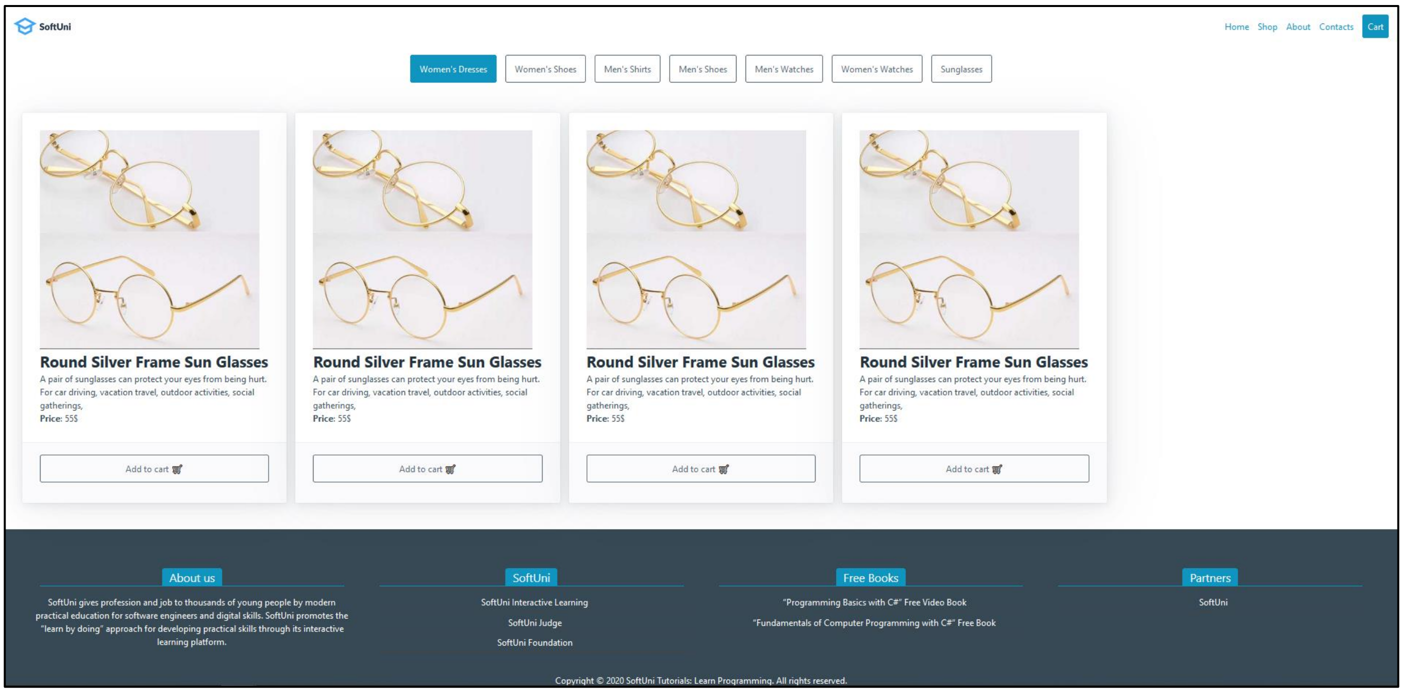
Routing and State Management - Workshop

This application is created as a workshop for the [VueJS Course](#) @ [Software University](#)

Task Requirements

You will need the code from your first workshop – Vue Components. In this workshop we will add more features to the application like routing and state management and create a new Cart page.



Round Silver Frame Sun Glasses
A pair of sunglasses can protect your eyes from being hurt. For car driving, vacation travel, outdoor activities, social gatherings,
Price: 55$
Add to cart

Round Silver Frame Sun Glasses
A pair of sunglasses can protect your eyes from being hurt. For car driving, vacation travel, outdoor activities, social gatherings,
Price: 55$
Add to cart

Round Silver Frame Sun Glasses
A pair of sunglasses can protect your eyes from being hurt. For car driving, vacation travel, outdoor activities, social gatherings,
Price: 55$
Add to cart

Round Silver Frame Sun Glasses
A pair of sunglasses can protect your eyes from being hurt. For car driving, vacation travel, outdoor activities, social gatherings,
Price: 55$
Add to cart

About us
SoftUni gives profession and job to thousands of young people by modern practical education for software engineers and digital skills. SoftUni promotes the "learn by doing" approach for developing practical skills through its interactive learning platform.

SoftUni
SoftUni Interactive Learning
SoftUni Judge
SoftUni Foundation

Free Books
"Programming Basics with C#" Free Video Book
"Fundamentals of Computer Programming with C#" Free Book

Partners
SoftUni

Copyright © 2020 SoftUni Tutorials: Learn Programming. All rights reserved.

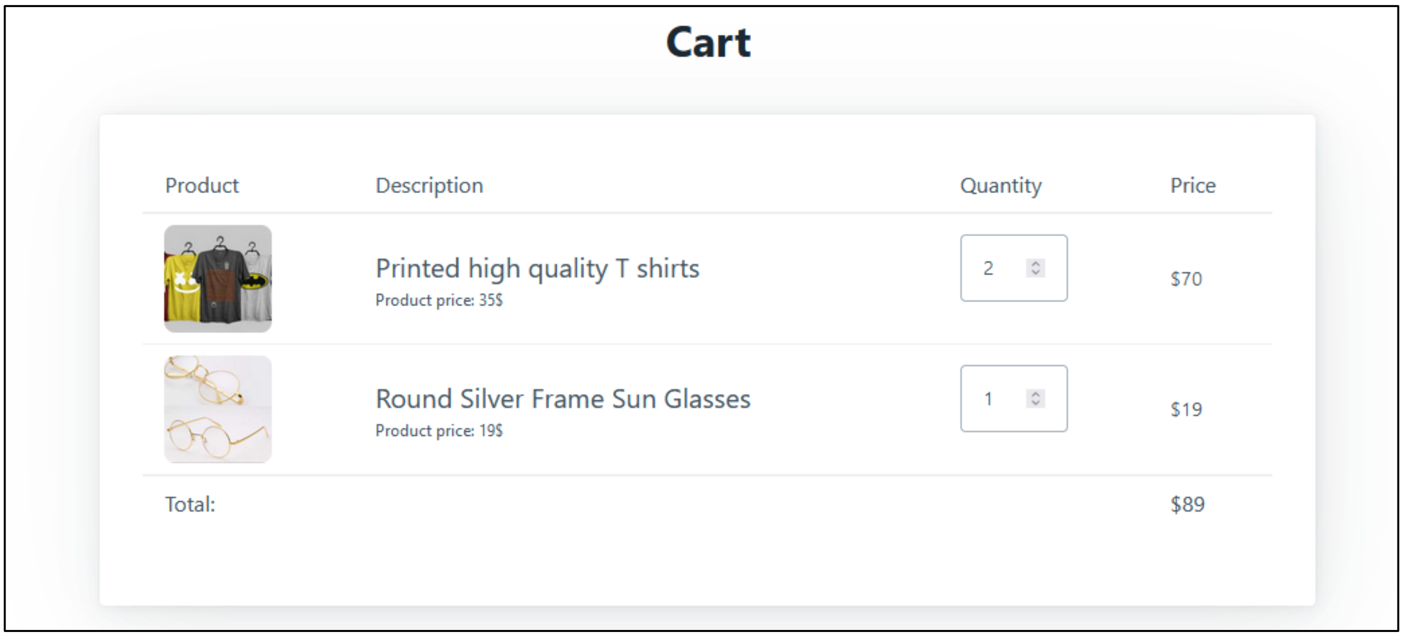
Routing

The **workshop will expect** that in lecture **Directives, HTTP and Routing** you have implemented **Vue Router** and fetching categories and products with **axios** from **dummyjson** - https://dummyjson.com/ in the project.

If this is not implemented, you can still work on this workshop, but you will need to **install and connect Vue Router**.

State management

Install and connect **Pinia** to your application. Then create a "Cart" component from the resources provided with the workshop. Remember to deconstruct the resource to meaningful components and not simply copy-paste it. You are also free to make your own design of the cart with the only requirement to show the name, price, quantity and total price of the product.



Cart

Product	Description	Quantity	Price
	Printed high quality T shirts Product price: 35$	2	$70
	Round Silver Frame Sun Glasses Product price: 19$	1	$19
Total:			$89

When the user clicks on "Add to cart 🛒 " button of a product, that product should be added with a quantity 1 to the cart. If the user clicks the same add to cart again – you need to increment only the quantity in the cart.

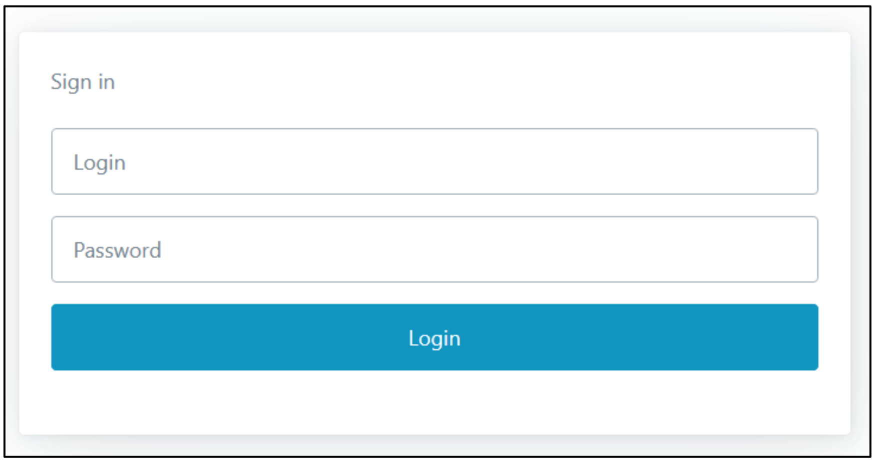
When the user is in the "Cart" page he can also modify the quantity from the input. If a quantity goes to zero or less – remove the product from the cart.

Add a **route guard** to navigate to the cart only if the card is **not empty** and has at least 1 product (1 quantity) in it.

Login page

Create a new **link** in the header and a **page** for login. For the form you need to create a simple email and password fields, like the screenshot bellow. Both fields should have some kind of **validation** to be **required** so you don't make API request without these fields. Your login form should make a request to **dummyjson's Auth** endpoint https://dummyjson.com/docs/auth.

Think about how you will keep the user information and validate if the user is authenticated or not.



Favorites page

After you have a login page, create a new Favorites page that is accessible **only if** the user is **authenticated** (logged in).

For this page use the same grid and products layout from the "Store" page, without the filters. In this page you should **show products** our **authenticated** user has marked as "favorite".

To mark a product as "favorite" add a new button (or icon) to the Product component, that is visible only if the user was **authenticated**. When clicking this button, you should add the product to "favorite" list and show it when we navigate to the page "Favorites".

Think about how we can **remove** an item from our favorite list. Should we see the button to add to favorites in the "Store" page only or in the "Favorites" page too?