PROJECT

Social Network for Pets

In this project, we'll build a simple social network for pets. You'll be able to view a pet's profile (which contains their name, bio, and friend list) and navigate to other profiles. There will also be a profile directory where you can see all of the profiles.

We've already set up three profiles for you: one for a cat, one for a dog, and one for a Komodo dragon.

Let's get started!

Tasks

10/10 Complete

Mark the tasks as complete by checking them off

Loading User Data

1.

Run the app and take a look at PetBook.

It's fairly barren right now, only showing the currently active username and placeholder data. It's also stuck in a loading state. We'll fix all of this and turn it into a proper pet profile.

The first thing we'll want to do is load user data into the component's state. In order to do that, however, we'll need a place to store that data.

Take a look at the `<Profile>` component in **Profile.js**. This is where we'll be doing our work, though it may be useful to look at other files too.

Create a constructor for this component. Inside, set `this.state` to `{ userData: null }`, which will represent a profile with no user data loaded. Don't forget to call `super(props)`!

Hint

To create a constructor, do something like this:

```
constructor(props) {
  super(props);
  // Your constructor's code goes here
}
```

You'll want to set `this.state` to `{ userData: null }` to start.

2.

Now that we have a place to store our data, let's start loading it.

Create a new method called `loadUserData()` that does two things:

1. Sets the `userData` state to `null` while the data is loading. Once it's loaded (in the next step), we'll update it.
2. Fetches the user data with `fetchUserData`. You'll call the function like this:

```
this.fetchID = fetchUserData(this.props.username,
(userData) => {
  this.setState({ userData });
});
```

(Note: `fetchUserData()` simulates real user data, but because this is a learning exercise, it doesn't actually do that. You don't need to read or understand the code, but know that it would be similar if you were loading real data.)

Hint

To create the `loadUserData()` method, do something like this:

```
loadUserData() {
  // Your method's code goes here
}
```

To update the state, use `this.setState({ userData: null })`.

To fetch the data, copy-paste the code snippet that starts with `this.fetchID =`.

3.

Now that we have a method for loading user data, let's call it when the component mounts.

Add a lifecycle method for when the component mounts, and call `this.loadUserData()` inside.

Hint

You'll be creating the `componentDidMount()` method. Inside, you'll have just one line that calls `this.loadUserData()`.

4.

We're now loading user data, but we aren't displaying it anywhere because we haven't updated the `render()` method. Let's fix that!

The first thing we'll need to change is `isLoading`. It shouldn't always be `true`—it should only be `true` when `this.state.userData === null`. Update `isLoading` to reflect that.

When you run the code, you won't see any user data displayed, but you may see the profile change slightly after data has loaded. In the next task, we'll actually put that data on the screen.

Hint

Inside of `render()`, set `isLoading` to `this.state.userData === null` to achieve the desired effect.

Displaying User Data

5.

Let's start by displaying the user's name. (We'll get to their bio and friend list next.)

Inside of `render()`, create a new variable called `name`.

If `isLoading` is `true`, set `name` to a sample value, such as `'Loading...'`. (The exact string is up to you.) Otherwise, set `name` to `this.state.userData.name`.

Finally, replace "Name goes here" with `{name}`.

When you run this and visit a profile, you should see the user's name appear!

Hint

You can use the existing `if (isLoading)` code as part of this. Here's how you might achieve something like that:

```
let name;
if (isLoading) {
  // …
```

```
  name = 'Loading...';
} else {
  name = this.state.userData.name;
}
```

6.

Time to do the exact same thing, but for the user's bio.

Inside of `render()`, create a new variable called `bio`.

When `isLoading` is `true`, set `bio` to a sample value. Otherwise, set `bio` to `this.state.userData.bio`.

Finally, replace "Bio goes here" with `{bio}`.

Now the user's bio should appear!

Hint

This should look very similar to the previous step, but instead of `name`, you'll use `bio`.

7.

Two more to go: let's update the user's friend list.

Create a new variable called `friends` inside `render()`. When the component is loading, `friends` should be the empty array (`[]`). Otherwise, it should be `this.state.userData.friends`.

Next, update the `usernames` prop of the `<Userlist>` component. It's currently an empty array—change it from `{[]}` to `{friends}`.

Hint

This should look very similar to the previous two steps, but instead of `name` and `bio`, you'll use `name`. Make sure to update the `usernames` prop of the `<Userlist>` component.

8.

The last thing we'll do is update the user's profile picture.

Find the `<div>` with the class name of `profile-picture`. It's currently empty…but not for long!

We'll want to put the following inside, but only if `isLoading` is `false`:

```
<img src={this.state.userData.profilePictureUrl} alt="" />
```

You can do this by defining a variable set conditionally (like `name` and `bio`), or by using an inline `&&` right inside the JSX.

Now, the profile picture should show up once it loads! We're not done yet, but the profile is looking good.

Hint

Here's what the `<div>` might look like:

```
<div className="profile-picture">
  {!isLoading && (
    <img src={this.state.userData.profilePictureUrl} alt="" />
  )}
</div>
```

Cleaning Up

9.

Profiles should now be showing up! There are two problems left to fix, though:

1. If you open a profile, then quickly return to the directory, then go to a different pet's profile, there will be some jitter. (This requires some quick clicking.)
2. If you click on a pet's friends, only the username updates. We don't fetch new data.

Let's start with the first problem.

We'll solve this by canceling the fetch when `<Profile>` unmounts.

Add a lifecycle method for when the component unmounts, and call `cancelFetch(this.fetchID)` inside.

Hint

You'll be creating the `componentWillUnmount()` method. Inside, you'll have just one line that is `cancelFetch(this.fetchID)`.

10.

Let's solve our final problem: if you click on a pet's friends, only the username updates. We don't fetch new data.

Add a lifecycle method for when the component updates. If the username has changed (in other words, if `this.props.username !== prevProps.username`), we should do two things:

1. Cancel the fetch currently in progress with `cancelFetch(this.fetchID)`.
2. Call `this.loadUserData()` again.

Once that's done, you should be able to navigate through pet profiles with ease! Be proud…your pet social network is going to be big.

Hint

You'll be creating the `componentDidUpdate(prevProps)` method.

Inside, you'll use an `if` statement to check if `this.props.username !== prevProps.username`. If so, you'll want to cancel the fetch and call `this.loadUserData()`.

Profile.js

```
import React from 'react';
import { fetchUserData, cancelFetch } from './dataFetcher';
import { Userlist } from './Userlist';

export class Profile extends React.Component {
  constructor(props) {
    super(props);
    this.state = { userData: null};
  }

  loadUserData() {
    this.setState({ userData: null })
    this.fetchID = fetchUserData(this.props.username, (userD
ata) => {
  this.setState({ userData });
});
  }

  componentDidMount() {
    this.loadUserData();
  }
```

```
componentWillUnmount(){
 cancelFetch(this.fetchID)
}

componentDidUpdate(prevProps){
  if(this.props.username !== prevProps.username){
    cancelFetch(this.fetchID);
    this.loadUserData();
  }
}

render() {
  const isLoading = this.state.userData === null;

  let className = 'Profile';
  if (isLoading) {
    className += ' loading';
  }

  let name;
  if (isLoading) {
    name = 'Loading...';
  } else {
    name = this.state.userData.name;
  }

  let bio;
  if (isLoading) {
    bio = 'Loading...';
  } else {
    name = this.state.userData.bio;
  }

  let friends;
  if (isLoading) {
    friends = [];
  } else {
    friends = this.state.userData.friends;
  }
```

```
    return (
      <div className={className}>
        <div className="profile-picture">
          {!isLoading && (
            <img src={this.state.userData.profilePictureUrl}
 alt="" />
          )}
        </div>
        <div className="profile-body">
          <h2>{name}</h2>
          <h3>@{this.props.username}</h3>
          <p>{bio}</p>
          <h3>My friends</h3>
          <Userlist usernames={friends} onChoose={this.props
.onChoose} />
        </div>
      </div>
    );
  }
}
```