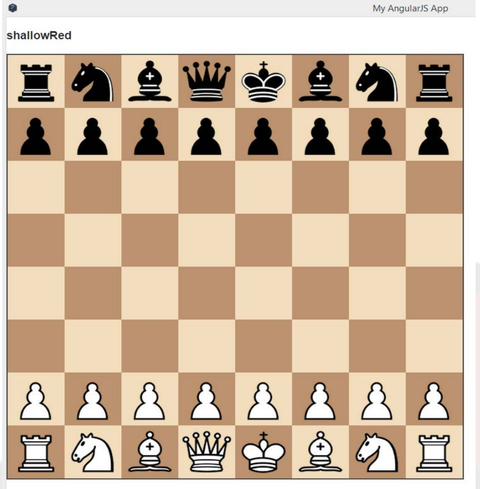
Shallow Red: A Chess AI

Sean Carter, Rocco DiVerdi, Manik Sethi, Jack Long

What was our project?

Our project is a chess AI that uses a "Min-Max" search algorithm to find the optimal move to make. It also uses "alpha-beta" pruning to reduce the number of calculations that it has to do, and utilizes a dictionary as an "opening book."

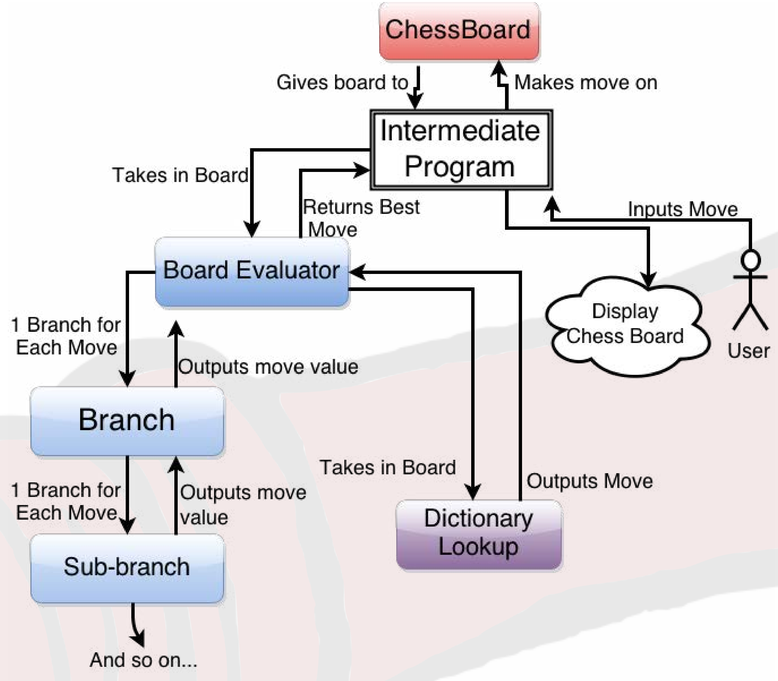


An screenshot of the program that works with the server

Want More Info?

- Go to our website:
  http://rdiverdi.github.io/shallowRed/
- Ask us! We know things!
- Become one with Wikipedia

How does our code work?



- The AI and the board are separate - despite drastic changes in the AI, we haven't had to wory about how to interact with our chessboard
- Our AI starts with a board, and creates branches (which create their own branches) based on all possible moves, and then they evaluate themselves recursively.
- The AI can also look up boards in a dictionary, to allow it to pick early moves more easily
- Finally, you interact with it through a command line interface that we created for it, or play against it on a server.

Min-Max and Alpha-Beta Pruning:

The min-max algorithm works by examining all of the possible moves that it could make. Then, it looks at all of the possible responses to each move. Then, it looks at all the things that it could do in response to THAT ... and this keeps going until it runs out of time, figuring out every possible move and counter-move.

Each time it looks at another layer of moves, it makes the moves on a board. Then, it uses an "evaluation function" to determine which resultant board is best. The AI is the "maximizing" player, which wants the best board, while its opponent is the "minimizing" player, which wants the worst board. The AI assumes that if its opponent can mess up the board, it will - so it won't make any moves that allow it to be put in checkmate.

An AI will always be a conservative player, making the most benificial move it can, while keeping the more risk free.