# Description

#### Store

• Add new value role to the initial state of user

```
const userInitialState = {
    isAuth: '',
    name: '',
    email: '',
    token: '',
    role: '', // new value
};
```

### Private route (new component)

Create PrivateRoute component.

- This component should render any component as a children.
- This component checks user's role and if it is **ADMIN** PrivateRoute return component as children otherwise redirect to /courses.

### Header

Add new functionality for the logout button

• When user clicks logout button you should make a request to logout.

```
Swagger:
url: "/logout",
method: "DELETE".
```

# Courses component

- Button Add new course and route /courses/add should be available only for ADMIN role.
- Make a request to get current user /users/me.
   Save a response to the store. Use **Thunk** for it.



Don't forget to add header Authorization: token

### CourseCard component



This components has different functionality for users with different roles

#### 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**: 08:00 hours **Created**: 01.02.2018

Show course

(CourseCard component without ADMIN role)



(CourseCard component with ADMIN role)

### Course card with ADMIN role

When user has role **ADMIN** CourseCard 2 new buttons should display:

1. Delete course button:

When user clicks this button - course should be deleted from back-end. Delete this course from store after success response.

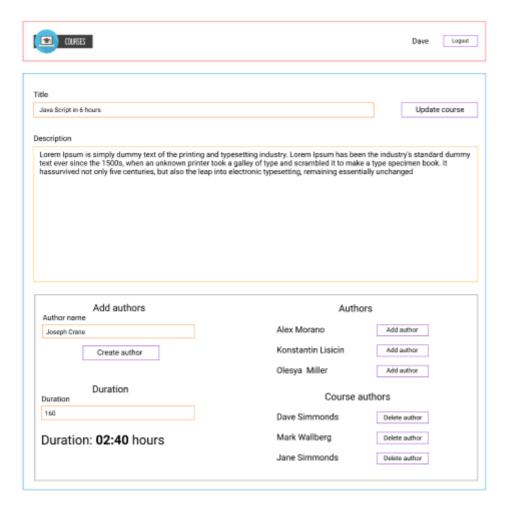
Use Thunk for it.

Swagger:
url: "/courses/:id",
method: "DELETE".

2. Update course button:

When user clicks this button **Update course** form opens by route /courses/update/:courseId.

## Update course



(example)

Now you should use CourseFrom (old CreateCourse) for course creation and course updating.

When user clicks Update button on the CourseCard:

- CourseForm renders instead of the Courses component by route
   /courses/update/:courseId(courseId id of course that should be updated).
   (Use useParams hook to get courseId).
- All fields on the CourseForm page have prefilled data from opened course.
- Route /courses/update/:courseId should be accessible only for a role ADMIN. (Use Private route component).
- After clicking on the Update course button, the course should be updated on API.

```
Swagger:
url: "/courses/:id",
method: "PUT".
```

A returned from API course should be added to **store** as updated course.

When user clicks the Create author button:

New author should be sent to back-end.

```
Swagger:
url: "/authors/add",
method: "POST".
```

- Save the response author result to the **store**.
- New author should be in the all authors list.

#### Add new course

Implement new features for CourseForm component:

- CourseForm should be displayed only for user with a role **ADMIN**.
- Use PrivateRoure for /courses/add route.
- When user clicks the Create course button:
  - New course should be sent to back-end.

```
Swagger:
url: "/courses/add",
method: "POST".
```

- Save the response course result to the **store**.
- Navigate to the Courses component.
- New course should be in the courses list.
- When user clicks the Create author button:
  - New author should be sent to back-end.

```
Swagger:
url: "/authors/add",
method: "POST".
```

- Save the response author result to the **store**.
- New author should be in the all authors list.



You do not have to generate **id** for new course and author.

API generates id for new course and author automatically.

Models of course and author info must be as bellow:

```
// Course
{
   title: string
   description: string
   creationDate: string
   duration: number
   authors: [authorId]
}

// Author
{
   name: string
}
```