Task 1:

```
<html lang="en">
<head>
  <script src="https://cdn.jsdelivr.net/npm/vue@2.7.14/dist/vue.js"></script>
  <meta charset="UTF-8" />
  <title>TranDucTin</title>
  <!-- Import Styles -->
  <link rel="stylesheet" type="text/css" href="assets/styles.css" />
</head>

<body>
                                                    <!--TASK 1-->
  <div class="nav-bar">

    <div id="app">
      <div class="product">
      <!--this is part 1-->
        <div class="product-image">
          <img v-bind:src="image">
        </div>
        <!--this is the image part 2-->

        <div class="product-info">
          <h1>{{title}}</h1>
          <p v-if="inventory > 10">In Stock</p>
          <p v-else-if="inventory <= 10 && inventory > 0"> Almost sold out!</p>
          <p v-else>Out of Stock</p>
          <!--this is the in stock out stock part 3-->
          <ul>
            <li v-for="detail in details">{{ detail }}</li>
          </ul>
          <!--this is the detail part 4-->
```

First, I started with an HTML code to structure the webpage. I included the necessary elements such as the head, title, meta tags, and link to the stylesheet. In the body, I created a navigation bar, sections for product images, and an inventory product information section. Each section was carefully designed to ensure a clean and user-friendly layout.

Java code

And then, I proceeded to implement the functionality for the shopping cart using JavaScript. This involved defining an object with methods to

handle various cart interactions, such as adding products to the cart and updating the selected product. The code snippet below illustrates how these methods were structured:

```
var app = new Vue({
    el: '#app',
    data: {
        brand: 'Vue Mastery',
        product: 'Socks',
        selectedVariant: 0,
        image: './assets/images/socks_blue.jpg',
        inventory: 100,
        inStock: true,
        details: ["80% cotton", "20% polyester", "Gender-neutral"],
        variants: [
            {
                variantId: 2234,
                variantColor: 'green',
                variantImage:'./assets/images/socks_green.jpg',
                variantQuantity: 10
            },
            {
                variantId: 2235,
                variantColor: 'blue',
                variantImage:'./assets/images/socks_blue.jpg',
                variantQuantity: 0
            }
```

By integrating these methods, I ensured that the shopping cart functionality was dynamic and responsive to user interactions. This approach allowed for a seamless and intuitive user experience, enhancing the overall usability of the web application.

Task 2:

```
            <ul>
              <li v-for="detail in details">{{ detail }}</li>                    <!--TASK2-->
            </ul>
            <!--this is the detail part 4-->
            <div v-for="(variant, index) in variants"
                 :key="variant.variantId"
                 class="color-box"
                 :style="{ backgroundColor: variant.variantColor }"
                 @mouseover="updateProduct(index)">                 <!--part 6 change color by mouse over-->
              <!--part 7 update product by index-->
            </div>
              <!--this is add to cart part 5-->
            <button v-on:click="addToCart"
            :disabled="!inStock"
            :class="{ disabledButton: !inStock}">Add to cart</button>
            <div class="cart">
              <p>Cart({{cart}})</p>
            </div>

          </div>
        </div>


        <!-- Import Js -->
        <script src="main.js"></script>
</body>

</html>
```

Next, I focused on enhancing the user interaction by incorporating Vue.js directives and event handling. This allowed for dynamic updates and a more interactive user experience. Here's a continuation of the text:

I used Vue.js to manage the state and behavior of the web application. By leveraging Vue directives like v-for and v-on, I was able to create a responsive and interactive interface. For example, I used the v-for directive to loop through the product variants and display them dynamically. Additionally, I implemented event handling to update the product details and change the color of the product image on mouseover.

The following code snippet demonstrates how these features were integrated:

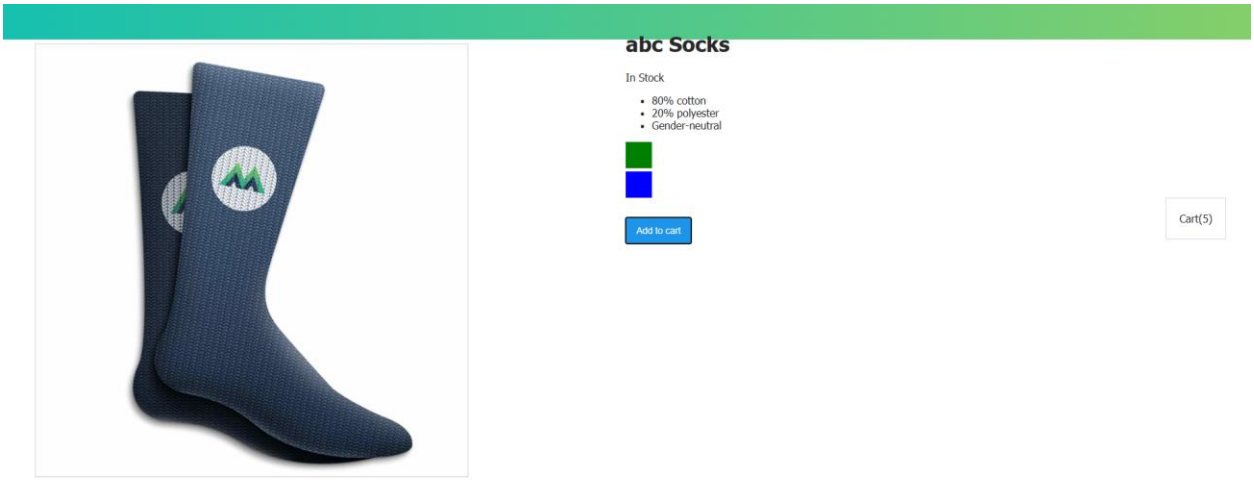
Java code

```
        ],
        cart: 0
    },
        methods: {
            addToCart() {
             this.cart += 1
            },
            updateProduct(index) {
             this.selectedVariant = index
             console.log(index)
            }
        },
        computed:{
            title(){
                return this.brand + ' ' + this.product
            },
            image(){
                return this.variants[this.selectedVariant].variantImage
            },
            inStock(){
                return this.variants[this.selectedVariant].variantQuantity
            }
    }
})
```

Then, I continued to refine the functionality and user experience by adding more interactive features. This included handling mouseover events to change the product color and updating the product details dynamically. Here's a continuation of the text:

To further enhance the user experience, I implemented additional interactive features using Vue.js. For instance, I added a mouseover event to change the product color when the user hovers over different variants. This provided a visual cue to the users, making the interface more engaging.

The Result:



- Finally, I reviewed the overall design and functionality of the web application to ensure everything was working as expected. The result was a polished and user-friendly interface that showcased the product details effectively. The final design included a product image, a list of product features, a color palette, and an “Add to cart” button. Here’s a summary of the final result:
  Product Image: Displayed a dark blue sock with two white circular logos featuring green mountain peaks.
- **Product Features**: Listed the key features of the product, such as “80% cotton,” “20% polyester,” and “Gender neutral.”
- **Color Palette**: Showed the available colors for the product, including dark blue, white, and green.
- **Add to Cart Button**: Provided a clear and accessible way for users to add the product to their shopping cart.

By integrating all these elements, I created a cohesive and engaging web application that provided a seamless shopping experience for the users. The dynamic updates and interactive features ensured that the application was both functional and visually appealing.

Explain:

The reason why I learn this is because I am passionate about web development and creating interactive, user-friendly applications. By learning HTML, CSS, and JavaScript, I can build dynamic and visually appealing websites. Additionally, understanding frameworks like Vue.js allows me to enhance the functionality and user experience of my web applications. This knowledge is essential for my studies in computer science and will help me in my future career as a software developer