# Assignment 1 - echo Group Report

The program works as follows:

**Game2** starts the game by asking each player to input their names, and then prints the board using the board.print() method. It then asks the players to make their moves. Further plans for this class involve reformatting to create an initializer method and a while loop to repeatedly ask the players to make moves in sequence. It would print out the board and each players pieces each time.

**Board2** implements the game board as a two-dimensional character array, and gives getter and setter methods for reading/changing what character is stored in each array. It also provides a .print() method that prints the current state of the board to the terminal.

**Player** implements an abstraction of a human player, with the ability to make a move using a Move.makeMove() method. It gives each player an ArrayList of Pieces and a character token i.e. what they will place on the board, x or o. They also each have a name.

**Move** implements the two key parts of making a move, checking if it is valid and then updating the board. It’s private classes implement these methods but everything aside from the makeMove method is private and hidden.

**Piece** implements the pieces themselves, which are private static final integer arrays hidden behind a static final Piece class. They are made as a square array with a length and a name, and various getter methods along with a rotate method which allows the pieces to be rotated.

**Tests** are implemented as needed for the classes and test the functionality of the methods described above. We strive to minimise the need for testing with strict use of access modifiers and flow control.

Model-Controller-View:

This program is implemented along the design pattern, and can be explained as follows:

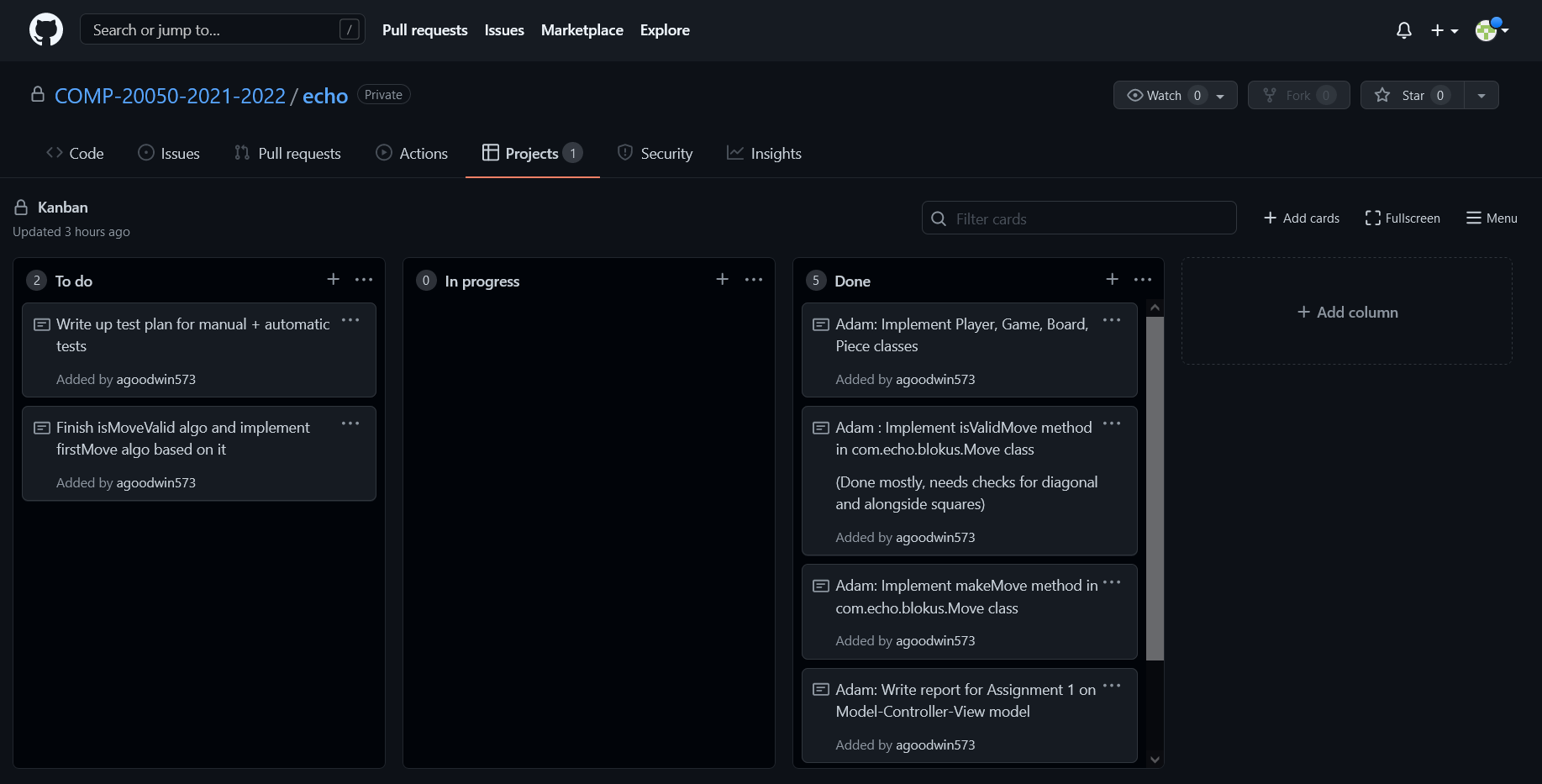
View: This is the Board.print() method until the point where a GUI is implemented.

Controller: This is the Game class and the Player.makeMove() method.

Model: Everything else, specifically the Piece, Move, Player and Board classes (aside from what is referenced above).

This design pattern works well for this kind of project where it is being completed in stages and allows us to leave a placeholder View class in place until we implement a GUI while still being able to test the functionality of the program.

Kanban:



Test Plan: To be completed

Result:

