## תרגיל 4

### React advanced + simple Spring backend

* נביאים:
* <https://classroom.github.com/a/Ke_DgSzD>
* תאריך הגשה: 10/6/2024 23:59
* שטראוס נשים:
* <https://classroom.github.com/a/wcK--gkq>

תאריך הגשה: 13/6/2024 23:59

* שטראוס גברים:
* <https://classroom.github.com/a/4hgsHG4q>

תאריך הגשה: 10/6/2024 23:59

בתרגיל זה תבנו אתר להרכבת פיצה ושליחת הזמנות: frontend ב-React, ו-backend rest API (Spring) לניהול תוכן ושמירת הזמנות.

A pizza consists of a list of ingredients chosen by the user.

An order consists of:

1. A code (some unique random number/code generated automatically server side)
2. A user first and last name (non-empty)
3. An address (street, house number, city), all fields required
4. A phone number (non-empty, 10 digits)
5. A list of pizzas (non-empty)

## Frontend (React):

1. The home page should allow either starting a new order, or check and order based on a code that the user must provide (the same code displayed at the end of the process of ordering)
2. Building a pizza: a pizza must include at least 2 ingredients. If the user interrupts a pizza building (for example by switching to the cart), the pizza selected ingredients should be saved when going back to pizza building.
3. Price computation of the pizza up to you (can be as simple as a   
   fixed price + number of ingredients \* some\_fixed\_price)
4. A pizza can be added to the cart.
5. The cart and its contents can be viewed (each pizza and its ingredients) with the total price.
6. You should allow editing a pizza in the cart (changing ingredients, or deleting the pizza from the cart)
7. You will allow placing the order if the cart is not empty: of course no payment here, but before placing the order you will require the user to fill in details: first and last name, full address (street, house number, city) and phone number all required non empty.
8. Upon placing the order (saving the order on the backend) the order code generated by the server should be displayed. At this point, the user can return to the home page and the cart will be emptied (the user can start a new order).
9. To check and order, the user enters a code, and all information should be displayed: pizzas with ingredients and full user details. If the code is wrong, you should display a relevant error message.

### Implementation Details

1. The client should load the ingredients from the backend (name and image)
2. The cart may be implemented locally in React (no need to store the cart in cookies or session). Therefore, reloading the page will empty the cart.
3. The frontend is a full SPA (no page reloads!)
4. Implement navigation should be implemented with browser-router: such as building/editing a pizza, showing the cart, entering user details, check order. You are free to design the UI, avoid mixing different tasks in the same route (for example don’t implement the pizza building and cart display in the same route).
5. Avoid storing data in the history (passing parameters to the browser-router navigate function). Instead use states/props/context.
6. Avoid props drilling and make use of all studied React hooks where appropriate: custom hooks, useEffect, Context, Reducer.
7. Use cookies to store user details (you may use a third-party library such as js-cookie) so that details are pre-filled when displaying the user details form next time the user reloads the page or starts a new order right after placing an order.
8. You should get authorization before using a library not part of the course material and examples.
9. Create a responsive and pleasant design (leave spaces, few colors, light colors, consistent UI, small same size ingredients images, keep it simple). Store images in your project (no external URLs).

## Backend (Spring):

The backend is a very simple REST api with basic read/write operations on data handled in memory (similar to the course examples). The goal is to get you started with Spring and combining a backend with a React frontend.

1. You will define at least 6 fixed ingredients of a pizza.
2. You will provide a REST API
   1. To read the available ingredients.
   2. To save orders from different users.
   3. To read an order.

### Implementation Details

1. You should define several data classes for your Rest API, such as classes to hold the list of ingredients and handle orders.
2. Handle critical sections if any.
3. The data is not persistent, no use of Database, Java Beans or Session is expected in this exercise, just simple in-memory data management. As a result, restarting the server will reset the orders to none.

**Submit**

* **a Spring repository with a folder inside named “frontend”, containing the React front end. The creation and execution of such structure is demo-ed in week 6 and recording is available** [**here**](https://hac.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=02ad1585-b0ca-4237-8016-b17d00c6be55) **(https://hac.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=02ad1585-b0ca-4237-8016-b17d00c6be55).**Note thatyou can connect a project created locally on your computer with the exercise repo using the menu **Git->Manage Remotes** (enter the <URL> of your repo), or in the terminal with the command: **git remote add origin <URL>**. Make sure to submit the ***default*** branch**.**If you can’t get around git, create the project from the repo as usual (New project from version control), create a second project (Spring boot) and drag all files inside the first.
* **Add a README.md with your details**

We strongly recommend that you develop and test your backend like a pro with Postman before connecting the front end!

Good luck!