Chess of Champions

Ethan Luong, Eldin Vujic, Victor Fong
Group 28

Project Overview
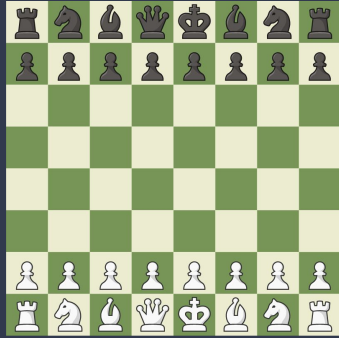
Chess of champions is an idea to incorporate elements of modern MOBA games into the traditional game of chess. The primary changes made will be to the board, the pieces, and the win condition. Another important part of this game is that players will be able to play against others online with the option to play casually or competitively.

Requirements

Our requirements are focused on…

- Server stability
- UI Usability
- Longevity
- Account data

Testing

- Allow up to thousands of players to access the server at any given time.
- Logging into a Game
- Creating account
- Buying Skins
- Buy DLC
- Access different menus
- Log out of the game
- Have different testing phases:
    - Closed Alpha
    - Closed Beta
    - Open Beta
- To test all of these test cases we will be using a AI testing service which will provide us information if the test either passed/failed. Which then our administrators will do the final check on the tests.
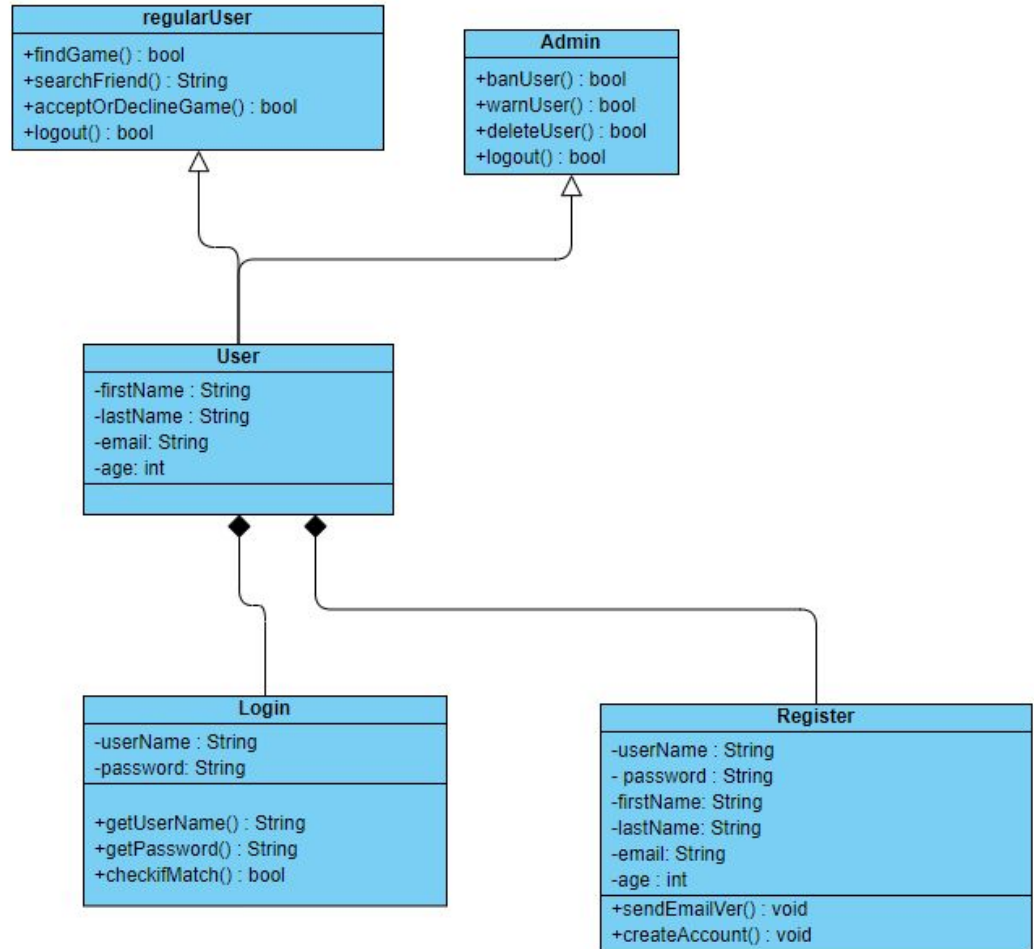
Project Design

Design Goals

- Our product must be enjoyable, competitive, and provide players to socialize.
- Anyone can play the game.
- Easy to learn, but hard to master.
- Our product must have adaptability, so the product must constantly be able push updates.
- Our product must have reliability to ensure players don't experience system crashes / No loss of their player data.
- Must include optimal performance to ensure there is little to no lag to so that players are able to play the game smoothly.
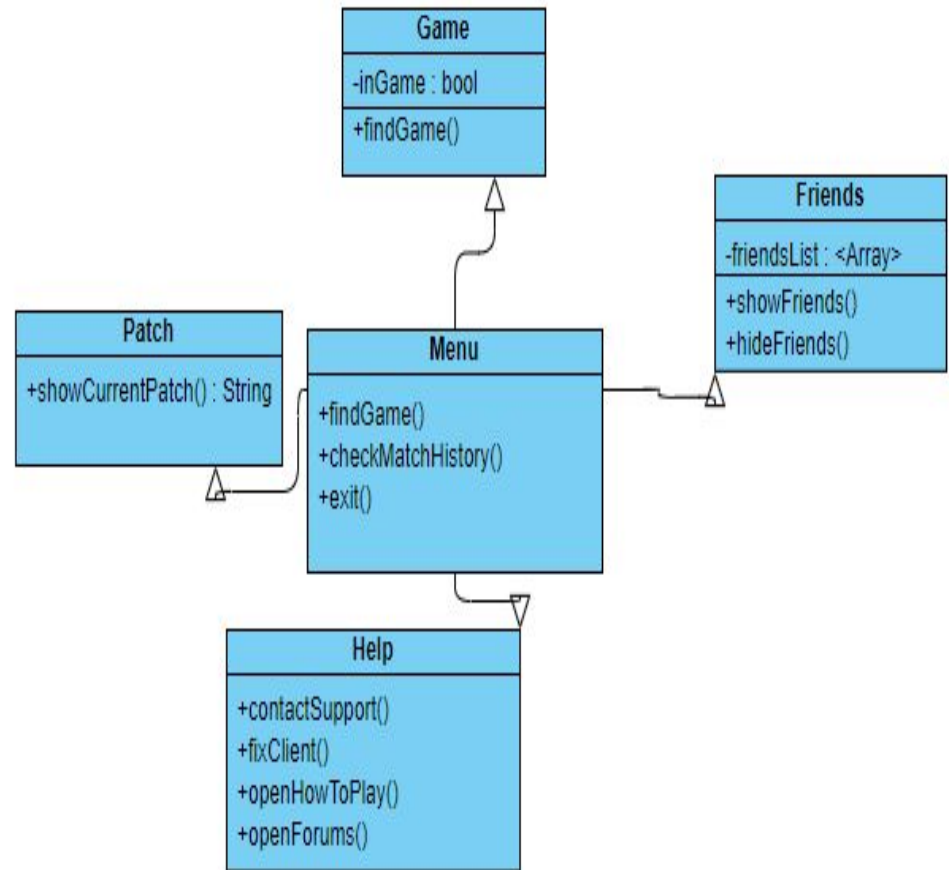
Project Design

Class Diagram - Login Menu

The Login Menu Class allows the user to log into the servers or create a account to access the servers.

regularUser
- +findGame() : bool
- +searchFriend() : String
- +acceptOrDeclineGame() : bool
- +logout() : bool

Admin
- +banUser() : bool
- +warnUser() : bool
- +deleteUser() : bool
- +logout() : bool

User
- -firstName : String
- -lastName : String
- -email: String
- -age: int

Login
- -userName : String
- -password: String

- +getUserName() : String
- +getPassword() : String
- +checkifMatch() : bool

Register
- -userName : String
- - password : String
- -firstName: String
- -lastName: String
- -email: String
- -age : int

- +sendEmailVer() : void
- +createAccount() : void

Project Design
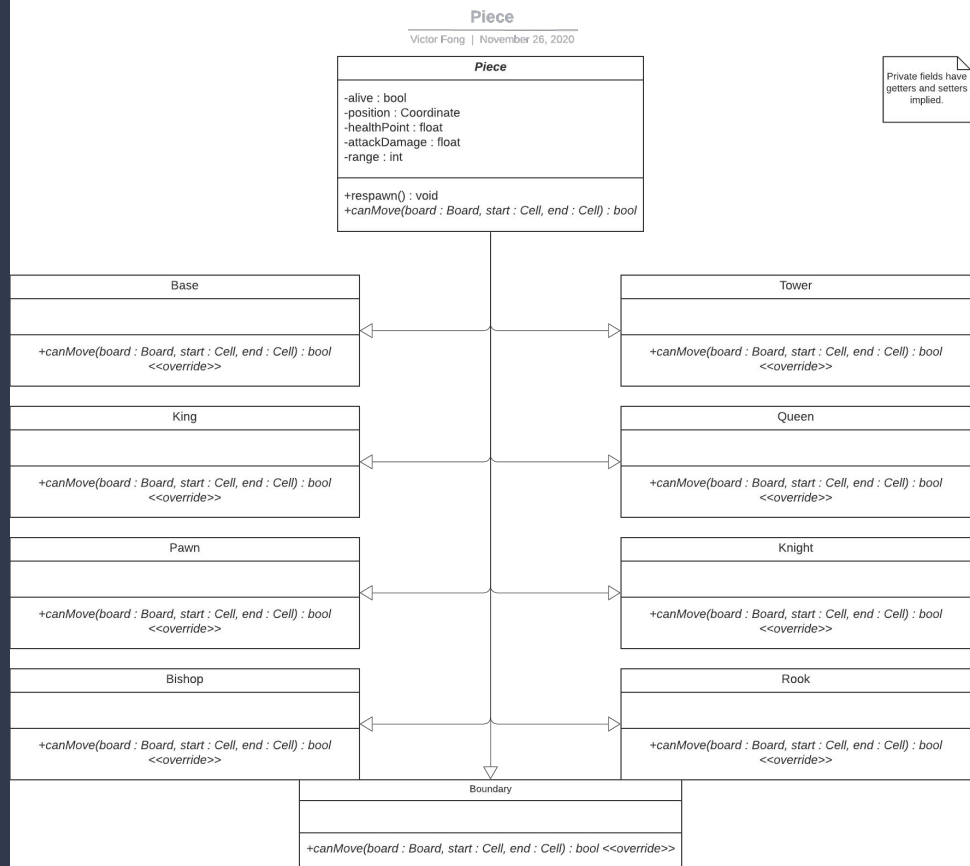

Class Diagram - Main Menu Class

This main menu class is used to navigate throughout the game, whether it is finding a game, contacting support, figuring out the current patch, or accessing your friends list.

Project Design

Class Diagram - Piece Class

A Piece refers to any and all playable characters/pieces of the board, all of which extend the Piece class to implement their own specific functionalities.
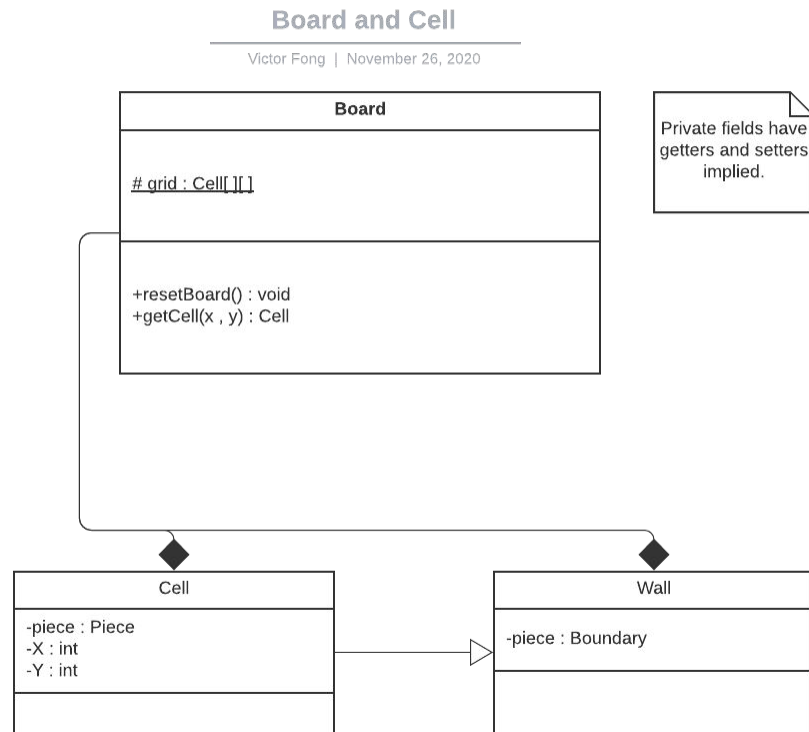

Piece

Victor Fong | November 26, 2020

Piece

-alive : bool
-position : Coordinate
-healthPoint : float
-attackDamage : float
-range : int

+respawn() : void
+canMove(board : Board, start : Cell, end : Cell) : bool

Private fields have getters and setters implied.

Base

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Tower

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

King

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Queen

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Pawn

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Knight

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Bishop

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Rook

+canMove(board : Board, start : Cell, end : Cell) : bool
<<override>>

Boundary

+canMove(board : Board, start : Cell, end : Cell) : bool <<override>>

Project Design

Class Diagram - Board Class

The Board is the field in which pieces will battle upon, composed of a
8x8 +
4*(8x3) +
2*4 +
4 grid.

Victor Fong | November 26, 2020

Board and Cell

Board

grid : Cell[ ][ ]

+resetBoard() : void
+getCell(x , y) : Cell

Private fields have getters and setters implied.

Cell
-piece : Piece
-X : int
-Y : int

Wall
-piece : Boundary

Project Design

Class Diagram - Player Class

A Player is any person or AI that can manipulate the pieces on the board. Aside from use of AI, this is typically a User.
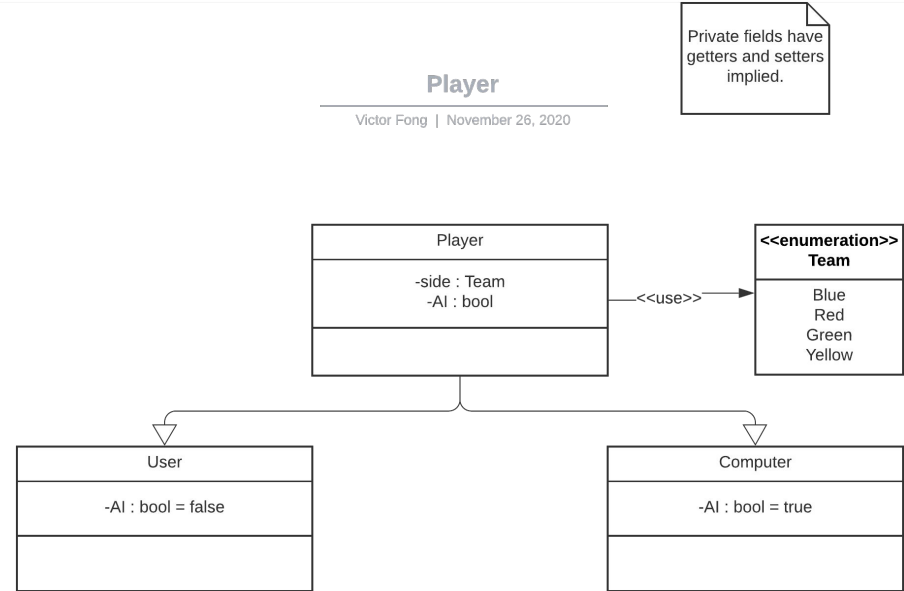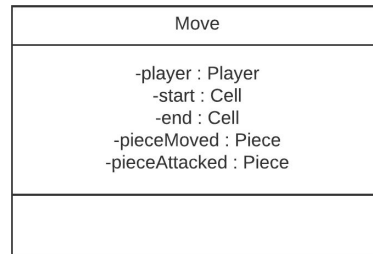
Project Design

Class Diagram - Move Class

A Move corresponds is an encapsulation of the information necessary to compute a given action that a player may want to accomplish during their turn.
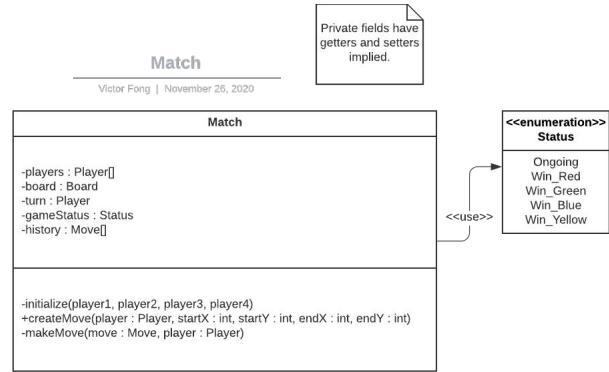
Move

Victor Fong | November 26, 2020

Move
-player : Player -start : Cell -end : Cell -pieceMoved : Piece -pieceAttacked : Piece

Project Design

Class Diagram - Match Class

The match is the active game that is played, serving as the interface for players to manipulate the board and its pieces.

Subsystem Decomposition