Elaeagnus Latifolia

Group 19 - Fullstack Web Application

s3977747- Le Nguyen My Chau
s3978165- Tran Tuan Trung
s3978290- Dong Manh Duc
s3978506- Nguyen Ba Duc Manh
s3927777-Do Thuy Linh

COSC2430 - Web Programming

RMIT

Table of Contents

I. Introduction..... 1

II. Project Description..... 1

III. Implementation Details.....2

IV. Conclusion.....11

V. Peer Evaluation..... 11

VI. Appendix..... 12

RMIT

I. Introduction

This project involves the development of an eCommerce system based on Lazada's model, catering to three main user roles: customers, vendors, and shippers. Users can register and log in, with specific validation constraints for each role. Vendors can add and view products, customers can browse and order products, and shippers manage the delivery process. The system ensures security through password hashing and stores user information in a MongoDB Atlas Database. The project aims to provide a user-friendly and secure eCommerce platform with responsive design for various devices, enabling seamless registration, login, and role-specific functionalities for users.

II. Project Description

- **User Authentication and Authorization:** User registration and login are implemented with strong password hashing techniques to ensure security. Aside from that, the system enforces role-based access control, allowing different functionalities based on the user's role (customer, vendor, shipper, admin).
- **Responsive Design:** The website is designed to be responsive, adapting to various device types and screen sizes (small mobile screens, medium tablets, and large desktop/laptop screens).
- **Data Storage and Management:** User data, including profile pictures, is stored in a MongoDB Atlas Database, ensuring data persistence and scalability, Product information, orders and distribution hubs are efficiently managed and stored in the database.
- **Vendor Pages:** Vendors can add new products with details such as name, price, image, and description and view their added products and manage their product listings.
- **Customer Pages:** Customers can browse and search for products, applying filters based on price and product name, the products matching the search criteria are fetched efficiently from the server. Aside from that, customers can view product details and add items to their shopping cart and a shopping cart system manages the selected products for purchase.
- **Shipper Pages:** The shipper system offers secure login, an organized order database, a user-friendly dashboard for managing active orders, real-time notifications, data security, optimized performance, error handling, scalability planning, regulatory compliance, backups, deployment automation, and robust testing for reliability.
- **Order Management:** Customers can place orders, and the system assigns the order to a distribution hub for processing where shippers can view and manage active orders assigned to their hub, updating order status.
- **Profile Management:** Users, regardless of their role, can access their "My Account" page to view and modify their information, including profile pictures.Moreover, The system enforces redirection to the login page for unauthorized users attempting to access the "My Account" page.

- **Validation and Error Handling:** The system performs client-side and server-side validation to ensure data integrity and enforce constraints. Aside from that, Error messages and validation feedback are provided to users for a smooth and error-free experience.
- **File Upload and Storage:** Profile pictures and other images can be uploaded and stored either as binary data in the MongoDB Atlas DB or as files in the public folder with corresponding file paths in the database.
- **Security Best Practices:** Passwords are securely hashed and stored to protect user credentials.Vulnerabilities, such as SQL injection and cross-site scripting (XSS), are mitigated through proper data handling and validation.
- **NPM Packages:** The selection of NPM packages for password validation and other critical functionalities is justified and documented.
- **User Experience (UX):** The website's design and layout focus on providing an intuitive and user-friendly experience for all user roles.
- **Distribution Hubs:** Distribution hubs are created with unique names and addresses, and shippers can select their assigned hub during registration.

III. Implementation Details

● Backend Implementation and Decisions

Note: In various features of the project, the associated package will be referenced.

a. Login and Registration

This is arguably one of the most vital aspects of this project, and possibly the most complicated part of it. To explain and understand deeply about this section, we need to look into how the schema for different types of users is constructed. There are 2 possible approaches that our team thought of when designing this, one is creating 3 different unrelated schemas of each type of user, or somehow creating 1 schema that all 3 different types can inherit from. For the first solution, it is not only not efficient since we will need to query 3 times in order to be able to login, but it is not scalable as well. Therefore, we went with the second solution, and this is achieved by utilizing a feature in mongoose: discriminator. Simply put, all 3 types of schemas will inherit from the user schema, which contains common fields that 3 schemas have: username, password and profileImage, this will ensure username be unique across all fields and we only need to query the main user schema. As for registering, we simply need to create a new record for their respective user types. However, login is more complicated as it involves other sections of the project. In order to persist user login throughout all pages, we use passport and session, with these 2 modules, when user login, their information such as username, profile image will be easily accessed by the pages. In addition, since the password is encrypted by bcrypt, when login, the password will also need to be decrypted first then stored in the current session.

b. Add to Cart

Instead of immediately saving the user's current cart when they add a new product to their cart, we instead store the cart along with its products inside the current session. The order will only be created when the user submit the form (Buy!). We designed this way so that it reduces the number of times the system will have to query. If instead we save a new order everytime user adds to their cart a product, the system will have to perform a query and update every time the cart is updated. .Although the difference won't be noticeable in a small system such as ours, but in a bigger system where there are thousands of records in the database, we believe it is more efficient to simply store the user's cart inside the current session.

c. Product Page (Varies for different users)

To implement the production section accurately, our team decided to pass different data storages for different types of users, therefore the product in different user types will have distinct value . For instance, customers will get the entire product data, which means they can access all data of the database, while the shipper role only can access the products that have been passed to their specific hub.To conducting that idea a route was made for different types of user. After that the backend will query specific data of products passing to that user type. Hence, users can access the product page by clicking a link or a button to view the detailed information of the product. The link will call method "GET" which will announce the backend to conduct the command. Each link will be product id so that the system will correctly seek for the data then retrieve the information to display back to the monitor . Moreover, purchasing also requires the custom role to visit the product to take action. There is a button add cart in each product page. Once it gets clicked a method "POST"- which receives a specific id of the product, will take action sending the request to the database. The system will find the product based on the id then add that to the temporary list in session - which has been explained above. Both shipper and vendor roles will have the same format for the product page since they only can view the information of the product. Shippers can only view the filtered product based on the hubs and the vendor can only see the product that they have created.

d. Vendor page

When a vendor login to the website, the system will retrieve a list of products that were sold by that vendor from the backend. In order to access the user's id so that the system can differentiate the source of the product, we utilize passport and session, which was explained above. The system will use that id to query by the vendor field that was defined in the Product schema. Moreover, vendors also view the product detail of the product they made by clicking a link in the product card. This link will send the id of the specific product to the backend system to query the product data in the main database. Then the retrieved data will be displayed in the vendor's monitor.

e. Customer page

To show all products to the user, we simply need to query all products from the product collection and display those products onto the front end. In addition, since customers

can also click on a product and view its content such as vendor, price and name, we store the product's id in the href of the link (a tag). When clicked, it will send a get request to the backend which will query for the product with that unique id and display it on the monitor. Additionally, customers can visit the product detail page to add products to cart. After clicking to the button "add to cart" the system will store the data of the product in a list created by session. This list will temporarily store the data of the product as long as the user stays logged in. By pressing the button "BUY" in the shopping cart page as mentioned above, an order will be automatically generated, keeping the data of both user and product which will be then delivered to random hubs.

f. Changing Profile Images

In order to change user's profile images, we use multer - a module. Unlike huge and commercial systems, which will store user's images in their own databases, we only store the path to the images in the database, as for the actual images, we store it in a directory in our codebase. When users change their profile, the system will upload a new image with a unique name into the uploads directory, at the same time, the backend will query and update the user's profile image.

g. Search and Filter

Search and filter are two main tools for querying and retrieving specific data. These user-friendly tools have been purposefully designed to facilitate seamless data exploration and extraction. To implement these tools, queries were made in the backend with the method "GET"receiving "/search-result" for search and method "POST" receiving "/filter-amount"for the filter. For the search tool, the frontend will receive the input from the search bar, then send the result to the backend. The system will find the document in the "Product" collection where the "name" matches a regular expression. For case-sensitive such uppercase letters or lowercase letters are also detected to facilitate the use of search. In filter tools, the system will collect the data of the highest product cost to lowest product cost for narrowing down the product cost range. Thus users will be more convenient in finding suitable products with their affordable ability.

h. Shipper Page

In the domain of logistics and shipping operations, shippers page is allocated increased effort and resources toward the task of querying and retrieving data. Because of displaying only the orders of current hubs, the system will initially find the hubs that the shipper belongs to, then it will try to populate the data of orders in that hub. Each order contains the information of the products and the customer who made the order. After that these orders will undergo a filter to find the order that has the status equal "active". The result data will then transfer to display at the user monitor. For changing the order status, the shipper can normally click the button "Delivered" or "Canceled" in each order. Each button will have a different value, the system will query that value in the body and the order by the id it receives, then update the order with new status. After that the user will be redirected to the main page which will not display the product has status not active.

• Frontend Implementation and Decisions

a. Partials Components

Since the pages are designed based on a fixed theme, we have implemented partial works for ejs and css styling, in order to save work and also maintain theme consistency between pages. This section contains headers, footers and profile pages. While the footer and the profile pages stay the same, there are minor differences in and attributes for different user types in the headers, so some modifications are applied accordingly. Detailed implementation for the header is justified later below, in the users' interface design.

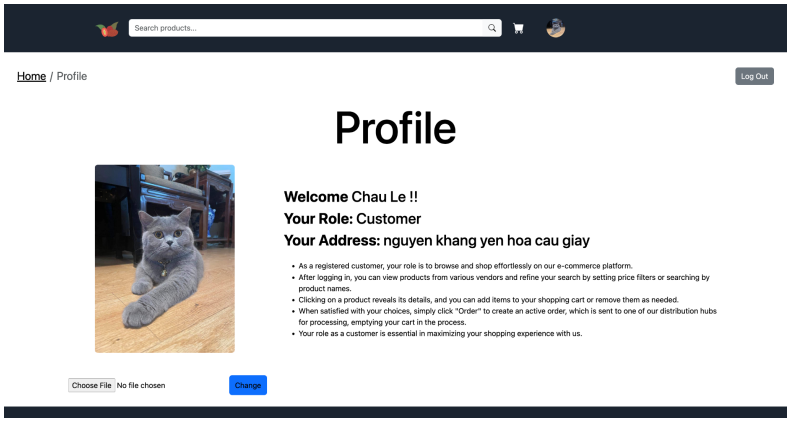


Figure 1: The design of the profile page for every users

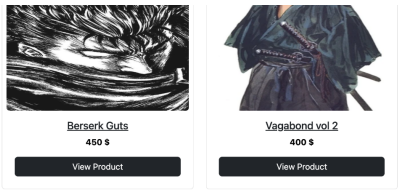
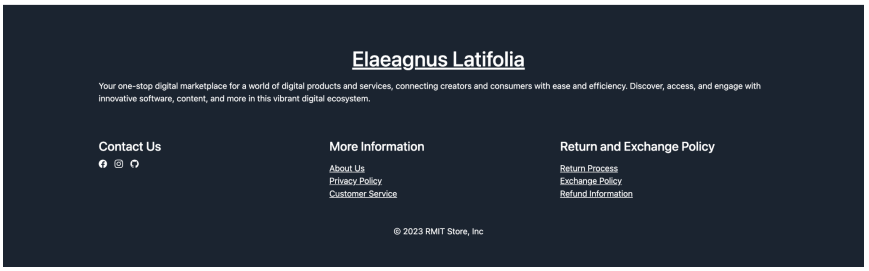


Figure 2: Footer design



b. Customer's Account

After logging in as a customer, the user will be redirected to the customer's home page *(Figure 3)* where the products are displayed. The footer remained the same for all pages to ensure the consistency while the header (aka. The navigation bar) is different for each user's interface, for the customer's page there is a search bar, a logo (which redirects to the home page) and a clickable profile picture which will redirect them to the customer's profile page. Aside from the product cards and the navigation bar, the home page also has a filter to sort products based on a price range which is situated on the top of the page.

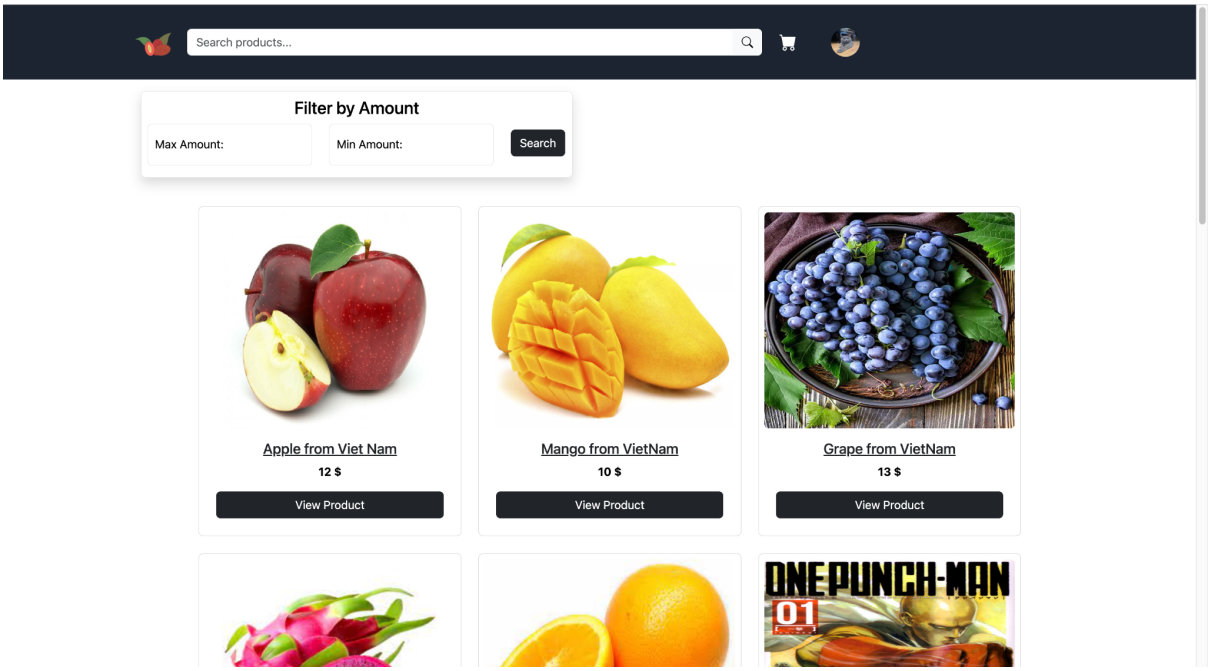


Figure 3: Customer's Home Page Interface

When the customer clicks on the "View Product" button on the home page they will move on to the product's details page *(Figure 4)*. On the top of the details page, there is a breadcrumb and the product's details such as the product's image, name, price, description and the product's seller and an add to cart button are kept in a container. All these elements are responsive in different screen sizes.

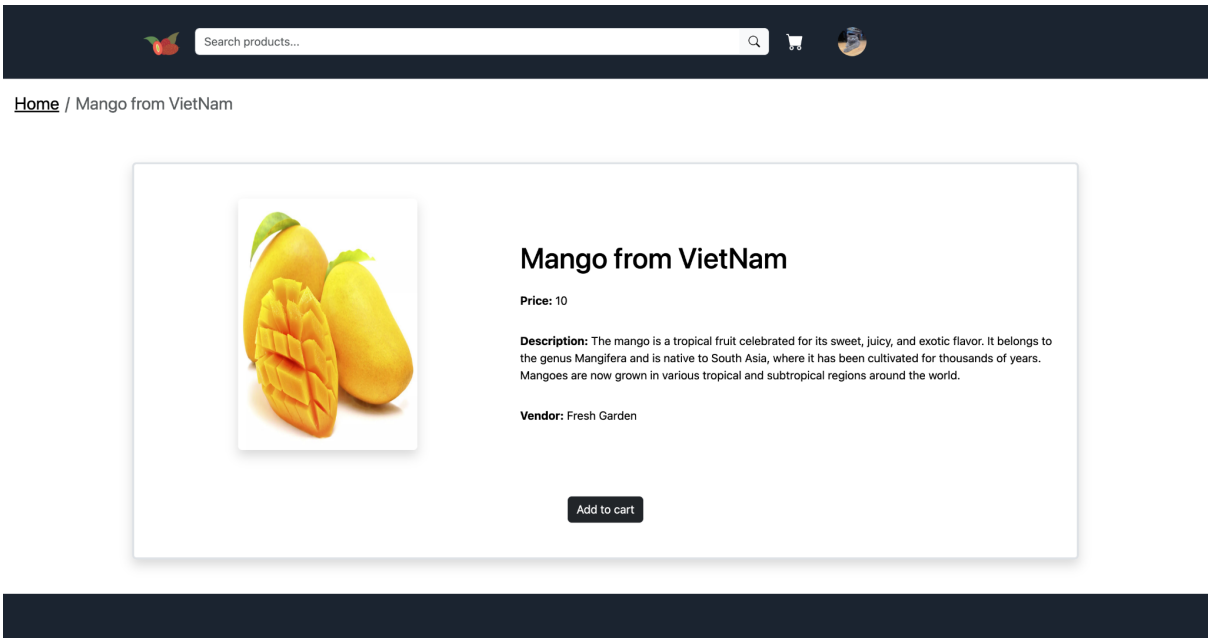


Figure 4: Customer's product detail page

After using the search/filter function of our website, the customer will be directed to the search query result page where the desired product will be displayed. This page will look like the home page but with only one card or two cards which are the result of the search/filter and an added breadcrumb on top of the page.

For the customer's account there will be a shopping cart page *(Figure 5)*, users can navigate there by clicking on the shopping cart icon and their shopping list will be displayed. If non item were selected it will be announced in the middle of the page inside a container. However, if there are any selected items, it will be displayed with the name, price of the product, a remove item button to remove the item from the list, the total amount of all the products inside the cart and the buy button to confirm the order.

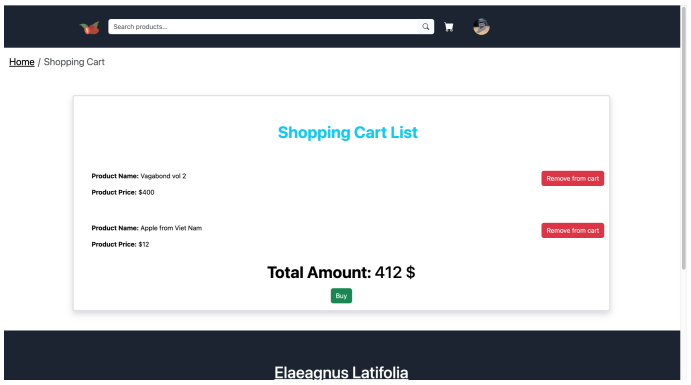


Figure 5: Customer's shopping cart

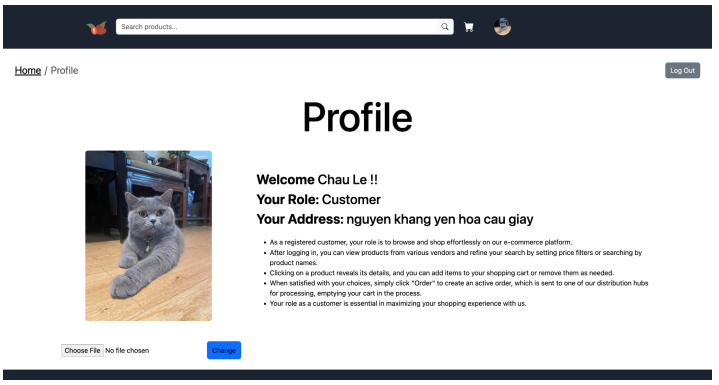


Figure 6: Customer's profile page

When the users click on their profile picture on the navigation bar it will redirect them to the user's profile page *(Figure 6)* where their name, role and address and their role description will be displayed. On this page, users will also be able to upload and change their profile picture. Aside from that, there is also a breadcrumb on top of the page just like some pages in the customer's account.

c. Vendor's Account

Once the seller logs into the website, they will be presented with a page that shows their current products for sale. If the seller hasn't sold any products yet, the "View My Products" section will indicate that there are no products listed as shown in *Figure 7*. However, if the seller has already sold products, those products will be displayed on the page.

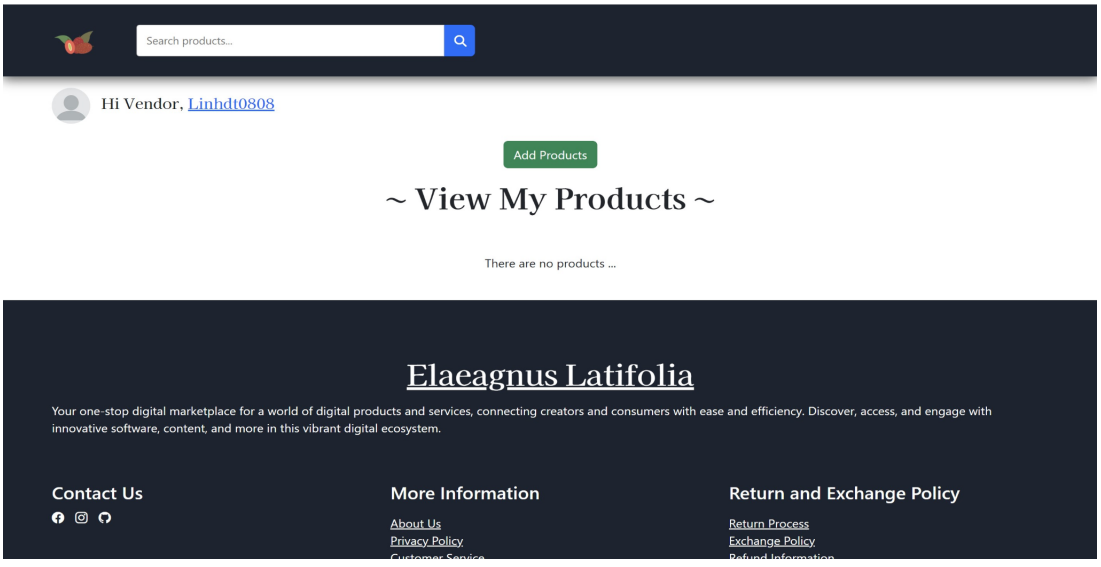


Figure 7: Vendor's Home Page

To sell a new item, the seller has to click on the "Add Products" button. This action will redirect them to a different page where they can create their product listing like *Figure 8*. On this page, the seller will need to provide the name, price, description, and an image (optional) of the product they want to sell. After entering the required information, the seller can confirm the creation of the product.

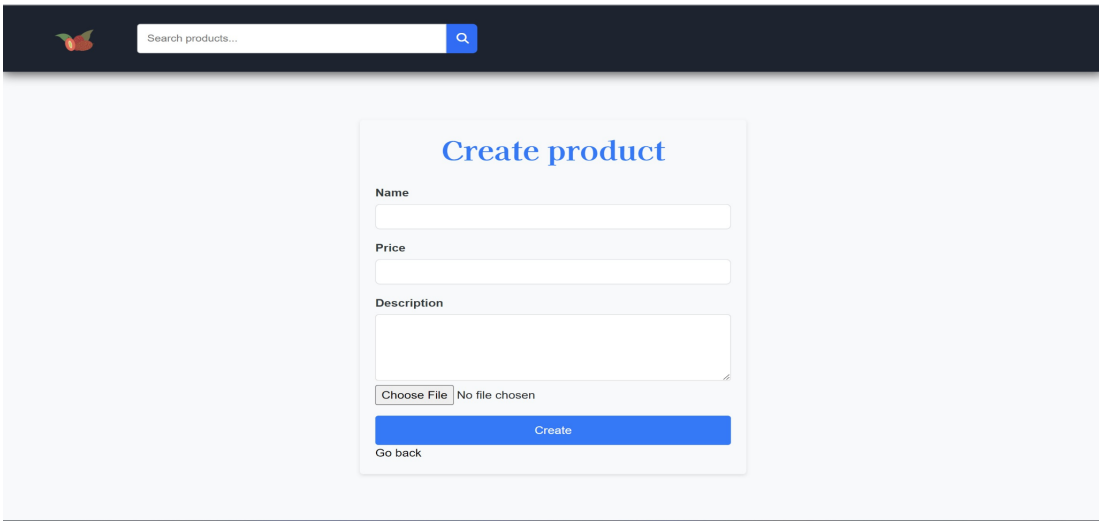


Figure 8: Vendor's creating product page

Once the product is successfully created, it will be immediately displayed on the screen along with all the details provided earlier. However, if any information is entered incorrectly or missing, the seller will receive an error message indicating the failure of product creation and highlighting the specific error. For instance, the name should be between 10 and 20 characters, the price must be a positive number, the description should not exceed 500 characters, and the product image is optional. If any of these requirements are not met, the seller will not be able to complete the product creation process.

After creating a product, it will appear on the vendor's main page in the "View My Products" section. Additionally, the product will be visible on customer pages, allowing potential buyers to view and purchase it (*Figure 9 and Figure 10*).

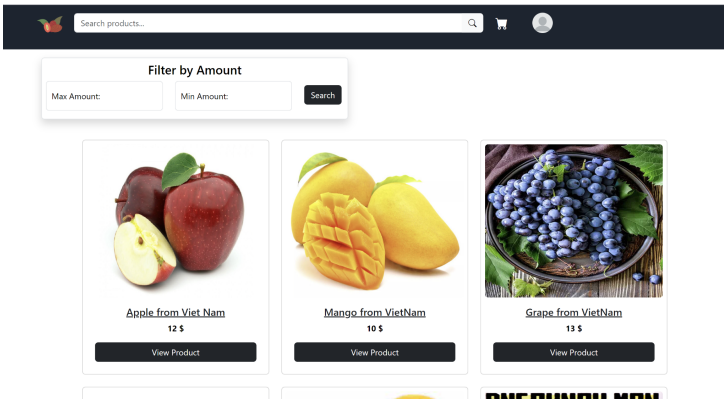


Figure 9: After creating product successful and being displayed in customer's page

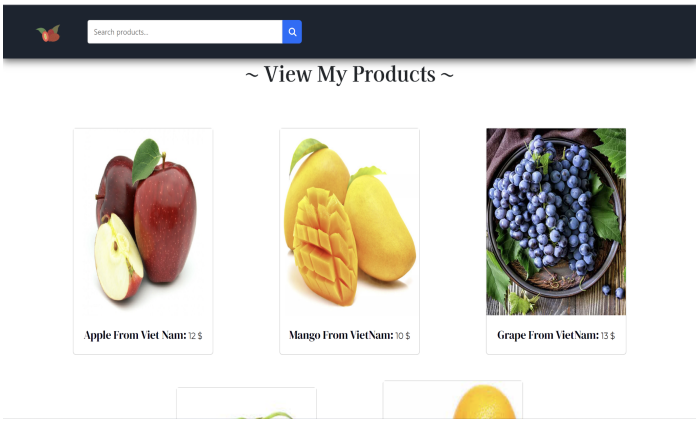


Figure 10: After creating product successful in vendor's page

d. Shipper's Account

Regarding the design for the shipper page, it is designed following the skeleton of the pages for the other 2 users type vendors and customers. While the footer stayed the same, the header for shipper page has changed, since there is no need for a search bar for active orders of shipper interface, so we have removed the middle search bar and replaced it with a heading indicating the shipper page.

In terms of the general interface for the shipper page, we implemented 3 pages namely index, order and product. All of the pages are styled accordingly by bootstrap and separated customized css files.

In the index page, all the currently active order in the hub that the shipper is assigned to is displayed on the main part of the screen as a grid table. Each order is put inside a bootstrap card, with title and text body. A fixed view detail button is put at the end of the card, to prevent orders with different detail length messing up the layout of the card.

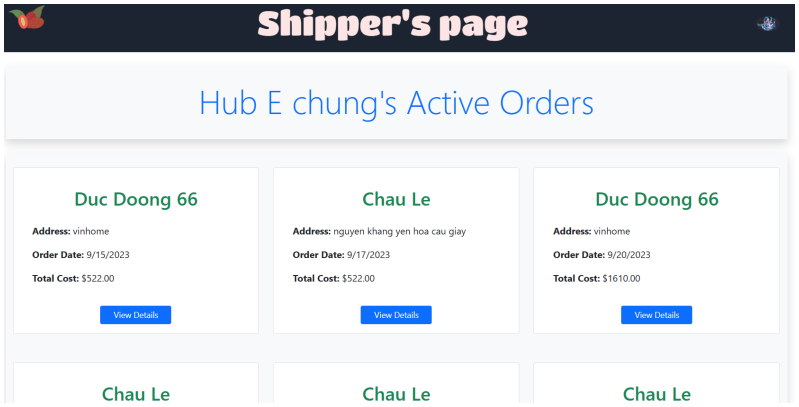


Figure 11: Main interface for shipper page

Next, the order page is where we redirect the shipper after clicking on view details of a specific order, so the interface only consists of a container with every detail of the order inside. Additionally, the list of products in the order is put inside a separated container, with fixed height so that it can be placed next to the order card. This is done for better usability, since the list of products sometimes might be overlength, and that would be a difficulty scrolling around buttons and order's items.

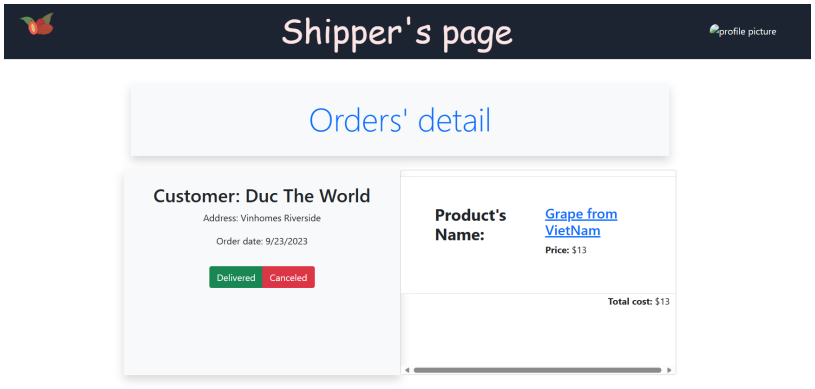


Figure 12: Order's detail page

Inside the list of products are links to the products' detail page, which consist of an image of the product and its details. These elements are put in different containers and are put next to each other for better visibility and appealing looks. For a small screen, the elements stacked on each other for better visibility.

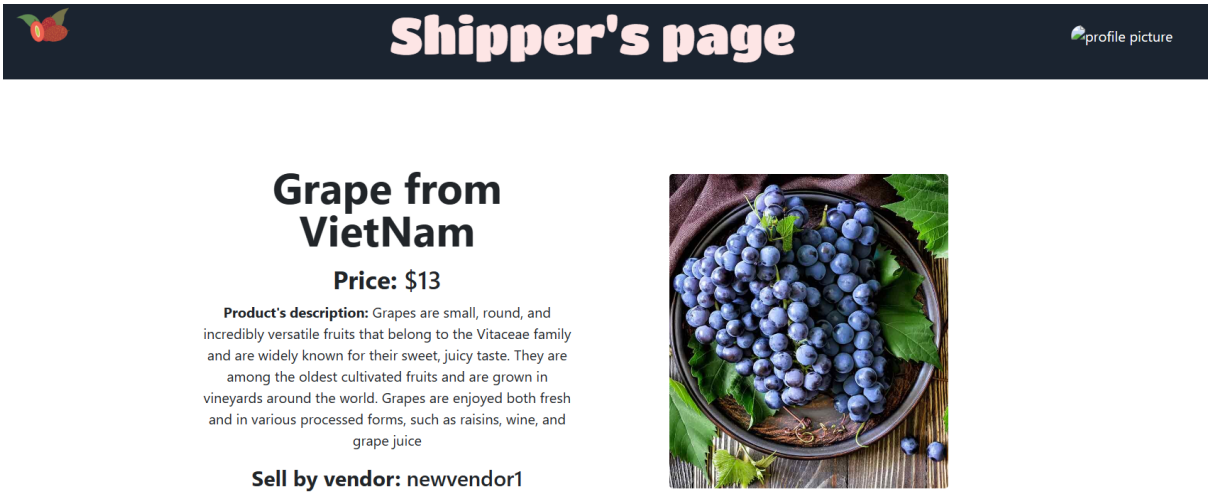


Figure 13: Product's detail page

By using a bootstrap grid system with rows and columns, we have saved a lot of time and effort in making the pages responsive, as we only need to specify the width of the column and the screen type, the element's container will automatically change according to the screen's size.

- **List of Features**

	Login & Register	Customer Page	Vendor Page	Shipper Page
List of Features	• Authenticate the input(username & password) • Generate new account for customer, shipper and vendor	• Display list of products • View product detail • Search bar • Filter amount • Add to cart • Create the order • Calculate total amount in shopping cart • Change profile picture • Responsive interface • Remove product from cart	• Display their own product • Create new product • View product detail • Change profile picture • Responsive interface	• View order list • Change order's status • View product detail • Change profile picture • Responsive interface

- ## Extra Features

In real life applications, when users add a new product to their cart, they can also remove that product from the cart, therefore, we implemented this functionality so that users can have an easier time buying. To create this feature, we simply need to remove the product from the user's current session, thus removing it from the cart.

- ## Screenshots for validator.w3.org and jigsaw.w3.org

1. **validator.w3.org** (We only uses ejs there isn't any html file for validation)
2. **jigsaw.w3.org** (Please look at **Appendix Section**)

By using the css validator, we have seen that most of our css files are validated, however 2 out of 6 files have errors occurred such as the vendor.css and customer.css files. This validator has pointed out some small mistakes that we have been making without noticing such as negative values for padding and margins. Thanks to that, we will take that into account for our future projects.

IV. Conclusion

In retrospect, this eCommerce project has been a significant learning experience, mirroring elements of Lazada's model. Our development journey has resulted in a robust platform serving customers, vendors, and shippers, with a strong emphasis on security via password hashing and data storage using MongoDB Atlas. The incorporation of responsive design ensures that the platform is accessible across various devices, promoting a seamless user experience.

However, as we take stock of our achievements, we also recognize the project's potential for growth and enhancement. Scalability emerges as a key concern, necessitating careful planning as the platform expands. Future endeavors include exploring opportunities to integrate analytics for better insights, implementing recommendation mechanisms for users, and continually refining the user experience. In this dynamic landscape, our commitment to staying abreast of evolving technology trends remains pivotal, ensuring our competitiveness and relevance in the ever-evolving eCommerce sphere. As we conclude this phase of the project, we embark on a new chapter, armed with the lessons learned and the excitement of further shaping this platform into a thriving hub for digital commerce, all the while remaining inspired by the ever-changing landscape and boundless possibilities of eCommerce.

V. Peer Evaluation

All Team Members	Role and Task Given	Individual Contribution (%)
1. Dong Manh Duc	Backend Developer	20%
2. Tran Tuan Trung	Frontend Developer	20%

3. Nguyen Ba Duc Manh	Backend and Frontend Developer	20%
4. Le Nguyen My Chau	Frontend Developer	20%
5. Do Thuy Linh	Frontend Developer	20%

VI. Appendix

- Pedro Tech (2020, December 24). 'Uploading Images With Multer' [video], *PedroTech*, YouTube website, accessed 29 August 2023. https://www.youtube.com/watch?v=wIOpe8S2Mk8&t=11s
- Web Dev Simplified (2019, July 13). 'Node.js Passport Login System Tutorial' [video], *Web Dev Simplified*, YouTube website, accessed 29 August 2023. https://www.youtube.com/watch?v=-RCnNyD0L-s

- **jigsaw.w3.org (SCREENSHOTS)**

(Please scroll down for the screenshots)

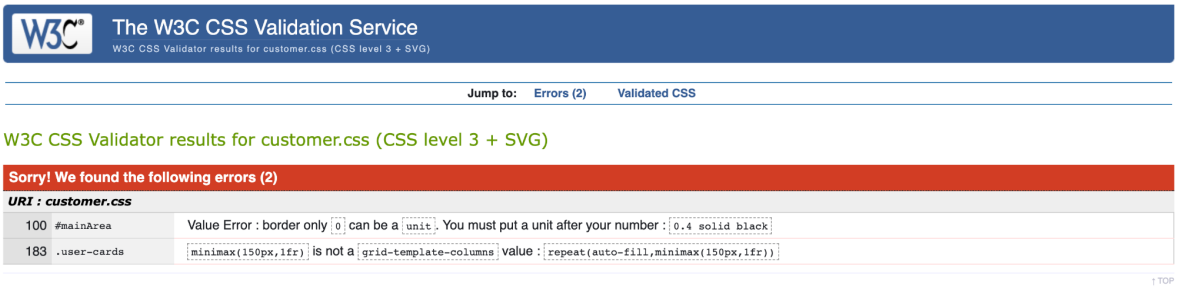


Figure 14: CSS validator result for customer.css file in public folder

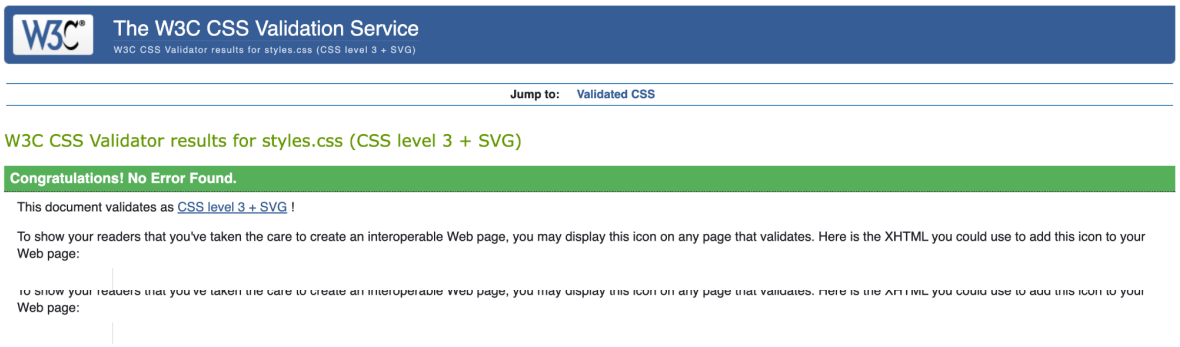


Figure 15: CSS validator result for styles.css file in public folder

Figure 16: CSS validator result for vendor-addproduct.css file in public folder

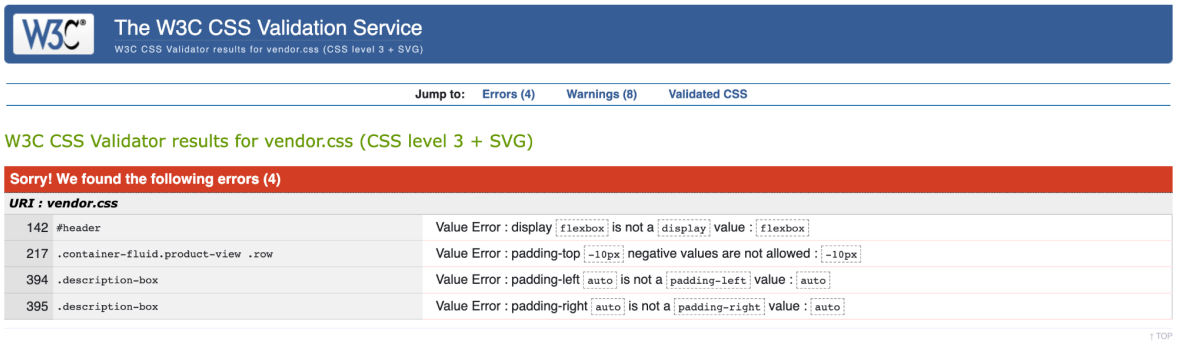


Figure 17: CSS validator result for vendor.css file in public folder

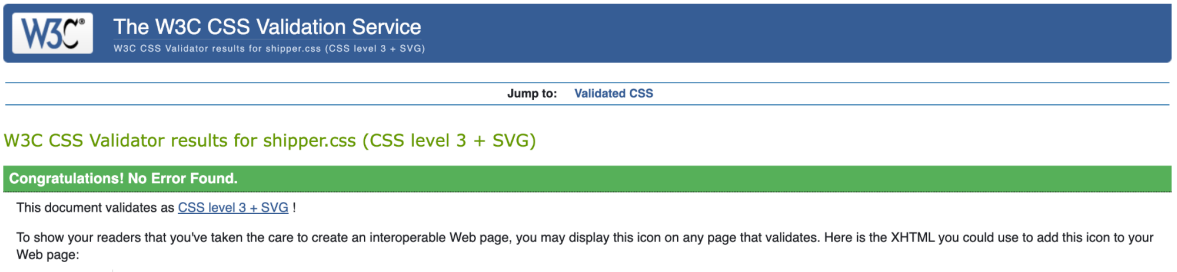


Figure 18: CSS validator result for shipper.css file in public folder

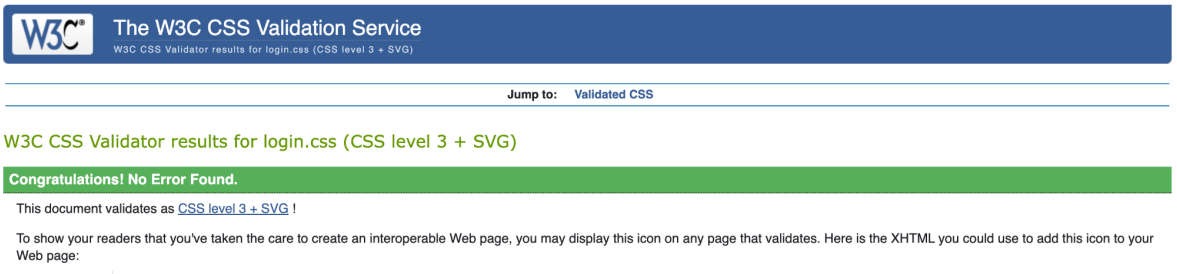


Figure 19: CSS validator result for login.css file in public folder