THE PROJECT:

The scope of the project is as follows:

Scope of this project is to make a own chess engine which is customizable like to implement the rules like castling in the game. And to make game is for entertainment purpose. To provide a friendly environment where user can improve game skills with more interest.

2.2 EXISTING SYSTEM:

In the present scenario, python has its module called chess which can do all the operations on the chess moves. But graphically its not. And it is the most used module for the chess game and for chess study purpose.

2.3 PROPOSED SYSTEM:

This system is proposed with certain features which makes user to feel comfortable.

The proposed system is a engine which detect whether the present move is right or wrong.

Checks the each and every move is going to the part of winning prediction of the player.

Notifications like player moves taken opponent's chess piece is and displayed.

Illegal moves are not considered and given to make a perfect move.

Understandable code for everyone using co-ordinates geometry concept.



Fig 1: Proposed System Figure 1

ADVANTAGES OF PROPOSED SYSTEM:

- User friendly interface.
- Illegal move filtration.
- Winner Prediction for every step.
- Understandable code for someone who wants to study it.
- Winner declaration.

3. SYSTEM ARCHITECTURE:

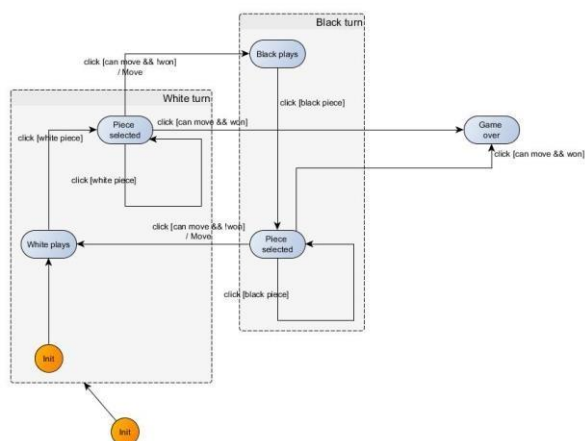


Fig 2: System Architecture

4. MODULE DESCRIPTION:

Project Modules:

- Game Intro Module
- User Input module
- Step Validation module
- Game Updation module
- Winner deaclaration module

I. Game Intro Module:

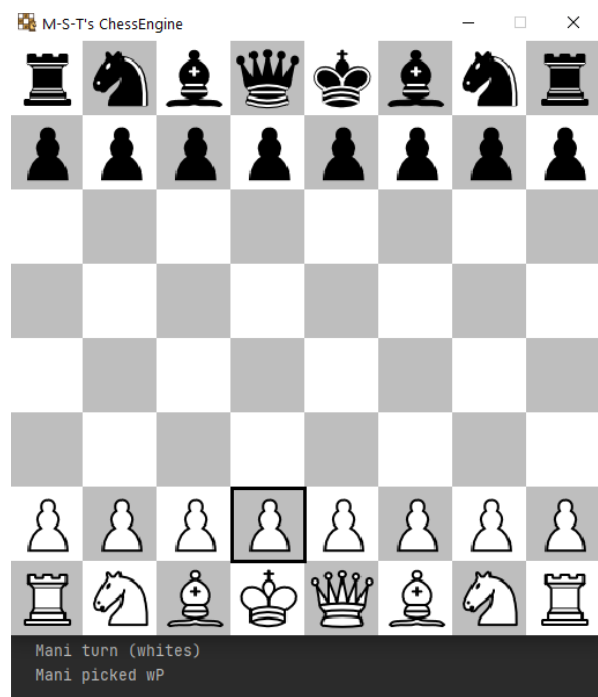
It is important to know the players information to specify them during the game. So in this module players information is taken and saved for future purpose. And also to change the chances of players to play.

```

"C:\Users\KOTA SRI SURYA TEJA\PycharmProjects\pythonProject\venv\Scripts\python.exe"
pygame 2.0.1 (SDL 2.0.14, Python 3.8.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
ENTER YOUR NAME PLAYER 1 Mani
ENTER YOUR NAME PLAYER 2 Surya
Mani turn (whites)
|
  
```

II. Game Intro Module:

User Input module is used to ensure the event of every mouse is not for the chess piece movement. And it captures the coordinates whenever it selects on the chess board and sends the coordinates for the validation, to check whether it's a legal move or not. And this project is made with window size of (512 px,512 px).



III. Step Validation module:

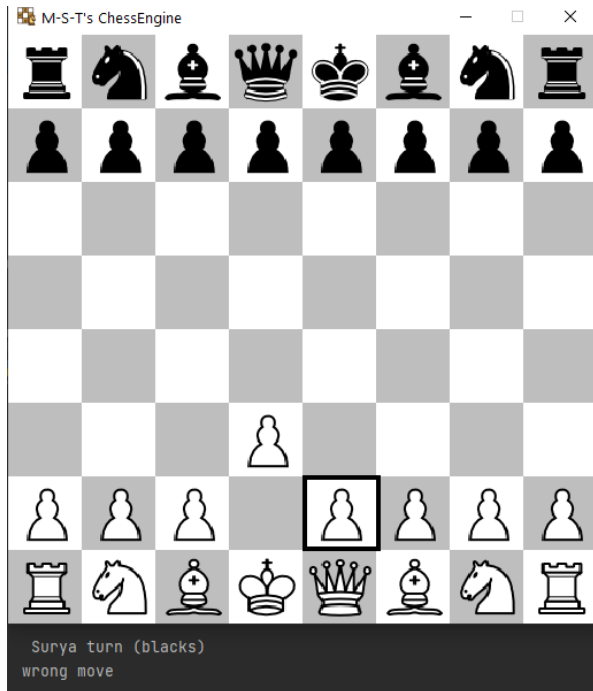
This module provides signal whether the move is

valid/legal or not. And in this module there are

multiple signals given for the input move. There are totally 4 types of signals / flags . They are :-

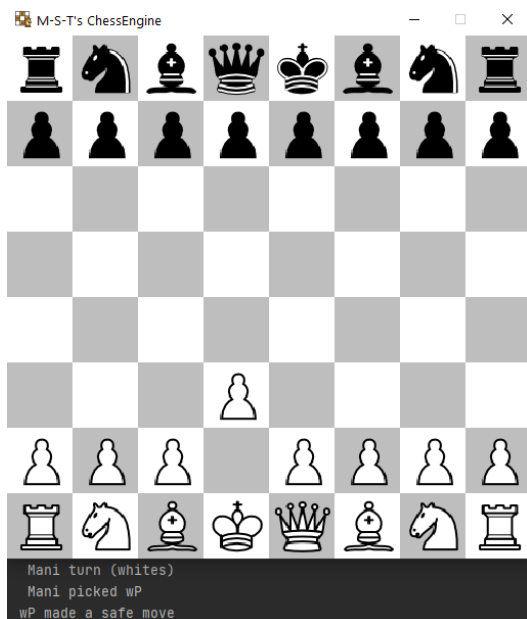
5. Signal 0:

This signal is for to indicate the input move / step is not valid / legal. And also if the player unwantedly selects the opponent's chess piece it generates a signal 0 and displays the message that "wrong move".

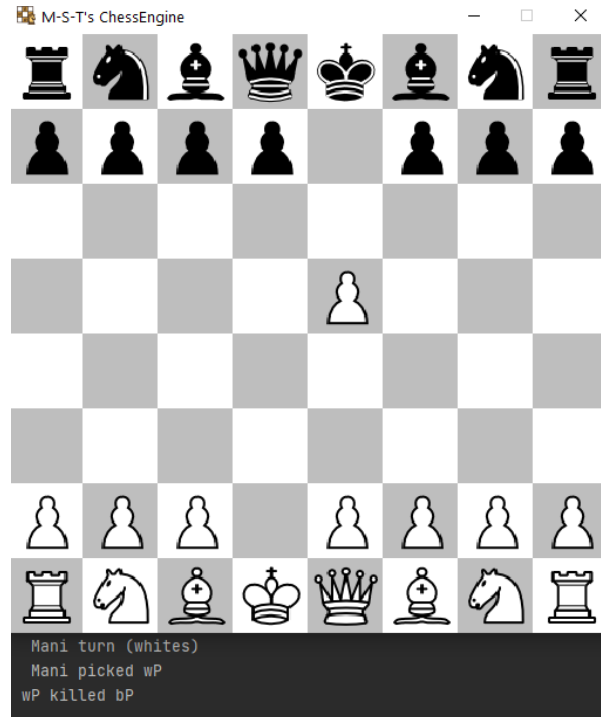


6. Signal 1:

This signal is for the chess piece whether it can move to its destination with no interaction of any chess piece in its way. If it can made the destination, then signal is generated and message is generated that it made a safe move.



This signal is for chess piece whether the player1 piece captured or taken the opponents piece. If this condition is true then system generates a message like "[player1 piece] killed [player2 piece]".



8. Signal 22:

This signal is for chess piece whether the player1 piece is perfectly captured the opponent's king. If it

7. Signal 11:

is done it returns a signal 22. Finally winner declaration will be done and game will be over.

And after completion of checking these validations it goes to the chess board for the next step updation of the chess



board. So that it can move chess piece in the code and display over the window.

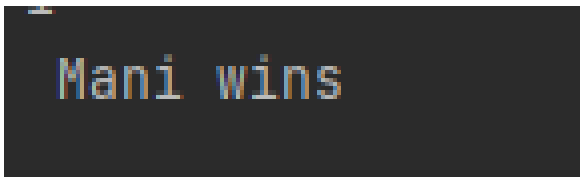
IV. Game Updation Module:

This module is used to update the chess board which is used to display current state of board for every move. There will be done two things.

Either to swap the piece
with square.
Or two replace the
opponents piece with the players piece.

V. Winner declaration module:

This module is used to fetch the game all the time to check whether the opponent's king is been captured or not. If yes, then game is closed and displays the message like "[winning player's name] win"



5. RESULTS:

By checking all the possible cases to run the chess engine. This project running successfully without any flaws. And also can be done in advanced way.

6. CONCLUSION:

is a chess game with more benefits. Firstly, its more benefitable for the ones who wants and tries to study the code and logic behind the code. And its GUI programming of the pygame is more flexible than the "Ktinker module" for gaming interface.

REFERENCES:

- [1] Co-ordinates concept in geometry.
- [2] Mathematical logics on co-ordinates.
- [3] Pygame module tutorials on youtube

<https://youtu.be/FfWpgLFMI7w> (link)