CS4530 Final Project: “Individualized Group Chats”

“Group 2B”: Raj Patel, Sunny Gu, Jared Bingham and Cam Gleichauf

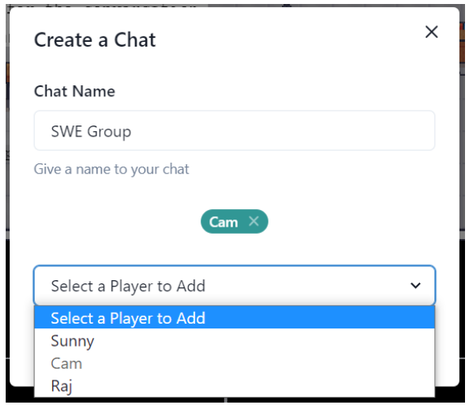
Our Feature: Expanded Chat Functionality

In the original Covey.Town release, users may connect to a ‘Town’ and converse with other users through conversation areas created within the town. While this allows for meeting within the town, which is an intended function of this software, this is a bare bones method of communication. What if a user wanted to be able to privately message someone else directly or some subgroup of the users in the town? What if a user had a bad experience with another user and wanted to block them? What if they changed their mind and wanted to unblock them? If a group of people wanted to converse in a specific chat regarding a given topic, how can they do that?

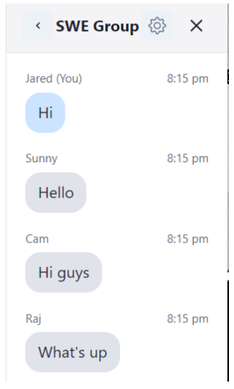
We decided to address the above questions by adding a chat function allowing users to message one another privately even if they are not physically present in a conversation area. Users may also create group chats and add or remove other players into their group chat. These group chats may have a title that specifies what the topic of conversation is about in addition to there being a chat ‘author’ for each chat. The chat ‘author’ is either the creator of the chat, or the longest tenured member of the chat if the author has left the chat after its creation. Furthermore, players can block each other if they so desire and unblock each other if they change their mind.

Demo and Source

Our demo site is available at: https://6265f3191aae251acc38b4aa--elegant-liger-2bdf78.netlify.app/ and our code at https://github.com/neu-cs4530-s22/team-project-2b.



Any user can create a new chat and add players currently in the town into the chat using this menu.



Users can send and receive messages to everyone within the group chat

Our Technology Stack and Design

We implemented the additional chat features on top of the existing covey.town code. Every new chat that is created is represented as a “chat object” and stored/tracked in the backend via the CoveyTownController. The name of the chat, author of the chat, and members of the chat are stored within the chat object, and a chat is created by a user on the frontend. Chats can be created and changed through different React/Chakra modals, and users can navigate through their chats using the new and improved chat menu in the Reach UI. This chat menu shows users chats which they are currently a part of (kept updated via React hooks), allows them to create new chats, and access/edit existing chats. Messages sent within chats are distributed to the appropriate users via socket.io, utilizing the fact that players already have their own unique socket.

This feature comes with a thorough test suite for the new components on both the frontend and the backend, and is deployed through Heroku and Netlify.

Future Work

With this increased capabilities of separate chats and group chats amongst players, we are currently only limited to text messages amongst players within the CoveyTown. A big future idea that could provide more capabilities to how players are able to communicate would include different types of multimedia, such as video messages, files, images, and even voice memos. Just like those that can be found on Facebook messenger or on IMessage with Apple. Instead of needing to resort to external applications or websites to send these multimedia messages, players would be able to seamlessly send these types of messages to as many people as they need within the conversation chat. This could help a lot of people such as athletes, business people, programmers, and just your typical student. All types of messages will be contained within the given chat and players can scroll through and access it all as long as they remain in the CoveyTown server.

In implementing this feature, we would approach this a little differently than how we programmed the messages. With text messages, we get the message from the user and then just display it to the chat, so there is no storing of each string in the server in any way. However, with multimedia messages, there needs to be a way we store the messages as each requires memory and isn’t as simple as text messages. This would most likely need to communicate with urls and the server in order to access the images, files, etc. We could also go the route of creating a separate interface for Multimedia and have different classes for the type of message so one for files, one for images, and another for voice memos.