# Git & App structure

The git repository is structured such that the node server is separated from the angular app. The node server is then also subdivided again between the core JavaScript files, which handle the requests. Then the REST endpoints are separated out from that, and finally any physical files, like JSON files are on their own. This allows me to keep the application modular and its parts interchangeable.

The angular application itself is also separated out into components and services where logical. The chat is a separate component from the system management buttons, this would allow me to replace that portion of the page for the user with another control, like a profile page. Then all the handlers for socket interactions are separated out again. The benefit of this is that a different library to sockets.io could one day be implemented with minimal if any changes to the actual application.

# Data Structures

The main data structures used in the application are as follows.

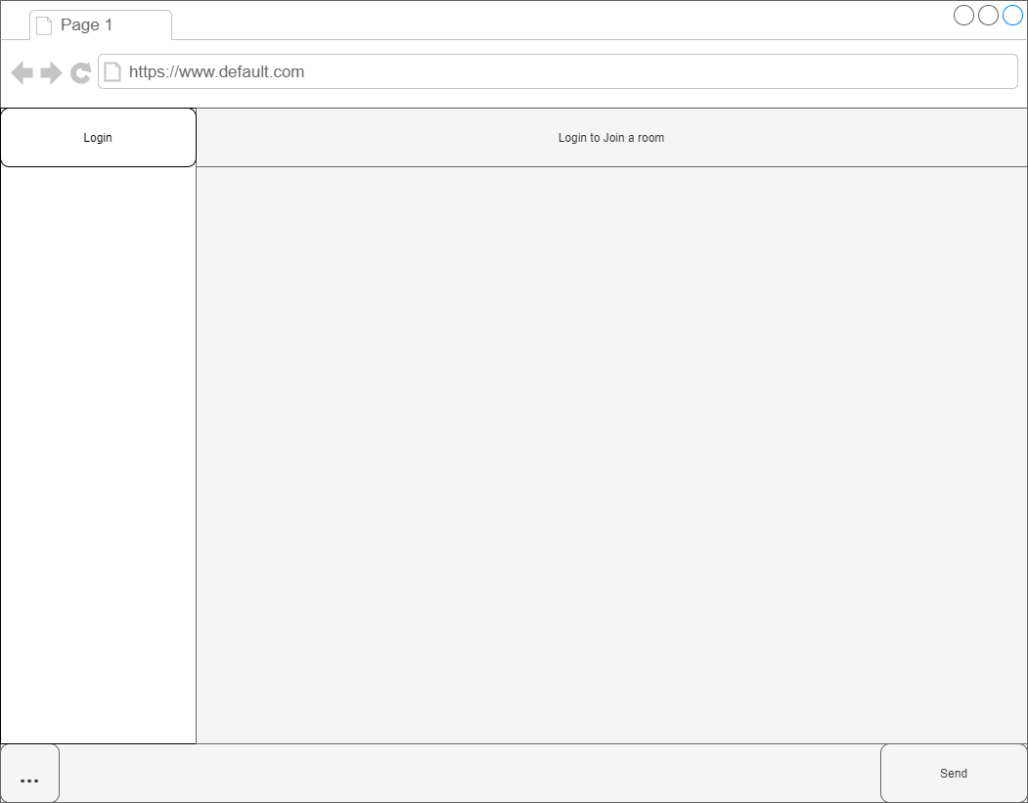
User: stores all the personal information for each operator of the application and allows the application to know which client should have what level of permission.

Channel: A channel is a part of the website where users can communicate with each other in a segregated way.

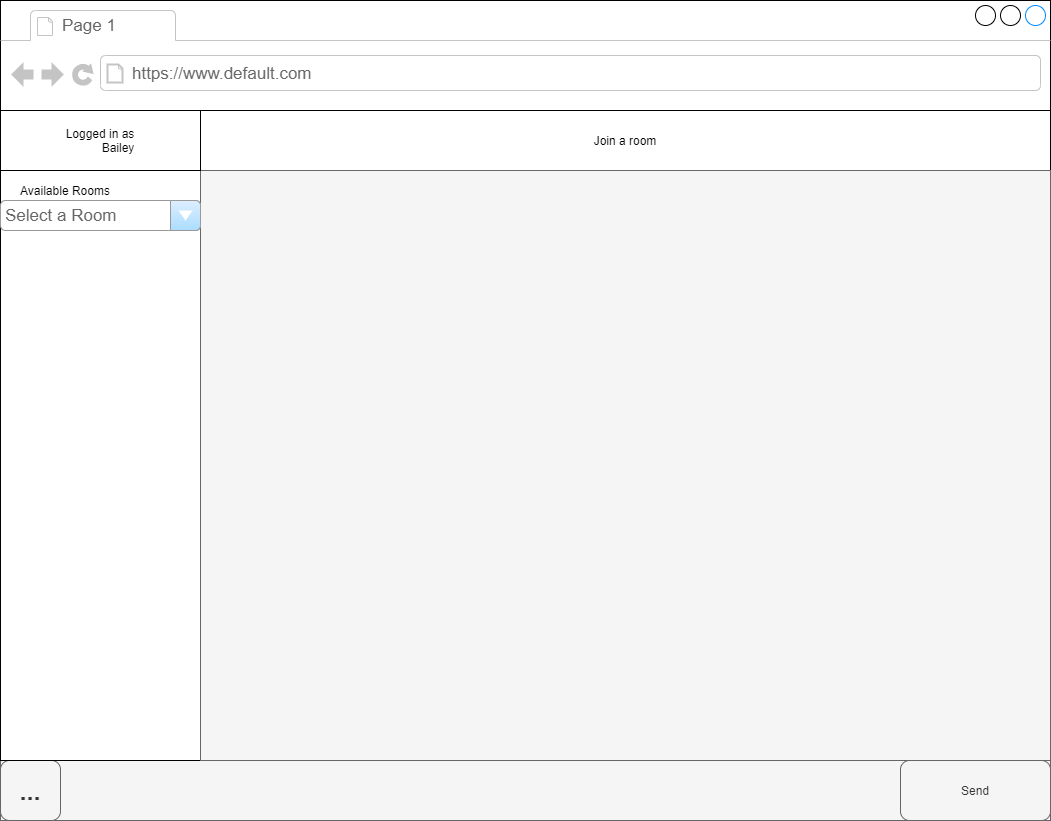
Group: This is a collection of people and channels and is primarily used for filtering things that each user sees on screen.

# Client vs Server roles

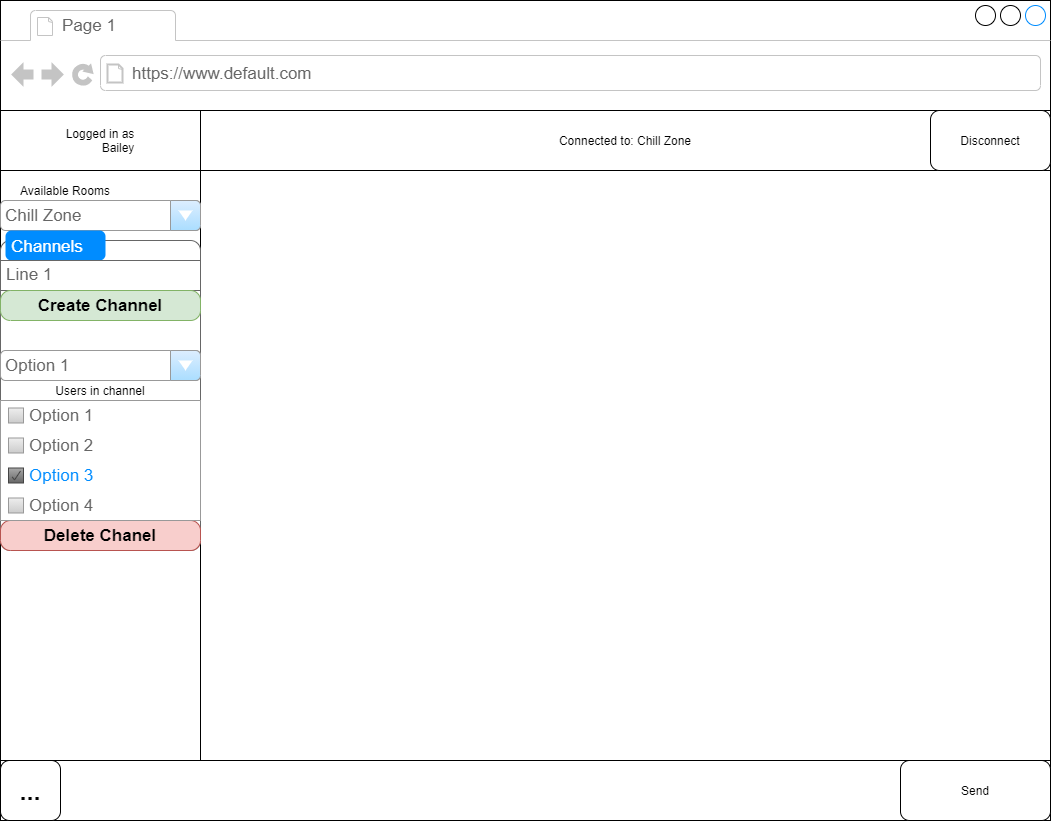
I kept as much of the logic as possible in the server side, this was for security reasons. A user could run scripts on their own machine and change what level of permission they have if the group information and permissions were stored in session memory. Instead, although the local client may be allowed to see some of these things, all changes requested to node are checked with the data stored in the JSON files/DB prior to the change taking effect.



Logged Out State



Logged in State



With extra options