Module	COMP1842
Class	COS 1103
Full Name	Nguyen Huu Phuoc Dat

1. Task 1

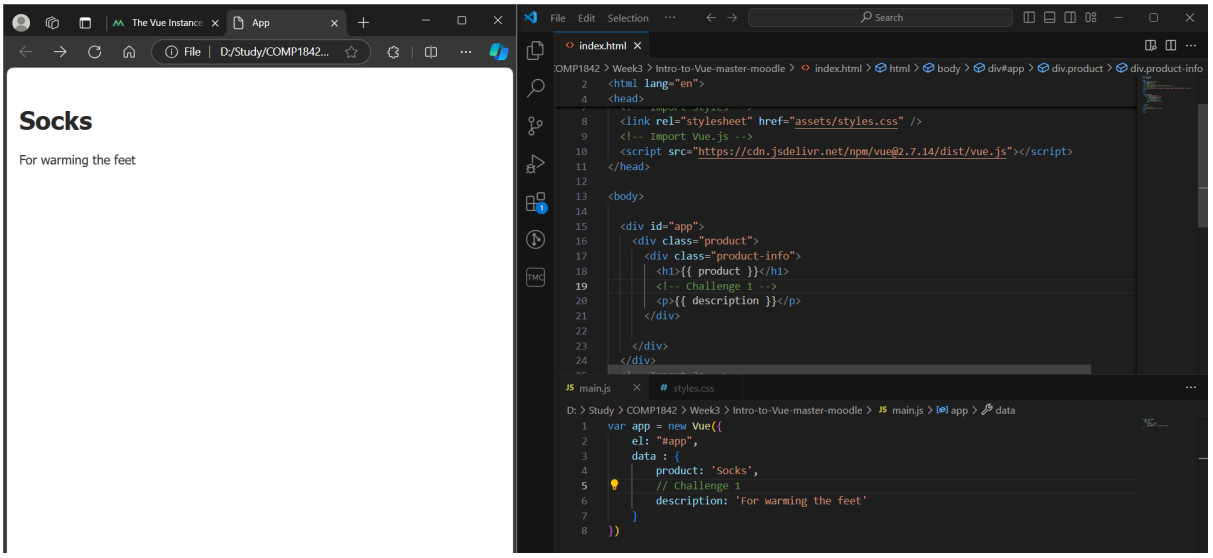

Figure 1: Lesson 1 with challenge

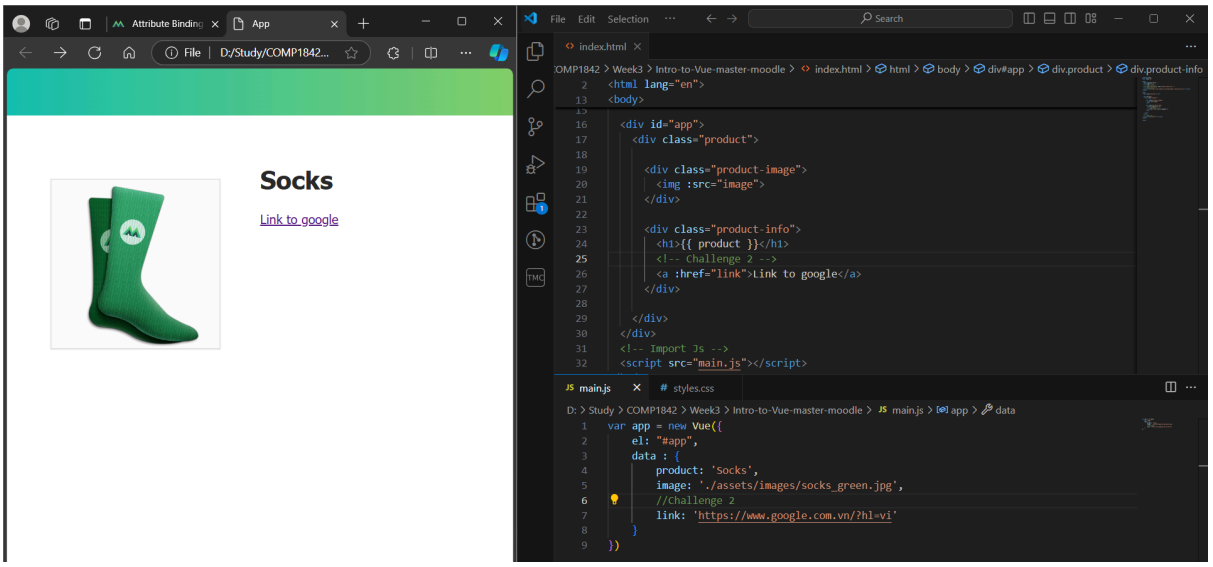

Figure 2: Lesson 2 with challenge

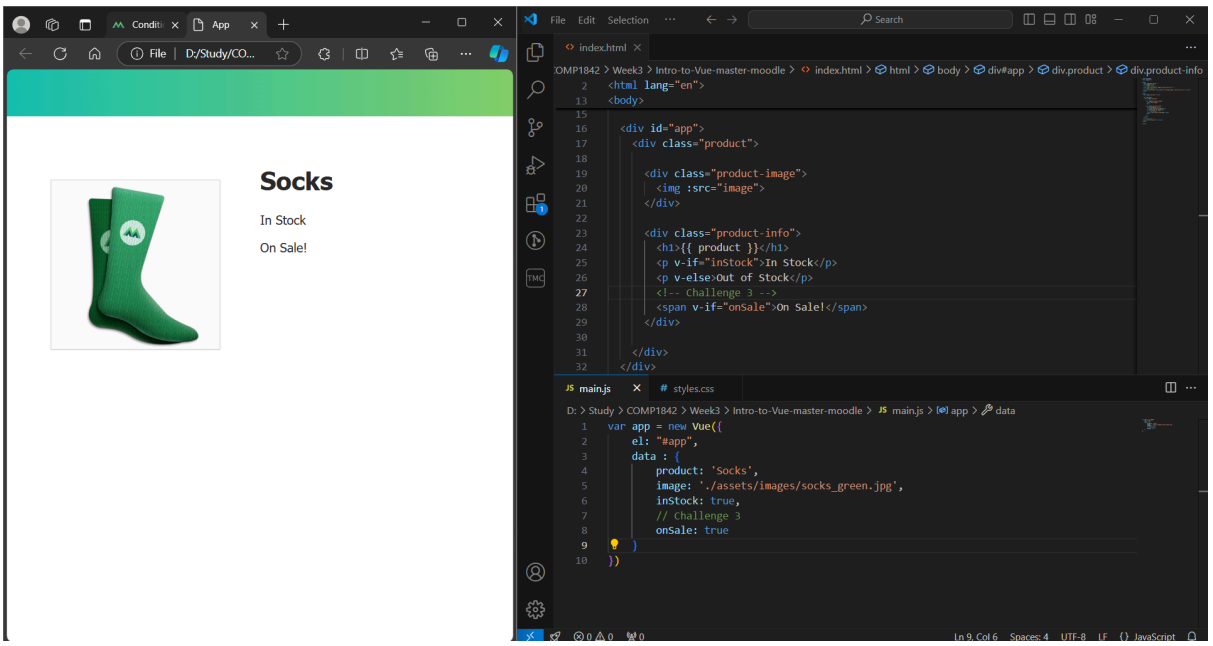

Figure 3: Lesson 3 with challenge

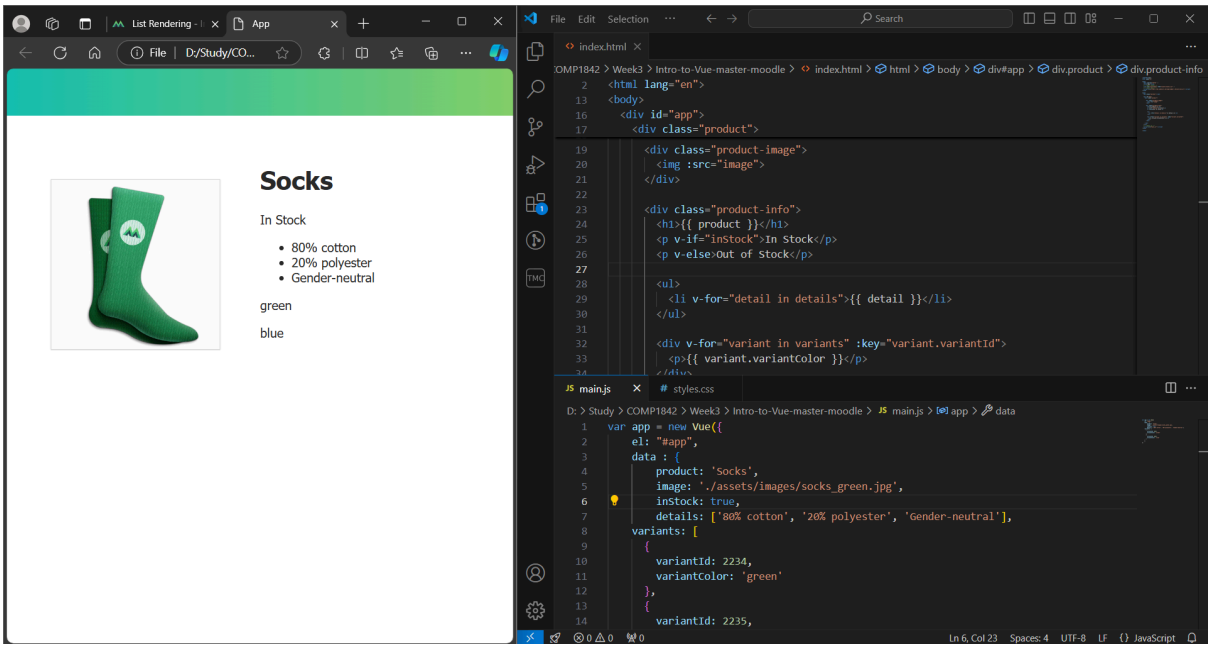

Figure 4.1: Lesson 4

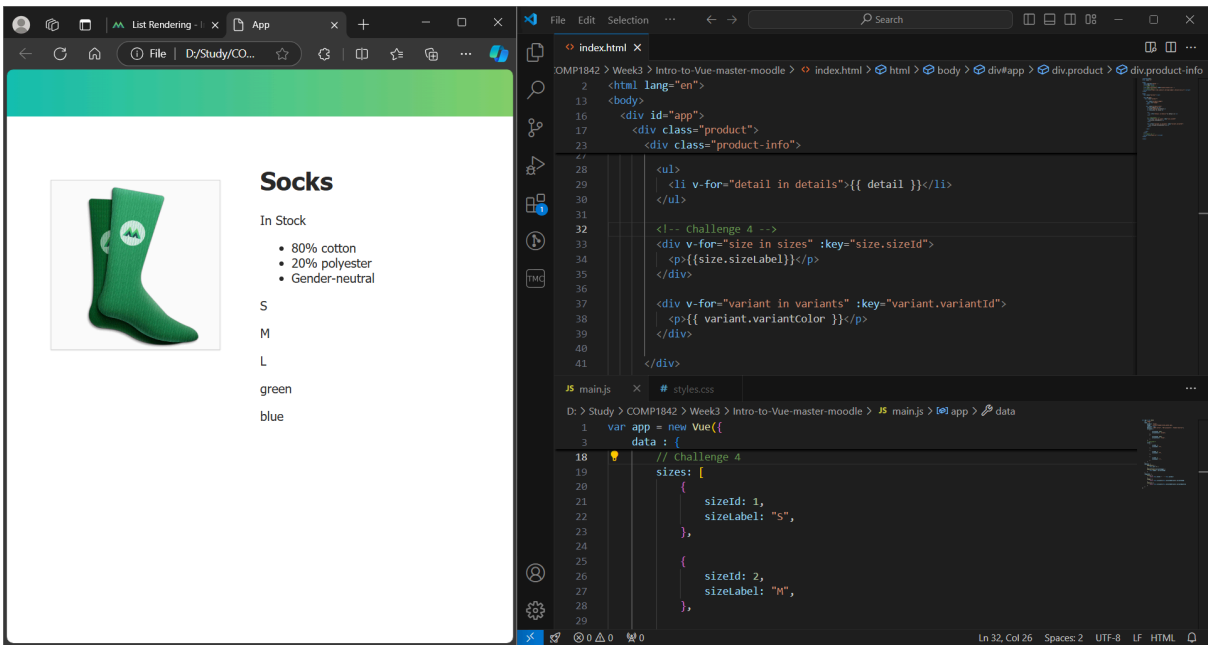

Figure 4.2: Lesson 4 with challenge

In Lesson 1, I learned that a Vue instance serves as fundamental to an application. By passing an options object, I can easily define data and associate it with the DOM. The essential takeaway is how Vue automatically updates the DOM anytime the data changes, as shown in {{ product }}, which dynamically replaces a template expression with actual data.

Lesson 2 highlighted the significance of attribute binding by bounding data to HTML attributes. I can use directives like :src to attach DOM elements to data attributes. It implies that any changes to the data property instantly show up in the view.

Lesson 3 looked into conditional rendering, which uses v-if, v-else-if, and v-else to simplify the display of objects based on their state. This feature enables the dynamic addition and removal of items based on expression assessment. Furthermore, the element's visibility can be toggled using v-show, which makes no changes to the element in the DOM.

Lesson 4 explored iterating over data arrays with v-for and showing the results. It reduces code repetition, making it easier to handle dynamic data. An array of objects can also be looped over and the values displayed using dot notation.

2. Task 2

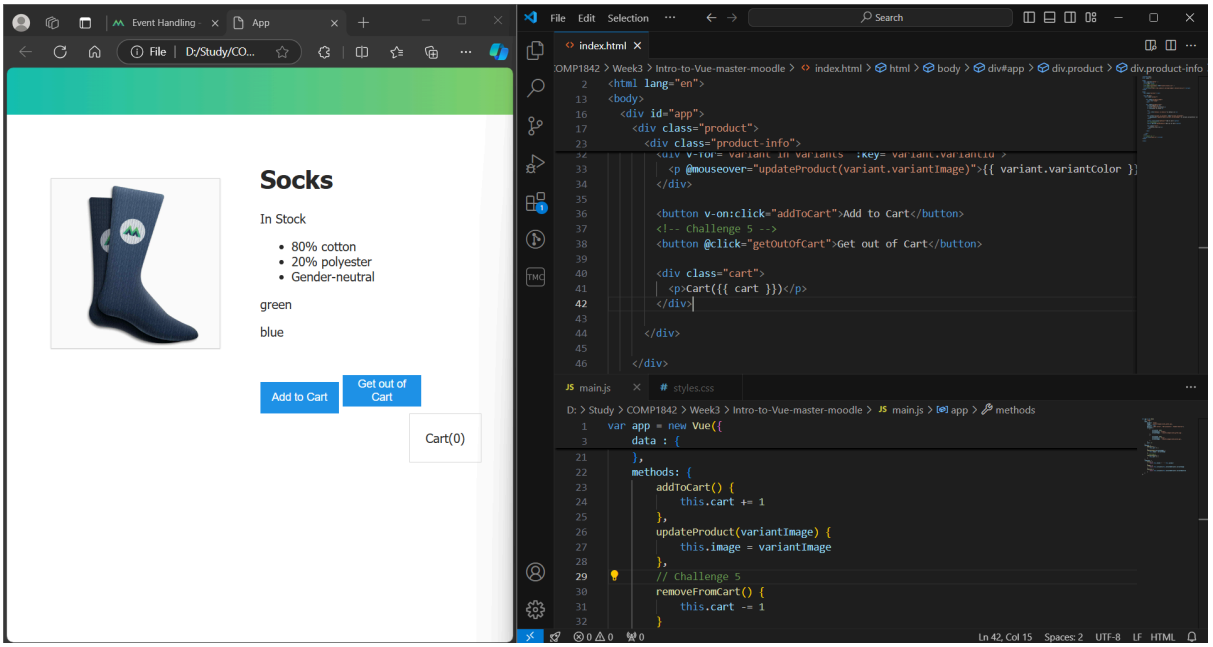

Figure 5: Lesson 5 with challenge

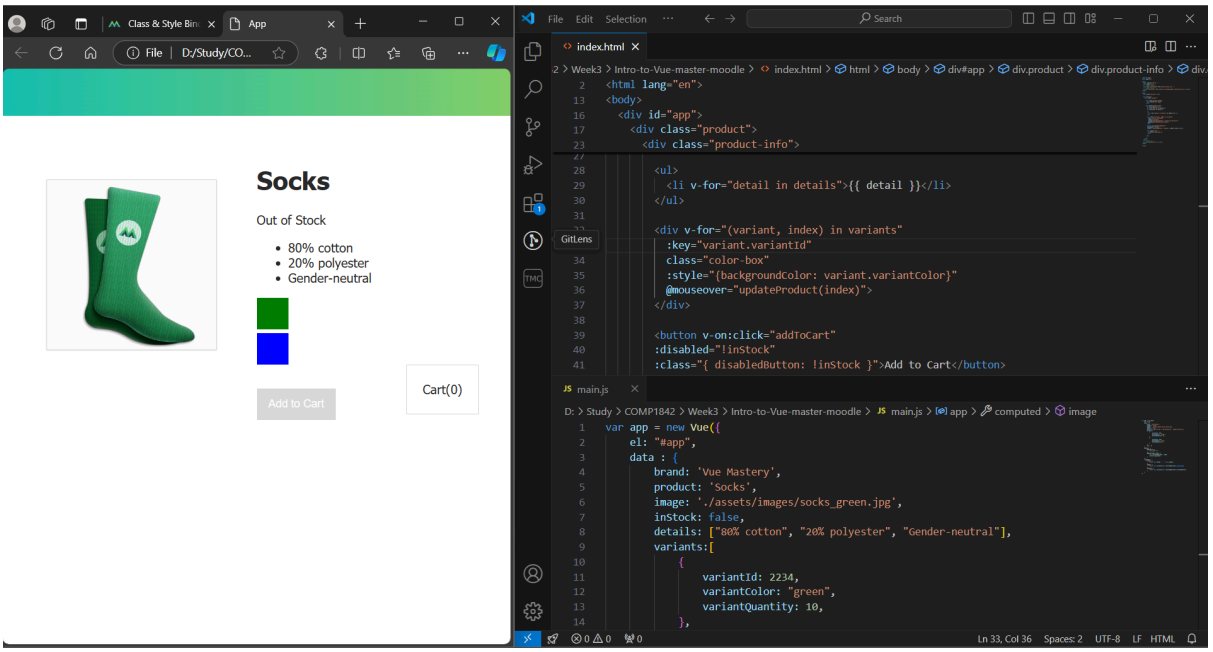

Figure 6.1: Lesson 6

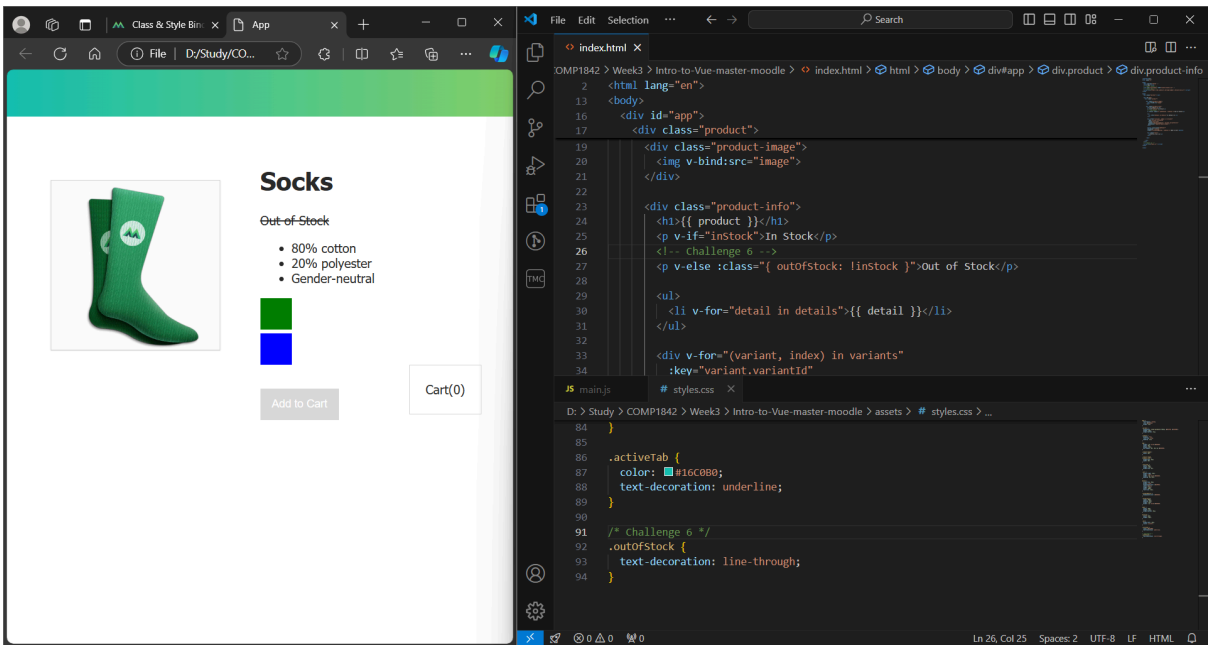

Figure 6.2: Lesson 6 with challenge

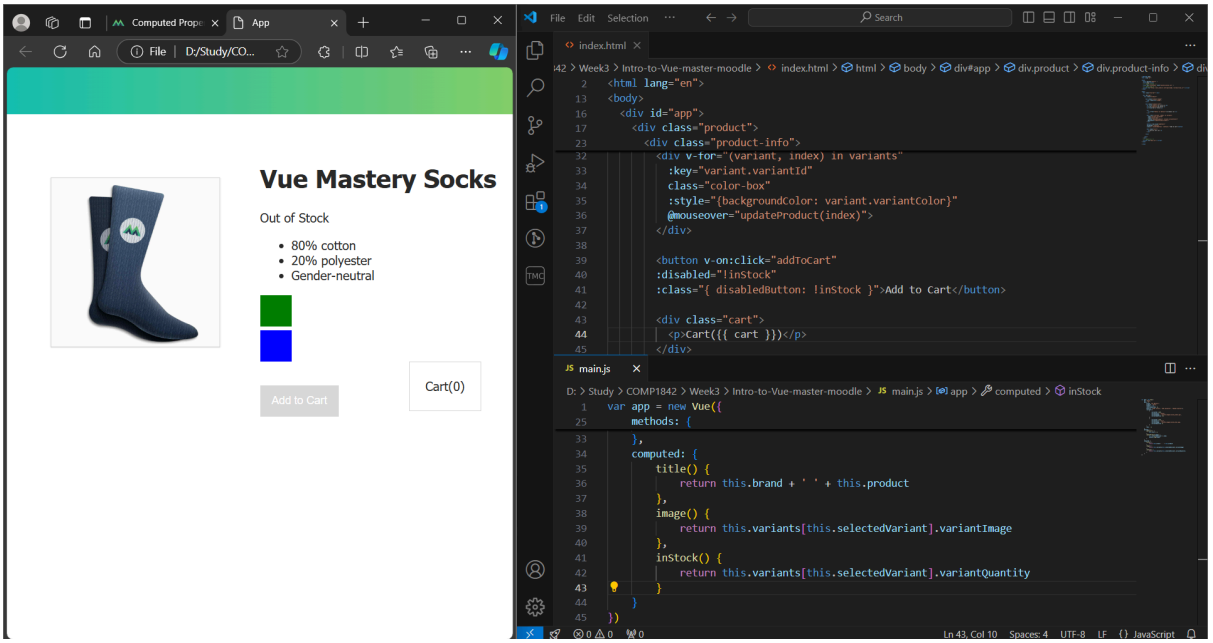

Figure 7.1: Lesson 7

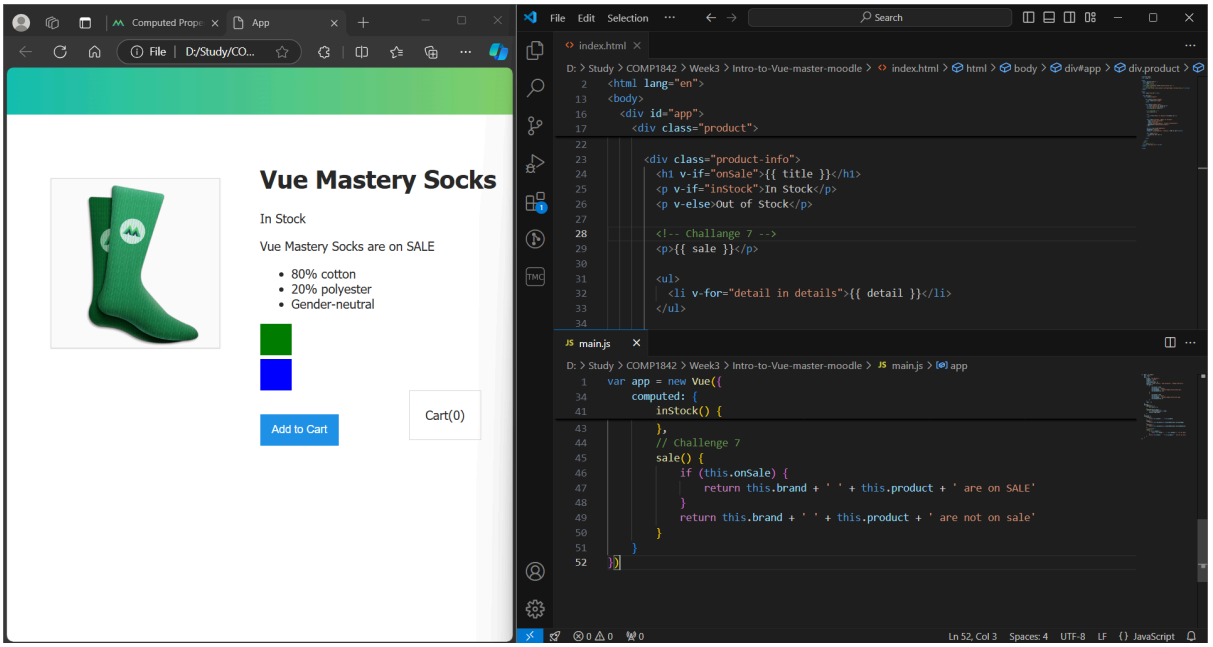

Figure 7.2: Lesson 7 with challenge

Lesson 5 introduced me to the v-on directive, which is used to listen for events like click and mouseovers. I learnt how to trigger methods when an event occurs, allowing me to interact with data and alter the user interface. Importantly, methods can accept parameters and utilise them to access the Vue instance's data, making it simple to alter state and call other instance methods.

In Lesson 6, I looked at how Vue dynamically binds data to style and class attributes. For example, a button's colour can vary depending on a condition such as inStock. Using expressions in class bindings, I learnt how to dynamically apply or delete classes, allowing styling to be responsive to data state. This is important for gathering customer input, such as removing a button when an item is out of stock.

Lesson 7 focused on computed properties that calculate a value. Instead of saving a value, computed properties calculate it on the fly using reactive data. This enables more efficient programming because computations take place only when the underlying data changes. I learnt how to use calculated properties to reorganise repetitious logic, such as picking a product variant's image, making the code easier to maintain.

3. Task 3

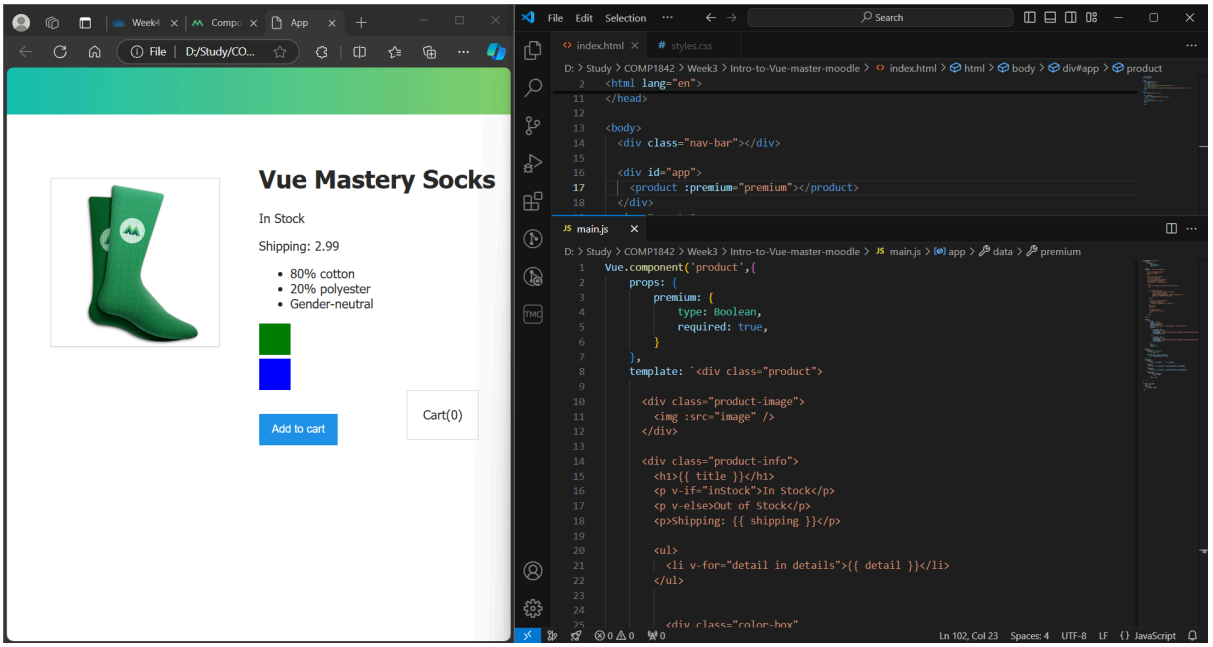

Figure 8.1: Lesson 8

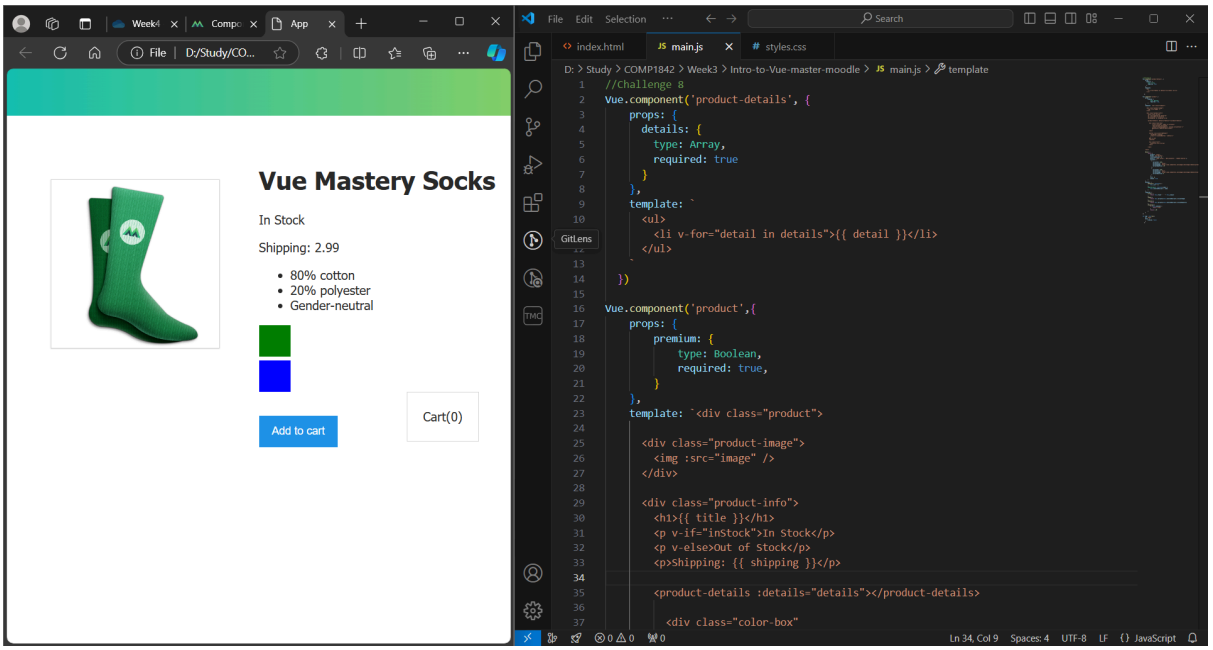

Figure 8.2: Lesson 8 with challenge

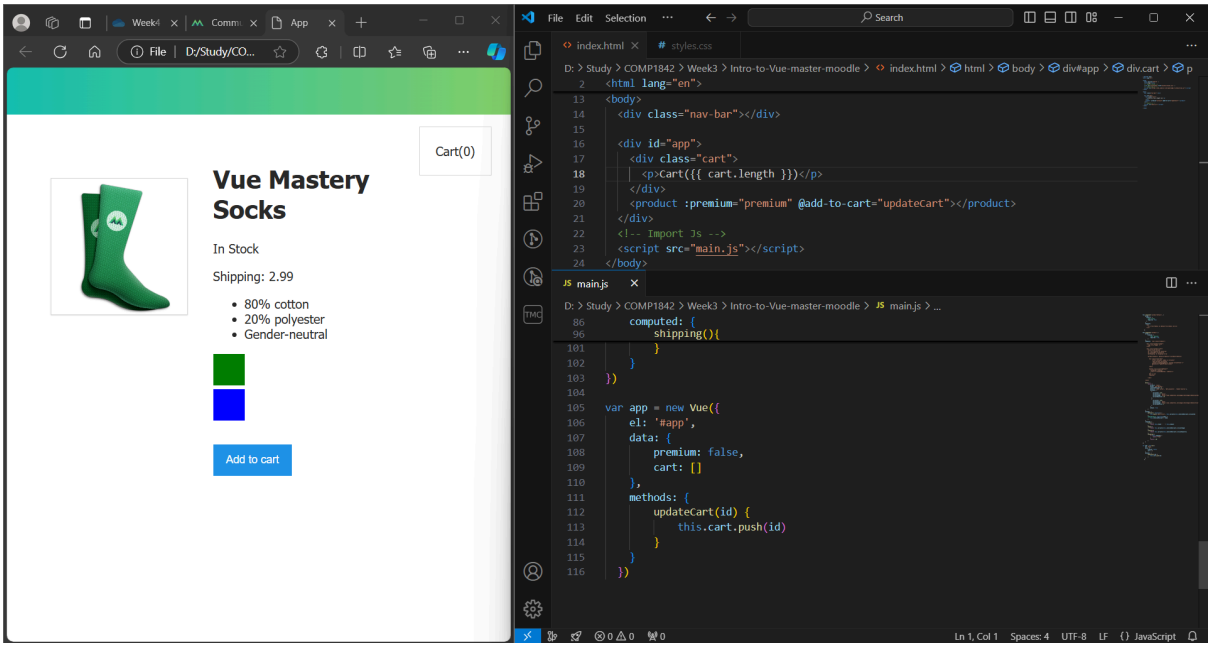

Figure 9.1: Lesson 9

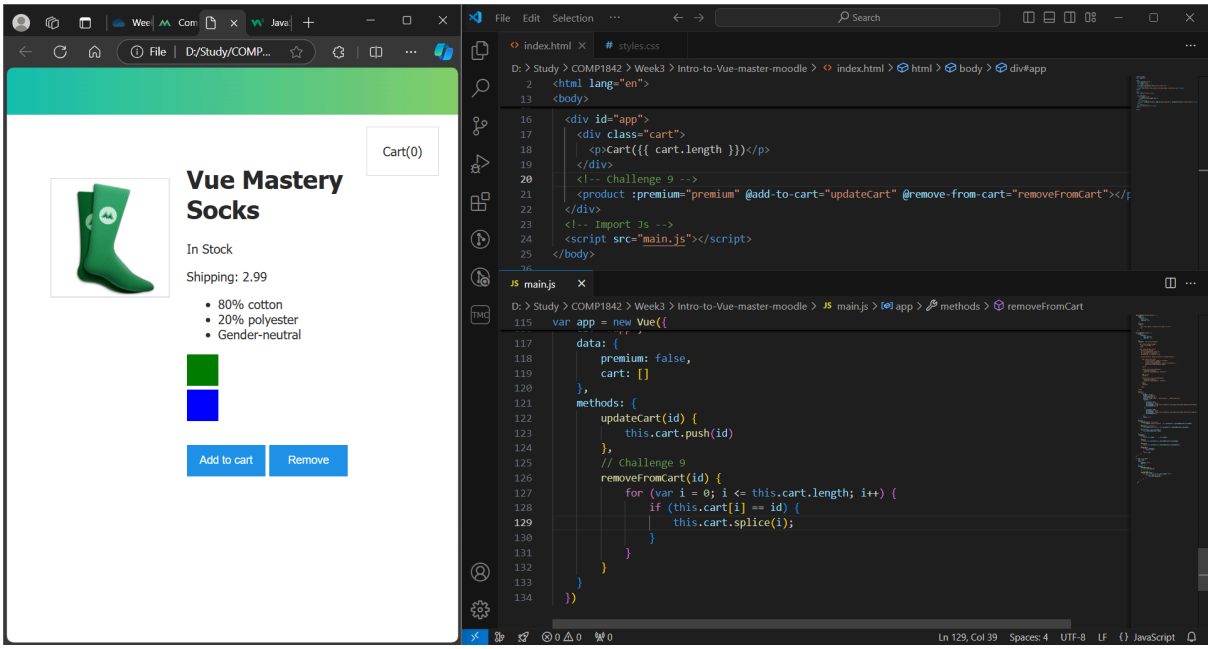

Figure 9.2: Lesson 9 with challenge

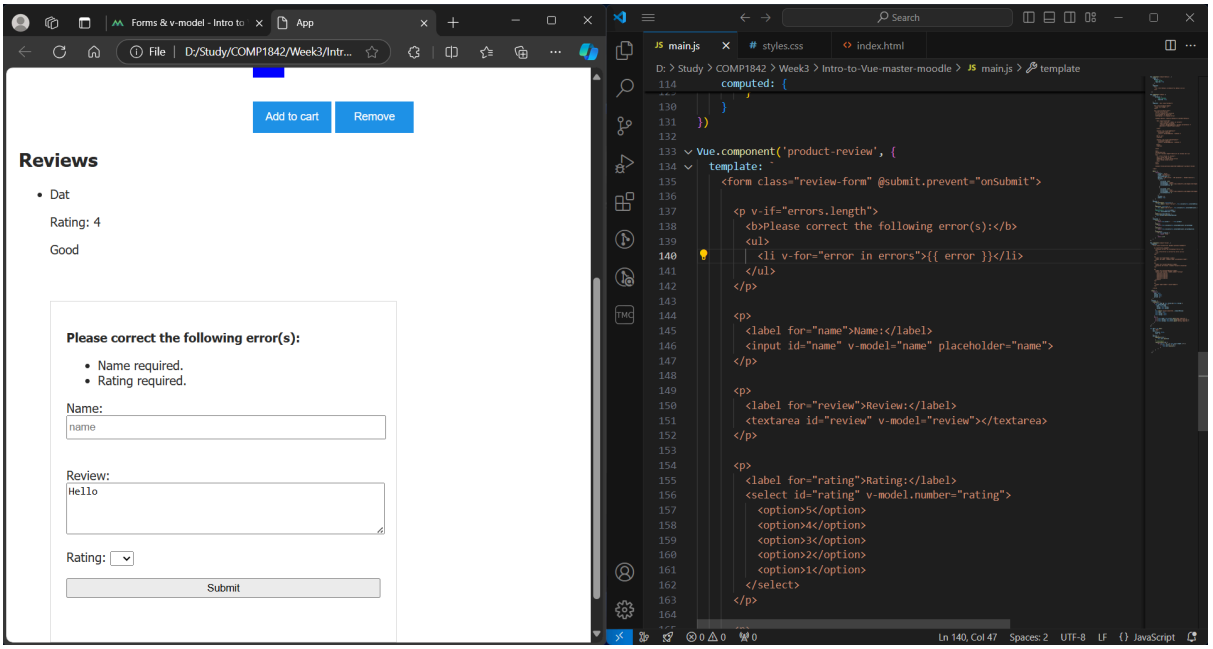

Figure 10.1: Lesson 10

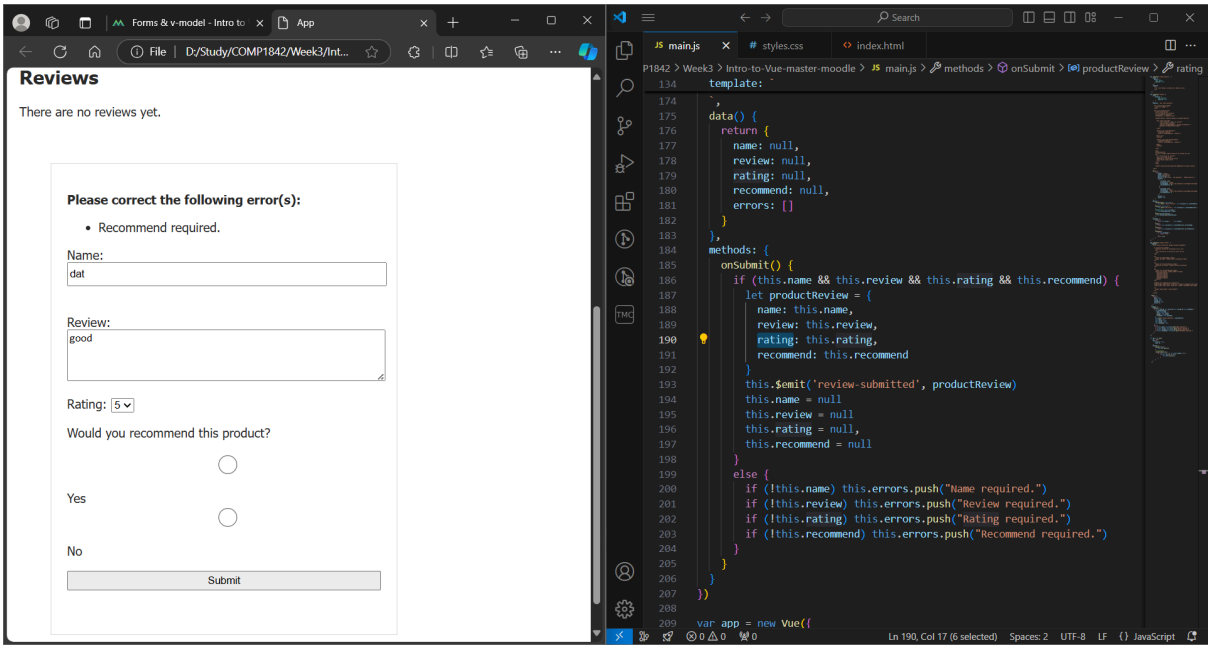

Figure 10.2: Lesson 10 with challenge

Lesson 8 explained to me how to develop modular source code to make the project more manageable. I registered product components and product details, which are reusable parts with individual behaviours. I also learned about the parent-child connection and how to use props to pass data between them. In the code , the parent passes the premium prop to the product component, and the @add-to-cart event listens for an emission from the child. The child component's addToCart() method then sends the variantId to the parent to update the cart.

In lesson 9, the cart is separated from the product because it has been moved to the root instance, and a communication link has been established between them by using "$emit" to transfer the data from child to parent. The parent listens for this event and modifies its state accordingly.

I also experimented with form handling by establishing two-way data binding between the component's data and form inputs using v-model in lesson 10. The form submission logic uses the onSubmit() method, where this.$emit('review-submitted', productReview) sends the review data to the parent when the form is submitted successfully.

4. MongoDB

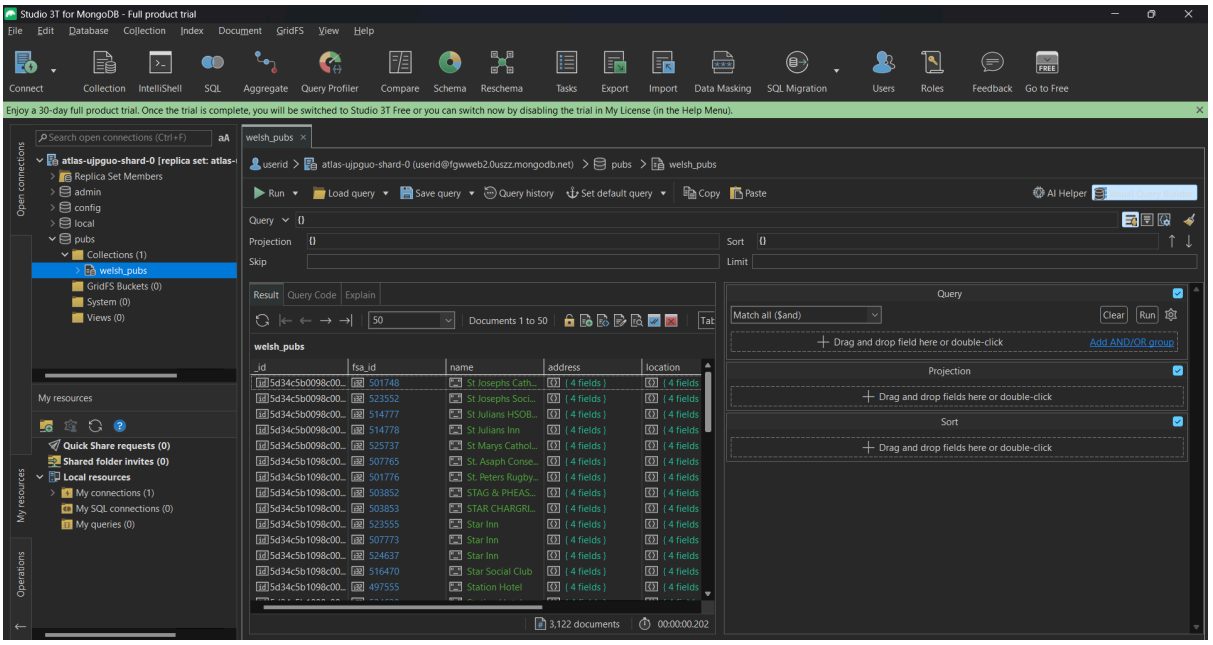

Figure 11: Lesson 4 exercise 1

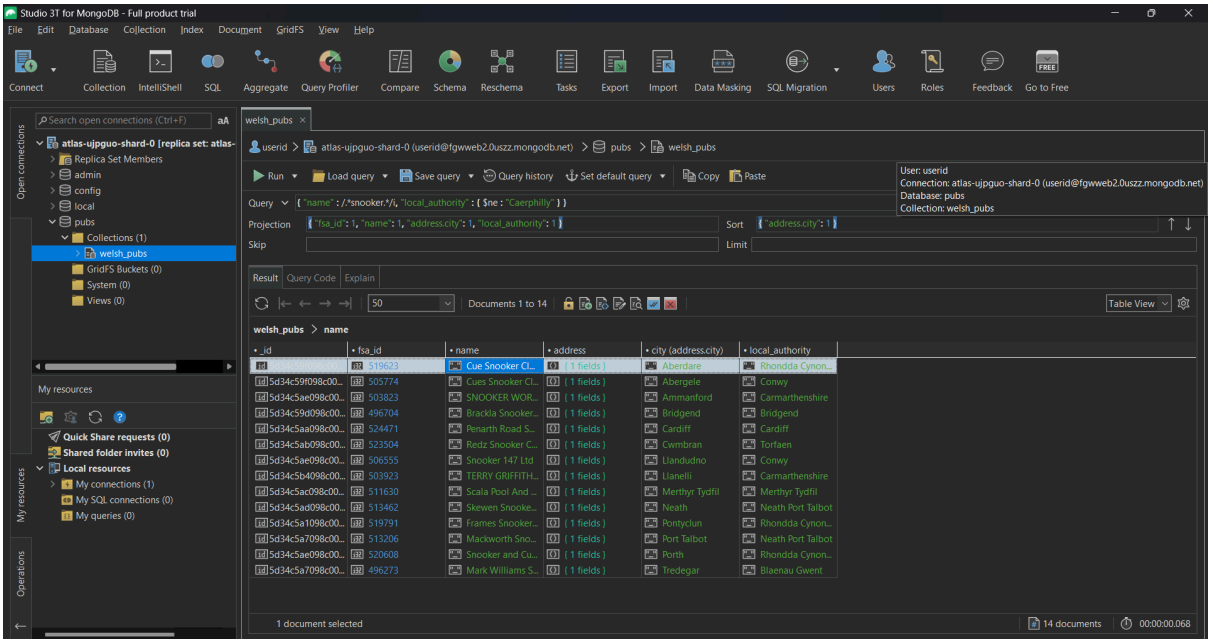

Figure 12: Lesson 4 exercise 3

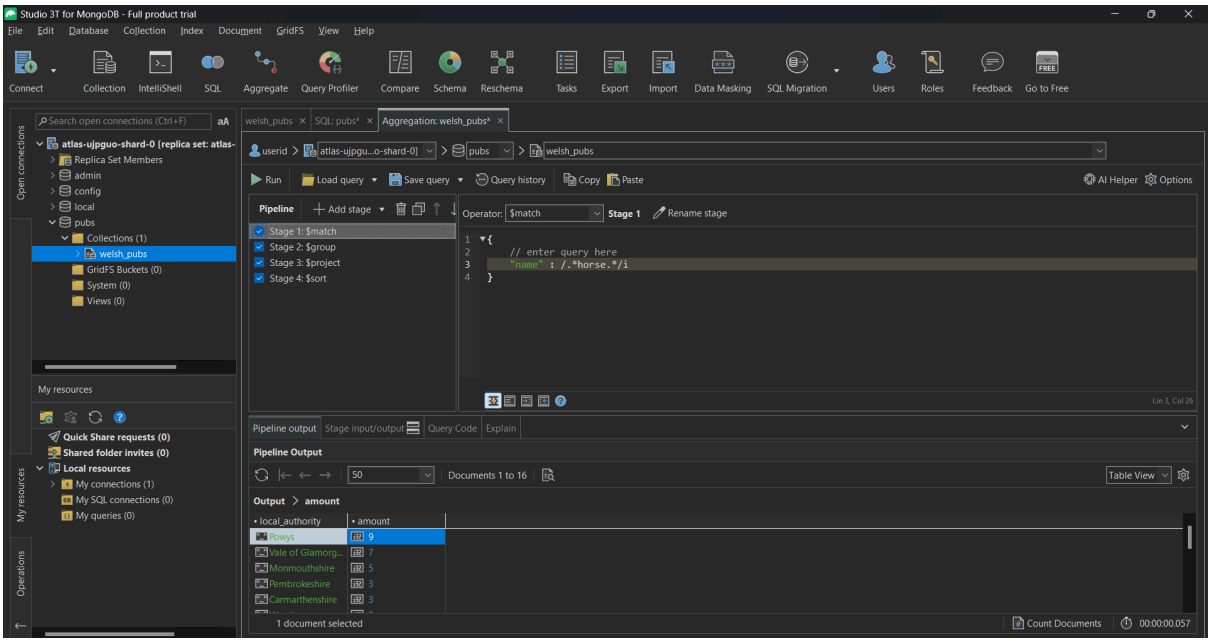

Figure 13: Lesson 5 exercise 3

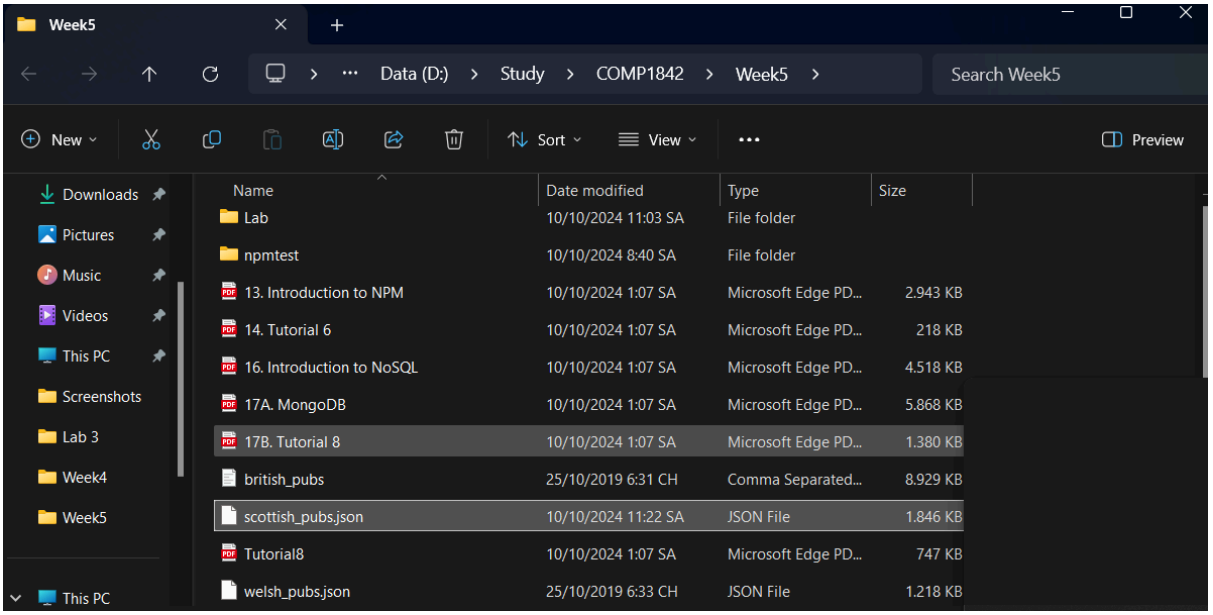

Figure 14: Lesson 6 exercise 2

MongoDB, one of the NoSQL databases, is a new approach to data storage that is significantly different from traditional relational databases. It is a document-oriented database with three hierarchical components: database, collection, and document. Strictly speaking, a database is made up of collections, and each collection contains documents, which are the main data structure in MongoDB. Documents are made up of field/value pairs that contain embedded arrays or documents, or even simpler values like characters or numbers. MongoDB can efficiently store and process a wide variety of data types due to the flexibility of its document model.

MongoDB also includes a unique feature as embedded documents, which allow a field to hold many fields within it. Compared to a relational database, this format requires fewer tables to store complex data relationships within a single document. JSON serialisation even makes embedded documents in MongoDB simpler to read and work with.

In addition to flexible data storage, MongoDB also provides aggregating procedures. Aggregation in MongoDB involves grouping data by a common field and performing operations on that group. This approach is quite similar to the GROUP BY clause and SQL aggregate functions. MongoDB's aggregation enables developers to execute complicated data transformations using an aggregation pipeline, which combines multiple operations in sequence.

Appendix

https://github.com/dat7554/Web-Programming-2