Websockets in the app

The app uses websockets in order to live stream parameters and chat messages to the user.

Live parameters:

When the user opens up patient details for a patient connected to a device it will open a websocket with the backend. There is currently no authorization required to open this socket from the backend supplied to us by infiniwell, which is something we suggest adding in the future. When the socket is opened, a message is sent from the client to the server over the socket. The message is a json object containing type, opcode, organization and device_id. Organization is currently hard-coded to “ntnu”, something we noticed had no effect on the response. The opcode is the most important parameter and is set to "stream_parameters” which tells the backend to only send parameters and not waveforms as the app does not support ekg waveforms and the amount of data used for ekg is quite significant. When a new websocket message is received by the client its data is subsequently splitted into news2 and parameters. If these are valid json objects the frontend monitor is updated with the new data. There is a status bar at the top of the monitor indicating whether the numbers are currently updating live. If no new data is received 10 seconds after the last datapoint, the data is considered stale and the status will change to disconnected. In order to reconnect, the user must reopen the view; or if the user simply receives a new message from the server it will return to connected.

Live chat:

The chat uses two websockets for messages, one for the chat overview and one for when the user is in a chat. The reason for why we decided to use separate sockets for these two is because we wanted the chat to be seperate from the chat overview to allow our users to access it from different tabs.

The first socket is located in the MessagesTab.js file and updates the latest messages for chat headers in the messages tab. The first socket is opened when the user enters the messages tab and closed when the user leaves the tab. It is worth noting that the first socket remains open when the user enters a chat, so the user can have both sockets open at once.

The second socket is opened when the user opens a specific chat. When the second socket receives a message, it checks if the message is for the current chat and if it is the message is appended to the bottom of the messages. The second socket is also used to send messages. When a message is sent it will appear in the chat when the user receives the same message back from the socket, effectively confirming that the message was indeed sent. Users on the receiving end will also get a push message regarding the message. The second socket is opened when a chat is opened and closed when the user leaves the chat.