Assignment 1

Chat System

Name: Ritchie WIlls, Student number: s2967766

2018

Contents

[Introduction 2](#_Toc523768529)

[Git 2](#_Toc523768530)

[Git Commands: 2](#_Toc523768531)

[Data Structures 2](#_Toc523768532)

[Client and Server 3](#_Toc523768533)

[Npm install: 3](#_Toc523768534)

[Angular Architecture 3](#_Toc523768535)

[Angular commands: 3](#_Toc523768536)

# Introduction

This document will go indeed into the process of creating a dashboard for a chat system and describing the functionary of the Angular client-side and node.js server-side responding between each other with a socket pipe service on the network and additionally reading from data. This dashboard needed to store user objects on the server and share information each time a user did any function with the dashboard of this chat system. The document will also go through how Git an open source software was used and how the development of angular components and models were used in the angular framework working with the node.js server.

# Git

Git is an open source software and webpage that provides a version control system for tracing changes in computer files and coordinating work on those files among multiple people editing the one project. First, I generated a Git repository that contains a readme file at the start of development. Readme is useful to tell the user how to install and run the files in the system within the directory of Git. The Git repository was updated each day by using branching and merging to the origin master head so there would be a backup and to keep all the files organized on the cloud server. On the local user end I used the Git init command to initialize the repository by using terminal commands.

## Git Commands:

* Git branch “Branch Name”
* Git checkout “Branch Name”
* Git add -A
* Git commit -m “updating”
* Git checkout master
* Git merge “Branch Name”
* Git push origin master

Then Git clone command was used which copied the existing repository over from the cloud service file into the local file. From working on the system for a while each day a new commit was added to the repository to keep a backup. By using terminal commands of Git add and Git commit the repository added any altered files running around. During development of the chat system some merge conflicts were found. A conflict happens when pushing two items to the origin master that are the same file at once. These merge conflicts were solved within the Git repository later. Git was used to the last second before submitting the last Git repository.

# Data Structures

The Data structures of the system has the additional group admin and super admin functionary for the user’s purpose when login into their account. A user object is identified by their username and email address but the email address will not be used to send emails to the user on webpage. Using a login function that take in two values of username and email this function checks if the input fields are empty and the input matches with the data on the node.js server if the user has successfully sign in the user’s information is stored into local-storage property and session-storage allowing access to the full webpage network. The session-storage will get cleared when the page session is destroyed when closing the webpage.

When user is logged in they can logout which will clears the local-storage and the session-storage by using a function called Logout that handles the clearing of data from the user. There are two admin types from the user object from the json file. The first is called super admin which can make fresh users by inputting username and email then pressing down on a button which is added to the node.js server and also can delete users too. The functions of create user which takes in two input fields returns a new user and delete users that would find the users identity deleted from the server-side. The other admin type is group which can also create new users but can’t any delete users from service-side but can add socket channels and remove socket channels from the chat system.

# Client and Server

REST API is an internet protocol that allows connection between the angular client and server-side of the node.js which transfer data across backwards and forwards. This was central component so the front end of angular framework can modify json data on the node.js server. Get request functions are key so the system can receive the API data through the URL string. Get requests can additionally do routing so webpage files can be loaded in. Get requests can also send back data to the angular client-side. And the other main function is the post functions are used to send data so it can be display it on the webpage. Npm package of express on the node.js side handles both of these functions for the server. On the client side a static directory called dist is made this will let the node.js server run the angular client on port 3000 or localhost:3000.

## Npm install:

These are the main ones uses:

* Npm init
* Npm install body-parse
* Npm install express
* Npm install socket.io
* Npm install cors

# Angular Architecture

The angular architecture is quite an important part of the development as it’s the client-side of the project that communicates data between the node.js server so the json object data can be updated and read with no problem. The first step was to install npm packages to angular framework then update the client by installing -g @angular/cli to kept everything up to date. After that the main project made within command prompt ng new “Project Name” and routing progress was added to the angular architecture framework so the end-user could navigate along through the webpages with significant ease. It was important to use the Angular port of localhost:4200 to do bug and error checking throughout the development. Importing stuff like Router Module into the app module file is very significant so the project doesn’t show a blank page with many errors while on the URL port.

## Angular commands:

* Ng add “Name” – routing
* Ng Build
* Ng Serve
* Ng g c “Component Name”
* Ng g service “Service Name”

The app components in the angular framework are necessary to be produced for this project so the user can navigate but also do functions within the webpage itself. The two main app components that were built in development were the home and 404 not-found component these are important so you can check if the client-side is working and likewise checking if the webpage exist. The other app components that are in the framework are chat, account, login and menu component which are also central. They were made by using the command prompt by inputting ng generate component “Component-Name” for each individual component for angular.

The second most key item of the framework was putting in the services on the angular client-side. The two that were required for functionary with json was the user and socket services having the ability to connect to the node.js server to grab data between each other. These services allow the flow of data and more options for the end-user on the webpage. The socket service is the only way for the user use the chat component and demonstrate that they are online within the socket but also permitting the group admin to add and remove users from the online socket service.

Angular Models were used during the development and one of them is Data binding. Data binding is an important part of displaying values to the HTML with stuff like ngif and ngfor. Which can be automatically update if the user inputs different values to the data bind. The layout design for each page was designed to be clear and not have to many items on the screen at once as it was very important so it would be made super user-friendly so anyone can use this webpage and not get stuck on anything. Ng build was used multiple times to check for faults with the system or any bugs coursing crashes on the client-side but also have the angular system built and ready to use for the node.js server.