Mentor-review Next.js Auth0 React GitHub

Online Shop Deployment

! You need to submit the project for review

Congratulations on reaching the last part of Module 5! In this document, you will find a detailed description of your final individual project in this course. Please read it carefully.

Technical Information

Deadline

We don’t have a strict deadline, but we recommend completing the project in 2 weeks.

Prerequisites

To participate in the project, you must have completed all the lessons in modules 1 to 5. This means mentors should have approved all your home assignments.

Task Description

Goal

Imagine you are already working as a web developer. Refocus hired you to create a proof of concept for its new initiative: an online shop.

> In IT startups, products evolve through several stages: POC, MVP, and regular releases.
>
> **Stage 1: POC.** POC stands for proof of concept. At this stage, developers do not think about security or reliability. They need to prove as fast as they can that the idea behind the product is worth being implemented.
>
> **Stage 2: MVP.** MVP stands for minimum valuable product. At this stage, developers implement the idea with a minimum set of features. The product should meet the customers' needs as easily and cheaply as possible. At the same time, the code should meet all the requirements for quality and safety.
>
> **Stage 3: Regular releases.** By this stage, an MVP has already proved that the idea is valuable, meaning there are first customers. Now, it's time to develop the project, adding new features and improving user experience.

Your ultimate goal is to create and deploy a website based on the ready-made design Refocus provided.

In the online shop, users should be able to:

- Authorize via Google
- Search for products
- Add a product to cart
- Place an order

Requirements

The website should consist of five types of pages.

Page 1: Products. It displays a list of product cards. All products are on one page. There are a lot of them, so place a “Load more” button at the bottom of the page.

Each card contains a product thumbnail with a picture, title, and price. A user can add the product to the cart or remove it and see its amount in the cart.

Page 2: Product details. It displays detailed information about the chosen product. A user navigates to this page by clicking on a product card.

The information contains the product picture, title, description, price, rating, brand, and category. A user can add the product to the cart or remove it and see its amount in the cart.

Page 3: Cart. It displays a list of items in the cart. A user can increase or decrease the number of selected products.

The page contains a button to continue shopping, which leads to the previous “Products” or “Product details” page.

The page also contains a button to place an order. When a user clicks the button, the cart is cleared, the “Success” page is shown, and the order object is displayed on the console. If the user is not logged in, redirect them to authorization before the “Success” page.

If you are struggling with redirection to authorization, there is a simpler solution. Disable the “Place order” button and display the following tip nearby: “To place an order, sign in.”

An order object can look like the following:

```
{
    userId,
    items: [
        { productId, amount }, …
    ],
    currency: {
        name: 'GBP',
        rate: 0.72007
    }
}
```

Page 4: Sign-in. It displays an authorization form. A user authorizes via their Google account. After that, they are redirected to the “Products” page, and their username and Google profile pic are displayed on the website menu.

Page 5: Success. It displays a list of purchased products and is shown after a user places an order.

All the pages should contain the website menu. It should display the following elements:

> **Cart icon.** It has the link to the cart and is displayed on all pages except the “Cart” page.
>
> **Link to the “Sign-in” page.** If a user is already signed in, show their username and profile picture instead.
>
> **Currency dropdown menu.** When clicked, the dropdown menu opens, and a user can choose a price currency. There should be 3 to 5 currencies. The default currency is USD. When a user selects a currency, prices are recalculated.
>
> Fetch currency rates from the Fixer API.

The website layout is responsive and adaptive: users can open it both on desktop and mobile. The number of cards in a row shrinks as the screen size decreases. On a mobile device, there is one card in each row.

Instructions

Step 1: Create and set up a Next.js project

Create a public repository on GitHub.

Create a Next.js project. Build and run it locally using the “npm run build” and “npm run start” commands. Open http://localhost:3000 in your browser.

Commit and push the changes to the repository.

Deploy your project to the Vercel hosting.

> After setting up deployment in Vercel, any push to the main branch automatically triggers Vercel to redeploy a website with the new changes.

Step 2: Explore the design on Figma

Open the design, and investigate its parts carefully: UI, desktop version, and mobile version. Analyze what approaches you may need to use to replicate the design and make the website adaptive.

Step 3: Set up authorization using Auth0

Register on Auth0. If you already have an account, just sign in.

Create a regular web application and select “Next.js.”

Configure the application callbacks for both localhost and Vercel URLs.

Install and configure Auth0 NextJS SDK to your project.

Put “clientId” and “secrets” generated in Auth0 to the “.env.local” file. Remember to add “.env.local” to “.gitignore.”

According to the design on Figma, add the “Sign in” and “Log out” functions. Show user profile information: username and profile picture.

Commit and push the changes to the main branch to trigger Vercel deployment.

Step 4: Implement components

According to the design on Figma, develop the necessary components listed in the table below. Use the DummyJSON public API and its documentation to get the product data. Conduct tests for each component.

Mind that the list of currencies and their rates should be done with static site generation. Currency selection and product price recalculation should be done on the client side as in the usual React component.

Component name	What’s inside
ProductItem	A product card
ProductList	A list of product cards
ProductDetails	A “Product details” page
Cart	A list of selected products for placing an order
Currency	A list of supported currencies

DummyJSON public API

DummyJSON documentation

Endpoint for products: https://dummyjson.com/products

Endpoint for currency rates: https://fixer.io/documentation#latestrates

Assessment Criteria

Criteria	Max. Points
Stack	
Stylesheets are used	2
React components are used	2
The application is deployed with Vercel	2
The project is uploaded to a public GitHub repository	1
Design	
The layout is adaptive to desktop and mobile	2
The news cards look close to the Figma design	2
Post titles and descriptions are cropped	1
Structure	
The main page with product cards is fully developed	2
The product details page	2
The cart page	2
The success page	2
The website menu	2
The sigh-in page	1
Functionality	
The authorization with Auth0 is implemented	2
The “Search” function	2
The “Add product to cart” function	2
The “Place an order” function	2
Maximum Overall Grade	**31**

Documents to Hand In

A Word document with a link to a public GitHub repository containing the React app you developed.

How to upload the project

Create a Word document, insert the link to your GitHub repository, and upload the file here.

refocus