CONTENTS

Git

The Git repository for this project is composed of two branches: main and dev. All development is done on the ‘dev’ branch and when a new feature is implemented, dev is merged into main.

Data Structures

User

Description

Members

- username: string
- email: string
- role: number

Message

Description A class with two properties: the sender of the message, and the content of the message

Members

- sender: string
- content: string

Channel

Description

Members

- name: string
- users: Array<User>
- messages: Array<Message>

Group

Description

Members

- name: string
- users: Array<User>
- channels: Array<Channel>

Angular Architecture

Components

LoginComponent

Route	/login
Description	
Methods	- login()

ChatComponent

Route	/chat
Description	
Methods	- open_group() - open_channel()

ChatwindowComponent

Route	/chat/chatwindow
Description	The page where the actual chat functionality exists. Displayed through a router outlet in ChatComponent.
Methods	- send_message()

GroupsettingsComponent

Route	/chat/groupsettings
Description	A page where users with elevated permissions can edit the properties of a group. Displayed through a router outlet in ChatComponent.
Methods	- send_message()

SettingsComponent

Route	/settings
Description	
Methods	

AccountComponent

Route	/settings/account
Description	
Methods	- set_username()

PreferencesComponent

Route	/settings/preferences
Description	
Methods	- set_theme()

PreferencesComponent

Route	/settings/preferences
Description	
Methods	- set_theme()

Services

SocketService

Description	SocketService provides socket functionality to other components of the application.
Methods	- listen() - join_channel() - emit()

GroupService

Description	GroupService provides a Subject object for both the current group and the current channel the user has open, along with methods to update those Subjects. It also contains the definitions of the Message, Channel and Group classes.
Methods	- set_current_group() - set_current_channel()

UserService

Description	Userservice handles user requests to the server, as well as providing the definition for the User class. All methods defined in UserService simply call SocketService to emit an event of the same name, passing the data passed as a parameter to the method as the data to be passed through sockets.
Methods	- create_user() - delete_user() - set_role() - create_group() - delete_group() - add_user_to_group() - remove_user_from_group() - create_channel() - delete_channel() - add_user_to_channel() - remove_user_from_channel()

ThemeService

Description	ThemeService provides a Subject object for components (namely AppComponent and PreferencesComponent) to access and update the current theme
Methods	- set_theme()

Node Server Architecture

Modules

fakeDB.js

Description	Routes are split into two files: user_routes.js and group_routes.js which contain routes related to user data and group data respectively.
Methods	- get_groups_of_user() - get_users_of_group() - get_channels_of_group() - get_group() - create_user() - delete_user() - set_role() - create_group() - delete_group() - add_user_to_group() - remove_user_from_group() - create_channel() - delete_channel() - add_user_to_channel() - remove_user_from_channel()

routes.js

Description	Routes are split into two files: user_routes.js and group_routes.js which contain routes related to user data and group data respectively.
Methods	A route description for each API endpoint described in the REST API section of this document.

sockets.js

Description	Sockets.js has two purposes: 1. Execute functions defined in fakeDB.js upon receiving a request through sockets. 2. Emit data sent through a channel to all clients connected to that channel.
Methods	Various calls to functions defined in fakeDB.js

REST API

All API routes are GET requests except for /api/auth, which is POST. Any time data needs to be changed on the server, sockets are used for the request. After the request is completed, any data changed by the request is then emitted under an event named after the data. For example, if a client requests to delete a user, the server will delete the user and then emit an event called “users” which contains the new list of users.

/api/auth

Description	Return User object on successful validation of credentials.
Method	POST
Return	User

/api/groups/names

Description	Return an array of all group names.
Return	Array<string>

/api/groups/:group_name

Description	Return the group with name “group_name”
Return	Group

/api/groups/:group_name/users

Description	Return an array of users who are a member of “group_name”.
Return	Array<User>

/api/groups/:group_name/channels

Description	Return an array of channels in “group_name”.
Return	Array<Channel>

/api/groups/:group_name/channels/:channel_name

Description	Return the channel “channel_name” from the group “group_name”.
Return	Channel

/api/users

Description	Return an array of all users.
Return	Array<User>

/api/users/usernames

Description	Return an array of all usernames.
Return	Array<string>

/api/users/:username/groups

Description	Return an array of group names which “username” is a member of.
Return	Array<Group>