My Development Journey of a Chess Game

A Detailed Report on Creating a Functional Chess Game Using Python and Pygame

LinkedIn - https://www.linkedin.com/in/haricharann-dv/

GitHub - https://github.com/Haricharann-DV/ChessCode_Python

Hi Reader,

My name is Haricharann D V, and I'm excited to share my journey in developing a fully functional Chess game.

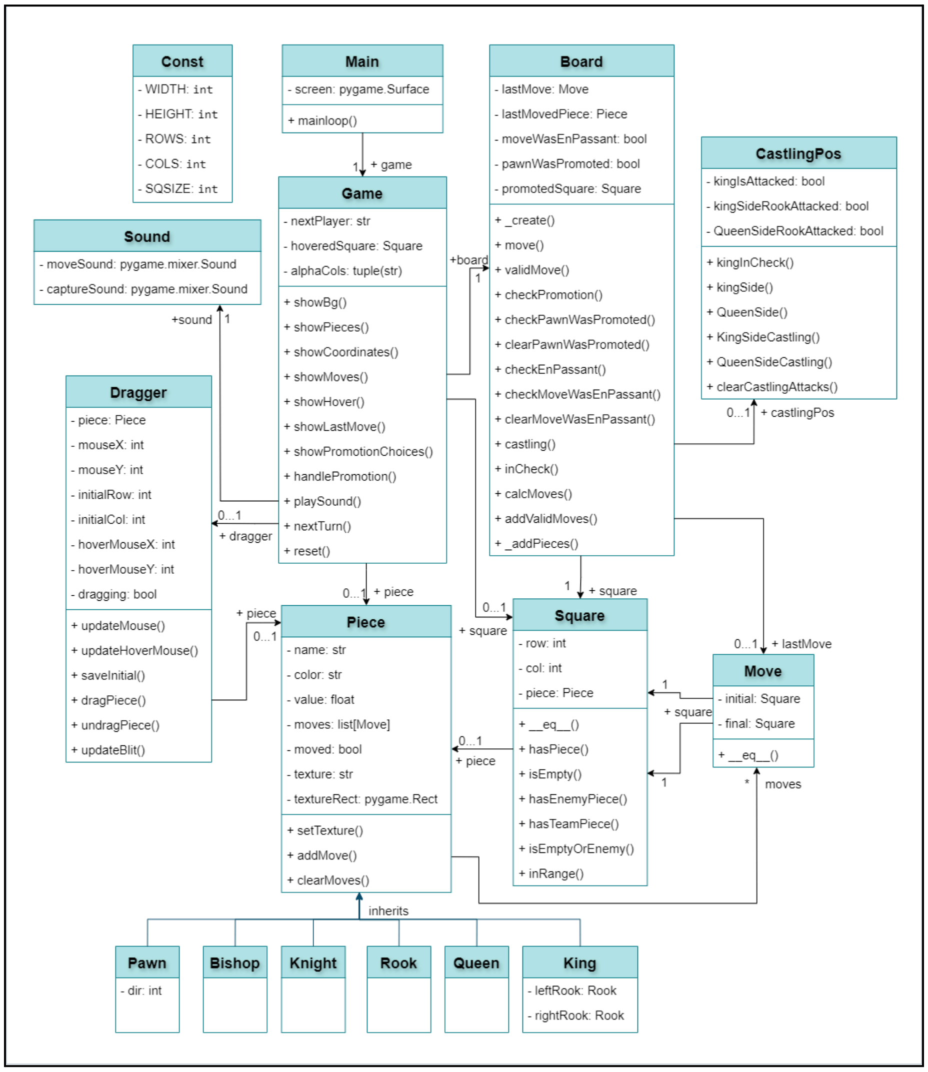
I began learning Python programming to enhance my skills and stay aligned with the rapidly evolving AI and IT industry. I believed that building a project would deepen my understanding of these technologies. Given my love for chess, I thought it would be cool to create a chess game. I developed this project by utilizing various resources such as GeeksforGeeks (GFG), YouTube tutorials, and prompt engineering with AI, all while leveraging Python and Pygame.

Throughout this project, I adhered to SOLID principles to ensure a robust and maintainable codebase.

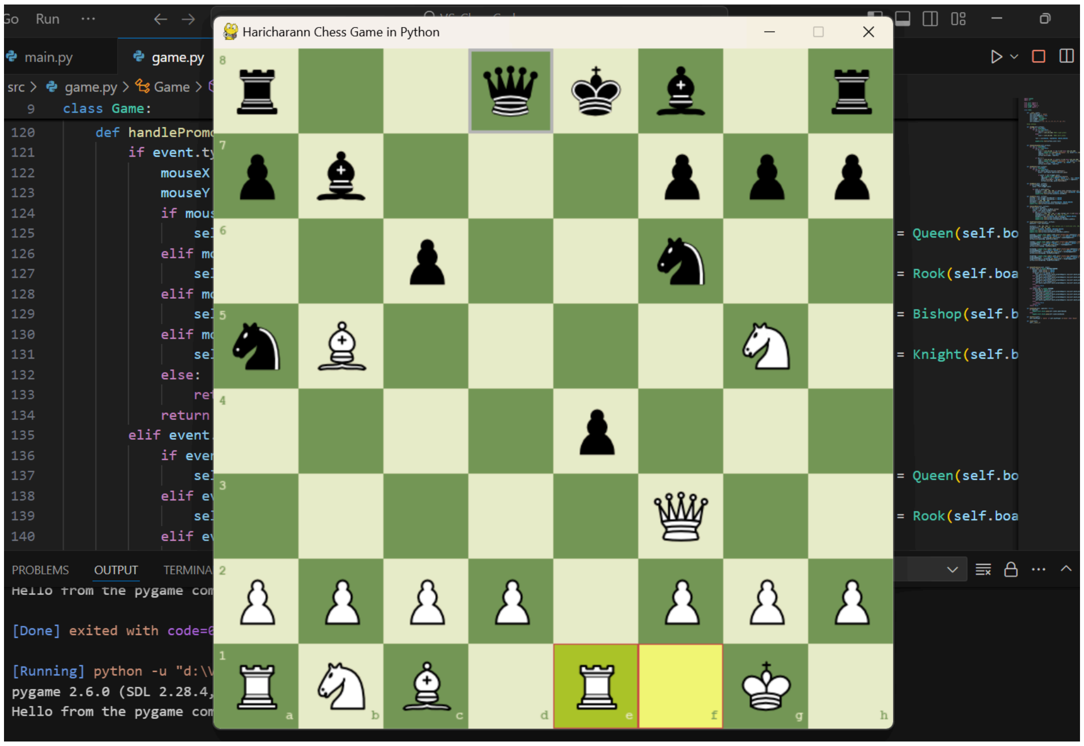
Documenting this journey not only allows me to reflect on my growth but also to share my experiences with fellow developers and chess enthusiasts. I hope you find this documentation informative.

Class Diagram

Here's an overview of the structure and relationships between different classes in the project.



Snapshot of the Final Product



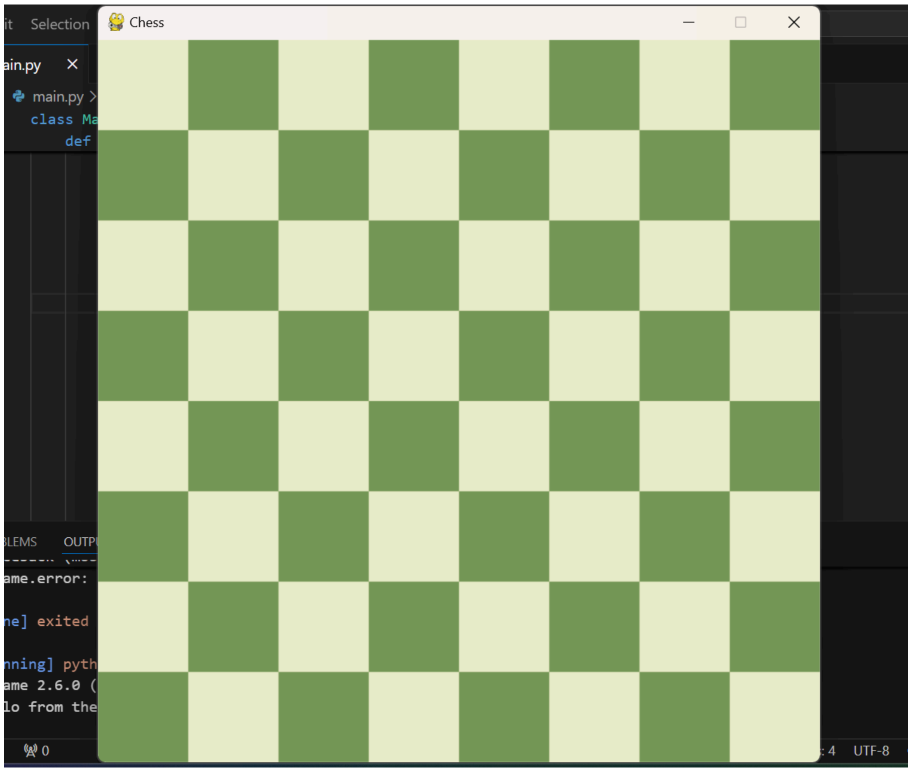
```
class Game:
    def handlePromo
        if event.ty
            mouseX
            mouseY
            if mous
                sel
            elif mo
                sel
            elif mo
                sel
            elif mo
                sel
            else:
                ret
            return
        elif event.
            if even
                sel
            elif ev
                sel
            elif ev
```

```
= Queen(self.bo
= Rook(self.boa
= Bishop(self.b
= Knight(self.b
= Queen(self.bo
= Rook(self.boa
```

```
Hello from the pygame com

[Done] exited with code=0

[Running] python -u "d:\\
pygame 2.6.0 (SDL 2.28.4,
Hello from the pygame com
```

Snapshots of progress

DAY 1



```
class Ma
    def
```

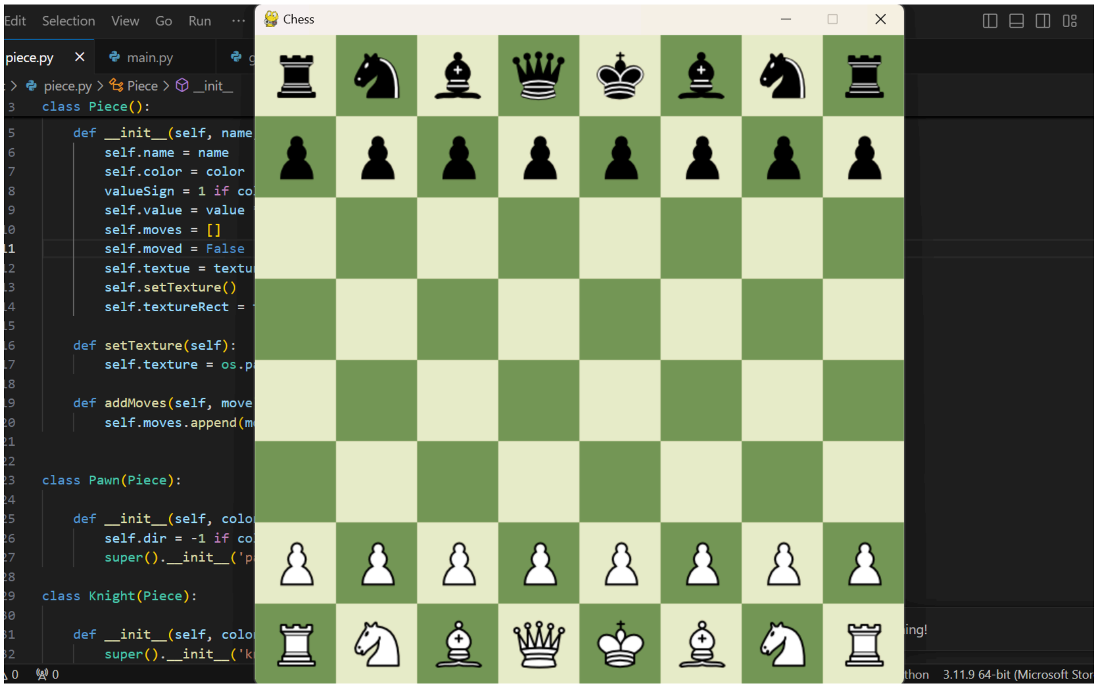
```
ame.error:

ne] exited

nning] pyth
ame 2.6.0 (
lo from the
```

I successfully created the chessboard structure using Pygame. The next step is to develop the pieces.

DAY 2

Encountered challenges in defining the pieces. There are multiple approaches, including defining the starting positions directly. However, I aim to decouple the code and reduce redundancy. Continuing to research optimal solutions.

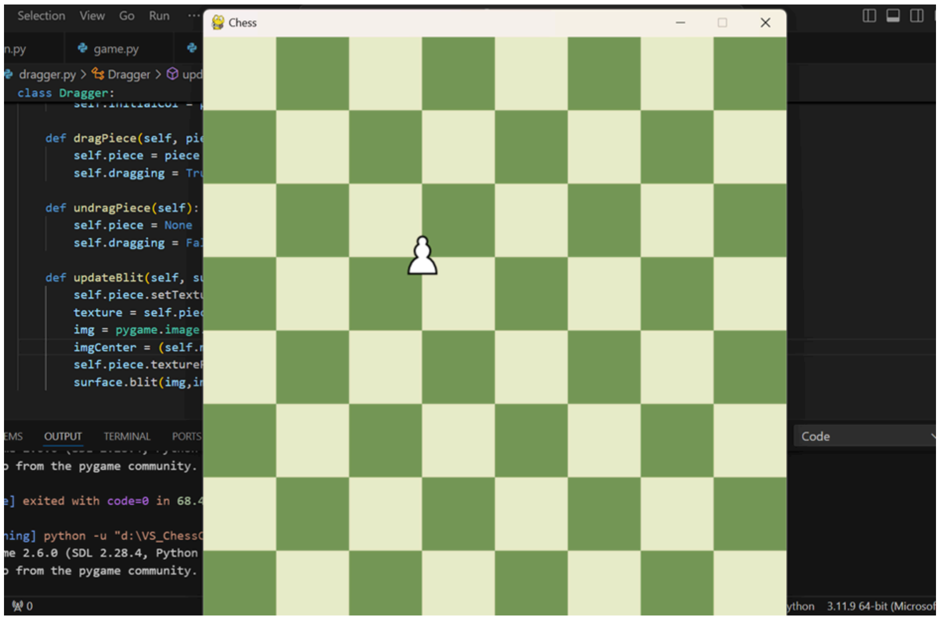
DAY 3



```
class Piece():

    def __init__(self, name
        self.name = name
        self.color = color
        valueSign = 1 if co
        self.value = value
        self.moves = []
        self.moved = False
        self.textue = textu
        self.setTexture()
        self.textureRect =

    def setTexture(self):
        self.texture = os.p

    def addMoves(self, move
        self.moves.append(m


class Pawn(Piece):

    def __init__(self, colo
        self.dir = -1 if co
        super().__init__('p

class Knight(Piece):

    def __init__(self, colo
        super().__init__('k
```

Achieved significant progress. Expanded the classes, defined the pieces correctly, and centered them on the board.

DAY 4

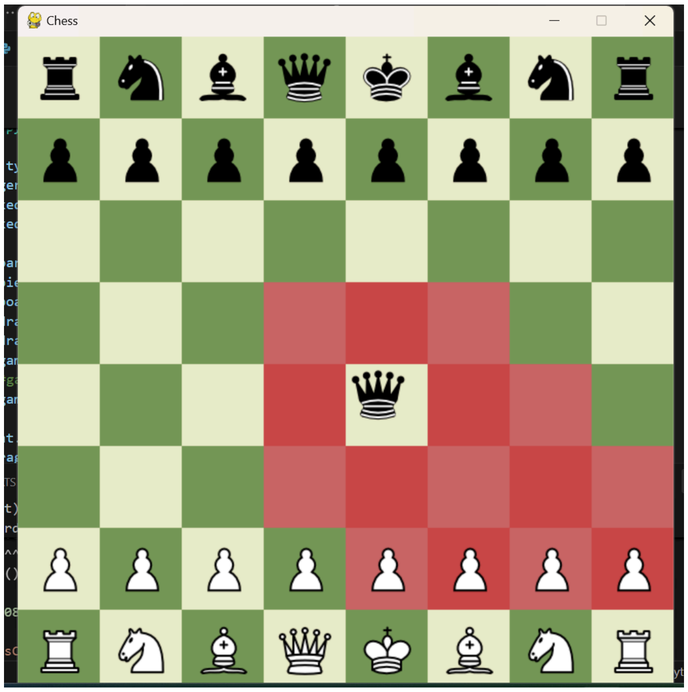


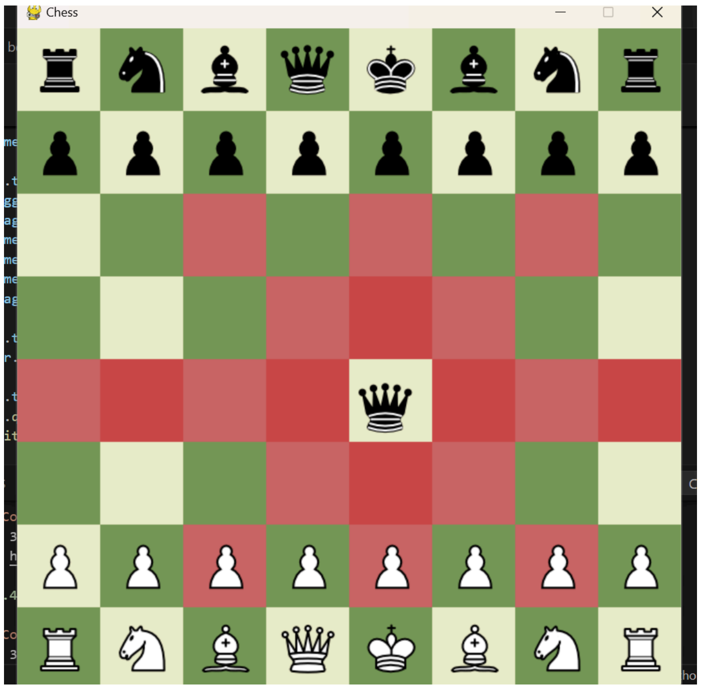
```
class Dragger:

    def dragPiece(self, pi
        self.piece = piece
        self.dragging = Tr

    def undragPiece(self):
        self.piece = None
        self.dragging = Fa

    def updateBlit(self, su
        self.piece.setTextu
        texture = self.piec
        img = pygame.image
        imgCenter = (self.
        self.piece.textureR
        surface.blit(img,i
```

```
o from the pygame community.

e] exited with code=0 in 68.4

ning] python -u "d:\VS_Chess
me 2.6.0 (SDL 2.28.4, Python
o from the pygame community.
```

Implemented **Dragger** class with appropriate dragging methods. Initially faced issues where other pieces would disappear when dragging a piece, but these bugs have been resolved. The next step is to calculate valid moves.

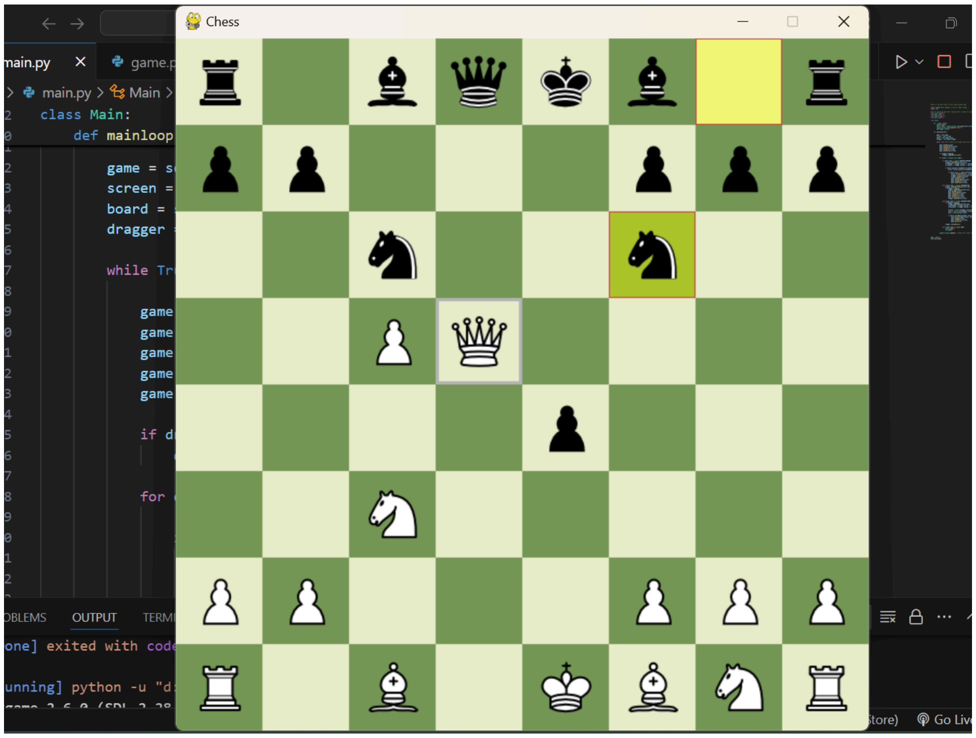
DAY 5,6





Encountered initial challenges but ultimately succeeded. The calculation of valid moves for all the pieces has been coded(exception - check calculation, pawn promotion, en passant, castling). The next task is to enable piece movement on the board.

DAY 7,8

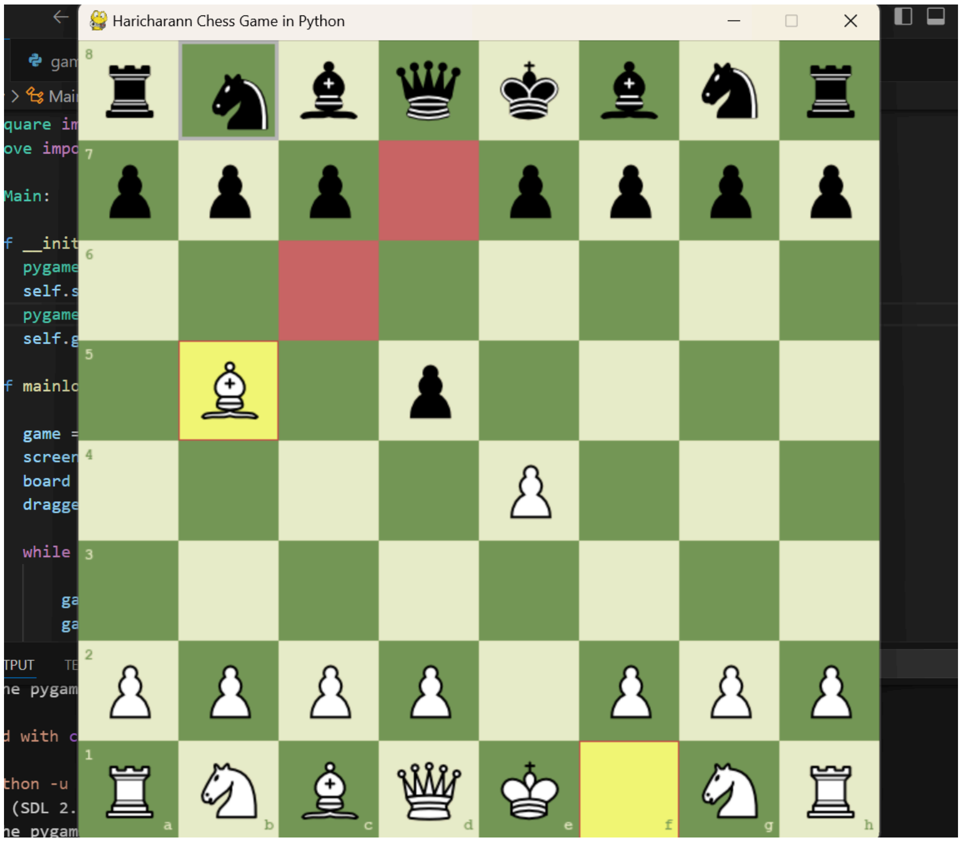


```
class Main:
    def mainloop
        game = s
        screen =
        board =
        dragger

        while Tr

            game
            game
            game
            game
            game

            if d

            for
```

```
one] exited with code
unning] python -u "d
```

Enabled piece movement on the board with proper validations, excluding features like en passant, promotion, castling, and check detection. Added designs to show the last move, implemented hover effect and created a reset feature activated by pressing 'r' on the keyboard.

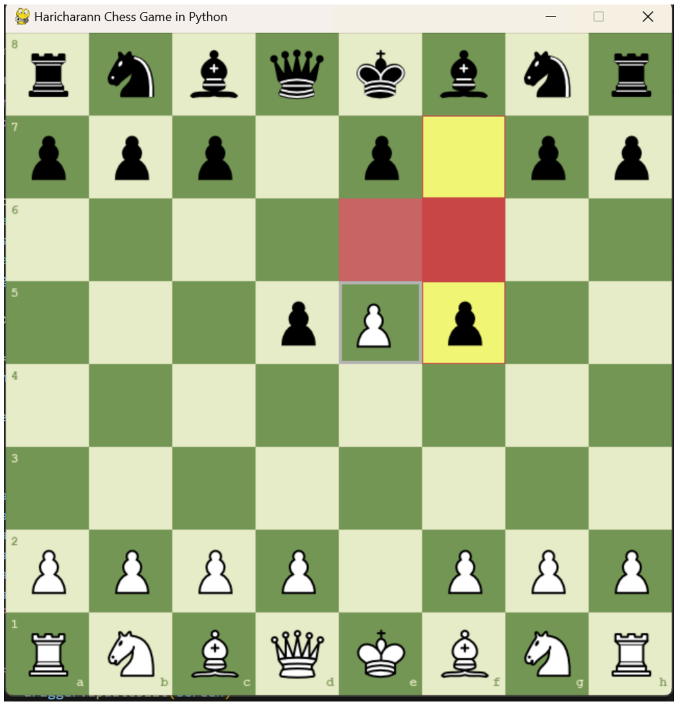
DAY 9

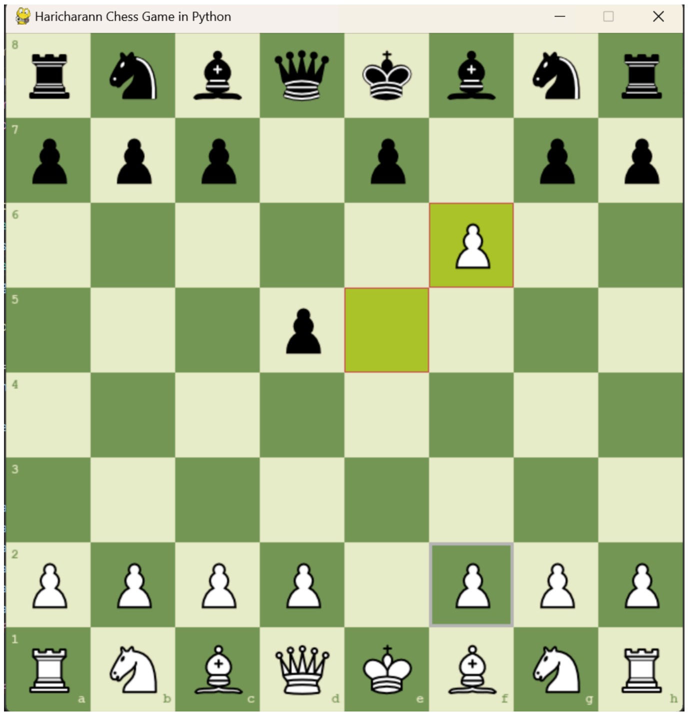


```
gam
Mai
quare im
ove impo

Main:

f __init
  pygame
  self.s
  pygame
  self.g

f mainlo

  game =
  screen
  board
  dragge

  while

      ga
      ga
TPUT    TE
he pygam

d with c

thon -u
 (SDL 2.
he pygam
```

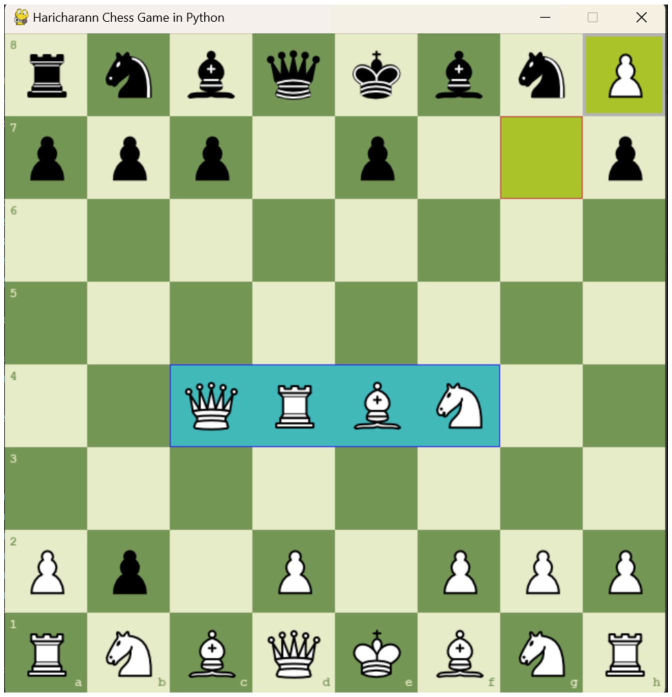
Implemented check detection functionality.

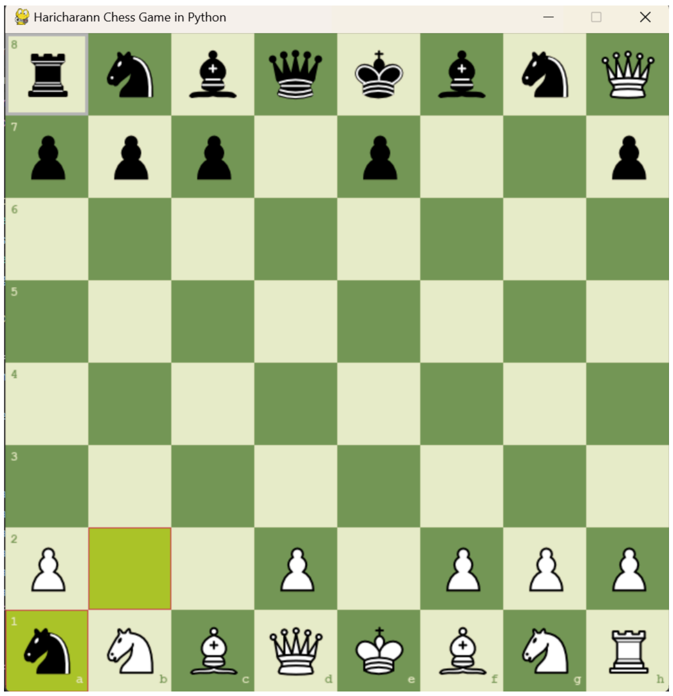
DAY 10





(Note - The last move was f5 so en-passant is possible for e-pawn, exf6 is a valid move)

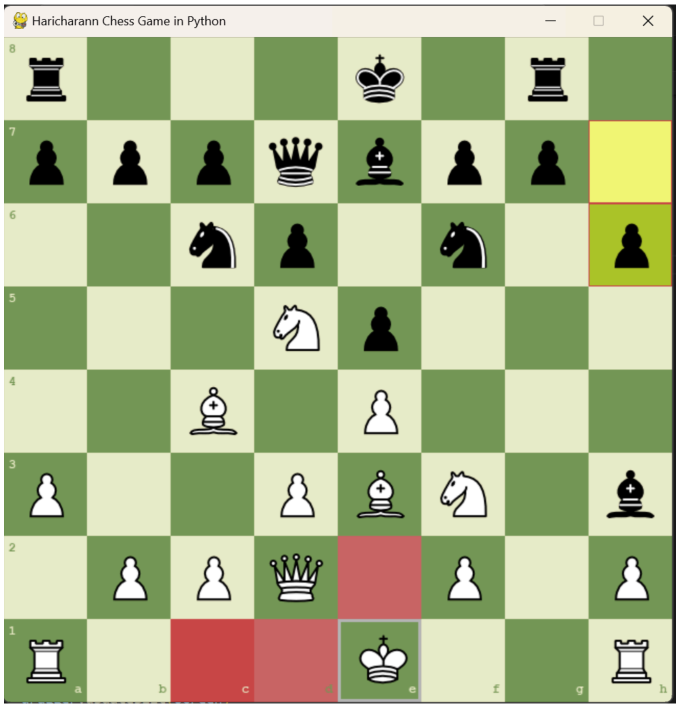


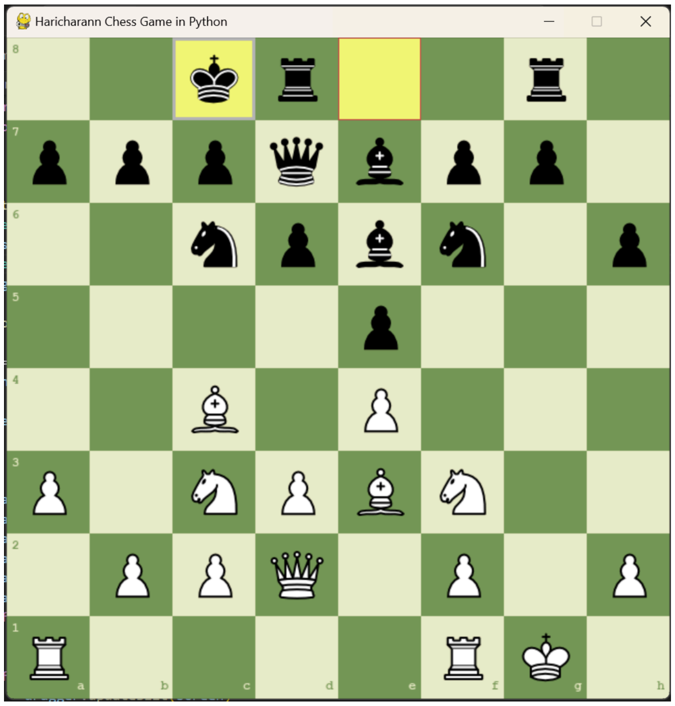


(Note - Pieces can be selected by clicking on the shown options or by pressing 1,2,3,4 in the keyboard for the same order)

Added en passant and pawn promotion features.

DAY 11





(Note - When in check/ if king or rook has moved/ castled squares are attacked - castling is invalid)

Completed the implementation of castling and integrated sound effects, significantly enhancing the game's functionality and user experience.

DAY 12

Performed final bug testing and adjustments. Pushed the project to GitHub for version control and community sharing.

Summary

This documentation outlines my experience in developing a fully functional chess game using **Python, Pygame, prompt engineering, and Git.** Driven by my passion for chess and a desire to enhance my programming skills, I embarked on this project and navigated various challenges along the way. These challenges included implementing complex piece movements and debugging visual issues. Through extensive research, experimentation with different approaches, and iterative feedback, I successfully overcame these obstacles and delivered a robust chess game.