WillowBunch Design Documentation

Requirements:

Make an adapted version of online chess with specific changes. Changes include new pieces, different movesets for the pieces, and other customizability.

Implementation:

Brainstorming:
We started off by reading the problem and making notes of what we can do, and initial ideas of how to implement it. We then met as a group and discussed our ideas, along with brainstorming other ideas. We assigned the tasks into what each team member wanted to do, with Manjyot doing server-client connection, Owen doing piece logic, and Ayden working on GUI.

We chose to implement our solution using java, as that is what all members of the team are most familiar with. The backend, networking, and GUI are all written in Java.

Work Division:
We broke the problem into three categories: server-client connection, piece logic and backend, and the GUI. Initially Owen was responsible for implementing the logic of piece movement, Ayden was responsible for implementing the gameboard and user interface and Manjyot was responsible for implementing the connection between two players.

Server-Client Connection:
For this, we used the Java server and client socket imports to allow for connection between two players. One player will choose to host the game, and they will act as the ‘server’ for that game. The second player will join the hosted game, and act as the ‘client’ in doing so. They maintain the connection throughout the session, using the input and output streams to communicate moves.

Piece Logic:
Movement logic was implemented by having a move method in the piece class. Each piece has its own version of this method, that returns a boolean value. It returns true if the piece can move and false if it can not. The method takes into account the current position of the piece and the position it is attempting to move to and determines if the movement is valid based on the rules described in the “rules” section of the prompt. A piece can not move to a square that has another piece of the same colour, however if a piece moves to a square that is occupied by a piece of the opposite colour, then the piece that was there is removed from the game. If the king is taken the colour that loses their king loses the game.

GUI:
For the GUI, Java Swing was chosen because we had the most experience with it. The visual board is implemented as a custom JPanel that can easily track mouse clicks and button presses. The GUI has a toolbar at the top containing a reset button and a spot for messages. The message bar is used to tell each player when it's their turn, as well as if the game has been won.

Known Bugs:
In terms of GUI there is a small visual bug where the bottom row of the checkerboard is longer than all the others, sometimes duplicating the bottom of the white pieces. This does not affect the gameplay and goes away when the window is resized.

Another known bug can be seen when one player leaves in the middle of a game without making their move, leaving the other to endlessly wait, without a prompt showing that they are the only one left.

There are times when the pawn and vanguard pieces do not work as expected-Pawn and Vanguard inconsistencies, queen cooldown not implemented

Future Considerations:
Along with fixing the known bugs, we would add a variety of changes to our program. We would start off by implementing the other features mentioned in the prompt. This includes the ability to leave and pick up where you left off. We also want to add the ability to change the board size, and new game modes as well. New features would be the ability to highlight possible moves, sending emotes, and a timer.