

LAB: Product Details

NEXT: LAB: ADMIN FEATURES ↩

To create the details component, we need to add another route to show product details. It's a dynamic route which accepts an ID as parameter and uses this ID to fetch a single product from the server:

```
// You can add this to your existing
// array of routes
{
  path: '/details/:id',
  name: 'Details',
  component: Details
}
```

Then we need to create the component for this route as well:

```
// ./src/pages/Details

<template>
  <div>
    <product-details :product="product" :isAdding="true" ></product-details>
  </div>
</template>

<script>
  import ProductDetails from '../components/product/ProductDetails'
  export default {
    created () {
      if (!this.product.name) {
        this.$store.dispatch('productById', this.$route.params['id'])
      }
    },
    computed: {
      product () {
        return this.$store.getters.productById(this.$route.params['id'])
      }
    },
    components: {
      'product-details': ProductDetails
    }
  }
</script>
```

First, we check if the product already exists before dispatching an action to fetch the product on the server. This is done immediately the component is loaded in the `created` lifecycle.

Table of Contents

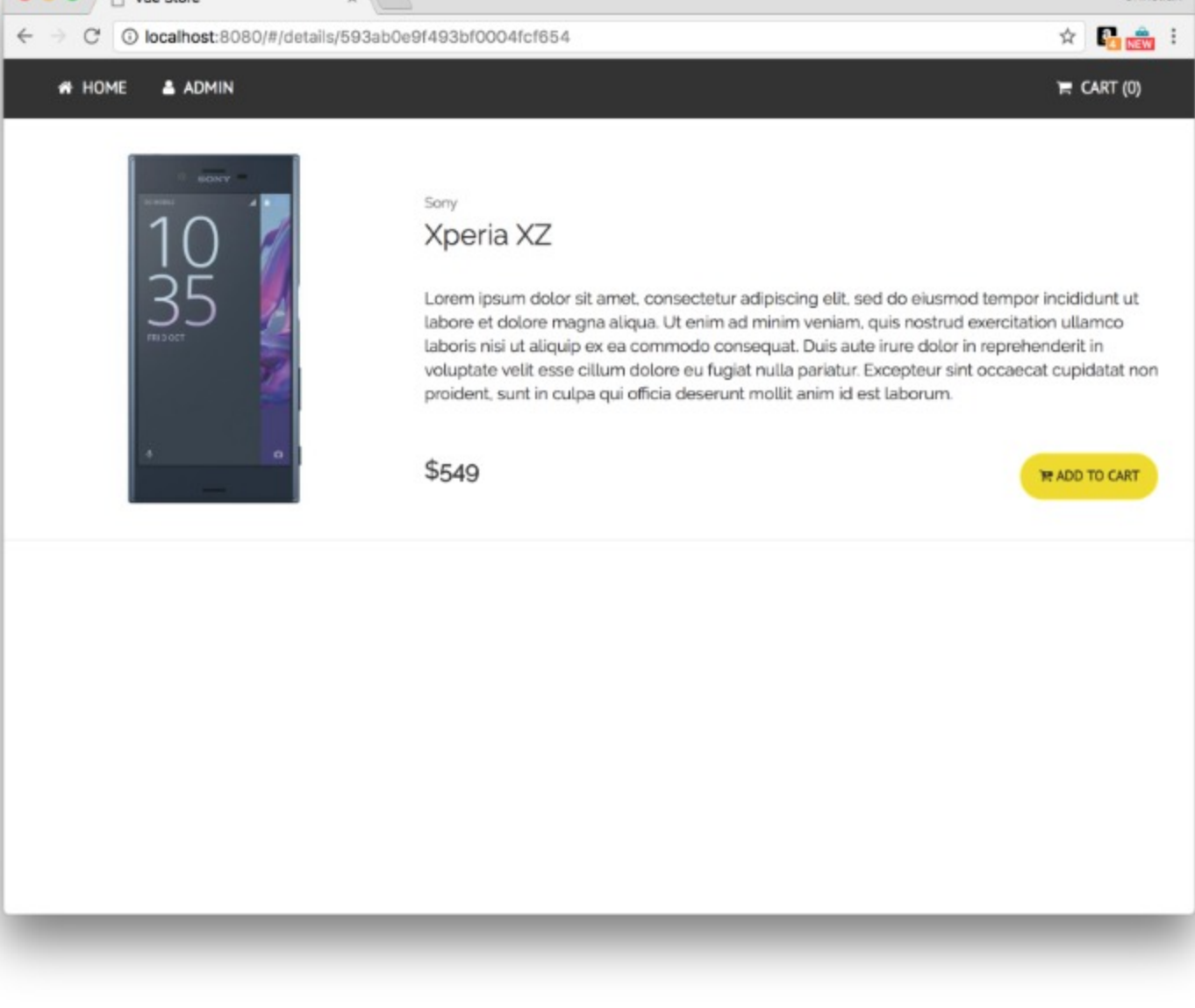
1 Cart Page

As usual, the `product` computed method holds the retrieved value which is received from `productById` getter. The `product` is not displayed immediately, but passed down to `ProductDetails` component. This abstract exists because we could use the same `ProductDetails` in the Cart component.

```
<template>
  <div class="product-details">
    <div class="container">
      <div class="row">
        <div class="col-lg-4 col-md-4 col-sm-6 col-xs-12 product-details_image">
          
        </div>
        <div class="col-lg-8 col-md-8 col-sm-6 col-xs-12 product-details_info">
          <div class="product-details_description">
            <small>{{product.manufacturer && product.manufacturer.name}}</small>
            <h3>{{product.name}}</h3>
            <p>
              {{product.description}}
            </p>
          </div>
          <div class="product-details_price-cart">
            <p>${{product.price}}</p>
            <product-button :product="product" ></product-button>
          </div>
        </div>
      </div>
    </div>
  </template>

<script>
  import ProductButton from './ProductButton'
  export default {
    props: ['product'],
    components: {
      'product-button': ProductButton
    }
  }
</script>
```

Just a UI component that receives a `product` from its parent component via `props` and displays it. It also uses the `ProductButton` to toggle the item in the cart.



Cart Page

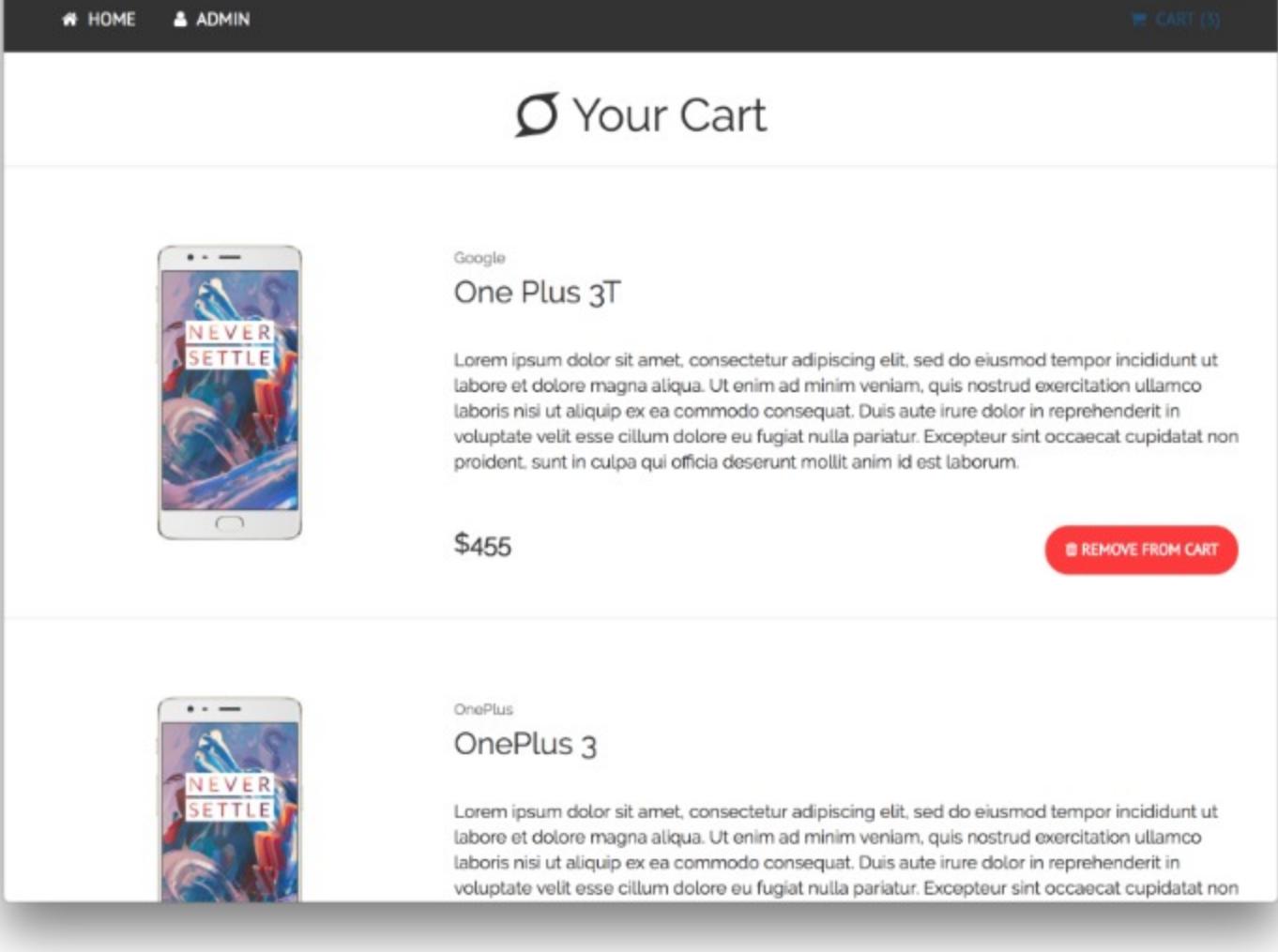
Speaking of cart, let's create a page/component for that:

```
// ./src/pages/Cart

<template>
  <div v-if="cart.length > 0">
    <div class="title">
      <h1><i class="fa fa-superpowers"></i> Your Cart</h1>
    </div>
    <template v-for="product in cart">
      <product-details :product="product" :key="product.id" ></product-details>
    </template>
  </div>
  <div v-else class="title"><h1><i class="fa fa-superpowers"></i> Your Cart is Empty</h1></div>
</template>

<script>
  import ProductDetails from '../components/product/ProductDetails'
  export default {
    data () {
      return {
        cart: this.$store.state.cart
      }
    },
    components: {
      productDetails: ProductDetails
    }
  }
</script>
```

It's basically a variant of `ProductDetails` component; in fact, it re uses the `ProductDetails` component.



Getting Started

- 1 Introduction Free
- 2 Tools Free
- 3 Our Task Free
- 4 Hello World Free

Vue Basics

- 5 Vue CLI Free
- 6 Components and the Vue Instance Free
- 7 Template Syntax Free
- 8 Conditional & List Rendering Free
- 9 Styles & Classes Free

Routing

- 10 Single Page Apps Free
- 11 Creating Routes Free
- 12 Route Outlets & Links Free
- 13 Nested & Dynamic Routes Free

Forms

- 14 Two Way Binding (Reactivity) Free
- 15 Form Validation Free
- 16 Handling Form Submissions Free

API Backend

- 17 Prelude Free
- 18 Setup Free
- 19 Provisioning a MongoDB Database Free
- 20 Enabling CORS Free
- 21 Schemas & Models Free
- 22 Routes and Controllers Free
- 23 Testing with POSTman Free

Vuex

- 24 The Current Problem Free
- 25 State Management Free
- 26 Getters Free
- 27 Mutations Free
- 28 Actions Free

Using Store In Components

- 29 State and Actions on Components Free
- 30 LAB: Product List Free
- 31 LAB: Product Details Free
- 32 LAB: Admin Features Free
- 33 LAB: Spinner, Cart & Store Subscription Free