129-Testing Login-setting inputs and submitting video

- go to Login.test.js and do a couple of more test cases

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import expect from 'expect';
import { mount } from 'enzyme';

import { Login } from './Login'

if(Meteor.isClient){
    describe('Login', function(){
        it('should show error messages', function(){
            const error = 'this is not working';
            const wrapper = mount(<Login loginWithPassword={()=>{}}/>);

            wrapper.setState({ error });
            const p = wrapper.find('p').text();
            expect(p).toBe(error);

            wrapper.setState({ error: ''});
            expect( wrapper.find('p').length).toBe(0);
        });

        it('should call loginWithPassword with the form data');

        it('should set loginWithPassword callback error')
    });
}
```

- now lets setup the function for the first one, we want to create two variables, one form email, and one for password, and we are also going to create a new spy, then we are going to create a mount as well, and we are going to pass in an instance of <Login>, and the only prop we need to provide is loginWithPassword and we are going to set that equal to spy

```
        it('should call loginWithPassword with the form data', function(){
            const email = 'jose@test.com';
            const password = 'password123';
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);
        });
```

- next we are going to set the email from the input from Login.js, and we target the email value from our form by using ref, and by coverting to node.value, and we set that equal to our variable email

```
        it('should call loginWithPassword with the form data', function(){
            const email = 'jose@test.com';
            const password = 'password123';
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);

            wrapper.ref('email').node.value = email;
        });
```

- do the same with password

```
        it('should call loginWithPassword with the form data', function(){
            const email = 'jose@test.com';
            const password = 'password123';
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);

            wrapper.ref('email').node.value = email;
            wrapper.ref('password').node.value = password;
        });
```

- now we need to simulate a form submit, first we need to find() all the elements with <form> and simulate ('submit')

```
        it('should call loginWithPassword with the form data', function(){
            const email = 'jose@test.com';
            const password = 'password123';
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);

            wrapper.ref('email').node.value = email;
            wrapper.ref('password').node.value = password;
            wrapper.find('form').simulate('submit');
        });
```

- now its time to make our assertion

```
        it('should call loginWithPassword with the form data', function(){
            const email = 'jose@test.com';
            const password = 'password123';
            const spy = expect.createSpy();
            const wrapper = mount(<Login loginWithPassword={spy}/>);

            wrapper.ref('email').node.value = email;
            wrapper.ref('password').node.value = password;
            wrapper.find('form').simulate('submit');

            expect(spy.calls[0].arguments[0]).toEqual({ email });
        });
```

- now we are going to check the second argument, which is the password, and we know that password is not an object like email is

```
        it('should call loginWithPassword with the form data', function(){
```

```
        const email = 'jose@test.com';
        const password = 'password123';
        const spy = expect.createSpy();
        const wrapper = mount(<Login loginWithPassword={spy}/>);

        wrapper.ref('email').node.value = email;
        wrapper.ref('password').node.value = password;
        wrapper.find('form').simulate('submit');

        expect(spy.calls[0].arguments[0]).toEqual({ email });
        expect(spy.calls[0].arguments[1]).toBe(password);
    });
```

- now lets go to the second test case, this is for the form error, first lets set the spy, next we'll do the wrapper, next is to simulate the form submitting, next is calling our third argument, next is to expect the state error, and we dont care what the error says, all we want is that it is a string that returns

```
    it('should set loginWithPassword callback error', function(){
        const spy = expect.createSpy();
        const wrapper = mount(<Login loginWithPassword={spy}/>);

        wrapper.find('form').simulate('submit');

        spy.calls[0].arguments[2]({});
        expect(wrapper.state('error').length).toNotBe(0);
    })
```

- now we are going to clear the error by calling the third argument again but with no object, just empty

```
    it('should set loginWithPassword callback error', function(){
        const spy = expect.createSpy();
        const wrapper = mount(<Login loginWithPassword={spy}/>);

        wrapper.find('form').simulate('submit');
```

```
        spy.calls[0].arguments[2]({});
        expect(wrapper.state('error').length).toNotBe(0);

        spy.calls[0].arguments[2]();
        expect(wrapper.state('error').length).toBe(0);
    })
```

-