Description

App component

Use combineReducers function to create `rootReducer`. Use createStore function to create store.
Import store to the `index.js`.
Wrap `App` component in the "Provider"
Pass `store` to the `Provider`.
Now all components have access to the store, and you can use `Redux Hooks`

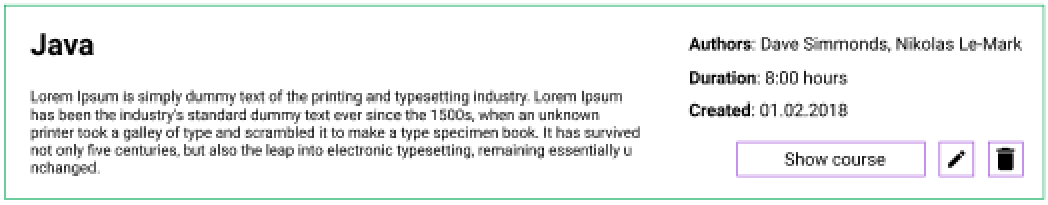
> Reading State from the Store with `useSelector`
> Dispatch actions using `useDispatch`

Courses component

Implement new feature for Courses component:

- Get courses data from the back-end. See SWAGGER `/courses/all` API.
- Save courses response to the store.
- Courses list should be rendered based on data from the `courses` property in **store**.

CourseCard component


Java
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially u nchanged.
Authors: Dave Simmonds, Nikolas Le-Mark
Duration: 8:00 hours
Created: 01.02.2018
Show course

(example)

Implement new features for CourseCard component:

- Update `CourseCard` component by adding two buttons: `Delete` and `Update`.
  (These buttons should be reusable component `Button`).
- When user clicks the `Delete` button current course should be removed from the store and new courses list should be rendered.
- `Update` button has **NO functionality in this module**.
  It will be added in the next module.

Add new course

Implement new features for CreateCourse component:

- When user clicks `Create author` button the new author should be saved to the store.
- When user clicks `Create course` button the new course should be saved to the store.

Header

Implement new features for Header component:

- Take information about user from the store.
- When user clicks `LOGOUT` button you should dispatch action to the store.

Login component

Implement new features for Login component:

- Save information (name, token, email) from response to the store.

> REMEMBER:
>
> If we have a token in the `localStorage` the user should be immediately directed to the Courses page when opening the application.
>
> You should save token to the `localStorage` after success login.
>
> You should delete token from the `localStorage` after logout.

GOOD PRACTICES THAT YOU CAN APPLY FOR THIS TASK:

1. Place `useEffect` before `return` statement.
2. Use `Redux DevTools` extension for Chrome browser and composeWithDevTools to see store changes in the browser.
3. Use separate `selectors.js` file for `useSelector` callbacks. For example:

```
// BAD

// Courses.jsx
const courses = useSelector(state => state.courses);


// GOOD

// selectors.js
export const getCourses = state => state.courses;
```

```
// Courses.jsx
const courses = useSelector(getCourses);
```