Assignment 1 Documentation

**Student Name: Waris Usmani**

**Student Number: s2921187**

**Git Repo: https://github.com/warisusmani/Assignment-1.git**

# Git Structure:

* Final touches (01/09/2019):

This commit indicates the final touches made to the application to fix bugs and finalize the working operations of the application

* Adding Users, Groups, Roles and Channels (01/09/2019):  
  This commit includes the operations of adding users in the JSON files, along with their specified roles, groups and channels which the users can be added to according to their specified groups.
* Removing Users, Channels and Groups Functionality (01/09/2019):

This commit includes adding the functionality to remove the users, channels from groups and groups from the JSON files. These functionalities have been assigned to the Super/Group admin role permissions

* Added Role permissions (29/08/2019):

The roles were assigned and added, in order to render certain functionalities within the dashboard. Controls were authorized only to certain roles

* Added Role operations (01/09/2019):

Controls were added for each role permission in order to perform a certain functionality such as viewing all users, groups/channels and assigning a certain role to a user.

* Chat history and chat page components added (29/08/2019):

The components to view the chat history in a channel and going to the chat page were added

* Added login/log out functionality with session storage (29/08/2019):

The functionality of adding details of the user to the session were added when the user logins in and details were cleared when the user logs out

# Angular Structure:

The following components have been added for the client side:

* Account component: This component renders the account details of the logged in user including their username, role, groups and channels. This is the most important component of the application as it allows redirection towards different components for performing all the main operations by different roles. This component renders all controls according to the assigned/specified role of the logged in user
* Channel component: This component is used to control the channels by adding them to a group
* Chat Component: This component is used to redirect the user towards the chat room to send/receive messages within the specified channel
* Group Component: This component performs certain operations which involve groups, these operations mainly affect the process and end result of the specified groups.
* History component: This component is used to show the chat history within the selected channel on the client side
* Login Component: This component renders a login form for users to login and then control the authentication process to allow the user to navigate to the account component or restrict them based on the available data within the JSON files. The session storage is also controlled in this process.
* Remove Component: This component involves the controls to remove the users, groups and channels from the JSON files
* User Component: This component performs certain operations which involve users, these operations mainly affect the process and end result of the specified users.

# Node architecture:

The node architecture mainly contains a master file on the server, this file is known as ‘server.js’. This file loads all the routes and matches them with the API requests coming from the client side. These routes perform functionalities which are requested from the client and in return sends back a response indicating whether the requested operation was successful or not. The route details are as follows:

* Auth.js: This route is responsible for authenticating the user on the application by receiving the user name from the login form on the client side and checking the username against a value of available users on the JSON files. If the user exists then send back the response of success, if the user doesn’t exists then send back a response of in valid. Once the client receives a response of success, the client authenticates the user.
* Channels.js: This route involves the operations of deleting a channel from the JSON file. This route checks if the requested channel exists in the JSON files. If the channel doesn’t exist then the response of invalid is sent, otherwise the channel gets deleted from all the JSON files (the users added to this channel, have no longer access to this channel). This route is used to fetch channels for the client side as well, in order to perform certain functionalities with them.
* CreateChannel.js: This route checks the JSON files if the channel already exists or not. If the channel exists then the channel is not re-created, otherwise the new channel is added to the JSON files.
* CreateGroup.js: This route achieves the functionality of creating a group that does not previously exists in the JSON files.
* DeleteGroup.js: This route achieves the functionality of deleting a group that does exists in the JSON files. When this group is removed, the user that were previously added to this group have no longer access to this deleted group.
* Groups.js: This is used to fetch the groups on the client side in order to perform certain functionalities with these groups.
* ListChannels.js: This route is used to fetch channels which each user has been added to according to their groups
* Roles.js: This route permits all the role-based operations such as adding a user to a specified role
* Register.js: This route is used to add a user which does not exists in the JSON files
* User.js: This route is related to performing user related functionalities that is removing a user from the JSON files which exists.

# File Structure:

* Authdata.json: This file stores the information regarding each user that is their name, group that they have been added to and the permission that they have been assigned with. This file is also used for authentication purposes by the client side.
* Channeldata.json: This file stores the channels which are added under each group
* Channels.json: This file stores all the channels, used for fetching purposes only
* Groupdata.json: This file stores all the groups, used for fetching purposes only
* Userchanneldata.json: This file stores the users and the specified channels that they have been added to by the admins. These channels are only available within the specified groups of the users.
* Userdata.json: This file stores all the users, used for fetching purposes only.

# Table of Index

|  |  |
| --- | --- |
| **Structure** | **Description** |
| Global variables | Global variables have been used throughout the client side and server side to store users, roles, groups and channels by fetching them from the server side. Similarly the server side has global variables to store responses, error messages, success messages and fetched data arrays. |
| Services | There are 2 services used: group and user services which initiate API request according to the group and user based operations respectively. |
| Models | There is one model used named as ‘fetchModel’, this model is used to match the response data from the server, in order to be received/rendered on the client side |