**Task for React CDP program task description**

*Create SPA using React.*

**Necessary tools:**

React

React-Router

PropTypes

Redux

redux-thunk/redux-saga

[**react-bootstrap**](https://github.com/react-bootstrap/react-bootstrap) **or material-UI**

————

jQuery or DOM selectors is forbidden

**Steps:**

1) Create pages with statically content.

Pages:

- Login – page for log in

- Courses – page with a list of courses

- Add/Edit Course – page add/edit course profile

Use Components from Bootstrap.

2) Define routes for pages, using React-Router.

Page ‐ Route:

- Login ‐ /login

- Courses ‐ /courses

- Add/Edit Course

- Add ‐ /courses/new

- Edit ‐ /courses/12 (12 – course id)

The fallback-url should redirect to Courses page otherwise to login page.

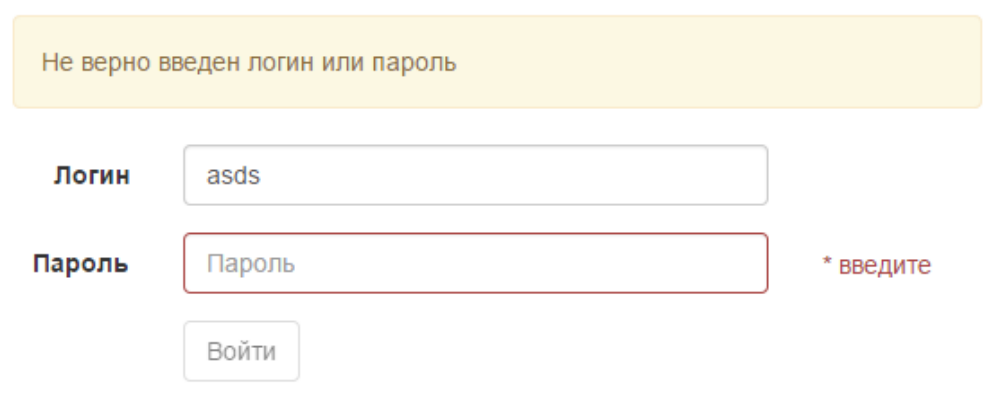
3) Create Login page

Validate fields in blur event. Red border should be added to field and red label with text “«\* enter»” should appear after the field.

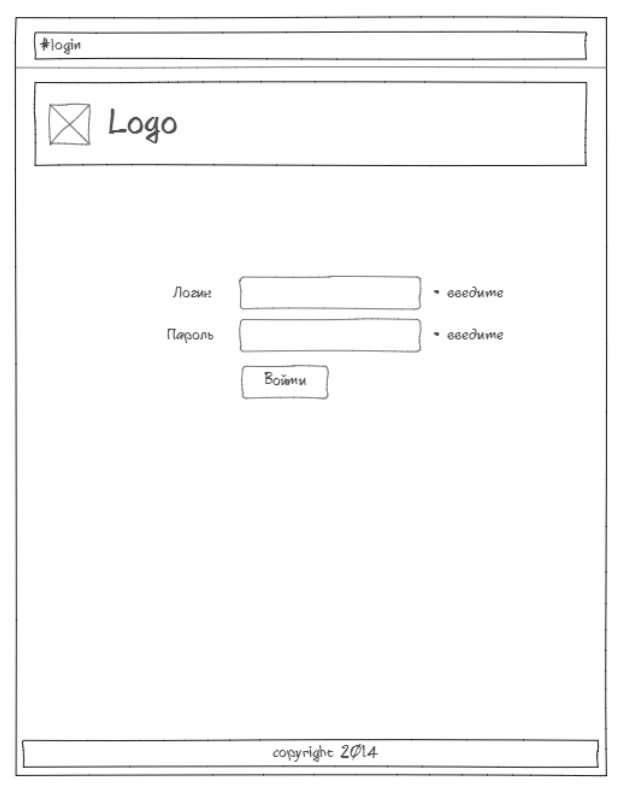


Validation:

* If field is empty please show “Please enter the <field title>” should be disabled.
* Field «Login» should contain only Latin letters. Field “Password” should contain only Latin letters and digits.
* If user enters incorrect login or password the message “Wrong login or password” should appear above the login field, field password should be empty.



After entering correct data into login and password fields, app redirects user to the page courses.



4) Page Courses

Page displays all courses. Work with courses is like REST service. Create course-API-service for get/create/save/delete courses. Get the list of courses or particular course (get), create course (post), update course (put), delete course (delete).

Course has fields: title, createDate, duration (in minutes), list of authors.

Show the user login and link logout in the header of the page. When user clicks logout, he logs out the application and app redirects him to login page.

When user enter something in the field search and clicks button «Search» the list of courses is filtered by title.

When user clicks the button «Add Course» he is redirected to the page /courses/new.

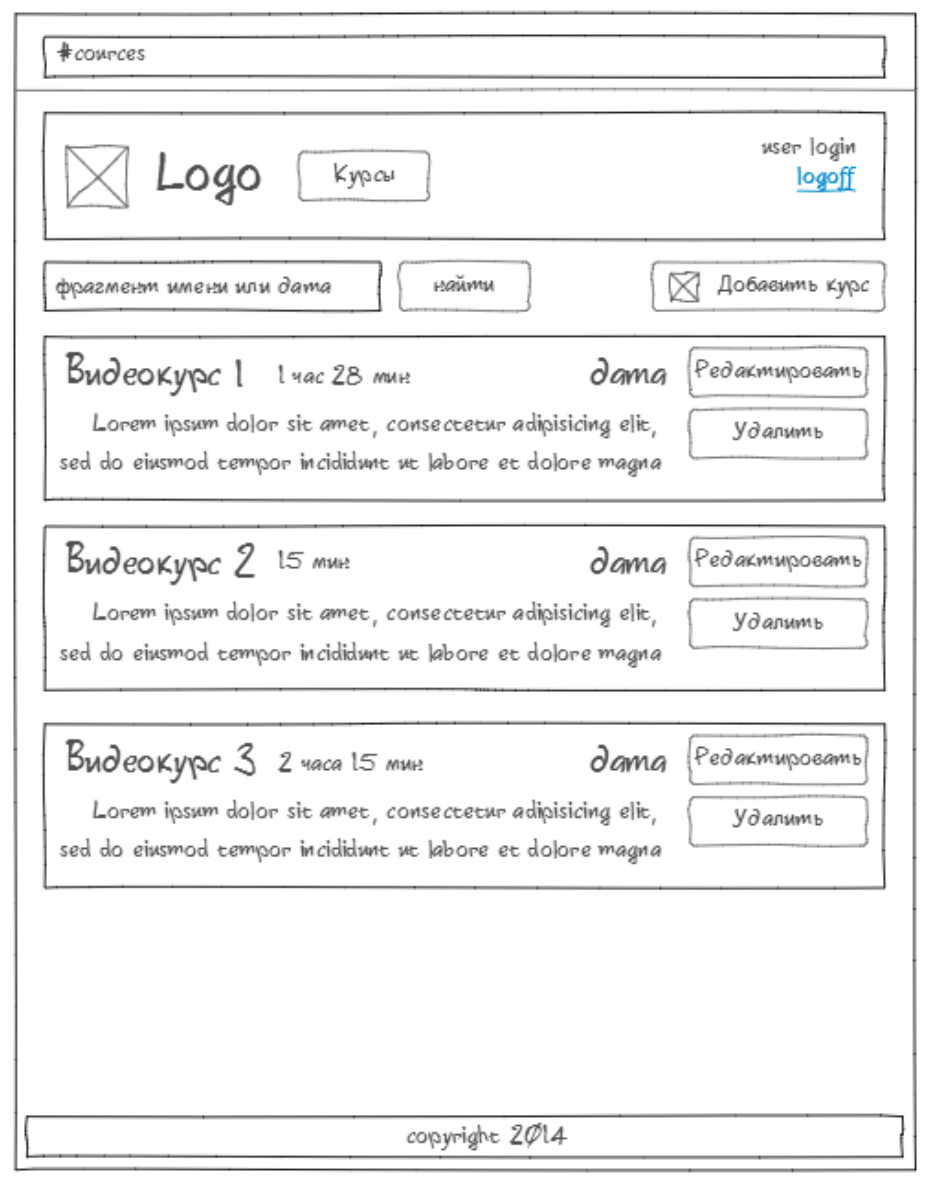
When user clicks the button «Edit Course» he is redirected to the page /courses/:courseId

When user clicks the button «Delete», show the confirmation custom modal window. Delete course when delete is confirmed.

Show the list of authors below the description of the course, separate them with comma.

The duration of course in the following format «1 Hour 30 min».

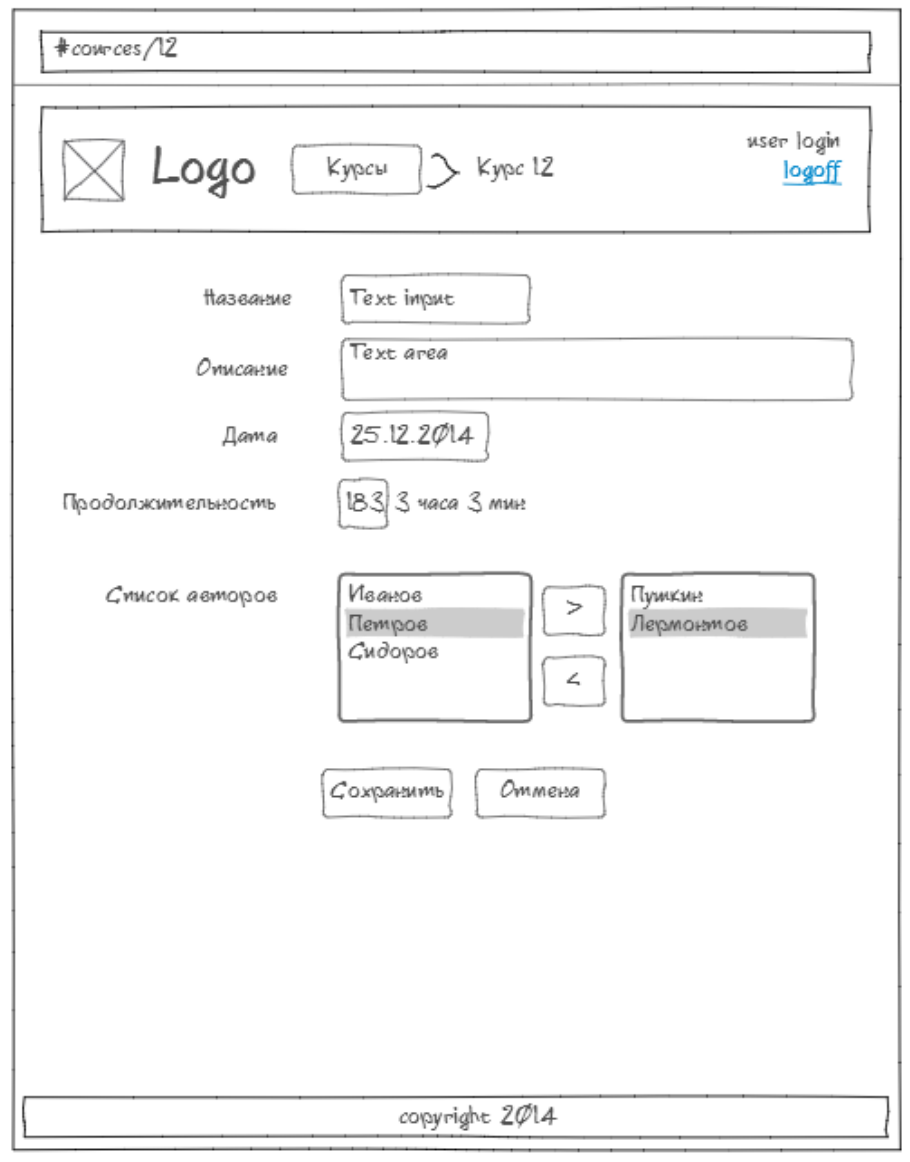
Format for «Date» «dd.mm.yyyy»



5) Page Course

Page for creating and editing course. If course id equals “new”, it is creating of new course otherwise editing course. For getting and saving changes use course-API-service (from step 4).

* Breadcrumbs «Courses‐> Course 12» should appear in the header. «Courses» is a link to the list of courses page.
* When Course title is changing the title in breadcrumbs should be changed too.
* All fields are **required**.
* Field «Date» is HOC which allows to enter only digits and dots according to the following format dd.mm.yyyy. (\d\d.\d\d.\d\d\d\d).
* «Duration» is Render-Prop Component (is applied to text field and on changing transform the number to Hours and Minutes and display them after the field).
* If fields are incorrect and user clicks the button “Save”, the message about error should appear in modal window.
* When user clicks the button «Cancel» he is redirected to the courses page and all changes should be lost.



6) Access to the pages.

If the user is logged in he can access all pages, otherwise he should be redirected to the login page.

**Tasks by weeks:**

1. Create file structure, components and static markup.

Must have requirements (-2 point for each):

* You are using webpack or create-react-appfor creating your project
* No code duplication in JSX and JS
* App state including only serialized data and not mutating the state
* You have common components and “feature” components
* Components props validated with PropTypes and PropTypes.object omitted

Minor requirements (-0.5 point for each)

* Consistent code style

1. lifecycle-hooks, HOCs, React-Router

Must have requirements (-2 point for each):

* You have no side-effects in View components
* You use proper lifecycle-hooks
* Router has redirection to 404 page
* Routings are nested
* You protect routes by HOC

1. App State, Redux Store

Must have requirements (-2 point for each):

* Reducers is clear and has no side-effects
* You use redux-thunk for side-effects
* You has different reducers for different business layers
* Common components are free from store
* Your state is “Single Source of Truth”, there is no state duplication

1. Performance, reselect, shouldComponentUpdate.

Must have requirements (-2 point for each):

* You use selectors to access store
* You memoize proper selectors

- All business logic placed inside HOCs

* Your courses video shows only visible elements (via infinity-list)

1. Unit testing

Must have requirements (-2 point for each):

* Your Selectors covered with tests
* Your Actions/Reducers covered with tests
* Your common components covered with tests
* Your tests independent from each other
* Tests only cover public interfaces for Components