From Wilhelm

- Josh Arnold
- Wilhelm Gonce
- Summary of the program: The program is a chess game where you need to create a user and password to log in and then you can create new games of chess and play with an opponent. There is also the possibility for other players to spectate the game.
- Feedback on the running program:
    - was it obvious how you were supposed to work with it? Explain.

In the beginning it is obvious that you need to create an account, but when you log in there are the checkers and connect four buttons that are not yet implemented so maybe hide them until they are supported to not confuse the user. After the game page is intuitive that you can join games and watch them. Finally, the board page looks good but changing the pieces from words to images and flipping the board so each player sees it from their perspective can make it more understandable.

- was it obvious how to exit the program? Explain.

After you make a checkmate there is no return to lobby button so to make it more obvious how to exit the game, adding a button to go back could be better.

- did you ever feel lost or out of control at any point? Explain.

I didn't feel lost in general but there are some scenarios where the game doesn't behave as it should.

- was it attractive, pleasing to look it? Share examples.

It is attractive and pleasing to look and includes several nice features that allow you to interact with several parts of the UI, but changing the pieces to their actual images, changing the color of the selected pieces from purple to a brighter color and changing the checkmate text because it overlaps with the board, and it is hard to read could make the game better looking.

- did it ever crash while you were using it? If so, share steps to reproduce the crash.

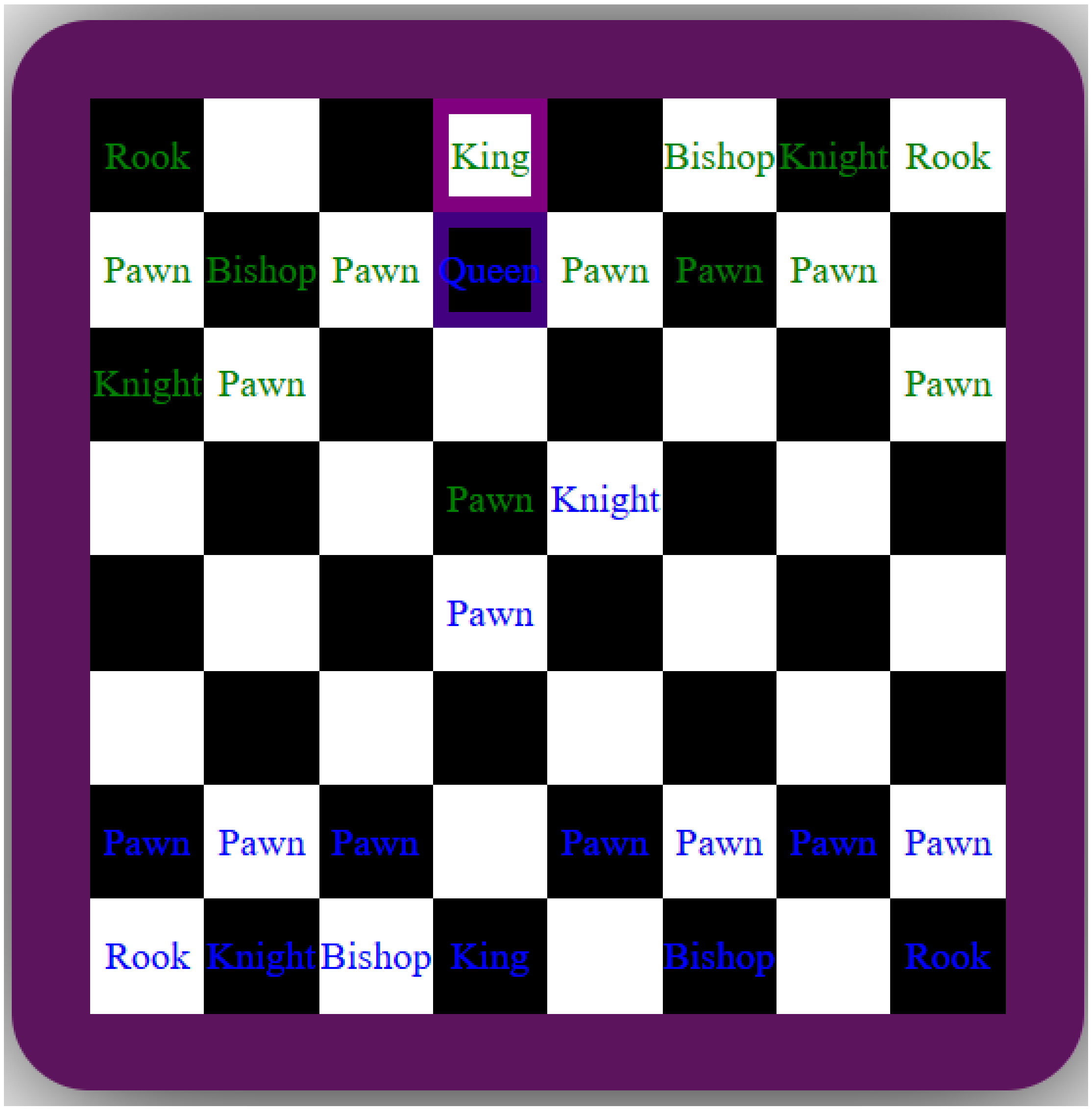
First It crashed when I tried to click the checkers and connect four buttons because they were not implemented. Also, in the beginning I struggled to run the game because on the Chess file on line 1038 the Filepath was set to test and the program couldn't run the file. Maybe you can set the constructor for the game to receive a parameter to decide whether it should use the blazor filepath of the test filepath and not change it manually.

- did you notice any flaws? If so, share where & how you found them.

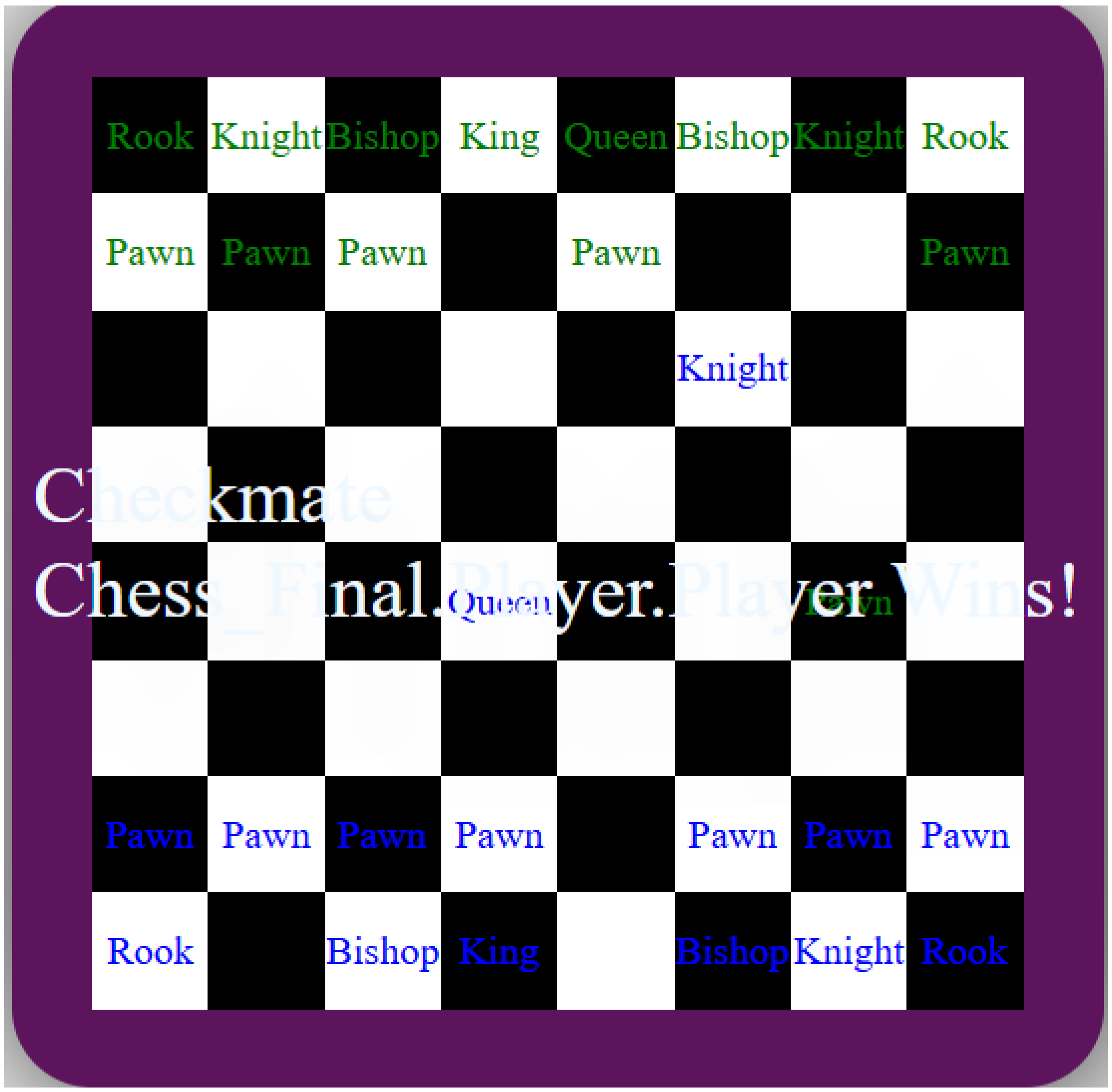
Overall, the game has the basic rules of chess, but it is missing some features like castling, promoting, handling illegal moves, stalemate and draw options and has some unintended behavior related to checks and checkmate.

- were you able to make it crash or perform in an unintended way? If so, share steps to reproduce the issue.

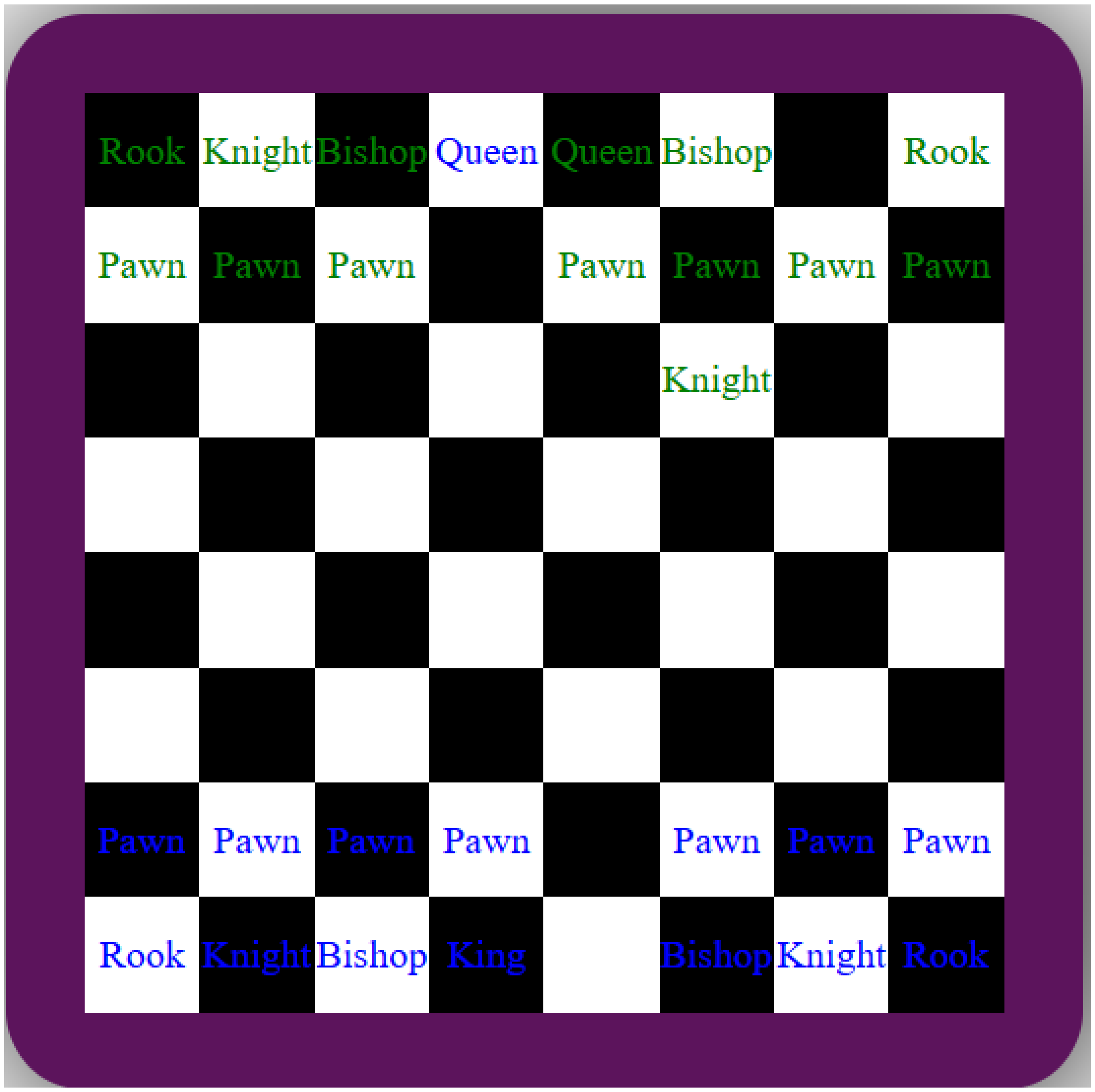
I found 3 ways to make unintended moves, the first one involves having the king in a checkmate where the checking piece is adjacent to the king, but it is being protected by another piece. In this case the game allows you to capture that piece with the king and not count the checkmate. You might want to add a method that checks if a piece is protected by the other pieces of the same color, so the king is not allowed to capture them.


Rook King Bishop Knight Rook
Pawn Bishop Pawn Queen Pawn Pawn Pawn
Knight Pawn Pawn
Pawn Knight
Pawn
Pawn Pawn Pawn Pawn Pawn Pawn Pawn
Rook Knight Bishop King Bishop Rook

Another unintended behavior occurs in this position where the king is in check, but both the queen and bishop can stop the check, but the game takes it as a checkmate. For this one you can set a method that calculates the path between the king and the checking piece if it is either queen, rook or bishop and then make the piece movement valid if they can move inside that path.



Finally, if you put the king in a position where a piece of the same color is in front and then another piece that can check is in the same line and then you move that piece between the king and the piece of the opposite color the game doesn't detect a check and allows you to capture the king. To solve this, you might try to change the algorithm to check for a check, so it evaluates all the pieces of the opposite color to see if they are in range to attack the king or not.



- Feedback on the code:
    - Did the variable names make sense?

Overall, most of the variable names made sense and reflected what they were storing.

- Did the class and/or method names make sense?

The names of the classes and the methods also made sense having short names and describing what they do.

- Was the vertical whitespace consistent?

The vertical whitespace was mostly consistent through all the files.

- Was the indentation consistent?

The indentation was also correct.

- Were any of the methods unreasonably long?

The methods weren’t really long except for the CalculateValidMoves methods for the pieces. They are long and can be reduced by including a loop that accounts for all the directions a piece can move, and inside of that have the loop for going through that specific path.

- Could you read the code and easily follow along, understanding what it was supposed to do, or did you get lost and confused?

The code was somewhat understandable but having all the classes for the pieces and the game in the same file can be confusing and overwhelming so having each one on their own file can lead to better organization, this includes the generics file as well.

- What changes would you suggest to method names?

I really wouldn’t change the method names, but you can include a comment explaining what they do for more clarity.

- What changes would you suggest to variable names?

The same as with methods. The variable names are fine but a description in a comment wouldn’t hurt.

- What changes would you suggest to method signatures (e.g. changing the return type or parameters)?

The method signatures are good, and I don’t think they need changes.