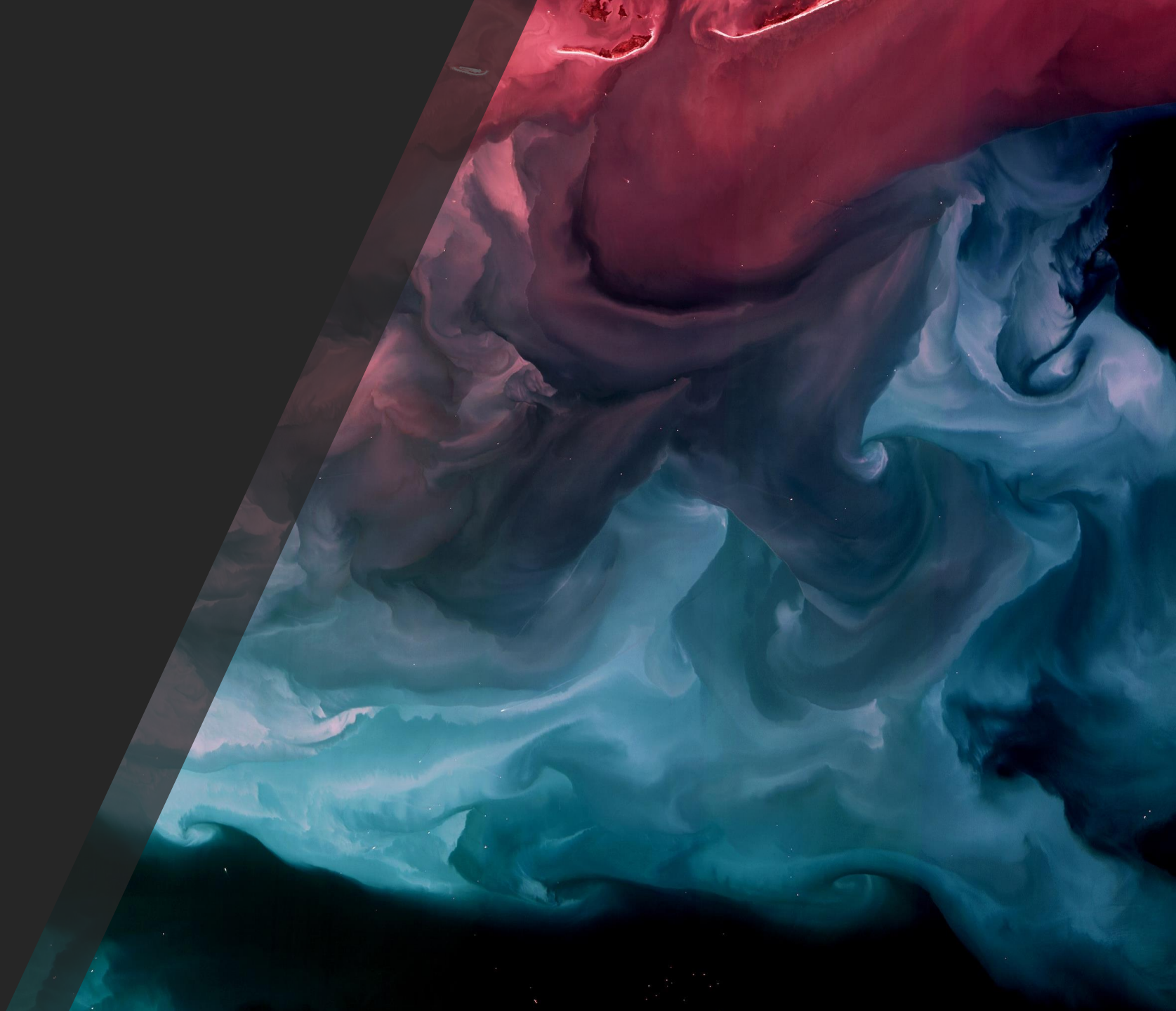


Chatap

A simple chat app

- Ali Nehme
- Gregory Barré
- Alina Gotcherian



Create channels around topics. Chat with others in real time.

- A simple way to chat with friends around topics
- Complex UIs can contribute to visual clutter and hinder accessibility
- Extra bells and whistles? Who the hell needs 'em?



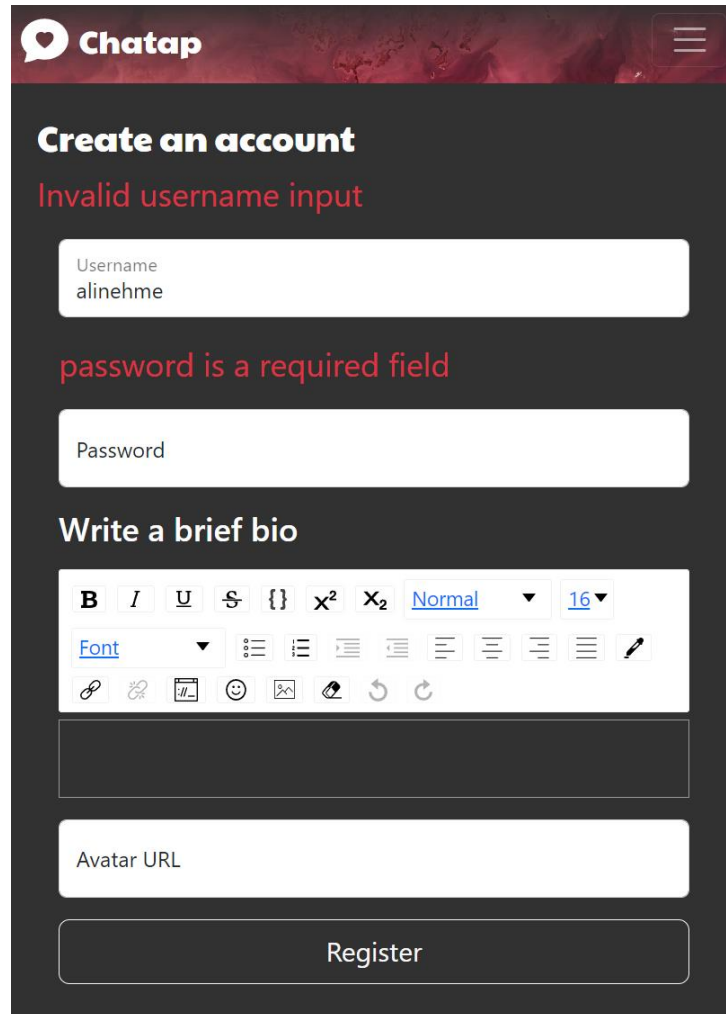
Inspiration

The screenshot shows a Slack interface for the 'Le Wagon - Alumni' channel. The left sidebar lists various channels, with '# montreal' highlighted. The main area displays a message from Ines Alvergne (@channel) dated Wednesday, September 14th. The message is a friendly reminder about a 5-year anniversary party on Saturday, September 24 at 6 PM. It includes a link to an Eventbrite ticket and mentions that the organizers are finalizing details. The message has 4 reactions and 2 replies. The bottom of the screen shows the message input area with a green border.

The screenshot shows a Discord interface for the 'Reactiflux' server. The left sidebar lists various channels, with '# help-react' highlighted. The main area displays a conversation in the '# help-react' channel. The messages are from Arankar XIII, mubashir, and kch, discussing React.js concepts like useEffect and useState. The bottom of the screen shows the message input area with a green border.

Registration & Profile

Frontend & backend
validation



Chatap

Create an account

Invalid username input

Username
alinehme

password is a required field

Password

Write a brief bio

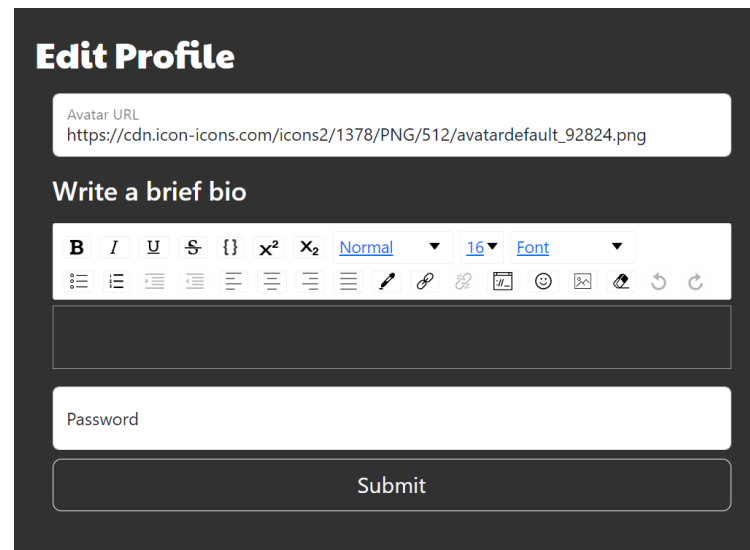
B *I* U ~~S~~ {} x² x₂ Normal 16 Font

Font

Avatar URL

Register

Backend: Password validation



Edit Profile

Avatar URL
https://cdn.icon-icons.com/icons2/1378/PNG/512/avatardefault_92824.png

Write a brief bio

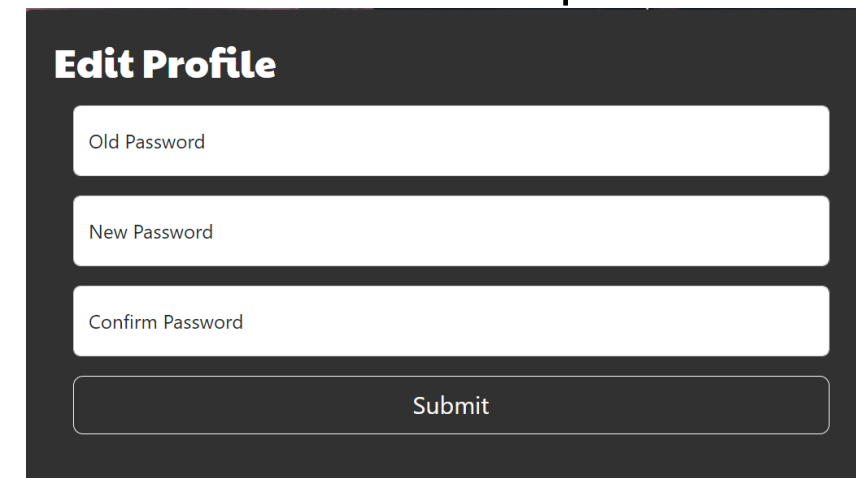
B *I* U ~~S~~ {} x² x₂ Normal 16 Font

Font

Password

Submit

Frontend: Confirm password
Backend: valid old password



Edit Profile

Old Password

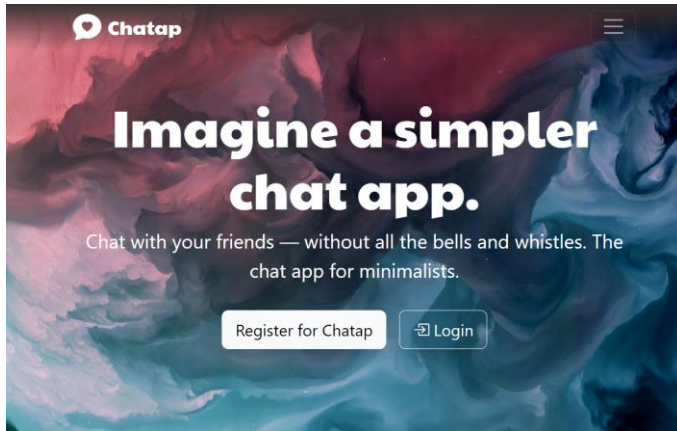
New Password

Confirm Password

Submit

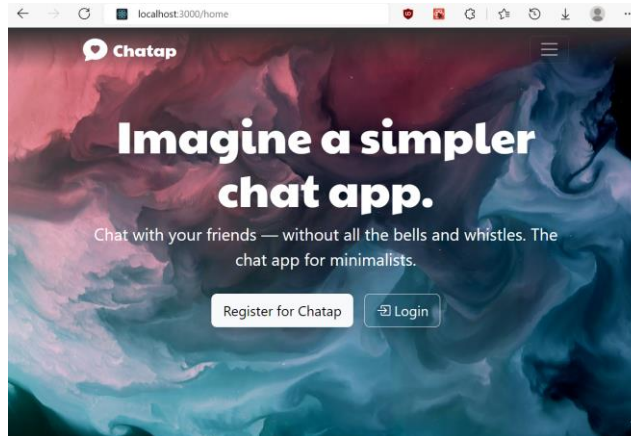
Authorization

Frontend: links



```
{!!authState && authState.roles === "admin" ?  
<>  
  <Nav.Link as={Link} to="/admin/users" >Users</Nav.Link>  
  <Nav.Link as={Link} to="/admin/channels" className="me-3">Channels</Nav.Link>  
</>  
  :  
  ""  
}
```

Frontend: Protected Routes



```
import React from "react";  
import { Navigate } from "react-router-dom";  
import { Outlet } from "react-router-dom";  
  
function ProtectedRoute({ isAllowed, redirectPath = "/", children }) {  
  if (!isAllowed) {  
    return <Navigate to={redirectPath} replace />;  
  }  
  
  return children ? children : <Outlet />;  
}  
  
export default ProtectedRoute;
```

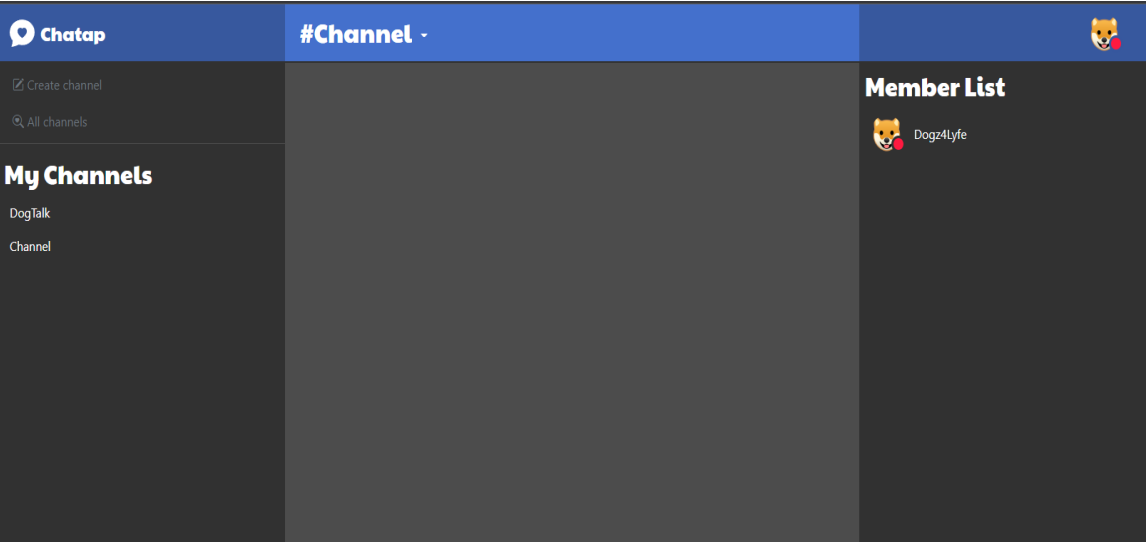
Backend: JWT token

```
const { verify } = require("jsonwebtoken");  
  
const validateToken = (req, res, next) => {  
  const accessToken = req.header("accessToken");  
  if (!accessToken) {  
    return res.status(401).send({  
      message: "user not logged in!",  
    });  
  }  
  try {  
    const decoded = verify(accessToken, "ljskffgoSAFKBISDHF", {  
      maxAge: "2 days",  
    });  
    if (decoded) {  
      req.user = decoded;  
      return next();  
    }  
  } catch (err) {  
    return res.status(401).send({  
      message: "You are not authorized",  
    });  
  }  
};  
  
module.exports = { validateToken };
```

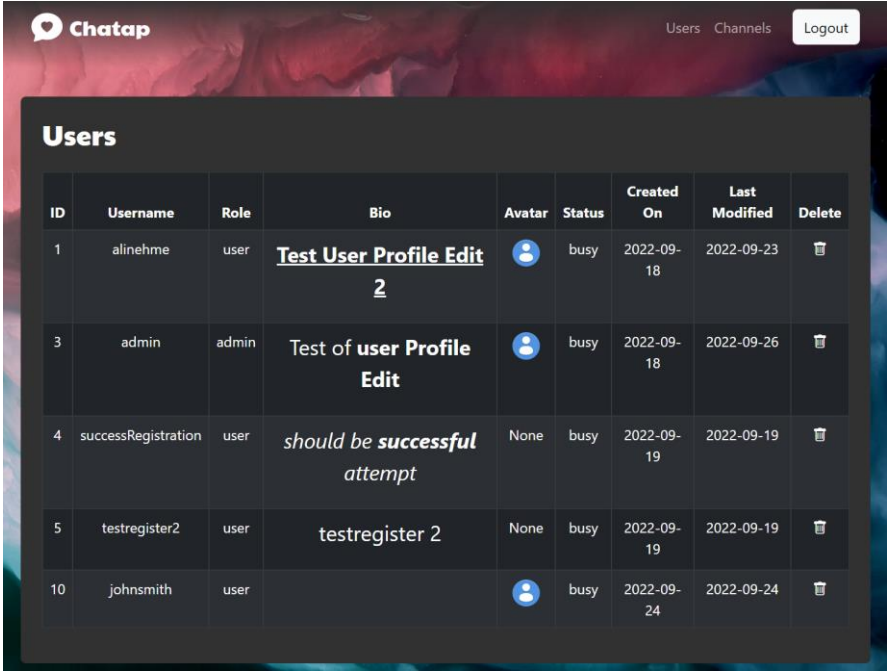
Except login and registration, all actions require backend authorization by JWT

Roles

Users

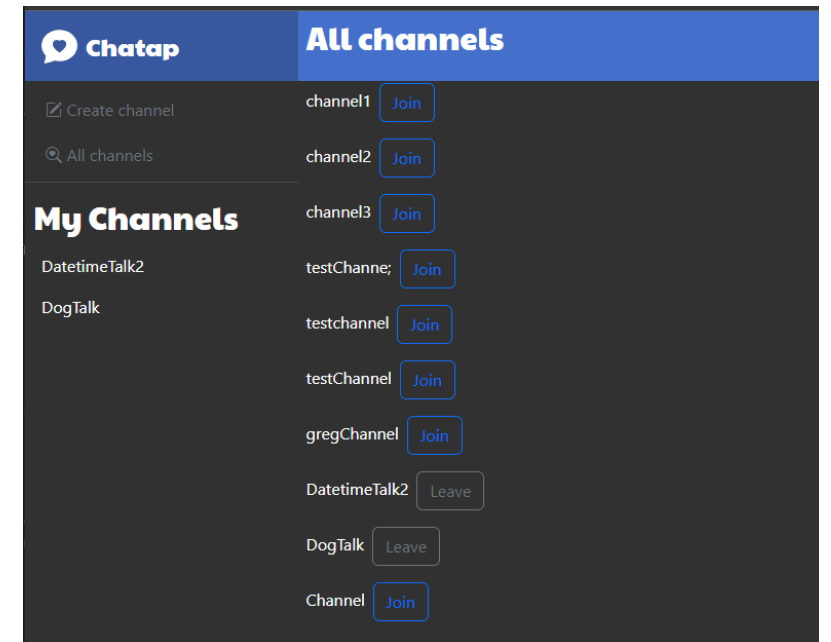
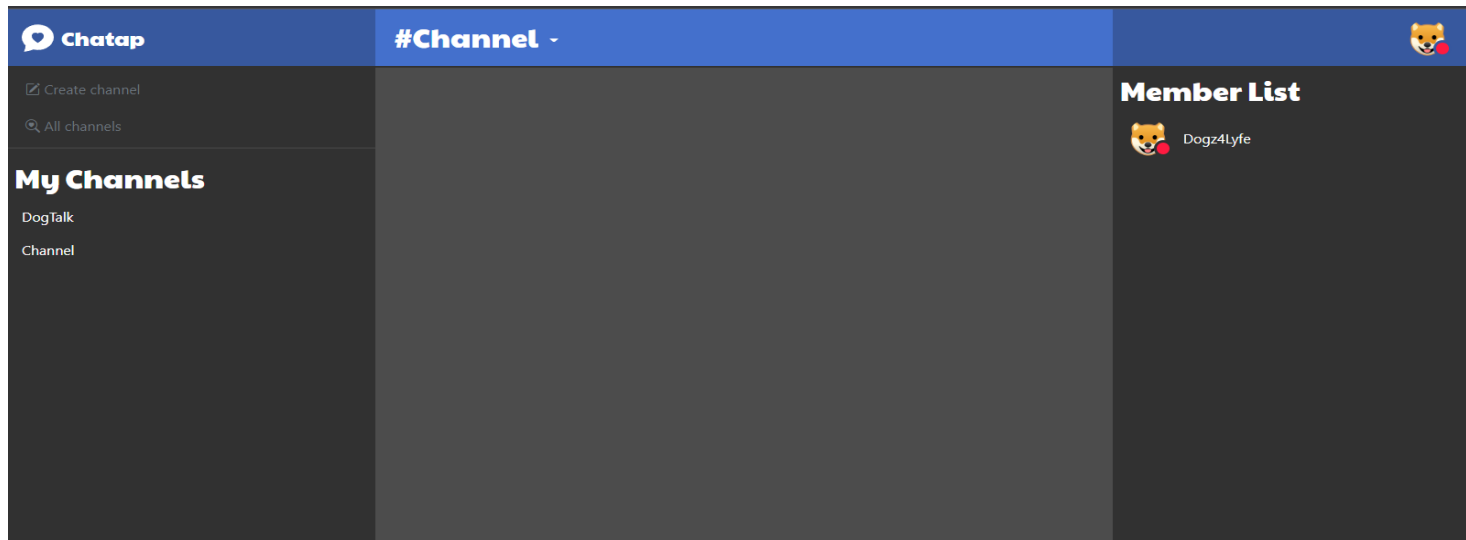


Admin





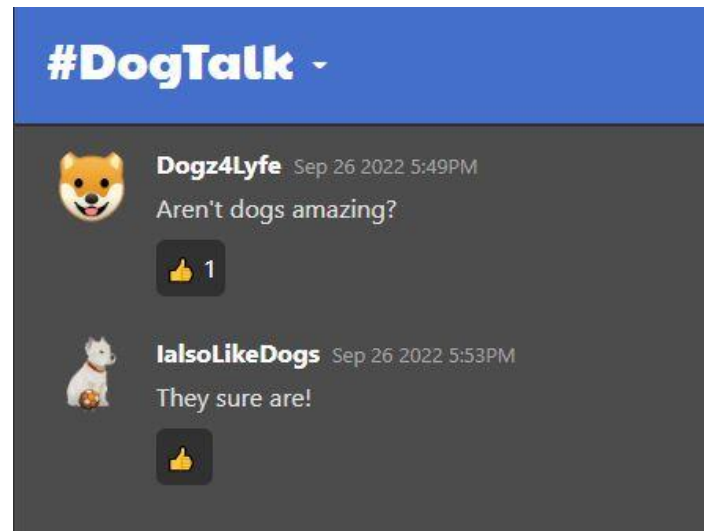
Solution Overview: Channels

- Users interact with each other in private channels
- Users can create a new channel and become owner of that channel
- Users can view existing channels and join them



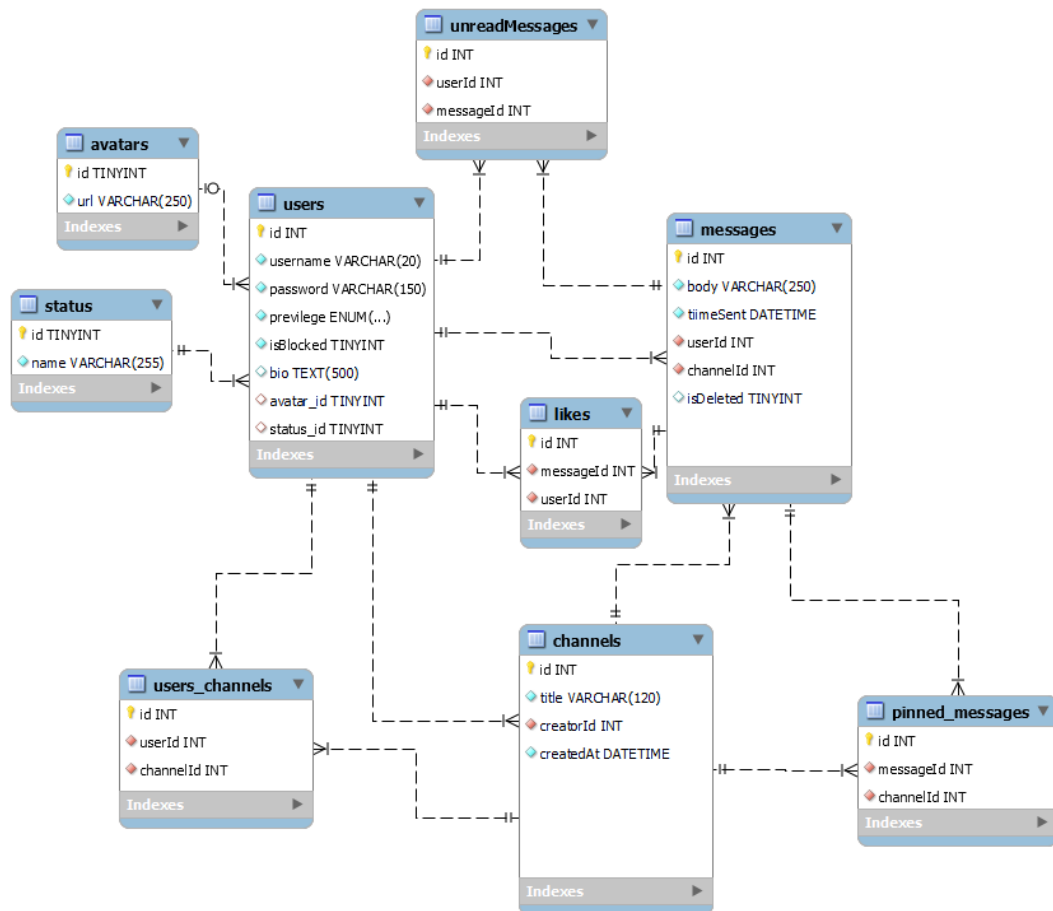
Solution Overview: Messaging

- Users post messages in real-time to all users in the channel
- An indicator shows when another user is typing 
- Users can react to each other's messages by "liking" them 
- Users can delete their own messages

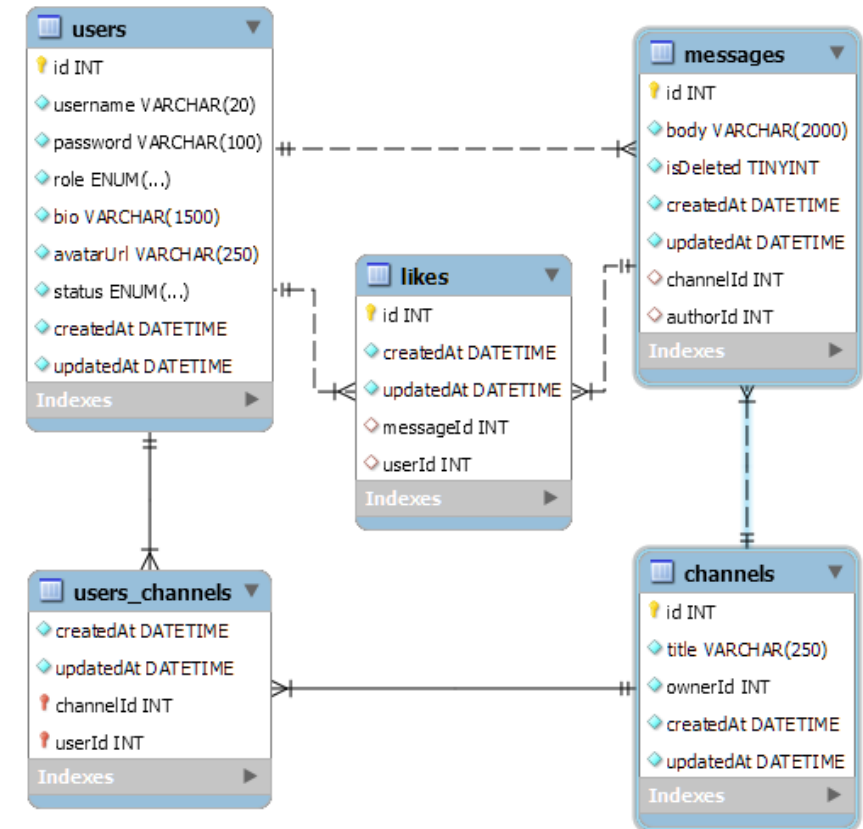


Challenges and Solutions: (D)evolving Database

Original Design

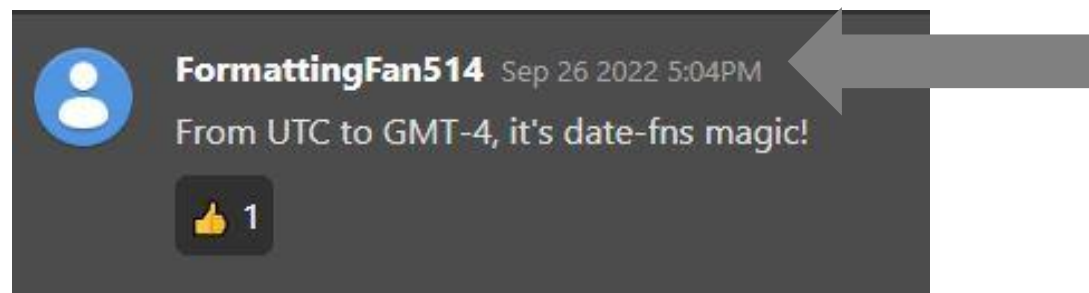


Final Design



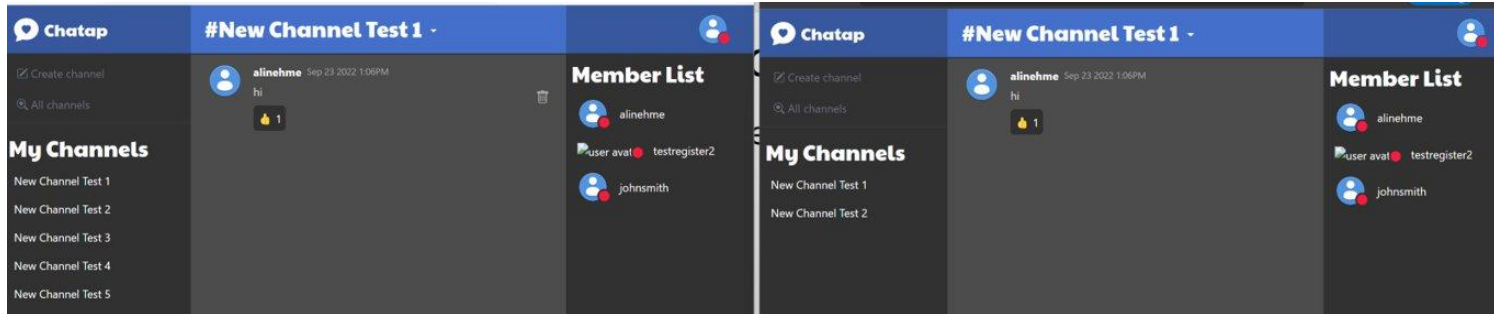
Challenges and Solutions: Date Formats

- Sequelize automatically creates **createdAt** and **updatedAt** columns
- Retrieved from DB in UTC format string ex: **2022-09-26T21:04:15Z** (↙) **date-fns**
- JS date utility library
- **2022-09-26T21:04:15Z** becomes...



Challenges and Solutions: Socket.IO

Real-time delivery of information to other users in the channels



Independent from, but works with, React/Node.js

Sender

```
const sendMessage = () => {
  socket.emit("stopTyping", activeChannel);
  const data = { body: newMessage, channelId: activeChannel };
  if (data.body !== "" && data.body.length <= 2000) {
    axios
      .post(`${process.env.REACT_APP_SERVER_URL}/messages`, data, {
        headers: { accessToken: localStorage.getItem("accessToken") },
      })
      .then((response) => {
        setListOfMessages(response.data);
        socket.emit("send_message", response.data);
      })
      .catch((error) => {
        if (error.response) {
          //
        }
      });
  }
  setNewMessage("");
};
```

backend

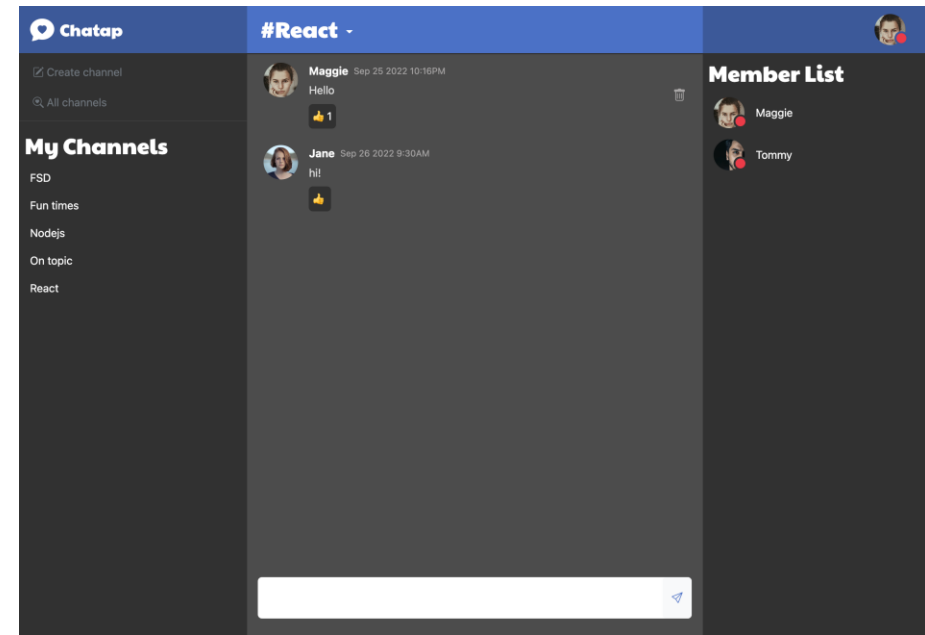
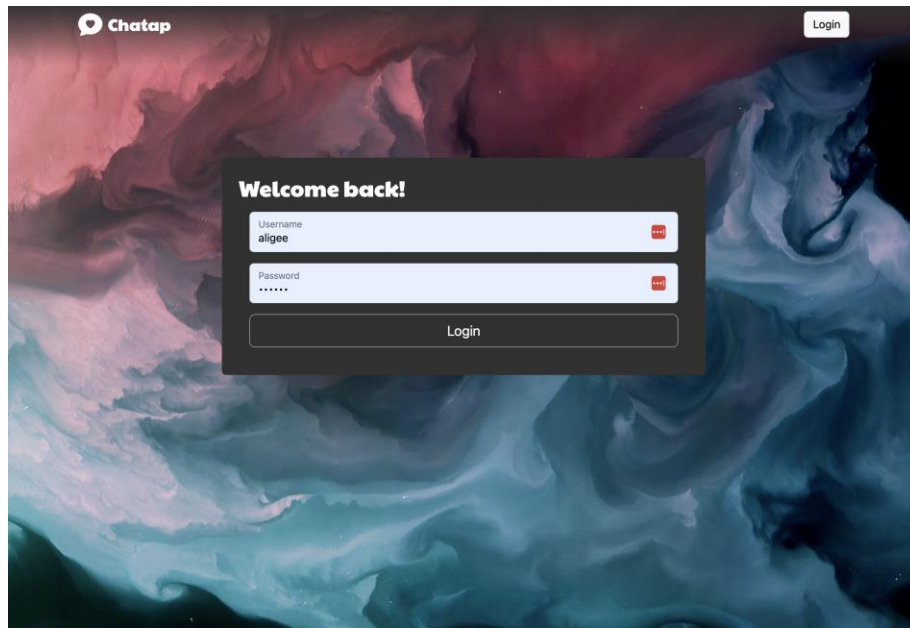
```
socket.on("send_message", (messages) => {
  socket.broadcast.emit("receive_message", messages);
});
```

Receiver

```
useEffect(() => {
  socket.on("receive_message", (messages) => {
    const newMessageReceived = messages[messages.length - 1];
    if (
      !activeChannelCompare ||
      activeChannelCompare !== newMessageReceived.channelId
    ) {
      console.log(newMessageReceived);
      if (!Notifications.includes(newMessageReceived)) {
        setNotifications([newMessageReceived, ...notifications]);
        console.log(notifications);
      }
    } else {
      setListOfMessages(messages);
    }
  });
});
```

Teaching: Wrapper components

- Use cases:
 - Many components that share the same layout?
 - Part of the page UI stays the same while other components change? (ex. Grid system)
 - Frequently used UI elements with unknown children?



Teaching: Wrapper components

- Step 1: Create your wrapper component
- Step 2: Insert {children} - don't need to know them in advance
- Step 3: Wrap children in the wrapper (don't forget to import!)

Add as many children to the wrapper as you like!

```
5  function AuthLayout({ children }) {  
6    return (  
7      <div className="d-flex flex-column landing  
        background-img">  
8        <Navigation />  
9        {children}  
10     </div>  
11   )  
12 }
```

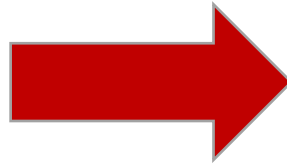
```
9  function LandingPage() {  
10    const navigate = useNavigate()  
11    const { authState } = useContext(AuthContext);  
12  
13    return (  
14      <AuthLayout>  
15        <section className='d-flex align-items-center' style={{ height:  
          '70%' }}> ...  
32      </section >  
33      </AuthLayout>  
34    )  
35  }  
You, 5 days ago • landing page, navbar
```

Utils File + Passing functions

- Code Reuse: DRY
- A Utilities file to hold reusable functions
- Example from the Home page

~40 lines of code

```
/**
 * A function that adds a user to a channel (if not exist) by creating a new record in the users_channels table
 * @param {channel, authState, setMsg, setMsgType} param0
 */
export const joinChannel = ({
  channel,
  authState,
  setMsg,
  setMsgType,
  joinedChannels,
  setChannelsData,
}) => {
  if (!authState || authState.roles !== "user") {
    const accessToken = localStorage.getItem("accessToken");
    axios
      .post(
        `${process.env.REACT_APP_SERVER_URL}/channels/join`,
        {
          channelId: channel.id,
        },
        { headers: { accessToken: accessToken } }
      )
      .then((response) => {
        if (response.data === 1) {
          joinedChannels.push(channel.id);
          setMsg(`You are now a member of channel "${channel.title}"`);
        }
        if (response.data === -1) {
          const index = joinedChannels.indexOf(channel.id);
          if (index > -1) {
            // only splice array when item is found
            joinedChannels.splice(index, 1); // 2nd parameter means remove one item only
            setMsg(`You have left channel "${channel.title}"`);
          }
        }
        fetchChannels({ setChannelsData });
        // setMsg(response.data.message);
        // setMsgType("text-success");
      })
      .catch((error) => {
        console.log(error);
        if (error.response) {
          setMsgType("text-danger");
          setMsg(error.response.data.message);
        }
      });
  }
};
```

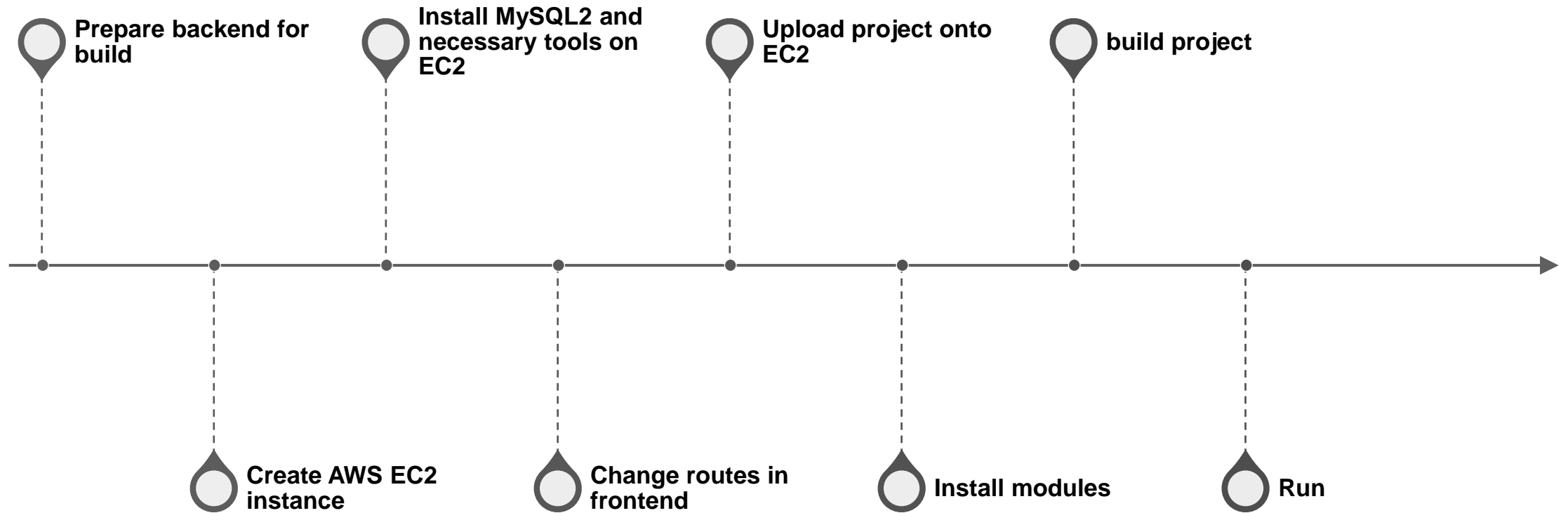


2 lines of code

```
import { joinChannel } from '../helpers/Utils';
```

```
onClick={() =>
  joinChannel({
    channel,
    authState,
    setMsg,
    setMsgType,
    joinedChannels,
    setChannelsData
  })
}
```


Deployment



Future Work

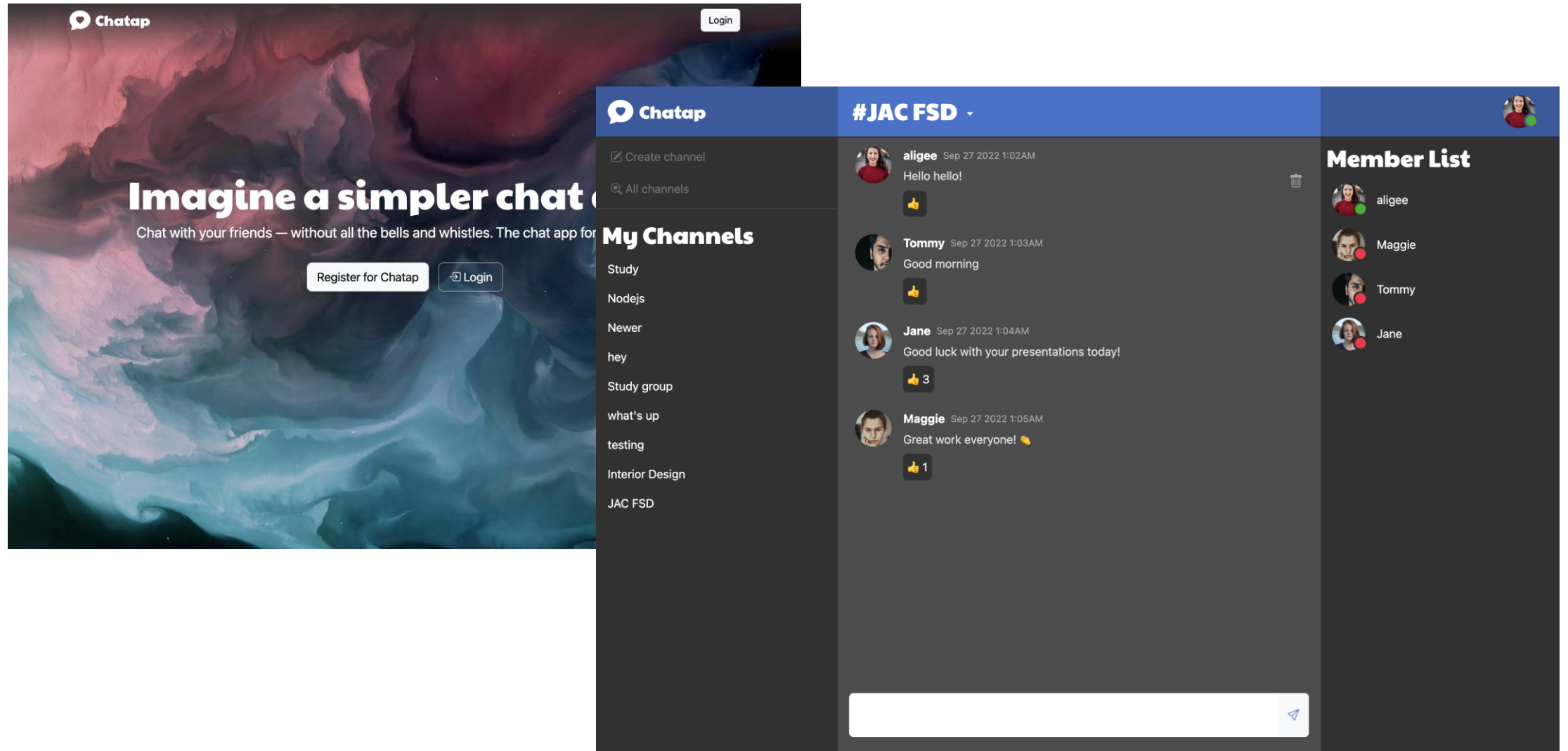


A word cloud of future work items, with words in blue and orange. The words are arranged in a roughly triangular shape, pointing downwards. The words include: notification, editing, io, ui text, socket, unread message, user profile, avatar, reappoint channel owner, channel member, layout, block, bottom, edit status, search, pin, scroll, and styling.

notification editing
io ui text socket
unread message user profile
avatar reappoint channel owner
channel member layout
block bottom edit status
search pin scroll
styling

...Oh, the features we would implement if we had an extra week!

Summary



Chatap

- Github: github.com/nehmea/2022-WDII-chatApp
- Trello: trello.com/b/IJYiJrCL/chat-app
- Hosted site:
 - <http://ec2-18-220-245-147.us-east-2.compute.amazonaws.com:3001/>
 - Please note that the EC2 instance running the link is shut off because it's not a free Tier

