3813ICT – ASSIGNMENT

Phase One

Casper Deserranno
S5149390

Contents

Documentation

GIT

Git was used throughout the entire development of the application. The assignment folder contains the entire project and is pushed directly up do github from the master branch on my PC. No branches were made because the development was procedural, meaning development of a new feature was not commenced until another feature was finished. This was simple because there is only one author of the system, using only a single computer. If this were a group assignment, I would have used branches, and will use them for each ongoing phase of the assignment, however for the 1st phase, I did not see the benefits.

Regarding commits, as mentioned before, the system was built procedurally, hence, each commit typically alters/adds the same 4 files. These are the home.component.html, the home.component.ts, the server.js and a new javascript file in the server/routes directory. Since the majority of the features are located on the home page, the home html and ts files were updated with each new feature implementation. A new server javascript file was added with each feature to read and write relevant data from storage (server/data directory). This new file is directed through the server.js file with include statements.

Using this process, it was easy to determine when to commit changes and push them to a remote repository. See figure 1 for a PrintScreen of the git log in the git bash terminal (bottom of the doc).

Data Structures

There are 3 data files which make up the storage aspect of the application which reside in the server/data/ directory. These files are:

users.json

This file contains the JSON code storing the login details of each user. This is just an array of objects storing the username and password of each user. This is the file used for authentication.

userData.json

This file contains the JSON code for storing all the other user information, this was separated from the users.json file because the password is never accessed once a user has logged in. It is also an array of objects like the users.json file. This file is accessed once the used has logged in to retrieve their role so different permission levels can be assigned to different users. It also stores this email address which currently is not being utilised.

groups.json

This file is currently the most accessed file in the entire system, it stores all the information regarding the groups and channels of the system. The JSON file contains an array of objects much like the other 2 storage files, however it contains much more information, hence each object in the array has nested arrays and objects. Each object at the top level of the array represents a group, inside the object, the groupName, members and channels are located. The **groupName** is a string containing the name of the group, **members** is an array containing the username of users which belong to that group, **channels** is another object which nests the **channelName**, and **members** of the channel. The **channelName** is just a string holding the name of the channel, the **members** attribute is another nested object storing the details of the members, currently only their username .

REST API

The REST API was used for communication between the Angular front-end and the Node.js server. The Angular front-end utilised port 4200 while the Node.js listened at port 3000 awaiting any requests sent to that port. All the posts sent to the server were redirected to their respective javascript file from the server.js file. All the files are directed to the routes directory in the server.js file. These routes are:

/api/auth

This route will be redirected to the postLogin.js file. It is responsible for user authentication when a user attempts to login using the login form.

Parameters:

- Object containing two string:
    - Username
    - Password

Returns:

- Object containing 4 variables:
    - Username
    - Role
    - User ID

The postLogin.js file will check the username and password which were passed through in the parameters against the users in the users.json file. If the username and password match, additional user information is fetched from the userData.json file for the relevant user and the information will be sent back to the front-end. If the username and password do not match, nothing is sent back to the front-end.

/addUser

This route will be redirected to the addUser.js file. It is responsible for checking validity of new user variables and storing new users in storage when created.

Parameters:

- Object containing 5 strings
    - NewUsername
    - NewEmail
    - NewPassword
    - NewPassword2
    - NewRole

Returns:

- An error message (null if not applicable)

The addUser.js checks the variables passed through the parameters for validity, if all variables are valid, a new user object is written to the users.json and userData.json files.

/findUsers

This route will be redirected to the findUsers.js file. It is responsible for fetching all the users from storage (used to list users which can be deleted.)

Parameters:

- None

Returns:

- usersArray
    - Contains the username of each user

The findUsers.js reads the users.json file and writes all usernames to an array which is sent back to the angular front-end.

/delUser

This route will be redirected to the delUser.js file. It is responsible for deleting users.

Parameters:

- User id (index of usersArray above)

Returns:

- usersArray

This file reads the users.json file and splices the specified user from the array using the user id given. The same is done for the userData.json file. The array with the user removed is returned to the front-end

/findGroups

This route will be redirected to the findGroups.js file. It is responsible for finding the groups for each logged in user.

Parameters:

- Username
- Role

Returns:

- userGroups array storing applicable groups

This file reads the groups.json file and checks which groups the user is a member of. If they are a member of that group, that group will be pushed to the userGroups array. If their role is Super Admin (SA) or Group Admin (GA), all groups will be pushed to the array. This array is returned to the front-end.

/findChannel

This route will be redirected to the findChannel.js file. It is responsible for finding the channels for each user in a specific group.

Parameters:

- Username
- Group

- Role

Returns:

- Channels array storing applicable channels.

This file reads the groups.json file and finds all the channels which a user belongs to. Applicable channels will be added to the channels array. If the user's role is Super Admin (SA) or Group Admin (GA), all channels will be pushed to the array. This array is returned to the front-end.

/deleteGroup

This route will be redirected to the deleteGroup.js file. It is responsible for deleting groups.

Parameters:

- Group name

Returns:

- Boolean

This file reads the groups.json file and finds the index of the group name specified in the parameter. Once the index is found, the groups is spliced from the array and the array is overwritten to the groups.json file. Returns true when it is overwritten.

/deleteChannel

This route will be redirected to the deleteChannel.js file. It is responsible for deleting channels within a group.

Parameters:

- Group name
- Channel name

Returns:

- Boolean

This file reads the groups.json file and finds the index of the group specified in the parameter. With this index, the channels within that group are searched for the channel with matching name. Once these indexes are known, the channel is spliced from the group's 'channels' object. This is the overwritten to the groups.json file. Returns true when it is overwritten.

/renameGroup

This route will be redirected to the renameGroup.js file. It is responsible for overwriting a groupName .

Parameters:

- New Group name
- Old Group name

Returns:

- Boolean

This file reads the groups.json file and finds the group with the specified group using the old group name. If it is found, the new group name overwrites the old name if it passes validity checks. This renames a specific group and is written back to the groups.json file. Returns true once overwritten, if anything fails, returns false.

/upgradeUser

This route will be redirected to the upgradeUser.js file. It is responsible for upgrading a user to super admin.

Parameters:

- ID

Returns:

- Boolean

This file reads the userData.json file and finds the user with the given index (ID). Once the user is found, their role is overwritten to SA. This is then written back to the userData.json file. Returns true once overwritten, otherwise returns false.

/findInvite

This route will be redirected to the findInvite.js file. It is responsible for finding users to invite to a specific group.

Parameters:

- inviteGroup

Returns:

- newUsers Array

This file will read the groups.json file and find the group with the specified group name (inviteGroup). Once found, the users.json file is read to find the names of all users. If a name does NOT exists in both the users array and the groups.members array, the user is added to the newUsers array. Once all the users are checked, the newUsers array is sent back to the front-end

/sendGroupInvite

This route will be redirected to the sendGroupInvite.js file. It is responsible for added a specific user to a group.

Parameters:

- Username
- Invite Group

Returns:

- Boolean

This file will read the groups.json file and find the group index of the specified group (invite group). Once found, the new user is added the its member's array. This is then rewritten to the groups.json file. Returns true if successful.

/deleteFromGroup

This route will be redirected to the deleteFromGroup.js file. It is responsible for finding users which exist in a group so they can potentially be deleted.

Parameters:

- Group name
- Username of current user in local storage

Returns:

- Array containing the group's members

This file will read the groups.json file and find the group which matches the specified group (group name). Once found, the members are found and if the current user is part of the group, their name is removed from the members array (prevents users from deleting themselves). This array is returned to the front-end.

/sendDeleteFromGroup

This route will be redirected to the sendDeleteGroup.js file. It is responsible for deleting users from a group.

Parameters:

- Username
- Group name

Returns:

- Boolean

This file will read the groups.json file. It will find the group using the groupName. The user index will be found in this group using the specified username. Once the user is found in the group's members array, they will be deleted from the group. Then they are removed from any channels which they may be a part of within this group. Once their name is spliced from arrays in both group and channel, it is written back to the groups.json file. Returns true once overwritten.

/deleteFromChannel

This route redirects to the deleteFromChannel.json file. It is responsible for finding users in a specific channel which can be deleted.

Parameters:

- Channel name
- Group name
- Username (of current user in local storage)

Returns:

- usersArray

This file will read the groups.json file. It will find the specific group using the group name. When found, the channel within this group is found using the channel name. The members within this channel are saved to the userArray unless they are the currently logged in user (username). This array is returned to the front-end.

/sendDeleteFromChannel

This route redirects to the sendDeleteFromChannel.js file. It is responsible for deleting a user from a specific channel.

Parameters:

- Channel name
- Group name
- Username

Returns:

- Boolean

This file will read the groups.json file. It will find the relevant group and channel within said group. The members of this channel are found then the user with the specified username is deleted from this object. This new members object with the user deleted is rewritten to the groups.json file. Returns true once data is overwritten.

Angular Architecture

Currently there are only 4 components which make up this system. These are:

App Component

This standard component of angular architecture handles importing modules to be used throughout the project. The html page has some standard information such as a title and the logout button. These elements are thus accessible from any page within the system.

Login Component

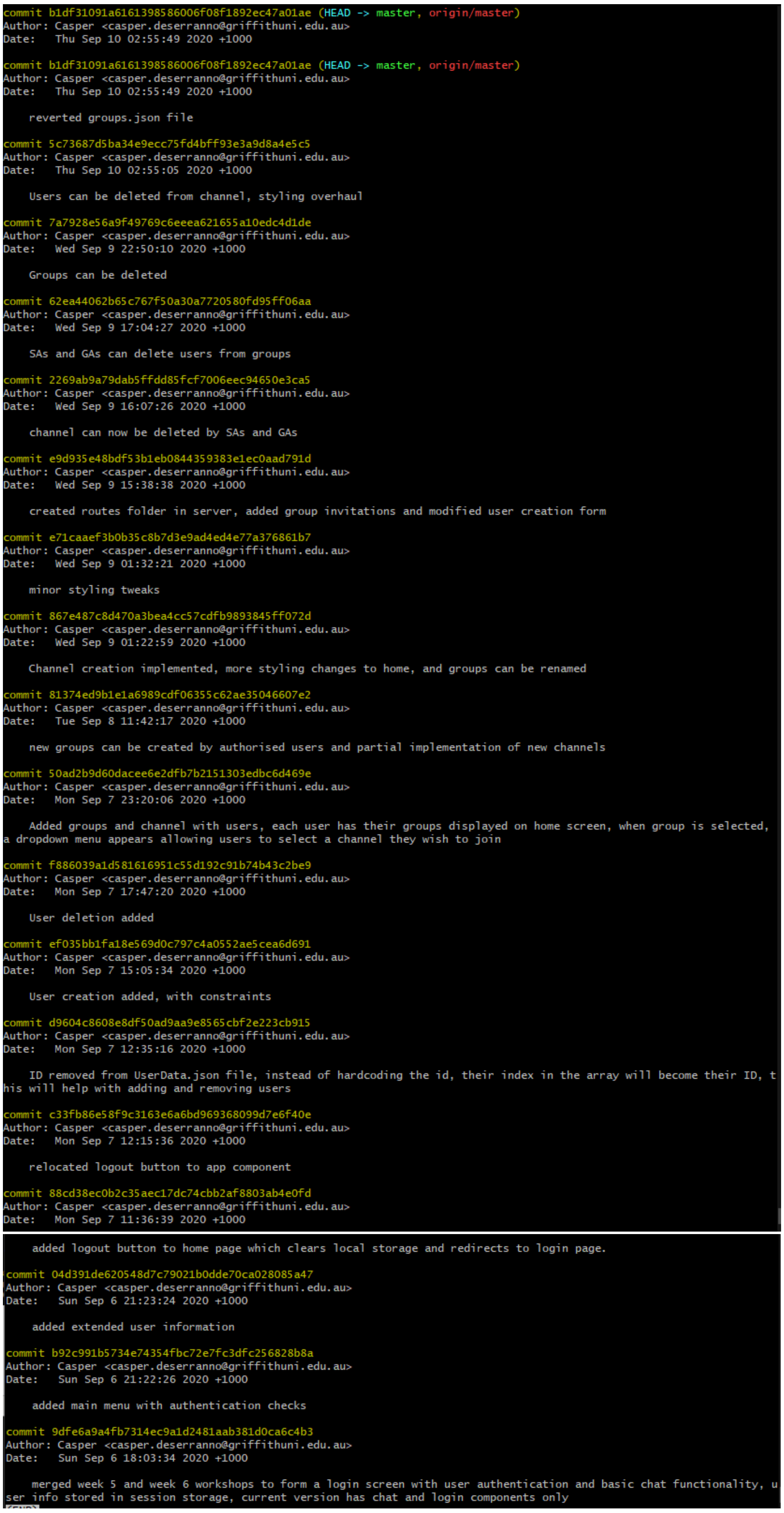
This component is home to the login form and the functions to make it work. The html page is a basic form which uses the information entered to authorise users. If a user is already logged in and tried to access this component, they will be redirected to the home page.

Home Component

This component houses the bulk of the system. This may be subdivided once there are more components to reduce clutter. All the routes mentioned in the REST API part of the documentation, stem from this component. Users can only access this page once they have successfully logged in. This is because the component requires user information to display anything since there is very little static content.

Chat Component

This component was implemented at the start of the project however is not accessible at the current state of the system. The chat system has not been altered to accommodate for multiple channels. This is currently just a placeholder for future development.



```
commit b1df31091a6161398586006f08f1892ec47a01ae (HEAD -> master, origin/master)
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Thu Sep 10 02:55:49 2020 +1000

commit b1df31091a6161398586006f08f1892ec47a01ae (HEAD -> master, origin/master)
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Thu Sep 10 02:55:49 2020 +1000

    reverted groups.json file

commit 5c73687d5ba34e9ecc75fd4bff93e3a9d8a4e5c5
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Thu Sep 10 02:55:05 2020 +1000

    Users can be deleted from channel, styling overhaul

commit 7a7928e56a9f49769c6eeea621655a10edc4d1de
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Wed Sep 9 22:50:10 2020 +1000

    Groups can be deleted

commit 62ea44062b65c767f50a30a7720580fd95ff06aa
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Wed Sep 9 17:04:27 2020 +1000

    SAs and GAs can delete users from groups

commit 2269ab9a79dab5ffdd85fcf7006eec94650e3ca5
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Wed Sep 9 16:07:26 2020 +1000

    channel can now be deleted by SAs and GAs

commit e9d935e48bdf53b1eb0844359383e1ec0aad791d
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Wed Sep 9 15:38:38 2020 +1000

    created routes folder in server, added group invitations and modified user creation form

commit e71caaef3b0b35c8b7d3e9ad4ed4e77a376861b7
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Wed Sep 9 01:32:21 2020 +1000

    minor styling tweaks

commit 867e487c8d470a3bea4cc57cdfb9893845ff072d
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Wed Sep 9 01:22:59 2020 +1000

    Channel creation implemented, more styling changes to home, and groups can be renamed

commit 81374ed9b1e1a6989cdf06355c62ae35046607e2
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Tue Sep 8 11:42:17 2020 +1000

    new groups can be created by authorised users and partial implementation of new channels

commit 50ad2b9d60dacee6e2dfb7b2151303edbc6d469e
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Mon Sep 7 23:20:06 2020 +1000

    Added groups and channel with users, each user has their groups displayed on home screen, when group is selected,
a dropdown menu appears allowing users to select a channel they wish to join

commit f886039a1d581616951c55d192c91b74b43c2be9
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Mon Sep 7 17:47:20 2020 +1000

    User deletion added

commit ef035bb1fa18e569d0c797c4a0552ae5cea6d691
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Mon Sep 7 15:05:34 2020 +1000

    User creation added, with constraints

commit d9604c8608e8df50ad9aa9e8565cbf2e223cb915
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Mon Sep 7 12:35:16 2020 +1000

    ID removed from UserData.json file, instead of hardcoding the id, their index in the array will become their ID, t
his will help with adding and removing users

commit c33fb86e58f9c3163e6a6bd969368099d7e6f40e
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Mon Sep 7 12:15:36 2020 +1000

    relocated logout button to app component

commit 88cd38ec0b2c35aec17dc74cbb2af8803ab4e0fd
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Mon Sep 7 11:36:39 2020 +1000

    added logout button to home page which clears local storage and redirects to login page.

commit 04d391de620548d7c79021b0dde70ca028085a47
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Sun Sep 6 21:23:24 2020 +1000

    added extended user information

commit b92c991b5734e74354fbc72e7fc3dfc256828b8a
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Sun Sep 6 21:22:26 2020 +1000

    added main menu with authentication checks

commit 9dfe6a9a4fb7314ec9a1d2481aab381d0ca6c4b3
Author: Casper <casper.deserranno@griffithuni.edu.au>
Date:   Sun Sep 6 18:03:34 2020 +1000

    merged week 5 and week 6 workshops to form a login screen with user authentication and basic chat functionality, u
ser info stored in session storage, current version has chat and login components only
```

Figure 1 - Git Log