Git

The git repository for the assignment is located at <http://github.com/s2802053/3813ict-assignment1>.

The repository contains the assignment1 directory, which contains source code for the angular front end and node.js backend of the application, and the documentation directory which contains the documentation files associated with the project in markdown format.

Branches were not used due to there being a lone contributor to the project. Regular commits were made to the master repository as changes were made to code and features were added and extended. In addition to tracking changes to source code, the git repository was used to synchronise the project over multiple development stations.

Data structures

A screenshot of a social media post

Description automatically generated

Angular

The front end of the application was kept as lightweight as possible. No real model was implemented bar a Response interface for api requests.

A login component handles user authentication. A text input retrieves a username from the user which is validated by the server. If the login credential is valid, the application redirects to the chat-client component.

The chat-client component retrieves and displays view components for an array of groups the logged-in user is a member of. When a group is selected, channels belonging to the group which the user is authorised to view are displayed. When a channel is selected, the message history of the channel is retrieved and displayed. Input options are presented to group and super admins which allows them to perform administerial tasks such as adding and removing users from groups and creating and deleting channels, groups and users.

A service was implemented to handle api access to prevent code duplication and to localise change to a single point in code.

State change

The client sends a http request to the server, for example to the /api/client/createGroup route to create a group, along with several data parameters which are required by the server to perform an action. The route handler on the server validates the parameters sent by the request, parses the chat client data structure from a json file, and calls the relevant method on the chat client object, passing in the data parameters. The method performs its actions and returns a status code to the route handler describing the outcome of the operation (whether it was successful, why it was unsuccessful). If the operation was successful, the route handler stores the updated chat client back in a json storage file and returns a json response to the client. The client evaluates the response to determine whether the intended action was performed successfully and then redraws any affected view components to reflect changes to the model.