**Authorization Form**

A client just called you to say that they love their new website! There’s only one problem: they don’t like how their contact page displays their personal information for all to see.

They’ve asked you to hide their website’s contact page behind a password form. In this project, you’ll accomplish this by using a React component to set up a simple authorization layer.

Let’s get started!

If you get stuck during this project or would like to see an experienced developer work through it, click “**Get Help**“ to see a **project walkthrough video**.

**Tasks**

**0/9Complete**

Mark the tasks as complete by checking them off

**1.**

Click Save to see the current state of things.

The contact info in the browser looks fine, but it should be hidden until you enter a password!

Look in the code editor. You can see a Contact component class. Contact‘s instructions object has three methods: constructor(), .authorize(), and .render().

constructor() is a special method. You’ll learn about it in the next unit. For now, just know that you can check whether a user has entered the right password by running the expression this.state.authorized.

**2.**

Let’s start with the <h1></h1> in the render function.

Right now, the <h1></h1> displays the text Contact. If a user hasn’t been authorized, then you want the <h1></h1> to display Enter the Password instead.

Make the <h1></h1> display Contact *only* if this.state.authorized is true. If this.state.authorized is false, then the <h1></h1> should display Enter the Password.

Stuck? Get a hint

**3.**

The browser should say ‘Enter the Password.’

To make sure it’s working properly, edit the constructor() method so that the user is authorized:

constructor(props) {

super(props);

this.state = {

password: 'swordfish',

authorized: true

};

this.authorize = this.authorize.bind(this);

}

This should change the text back to ‘Contact’.

If it works, then make sure to change authorized back to false!

Hint

We haven’t talked about this line:

this.authorize.bind(this);

just yet! This line is necessary because authorize()‘s body contains the keyword this. We’ll talk about it more soon!

**4.**

If the user isn’t authorized, then you want them to see a login form into which they can enter a password. Let’s make that login form!

In the .render() method, before the return statement, declare a new variable named login.

Set login equal to a JSX <form></form> element. This <form></form> is going to have multiple children, so wrap it in parentheses!

Give the <form></form> an attribute of action="#".

Hint

render() {

const login = (

<form action="#">

</form>

);

return (

**5.**

Good! Now let’s give your form some <input />s for the user to fill out.

In between the <form></form> tags, write two <input /> tags. Give the first <input /> two attributes: type="password" and placeholder="Password". Give the second <input /> one attribute: type="submit".

Hint

const login = (

<form action="#">

<input

type="password"

placeholder="Password" />

<input type="submit" />

</form>

);

return (

**6.**

Now let’s hide the contact info.

After your login variable, declare another variable named contactInfo. Set it equal to empty parentheses:

const contactInfo = (

);

return (

Cut the <ul></ul> out of the return statement, and paste it in between those parentheses!

Hint

const contactInfo = (

<ul>

<li>

client@example.com

</li>

<li>

555.555.5555

</li>

</ul>

);

return (

**7.**

Great! By saving two JSX expressions as variables, you’ve set yourself up nicely to toggle between them.

In the render function’s return statement, make a new line right below the <h1></h1>. On this new line, use a ternary operator. If this.state.authorized is true, make the ternary return contactInfo. Otherwise, make the ternary return login.

Hint

</h1>

{ this.state.authorized ? contactInfo : login }

</div>

**8.**

On lines 14 through 21, you can see a method named .authorize().

This method will check whether a submitted password is equal to ‘swordfish’. If it is, then this.state.authorized will become true.

You need authorize to get called whenever a user hits “Submit!”

Give the <form></form> an onSubmit attribute. Set the attribute’s *value* equal to the authorize function.

Hint

<form action="#" onSubmit={this.authorize}>

**9.**

Try entering an incorrect password and hitting ‘Submit.’ Nothing should happen.

Now try entering ‘swordfish.’ Your screen should change!