Report of Appendix 1, Appendix 2 and Appendix 3

Student Name/ Student ID	Trần Hoàng Vũ - GCS220851
Unit Number and Title	Web Development 2 - COMP1842
Academic Year	2024-2025
Project Title	Report of Appendix 1, Appendix 2 and Appendix 3

TABLE OF CONTENTS

Task 1

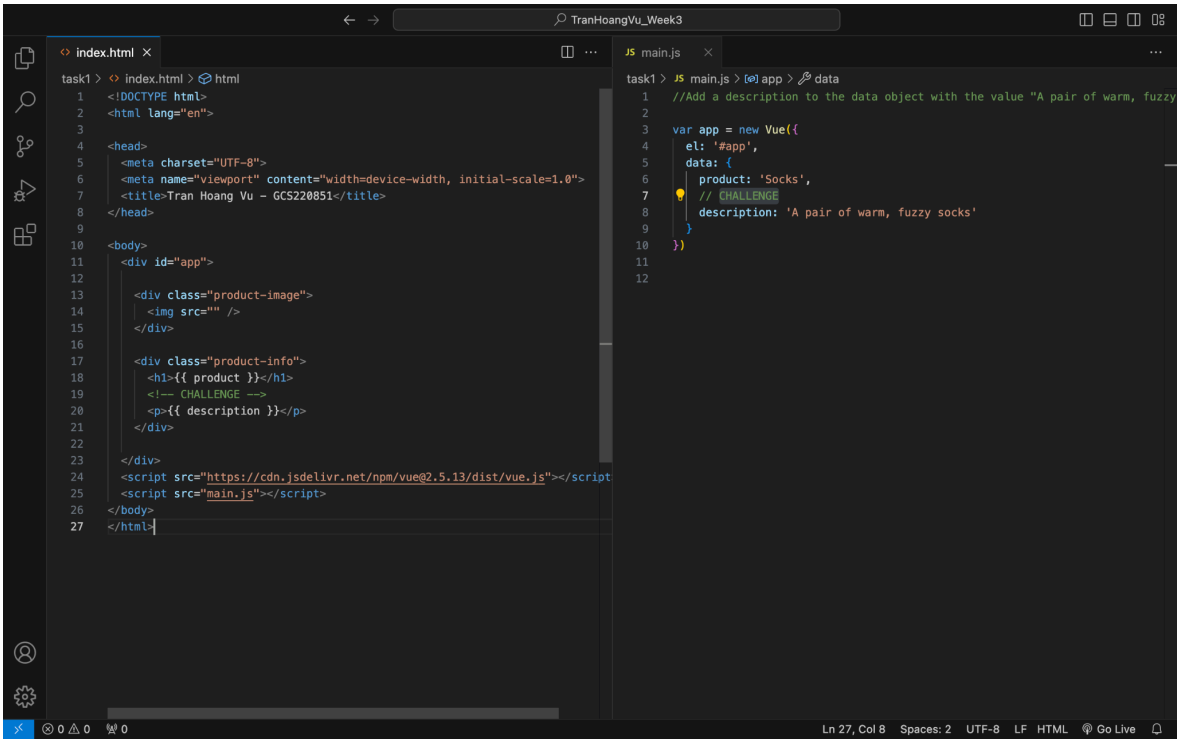

Figure 1: Lesson and Challenge of Task 1

Figure 2: Result of lesson and challenge of task 1

Task 2

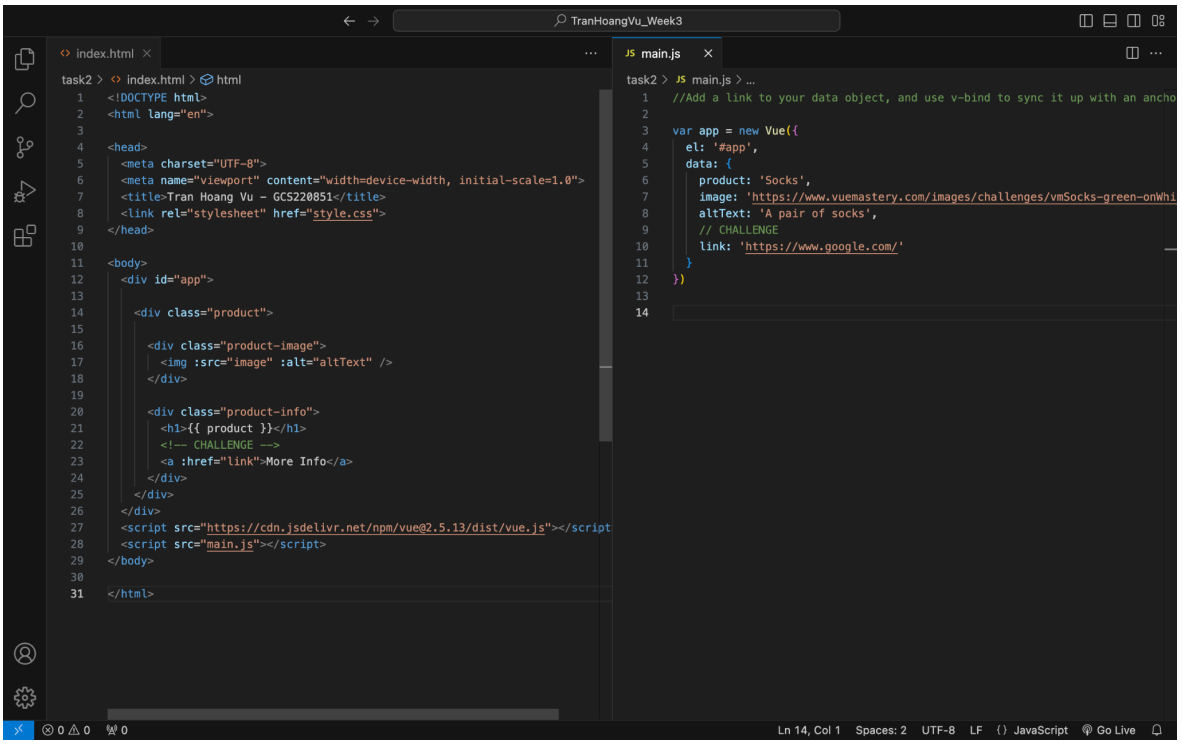

Figure 3: Lesson and Challenge of Task 2

Figure 4: Result of lesson and challenge of Task 2

Task 3

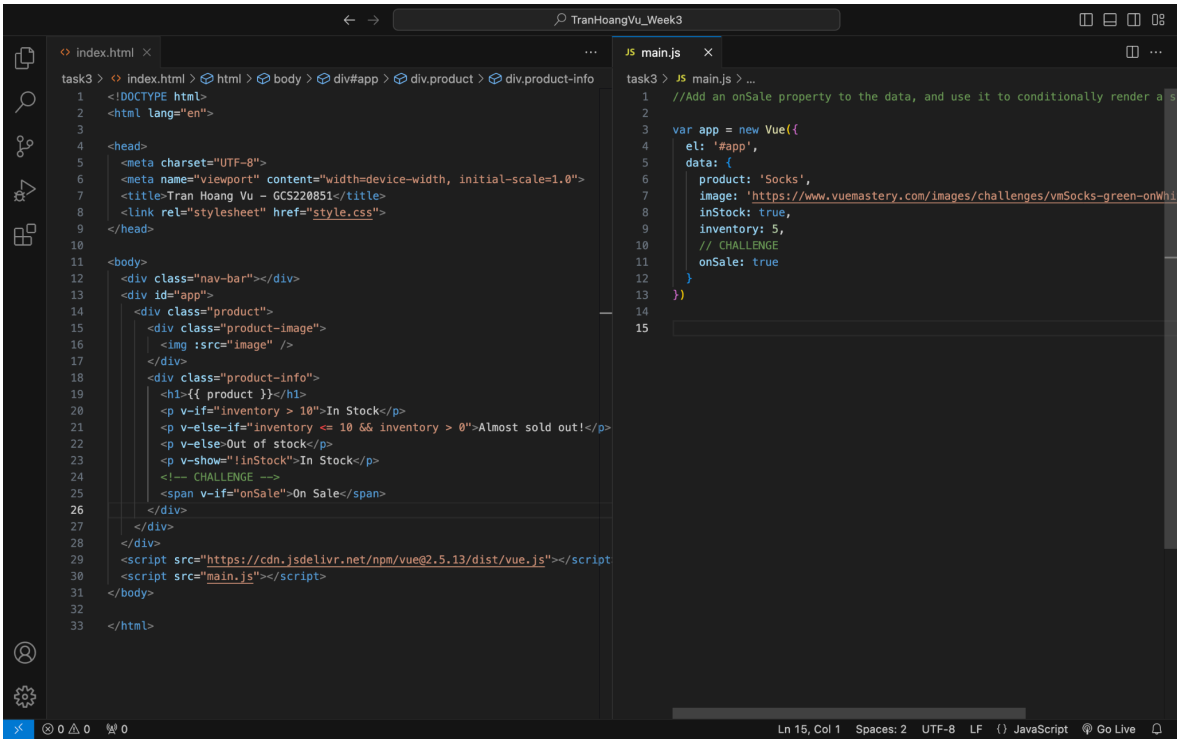

Figure 5: Lesson and Challenge of Task 3

Figure 6: Result of lesson and challenge of Task 3

Task 4

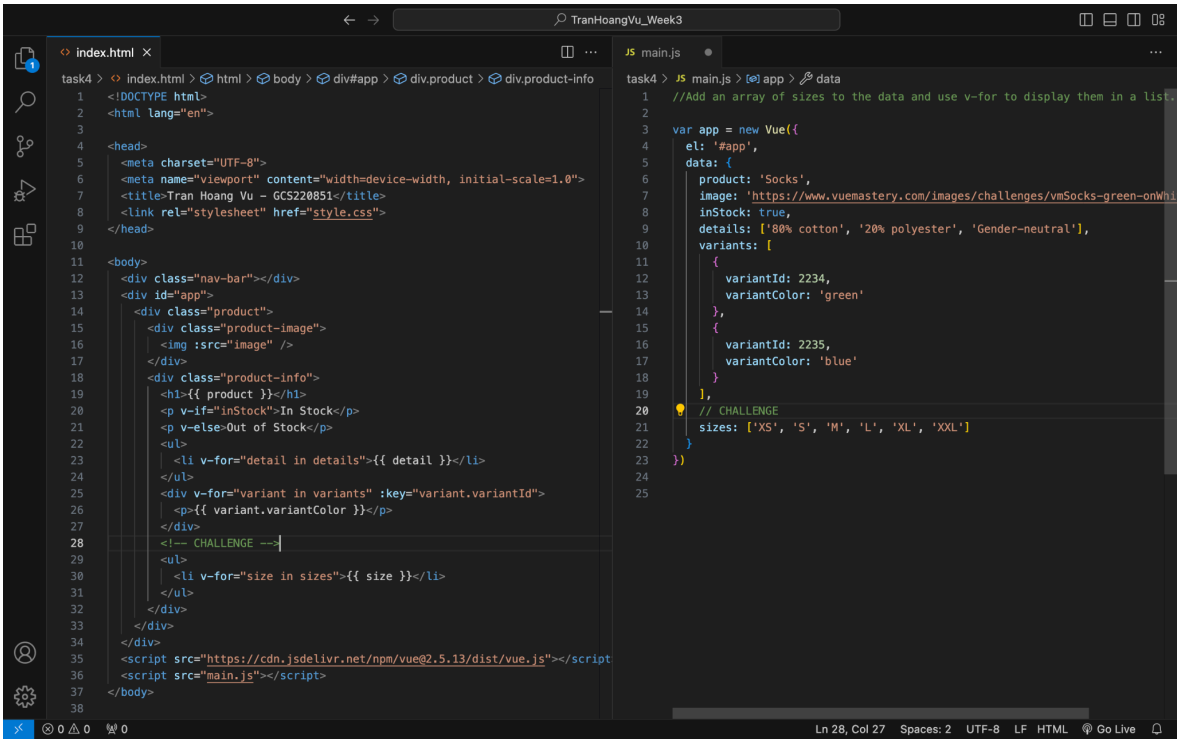

Figure 7: Lesson and Challenge of Task 4

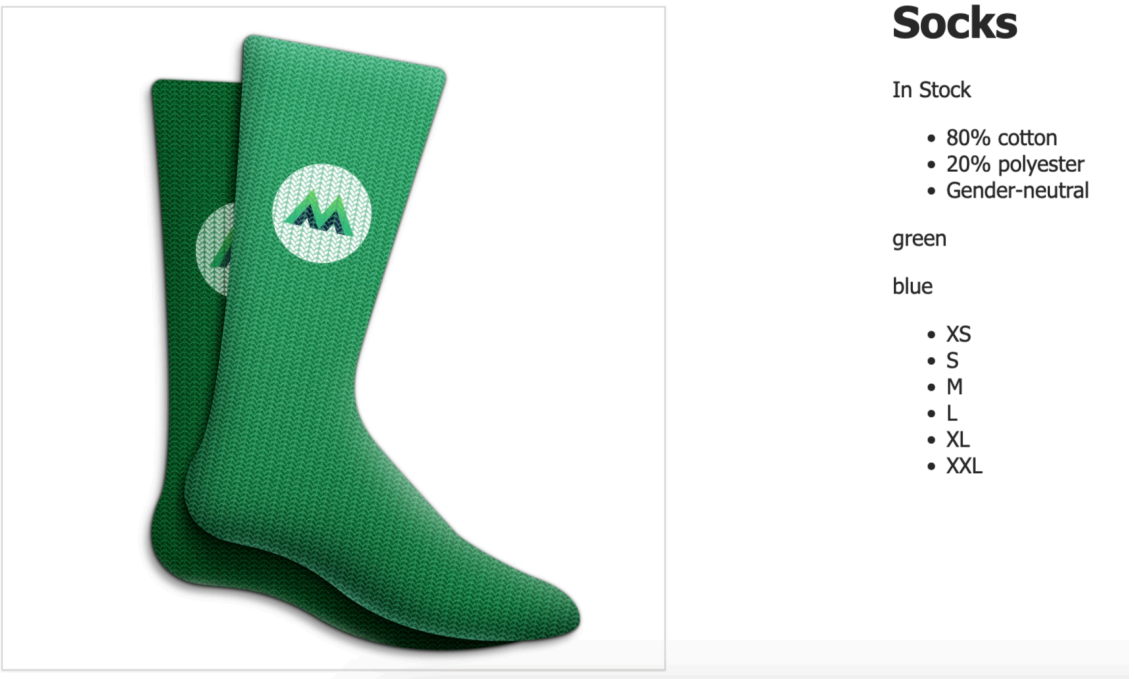

Figure 8: Result of lesson and challenge of Task 4

Appendix 1

In this lab, I gained a solid understanding of building Vue.js applications, focusing on creating a Vue instance and leveraging its core features. Central to this is Vue's reactivity system, which allows data to automatically update the UI using simple syntax like {{ }}. This data-binding capability is crucial for creating dynamic and responsive interfaces.

I also learned how to use Vue's directives, such as v-bind, which binds data to HTML attributes, enabling seamless interaction between the data model and the view. Additionally, conditional rendering with directives like v-if and v-show allows for efficient control over element visibility based on the application's state. This flexibility in rendering enhances user experience without unnecessarily altering the DOM structure.

Another key feature I explored was v-for, which enables looping through arrays to display data dynamically. Assigning unique keys to elements in these loops ensures Vue efficiently updates only the necessary parts of the interface, improving performance in larger applications.

Looking ahead, these skills will be invaluable for building scalable, reactive, and efficient web applications. The reactivity system will allow me to create user interfaces that respond seamlessly to changes, while Vue’s directives will help me optimize performance and manage complex interactions in real time. These features will be essential for developing interactive, data-driven applications that provide a smooth user experience.

Task 5

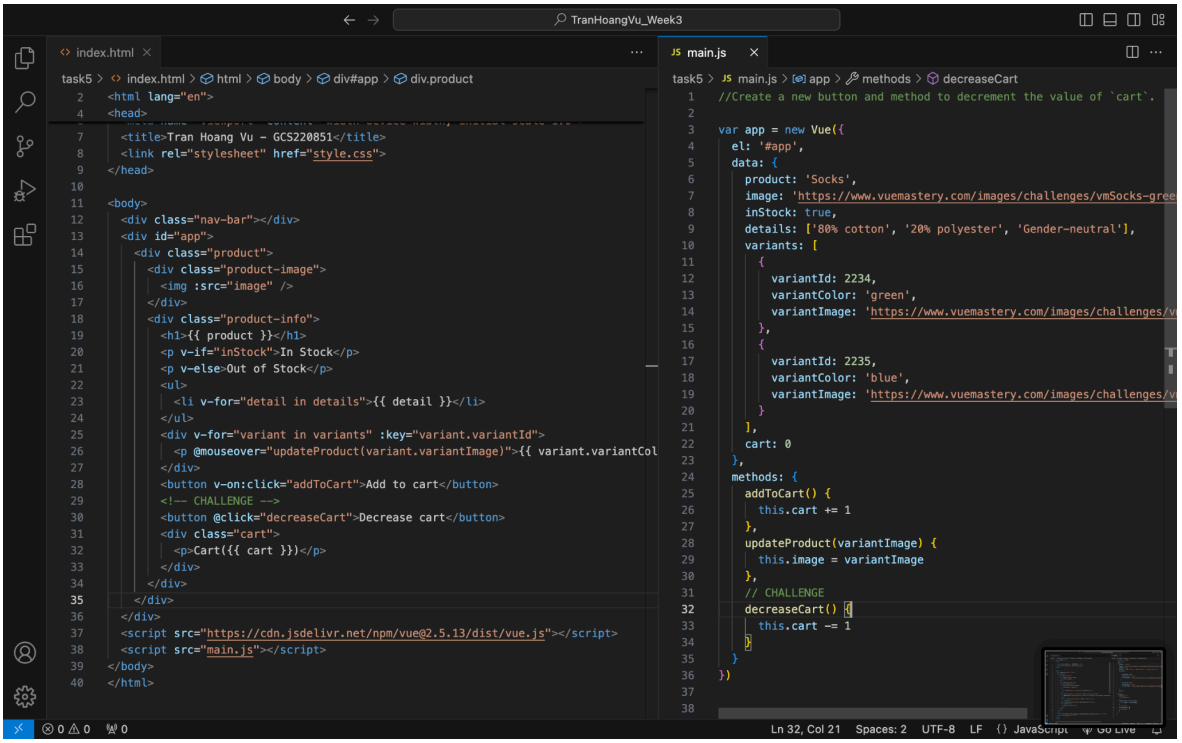

Figure 9: Lesson and Challenge of Task 5

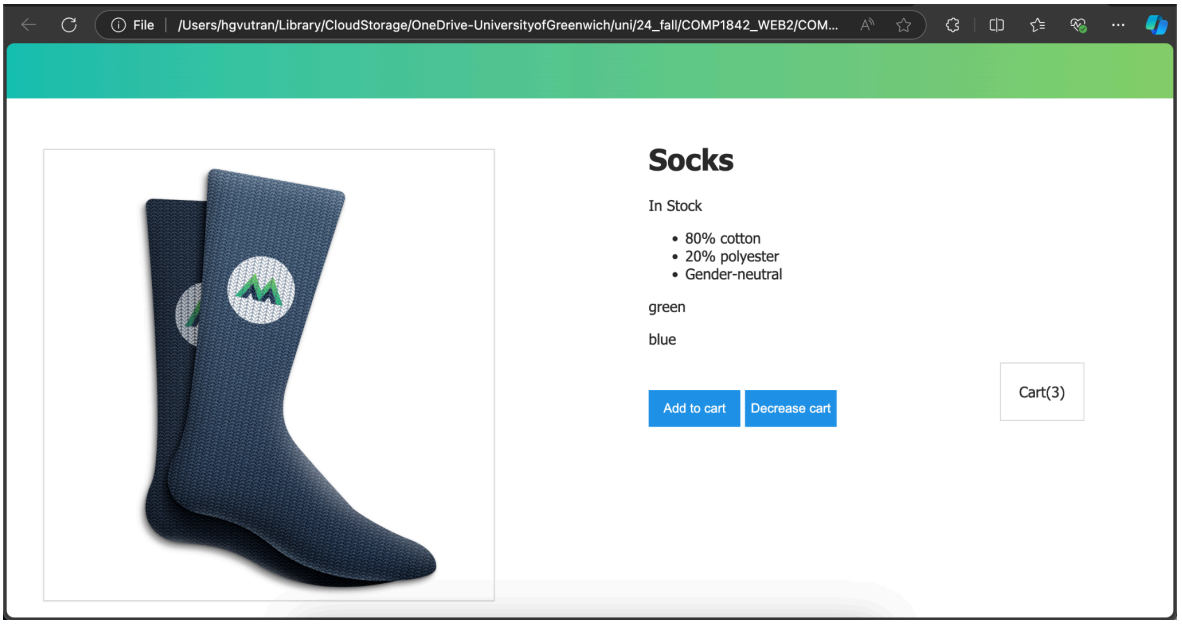

Figure 10: Result of lesson and challenge of Task 5

Task 6

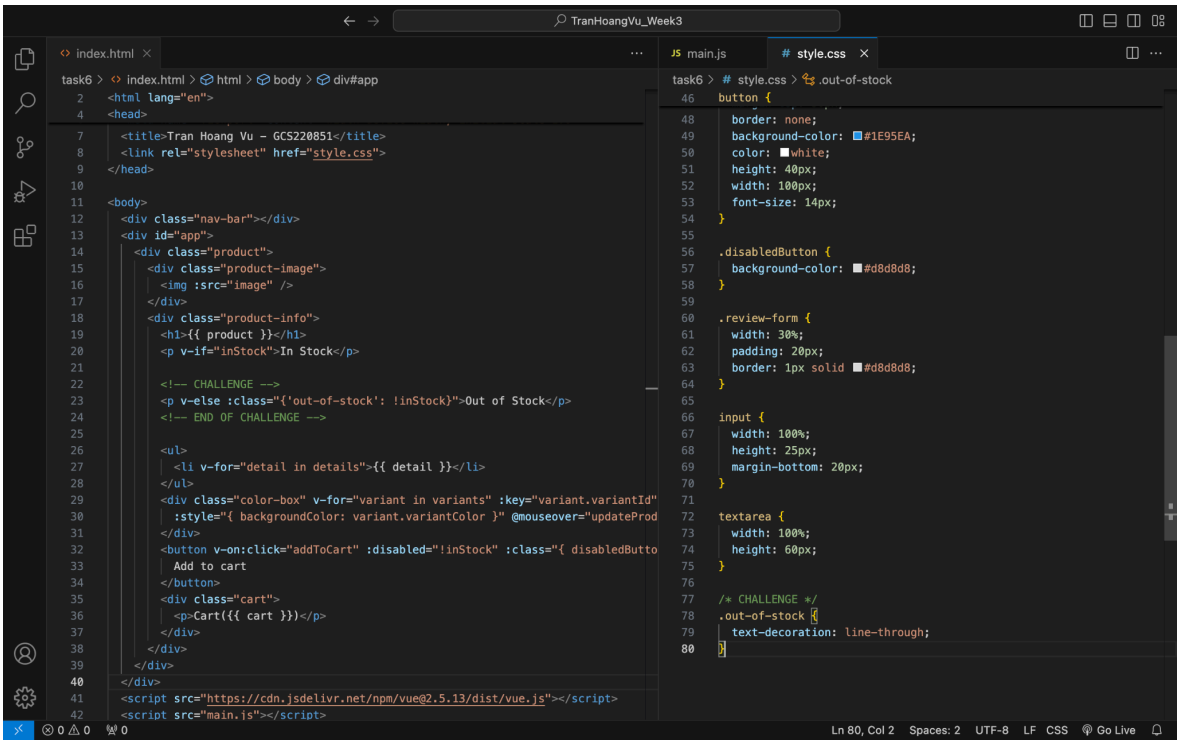

Figure 11: Lesson and Challenge of Task 6

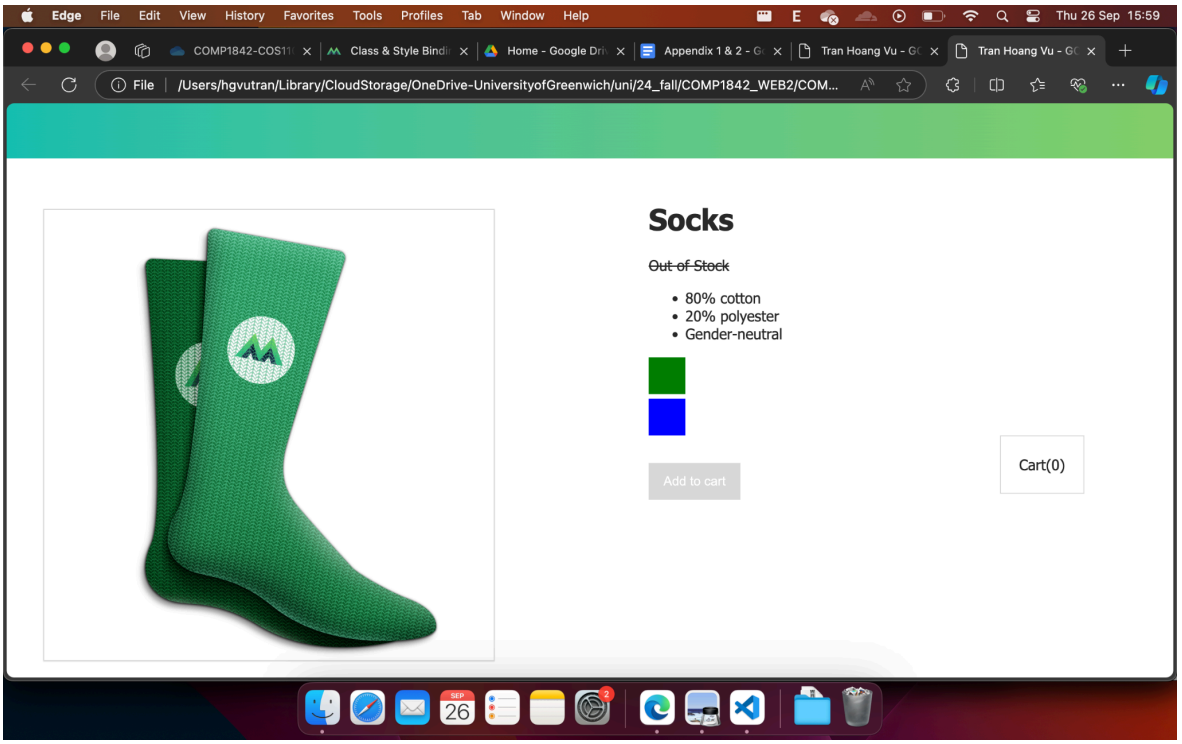

Figure 12: Result of lesson and challenge of Task 6

Task 7

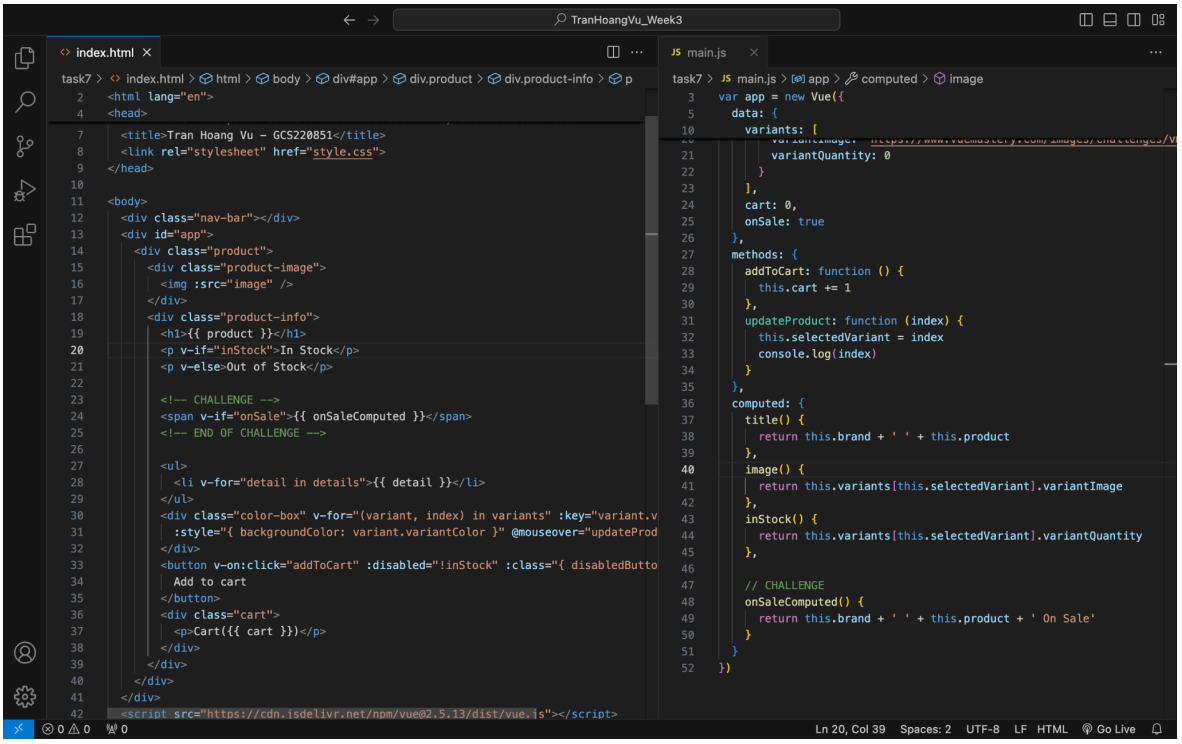

Figure 13: Lesson and Challenge of Task 7

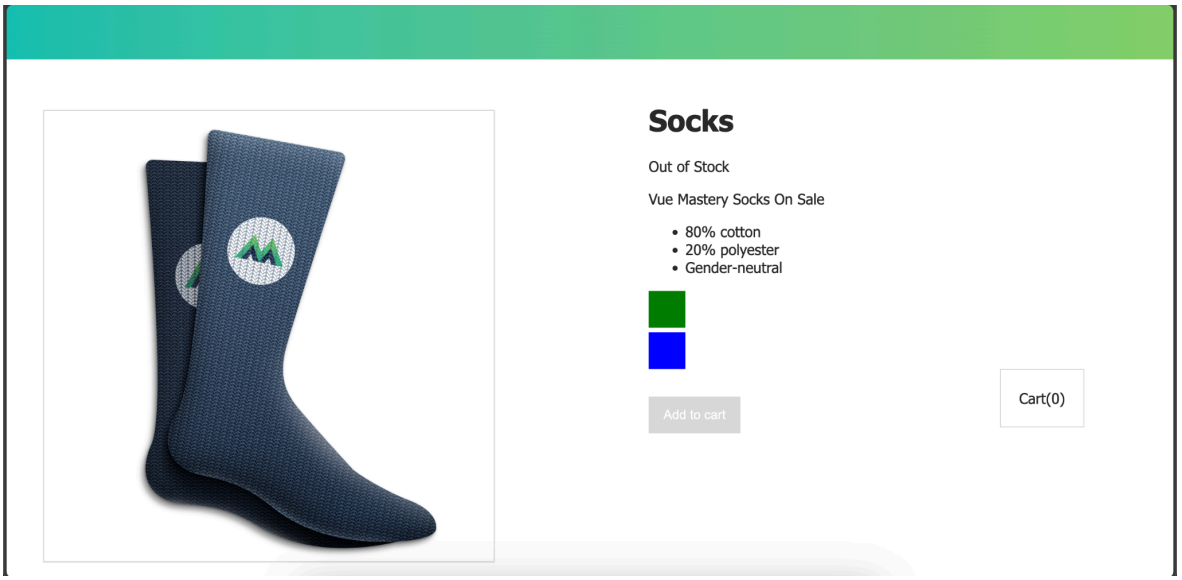

Figure 14: Result of lesson and challenge of Task 7

Appendix 2

In this lab, I delved into key advanced features of Vue.js that improve both interactivity and performance. I focused on event handling, dynamic styling, and using computed properties for efficient data management.

By learning how to use the v-on directive, I gained the ability to efficiently handle user-triggered events like clicks and keypresses, enhancing application responsiveness. Dynamic data binding to style and class attributes allowed me to control the visual appearance of elements based on real-time application state, making the interface more adaptable and user-friendly.

A significant highlight was computed properties, which dynamically calculate values based on dependencies and are only recalculated when necessary. This caching ability is particularly useful for operations that would otherwise be computationally expensive, making Vue.js applications more efficient.

Looking ahead, these skills will be essential for building interactive, scalable applications. Handling events effectively will improve user engagement, while dynamic styling ensures that the interface can adapt to different states seamlessly. Computed properties will be especially valuable in larger applications, where performance optimization is critical for complex data-driven features like filtering large datasets or managing real-time updates. Overall, these techniques will help me create responsive, dynamic, and performance-optimized Vue applications in the future.

Task 8

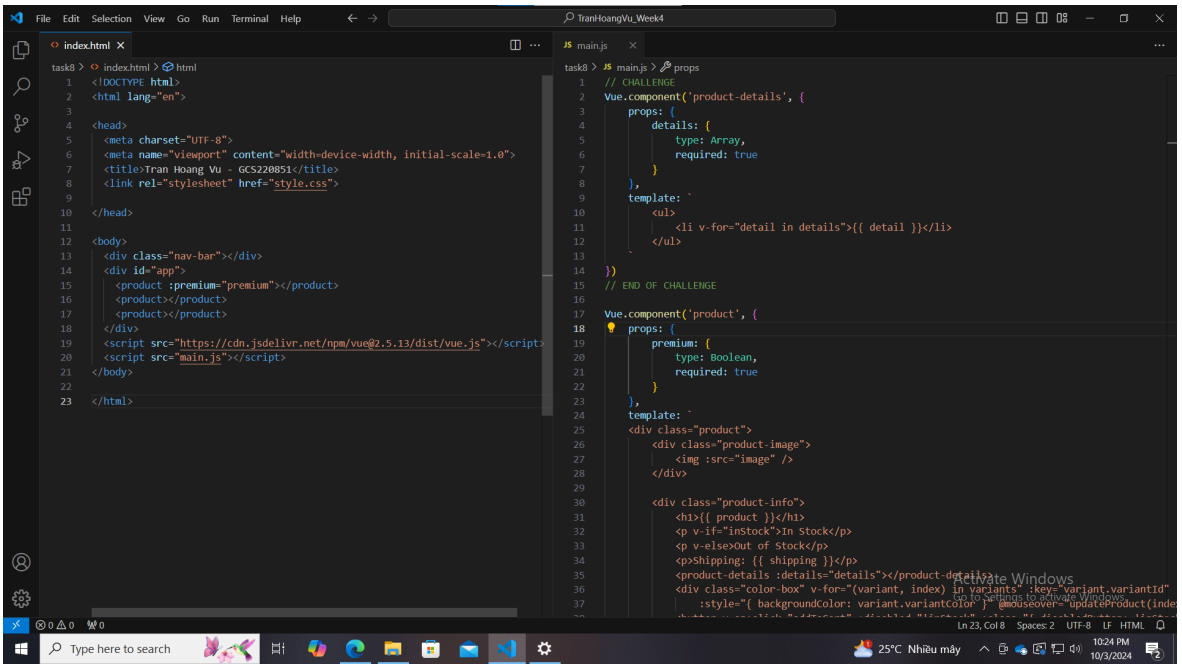

Figure 15: Lesson and Challenge of Task 8

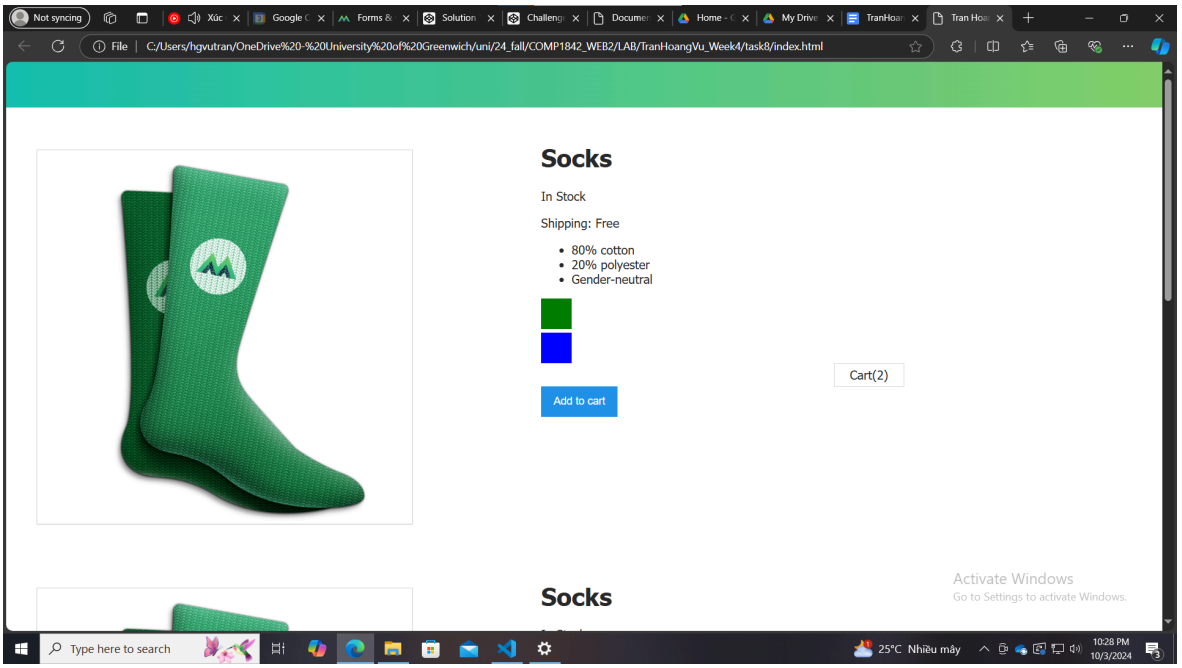

Figure 16: Result of lesson and challenge of Task 8

Task 9

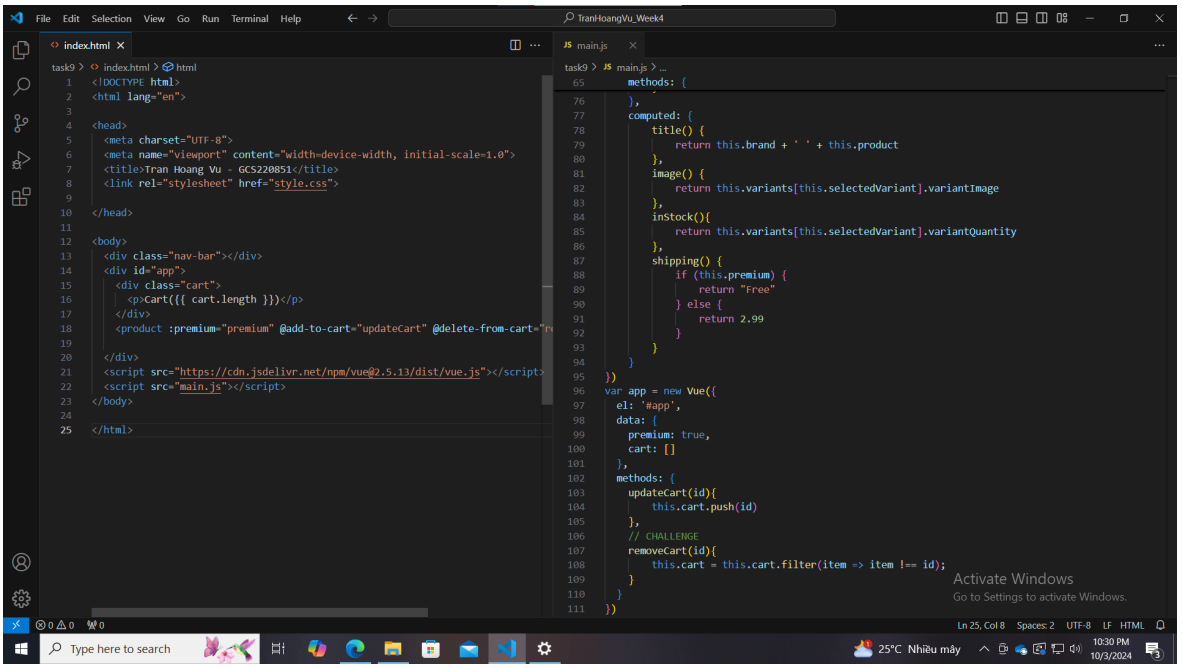

Figure 17: Lesson and Challenge of Task 9

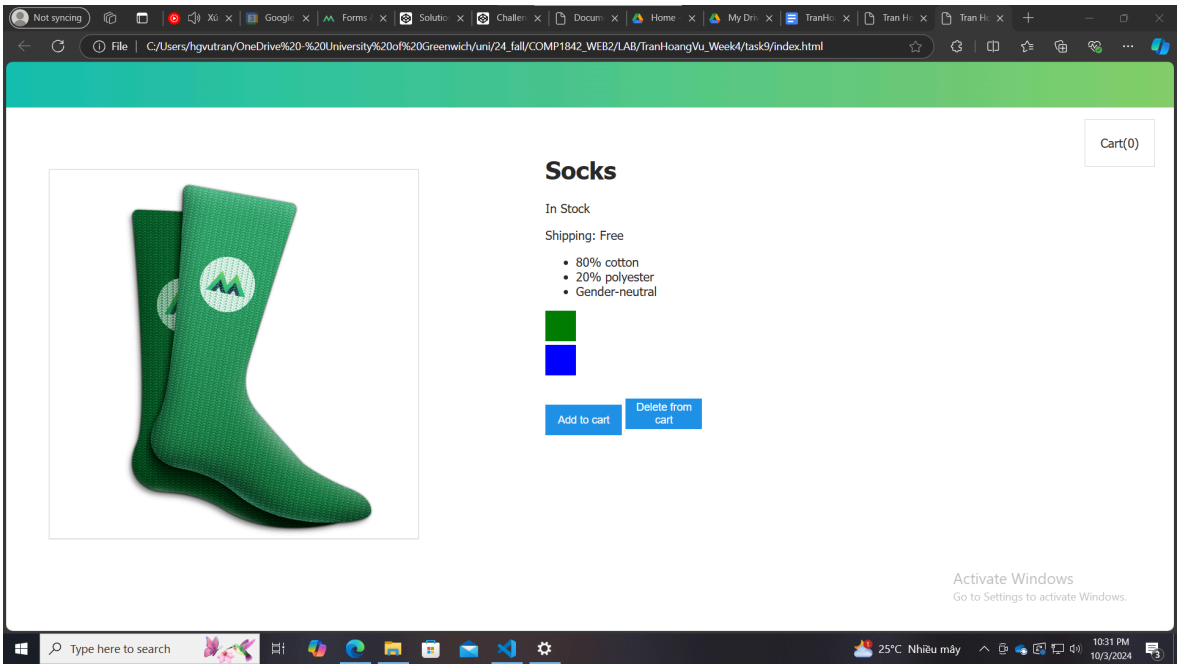

Figure 18: Result of lesson and challenge of Task 9

Task 10

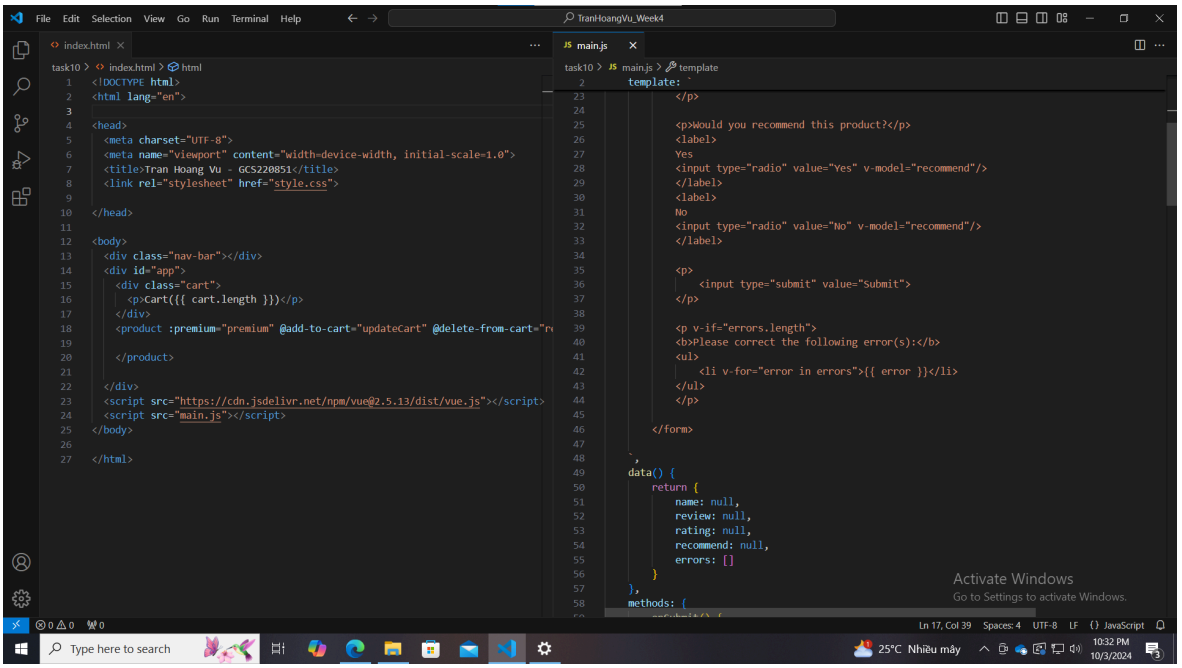

Figure 19: Lesson and Challenge of Task 10

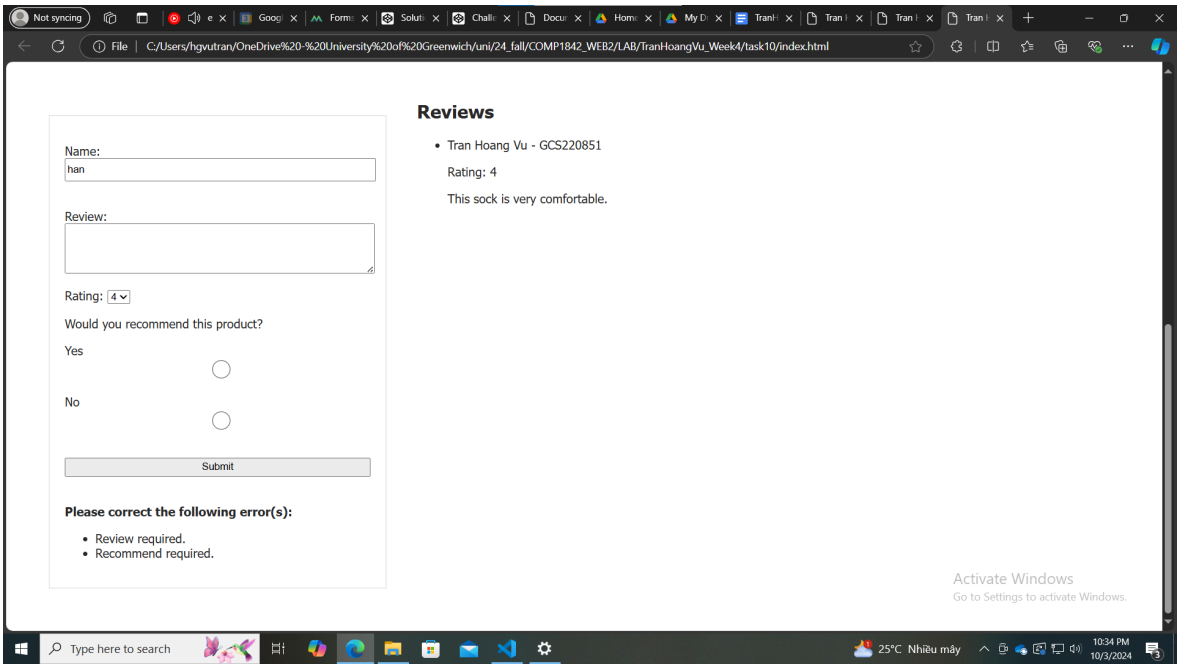

Figure 20: Result of lesson and challenge of Task 10

Appendix 3

Through Appendix 3, I learned how to create components, which makes it easier to reuse code. In my view, this is similar to creating functions in other programming languages. This approach has been extremely helpful in managing code and fixing bugs. Additionally, I discovered how to pass data from parent tags to child tags using props, as well as how to send data back from child components to parent components. This concept of data flow between components has been particularly useful in improving my understanding of React's architecture.

Moreover, I learned how to create forms and handle custom validation, which is crucial in ensuring that the user input meets specific criteria. Understanding how to set up and manage forms efficiently can significantly reduce the likelihood of errors or invalid data submissions.

Looking forward, I can see how these skills will help me in larger projects. For instance, component reusability will streamline the development of scalable applications by reducing redundant code. Additionally, mastering data flow between components will allow me to build more interactive and dynamic user interfaces, where parent and child components can seamlessly communicate. The ability to handle forms with custom validation could be vital for building applications that require user authentication, payment processing, or any other data-sensitive operations, ensuring a smooth user experience.

Appendix 4

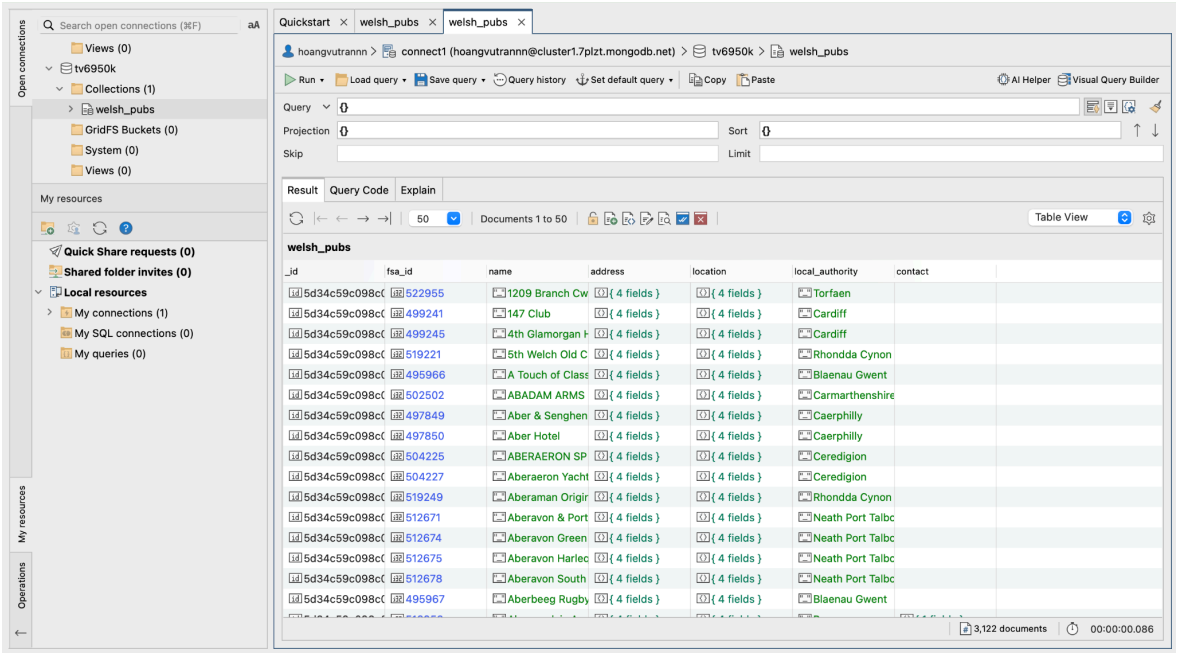

Figure: Lesson 4 Exercise 1

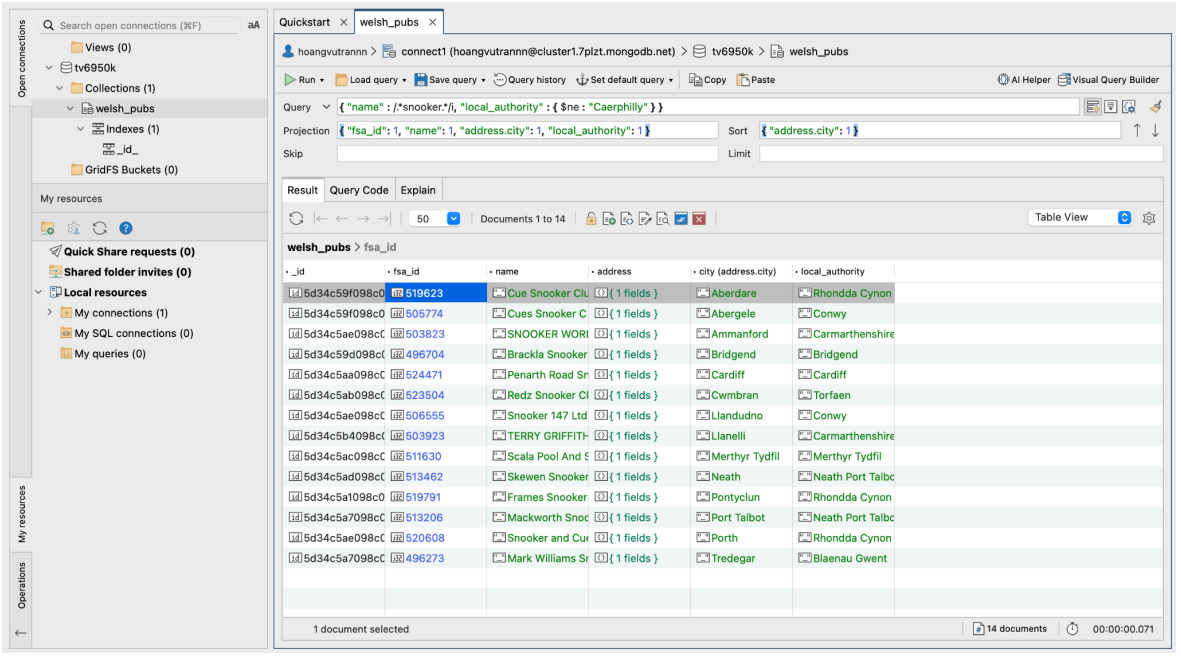

Figure: Lesson 4 Exercise 3

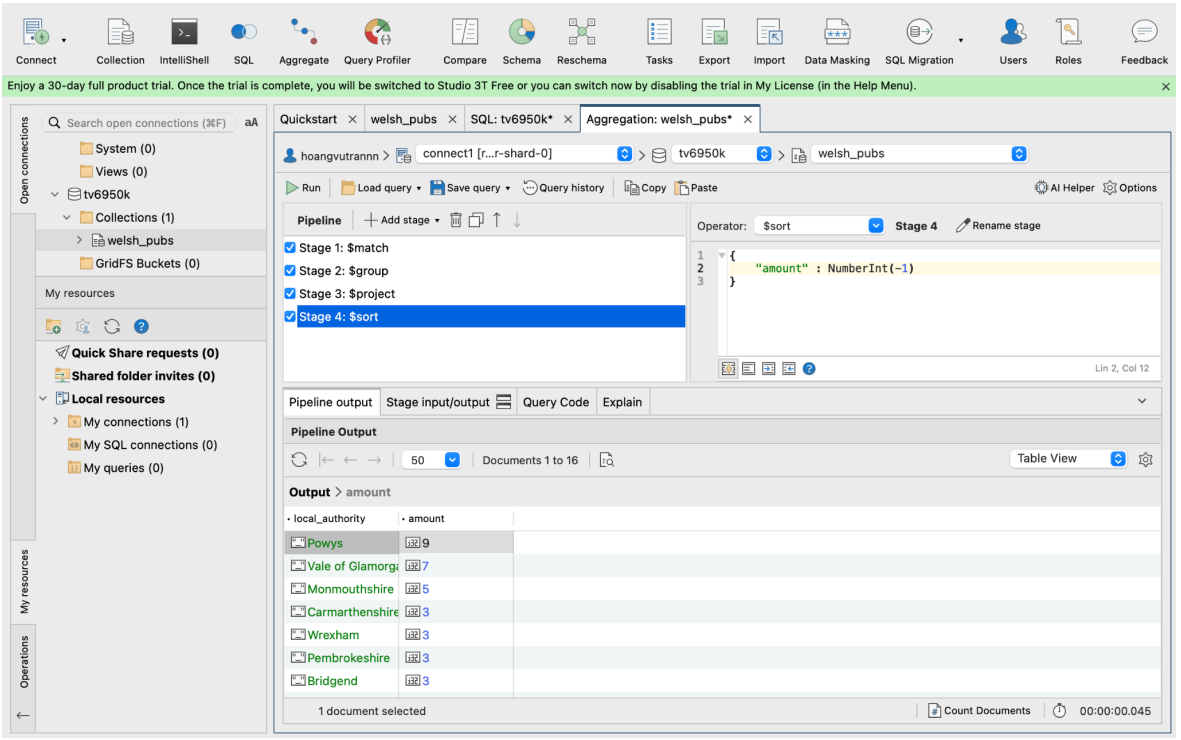

Figure: lesson 5 exercise 3

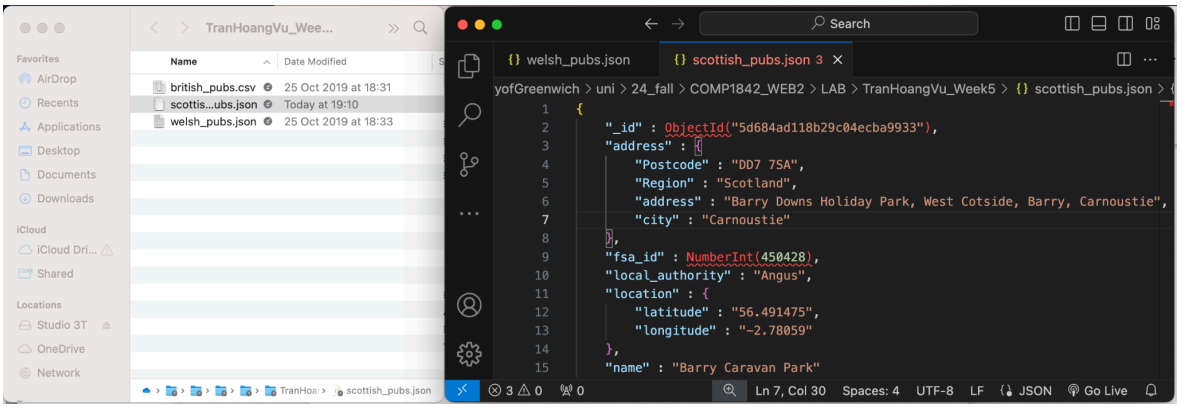

Figure: lesson 6 exercise 2

Understanding NoSQL and MongoDB

NoSQL, which stands for "Not Only SQL," refers to a category of database management systems that provide flexible schema design and scalable performance, making them suitable for handling large volumes of unstructured or semi-structured data (McCool, 2019). Unlike traditional relational databases, which use fixed schemas and SQL for data manipulation, NoSQL databases offer various models such as document, key-value, column-family, and graph databases (Chodorow, 2013). This flexibility enables organizations to adapt to changing data requirements and support high-velocity data ingestion.

MongoDB is a leading NoSQL database that utilizes a document-oriented data model (MongoDB Official Documentation, n.d.). In MongoDB, data is stored in JSON-like

format called BSON (Binary JSON), allowing for easy representation of complex data structures. Each document in a collection can have a different structure, accommodating diverse data types and providing the flexibility to evolve the data model over time. This makes MongoDB particularly well-suited for applications requiring quick iterations, such as content management systems and real-time analytics.

One of the primary advantages of MongoDB is its scalability. It supports sharding, which allows data to be distributed across multiple servers, ensuring high availability and improved performance (Chodorow, 2013). Additionally, its rich query capabilities and built-in support for indexing enhance data retrieval efficiency.

In summary, NoSQL databases, particularly MongoDB, empower organizations to manage vast amounts of data with agility and efficiency, making them essential for modern application development and data-driven decision-making (McCool, 2019).

References

Jahr, A. (2024). *Intro to Vue 2 - Intro to Vue 2 | Vue Mastery*. [online] Vue Mastery. Available at: https://www.vuemastery.com/courses/intro-to-vue-js [Accessed 26 Sep. 2024].

MongoDB: The Definitive Guide by Kristina Chodorow (2013).

R. McCool, "NoSQL Databases: A Survey and a Future Research Direction," Journal of Computer Science, vol. 10, no. 2, pp. 1-15, 2019.

MongoDB Official Documentation. (n.d.). Retrieved from https://www.mongodb.com/docs [Accessed 10 Oct. 2024].