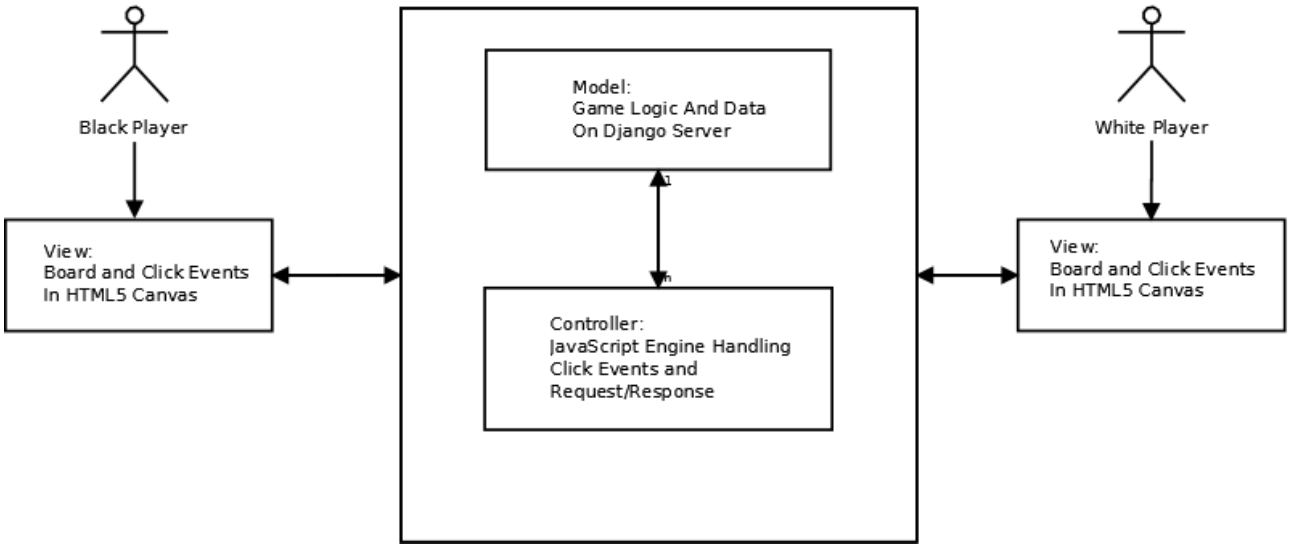
Document for Discussion.

Client Side Architecture.

The system should utilise a Model View Controller Architecture.
For the application:

- The View is the Board and UI Elements, which will be dynamically drawn on a HTML Page.
- The Model will be the game logic and game state which will be stored on the Django web server.
- The Controller will be the JavaScript engine handling user actions and communicating with the Model.

MVC Architecture Schematic



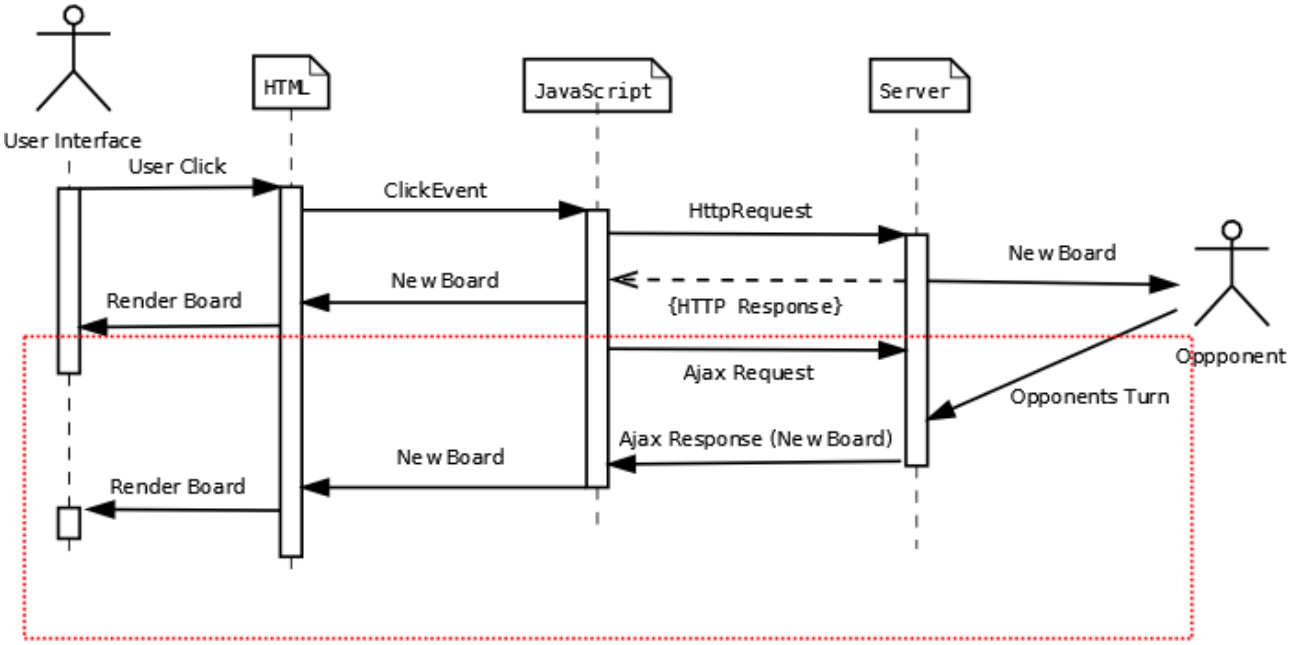
View.

- The user interface will be implemented in HTML5.
- The UI will consist of a board, a pass button, a score label and another element to relay state to the user.
- The board will be displayed in Canvas and will be drawn dynamically using JavaScript with JQuery.
- Pieces will be placed by clicking on a point on the board, provided the move is a legal move it will place a piece at the appropriate position nearest the location of the click.

Controller.

- The Controller will be implemented in JavaScript (with JQuery).
- Communication with the Model, (on the Django Server), will take place using HTTP POST and AJAX calls.

Sequence Diagram for a user Turn:



- A users move will first be passed to the JavaScript engine where it will be processed and sent to the server. Some state checking may be performed on the Client side in future versions.
- The Server will then return the **new board state**, **notification to the player of a pass, or notification of an illegal move,** depending on the situation. New Board state will be returned to both players.
- When the Controller (JavaScript Engine) returns a new board, an AJAX request (Highlighted in the red box) will be sent to the Model. This is essentially a response which waits for the opponents move to be made. Thus when the Model returns a new board it will return a new board to both the player whose turn it is and the opponent, via HTTP response.

JavaScript Design:

(Note: This is an unfinished design)

Objects:

Move:

- Has a position. [x,y]
- Has a colour. 'black' or 'white'
- Pass: True or False

The Pass functionality has not been implemented yet. It is proposed that a constructor will either take position and colour OR Pass = True and colour.

xmlHttp:

- An XMLHTTP request object.
- Will be created by createHTTP()
-

Poorly named, we are using JSON. (JavaScript Object Notation)

jsonObject:

- Represents board state received from the server.
- converts from JSON to JavaScript Object.
- Consists of an object with a number position and colour for each piece on the board

Controller Methods:

fetchBoard()

- Retrieves board state from Server.
- returns a JSON document.

(“canvas”).click

- Click Handler for new move.

Yet to be implemented.

1. AJAX Method to be written.
2. Possibly method to use local storage to prevent sending a move to the server with a position that is already occupied.
3. Pass Functionality.

View Methods

loadCanvas():

- Draws the board.
- Will be abstracted to draw different boards depending on message from server.

drawPieces():

- Draws pieces on board based on jsonObject passed from the server

getCursorPosition():

- Locates relative position of click in board.
- Returns position in terms of the nearest board position. (i.e. the nearest intersection)

Yet to be implemented.

1. Off board UI elements, such as pass button or score label.