Comp1842- Web Programming 2
Vuong Quoc Anh-GCS220150
Class: Cos1103
Submission: Week 4

Table of Contents:

Appendix1:

1.1Task:

In this section, we learn how to create Vue instances and load data onto a website. Vue instances are the core of every Vue app, connecting elements to the DOM and making Vue reactive. We use v-bind: (or : for short) to bind data to HTML attributes. Vue has directives like v-if, v-else-if, v-else and v-show for conditional rendering. If the condition is true, the element shows v-show only toggles visibility without altering the DOM. The v-for directive lets us loop through arrays to display data, and it’s best to use a unique key for each element.

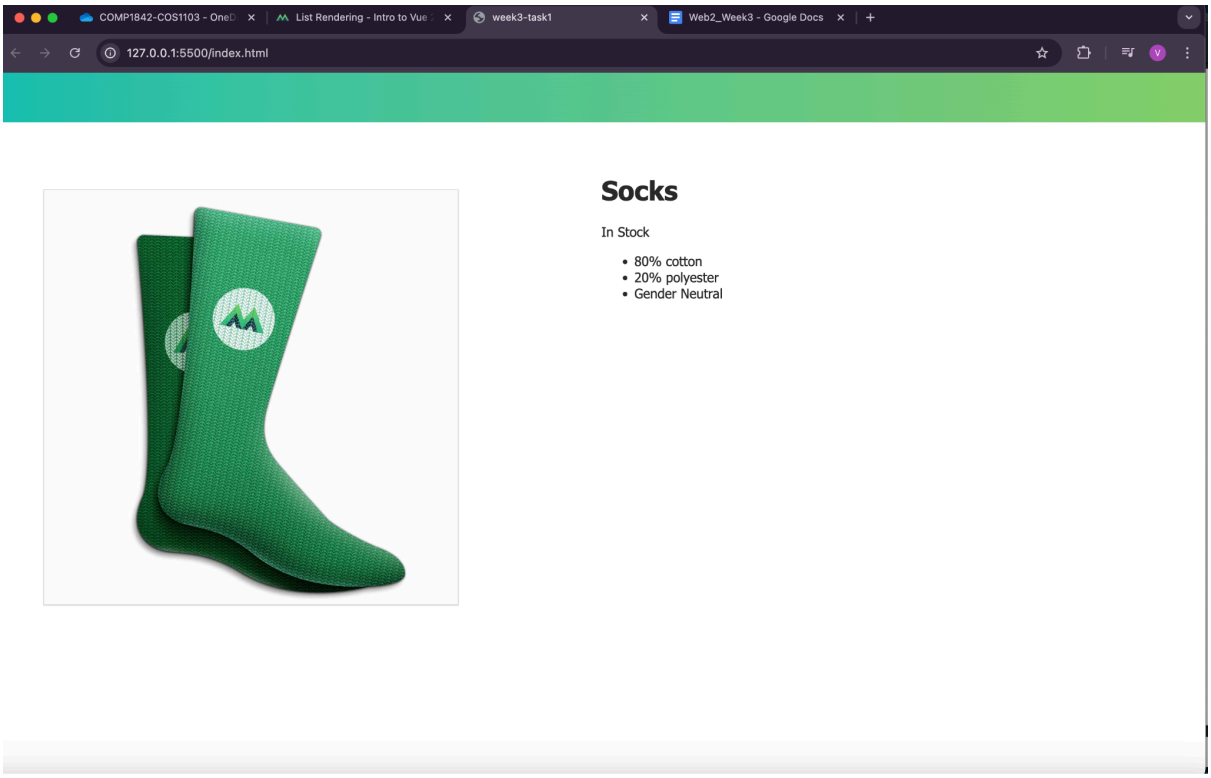

Fig 1:Browser

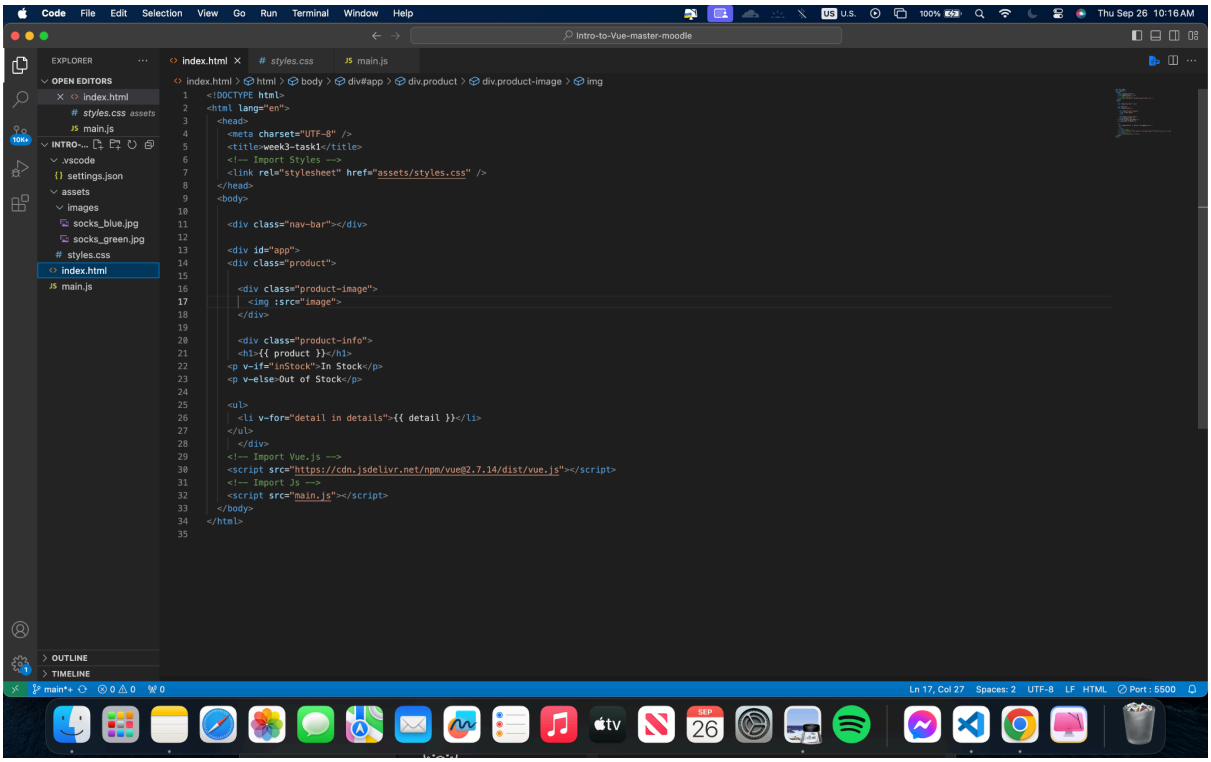

Fig 2:HTML

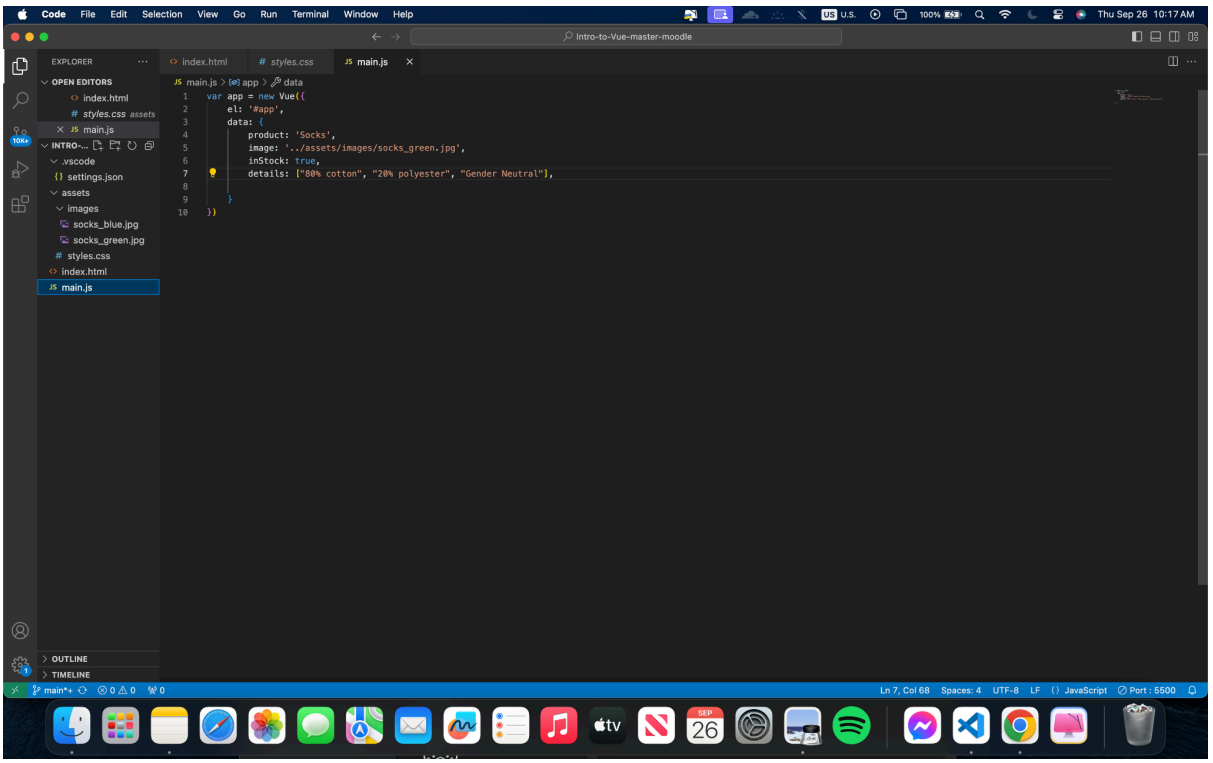

Fig 3: JS

1.2Challenges:

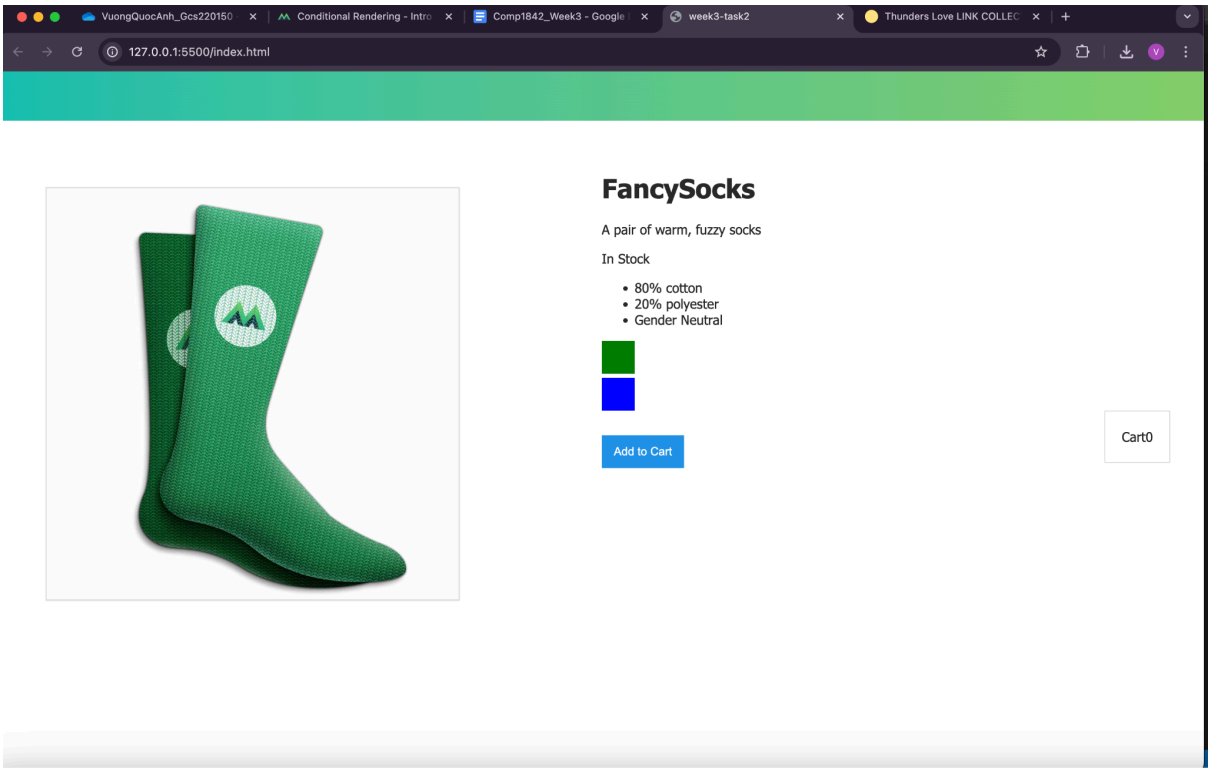

Fig4:Adding description

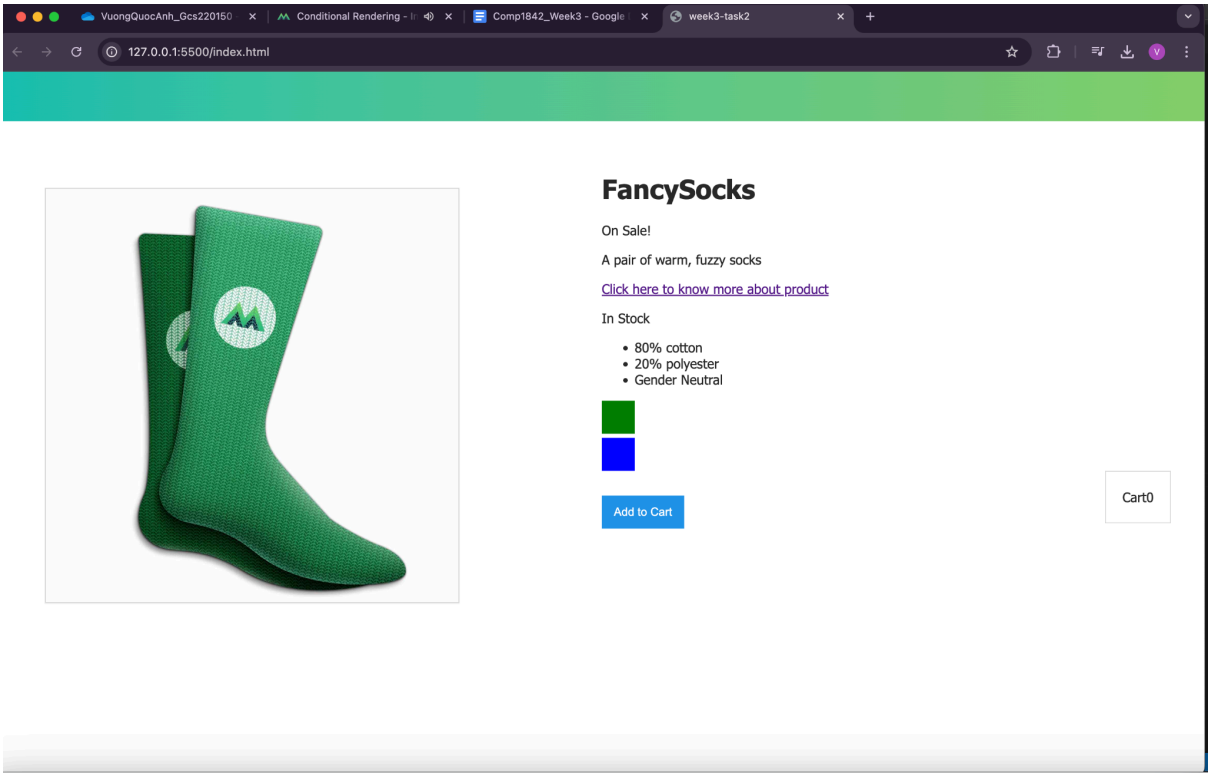

Fig5:Adding On Sale

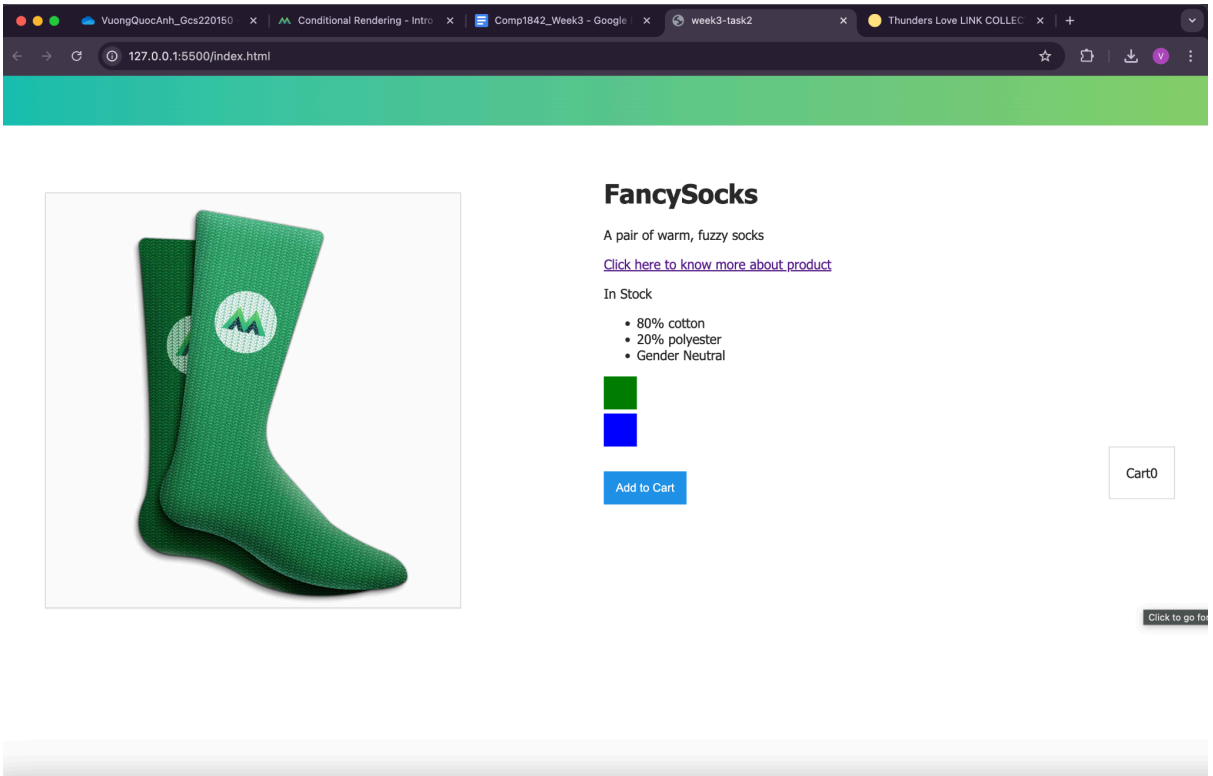

Fig6:Adding link

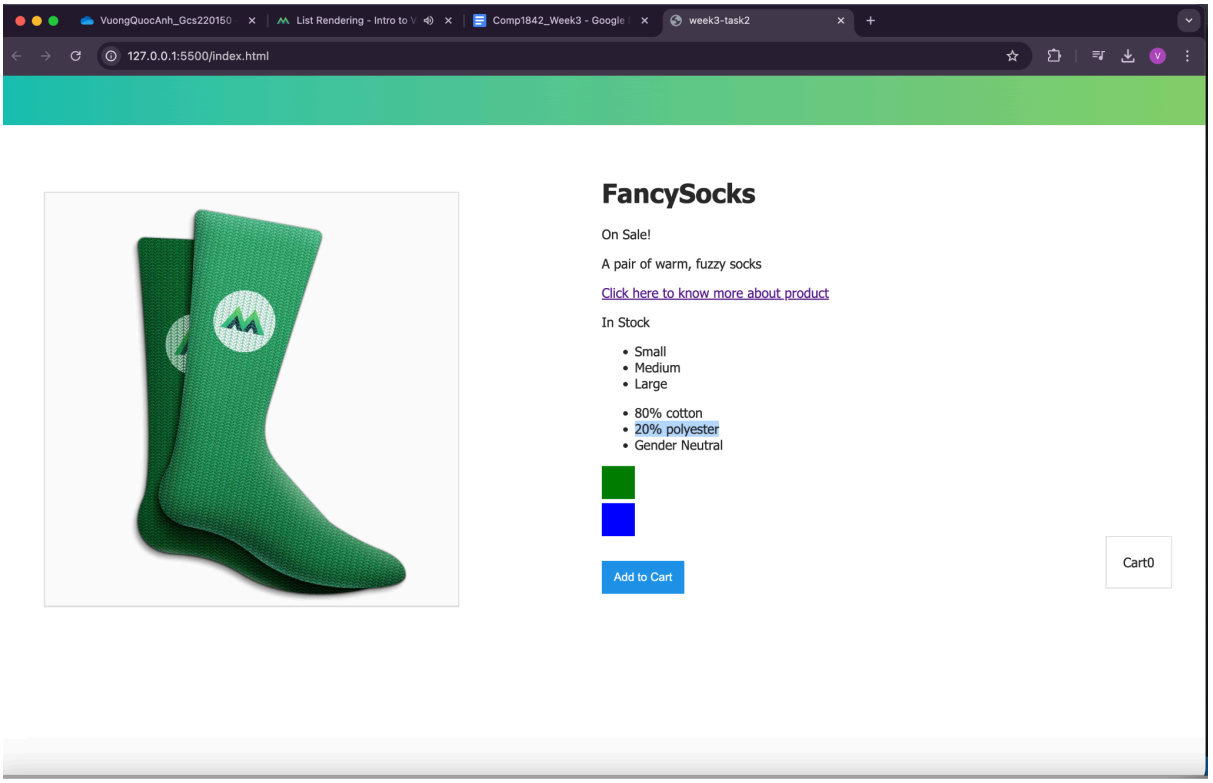

Fig7:Adding Size

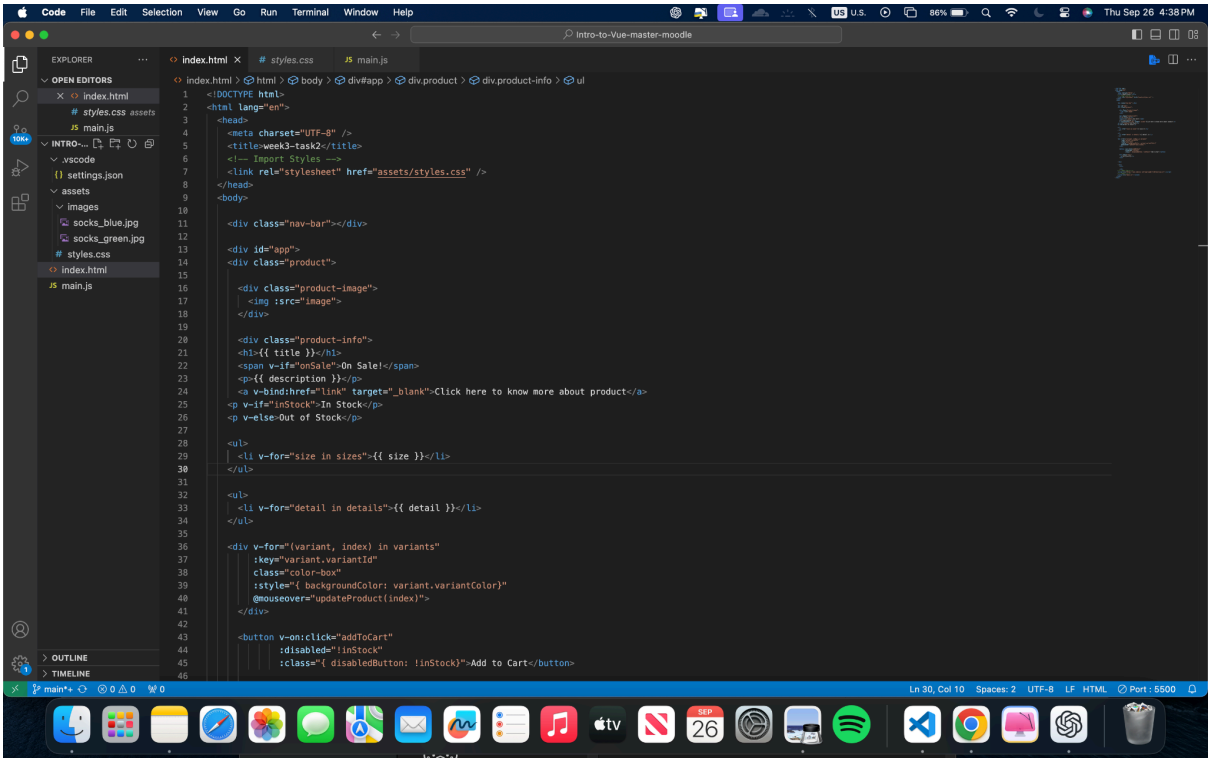

Fig8: HTML after doing challenges

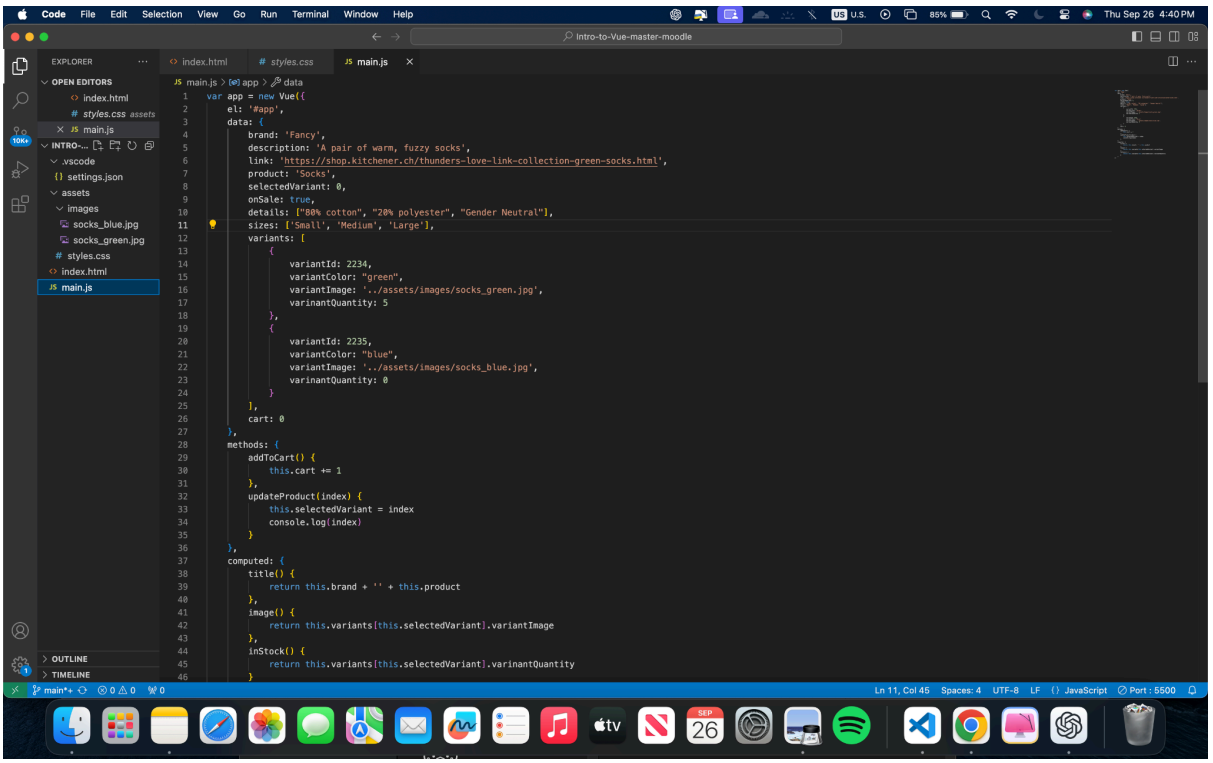

Fig9:JS after doing challenges

Appendix2:

2.1Task:

In this section, we use v-on to allow elements to listen for events. We know that the shorthand for v-on is @. V-on can directive trigger a method.

Data can be bound to an element's style and class.We can use expressions inside an element's class binding to evaluate whether a class should appear or not.

Computed properties calculate a value rather than store a value and use data from the application to calculate its values.

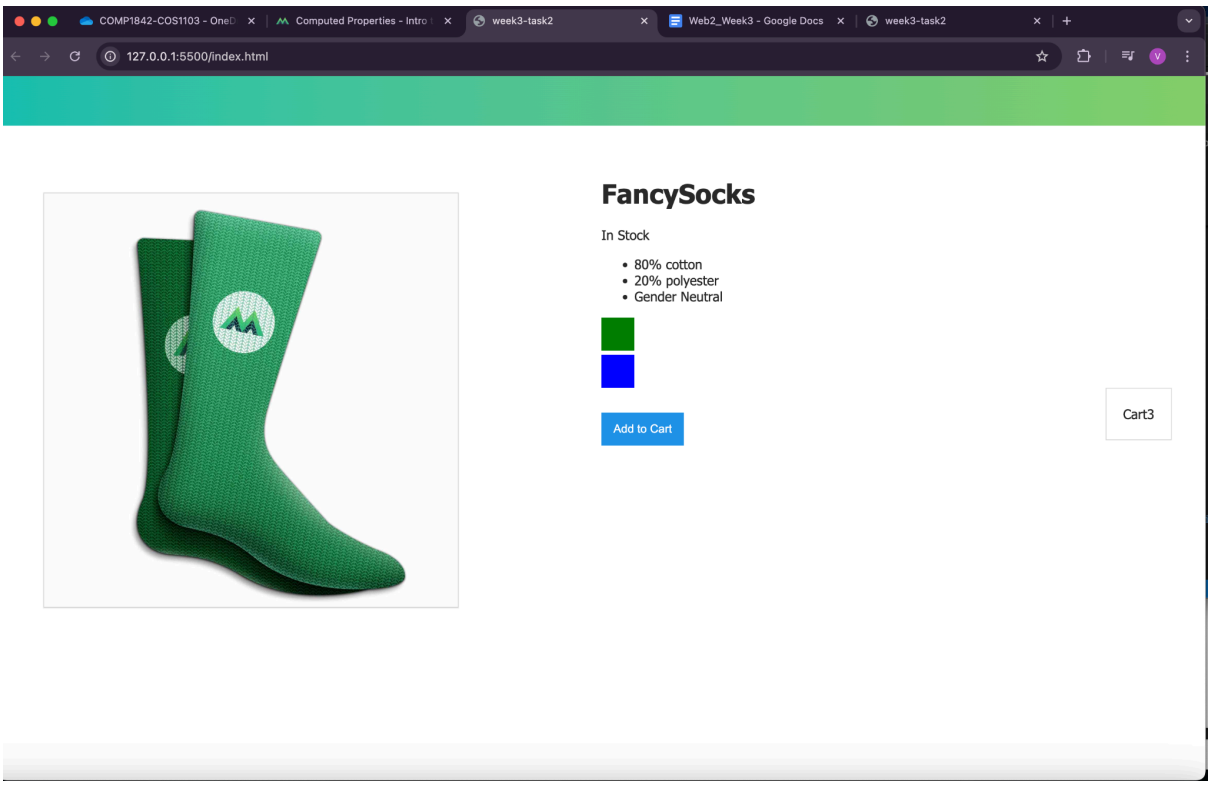

Fig10:Browser 1

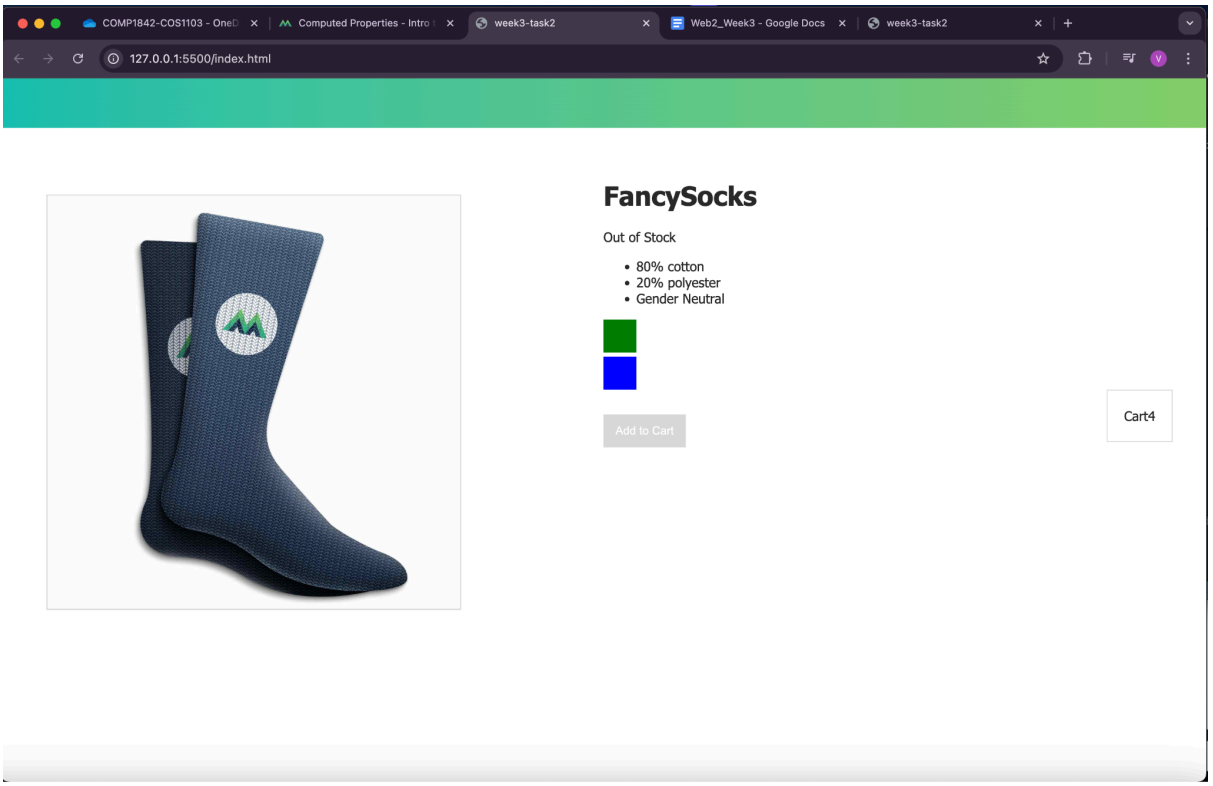

Fig11:Browser2

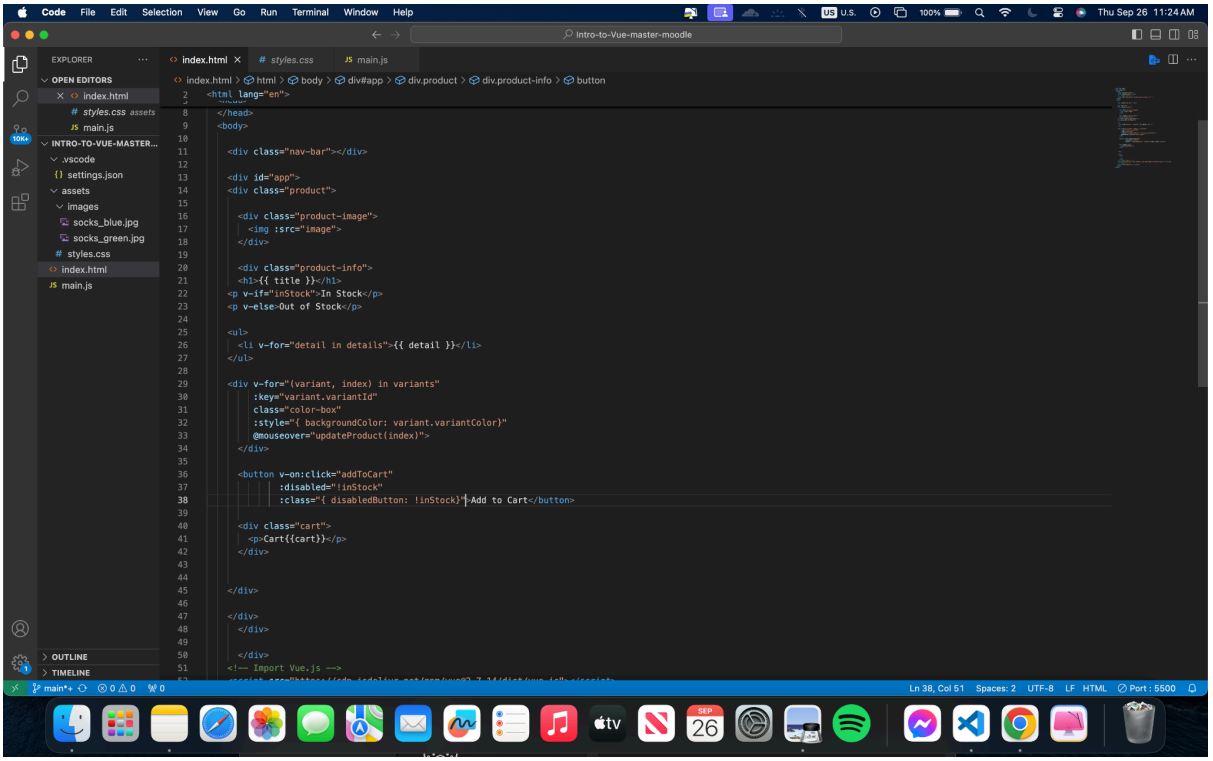

Fig12:HTML

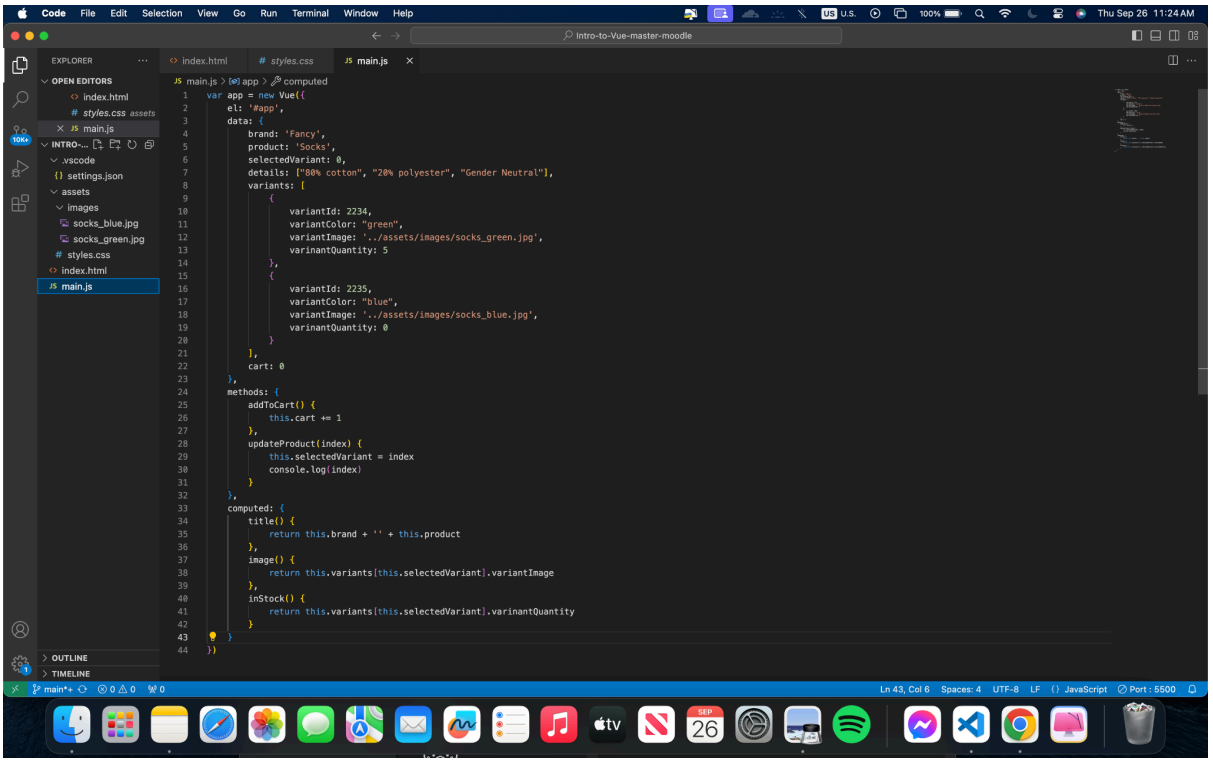

Fig13:JS

2.2Challenges:

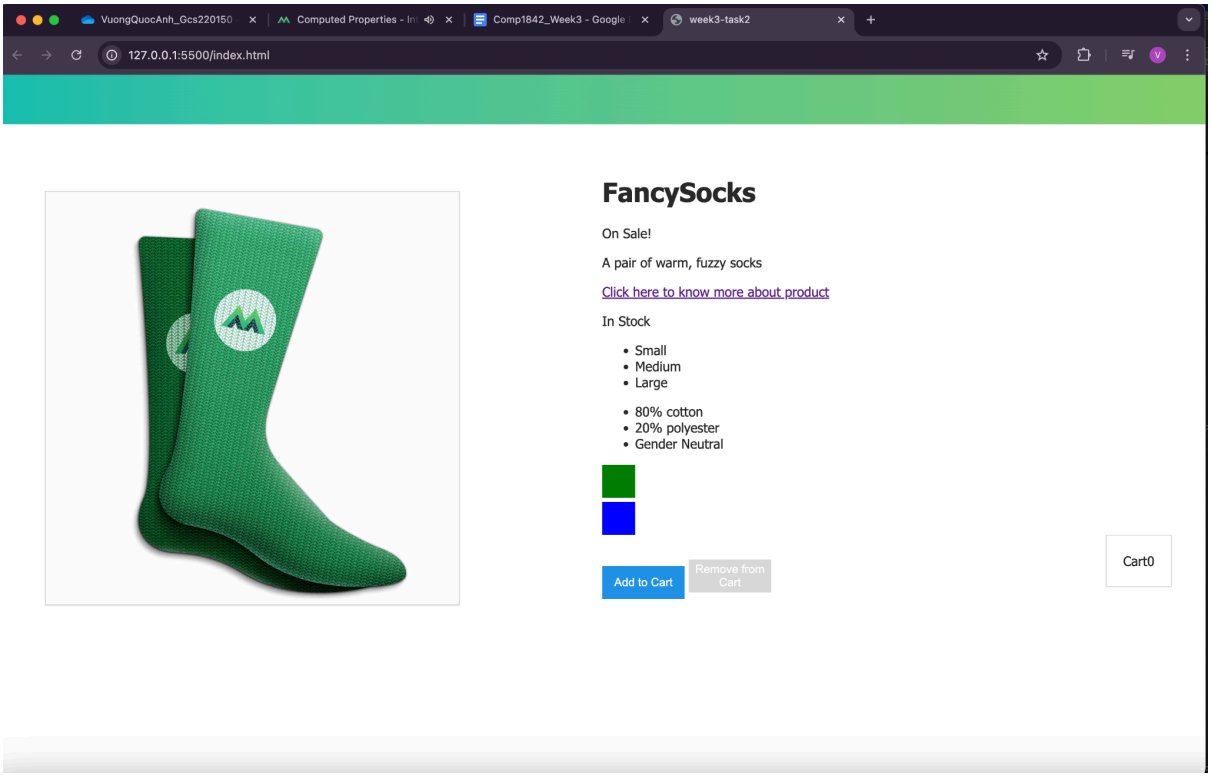

Fig14:Adding remove from cart button

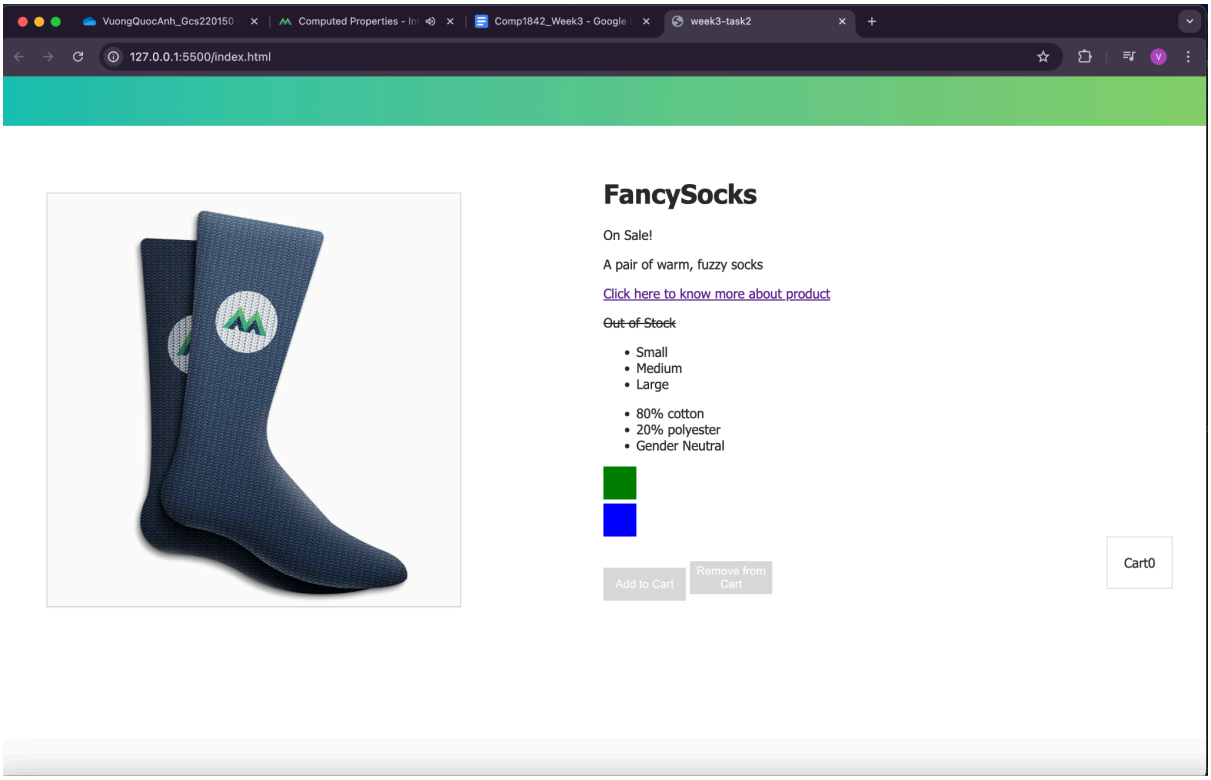

Fig15:Adding line-through

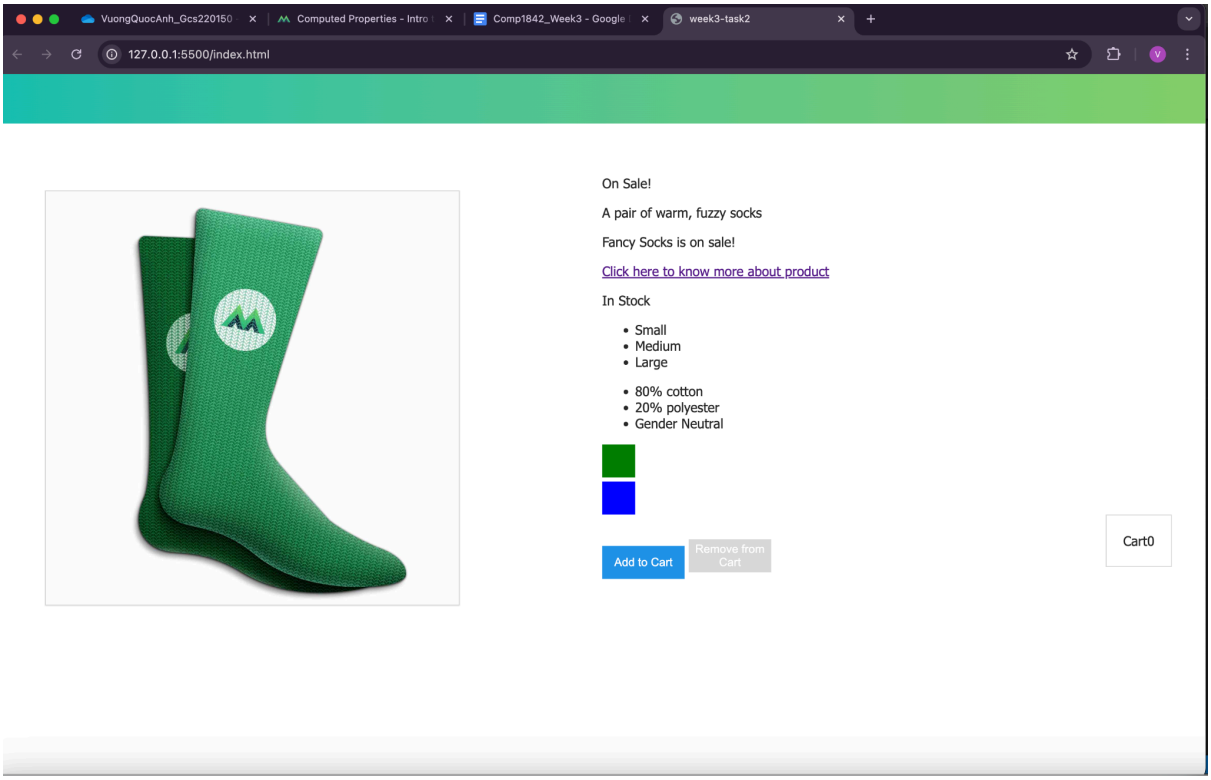

Fig16:Adding SaleMessage computed property

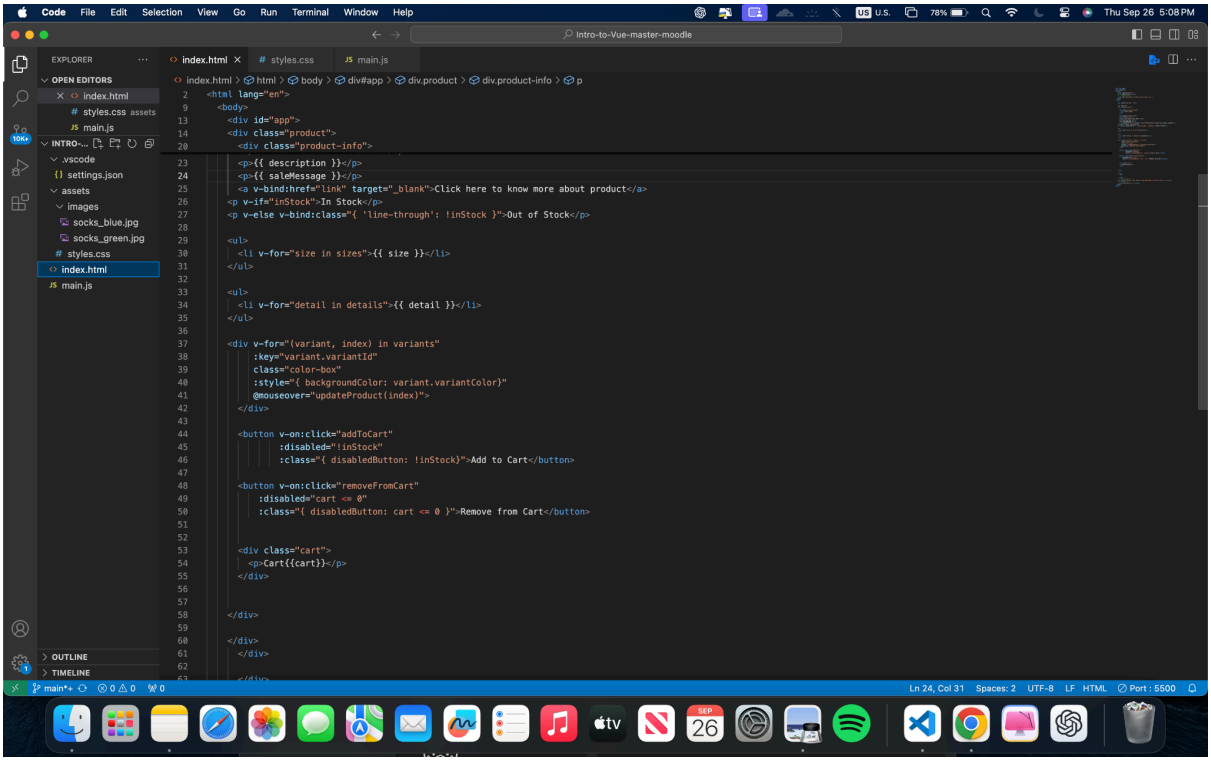

Fig17:Html after doing challenges

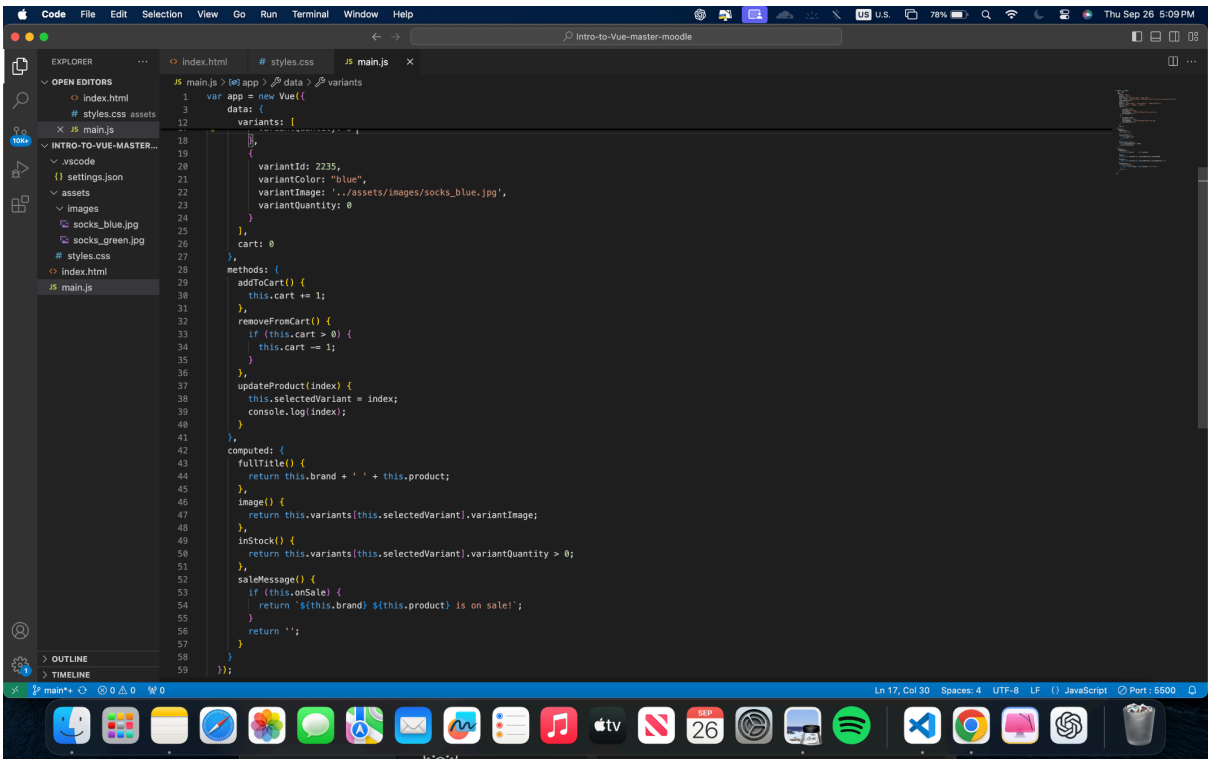

Fig18:Js after doing challenges

Appendix3:

3.1Task:

Vue components are like building blocks for an app. They help break up the app into smaller, reusable pieces, each with its own data and behaviour. This makes the app easier to manage and build.

In Vue, “props” are used to send data from a parent to a child component. Think of props as a way for the parent to give information to the child. You add props to a component through a special attribute, and you can set rules for the kind of data it can receive.

Components can also talk back to the parent. A child component can use `$emit` to send a message or data when something happens. The parent listens for this with `v-on` (or `@` for short) and runs a method when it hears the event.

For forms, Vue’s `v-model` creates a two-way data link between the input and the data. The `.number` modifier can turn the input into a number, but watch out—there’s a known bug. Use `.prevent` to stop the form from reloading the page when you submit. With Vue, you can add simple checks to make sure the form is filled out correctly.

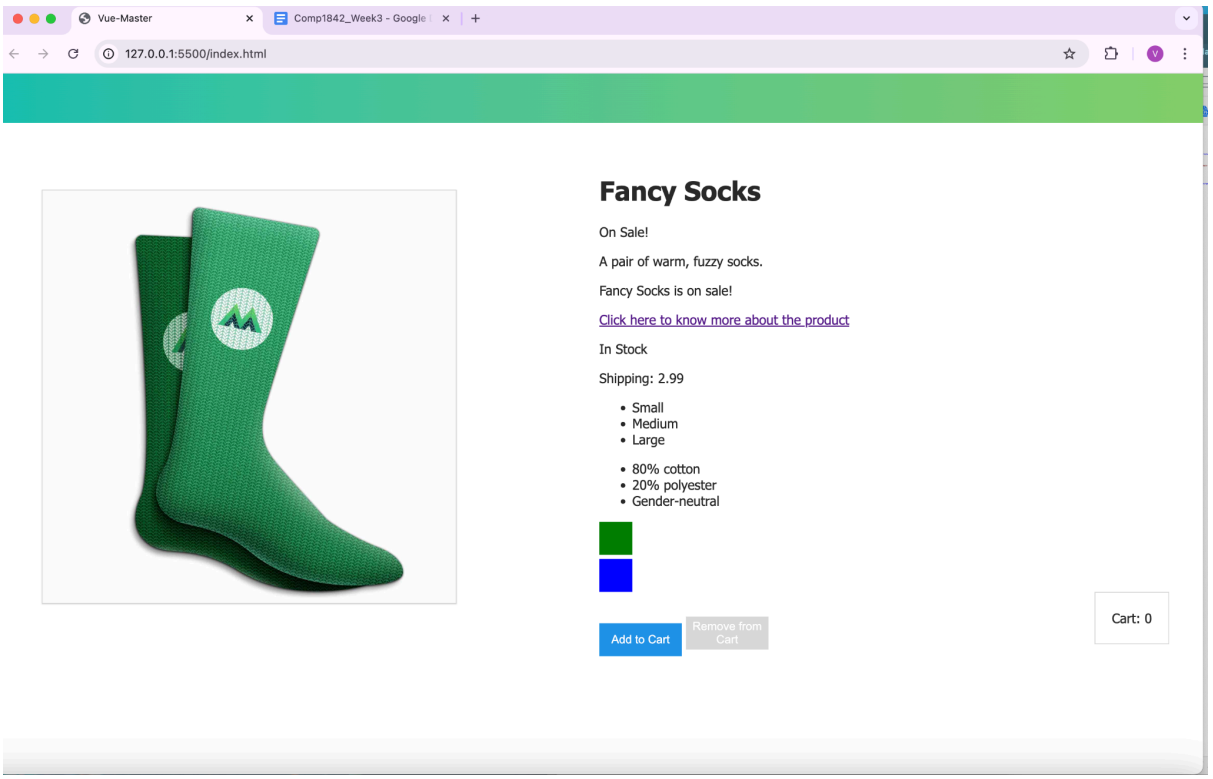

Fig19: Add components

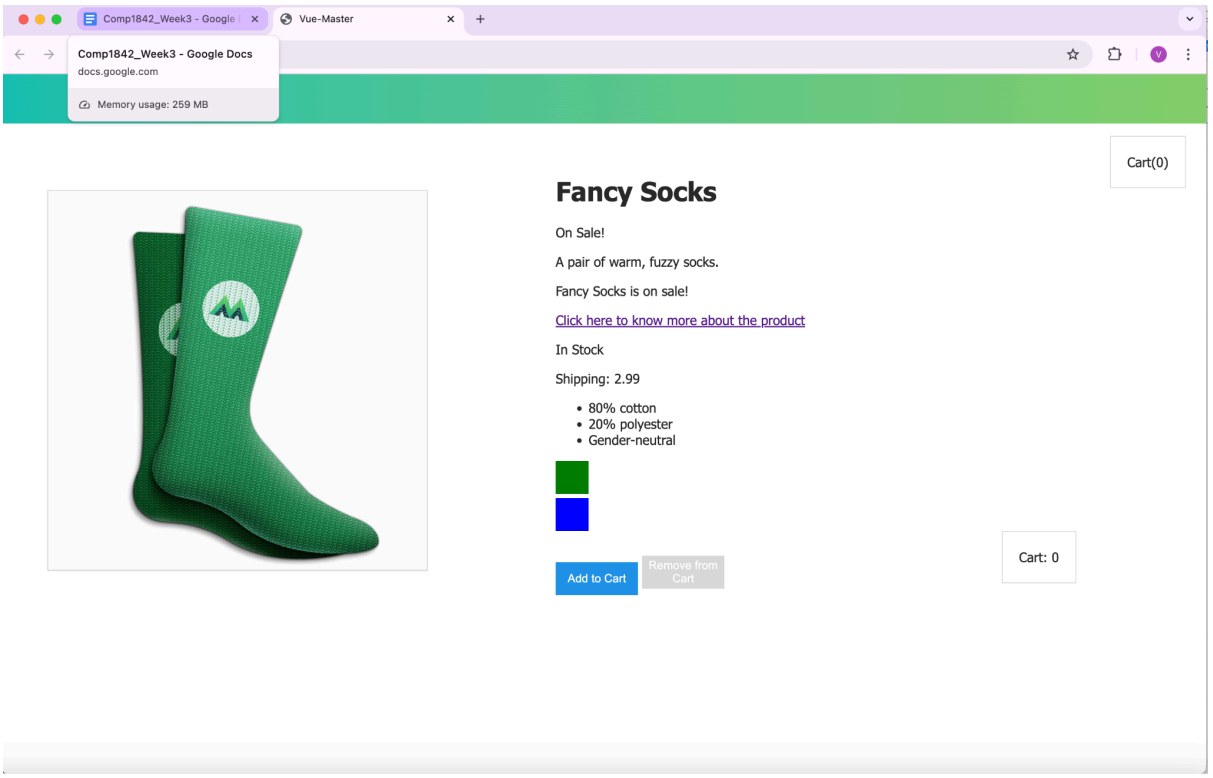

Fig20: Using emit

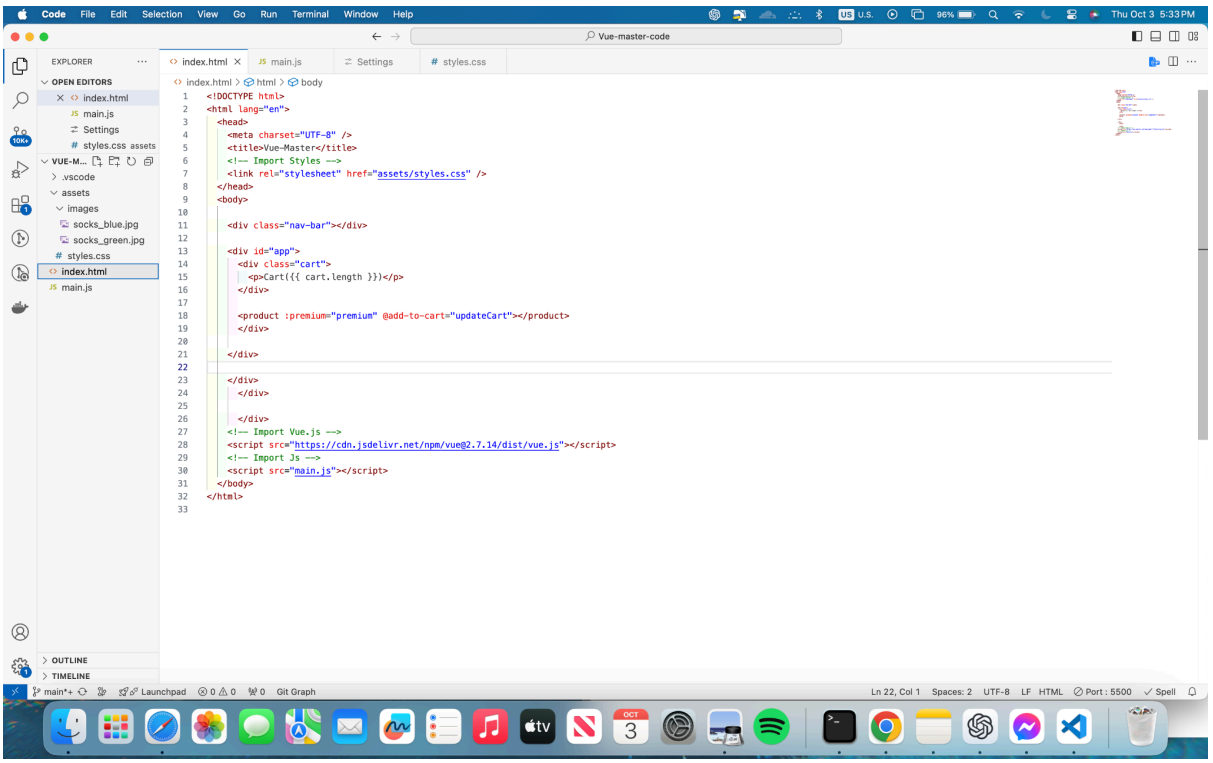

Fig21: html

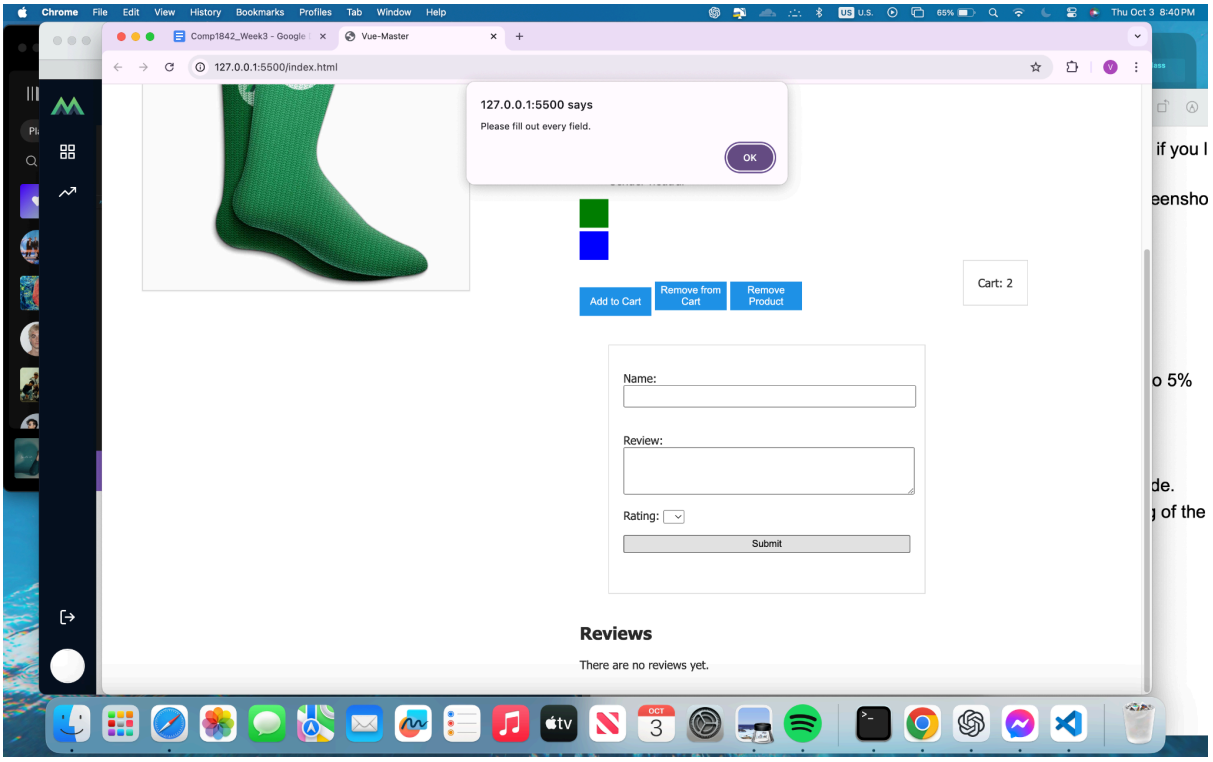

Fig22: Without filling any blank

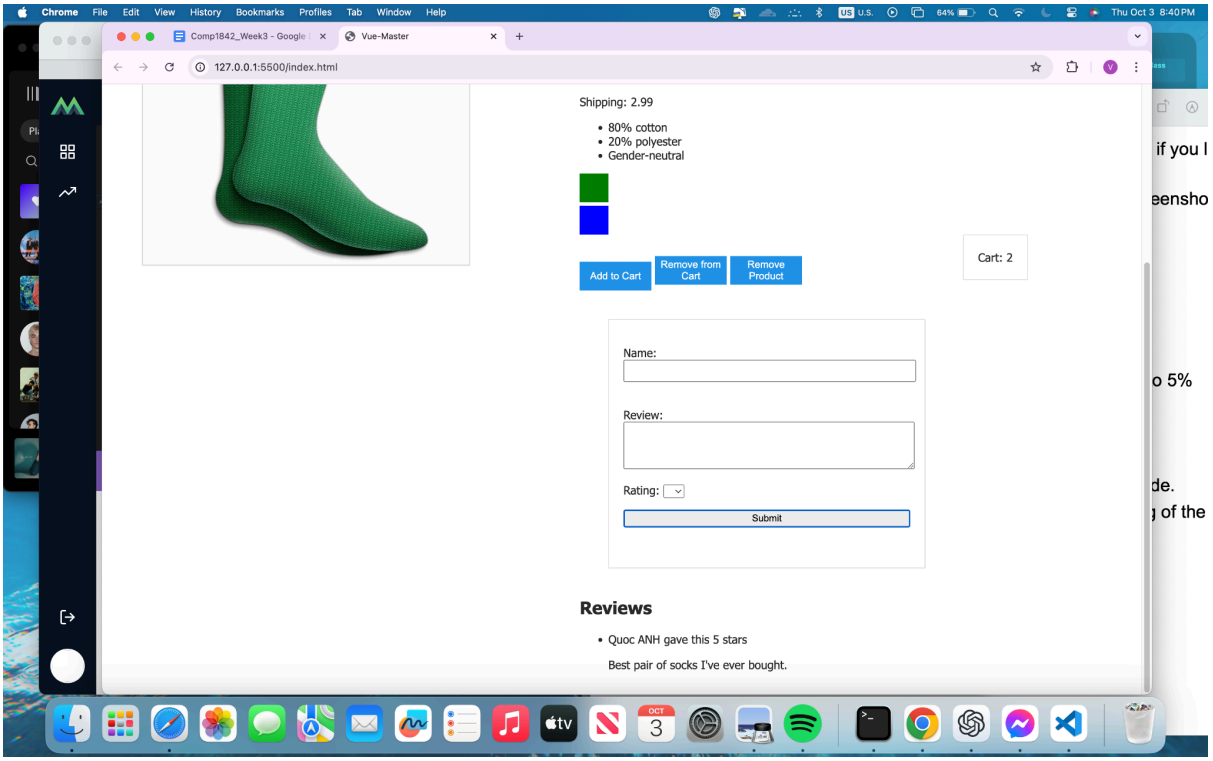

Fig23: Add name and review

3.2Challenges:

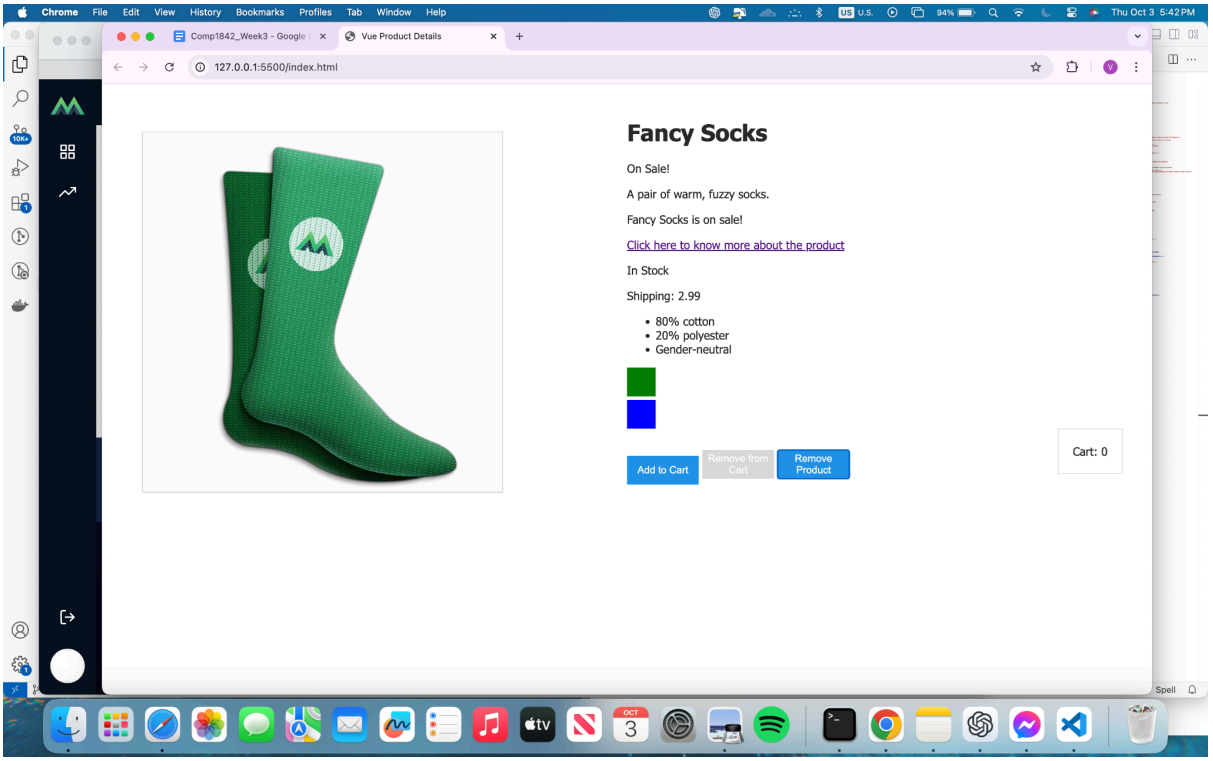

Fig24:Using emit to add remove product

Using following emit code to add remove product

```
  },
  emitRemoveProduct(productId) {
    // Emit the event to remove the product with its ID
    this.$emit('remove-product', productId);
  }
},
```

```
methods: {
  removeProduct(productId) {
    // Remove the product from the cart by its ID
    this.cart = this.cart.filter(item => item !== productId);
  }
}
);
```

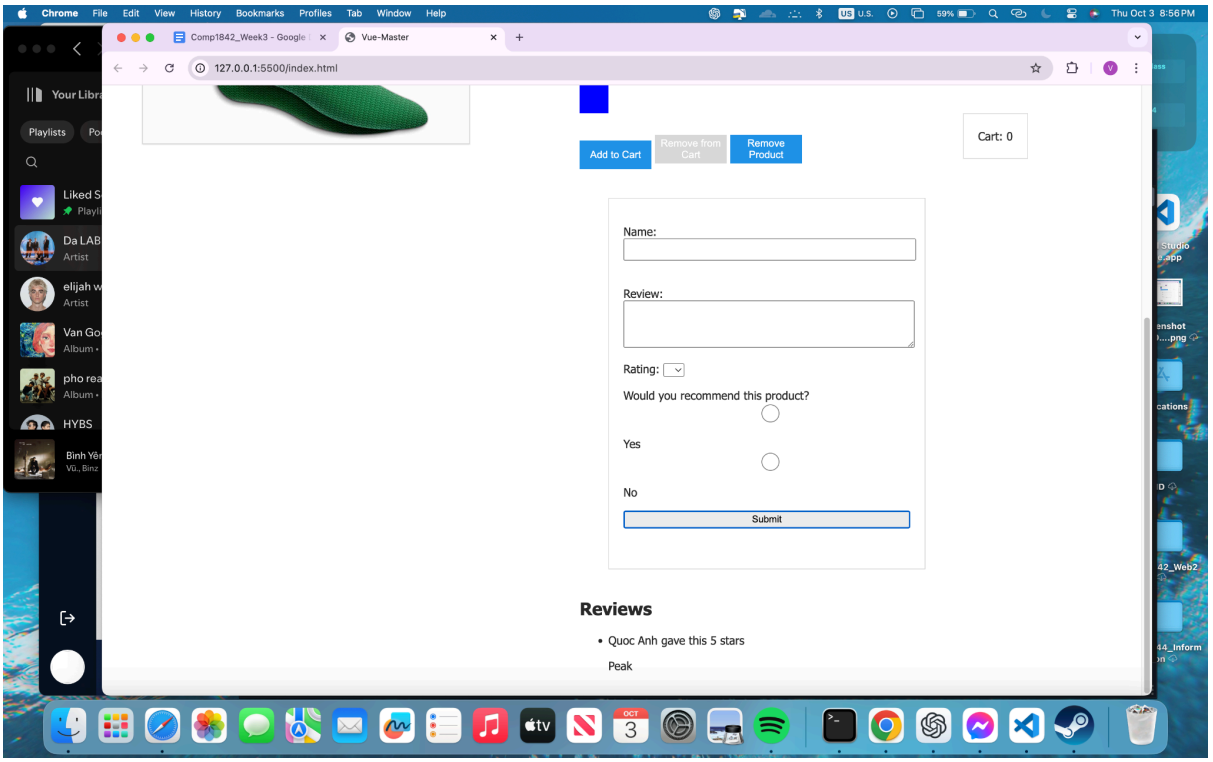

Fig25: Add recommendation with yes/no button

Fig26: main.js after doing challenges

Appendix4:

In Lesson 4, I am learning how to manage collections and query data effectively. **Exercise 1** teaches me how to create a collection from a JSON file, a common format for data storage and transfer. In **Exercise 2**, I explore different views of the collection, which is vital for organising and analysing data efficiently. Finally, **Exercise 3** introduces me to using a Visual Query Builder to extract information from the dataset without writing code manually. These exercises help develop key skills in data management, ensuring I can handle collections and perform queries in various practical applications.

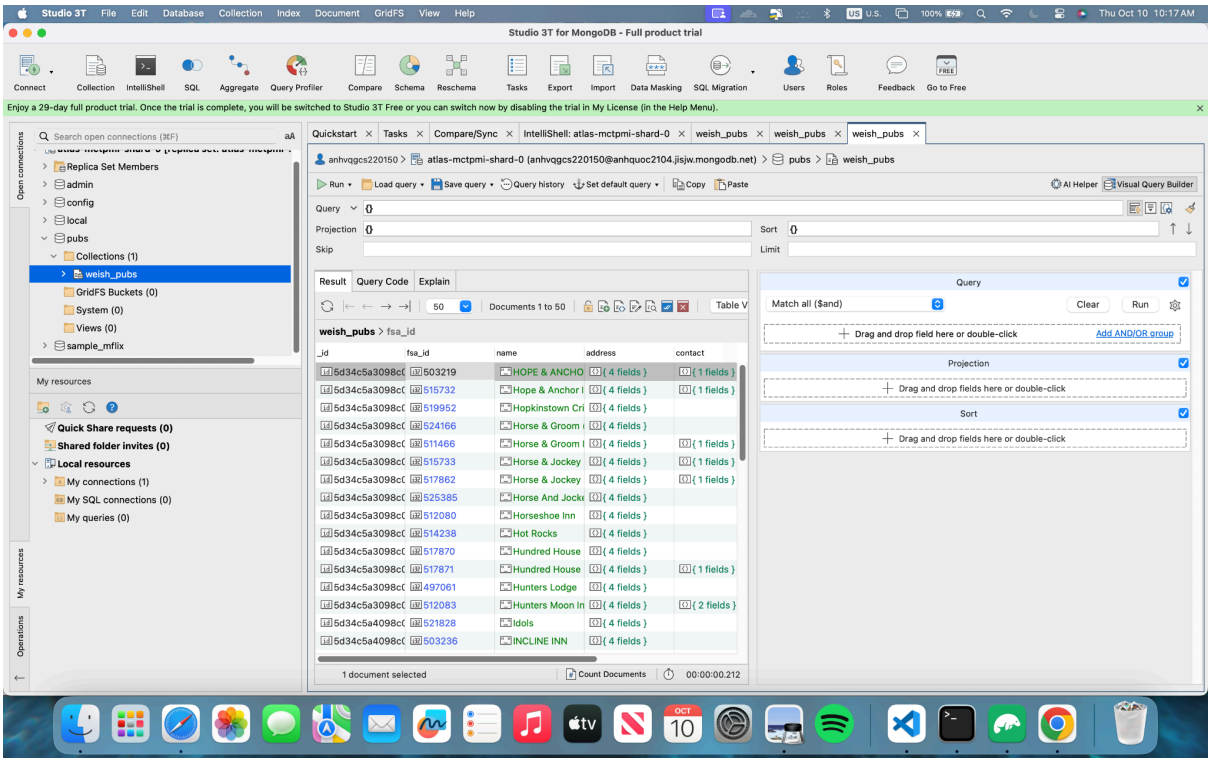

Fig27:Lesson4 exercise 1

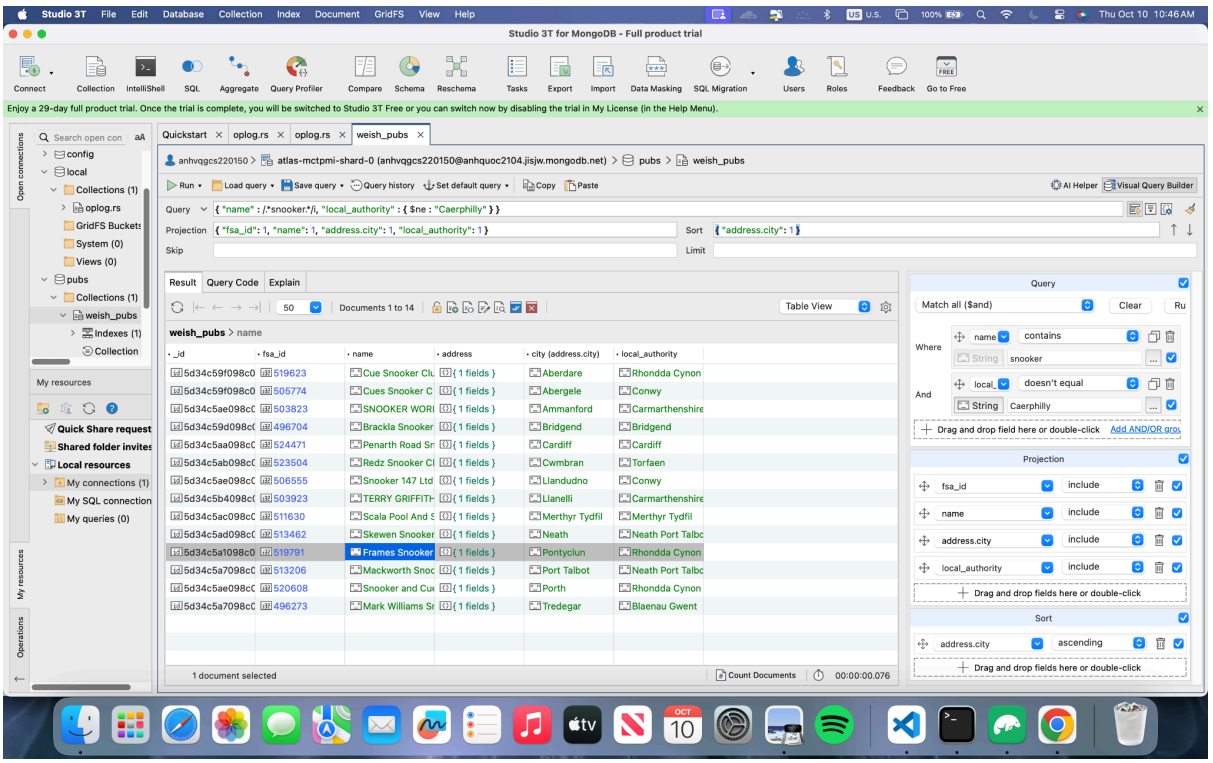

Fig28:Lesson4 exercise 3

In Lesson 5, I am developing proficiency in executing and refining SQL aggregate queries within MongoDB, a key skill for working with large datasets. Exercise 1 introduces the process of running a SQL aggregate query in MongoDB, demonstrating how SQL-based logic can be adapted to a NoSQL environment. Exercise 2 emphasises exporting SQL queries into the Aggregation Editor, providing a seamless method to transition between SQL and MongoDB's native aggregation framework. Finally, Exercise 3 covers editing queries in the Aggregation Editor, enhancing my ability to modify and optimise queries for more efficient data processing and analysis within MongoDB.

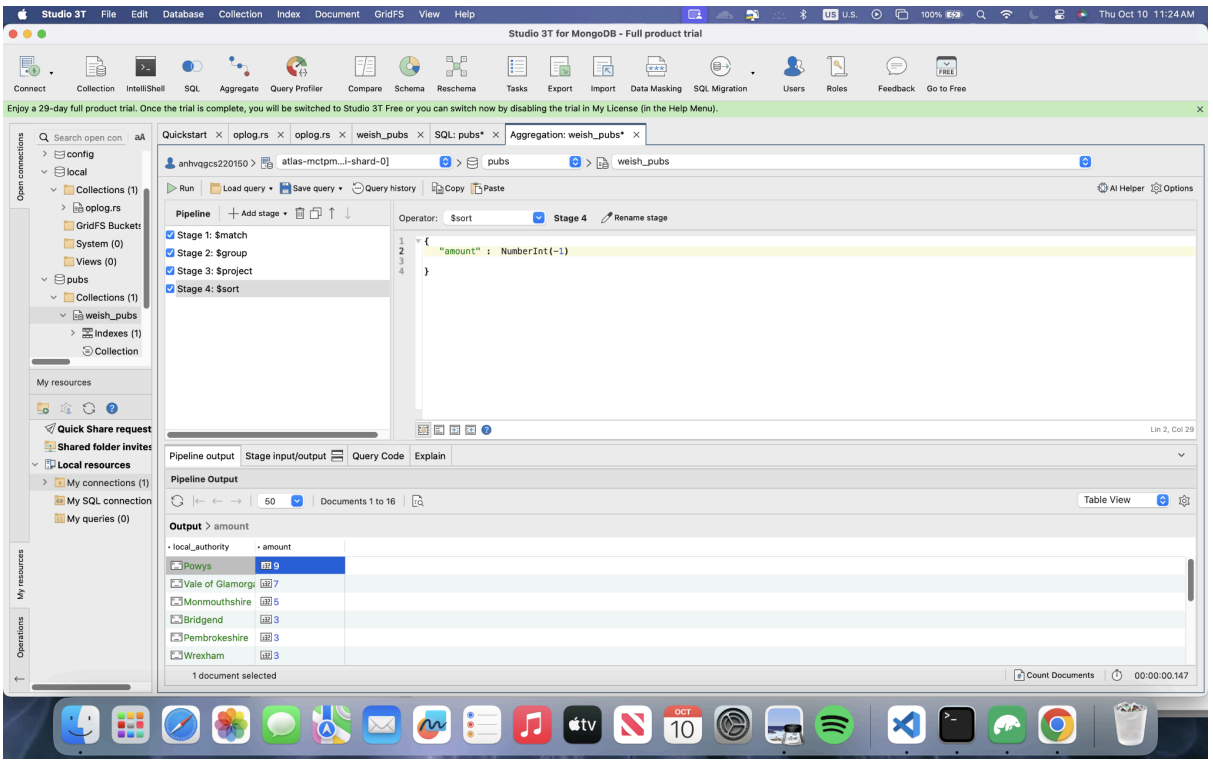

Fig29:Lesson5 exercise 3

In Lesson 6, I am learning how to manage document data through import and export processes. Exercise 1 focuses on importing data from a CSV file, a crucial skill for handling structured datasets. Exercise 2 teaches how to export document data to a JSON file, enabling data transfer and storage.

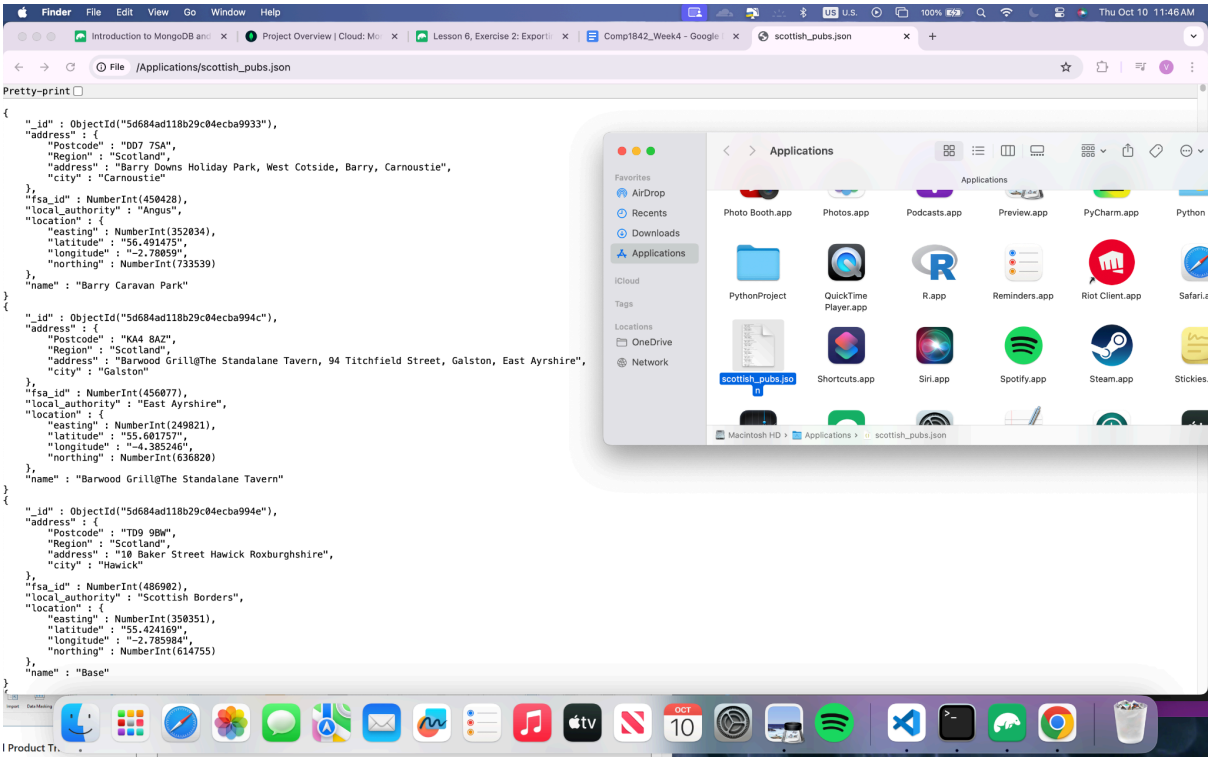

Fig30: Lesson6 exercise 2

Git Code: https://github.com/AnhQuoc1234/lab