Lab 2 Understanding

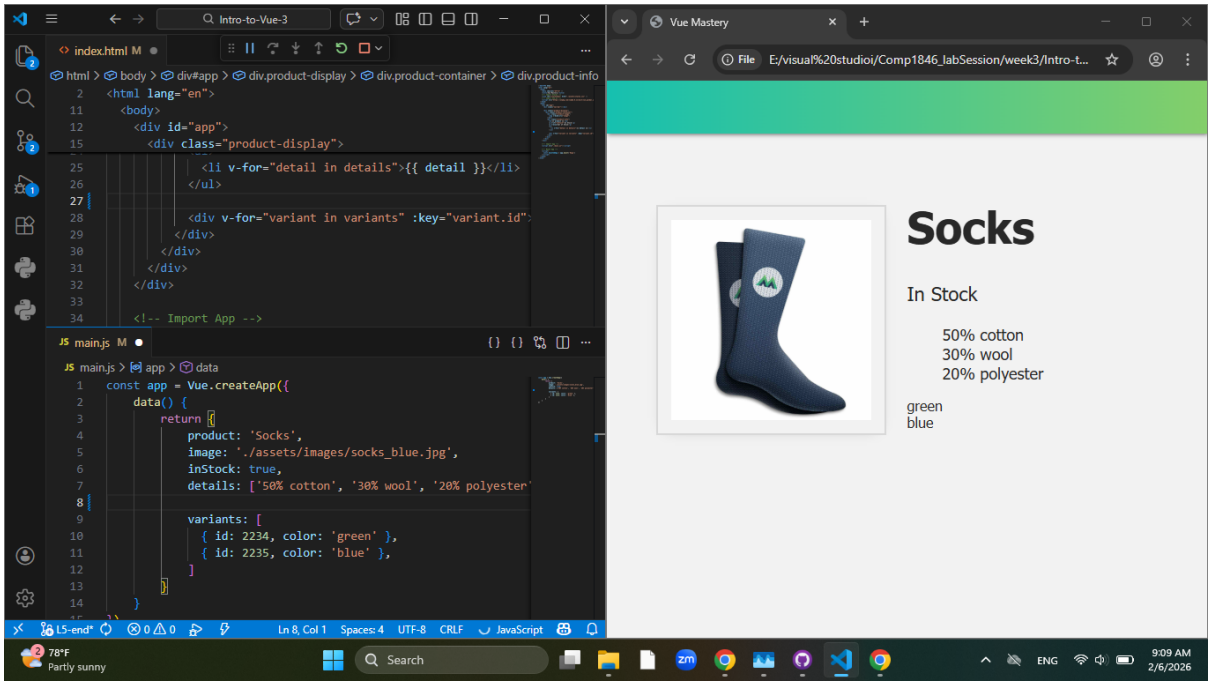
From these lessons, I understand how Vue uses data-driven logic to control both the behavior and appearance of an application. Vue allows the UI to automatically update whenever the underlying data changes, which helps keep everything consistent and reactive without manually manipulating the DOM.

Using v-for, I learned how to render lists from arrays stored in the app's data, such as product details and variant options. Each item in the list is generated dynamically, and the :key attribute plays an important role by giving Vue a way to uniquely identify each element. This improves performance and ensures Vue can properly track changes when items are updated or reordered.

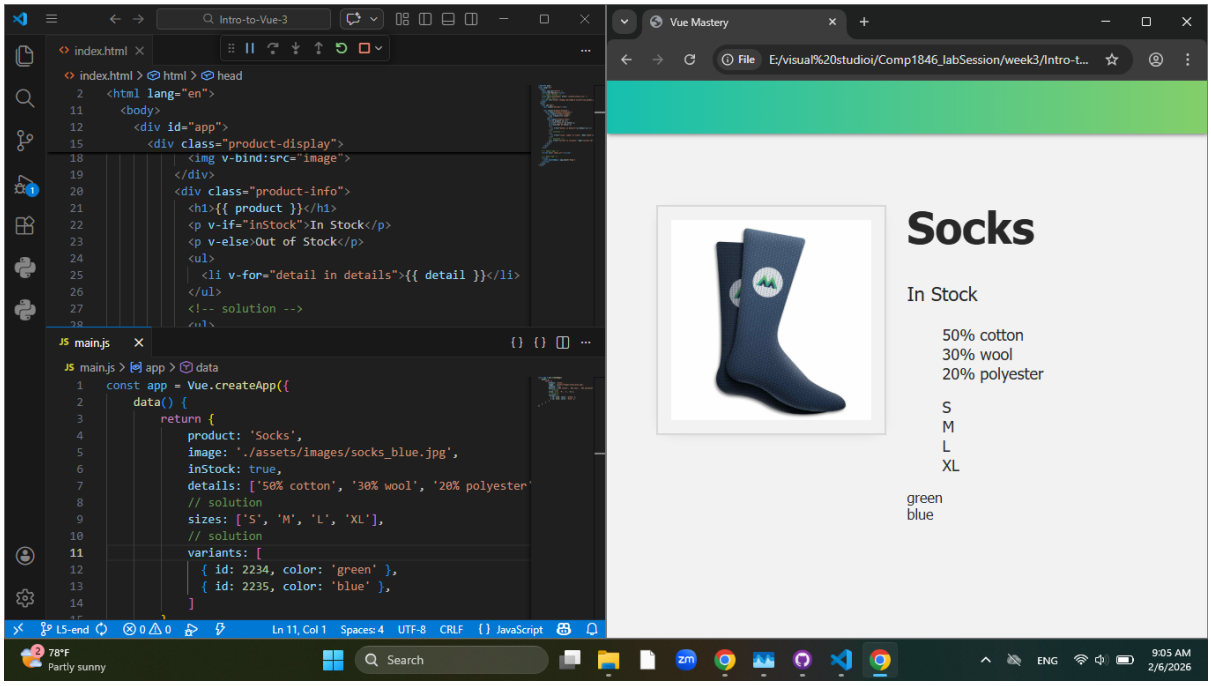
I also learned how event handling works through v-on or the shorthand @. This makes it possible to respond to user actions like clicks and mouseovers. For example, clicking the "Add to Cart" button triggers a method that increases the cart value, while hovering over a product variant updates the displayed image. This interaction makes the app feel more engaging and intuitive for users.

Additionally, class and style binding allow the UI to visually react to data. Style binding lets me dynamically apply CSS styles, such as changing background colors based on variant data. Class binding allows CSS classes to be conditionally applied, like disabling and visually dimming a button when a product is out of stock. Overall, these concepts show how Vue ties together data, events, and styling to create a smooth and user-friendly experience.

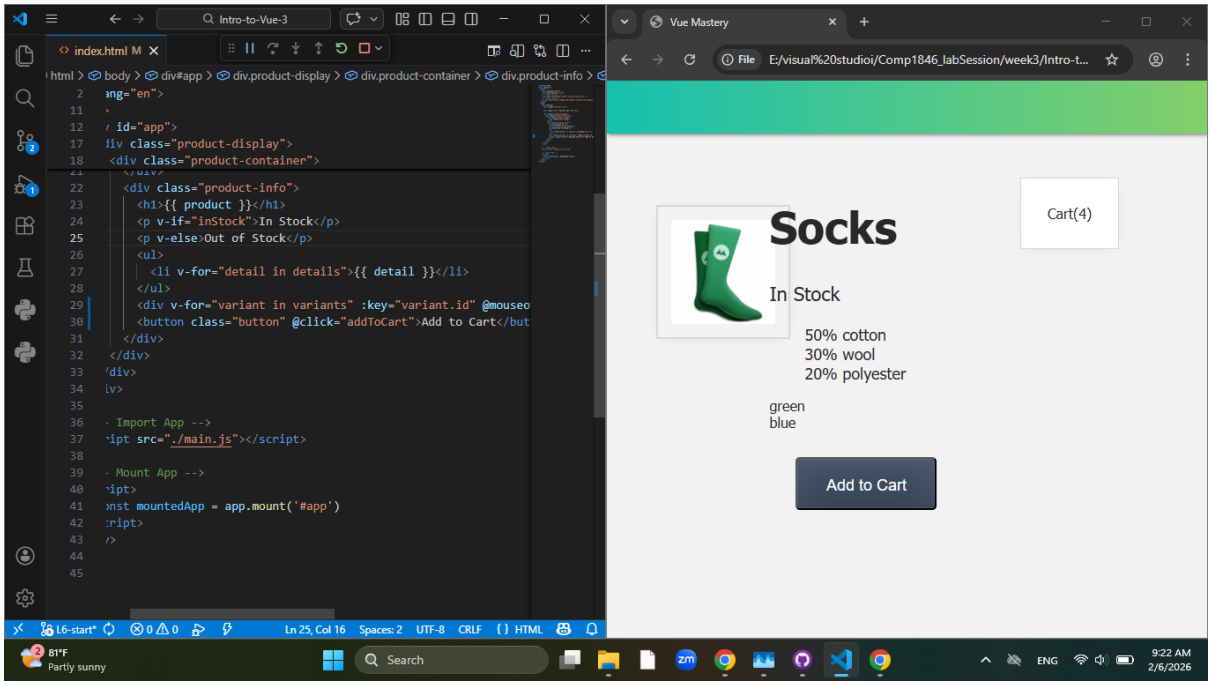
Task of lesson 5 picture:



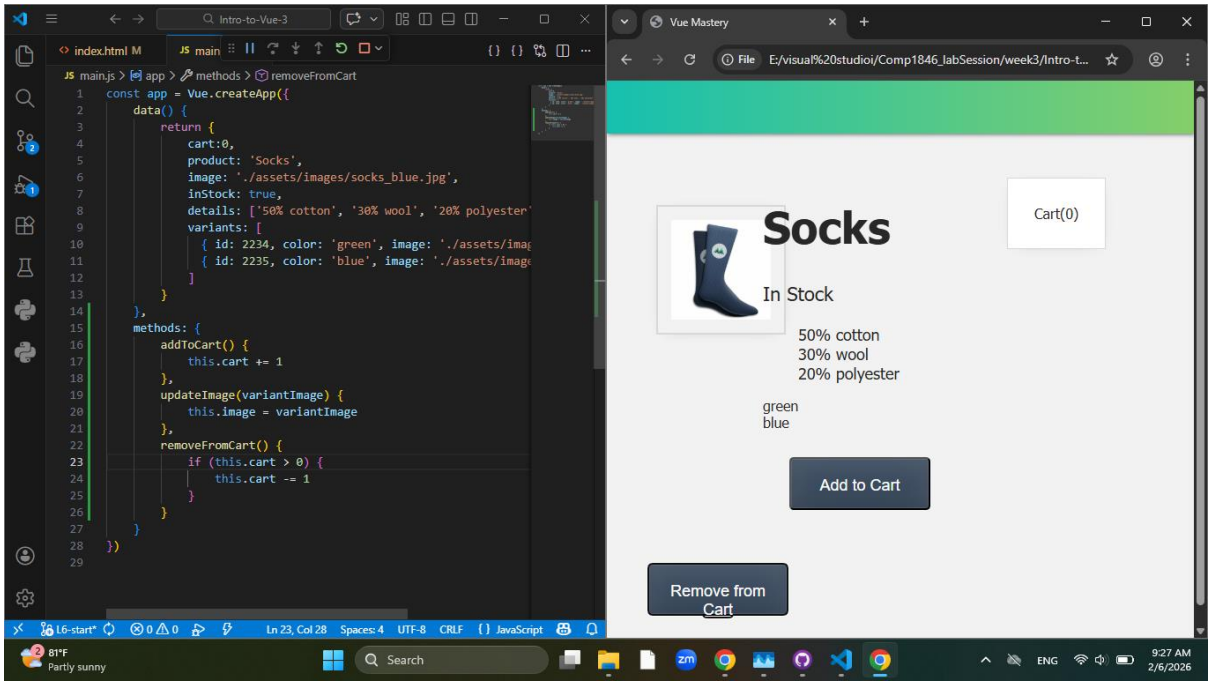
Challenge of lesson 5 picture:



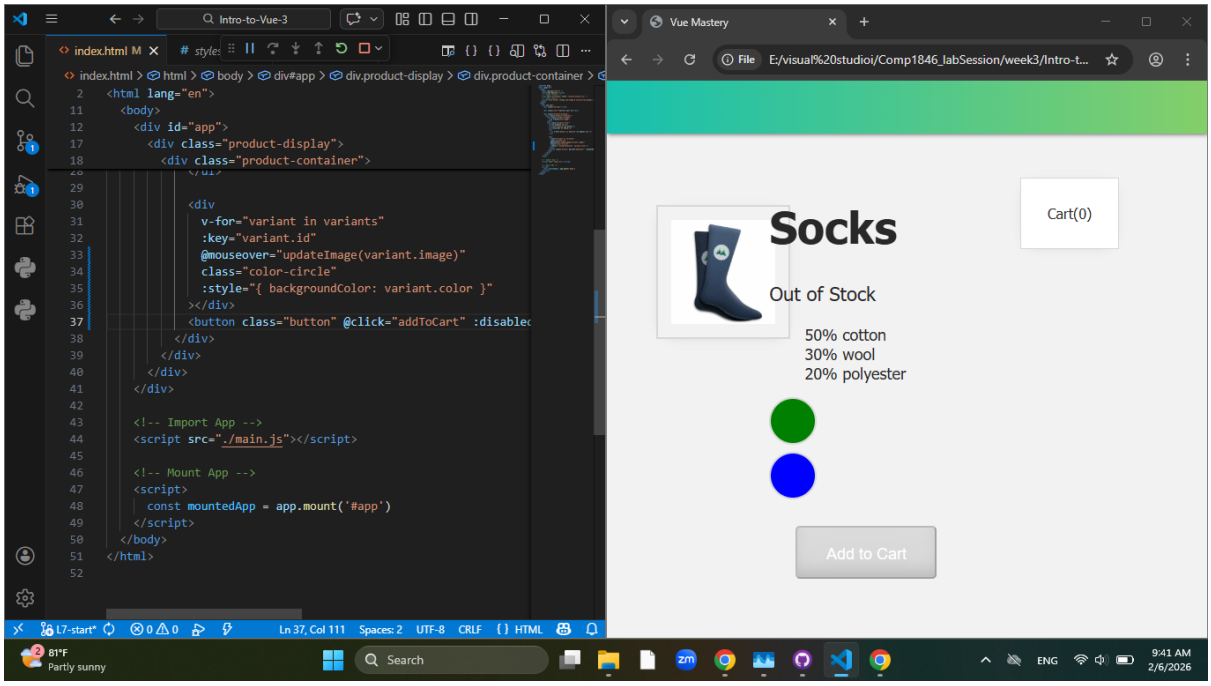
Task of lesson 6 picture:



Challenge of lesson 6 picture:



Task of lesson 7 picture:



Challenge of lesson 7 picture: