# Clue Design Document

We need to stream-line and have a solid plan for this:

Index.php asks the user if they want to join or create a game.

Create.php asks how many players( 3 – 6 )- asks for players name- and asks for character selection

* + when this form is entered and the submit button is clicked,
  + Insert generated gamekey, number of players, and 1 into players joined into main
  + Insert gamekey, name into player\_assignment –character name (“scarlet” ) = player’s name
  + Set session variable [“initialize”] to something – session variable [“gamekey”] – [session variable [“playerName”] -- session variable[“numberPlayers”]
  + Proceed to wait.php

Join.php asks for gamekey, player’s name, and character selection(with whatever characters taken up being omitted)

* When the form is submitted via POST
* Select all from player\_assignment where gamekey=@gamekey
* You have to manually loop through this with if statements due to the construction of the table, check to see if each character has an assignment ( ie $result[“Scarlet”] == null)
* Then just write out the script that shows available characters based on this
* Select players joined from main
* Add one to received value and update main table
* update player\_assignment table set the selected character name [“scarlet”] = playername where gamekey=@gamekey
* Set session variable[“gamekey”] session variable [“playerName”]
* Proceed to wait.php

Waiting.php refreshes every 5 seconds

* If [“initialize”] is set – select number of players and players joined from main table,
* If they match, run initialize script
* If [“initialize”] isn’t set – select gameActive from main, if gameActive == true, grab all of the player’s names from the player\_assignment table and populate a player’s array with player names in the order of their turns, then proceed to gameboard.php

Initialize Script

1. Grab all of the player’s names from player\_assignment and put them all in an array in their turn of order, probably another instance of manually loop with if statements again
2. Hand out the cards and populate card location table
   1. Initialize cards in their own arrays: suspect[5] weapons[5] location[8] – each with the card name as their value
   2. Grab a random value from suspect[5], weapon[5], location[8], set “solution” for a value to them
   3. Update the values for each of those arrays locally to “solution”
   4. Loop through the rest of the unassigned cards (random # 1 -3 to pick which array, then another random number appropriate to array’s size to pick which card). Just check to see if the value is any of the player’s names or solution and reroll if true. Assign their values to playerNames, starting with whoever is first in the player array, going to the next player, and resetting the array incrementor when it’s greater than [“numberPlayers”]
   5. Once you’re done setting the arrays locally, insert into card\_location table each value from each array and the gamekey
3. You have to set default locations for all characters in the player\_location table, I’ll make a list of the coords
4. Update main table set gameActive = true, playerTurn = player[0]

gameBoard.Php will be where the whole game takes place

* Run some automatic scripts: gameboard and chat drawing and determine what state the player should be in
* 5 possible states the gameboard can be in : waiting, movement, suggestion, accusation, and game finish

Gameboard is a grid of divs 24x25 that have an id of “x$x-y$y” and a name of “$x-$y”

* The room divs are transparent and the wrapper div holds the images of the rooms as background
* Each of the rooms, doors, and out of bound coords are mapped out
* Select \* from player\_location table and make those corresponding locations have the background color appropriate to their character
* Each room will have 6 different assigned spaces for characters to be placed when they are within the room. Select \* from player\_location and check against the room, count how many people are standing that room and use this information to assign the character into the right spot [ 1- 6].

Chatlog is a textbox that has its values get populated by select statements in the chatlog

* A user who enters a message in the field will do an insert into statement to the chatlog table of the current datetime and the message they entered
* The one issue I can think of is that pulling all of the messages everytime the page is drawn may be a bit heavy on the SQL server, so we should structure the sql statement to only pull messages newer than the latest message and possibly store the chatlog as an object in session, otherwise the functionality of it is pretty straightforward

The waiting state of the gameboard.php page should refresh every 5 seconds.

* Essentially, on refresh, the page should check to see if the session [“playerName”] matches the main table playerTurn value
* If it does, reload the page and run the movement state script instead of the waiting state script

The movement state of the gameboard.php page will handle player movement on the player board

* There should be a div that’s contents get changed with what state the gameboard is in. If you’re in waiting mode, it should probably just state basic info (whose turn it is etc). If you’re in movement mode, there should be a picture of two dice and a button the runs the dice roll script. On refresh, the page should show the outcome of the dice roll and the actual gameboard should include onclick event handlers that will run a script that will see if the player clicked on a space that is valid to move to based on the dice roll, the other player’s locations, and the door locations. If it isn’t, display an error to the player
* Once the player clicks on the board in a valid position, update the player\_location on the db with the new coords. If they clicked on a door position, move them to appropriate coords based on the room and who is in the room, and set their position in the player\_location table to the name of the room.
* If they are currently in a room when their turn starts, have the valid position script run from both door tile coords to give them valid coords based on all door locations (they can leave from any door). Also, if applicable, allow the player to move through a secret passage instead of rolling the dice
* Once the player is finished moving, check to see if they’re in a room, and if they are, on refresh, run the suggestion state scripts. If they’re not in a room, move directly to the accusation script

The suggestion state of the gameboard.php page will handle suggestions and card\_location calls to check to see who has what cards

* When the script runs, populate the div that previously held the dice and dice roll buttons with three drop down boxes that hold the selection of all possible cards and combinations (suspects, weapons—location must be the room the player is currently in) and a post submit button
* When the player submits their selections, find the current player’s name in the saved session array, then find the next player in line, and do a select statement on the card\_location table for any of the selected cards that match the player’s name. If it doesn’t come up with any results, then proceed to the next player. When a player comes up with a match, show a graphic of the card to the current player with another post submission button that runs the accusation script on refresh. If there more than one card in the match, just show one card.
* Also, change the location of the player who is listed in the suggestion to the current player’ room (when you suggesting
* Make a chatlog entry of the player’s name and the suggestion in the chatlog.
* When there is a match, make another chatlog entry of the player that is holding the matched card showing the current player “ a card” – don’t show which card
* If there are no matches among the rest of the other players, just display a message that indicates this to the user. Don’t check the current player’s name against the cards, due to it being a strategy in the game
* When the player is done and clicks the finish post button, then refresh and run the accusation state scripts

The accusation state should populate the same the div that suggestion and dice roll populate

* It should ask the user if they want to make an accusation with a warning that if they’re not right, they lose the game. Have a yes or no button.
* If they hit yes, create fields in the box similar to suggestion, but have every card available for selection. Have a enter button and a cancel button just in case they change their mind. If they select cards and hit finish, then do a select statement on the card\_location looking for matches with “solution”. If all 3 cards match, change the gameActive column in the main\_table to the current player’s name so all players run the gameFinish state scripts. If they don’t match, set the Session[“playerDead”] and run the gameFinish state script for just the current user manually through refresh.
* If they hit cancel, just change the playerTurn in the main table to the next player’s name in the array and refresh into the waiting state

The gameFinish state should display the value of gameActive column in the main table as the winner of the game and have buttons to start or join another game.

If you guys have any changes to the structure or anything with the design just update the doc, put the date on the bottom, and push it to the github

4:47 – 5/7/18