Assignment Phase 1

Individual Assignment

Due: 8 am Thursday 5 Sep 2024

Introduction

You are required to build the server and front end for a text/video chat system.

The chat system will allow users to communicate with each other in real-time within different groups and channels. There will be three levels of permissions on the site.

1. Super Admin
2. Group Admin
3. User

The solution must be implemented using the MEAN stack (MongoDB, Express, Angular, Node) along with sockets.io and Peer.js.

Documentation

Documentation of your implementation is required. You will need to provide the following:

- Describe the organization of your Git repository and how you used it during the development of your solution (branching, update frequency, server/frontend etc.)
- Description of data structures used in both the client and server sides to represent the various entities, e.g.: users, groups, channels, etc.
- Angular architecture: components, services, models, routes.
- Node server architecture: modules, functions, files, global variables.
- A list of server side routes, parameters, return values, and there purpose
- Describe the details of the interaction between client and server by indicating how the data on server side will be changed and how the display of each angular component page will be updated.

Git (Source code versioning)

Git must be used during the development of the chat system. We recommend that you use GitHub and share the repository with your marker. You will be marked on frequent updates to the repository and the usage of git features.

Requirements

Group

- A collection of chat users that have been given permission from a group admin or super admin to be able to be a member of a group.
- Multiple groups can exist.
- Multiple admins per group can exists.
- Each chat user can exist in multiple groups.
- A group admin can administer more than 1 group.
- The super admin approves upgrades(promotion) of users to group admins.
- A group admin can only administer a group they created.
- Super admin has access to all groups in the event a group admin is deleted.

Channel

Each group will have access to a number of channels for the purpose of chatting.

Once a user is a member of a group they can select any existing channel in that group to chat in.

Users

The model of a chat user will require a minimum of:

username, email, id, roles[ ], groups[ ]

Super Administrator

- A Super Admin can promote a chat user to a Group Admin role.
- A Super Admin can remove any chat users.
- A Super Admin can upgrade a chat user to Super Admin role.
- A Super Admin has all of the functions of a group administrator

Group Administrator

- A Group Admin can create groups.
- A Group Admin will create channels (subgroups) within groups.
- A Group Admin can remove groups, channels, and chat users from groups they administer. • A group admin can delete a chat user (from a group they administer)
- A group admin can only modify/delete a group that they created.
- A group admin can ban a user from a channel and report to super admins.

Chat User

- A user of the system can create a new chat user. (Usernames are unique)
- A chat user can join any channel in a group once they are members of a group.
- A chat user can register an interest in a group, to be added by the group admin.
- A chat user can leave a group or groups they belong to.

- A chat user can delete themselves
- A chat user is uniquely identified by their Username
- A chat user may logout.

User Authentication

Initially there is one user called 'super' with a password of “123” who is also a Super Admin.

You should add support for chat users entering a username/password. If the username/password does not match it should ask the user to login again. (Simple username/password authentication is not best practice but will suffice for the purposes of this subject).

A User should not be able to access features of the site without being authenticated to some role and should only see features that their role permits.

Once a user is authenticated the page should display the groups that the chat users have been added to. Once a group and channel combo have been selected the user should be able to begin chatting with other users in that channel.

For Super Admin, Group Admin the page should also display an appropriate UI according to their roles abilities for adding / modifying /deleting users, groups and channels, respectively.

Data storage

Browser(client) based local storage can be used to store data structures until we introduce MongoDB for the second phase of the assignment.

Submitting your assignment

For assignment 1 your will submit the documentation along with zips of your front end angular and server side source code. Include your snumber in the name of your files.

What to submit

1. Submission is via a github repository. Your tutor should be given access to the private repository. (Remember to exclude any “node_modules” directories from being added to your git repository.)
2. Documentation should be in a file named README.md in your github repository. This should be authored using markdown syntax.
3. A copy of the text in the README.md file should also be submitted to canvas as a word document

Further Information for Assignment 2. (A heads up)

The MongoDB, Sockets, Image and Video support are not implemented in assignment phase 1.

MongoDB

The Node.js server storage will use a MongoDB database. The mongo database should store all of the user, group, and channel data as well as chat history.

Sockets

Socket.io will be used to support bi-directional communication with the front end app and the server for real-time data (chat etc.)

Image support

The chat system should allow users to specify a profile image (i.e. avatar). The profile image should be displayed in the chat history alongside their username for messages that they posted. The chat system should also support sending images in a chat message and will appear to all users viewing the chat. Image storage on the server can be as files in a specified directory, with the path to the file stored in the mongo database.

Video support

The chat system should allow users to video chat. PeerJS can be used in Angular side and Peer server can be implemented on your sever side. This requires an encrypted connection (SSL). For demonstrating the function of video chat, you can use the elf server to host your server. (https://elf.ict.griffith.edu.au)

Channel History

On joining a channel the chat user should see some past history (messages) from the channel. (The volume is up to you (min past 5 messages)).

New messages are broadcast to all users currently viewing the channel and added to the history