CHESS

This project was originally proposed by a former CS320 student: Marie Kiley

Project Summary

The goal of this project is to make a client for playing chess asynchronously with other players on the same service. The project should also include features that help facilitate multiple forms of chess, such as a user-to-user messaging component.

Basic Requirements

Remember this is a web browser front end based application, so it must have a log in. To make this chess application work, it must know the rules of chess, have persistence of game states between logins, and display information in an intuitive manner. These chess games should be independent from each other so that players can have multiple games of chess at the same time.

A Working Chess Game

This project must allow users to play chess. This includes displaying the board, prompting the user for input, validating that input, and executing the moves described by that input. This includes any valid move in a game of chess, including Castling and En Passant Captures. It is important that users can see the moves already taken in the game and know how the game state got to where it is. The best way to view previous moves is algebraic notation. Users will want to see their previous games as well. This application needs to declare a winner if checkmate is reached and provides options for surrenders and draws. Some players will want to show off their games to others, so it is important to save these and make them available to view.

A System of Matchmaking

This project must match players together so that they have opponents to compete against. The most basic form of matchmaking is pairing up two users regardless of skill level and with no restrictions on how long the user has to take their turns. Matchmaking for users based on skill level requires a ranking system. Use the ELO ranking system. Timed games of chess should make it so the user has a set amount of time since the last move to make their move, or else they lose the game. This amount of time must be agreed upon by both players and specific to the game and not a set amount of time for all games. Unranked untimed games, unranked timed games, ranked untimed games, and ranked timed games should all be possible. Some users will want to avoid a matchmaking system altogether and just play with friends, therefore users will need to be able to challenge other users to chess games.

A Messenger Component

This project must allow users to communicate with each other. This messaging system will sometimes be abused, so unruly individuals should be reported to moderators. Moderators will be appointed by administrators. Moderators and administrators need to be able to punish users that are inappropriate. If there is a message in a user's inbox that they would like to reply to, let them do so without having to leave that message. Users should be able to remove messages from their inbox or outbox.

User Interface

Consider how the board is displayed: you may start out with a text-based display, but you MUST implement a spring-based display for your final project presentation. Consider how the user gives input: at first by text commands. For the final project presentation, the user MUST be able to drag and drop their pieces via the mouse. You should also consider how/where to indicate pieces that have been captured and removed from the board, as well as replacement of pawns with chosen pieces when a pawn reaches its opponent's end of the board.