### **Single Page App with React.js (6+)**

**6th work** - Landing (Home) page with React.js

**7th work** - Items list (Catalog) page. React Routing

**8th work** - Item page. Adding interactivity to a website.

**9th work** - Making everything work. (connecting to REST API)

**Bonus works:**

**10th work** - Cart page (Items list). First look at Redux

**11th work** - Cart page (Shipping form) with validation (+ Formik)

**Description**: For all of this works, you have to create a SPA(Single Page Application) E-commerce platform with the following pages: Home, Catalog, Item, Cart, using React.js library and your REST API.

*Note*: You probably have to extend your existing backend app with a bunch of new features, so choose backend tech stack *wisely!*

**General requirements for all next works**:

* The backend can be implemented from scratch, but we recommend extending the project you used for 3-5 works.
* All of the pages have to be made with React.js and all interaction should work properly (without many visible issues)
* **Crossbrowserness / Responsiveness** are not required, but at least latest versions of Chrome should be supported (so we can check your works on our systems)
* **UX**: All pages should stick to the following wireframes. Design and layout of some elements is totally up to you, but if the element is presented on the wireframe, you should implement it as well  
  [**https://wireframepro.mockflow.com/view/lviv-iot-react-app**](https://wireframepro.mockflow.com/view/lviv-iot-react-app)
* **UI (Design, css techniques etc.):**

Is **totally** up to you (external CSS/SCSS files / styled-components / react-bootstrap / MaterialUI / etc.)  
But **choose** at least **one**.

A website example (from Live Coding):

***Coming soon...***

### 

### **6. React.js: Home page**

**Description:** Start creating your React App with a simple Home page (see the link to wireframe above). Your e-commerce app subject is about your **entities from previous (3-5) works.**

***Variants -*** *(products that you are ‘selling’) the same as for previous works. (see the description to 3rd work)*

**Requirements:**

* You have to use only React.js library for all of your mark-up. Which means - your index.html file shouldn’t be touched.
* Use **create-react-app** tool for creating base of your project  
  <https://github.com/facebook/create-react-app>
* Your Home page should **follow the wireframe,** i.e all of the elements ( header, navigation, footer etc. ) from wireframes **must be** presented on your website.
* **Design:** You have to use **CSS** styling or/and component libraries. Don't overthink it, use your imagination for UI, BUT the work with almost no styling is unacceptable.
* **Functionality:** For this work nothing but view only part is required. Any interaction (links/buttons) is not necessary, but you will have to complete them in your next labs.
* **Code style**:
  + Project structure: Your UI elements should be logically separated into React components (one file for each component) - at least **5 component files are required.** In other words, you can’t just put all your JSX into 1-2 files.
  + Use **Functional components** instead of **Class components**  
    <https://uk.reactjs.org/docs/components-and-props.html>

**LIVE CODING LINK (YouTube video)**

<https://youtu.be/aEp5Ssg8m6c>

### 

### **7. React.js: Catalog page**

**Description:** Continue work on your React App by adding a page with Items list (see the link to wireframe of **Catalog page** above).

***Variants -*** *(products that you are ‘selling’) the same as for previous works. (see the description to 3rd work)*

**Requirements:**

* All of the requirements for previous React.js works should be kept.
* **Code style**:
  + Use **array.map()** method for rendering your items list
  + Routing (switching between pages) should work now.   
    Use **react-router-dom** library: <https://reactrouter.com/web/guides/quick-start>
  + All UI elements (buttons / select) should have corresponding React components (PrimaryButton.jsx / Select.jsx etc.)
* **Functionality** (filter / search / view more) is still **not required** (you have to complete it on next works)

### 

### **8. React.js: Item page**

**Description:** Continue work on your React App by adding a page for your Item (see the link to wireframe of **Item page** above). Also, now, you have to make all your previous pages (Home & Catalog) **more interactive**.

***Variants -*** *(products that you are ‘selling’) the same as for previous works. (see the description to 3rd work)*

**Requirements:**

* All of the requirements for previous React.js works should be kept.
* **Code style**:
  + Your **items** should be stored inside the **state or context (your choice)** of your page  
    <https://uk.reactjs.org/docs/hooks-state.html>  
    <https://uk.reactjs.org/docs/hooks-reference.html#usecontext>
  + For your state management use **useState()** inside **Functional Component**  instead of ***this.state*** *and* ***Class component***
  + *If you decided to use context****,*** use **useContext()** hook instead of **Context.Consumer**<https://www.robinwieruch.de/react-usecontext-hook>
* **Functionality (IMPORTANT):**
  + **Home page: “**View more” button should display more elements on the same page *Tip:* Elements can be just random paragraph & heading, use your imagination ;)
  + **Catalog page:** You should be able to filter your items list, by applying different filters by item's properties (i.e size/color/type)
  + **Catalog page:** Search by any text property option should also work
  + **Catalog & Item pages:** “View more” action on every item should refer to corresponding **Item page**, with correct information about item (get the info from your **state/context**)

### **9. React.js: Connecting to REST API**

**Description:** Finally! Now, you are about to put a final touches on all pages you created - implement interaction with your REST API server.

***Variants -*** *(products that you are ‘selling’) the same as for previous works. (see the description to 3rd work)*

**Backend** - just as discussed before, can be the one you used for your **3-5 work** or a new one created **from scratch**. Tech stack - **absolutely up to you.**

**Requirements:**

* All of the requirements for previous React.js works should be kept.
* **Code style**:
  + For any **http request -** use **axios** library  
    <https://github.com/axios/axios#installing>
  + All your API functions should be separated into **single** **file (or folder, if you want) -** just like you saw in Live coding for 5 lab with **fetch()** function
* **Functionality:** 
  + On **Catalog Page** - all items should now be fetched from your backend with GET method (using axios)
  + **Search with filters -** should also be implemented with GET request (*search by text field can be left as it is*)  
    *Hint: pass filters as* ***url parameter***
  + Before response from your GET method is received you have to display a **Spinner(Loader component)** to the user. Something like this: <https://projects.lukehaas.me/css-loaders/>

### **10. React.js: Redux: Cart page (shopping cart)**

**Description:** You are on your way to finishing this insane project… Create the **first of three** cart pages - **Shopping cart page.**

Also, here you meet one of the most popular React library - **Redux.**

***Variants -*** *(products that you are ‘selling’) the same as for previous works. (see the description to 3rd work)*

**Requirements:**

* All of the requirements for previous React.js works should be kept.
* **Functionality:**
  + **Item page:** “Add to cart” action should be implemented using Redux flow: when you add an item to cart, it should be added to your **redux store.** On **Cart page** you takeall of the items from the **store**
  + **Cart page:** “add/remove” actions should be implemented through redux actions & reducers as well.
* **Code style**:
  + Redux: All Redux parts (actions / reducers / store) should be kept in separate and specific files (actions.js / reducers.js / store.js etc.)
  + Use **useSelector** hook for getting the data from **redux store** (instead of **connect()** function)<https://react-redux.js.org/api/hooks#useselector-examples>
  + Use **useDispatch** hook for dispatching your actions (instead of **connect()** function)  
    <https://react-redux.js.org/api/hooks#usedispatch>

### **11. React.js: Formik: Cart page (Checkout & Success)**

**Description:** Finish your project by creating the last **of three** cart pages - **Checkout & Success pages.**

As a bonus, you will learn a very handy and powerful form validation library - **Formik.**

***Variants -*** *(products that you are ‘selling’) the same as for previous works. (see the description to 3rd work)*

**Requirements:**

* All of the requirements for previous React.js works should be kept.
* **Functionality:** 
  + **Form:** Your form should have at least **5 fields**
  + **Form:** Every field should have a validation rule (i.e max length / no special characters / only numbers), just **required** option - is not enough
  + **Form:** You should have at least one field that uses **RegEx** and at least one field that doesn’t required **string** value(i.e Phone number)
  + **Form:** In the error message you should describe **all errors in all fields with clear reason (**i.e Email is incorrect, First name is a required field...**)**
  + **Form: Error message** should be a separate React Component
  + After successful form submit - the user is redirected to final page (**Success page**)
* **Code style**:
  + Use **Formik & Yup** libraries or any functionality related to form validation (fields validating, error messages, form submit)  
    <https://youtu.be/3sXYK60T6Us?t=390>

**You did it, you …**



**My congratulations to everyone that has passed all this way by himself!**

**You did a lot of work and I am really proud of you, my children.**

*“Student life is the seed of your life,*

*Plaint it wisely…”*