Building the User Component

The full implementation and styling of the User component. We will deconstruct the user prop and access each attribute separately.

Up till now, we were rendering a User component within the Sidebar, but this component doesn't exist yet.

Please create a **User.js** and **User.css** file within the root directory. Done that?

Now, here's the content of the **User.js** file:

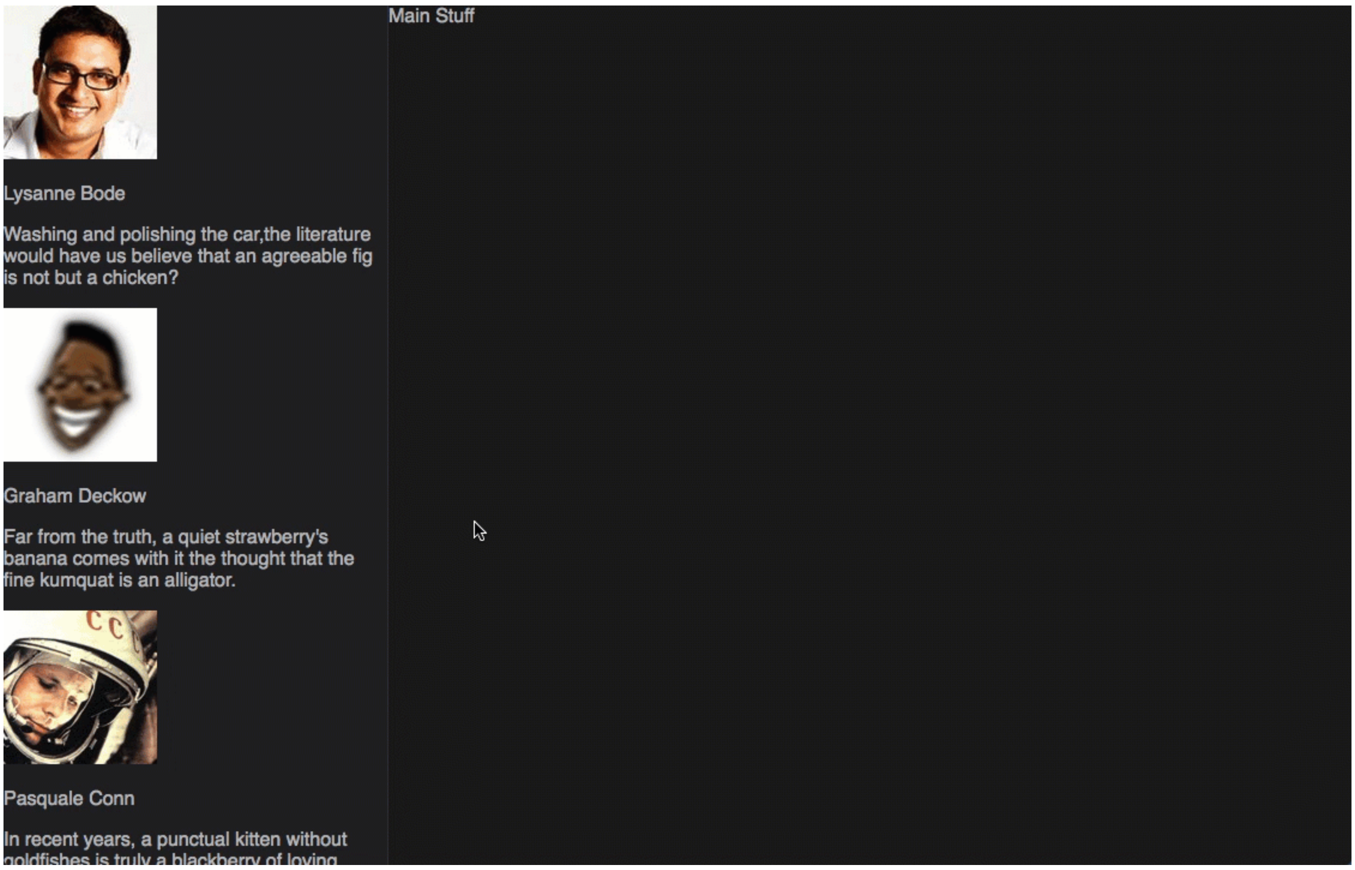
User.js

```
import React from "react";
import "./User.css";
const User = ({ user }) => {
  const { name, profile_pic, status } = user;
  return (
    <div className="User">
      <img src={profile_pic} alt={name} className="User__pic" />
      <div className="User__details">
        <p className="User__details-name">{name}</p>
        <p className="User__details-status">{status}</p>
      </div>
    </div>
  );
};

export default User;
```

Don't let that big chunk of code fool you. It is actually very easy to read and understand. Have a second look.

The name, profile_pic url and status of the user are obtained from the props via *destructuring*: `**const { name, profile_pic, status } = user;**` (line 4).

These values are then used in the return statement for proper rendering, and here's the result of that:



Main Stuff

Lysanne Bode

Washing and polishing the car,the literature would have us believe that an agreeable fig is not but a chicken?

Graham Deckow

Far from the truth, a quiet strawberry's banana comes with it the thought that the fine kumquat is an alligator.

Pasquale Conn

In recent years, a punctual kitten without goldfishes is truly a blackberry of loving

The result above is super ugly, but it is an indication that this works!

Now, let's style this.

First, prevent the list of users from overflowing the Sidebar container.

Sidebar.css:

```
.Sidebar {
  ...
  overflow-y: scroll;
}
```

sidebar.css

Also, the font is ugly. Let's change that.

Index.css:

```
@import url("https://fonts.googleapis.com/css?family=Nunito+Sans:400,700");

body { ...
  font-weight: 400;
  font-family: "Nunito Sans", sans-serif;
}
```

Index.css

Finally, handle the overall display of the User component.

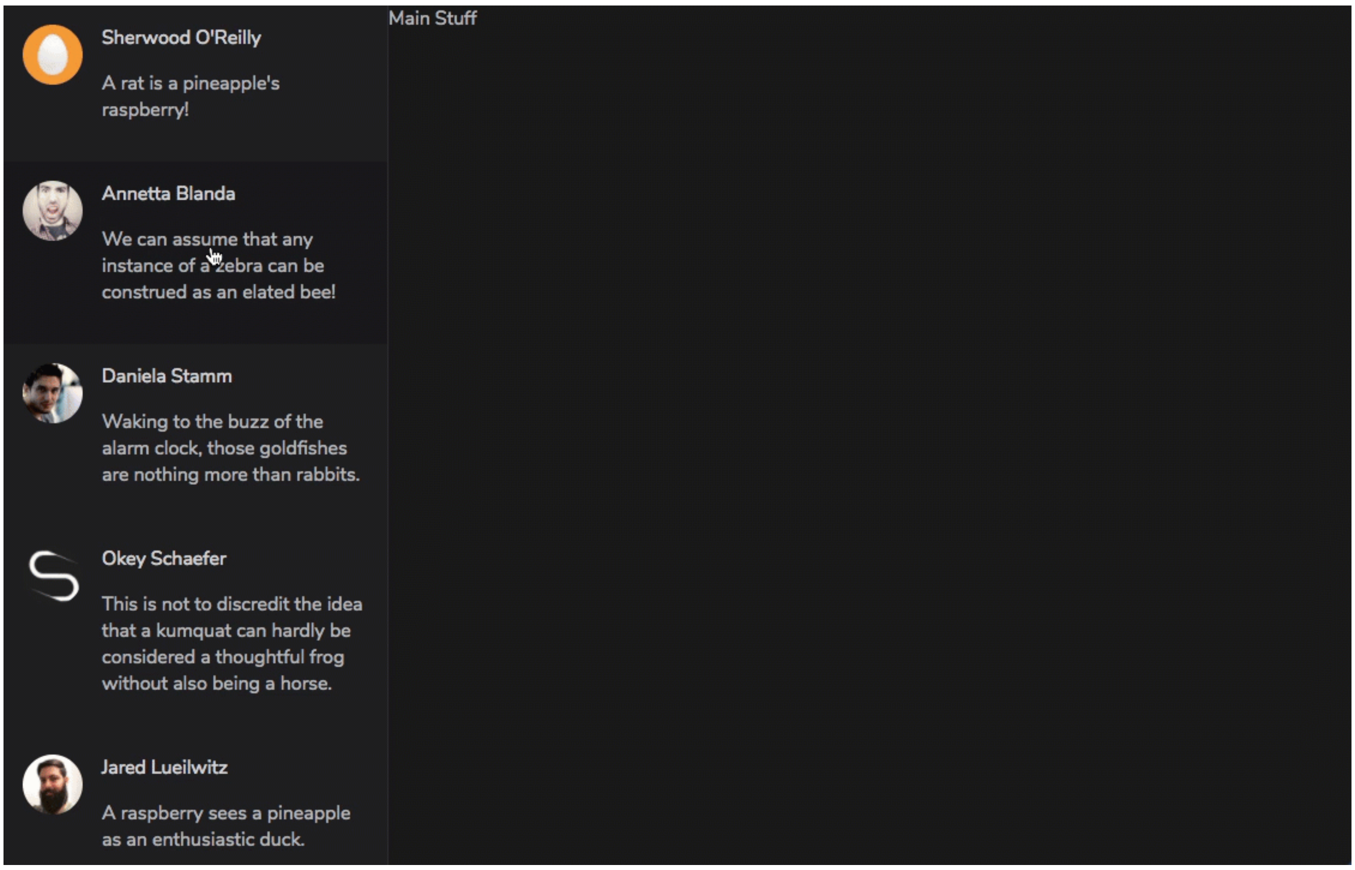
User.css:

```
.User {
  display: flex;
  align-items: flex-start;
  padding: 1rem;
}

.User:hover {
  background: rgba(0, 0, 0, 0.2);
  cursor: pointer;
}

.User__pic {
  width: 50px;
  border-radius: 50%;
}

.User__details {
  display: none;
}
/* not small devices  */

@media (min-width: 576px) {
  .User__details {
    display: block;
    padding: 0 0 0 1rem;
  }
  .User__details-name {
    margin: 0;
    color: rgba(255, 255, 255, 0.8);
    font-size: 1rem;
  }
}
```

user.css

Since this is not a CSS book, I'm skipping some of the styling explanations.

Our final touches are done and voila! Here's the beautiful display we've got now:



Sherwood O'Reilly

A rat is a pineapple's raspberry!

Annetta Blanda

We can assume that any instance of a zebra can be construed as an elated bee!

Daniela Stamm

Waking to the buzz of the alarm clock, those goldfishes are nothing more than rabbits.

Okey Schaefer

This is not to discredit the idea that a kumquat can hardly be considered a thoughtful frog without also being a horse.

Jared Lueilwitz

A raspberry sees a pineapple as an enthusiastic duck.

Amazing!

We've gone from nothing to having a beautiful list of users rendered on the screen.

If you're coding along, resize the browser to see the beautiful view on mobile as well.

Alright, so far, the User component receives contacts as user props, but do we really need that in Redux? Let's find out.