React Task 3 - State management, Redux

This is the third part of the application.
In this assignment, you should continue to work on your project with a slight modification.
This module is designed to teach how to work with state management in React and what is Redux.

Prerequisite:

Firstly, you need to install the Redux and React-redux modules in your project using npm:

1. Go to your project directory
2. Install redux and react-redux

```
npm i redux react-redux --save
```

3. Run your project

```
npm run start
```

Project structure requirements:

1. Create new folders for `redux` where put the code for redux `src/store`.
2. Create new file `services.js`.
3. Folder structure `src/store`:

```
src
|-- store
|   |-- index.js // Add store creation, root reducer
|   |
|   |-- user
|   |   |-- reducer.js // Code with reducer for user
|   |   |-- actionCreators.js // Code with actions
|   |   |__ actionTypes.js // Code with action types
|   |
|   |-- courses
|   |   |-- reducer.js // Code with reducer for courses
|   |   |-- actionCreators.js // Code with actions
|   |   |__ actionTypes.js // Code with action types
|   |
|   |__ authors
|       |-- reducer.js // Code with reducer for user
|       |-- actionCreators.js // Code with actions
|       |__ actionTypes.js // Code with action types
```

```
    |
    |-- servisces.js
    |
    |__ ...
```

3. Store should have model where `user`, `authors` and `courses` should have own reducer:

```
    const store = {
        user: {
            isAuth: boolean, // default value - false. After success login - true
            name: string, // default value - empty string. After success login - name of user
            email: string, // default value - empty string. After success login - email of user
            token: string, // default value - empty string or token value from localStorage. After success login - value from API /login response. See Swagger.
        },
        courses: [], // list of courses. Default value - empty array. After success getting courses - value from API /courses/all response. See Swagger.
        authors: [] //  list of authors. Default value - empty array. After success getting authors - value from API /authors/all response. See Swagger.
    }
```

4. APIs from SWAGGER for Module 3:

- `courses/all`[GET];
- `authors/all`[GET].

Criteria (15 points max)

APIs

- [2 points] - All functionality with back-end should be in `services.js`.
  (Create **api service** for each endpoint that you use and call methods or functions from service into your components.)
- [2 points] - Get courses from back-end. See API `/courses/all` in the SWAGGER. Save courses list to the store.
- [2 points] - Get authors from back-end. See API `/authors/all` in the SWAGGER. Save authors list to the store.

Store

- [2 points] - Store should have **User** reducer.
  User reducer manage data with user's info and should have next model:

```
 const userInitialState = {
     isAuth: boolean, // default value - false. After success login - true
     name: string, // default value - empty string. After success login -
name of user
     email: string, // default value - empty string. After success login -
email of user
     token: string, // default value - empty string or token value from
localStorage.
     // After success login - value from API /login response. See Swagger.
 };
```

User reducer has logic:

- After success login `isAuth` property have value `true`, save token, email and name.
- After logout `isAuth` property have value `false`, token, email and name have value as empty string.
  Clean localStorage.

- [1 point] - Store should have **Courses** reducer.
  Courses reducer manage data with courses list and should have next model:

```
   const coursesInitialState = [] // default value - empty array. After
success getting courses from API - array of courses.
```

Courses reducer has logic:

- Save new course.
- Delete course.
- Update course.
- Get courses. Save courses to **store** after getting them from API. See Swagger `/courses/all`.

- [1 point] - Store should have **Authors** reducer.
  Authors reducer manage data with authors list and should have next model:

```
  const authorsInitialState = [] // default value - empty array. After
success getting authors from API - array of authors.
```

Authors reducer has logic:

- Save a new author.
- Get authors. Save authors to **store** after getting them from API. See Swagger `/authors/all` .

Courses Component

- [1 point] - Get courses from store and render them into **Courses** component.

CreateCourse component

- [2 points] - After saving course in **CreateCourse** component course should be added to **store**.

CourseCard Component

- [1 point] - Add a new button "Delete course" into CourseCard. *(**CourseCard** in COMPONENTS)*.
- [1 point] - After clicking on the `Delete course` button a selected course should be deleted from **store**.

> Please find detailed description of components and functionality in the file COMPONENTS