**Assignment 2: Draughts**

**Interim Progress Report**

So far with my Draughts app I have created three activities: The main menu, settings screen and the game itself. MenuActivity simply has five buttons: New game, Load game, Leaderboards, Users and Settings. Currently the only two active buttons are new game and settings. When fully implemented, the new game button will send the user to the GameActivity with intent to create a new game state, the load game button will send the user to the GameActivity with intent to load the game from a saved state. The Leaderboard button will send the user to an activity with a ListView of all users sorted by high scores, with the name of the player and their score. The Users button will send the user to an activity which displays a ListView all of the users in the app, and also an option to add a new user. The Settings button currently creates a PreferenceFragment with two switches for the gamemode and background, however they do not currently affect anything within the game, and the app often crashes when opening the settings activity.

I have spent most of my time so far implementing the game rules and how it is displayed. The layout is a single GridView, with each cell being its own board\_cell layout: a background ImageView with a piece ImageView on top. The class BoardAdapter sets each alternate cell to either cell\_black.png or cell\_white.png, however depending on settings it will only display cell\_white.png when that is implemented. The adapter then checks whether there is a piece at the position it is displaying, if there is it checks what team it is, then whether or not it is a king, and displays the correct image based on that information. If there is no piece at that position, it sets the piece ImageView to 0 to remove pieces that have been moved or taken. If a piece is selected, the adapter changes the cell background to cell\_selected.png so that the user can easily visually identify the piece they selected by its green background.

The game is represented by the class GameBoard, which contains the game mode and the game board: an array of GamePiece’s, with every position in the array containing a piece. The GamePiece class contains values for if it is selected, if it is king and the team it is assigned to. If a piece’s team is 0, then it is a blank square, this is to prevent null pointer exceptions.

The GameBoard class also holds much of the code for movement and taking pieces. Moving a piece to either end of the board turns it into a king and once a king, a piece can move in any direction. I currently have two systems for taking pieces: my initial version, (move 🡪 legalMove), which obeys most of the rules of the game, but doesn’t automatically take pieces. This is controlled by clicking on the piece you want to move, and then clicking where you want to move it.

I also have a second version (movesExtra 🡪 findMoves), which is more complicated and powerful as it uses recursion to find all the possible routes a piece can move in. Long clicking on the piece that you want to move runs this method, however it will remove all possible pieces that can be taken if it is run. It is supposed to only find the possible moves to make and then relay the information back so the best move can be done, however it is iterating on the main game board when it should be using its own separate one.