SE Capstone Fall 2021 Group 2

Chess Webapp

Problem it solves

Users would like to be able to engage in chess matches in a modern web browser.

MVP User Acceptance Criteria

- Users will access the game via a web interface, compatible with all modern browsers
- Upon accessing the application, Users will find a simple landing page with an overview of what to expect from the application and a call to action to sign up/sign in.
- Multiple Users should be able to access our application at once
- A User should be able to sign up with our application as a unique entity
- A User should be able to log into our application as a unique entity
- A User should be able to start a game of chess against an AI algorithm
- A User should be able to close the browser window or browser, reopen it, and expect to resume the chess game
- User should be able to log out, log back in, and expect to resume their chess game
- Application will update game state in real-time or near real time (no need to refresh the browser window to see changes to game state)
- User will interact with the chess pieces via a interactive drag and drop UI
- User will see opponent's moves in UI
- User will be notified via the UI in the browser when it's their turn.
- User will be prohibited from moving their game pieces when it's not their turn
- Player movements will be constrained to the rules of chess
- Invalid movements will return the game piece to its orignal location, provide an indication of the prohibited movement, and allow the User to try the move again.
- Upon completion of the game, the User will be presented with state of win, loss, or stalemate and given the option to play again.
- The application should keep a game log in algebraic notation and the User should be able to export the log as a .PGN file to be used with other chess software.

Stage 2..n Features (Stretch Goals)

- A User may engage in multiple simultaneous games of chess independent of one another
- A User may select from one of several AI alogrithms to play against
- A User may play against another User
- A User can view their game history
- Hovering a game piece in the UI will show potential paths of movement
- Users should have the option to play timed matches, where running out of time counts as a loss.
- "Mini game" where the board is set up so that checkmake is possible in X number of moves. If User does not checkmate in X moves the board is reset to original position, if User wins they can choose to play again.

Implementation

High Level

- Authentication will be provided by a free tier of a third-party solution, vendor TBD
- UI will be written as a thin-client SPA using Javascript in the browser
- Business logic will be written in Java and run on a persistent server in the cloud, vendor TBD
- Game state may be stored in memory or in a database, depending on available time. MVP functionality will see it stored in memory.
- At a minimum, we will define two business entities, the **Game** and the **Player**
- A Chess AI and a User will both be categorized under the **Player** entity abstraction
- A **Player** entity will interact with a **Game** entity via a well-defined interface
- Chess AI(s) will use the adapter pattern to implement the **Player** interface. The result is both a decoupling of algorithm development from infrastructure development, allowing parallel development trajectories, as well as risk mitigation by offering the ability to rapidly expand or swap out algorithm implementations with minimal code change.