Storybook

Goal

The goal of this lab is to use Storybook to develop React components.

The Mission

In this lab you will be building a set of Storybook scenarios to enhance and fix a series of React components.

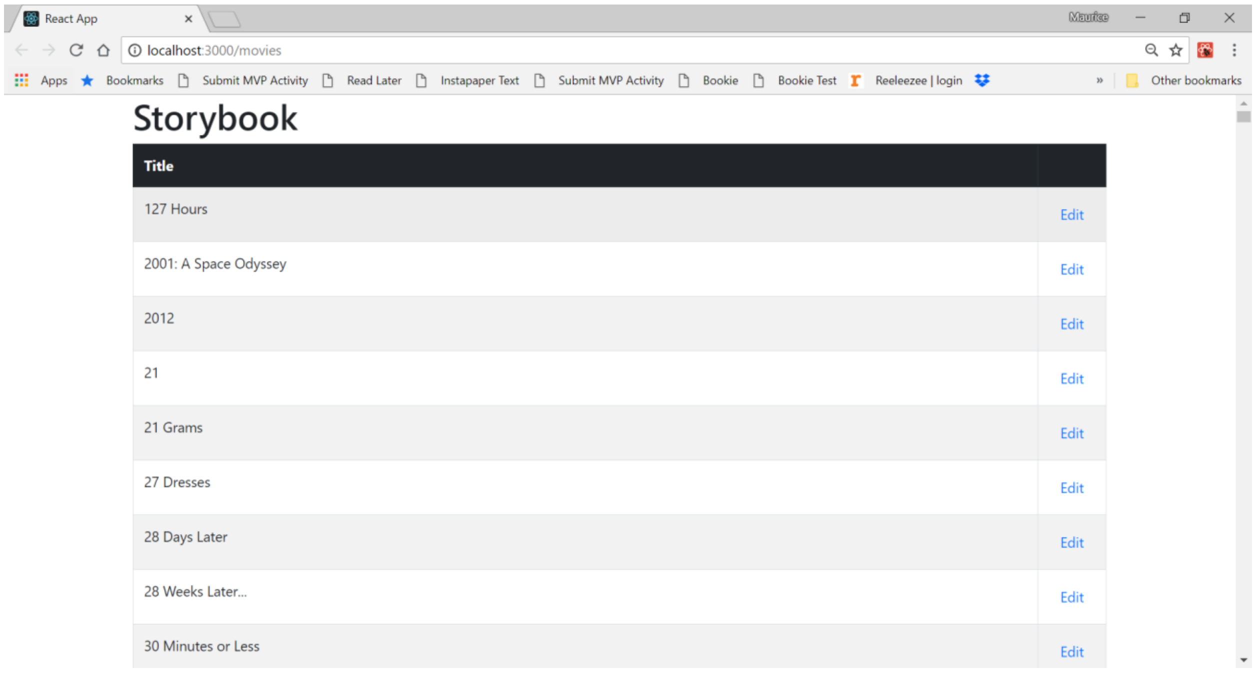
Please note that the first time you start this project it will take some time, as a number of NPM packages will download. This requires an Internet connection where the NPM registry is not blocked by a firewall.

Note: The **FINISHED** folder contains a finished version of this lab as a reference.

Type it out by hand?
Typing it drills it into your brain much better than simply copying and pasting it. Because you're forming new neuron pathways that are going to help you in the future, help them out now!

Assignment 1

Start the application by running the START.BAT file located in the in the **BEGIN** folder and observe that the main page displays a list of movies. When the edit button for one of these movies is clicked a data entry form is shown. The application is basically functional but does have some small issues you will fix.



Storybook

Title	
127 Hours	Edit
2001: A Space Odyssey	Edit
2012	Edit
21	Edit
21 Grams	Edit
27 Dresses	Edit
28 Days Later	Edit
28 Weeks Later...	Edit
30 Minutes or Less	Edit

The first task is to add support for Storybook

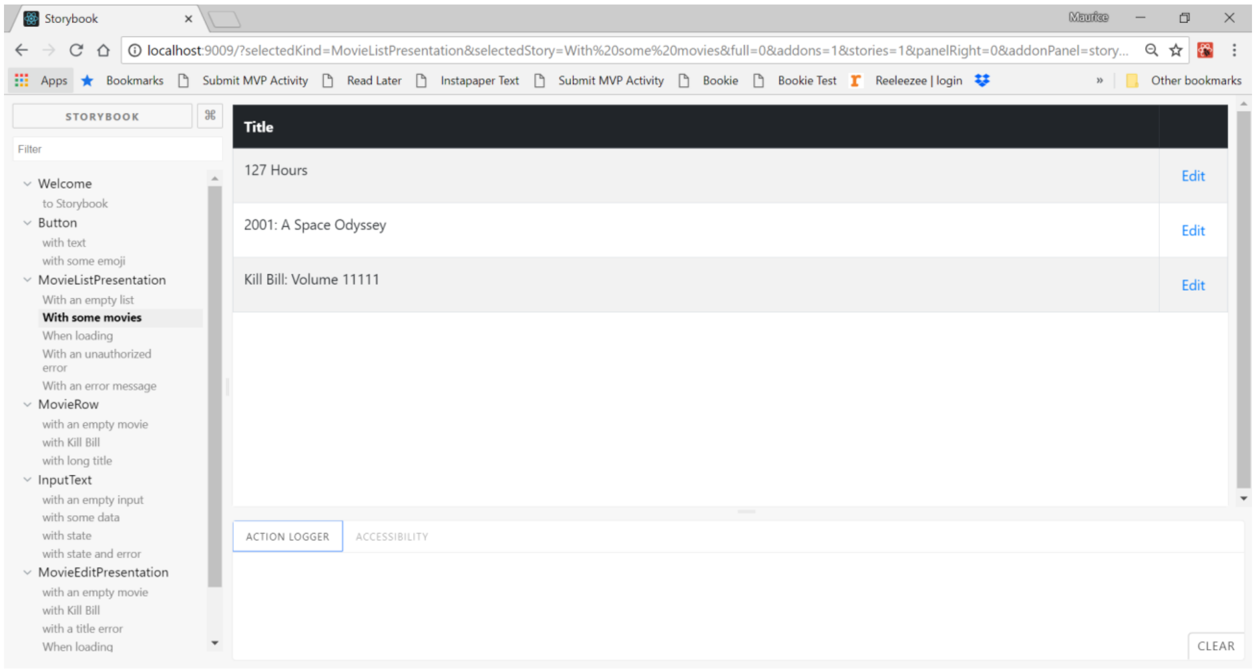
Open a command window and install the Storybook CLI if you haven't previously done so. This can be done with NPM as a global install using the command *npm i -g @storybook/cli*.

Add support for Storybook to the project by executing *getstorybook* in the same folder as the PACKAGE.JSON is located. You should now be able to start Storybook with the command *npm run storybook*. Once storybook is running you can open it in the browser at http://localhost:9009/.

Create stories to render the movie list presentational component

Create a new file MOVIELISTPRESENTATION.JS in the STORIES folder and add a basic story to render an empty array of movies. Add two more stories to display the loading and error states of the component.

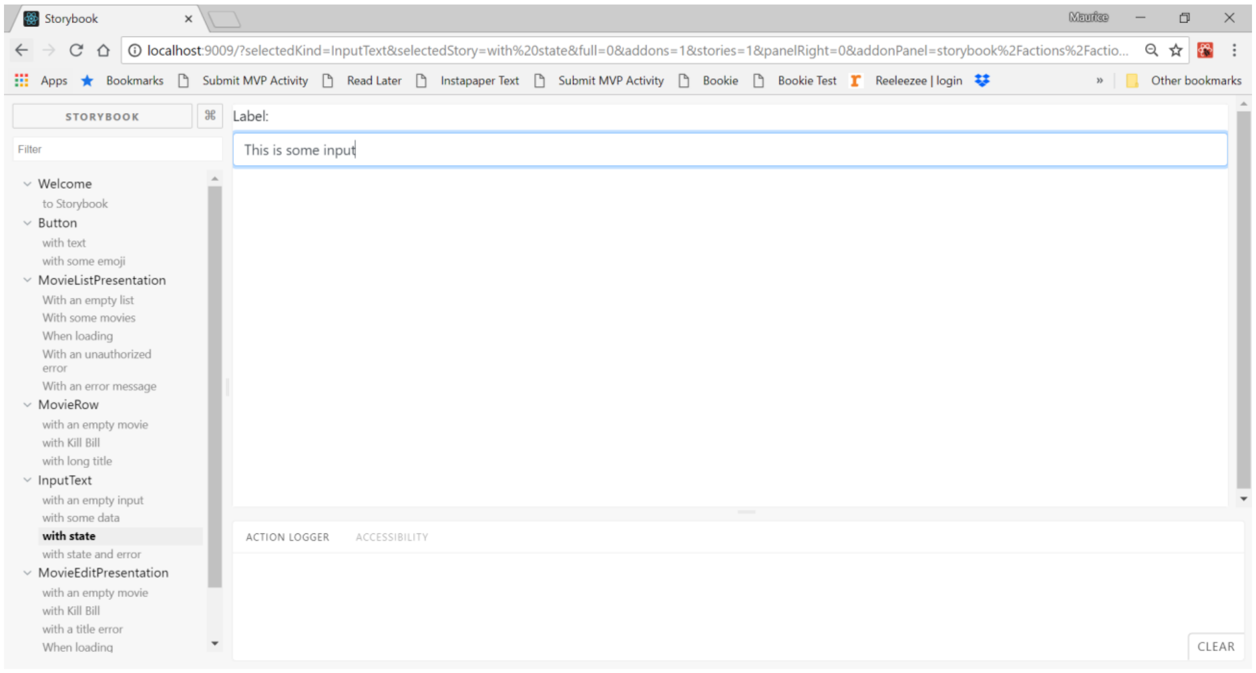
Add a second story to render a few movie objects. Each movie needs an *id* and *title* property. Notice that this results in an error: **You should not use <Link> outside a <Router>** because you are using the React-Router *<Link>* component in each *MovieRow*. Add a decorator to the *MovieListPresentation* stories to wrap each instance in a *<MemoryRouter>* component. Now all your stories should work just fine.



Title	
127 Hours	Edit
2001: A Space Odyssey	Edit
Kill Bill: Volume 11111	Edit

Create stories to render the input text component

Create a new file INPUTTEXT.JS in the STORIES folder and add a basic story to render a simple input component with a fixed string as value. Use the *action* addon to display when the *onChange* callback is executed. Make sure the story renders properly and the *onChange* action is logged when you type into the input. Note that you can’t really change the value as the component will always render the same fixed string.



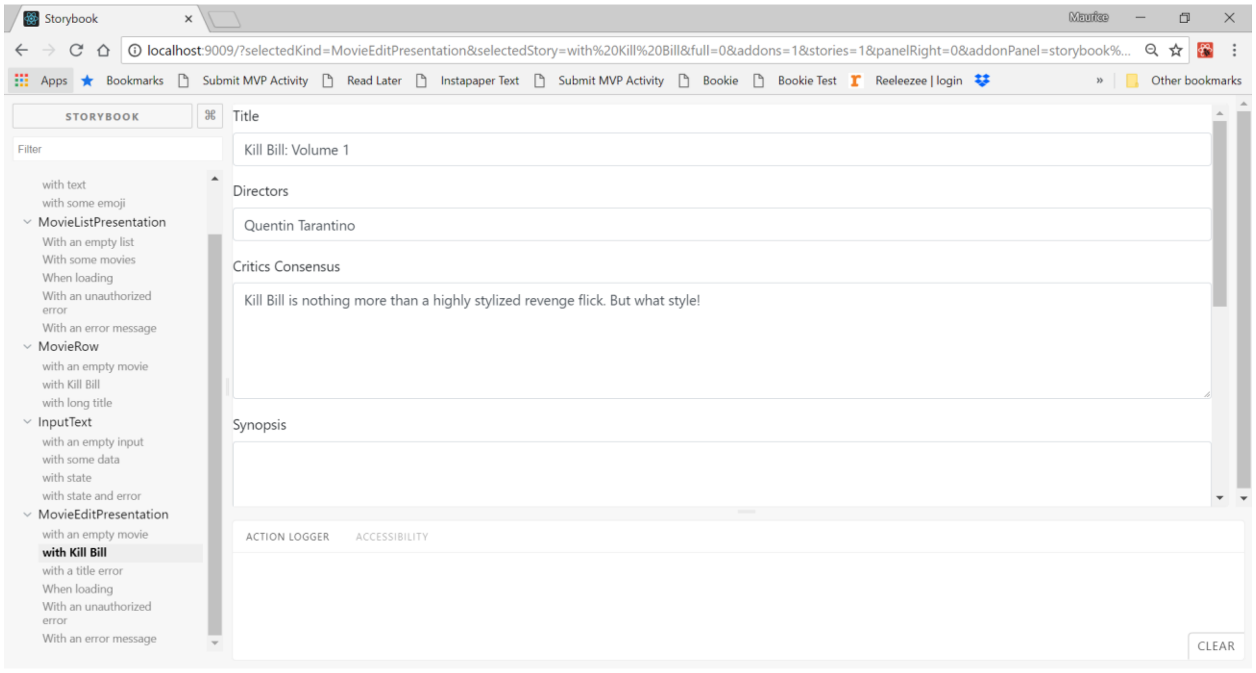
Create a *StateDecorator* helper component so we can interact with the *InputText* component and update its value.

```
class StateDecorator extends Component {
  state = {
    value: 'Hello'
  };

  onChange = e => this.setState({ [e.prop]: e.value });

  render() {
    const props = { prop: 'value', onChange: this.onChange, ...this.state };
    return cloneElement(this.props.children, props);
  }
}
```

Create another story with the *InputText* component decorated by this *StateDecorator* helper component. You should now be able to edit the value. Add one more story to display the *InputText* component with a few error messages.

Create stories to render the movie editor component

Create a new file MovieEditPresentation.js in the stories folder and add a basic story to render the complete movie editor. This component uses the React-Router *<Link>* component again so you will need a decorator with the *<MemoryRouter>* component.



with text
with some emoji
MovieListPresentation
With an empty list
With some movies
When loading
With an unauthorized error
With an error message
MovieRow
with an empty movie
with Kill Bill
with long title
InputText
with an empty input
with some data
with state
with state and error
MovieEditPresentation
with an empty movie
with Kill Bill
with a title error
When loading
With an unauthorized error
With an error message

Title: Kill Bill: Volume 1

Directors: Quentin Tarantino

Critics Consensus: Kill Bill is nothing more than a highly stylized revenge flick. But what style!

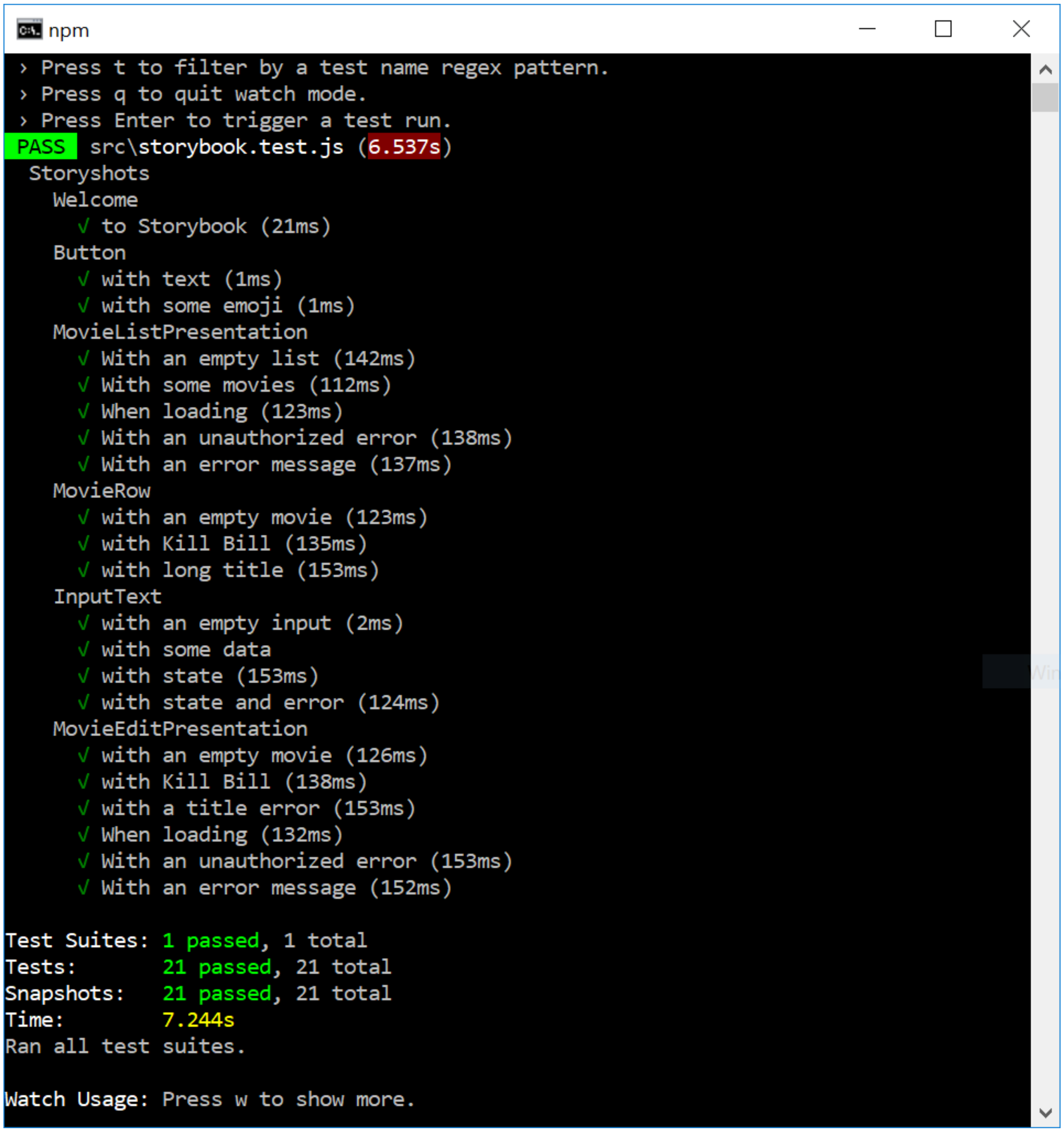
Synopsis:

Add different stories for:

1. Normal movie
2. Movie with an error message for the title. Note that the save button should be disabled
3. With a loading status. Notice this will produce an error without a movie object. This seems wrong and can be fixed my moving the "const ratings ..." line after the loading and error check.

Assignment 2

Storybook has many addons. Use the Storyshots addon to create snapshot tests for your stories. Make sure these run without any errors.



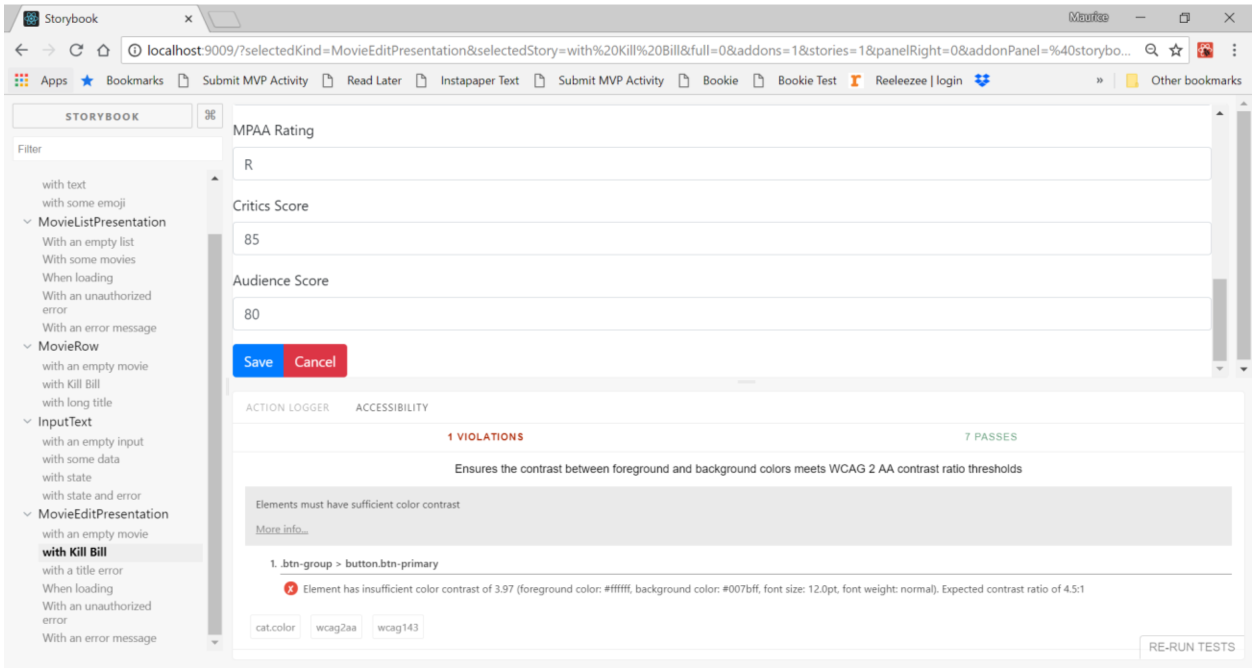
npm

```
 › Press t to filter by a test name regex pattern.
 › Press q to quit watch mode.
 › Press Enter to trigger a test run.
 PASS  src\storybook.test.js (6.537s)
  Storyshots
    Welcome
      √ to Storybook (21ms)
    Button
      √ with text (1ms)
      √ with some emoji (1ms)
    MovieListPresentation
      √ With an empty list (142ms)
      √ With some movies (112ms)
      √ When loading (123ms)
      √ With an unauthorized error (138ms)
      √ With an error message (137ms)
    MovieRow
      √ with an empty movie (123ms)
      √ with Kill Bill (135ms)
      √ with long title (153ms)
    InputText
      √ with an empty input (2ms)
      √ with some data
      √ with state (153ms)
      √ with state and error (124ms)
    MovieEditPresentation
      √ with an empty movie (126ms)
      √ with Kill Bill (138ms)
      √ with a title error (153ms)
      √ When loading (132ms)
      √ With an unauthorized error (153ms)
      √ With an error message (152ms)

Test Suites: 1 passed, 1 total
Tests:       21 passed, 21 total
Snapshots:   21 passed, 21 total
Time:        7.244s
Ran all test suites.

Watch Usage: Press w to show more.
```

Assignment 3 (Optional)

Another useful Storybook can be used to check the accessibility of your components.

Install *@storybook/addon-a11y* from https://www.npmjs.com/package/%40storybook%2Faddon-a11y and add it to al stories to check for issues in the various components. There will be a few warnings. One of these is: **Ensures every form element has a label**. Add the *htmlFor* to the label in the *InputText* and *TextArea* components to fix this.



MPAA Rating: R

Critics Score: 85

Audience Score: 80

1 VIOLATIONS — 7 PASSES

Ensures the contrast between foreground and background colors meets WCAG 2 AA contrast ratio thresholds

Elements must have sufficient color contrast

More info...

1. .btn-group > button.btn-primary

Element has insufficient color contrast of 3.97 (foreground color: #ffffff, background color: #007bff, font size: 12.0pt, font weight: normal). Expected contrast ratio of 4.5:1

cat.color wcag2aa wcag143