Documentation

1. Git implementation and Organization:

During the development phase of any project, the regular backup and versioning of the system's files is very important. In order to perform a continuous backup and version our project, we used GitHub as our remote repository. We versioned the changes and committed it to our Git repos so that we can revert back to the desired version if something goes wrong or the new version of the system crashes. The following is a link to my git repository

Git Link: https://github.com/Samiben1/Angular-A2-.git

The repo consists of two main files, Chat and Chat-system. The chat file is the dashboard system which was created previously during the assignment progression. It consists of all the functionalities of the assignment 1.

The chat file consists of angular front-end and a server file with a backend files using node. The angular part consists of two main components i.e. login component and the profile component. The login component handles all the user authentication and approval to the profile page. The profile is the main dashboard system where we can see the users details, create users/groups/channels and so on.

Similarly, the chat-component file consists of user and chat components in the angular section. The server side (node part) consists of the code to connect to the database and other authentication.

I used git to version and create a virtual backup of all my files and the changes I have made to that file so that even if I loose my file from the local repo, I can easily retrieve it through my git repo.

2. Data structure user in the system:

Data structure is an organization of data for management and storage making it efficient to access and modify the data. To be more precise, a data structure is a collection of data values, relationship amongst them and the functionalities that can be applied to those data.

My code file consists of a client side and a server side code. Angular handles all the front-end and node handles all the backend parts. In the backend part, the entities like users, groups and channels are a group of arrays that have various attributes. In the angular section, all the data are structured in an array format and passed on to the server through http requests. In the server side however, different models are used to structure datatypes and query and store the incoming data. The user contains attribute like username, age, birth date and password which are stored as string. The rest like groups associated with the specific user are stored as array. Similarly, the user class also stores whether he/she is an admin as a Boolean value.

The groups also stores values in arrays within which contains strings. It stores an array of admins of that group, channels specific to that group and all the information to the channels. It also saves the creator of that group as a string along with a list of users.

Similarly, for the chat-component system all the values of the user and chats are stored as a string in the database. All the data's are retrieve in a Json file format and queried to respective attributes.

3. A list of routes, values returned and parameters:

The following are all the routes used for the chat system. Similarly, the values they transfer and their functionalities are mentioned as well.

Routes:

```
router.post('/authenticate', user_controller.authenticate);
router.post('/register', user_controller.register);
router.get('/', user_controller.getAll);
router.get('/current', user_controller.getCurrent);
router.get('/:id', user_controller.getById);
router.put('/:id', user_controller.update);
router.delete('/:id', user_controller.delete);
```

The above routes are for user creation and authentication. When the authenticate route is called it queries the database and retrieves the data from a particular user.

e.g. {

username: "super",

password: "super",

…

}

It then checks the login form data and the json data from database and authenticates and directs user to the particular page. The user is asked to log in again if the data does not match.

The register route takes in all the parameters defined in the users module. It then creates a new user of that name in the Users collection of the database.

e.g.

{

username: "super",

password: "super",

firstName: "super",

lastName: "super"

}

Similarly other respective routes are for updating users information and deleting user based on the users id. They take in the parameter of username that matches in the mongo database.

```
router.get('/:room', chat_controller.getChat);

/* SAVE CHAT */
router.post('/', chat_controller.saveChat);
```

The above image shows the routes for retrieving the rooms registered in the database and the messages sent in that room. Similarly, the post route posts the messages sent by the user and adds to the database of the particular room. The route for room takes in the parameter for just room name, hut chat however takes in parameters like

{
    Username: string,
    Message: string
}

And posts it to the database.

4. Angular architecture:

Angular is a front end JavaScript frame work for creating a single page application. For the purpose of developing the chat system application and the dashboard, we have implemented Angular as our main web frame.

For this project, Angular side of the system consists of two components namely login and chat. These components perform all the authentication and chat system works with the help of few helper, modules and services files.

The module file consists of User and chat classes that export the type of values that classes accept. It consists of the information needed for user like id, name, username, password etc and determine data type each attributes are expected to contain.

e.g.

```
export class Chat{
        room: string;
        nickname: string;
        message: string;
}
```

Services provide the functions for authenticating user login, sending and retrieving messages and creating and deleting/updating users.

e.g.

```
export class Chatservice{
        getChatByRoom(room){
                return this.http.get(‘${environment.apiUrl}/chat/’+ room);
        }
}
```

Taking about the helpers file, it contains all files that performs error inspection and adds jwt code to encrypt the password.

The chat component interact with the system using sockets that passes on the values through http requests.

5. RESTful Apis:

REST Apis are conventions used for creation of http services so that workload can be divided between the server and the client. As the client and server of any website respond to each other requests trough http requests, The rest api makes it easy for the client side to interact with the server functions through the creation of static http requests created through this api.

In these processes the Restful api are included amongst which are as follows:

Post: http://localhost:1234/Users/register: this api is used to register users in the database.

Similarly, others include :

Post: http://localhost:1234/Users/authenticate
get: http://localhost:1234/Users/
get: http://localhost:1234/Users/current
get: http://localhost:1234/Users/:id