WEB PROGRAMMING 2811ICT Assignment 1

Documentation

Jesse Schneider, S5087288

Github Link: https://github.com/jesse-schneider/MEANChat

Running the Application

Currently, to run there is a requirement of running the frontend and backend separately.

To run the angular project -> run ng serve --open from the root directory of the project.

To run the node.js backend-> run cd server | node server.js from the root directory of the project.

Git Structure

The directory of the git repository follows a standard layout, with the majority of it representing a standard angular project, and the node server and API source code found in /MEANchat/server.

Throughout development, the approach of running 2 separate servers, 1 for angular and 1 for node was favoured, it led to a more accurate testing environment, and was faster to refresh following changes.

Data Structures

Other than simple data types such as strings, integers, arrays, etc, only 1 custom data type exists in the repository - the User Object.

The user object is as follows:

```
{
        Username: string,
        Email: string,
        ofGroupAdminRole: boolean,
        ofGroupAssisRole: boolean,
        groupList: [],
        adminGroupList: [],
        groupChannels: []
}
```

This User object can represent and store everything required when a user is logged in and is passed between both frontend and backend as needed. It is stored in SessionStore while logged in, and is updated following any changes made in the backend. These changes are then also reflected on the frontend.

Angular Architecture

Components:

- Nav
  The navigation bar at the top of every page
- Login
  The login page found at root directory
- Home
  A user's home screen, where the login redirects following success
- Group
  The group homepage, where clicking on a group on home will redirect to
- Channel
  The child component inside group, that reflects the content of the current channel

Services:

- AuthService
- GroupService

Routes:

- /
  The root directory
- Home
  The home page, redirected to following login
- Group
  The group page, redirected to after clicking on a group, contains channel component

• Node server architecture: modules, functions, files, global variables.

Node Server Architecture

The general backend is run by the `server.js` file located in the /server folder. All the routes are broken up into a module each, in their own separate file in /server/routes.

MEANchat/Server
\server.js
/routes
\addchannel.js
\addgroup.js
\adduser.js
\adduserchannel.js
\auth.js
\removechannel.js
\removegroup.js
\removeuser.js
\removeuserchannel.js

Functions:
All routes files contain 1 function per route. In the server.js, the only other functions used are applying middleware, requiring modules, and running the server.

Responsibilities between frontend and REST API

Angular frontend:

The frontend simply displays all GUI elements, and a representation of data that it has retrieved from the backend. No data is changed in the front, if something is needed to change/add/remove, then a request has been sent to the backend.

Node.js Backend:
All data processing has been done in the backend, the frontend simply sends a request to the backend, the backend makes the change, serialises and saves in the filesystem, and returns the new object.

REST API Routes

All server-side endpoints have a request object coming in, and a response object returned.

POST /api/adduser

Request: All fields required to create a user
Response: The created user object
Purpose: Used when a super user or GroupAdmin wishes to add a new user to the system, and they immediately are added into the current group

POST /api/removeuser

Request: Request object containing username of user to be removed
Response: Response object, with string of removal status
Purpose: Used when a super user or GroupAdmin wishes to remove a user from the system

POST /api/addgroup

Request: Request Object containing User Object including new group
Response: : Response object, with string of add status
Purpose: Used when a super user or GroupAdmin wishes to add a new group to the system

POST /api/removegroup

Request: Request object containing name of group to be removed, and the user who did it
Response: Updated User Object Absent the removed group
Purpose: Used when a super user or GroupAdmin wishes to remove a group from the system

POST /api/addchannel

Request: Request Object containing the group, channel to be created and the user who requested it
Response: Updated User Object including the added group
Purpose: Used when a relevant user wishes to add a new channel to the current group

POST /api/removechannel

Request: Request Object containing the group, channel to be removed and the user who requested it
Response: Updated User Object absent the removed group
Purpose: Used when a relevant user wishes to remove a channel from the current group

POST /api/adduserchannel

Request: Request Object containing the group, channel to add the user to and the user to add
Response: Updated User Object including the added channel
Purpose: Used when a relevant user wishes to add a user to the current channel

POST /api/removeuserchannel

Request: Request Object containing the group, channel to remove the user from and the user to remove
Response: Updated User Object absent the removed channel
Purpose: Used when a relevant user wishes to remove a user from the current channel

/api/auth

Request: Request Object containing the username of the user to authenticate
Response: The authenticated User Object, or in the case of the super user, an array of all users
Purpose: Used when the user first loads into the application and wishes to gain access

Client<->Server interactions

Upon a request sent from the client, the server receives this request and processes, according to what route it was posted to. Since there is only one object (User), this object is then updated into the JSON located in the filesystem, and the new object is sent back to the client. The client will now update the 'Authenticated_user' object in the SessionStore with the new object, and, if required, refresh the local client variables to be pointing at the new values of the SessionStore, which will change the angular components if necessary.