Networking Document

The networking architecture chosen is the client server architecture. In order to implement this, the game should be split into two main components: the server side, and the client side.

Client Side

The client side will control the user interface of the game and be installable on the player's computer. A handler class will be added that serves two functions. Firstly, keeping the user interface and the game player in the server in sync. Secondly, sending player input to the server and receiving changes back from the server.

Server Side

The server side will handle the logic of the game. When a player initiates the game from the client side, a connection will be made to the server informing it that a client wants to connect to the server. The server will also store two types of data: player data, and game state data. The player data includes the following: usernames, passwords, scores and players' history; while the game state data includes the following: current game players, current scores and current board layout. Saved games will be stored on the server and can be loaded from the client side.

Other Aspects

A player management service will be added which will work as follows. When a user initiates the game on his/her computer, a connection will be made to the server and the server will respond with instructions to the UI telling it to render the start screen of the game. At this point, the user will be asked to log in or register, if they haven't registered previously. This service serves a useful function by enabling the game to uniquely identify players and help keep track of their statistics even when they are logged out.

The player will start a game and can decide to play with a bot or publish the game where other online players can join. If the player chooses to play with a bot, the server will respond with instructions for the UI to render the board, whereas if the user chooses to player with another person, the server will respond with a list of available games which are waiting for players to connect to them. If there are no existing games, it will publish the player's own game as a new game to other players so that they may connect.

The server will keep track of the game statistics, which includes the players of the game, the appropriate players' turn, the score of each player, and the current state of the board. The client side will handle the dice roll in order to randomize it. The hander will send changes from a player's side and send them to the server, the server will then send instructions to the other players' handlers on how to update the UI. In the case where a user rolls dice, the resulting number will be sent to the server and it will assess the current state of the game and check if the move is valid. If it is valid the server will send instructions to each player's handler to update the UI accordingly.

When a player chooses to save the game, the other players will be notified, and the game will be saved on the server so that the players can resume playing from the stored state. When a user quits the game, the other players will be notified and if there are enough remaining players to continue playing, they will be able to do so, if not, the game will end.