# Girdler Ex3

(web application + server)

# General description of the project

## Flow

When the user first enters the app , i check if the session have a user already logged in,
if so , a “welcome back” message will be shown. and a “continue\log out” actions will be available.
Otherwise, the user will have to enter the name he want to log in with and the type of user logging in, after pressing login he will be moved to the main lobby.
At the top part of the main lobby the user will have the select file & upload, or the logout options.
also  logged in players section in the left part of the screen , and a list of current games in the right part. at the top
When a game is clicked, a game info section will appear at the bottom of the page, including a small board preview, room details and a Join button (which will be disabled if room is full or user already playing there)
after joining a game, the user will be transferred to the game screen.
there , similar to last excursive will have two sections, the board and the info
A whole row\column can be selected by pressing the row\column wrapper (where the block numbers are) and a single cell can be selected by pressing it directly.
after selecting cell a user could submit his move by selecting the wanted color or just end his turn.

## server

The server i built is used as a”pipe”  between the the web client and the game engine JAR.
it contains two context maps to “save” and “restore” data for the users, one for the actual users, which is used mainly for lobbies and login\out during the game. and another map used to save games , using their names as the key (because not allowing two games with the same name, i thought hash map would be perfect for storing them).
set the context data (users\games) aside, i created a servlet for each action the user can do.

## client

The client is an angular web app that communicates with the server. all server communications are wrapped with “Services” that are used by all the other components. also Promises are used to handle every Async call in order to create a more clean and understandable code. for the template (html) and styling i relied heavily on bootstrap grid system and FontAwesome for additional SVG icons.

## peripheral

The client scaffolding was created with “YeoMan” a scaffolding framework that helps initialise projects ,have a lot of templates to start with for services, directives, filter and about every angular component that you can think of.
The styling was created with SASS

# Files structure

## server

the server have 4 folders (initially it was heavily based on the Chat example).

- **logic** - have classes that holds the map objects for the context data (users\games).
- **models** - contains all classes used to hold data passed from the server to the client.
- **servlets** - every interactive action is separated to its own servlet (EndMove,GetGameSettings,GetGeneralGameState,JoinRoom,LeaveGame,Login,LogOut,MainLobby,Redo,SubmitMove,Undo,UploadGameFile)

- **utils** - contains 3 util classes that contains common actions that are used in the code for reusability and easy maintenance.

## client

there are 3 important folders in “web” folder:

- **scripts** - contains all of the javascript for this project and separated to its own sub folders as well:
  - controllers - hold the **main** app controller, the root scope of the project
  - directives - contain a directive for each area of the project (login,main lobby, and game room)
    in each directive lies the main logic for this area, how things should behave.
  - filters - contains template filter  (mapping color to icons for the history section)
  - models - contains structures for data that is passed to\from the server
  - services - contains mostly communications services that wraps the hole ajax mess into nice classes. also we have there is a service in charge of the “complete blocks” logic in order to keep a maintainable “gameroom” class (it is bloated as it is.. no need to bloat it more)
- **styles** - contain the SASS files the compiles to the css file used
- **views** - contains the html template used with the directive (for each directive i have a matching html template goes by the same name ), also “main.html” is the view for the root scope. it contains the three base directives

# Bonuses

* In the main lobby whenever you click on a game, opens a game details section,which also contains a small preview of the board.

* a player can log in from different browsers and play with his the logged in user (only for the same session) in multiple games, note that this bonus is not fully complete, because it won’t work with different browsers, because it needs the user from the same session. otherwise it will prevent logging in with the same name)