Chessboard:

There are 64 squares. Each square is referred to by its file first, depending on its horizontal position, and then rank depending its vertical position.

files-> a b c d e f g h

rank-> 1 2 3 4 5 6 7 8

1 a1 b1 c1 d1 e1 f1 g1 h1

2 a2 b2 c2 d2 e2 f2 g2 h2

3 a3 b3 c3 d3 e3 f3 g3 h3

4 a4 b4 c4 d4 e4 f4 g4 h4

5 a5 b5 c5 d5 e5 f5 g5 h5

6 a6 b6 c6 d6 e6 f6 g6 h6

7 a7 b7 c7 d7 e7 f7 g7 h7

8 a8 b8 c8 d8 e8 f8 g8 h8

a b c d e f g h

Chess Pieces:

Chess pieces are defined in a constant object that mimics an enum. Each piece is a property of the “Piece” variable and each piece also has its own properties that store its respective name, icon, color, abbreviation and value. There is also a 2-dimensional array that stores a single instance of each piece so that they can be dynamically referenced by other functions.

Pieces on the board are stored in a map using the id of the square they are located on as their key. This is for ease of access when looking for pieces that are currently active. The function, getPieceFromAbbr() is used to generically get the other properties for pieces from the movement log when undoing previous moves or when promoting pawns. This is because pieces are referred to by their color and abbreviation in the code which made accessing a piece’s icon/image from those values a necessity.

Generating a Move:

There is a separate function for each piece that generates their movement options irrespective of color. None of these functions are called directly, however. Instead, a function containing a switch statement is used to call these functions based on the parameters passed into it. The reason why separate functions are used is because each piece moves in a different way, and this was the simplest way to implement those differences.

Each of these functions first parse out the pieces current position. This is so the code can increment or decrement over the piece’s position to find the valid moves available to it. If the move is valid, it will continue to move in that direction, if able, until it either reaches the end of the board or is obstructed by another piece. To determine a move is valid, a piece will check if the movement will put or keep the turn player in a check state. If it does, then the move will be rejected. All other movements, if any, are kept and displayed by highlighting the squares matching the positions generated.

Game State:

Game states are determined from the available move options. If an opposing piece is targeting the King, then the game will recognize that the turn player is in check. In either