



**CSE471 : System Analysis and Design
Project Report**

Project Title : Wearwise - An Online Clothing Rental Service Website

Group No : 03, CSE471 Lab Section : 09, Summer 2023	
ID	Name
21301610	Shihab Muhtasim
21301274	Nusaiba Alam
20301326	Sartaj Emon Prattoy

Table of Contents

Sectio n No	Content	Page No
1	Introduction	3
2	Functional Requirements	5
3	User Manual	9
4	Frontend Development	31
5	Backend Development	62
6	Technology (Framework, Languages)	95
7	Github Repo Link	97
8	Individual Contribution	97

Introduction

Introduction:

Welcome to the project report of "Wearwise - An Online Clothing Rental Service Website." The genesis of this project traces back to the semifinals of the esteemed Hult Prize competition, where our innovative concept caught attention and stirred excitement. Our journey through this project has been marked by consistent progress and dedication, with each week bringing us closer to creating a seamless and user-friendly platform.

Project Overview:

The Wearwise project is a testament to the synergy of fashion, technology, and convenience. We've divided our project into modules, and each week, we focused on developing a specific module to ensure comprehensive coverage of all aspects of the platform.

Admin Panel

In the first week, our attention was on setting up the administrative backbone of Wearwise. This module encompasses the core functionalities required for smooth platform management. We implemented features such as login and registration, session management, and the ability to add, edit, and delete products. Moreover, the management of categories and apparel was integrated, laying the foundation for efficient cataloging.

Home/Shop

With the administrative framework in place, we turned our focus to the user-facing aspect of Wearwise. The second week revolved around creating an inviting and intuitive homepage for users to browse through our selection. We implemented features for displaying products, including their size, description, price, and vendor name. The ability to sort products by category and apparel, along with a search function, further enhances the shopping experience. We also worked on refining the cart management, allowing users to update and delete items.

User Dashboard

Continuing our march towards a complete platform, the third week was dedicated to developing the user dashboard. We focused on features that enable users to have a personalized experience. This includes viewing and editing their profiles, as well as checking their order history. The integration of an online payment gateway using cards enhances the convenience of transactions. While we worked on order confirmation and customer reviews, some aspects, such as viewing return dates of ordered items, remain works in progress.

Vendor Management

In the final week, our attention shifted to empowering vendors to participate in the Wearwise ecosystem. We created the vendor registration process, ensuring vendor approval by the admin. Vendors can log in, post products for approval, and manage their pending and approved products. We also worked on a dashboard tailored for vendors, providing essential insights into their orders.

Conclusion:

The Wearwise project encapsulates our dedication to creating a revolutionary online clothing rental service website. By systematically developing each module over the course of four weeks, we have laid the groundwork for a platform that seamlessly connects users and vendors in the world of fashion. This project report will delve into the details of each module, showcasing the progress made and the features available, with a keen eye on enhancing user experience and convenience.

Functional Requirements

Module 1: Admin Panel

1. Login & Registration:

- Admin should be able to log in using valid credentials.
- New admins should be able to register with unique details.
- The password provided by the admin should be protected by hashing.

2. Logout & Session Management:

- Admin should be able to log out securely.
- Session should expire after logout

3. Product Management:

- Admin should be able to add new products with details.
- Admin should be able to view, edit, and delete existing products.
- Admin should be able to approve or reject vendor projects

4. Category & Apparel Management:

- Admin should be able to add, edit, and delete categories and apparels.
- Products should be associated with the appropriate category and apparel.

5. Dashboard Features:

- Admin should have a dashboard showing important statistics from database
- The dashboard should have options to view order details, customer details, manage products, adding apparels and categories, logout, different account signup options in a sidebar and header.

6. Order & Customer Management:

- Admin should be able to view orders placed by customers.

- Admin should be able to delete the order when it is captured
- Admin should be able to view customers and delete anyone.

7. Vendor Management:

- Admin should be able to approve or reject vendor registration.
- Admin should be able to manage vendor accounts and products.

Module 2: Home / Shop

1. Homepage Products Display:

- Users should see a list of products on the homepage.
- Products should display relevant information like image, description, price, etc.

2. Product pagination & Search:

- Users should be able to paginate products on different pages.
- Users should be able to search for products using keywords.

3. Shopping Cart:

- Users should be able to add products to their shopping cart from the homepage or product details page.
- User can rent items according to their preferable quantity
- Users should be able to rent for minimum the days given by the admin or vendor and maximum to their choice.

4. Add review/reply/comment on product

- User can add review to each product
- They can also reply.
- Comments are visible below the product details

5. User Authentication & Registration:

- Users should be able to register and log in.
- Logged-in users should have access to additional features.

Module 3: User Dashboard

1. Profile Management:

- Users should be able to view and edit their profile information.

2. Online Payment Gateway:

- Users should be able to make online payments securely.

3. Order Placement & History:

- Users should be able to place orders and view their order history.
- Users should see estimated return dates for items they've ordered.
- Users should be able to download their ordered item details in pdf form.

4. Review:

- Users should be able to add reviews and ratings for products.

5. Delete or remove cart items:

- Users should be able to remove and delete items in the cart.

6. Checkout:

- Users should be able to proceed to checkout from the cart.
- They can checkout or continue shopping which will take them back to the home page.
- They can confirm their order either by cash on delivery or card payment

Module 4: Vendor Management

1. Vendor Registration & Login:

- Vendors should be able to register and await admin approval.
- The password provided by the vendor should be protected by hashing.
- Vendors should be able to login once approved by admin and logout when they want.
- After logout, none of the pages should be accessible.

2. Vendor Product Submission:

- Vendors should be able to submit products for approval.
- Vendors should be able to view, edit and delete their unapproved products.
- Vendors should be able to see and delete their approved products.

3. Order Management:

- Vendors should be able to view orders placed for their products and delete those.

4. Vendor dashboard:

- Vendor dashboard should have necessary information fetched from the database to give an overall look of the business.
- Should contain options for necessary actions in the sidebar and header.

User Manual

Admin panel:

There are 11 major features for the admin panel. Those are, Login & Registration, logout & session management on all pages, product management, category management, apparel management, admin sidebar, header, dashboard features design, view orders, view Customers.

1. Registration: To be an admin one needs to fill up a form where they have to put a reference key, use a unique username and match their passwords mandatorily to become an admin

The screenshot shows a registration form titled "SIGN UP AS ADMIN". The form is divided into two columns. The left column contains fields for "Name" (with a placeholder "Enter your name"), "Reference Code" (with a placeholder "Enter reference code given by other admins"), "Email address" (with a placeholder "Enter your email address"), and "Password" (with a placeholder "Enter your password"). The right column contains fields for "Phone number" (with a placeholder "Enter a valid phone number"), "Your reference Code" (with a placeholder "Enter a reference code that you'd like to use for other new admin signups"), "Username" (with a placeholder "Enter your username"), and "Confirm Password" (with a placeholder "Please enter the password again"). Below the form is a green "Sign Up as Admin" button. At the bottom of the page, there is a footer with the "WearWise" logo and contact information: "Contact", "Email: wearwise@gmail.com", "Phone: +8801928363", and "Address: Brac University, Dhaka, Bangladesh".

2. Login: An admin can login to see an admin dashboard with username and password if both the username and password matches in the database. Admin will see a dashboard with many necessary fields of information of the business to understand current status.

WEAR WISE Home Login Signup Admin Login Admin Signup Vendor Login Vendor Signup

ADMIN LOGIN

Username

Password

Log in

WEAR WISE

About Us

We are an online clothing rental brand where you can find a solution to all your clothing needs at just the right price for the right occasion.

Contact Us

Phone: +8801930438736
Email: wearwise@gmail.com
Address: Brac University, Mohakhali, Dhaka, Bangladesh

Wearwise

panda Administration Member

Navigation

- Dashboard
- Manage Products
- Manage Category
- Manage Apparel
- View Customers
- View Orders
- Approve vendors
- Certified vendors
- Admin's Profile

HELLO, panda
WELCOME TO WEAR WISE
Get started managing your business

5 Types
Total Product Types

4 Orders
Total orders

840 TK
Total sale

5 Customers
Total customers

Wear wise sales
720 TK
Original wearwise sales

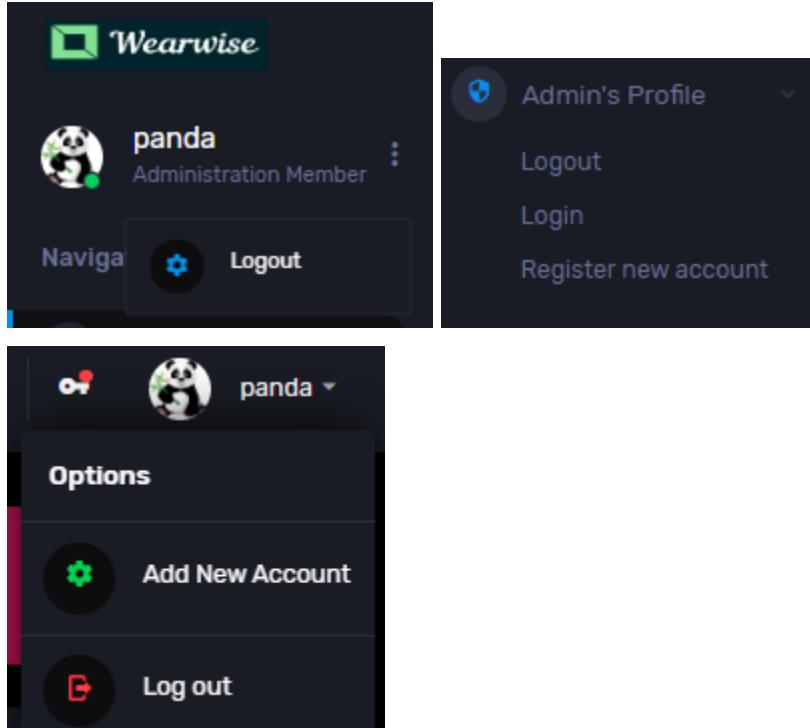
Vendor Sales
120 TK
Vendor partner sales

Vendor sale earnings
12 TK
10%commission on each sale

To Do List

enter task... **Add**

- Logout: Admin can log out using the logout button that can be found in the admin dashboard.



4. Add product: An admin can add products to the website under name Wear wise brand by going through Manage products-> Add products-> fill up the form and launch the product.

Insert Products

Product Name:

Product Category:

Product Apparel:

Product Price:

Product Discount Price:

Days:

Product Quantity:

Product Image:

Product Description:

Launch product

5. View product: This button takes admin to see all the products with option to edit any details or delete those.

ID	Name	Image	Price	Discount Price	Quantity	Day count on price	Vendor	Edit	Delete
1	Blue shirt		100	100	20	5	Wear Wise	<button>Edit</button>	<button>Delete</button>
2	Black panjabi		120	100	30	12	Wear Wise	<button>Edit</button>	<button>Delete</button>
3	Red gown		500	400	5	5	Wear Wise	<button>Edit</button>	<button>Delete</button>
4	New		122	100	123	5	Wear Wise	<button>Edit</button>	<button>Delete</button>

6. Approve vendor products: Admin can see all product post requests from vendors and he can delete it or approve it.

ID	Name	Image	Price	Discount Price	Quantity	Day count on price	Vendor	Approve	Delete
	Antheaa		500	400	40	5	Mynta	<button>Approve</button>	<button>Delete</button>

7. Approve vendors: Admin can see all requests for vendors and he can approve or delete those by selecting these buttons.

ID	Name	Phone	Username	Business Name	Licence	Email	Approve	Reject
6	Shihab Muhtasim	01993043534	shihab	TEST1	1234546	shihab.muhtasim@g.bracu.ac.bd	<button>Approve</button>	<button>Delete</button>
7	Sole harvest	01993043534	sole	SOLE HARVEST	1234546	shihab.muhtasim@g.bracu.ac.bd	<button>Approve</button>	<button>Delete</button>

8. View certified vendors: Admin can see all approved vendors and delete any where if any product is up by this vendor also gets deleted.

ID	Name	Phone	Username	Buisness Name	Lisence	Email	Remove
6	Mr Hannan Chowdhuri	01283646373	fashionx	Fashion X	2343242334	hannan@gmail.com	<button>Delete</button>
7	Mr Hannan	345345	myntra	Myntra	342423423	hanu@gmail.com	<button>Delete</button>

9. Manage category: This button shows all categories available along with options to add, edit or delete any.

Category ID	Category Name	Edit	Delete
1	Men Casual Wear		<button>Delete</button>
2	Men Formal wear		<button>Delete</button>
3	Women Party wear		<button>Delete</button>

10. Manage apparel: This button shows all apparels available along with options to add, edit or delete any.

Category ID	Category Name	Edit	Delete
1	Suits		<button>Delete</button>
2	Shirt		<button>Delete</button>
3	Panjabi		<button>Delete</button>
4	Gown		<button>Delete</button>

11. View orders: Admin can view all orders and delete any by selecting the red delete button.

Order ID	Customer Name	phone	address	Product Name	Image	Price	Quantity	Start date	End date	Vendor	Delete
2	labuser	4545	fhfjf	Black panjabi		120	1	2023-08-06	2023-08-06		<button>Delete</button>
3	test1	01993043534	21301610	Blue shirt		100	1	2023-08-11	2023-08-11	Wear Wise	<button>Delete</button>
4	test1	01993043534	21301610	Black panjabi		120	1	2023-08-11	2023-08-11	Wear Wise	<button>Delete</button>

12. View customers: Admin can view all customers and delete any by selecting the red delete button.

ID	Name	Email	Phone	Address	Registered Date	Delete
1	Test5	shihab.muhtasim@g.bracu.ac.bd	01993043534	21301	2023-07-28 09:25:48	<button>Delete</button>
2	Nusaiba	n@g.bracu.ac.bd	45454	Chorer elaka	2023-07-28 15:35:59	<button>Delete</button>
3	user1	user1@gmail.com	123412	user	2023-07-28 17:42:53	<button>Delete</button>
4	labuser	a@gmail.com	4545	fhfjf	2023-07-30 05:51:37	<button>Delete</button>
5	test1	shihab.muhtasim@g.bracu.ac.bd	01993043534	21301610	2023-08-11 09:48:41	<button>Delete</button>

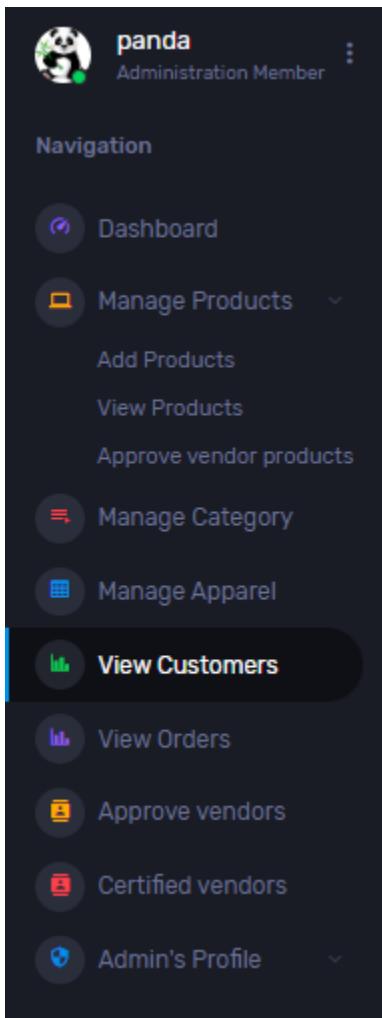
13. Header Options: Admin can find many options, such as sign up as user/vendor or login as user/vendor in the header. Also he can find options to add products from header as well.

The screenshot shows the Admin header with the following elements:

- + Add new in business
- User Login: Click here to log in as a user
- User Sign Up: New user? Sign up here
- Vendor Login: Click here to log in as a vendor
- Vendor Sign Up: New vendor? Sign up here
- Projects
- Add Product
- Add Apparel
- Add category

14.

15. Sidebar: Admin can find all options for the actions of admin in the sidebar.



Vendor Panel:

The Vendor Panel of Wearwise offers a range of features designed to empower vendors in managing their products, orders, and profiles effectively. This user manual outlines the functionalities available within the Vendor Panel, providing a comprehensive guide to navigating this platform.

1. Registration: Prospective vendors can initiate the registration process by completing a form. The form requires a distinct username, and passwords that match. This ensures secure access to the vendor panel upon approval.

SIGN UP AS VENDOR

Rent away your clothes with us

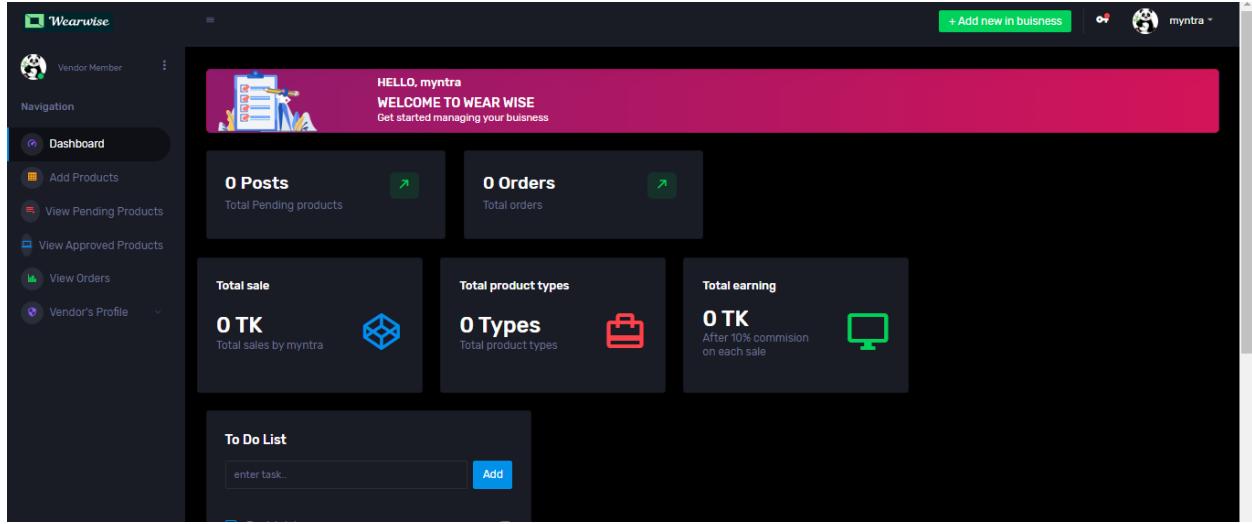
Name	Phone number
<input type="text"/>	<input type="text"/> Enter a valid phone number
Address	Business name
<input type="text"/> Enter Address	<input type="text"/> Enter display name
Business Licence	Email address
<input type="text"/> Enter business licence no	<input type="text"/>
Username	Password
<input type="text"/>	<input type="text"/>
Confirm Password	
<input type="text"/> Please enter the password again.	
Sign Up as Vendor	

2. Login: Upon successful registration and approval, vendors can log in using their designated username and password. This will grant access to the Vendor Dashboard, where crucial insights into product management and order tracking are provided.

VENDOR LOGIN

Username	Password
<input type="text"/>	<input type="password"/>
Log in	

3. Vendor Dashboard: The Vendor Dashboard serves as a hub for assessing the current status of one's business operations. It offers a comprehensive view of key performance metrics, pending product requests, approved product listings, and order tracking details.



4. Logout: vendors can securely log out of the Vendor Panel by selecting the "Logout" option available within the dashboard.
5. Add Product: Vendors looking to list new products on Wearwise can utilize the "Add Product" feature. This involves populating a form with product-specific details. The submitted product will be subject to admin approval before becoming visible on the platform.
6. View Pending Products: The "Pending Products" section provides a catalog of products awaiting administrative approval. Vendors possess the ability to review and manage these pending products, including the option to edit or delete entries.

ID	Name	Image	Price	Discount Price	Quantity	Day count on price	Vendor	Edit	Delete
3	Antheaa		500	400	40	5	Myntra	<button>Edit</button>	<button>Delete</button>

7. View Approved Products: The "Approved Products" section showcases products that have received admin approval. This overview allows vendors

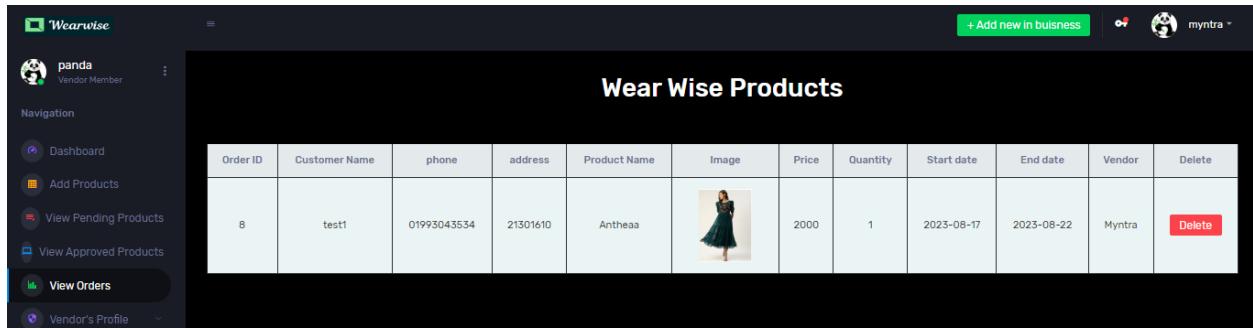
to access detailed product information and offers the option to remove listings as needed.



The screenshot shows a dark-themed user interface for managing products. On the left, a sidebar titled "Navigation" includes links for Dashboard, Add Products, View Pending Products, View Approved Products, and View Orders. The main area is titled "Wear Wise Products" and displays a single product row in a table:

ID	Name	Image	Price	Discount Price	Quantity	Day count on price	Vendor	Delete
9	Antheaa		500	400	40	5	Mynta	<button>Delete</button>

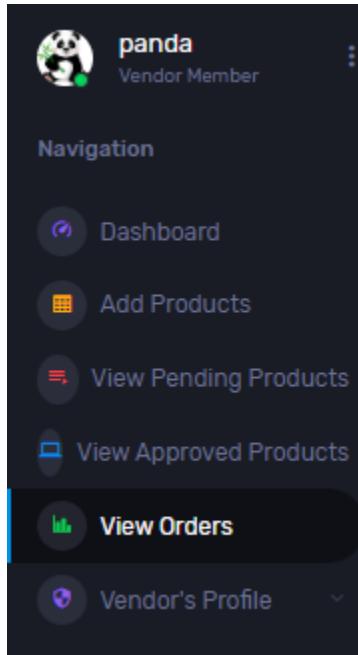
- View Orders: Vendors can monitor orders that pertain to their specific product offerings within the "View Orders" section. This display provides essential order information, including customer details and order status.



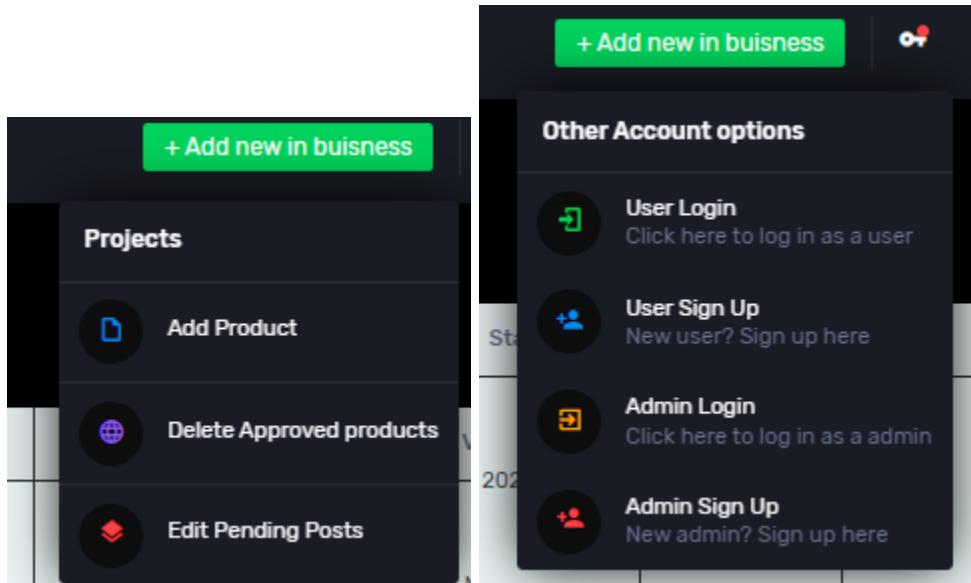
The screenshot shows a dark-themed user interface for managing orders. On the left, a sidebar titled "Navigation" includes links for Dashboard, Add Products, View Pending Products, View Approved Products, View Orders, and Vendor's Profile. The main area is titled "Wear Wise Products" and displays a single order row in a table:

Order ID	Customer Name	phone	address	Product Name	Image	Price	Quantity	Start date	End date	Vendor	Delete
8	test1	01993043534	21301610	Antheaa		2000	1	2023-08-17	2023-08-22	Mynta	<button>Delete</button>

- Sidebar Options: Access to all vendor-specific actions is available through the sidebar.



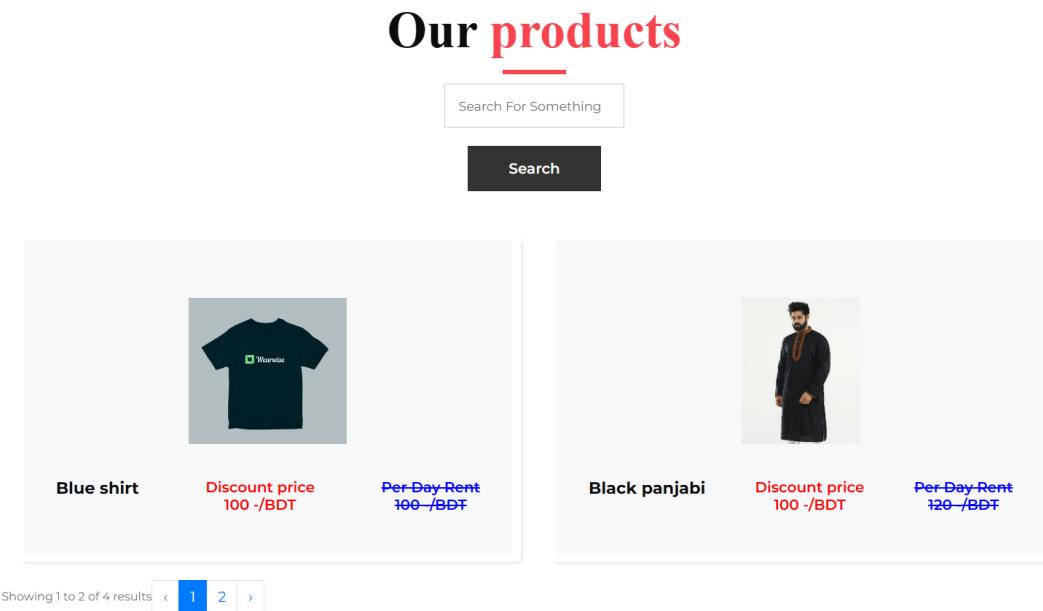
10. Header Options: Located at the top of the page, the header section provides access to user and vendor sign-up options. Vendors can also initiate the login process and manage their account from this point. The "Add Product" option is also conveniently accessible in the header.



Homepage:

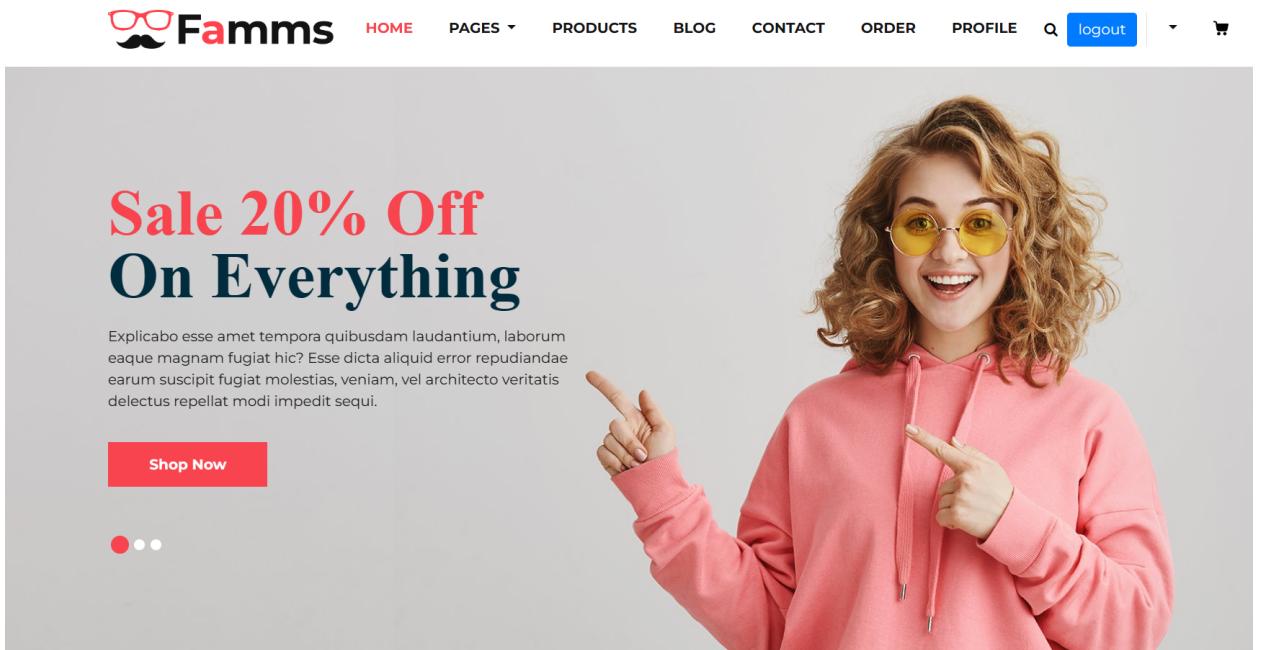
The Home / Shop module in Wearwise is designed to provide users with an effortless and engaging shopping experience. This user manual will guide you through the functionalities of the homepage and the products page, as well as how to navigate and utilize the Cart, Order, Profile, User Login, and User Registration features.

1. Homepage Products Display: Upon entering the Wearwise website, the homepage greets you with a curated display of products. This section showcases featured items, trends, and new arrivals, providing an overview of the product offerings.



2. Specific Homepage: For guest users, in the header section it will show the Signup or register button which will redirect them to the signup or register

panel. But registered users can access the logout button instead.



3. Header : Located at the top of the page, the header section provides access to user, admin and vendor sign-up options. There are specific on click buttons for Order, Profile, Cart.



4. Products details: When you navigate to the Products details page, you'll find a comprehensive catalog of items available for rent. Each product listing includes essential details such as description, price, rental duration in days,

and the name of the vendor offering the product.



Blue shirt

Discount price
100 -/BDT

Per day Rent
100 -/BDT

Product Details : kkj

Available Quantity : 20

Vendor : Wear Wise

Quantity:

Days:

Add To Cart

5. Search Products: Looking for something specific? Utilize the search bar to find products that match your criteria. You can even sort the search results

based on price, from low to high.

Our products

Search For Something

Search

6. Cart Management: The Cart is where you can review and manage the items you intend to rent. You can add products to the cart, update the quantities, or remove items that you no longer wish to rent.

Product Title	Product quantity	Day	Price	Image	Action
Black panjabi	1	12	1200		<button>Remove</button>
Black panjabi	1	12	1200		<button>Remove</button>

Total Price : 2400 ~/BDT

[Continue Shopping](#)

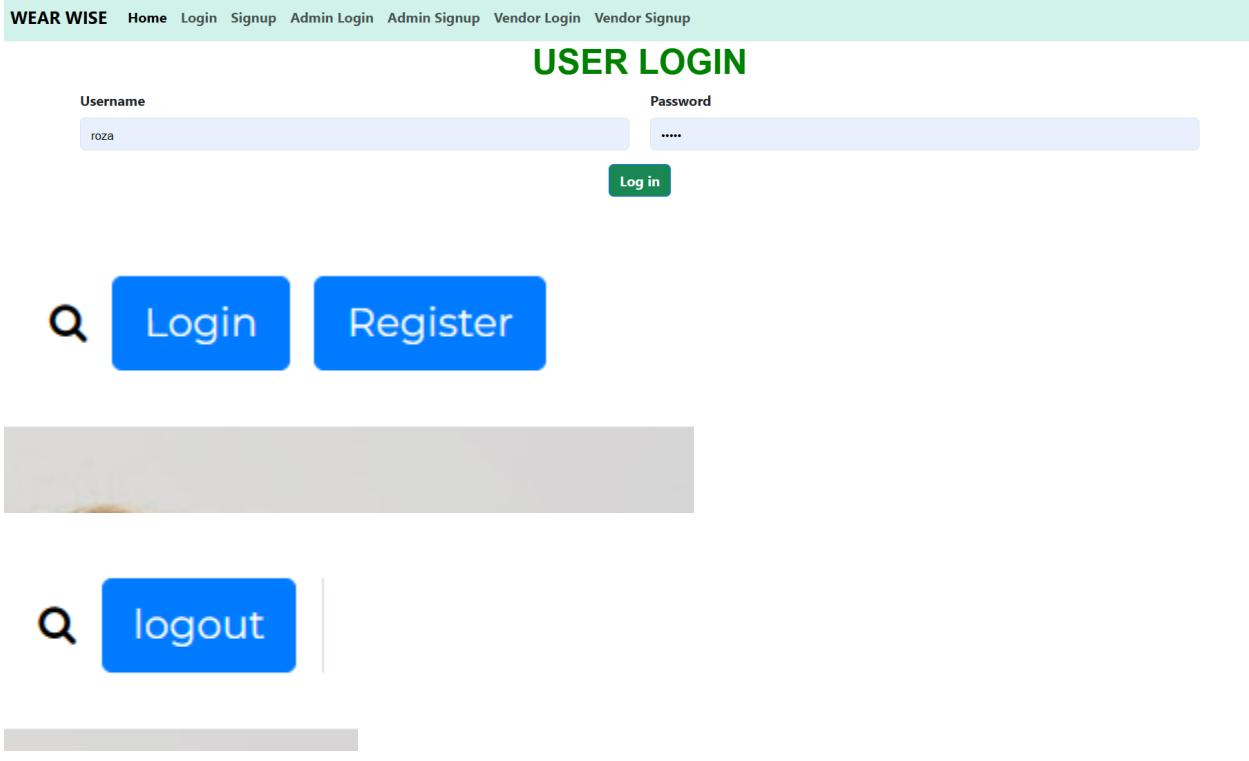
[Proceed to Order](#)

[Cash On delivery](#)

[Pay Using Card](#)

7. User Login and Logout: To access your personal account and dashboard, use the User Login feature. Input your credentials (email and password) to

securely log in. Logging out can be done from any page, ensuring the protection of your account.



8. User Registration: New to Wearwise? Registering is easy! Click on the User Registration option, fill in the required details, and create a unique username and password. Upon successful registration, you'll gain access to your

personalized User Dashboard.

WEAR WISE Home Login Signup Admin Login Admin Signup Vendor Login Vendor Signup

SIGN UP AS USER

Name **Phone number**
Enter a valid phone number

Address **Email address**
Enter Address

Username **Password**

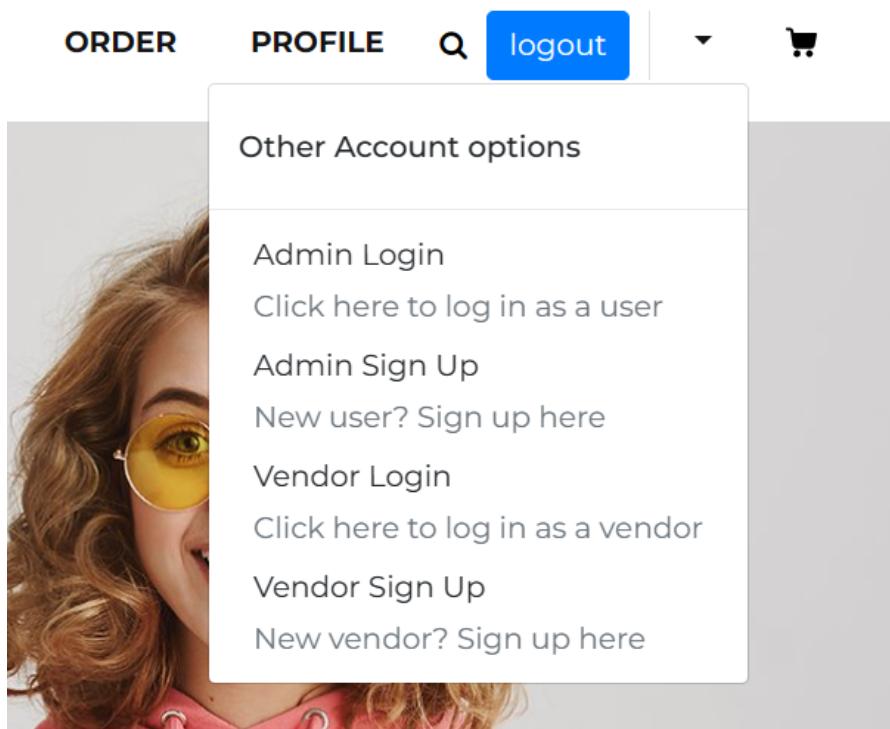
Confirm Password
Please enter the password again.

Sign Up as Admin

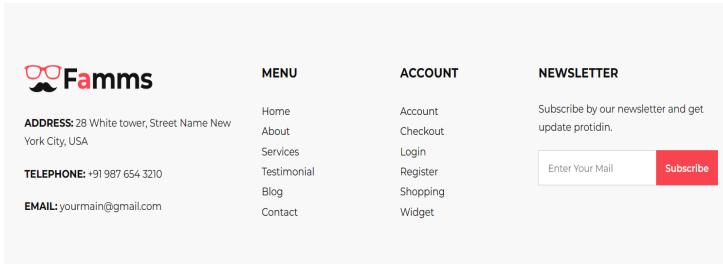
WearWise
Wear all wear wise

Contact
Email: wearwise@gmail.com
Phone: +8801928363
Address: Brac University, Dhaka, Bangladesh

- Session Management: Enjoy a seamless browsing experience with session management across all pages. This feature ensures that your login status and browsing history are maintained as you navigate through the website.



10. Footer : This code snippet defines a footer section for a website. It consists of a container with two columns, the left column displaying the logo and contact information, while the right column contains navigational menus, account links, and a newsletter subscription form. The footer provides a well-structured layout with essential website elements, including links to different pages, account-related actions, and a newsletter subscription option.



User Dashboard:

The User Dashboard of our ecommerce project offers a centralized and intuitive hub for customers to manage their shopping experience. Through this dashboard, users can effortlessly navigate and control various aspects of their account and orders. Key features include:

1. View Profile: Upon accessing your User Dashboard, you can view and manage your profile information. Make updates as needed to maintain accurate and current details.

A screenshot of the User Profile form. It contains fields for Username (roza), Email Address (nusaiba.alam@g.bracu.ac.bd), Phone Number (01794725491), Name (roza), and Address (asdfghjkl). At the bottom is a blue 'Edit Profile' button.

User Profile	
Username	Email Address
roza	nusaiba.alam@g.bracu.ac.bd
Phone Number	Name
01794725491	roza
Address	asdfghjkl
Edit Profile	

2. Online Payment Gateway: For secure and convenient transactions, Wearwise offers an online payment gateway that supports card payments. This allows you to easily make rental payments for selected products.

Pay Using Your Card - Total Amount 2400 Taka

A screenshot of the Payment Details form. It includes fields for Name on Card (empty), Card Number (empty), CVC (Ex. 311), Expiration Month (MM), Expiration Year (YYYY), and a large 'Pay Now' button at the bottom.

Payment Details		
Name on Card		
<input type="text"/>		
Card Number		
<input type="text"/>		
CVC	Expiration Month	Expiration Year
<input type="text"/> Ex. 311	<input type="text"/> MM	<input type="text"/> YYYY
Pay Now		

3. Order Confirmation: Receive order confirmations for successful transactions, ensuring that your rental choices are accurately recorded.

Order Details

Customer Name :
roza

Customer Email :
nusaiba.alam@g.bracu.ac.bd

Customer Phone :
01794725491

Customer Address :
asdfghjkl

Product Name :
Blue shirt

Product Price :
500

Product Quantity :
1

Payment Status :
cash on delivery

Return date :
2023-08-20

4. Edit Profile: Need to update your profile information? The Edit Profile feature allows you to modify your details as necessary.

Update Profile

<p>Username</p> <input type="text" value="Roza"/>	<p>Email Address</p> <input type="text" value="Nusaiba.Alam@g.Bracu.Ac.Bd"/>
<p>Phone Number</p> <input type="text" value="01794725491"/>	<p>Name</p> <input type="text" value="Roza"/>
<p>Address</p> <div style="border: 1px solid #ccc; padding: 5px; height: 60px; width: 100%;">"Asdfghjkl!"</div>	

5. View Orders: Stay informed about your rental history by accessing the View Orders section. This provides an overview of your previous rental transactions.

Product Title	Vendor Name	Quantity	Day	Price	Payment Status	Delivery Status	Date of order	Return Date	Image	Cancel Order	Print
Blue shirt	Wear Wise	1	5	500	cash on delivery	processing	2023-08-15	2023-08-20		<button>Cancel</button>	<button>Print PDF</button>
Black panjabi	Wear Wise	4	12	4800	cash on delivery	processing	2023-08-15	2023-08-27		<button>Cancel</button>	<button>Print PDF</button>

6. View Return Dates of Ordered Items (Incomplete): An upcoming feature will allow you to view the expected return dates of items you've rented, ensuring that you can plan accordingly.



2023-08-20

2023-08-27

7. Customer Review: Share your experiences and insights with other users by adding customer reviews. This feature encourages a community-driven approach to product evaluation.

Comments

Comment Something Here

Comment

All Comments

abcd

The product was good.

[Reply](#)

Sartaj

yes thats right

[Reply](#)

8. Cart Checkout: When you're ready to finalize your rental selections, navigate to the Cart and proceed to the checkout process. This will guide you through the steps to complete your rental.

[Proceed to Order](#)

[Cash On delivery](#)

[Pay Using Card](#)

If you encounter any challenges or have inquiries, our support team is available to assist you. Enjoy your shopping experience and maximize the benefits of our platform!

Frontend Development

Module 1: Admin Panel

1. Signup: (Login page similar)

```
<div class="content-wrapper">
    @if(session() ->has('message'))
        <div class="alert alert-info">
            {{session() ->get('message')}}
        </div>
    @endif
```

Form:

```
<h1 class="text-center fw-bold" style="color: green; font-family: 'Montserrat', sans-serif;">SIGN UP AS ADMIN</h1>
<form action="{{url('/')}}/adminsignup" method="post">

    @csrf
    <div class="row">
        <!-- Name box-->
        <div class="col-md-6 mb-3">
            <label for="Name" class="form-label fw-bold">Name</label>
            <input type="text" maxlength="30" class="form-control" id="Name" name="Name" aria-describedby="nameHelp" required="required">
        </div>
        <!--pohone num box-->
        <div class="col-md-6 mb-3">
            <label for="phone_number" class="form-label fw-bold">Phone number</label>
            <input type="tel" class="form-control" id="phone_number" name="phone_number" aria-describedby="phoneHelp" required="required">
            <div id="phoneHelp" maxlength="15" class="form-text">Enter a valid phone number</div>
        </div>
```

```

<!--Reference box-->
<div class="col-md-6 mb-3">
    <label for="ref_code" class="form-label fw-bold">Reference
Code</label>
    <input type="text" maxlength="5" class="form-control"
id="ref_code" name="ref_code" aria-describedby="ref_code_Help"
required="required">
        <div id="ref_code_Help" maxlength="5"
class="form-text">Enter reference code given by other admins</div>
    </div>
    <!--Reference Generate box-->
    <div class="col-md-6 mb-3">
        <label for="ref_code_gen" class="form-label fw-bold">Your
reference Code</label>
        <input type="text" maxlength="5" class="form-control"
id="ref_code_gen" name="ref_code_gen" aria-describedby="ref_code_Help"
required="required">
            <div id="ref_code_Help" maxlength="5"
class="form-text">Enter a reference code that you'd like to use for other
new admin signups</div>
    </div>
    <!--Email box-->
    <div class="col-md-6 mb-3">
        <label for="email" class="form-label fw-bold">Email
address</label>
        <input type="email" maxlength="50" class="form-control"
id="email" name="email" aria-describedby="emailHelp" required="required">
    </div>
    <!--username box-->
    <div class="col-md-6 mb-3">
        <label for="username" class="form-label
fw-bold">Username</label>
        <input type="text" maxlength="15" class="form-control"
id="username" name="username" aria-describedby="usernameHelp"
required="required">
    </div>
    <!--Password box-->
    <div class="col-md-6 mb-3">
        <label for="password" class="form-label
fw-bold">Password</label>

```

```

        <input type="password" maxlength="30" class="form-control"
id="password" name="password" required="required">
    </div>
    <!--Confirm pass box-->
    <div class="col-md-6 mb-3">
        <label for="cpassword" class="form-label fw-bold">Confirm
Password</label>
        <input type="password" class="form-control" id="cpassword"
name="cpassword" required="required">
        <div id="cpassword" maxlength="30"
class="form-text">Please enter the password again.</div>
    </div>
    </div>
    <!--Sign up button button-->
    <div class="text-center" >
        <button type="submit" class="btn btn-primary bg-success
fw-bold">Sign Up as Admin</button>
    </div>
</form>

```

This code represents a signup form for the admin panel. It captures details such as name, phone number, reference code, email, username, and password for admin registration. Validation checks are included, and a signup button is provided for admin registration. An alert is displayed if there's a session message. Login code is similar to this.

2. Add Product (Edit product similar):

```

<form action="{{url('/add_product')}}" method="post"
enctype="multipart/form-data">
    @csrf

```

```

        <!-- Product Name-->
        <div class="form-outline mb-2 w-30 m-auto">
            <label for="product_title" style="color: white;" class="form-label fw-bold">Product Name</label>
            <input type="text" class="form-control" id="product_title" name="product_title" aria-describedby="Product_name_help" placeholder="Enter product name" required="required" autocomplete="on">
        </div>

        <!-- Category select-->
        <div class="form-outline mb-2 w-30 m-auto">
            <label for="product_category" style="color: white;" class="form-label fw-bold">Product Category</label>
            <select name="product_category" class="form-select" aria-label="Default select example">
                <option selected>Pick a category</option>
                @foreach ($cat as $cat)
                    <option value="{{ $cat->id }}>{{ $cat->category_name }}</option>
                @endforeach
            </select>
        </div>
        <!-- Appreal select-->
        <div class="form-outline mb-2 w-30 m-auto">
            <label for="product_apparel" style="color: white;" class="form-label fw-bold">Product Apparel</label>
            <select name="product_apparel" class="form-select" aria-label="Default select example">
                <option selected>Pick an apparel</option>
                @foreach ($app as $app)
                    <option value="{{ $app->apparel_id }}>{{ $app->apparel_name }}</option>
                @endforeach
            </select>
        </div>

        <!-- product_price-->
        <div class="form-outline mb-2 w-30 m-auto">

```

```

        <label for="product_price" style="color: white;">
class="form-label fw-bold">Product Price </label>
        <input type="text" class="form-control" id="product_keywords"
name="product_price"
            aria-describedby="product_price_help" placeholder="Enter
product Price" autocomplete="on" required="required" >
</div>

        <!-- product_discount_price-->
<div class="form-outline mb-2 w-30 m-auto">
        <label for="product_price" style="color: white;">
class="form-label fw-bold">Product Discount Price</label>
        <input type="text" class="form-control" id="product_keywords"
name="product_discount_price"
            aria-describedby="product_price_help" placeholder="Enter
product discount Price" autocomplete="on" required="required" >
</div>

        <!-- product_price-->
<div class="form-outline mb-2 w-30 m-auto">
        <label for="product_price" style="color: white;">
class="form-label fw-bold">Days </label>
        <input type="text" class="form-control" id="product_keywords"
name="product_days"
            aria-describedby="product_price_help" placeholder="Enter
product Price" autocomplete="on" required="required" >
</div>

        <!-- product_quantity-->
<div class="form-outline mb-2 w-30 m-auto">
        <label for="product_quantity" style="color: white;">
class="form-label fw-bold">Product Quantity</label>
        <input type="text" class="form-control" id="product_quantity"
name="product_quantity" placeholder="Enter product Quantity"
autocomplete="on" required="required" >
</div>
        <!-- Image-->
<div class="form-outline mb-2 w-30 m-auto">
        <label for="product_image" style="color: white;">
class="form-label fw-bold">Product Image</label>

```

```

        <input type="file" class="form-control" id="product_image"
name="image" required="required">
    </div>

        <!-- Product Description-->
    <div class="form-outline mb-2 w-30 m-auto">
        <label for="product_description" style="color: white;">
            <!-- Product Description-->
        <input type="text" class="form-control" id="product_description" name="product_description" aria-describedby="product_description_help" placeholder="Enter product description" autocomplete="off" required="required">
    </div>

        <!-- Submit-->
    <div class="form-outline mb-4 w-50 m-auto">
        <input type="submit" class="form-control" id="product_insert" name="product_insert" class="btn btn-info" value="Launch product" >
    </div>

</div>
</form>

```

This HTML form lets admins add new products to the system. It includes fields for product details like name, category, apparel, price, discount price, days, quantity, image, and description. When the "Launch product" button is clicked, the data is sent to the add_product route.

3. View Product:

```

        <!-- TABLE-->
    <h1 class="text-center text-white">Wear Wise Products</h1>
    <div class="table-responsive">
        <table class="table table-bordered mt-5">
            <thead class="bg-secondary text-light text-center">
                <tr class="text-center">
                    <th>ID</th>
                    <th>Name</th>

```

```

<th>Image</th>
<th>Price</th>
<th>Discount Price</th>
<th>Quantity</th>
<th>Day count on price</th>
<th>Vendor</th>

<th>Edit</th>
<th>Delete</th>
</tr>

<tbody style="background-color: #eaf4f4; color: #333;">

@foreach($product_data as $product_data)
<tr class='text-center'>

    <td>{{ $product_data->product_id }}</td>
    <td>{{ $product_data->product_title }}</td>
    <td>
        
    </td>
    <td>{{ $product_data->price }}</td>
    <td>{{ $product_data->discounted_price }}</td>
    <td>{{ $product_data->quantity }}</td>
    <td>{{ $product_data->days }}</td>
    <td>{{ $product_data->vendor_name }}</td>

    <td><a onclick="return confirm('Confirm Edit?')" class="btn btn-success" href="{{ url('edit_product', $product_data->product_id) }}>Edit</a></td>
        <td><a onclick="return confirm('Confirm Delete?')" class="btn btn-danger" href="{{ url('delete_product', $product_data->product_id) }}>Delete</a></td>
    </tr>

```

```
@endforeach
```

This code segment displays a table of Wear Wise products using Bootstrap styling. It lists various attributes of the products like ID, name, image, price, discount price, quantity, day count on price, and vendor. For each product in the dataset, it generates a table row with corresponding data. The "Edit" and "Delete" buttons allow administrators to perform respective actions on the products.

4. Manage Category:(Manage Apparel similar)

```
<h2 class="text-center">Insert Catagories</h2>

<form action="{{url('add_catagory')}}" method="post" class="mb-2" >

    @csrf

    <div class="input-group w-90 mb-2" >
        <span class="input-group-text bg-info" id="basic-addon1"><i class="fa-solid fa-receipt"></i></span>
        <input type="text" class="form-control" name="catagory_name" placeholder="Insert Catagories" aria-label="Categories" aria-describedby="basic-addon1" style="color: white;">
    </div>
    <div class="input-group w-10 mb-2 m-auto" >
        <input type="submit" class="bg-info border-0 p-2 my-3" name="insert_catagories" value="Add Catagories" aria-label="Insert Categories" aria-describedby="basic-addon1" class="bg-info" >
    </div>
</form>
<!-- TABLE-->
<h1 class="text-center text-white">Wear Wise Categories</h1>
<table class="table table-bordered mt-5" >
    <thead class="bg-secondary text-light text-center" >
        <tr class="text-center" >
            <th>Category ID</th>
            <th>Category Name</th>
            <th>Edit</th>
        </tr>
    </thead>
```

```

<th>Delete</th>
</tr>

<tbody style="background-color: #eaf4f4; color: #333;">

@foreach ($data as $data)
<tr class='text-center'>

    <td>{ ${$data->}id }</td>
    <td>{ ${$data->}catagory_name }</td>

    <td><a href=' ' class='text-dark'><i class='fa-solid fa-pen-to-square'></i></a></td>
    <td><a onclick="return confirm('Confirm Delete?')" class="btn btn-danger" href="{{url('delete_catagory',$data->}id)}>Delete</a></td>
</tr>
@endforeach

```

This code snippet manages categories in the Wear Wise app. It offers a form to add categories and displays them in a table. The table shows Category ID, Name, and Delete option. The Edit option is a placeholder link.

5. Dashboard:

```

<!--New card Total prods-->
@php
    $proCount = $product_data->count();
@endphp

<div class="row">
    <div class="col-xl-3 col-sm-6 grid-margin stretch-card">
        <div class="card">
            <div class="card-body">

```

```

        <div class="row">
            <div class="col-9">
                <div class="d-flex align-items-center
align-self-start">
                    <h3 class="mb-0">{ { $proCount } } Types</h3>
                    </div>
                <div class="col-3">
                    <div class="icon icon-box-success ">
                        <span class="mdi mdi-arrow-top-right
icon-item"></span>
                    </div>
                </div>
            </div>
            <h6 class="text-muted font-weight-normal">Total
Product Types</h6>
        </div>
    </div>
</div>

<!--New card orders total-->

@php

$orderCount = $order_data->count();
@endphp

@php

$totalPrice = $order_data->sum('price');
@endphp
@php
$userCount = $cus_data->count();
@endphp
@php

$w_Price = $w_order->sum('price');
@endphp

```

```
@php
$t_Price = $order_data->sum('price');
$w_Price = $w_order->sum('price');
$vp_sale= $t_Price-$w_Price

@endphp
```

```
@php
$t_Price1 = $order_data->sum('price');
$w_Price2 = $w_order->sum('price');
$vp_sale3= ($t_Price1-$w_Price2)*0.1
@endphp
```

This code segment creates a dashboard displaying statistics for a Wear Wise app. It calculates and presents counts of product types, orders, users, and vendor-specific sales. It also computes and displays the difference between total and vendor sales, considering a percentage.

6. Sidebar, Navbar, header:

Navbar:

```
<li class="nav-item">
    <a class="nav-link" href="userlogin" style="font-weight: bold;">Login</a>
</li>

<li class="nav-item">
    <a class="nav-link" href="usersignup" style="font-weight: bold;">Signup</a>
</li>
<a class="nav-link" href="adminlogin" style="font-weight: bold;">Admin Login</a>
</li>
```

Header:

```
<li class="nav-item dropdown">
    <a class="nav-link" id="profileDropdown" href="#" data-toggle="dropdown">
```

```

<div class="navbar-profile">
    
    <p class="mb-0 d-none d-sm-block navbar-profile-name">{{ session('admin') }}</p>
    <i class="mdi mdi-menu-down d-none d-sm-block"></i>
</div>

```

```

<a class="dropdown-item preview-item" href="{{ url('/logout') }}">
    <div class="preview-thumbnail">
        <div class="preview-icon bg-dark rounded-circle">
            <i class="mdi mdi-logout text-danger"></i>
        </div>
    </div>
    <div class="preview-item-content">
        <p class="preview-subject mb-1">Log out</p>
    </div>
</a>

```

Sidebar:

```

<span class="menu-title">Manage Products</span>
    <i class="menu-arrow"></i>
</a>
<div class="collapse" id="ui-basic">
    <ul class="nav flex-column sub-menu">
        <li class="nav-item"> <a class="nav-link" href="{{ url('/view_product') }}>Add Products</a></li>
        <li class="nav-item"> <a class="nav-link" href="{{ url('/show_products') }}>View Products</a></li>
        <li class="nav-item"> <a class="nav-link" href="{{ url('/a_v_show_products') }}>Approve vendor products</a></li>
    </ul>

```

```

<li class="nav-item menu-items">
    <a class="nav-link" href="{{ url('add_catagory') }}>
        <span class="menu-icon">

```

```

        <i class="mdi mdi-playlist-play"></i>
    </span>
    <span class="menu-title">Manage Category</span>
</a>
</li>

```

7. View Orders:

```

<h1 class="text-center text-white">Wear Wise Products</h1>
<div class="table-responsive">
<table class="table table-bordered mt-5">
<thead class="bg-secondary text-light text-center">
<tr class="text-center">

    <th>Order ID</th>
    <th>Customer Name</th>

    <th>phone</th>

    <th>address</th>
    <th>Product Name</th>
    <th>Image</th>
    <th>Price</th>
    <th>Quantity</th>
    <th>Start date</th>
    <th>End date</th>
    <th>Vendor</th>
    <th>Delete</th>
</tr>

<tbody style="background-color: #eaf4f4; color: #333;">

    @foreach($product_data as $product_data)
    <tr class='text-center'>

        <td>{ ${$product_data->}id }</td>
        <td>{ ${$product_data->}name }</td>
        <td>{ ${$product_data->}phone }</td>
        <td>{ ${$product_data->}address }</td>

```

```

<td>{ ${product_data->product_title}}</td>
<td>
    

</td>
<td>{ ${product_data->price}}</td>
<td>{ ${product_data->quantity}}</td>
<td>{ ${product_data->created_at->format('Y-m-d')} }</td>
@php
    $daysToAdd = $product_data->day; // Change this to the number of days
you want to add
    $newDate = $product_data->created_at->addDays($daysToAdd);
@endphp

<td>{ ${newDate->format('Y-m-d')} }</td>

<td>{ ${product_data->vendor_name} }</td>

<td><a onclick="return confirm('Confirm Delete?')" class="btn
btn-danger"
href="{{url('delete_orders', $product_data->id)}}>Delete</a></td>
</tr>
@endforeach
</tbody>
```

This code snippet represents a table that displays order details, including customer information, product details, dates, and vendor information. The foreach loop iterates through the \$product_data collection to populate the table rows with the respective order details. The code also calculates the end date based on the specified number of days to add to the creation date.

Module 2: Home

1. Header for user:

```
<!-->
<li class="nav-item">
  <a class="nav-link" href="{{url('product_show')}}">Products</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="blog_list.html">Blog</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="contact.html">Contact</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{{url('show_order')}}">Order</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="{{url('profile')}}">Profile</a>      You, last week •
</li>
<form class="form-inline">
  <button class="btn my-2 my-sm-0 nav_search-btn" type="submit">
    <i class="fa fa-search" aria-hidden="true"></i>
  </button>
</form>
```

This code snippet represents a responsive header section for a website. It includes a navigation bar with a logo, links to various pages such as Home, About, Testimonial, Products, Blog, Contact, Order, and Profile, along with a search button. The navigation items are organized in a collapsible dropdown menu for smaller screens. The code uses Bootstrap classes and dynamic URL references for routing.

2. Header for guest user:

```
<li class="nav-item">
|   <a class="nav-link" href="{{url('product_show')}}">Products</a>      You, 2 weeks ago
</li>
<li class="nav-item">
|   <a class="nav-link" href="blog_list.html">Blog</a>
</li>
<li class="nav-item">
|   <a class="nav-link" href="contact.html">Contact</a>
</li>
<form class="form-inline">
|   <button class="btn my-2 my-sm-0 nav-search-btn" type="submit">
|       <i class="fa fa-search" aria-hidden="true"></i>
|   </button>
</form>

<li class="nav-item">
|   <a class="btn btn-primary" id="logincss" href="{{url('/userlogin')}}">Login</a>
</li>
<li class="nav-item">
|   <a class="btn btn-primary" id="logincss" href="{{url('/usersignup')}}">Register</a>
</li>
```

This code snippet creates a responsive header section for a website using Bootstrap's navigation components. It includes a logo, a collapsible navigation menu with links to Home, About, Testimonial, Products, Blog, and Contact pages. There's also a search button within the header. Additionally, there are "Login" and "Register" buttons styled as primary buttons. The navigation items and buttons are organized and styled within a container.

3. Slider:

```
<div id="customCarousel1" class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <div class="container ">
        <div class="row">
          <div class="col-md-7 col-lg-6 ">
            <div class="detail-box">
              <h1>
                <span>
                  Sale 20% Off
                </span>
                <br>
                On Everything
              </h1>
              <p>
                Explicabo esse amet tempora quibusdam laudantium, laborum eaque magnam fugiat hic? Esse dicta aliquid error repudiandae earum suscipit
              </p>
              <div class="btn-box">
                <a href="#" class="btn1">
                  Shop Now
                </a>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
    <div class="carousel-item ">
      <div class="container ">
        <div class="row">
          <div class="col-md-7 col-lg-6 ">
```

This code snippet creates a responsive image slider section with a promotional message and a "Shop Now" button, utilizing Bootstrap's carousel component.

4. Product display with pagination :

```
<section class= product_section layout_padding >
  <div class="container">
    <div class="heading_container heading_center">
      <h2>
        Our <span>products</span>
      </h2>
      <div>
        <form action="{{url('product_search')}}" method="GET">
          @csrf
          <input type="text" name="search" placeholder="Search for Something">
          <input type="submit" value="search">
        </form>
      </div>
    </div>
    <div class="row">
      @foreach($products as $product)
        <div class="col-sm-8 col-md-6 col-lg-6">
          <div class="box">
            <div class="option_container">
              <div class="options">
                <a href="{{url('product_details',$products->product_id)}}" class="option1">
                  Product Details
                </a>
              </div>
            </div>
          </div>
        </div>
      @endforeach
    </div>
  </div>
</section>
```

This code snippet creates a product section displaying items with details, vendor information, and an "Add to Cart" option. Users can search for products and adjust quantity and rental days. It also handles discounts and pagination, with Bootstrap styling.

5. Search for products:

```
<div class="container">
  <div class="heading_container heading_center">
    <h2>
      Our <span>products</span>
    </h2>
    <div>
      <form action="{{url('product_search')}}" method="GET">
        @csrf
        <input type="text" name="search" placeholder="Search for Something">
        <input type="submit" value="search">
      </form>
    </div>
  </div>
  <div class="row">
```

This code snippet represents a product section within a webpage layout. It includes a container with a centered heading indicating "Our Products." The section incorporates a search form allowing users to search for specific items. The section also starts a row for displaying individual product items.

6. Why shop with us :

Why Shop With Us

Describes the pros of our product and website.

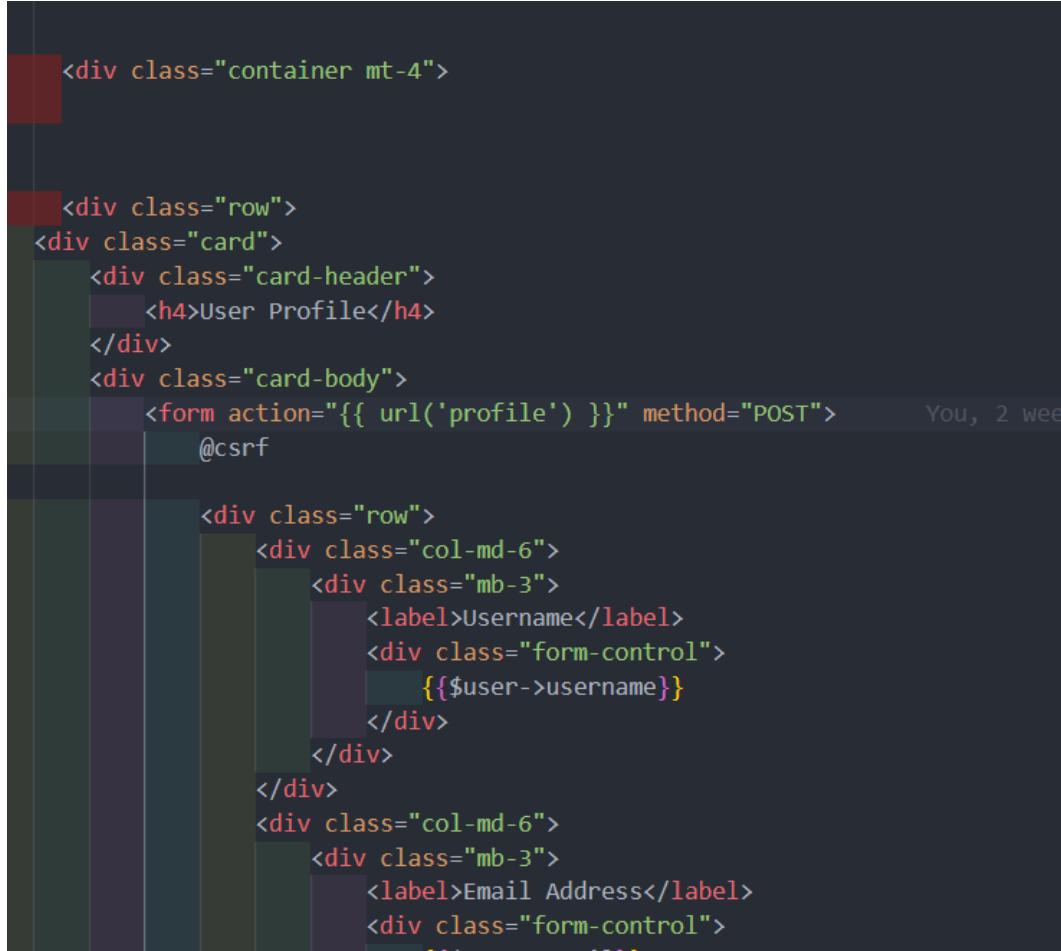
7. Footer :

```
<div class="information_f">
  <p><strong>ADDRESS:</strong> 28 White tower, Street Name New York City, USA</p>
  <p><strong>TELEPHONE:</strong> +91 987 654 3210</p>
  <p><strong>EMAIL:</strong> yourmain@gmail.com</p>
</div>
</div>
</div>
<div class="col-md-8">
  <div class="row">
    <div class="col-md-7">
      <div class="row">
        <div class="col-md-6">
          <div class="widget_menu">
            <h3>Menu</h3>
            <ul>
              <li><a href="{{url('/')}}">Home</a></li>
              <li><a href="#">About</a></li>
              <li><a href="#">Services</a></li>
              <li><a href="#">Testimonial</a></li>
              <li><a href="#">Blog</a></li>
              <li><a href="#">Contact</a></li>
            </ul>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

This code snippet defines a footer section for a website. It consists of a container with two columns, the left column displaying the logo and contact information, while the right column contains navigational menus, account links, and a newsletter subscription form. The footer provides a well-structured layout with essential website elements, including links to

different pages, account-related actions, and a newsletter subscription option.

8. Product description :



The screenshot shows a code editor with an HTML file open. The code is a template for a web page, likely for a fashion-related website. It includes meta tags for basic information, references to CSS and JavaScript files for styling and functionality, and a structure for displaying user profile information. The template also includes a navigation header, a hero section, and a user profile card with details such as username, email, phone number, name, address, and an edit profile link. It utilizes Bootstrap for styling and interactivity, and includes a script to store and restore scroll position when the page is refreshed.

```
<div class="container mt-4">
    <div class="row">
        <div class="card">
            <div class="card-header">
                <h4>User Profile</h4>
            </div>
            <div class="card-body">
                <form action="{{ url('profile') }}" method="POST">
                    @csrf
                    <div class="row">
                        <div class="col-md-6">
                            <div class="mb-3">
                                <label>Username</label>
                                <div class="form-control">
                                    {{ $user->username }}
                                </div>
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="mb-3">
                                <label>Email Address</label>
                                <div class="form-control">
                                    {{ $user->email }}
                                </div>
                            </div>
                        </div>
                    </div>
                    <div class="d-grid gap-2 d-md-flex justify-content-md-end">
                        <a href="#" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#editProfileModal">Edit Profile</a>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
```

This HTML document is a template for a web page, likely for a fashion-related website. It includes meta tags for basic information, references to CSS and JavaScript files for styling and functionality, and a structure for displaying user profile information. The template also includes a navigation header, a hero section, and a user profile card with details such as username, email, phone number, name, address, and an edit profile link. It utilizes Bootstrap for styling and interactivity, and includes a script to store and restore scroll position when the page is refreshed.

9. Product Details :

```

<link href="home/css/style.css" rel="stylesheet" />
<!-- responsive style -->
<link href="home/css/responsive.css" rel="stylesheet" />
</head>
<body>
    <div class="hero_area">
        <!-- header section starts -->
        @include('home.header')
        <!-- end header section -->

        <div class="col-sm-8 col-md-6 col-lg-6" style="margin: auto; width: 50%; padding: 30px">
            <div class="card" style="padding: 20px">      You, 1 second ago • Uncommitted change
                
            </div>
            <div class="detail-box">
                <h5>
                    {{ $product->product_title}}
                </h5>
                @if($product->discounted_price!=null)
                    <h6 style="color: red">
                        Discount price
                    <br>
                
```

This HTML document serves as a template for a web page, likely related to a fashion website. It features a header section, a central content area displaying detailed product information, and a footer section. The content area includes an image of the product, its title, pricing details (including discounts), product description, available quantity, and vendor name. Users can specify the quantity and rental days before adding the item to the cart. The template employs Bootstrap for styling and interactivity and provides a responsive design for various devices. It concludes with copyright information and attribution links in the footer.

Module 3: User Dashboard

1. Add to Cart :

```
<body>
    @include('sweetalert::alert')
    <div class="hero_area">
        <!-- header section strats -->
        @include('home.header')
        <!-- end header section -->
        <!-- slider section -->
        @if(session()->has('message'))
            <div class="alert alert-success">
                <button type="button" class="close" data-dismiss="alert" aria-hidden="true">>x</button>
                {{session()->get('message')}}
            </div>
        @endif
        <!-- end slider section -->

        <!-- why section -->
        <div class="center">
            <table>
                <tr>
                    <th class="th_deg">Product Title</th>
                    <th class="th_deg">Product quantity</th>
                    <th class="th_deg">Day</th>
                    <th class="th_deg">Price</th>
                    <th class="th_deg">Image</th>
                    <th class="th_deg">Action</th>
                </tr>

```

The frontend development of the e-commerce template comprises a well-structured layout with a header, shopping cart display in tabular format, and options for proceeding to checkout. It features responsive design using Bootstrap, includes a 'sweet alert' library for user alerts and confirmations, and enables users to remove items from the cart. The template also offers buttons to continue shopping, opt for cash on delivery, or make a card payment. Overall, it creates a seamless and user-friendly shopping experience with clear product information and payment choices.

2. Order Details :

```
</head>
<body>
    @include('sweetalert::alert')

    
    @include('home.header')
    
    <div class="center">
        <table>
            <tr>
                <th class="th_deg">Product Title</th>
                <th class="th_deg">Vendor Name</th>
                <th class="th_deg">Quantity</th>
                <th class="th_deg">Day</th>
                <th class="th_deg">Price</th>
                <th class="th_deg">Payment Status</th>
                <th class="th_deg">Delivery Status</th>
                <th class="th_deg">Date of order</th>
                <th class="th_deg">Return Date</th>
                <th class="th_deg">Image</th>
                <th class="th_deg">Cancel Order</th>
                <th class="th_deg">Print</th>
            </tr>
        </table>
    </div>
</body>
```

This HTML document represents a webpage for displaying and managing orders in this platform. The page includes a header section with basic meta information and styling references. The main content area is centered and contains a table displaying order details, such as product title, vendor name, quantity, day, price, payment status, delivery status, order date, return date, product image, cancel order button, and a print PDF button. The table is populated dynamically using a loop that iterates through orders fetched from the backend. The "Cancel Order" button allows users to cancel orders with a confirmation prompt, and the "Print PDF" button generates a PDF version of the order. The script tags at the end handle scroll position preservation and load JavaScript dependencies for jQuery, popper, bootstrap, and custom scripts. Overall, the page provides a comprehensive view and management options for customer orders.

3. Profile view:

```
<div class="row">
  <div class="col-md-6">
    <div class="mb-3">
      <label>Username</label>
      <div class="form-control">
        {{$user->username}}
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <div class="mb-3">
      <label>Email Address</label>
      <div class="form-control">
        {{$user->email}}
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <div class="mb-3">
      <label>Phone Number</label>
      <div class="form-control">
        {{$user->phone}}
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <div class="mb-3">
      <label>Name</label>
      <div class="form-control">
        {{$user->name}}
      </div>
    </div>
  </div>
  <div class="col-md-12">
    <div class="mb-3">
      <label>Address</label>
      <div class="form-control">
        {{ $user->address }}
      </div>
    </div>
  </div>
</div>
```

The page includes user details such as username, email address, phone number, name, and address, displayed in a structured form layout. Users can view their profile information and click on an "Edit Profile" button to make changes. The template also includes a header section with navigation, and it features responsive design for compatibility with different devices. It utilizes Bootstrap for styling and includes JavaScript libraries for interactivity and functionality.

4. Profile Edit :

```
<div class="mb-3">
<label>Username</label>
<input type="text" placeholder="Enter your username here" name="username" value="{{\$user->username}}" class="form-control" />
</div>
</div>
<div class="col-md-6">
<div class="mb-3">
<label>Email Address</label>      You, last week • User Module ...
<input type="text" readonly value="{{\$user->email}}" class="form-control" />
</div>
</div>
<div class="col-md-6">
<div class="mb-3">
<label>Phone Number</label>
<input type="text" name="phone" placeholder="Enter your phone number here" value="{{\$user->phone}}" class="form-control" />
</div>
</div>
<div class="col-md-6">
<div class="mb-3">
<label>Name</label>
<input type="text" name="name" placeholder="Enter your name here" value="{{\$user->name}}" class="form-control" />
</div>
</div>
<div class="col-md-12">
<div class="mb-3">
<label>Address</label>
<textarea name="address" placeholder="Enter your address here" class="form-control" rows="3">{{\$user->address}}</textarea>
</div>
</div>
<div class="col-md-12">
    <button type="submit" class="btn btn-success btn-block">Update Account</button>
</div>
```

The page allows users to edit and update their profile information. It includes a form with fields for username, email address (readonly), phone number, name, and address. Users can input their updated information and submit the form to update their account details. The template features responsive design, utilizes Bootstrap for styling, and includes header navigation from the main template. The provided code allows users to interactively update their profile information.

5. Payment gateway:

```

var $form = $(".require-validation");

$('.form.require-validation').bind('submit', function(e) {
    var $form = $(".require-validation"),
        inputSelector = ['input[type=email]', 'input[type=password]',
                        'input[type=text]', 'input[type=file]',
                        'textarea'].join(', '),
        $inputs = $form.find('.required').find(inputSelector),
        errorMessage = $form.find('div.error'),
        valid = true;
    errorMessage.addClass('hide');

    $('.has-error').removeClass('has-error');
    $inputs.each(function(i, el) {
        var $input = $(el);
        if ($input.val() === '') {
            $input.parent().addClass('has-error');
            errorMessage.removeClass('hide');
            e.preventDefault();
        }
    });
}

if (!$form.data('cc-on-file')) {
    e.preventDefault();
    Stripe.setPublishableKey($form.data('stripe-publishable-key'));
    Stripe.createToken({
        number: $('.card-number').val(),
        cvc: $('.card-cvc').val(),
        exp_month: $('.card-expiry-month').val(),
        exp_year: $('.card-expiry-year').val()
    }, stripeResponseHandler);
}

});

```

The template is integrated with the Bootstrap CSS framework and includes jQuery and Stripe's JavaScript library for handling the payment process. The form data is submitted to the server for processing the payment using the Stripe API. It features responsive design and includes header navigation from the main template. The template enables users to securely make payments using their credit cards through Stripe's payment gateway.

6. Print PDF:

```
<h1>Order Details</h1>

Customer Name :<h3>{$order->name}</h3>
Customer Email :<h3>{$order->email}</h3>
Customer Phone :<h3>{$order->phone}</h3>
Customer Address :<h3>{$order->address}</h3>
@php
$daysToAdd = $order->day; // Change this to the number of days you want to add
$newDate = $order->created_at->addDays($daysToAdd);
@endphp
You, 2 weeks ago • Order details ...
Product Name :<h3>{$order->product_title}</h3>
Product Price :<h3>{$order->price}</h3>
Product Quantity :<h3>{$order->quantity}</h3>
Payment Status :<h3>{$order->payment_status}</h3>
Return date : <h3>{$newDate->format('Y-m-d')}</h3>
```

This HTML template generates an order details PDF document. It displays information related to a customer's order, including their name, email, phone number, and address. Additionally, it provides details about the purchased product, such as its name, price, and quantity. The template also includes the payment status of the order and the calculated return date based on the provided number of days.

The template is designed to be used for generating order-related PDFs and is intended to be dynamically populated with data from the server. It does not include any styling or formatting beyond the basic structure required to present the order information in a clear and organized manner.

Module 4: Vendor Panel

1. Dashboard:

```
@php
$prodCount = $order_data->count();
@endphp


```

```

<div class="col-9">
    <div class="d-flex align-items-center align-self-start">
        <h3 class="mb-0">{ ${prodCount} } Posts</h3>
        </div>
    </div>
    <div class="col-3">
        <div class="icon icon-box-success ">
            <span class="mdi mdi-arrow-top-right icon-item"></span>
        </div>
        </div>
    </div>
    <h6 class="text-muted font-weight-normal">Total Pending products</h6>
    </div>
</div>
</div>

@php
    $orderCount = $order_data->count();
@endphp

```

```

@php

$totalPrice = $order_data->sum('price');
@endphp

```

```

@php
    $proCount = $product_data->count();
@endphp

```

```

@php
$totalPrice1 = $order_data->sum('price');
$earn=$totalPrice1*0.95
@endphp

```

It calculates and displays various statistics related to the vendor's products and orders. The code calculates the number of pending products, total orders, and total earnings based on the orders' prices.

2. Show approved vendor products:

```

<th>ID</th>
    <th>Name</th>
    <th>Image</th>
    <th>Price</th>
    <th>Discount Price</th>
    <th>Quantity</th>
    <th>Day count on price</th>
    <th>Vendor</th>
    <th>Delete</th>
</tr>

<tbody style="background-color: #eaf4f4; color: #333;">

@foreach($product_data as $product_data)
    <tr class='text-center'>

        <td>{{ $product_data->product_id }}</td>
        <td>{{ $product_data->product_title }}</td>
        <td>
            
        </td>
        <td>{{ $product_data->price }}</td>
        <td>{{ $product_data->discounted_price }}</td>
        <td>{{ $product_data->quantity }}</td>
        <td>{{ $product_data->days }}</td>
        <td>{{ $product_data->vendor_name }}</td>
    </tr>
@endforeach

```

```
<td><a onclick="return confirm('Confirm Delete?')" class="btn  
btn-danger"  
href="{{url('delete_product',$product_data->product_id)}}>Delete</a></td>  
</tr>  
  
@endforeach
```

Table displaying product details, including ID, Name, Image, Price, Discount Price, Quantity, Day Count on Price, Vendor, and Delete option. The table body iterates through product data, populating each row accordingly.

3. Rest of the pages are similar to that of Admin panels

Backend Development

Admin Panel:

1. Signup:

```

public function admin_data_store(Request $request) {

    if
(Adminsignup::where('username', $request['username'])->exists()) {
        return redirect()->back()->with('message', 'Username Taken');
    }
    elseif ($request['password'] != $request['cpassword']) {
        return redirect()->back()->with('message', 'Password Did not
match');
    }
    elseif
(Adminsignup::where('reference_code', $request['ref_code'])->doesntExist())
{
        return redirect()->back()->with('message', 'Reference code Did
not match');
    }

    else{
        $admin_info= new AdminSignup;
        $admin_info->name= $request['Name'];
        $admin_info->phone= $request['phone_number'];
        $admin_info->username= $request['username'];
        $admin_info->email= $request['email'];
        $admin_info->password= Crypt::encrypt($request['password']);
        $admin_info->reference_code= $request['ref_code_gen'];
        $admin_info->save();

        $request->session()->put('admin_info', $request['username']);

        return redirect('adminlogin');
    }
}

```

```
}
```

This code handles admin registration. It checks username availability, password matching, and reference code existence. If conditions are met, it stores encrypted admin data and sets a session for the username, then redirects to the login page. If conditions aren't met, relevant error messages are shown using redirect().

2. Login:

```
public function admin_login(Request $request) {
    if (Adminsignup::where('username', $request['username'])->doesntExist()) {
        return redirect()->back()->with('message', 'Wrong username');

    }
    else{
        $admin= AdminSignup::where ("username",
        $request->input ('username'))->get();

        $decrepted= Crypt::decrypt ($admin[0]->password);

        if ($decrepted==$request['password']){
            $request->session()->put ('admin', $request['username']);

            return redirect ('admin_dashboard');
        }
        else{

            return redirect ()->back ()->with ('message', 'Password did
not match');

        }
    }
}
```

This code manages admin login. It verifies the username's existence, decrypts the password for comparison, and checks whether it matches. If successful, it sets a session for the admin's username and redirects to the admin dashboard. If not, error messages are displayed using redirect().

3. Session Management:

i)

```
class AdminAuthMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure(\Illuminate\Http\Request):
     * (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse) $next
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle($request, Closure $next)
    {
        if ($request->session()->has('admin')) {
            return $next($request);
        }

        return response()->view('adminpanel.adminlogin');
        //return redirect()->route('adminlogin')->with('message', 'Please
        login to access the admin dashboard.');
    }
}
```

Similarly user, vendor

ii) Kernel.php:

```
'admin.auth' => \App\Http\Middleware\AdminAuthMiddleware::class,
'user.auth'  => \App\Http\Middleware\UserAuthMiddleware::class,
'vendor.auth' => \App\Http\Middleware\VendorAuthMiddleware::class,
```

iii) Web.php:

```
Route::middleware(['admin.auth'])->group(function () {
```

```
Route::middleware(['user.auth'])->group(function () {
```

```
Route::middleware(['vendor.auth'])->group(function () {
```

- i) This code defines middleware named AdminAuthMiddleware, UserAuthMiddleware, and VendorAuthMiddleware to manage session authentication for admin, user, and vendor, respectively. If the respective session exists, the user proceeds; otherwise, they're redirected to the login page.
- ii) In the Kernel.php file, middleware aliases are added to refer to the defined middleware classes for admin, user, and vendor authentication.
- iii) In web.php, routes are grouped with corresponding middleware to enforce session-based authentication for admin, user, and vendor sections of the application.

4. Add product:

```
public function add_product(Request $request) {
    $data= new products;
    $data->product_title= $request['product_title'];
    $data->product_description= $request['product_description'];
    $data->price = $request['product_price'];
    $data->days = $request['product_days'];
    $data->discounted_price= $request['product_discount_price'];
    $data->quantity= $request['product_quantity'];
    $data->catagory_id= $request['product_category'];
    $data->apparel_id= $request['product_apparel'];
    $data->vendor_name= "Wear Wise";

    $image=$request->image;
    $imagename= time().'.'.$image->getClientOriginalExtension();
    $request->image->move('added_products', $imagename);
    $data->image=$imagename;
```

```

        $data-> save();

        return redirect()->back()->with('message', 'Product Added
successfully');
    }
}

```

The add_product function creates a new product entry with user-input details, including title, description, price, days, and image. It then saves the entry and provides a success message.

5. Add product initialize (edit products initialize similar):

```

public function view_product(){
    if (session()->has('admin')) {

        $cat= Catagory::all();
        $app= Apparel::all();

        return
view('adminpanel.add_product', compact('cat'), compact('app'));
    }
    else {
        return redirect('adminlogin')->with('message', 'Please
login to access the admin panel features.');

    }
}

```

This function, named view_product, renders the "add_product" view in the admin panel. It fetches all categories and apparels from the database. Then, it loads the "add_product" view, passing along the fetched category and apparel data for selection.

6. Show products/ View products:

```

public function show_products(){
    $product_data= products::all();

    return view('adminpanel.show_products', compact('product_data'));
}

```

Fetches all product data and displays it using the 'show_products' view.

7. Delete products:

```
public function delete_product($product_id) {

    $data=products::where('product_id', $product_id);
    $data->delete();
    return redirect()->back()->with('message', 'Product Deleted successfully');
}
```

Deletes a specific product by its ID and provides a success message.

8. Update products:

```
public function update_product(Request $request, $product_id){

    $data=products::find($product_id);
    $data->product_title= $request->product_title;
    $data->product_description= $request->product_description;
    $data->price =$request->product_price;
    $data->days =$request->product_days;
    $data->discounted_price= $request->product_discount_price;
    $data->quantity= $request->product_quantity;
    $data->catagory_id= $request->product_category;
    $data->apparel_id= $request->product_apparel;
    $data->vendor_name= "Wear Wise";

    $image=$request->image;
    if ($image) {

        $imagename= time() . '.' . $image->getClientOriginalExtension();
        $request->image->move('added_products', $imagename);
        $data->image=$imagename;

    }

    $data->save();
```

```
        return redirect()->back()->with('message', 'Product Updated  
successfully');  
  
    }  

```

Modifies the product's details based on user input and saves the changes. Also handles the updating of the product image if a new image is provided. Returns to the previous page with a success message.

9. View Order & Delete order: (Customer similar)

```
public function order()
{
    $product_data = Order::all();

    return view('adminpanel.order', compact('product_data'));
}

public function delete_orders($product_id) {

    $data=Order::where('id', $product_id);
    $data->delete();
    return redirect()->back()->with('message', 'Order Deleted successfully');
}
```

Order: Retrieves all orders from the database and displays them in the admin panel's order page.

Delete orders: Deletes a specific order identified by its ID from the database and redirects back to the previous page with a success message.

10.Show, delete & approve vendors:

```

public function show_vendors(){
    $vendor_data= VendorSignup::all();

    return view('adminpanel.show_vendors',compact('vendor_data'));
}

public function delete_vendor($vendor_id){

    $data=VendorSignup::where('id', $vendor_id);
    $data->delete();
    return redirect()->back()->with('message', 'Vendor Deleted successfully');
}

public function approve_vendor($vendor_id) {
    // Retrieve the data from the VendorSignup model
    $vendorData = VendorSignup::where('id',
$vendor_id)->first();

    if ($vendorData) {
        // Create a new instance of the FinalVendors model
        $data = new final_vendors;

        // Assign the properties from the $vendorData object
        $data->name = $vendorData->name;
        $data->phone = $vendorData->phone;
        $data->username = $vendorData->username;
        $data->email = $vendorData->email;
        $data->password = $vendorData->password;
        $data->buisness_name = $vendorData->buisness_name;
        $data->address = $vendorData->address;
        $data->buisness_lisence_no =
$vendorData->buisness_lisence_no;

        $data->save();
        $vendorData->delete();

        return redirect()->back()->with('message', 'Vendor Approved successfully');
    }
}

```

```
        } else {
            // Handle the case when the vendor data with the given
ID is not found
            return redirect()->back()->with('error', 'Vendor data
not found');
        }
    }
}
```

Show Vendors: Retrieves all vendor information from the VendorSignup model and displays it in the admin panel's vendors page.

Delete Vendor: Deletes a specific vendor identified by its ID from the VendorSignup model and redirects back to the previous page with a success message.

Approve Vendor: Retrieves the data of a specific vendor by its ID from the VendorSignup model, creates a new entry in the FinalVendors model using the retrieved data, and then deletes the vendor's data from the VendorSignup model. Finally, it redirects back to the previous page with a success message. If the vendor data is not found, an error message is displayed.

11. View certified vendors & delete them (while deleting the products posted by them):

```
public function final_vendors(){
    $vendor_data= final_vendors::all();

    return

view('adminpanel.final_vendors',compact('vendor_data'));

}

public function delete_f_vendor($vendor_id){
    $vendor = final_vendors::find($vendor_id);
    $matchingProducts = products::where('vendor_name',
$vendor->buisness_name)->get();
    if ($matchingProducts->isNotEmpty()) {
        foreach ($matchingProducts as $product) {
            $product->delete();
        }
    }
}
```

```

    $data=final_vendors ::where('id', $vendor_id);

    $data->delete();
    return redirect()->back()->with('message', 'Vendor and the
vendor products deleted');
}

```

Final Vendors: Retrieves all finalized vendor information from the final_vendors model and displays it in the admin panel's final vendors page.

Delete Final Vendor: Deletes a specific finalized vendor identified by its ID from the final_vendors model. If the vendor has associated products in the products model, those products are also deleted. The function then redirects back to the previous page with a success message indicating that both the vendor and their associated products have been deleted.

12.Approve vendor products:

```

public function approve_product($product_id) {
    // Retrieve the data from the productSignup model
    $productData = Temp_product::where('id', $product_id)->first();

    if ($productData) {
        // Create a new instance of the Finalproducts model
        $data = new products;

        // Assign the properties from the $productData object
        $data->product_title = $productData->product_title;
        $data->product_description = $productData->product_description;
        $data->price = $productData->price;
        $data->days = $productData->days;
        $data->discounted_price = $productData->discounted_price;
        $data->quantity = $productData->quantity;
        $data->catagory_id = $productData->catagory_id;
        $data->apparel_id = $productData->apparel_id;
        $data->vendor_name = $productData->vendor_name;
        $data->image = $productData->image;
    }
}

```

```

        $data->save();
        $productData->delete();

        return redirect()->back()->with('message', 'product Approved
successfully');
    } else {
        // Handle the case when the product data with the given ID is not
found
        return redirect()->back()->with('error', 'product data not
found');
    }
}

```

Approve Product: This function is used to approve a product that was initially added by a vendor but was pending approval. It retrieves the product data from the Temp_product model based on the provided product ID. If the product data exists, it creates a new instance of the products model, assigns the properties from the productData object, and then saves the new product to the products model. The productData entry is then deleted from the Temp_product model. The function redirects back to the previous page with a success message indicating that the product has been approved. If the product data with the given ID is not found, it redirects with an error message.

14. Admin dashboard:

```

public function index(){
    $order_data= Order::all();
    $product_data=products::all();
    $cus_data=UserSignup::all();
    $w_order= Order:: where('vendor_name', 'Wear wise')->get();

    return view('adminpanel.admin_dashboard', compact('order_data',
'product_data', 'cus_data', 'w_order'));

}

```

Fetches all order, product, and customer data, as well as specific orders from the 'Wear wise' vendor, then sends this data to the 'admin_dashboard' view for display.

Vendor Panel:

1. **Vendor signup, Login** : similar to admin and user except getting approved by admin.

2. Vendor add product:

```
public function v_add_product(Request $request) {
    $data= new Temp_product;
    $log = session('vendor');
    $row = final_vendors::where('username', $log)->first();

    if ($row) {
        $vendorName = $row->buisness_name;
    }

    $data->product_title= $request['product_title'];
    $data->product_description= $request['product_description'];
    $data->price = $request['product_price'];
    $data->days = $request['product_days'];
    $data->discounted_price= $request['product_discount_price'];
    $data->quantity= $request['product_quantity'];
    $data->catagory_id= $request['product_category'];
    $data->apparel_id= $request['product_apparel'];
    $data->vendor_name= $vendorName;

    $image=$request->image;
    $imagename= time().'.'.$image->getClientOriginalExtension();
    $request->image->move('added_products', $imagename);
    $data->image=$imagename;

    $data-> save();

    return redirect()->back()->with('message', 'Product Added successfully');
}
```

Creates a new temporary product entry by retrieving vendor information from the session, then populates the entry with product details provided by the vendor. The

product image is uploaded and linked, and the entry is saved, followed by a redirection to the previous page with a success message.

3. Vendor show , edit , delete products:

```
public function show_vendor_products() {
    $product_data= Temp_product::all();

    return view('vendor.show_vendor_products',compact('product_data'));
}

public function v_delete_product($product_id) {

    $data=Temp_product::where('id', $product_id);
    $data->delete();
    return redirect()->back()->with('message', 'Product Deleted successfully');
}

public function v_edit_product($product_id) {
    $data=Temp_product::where('id', $product_id)->get();
    $cata= Catagory::all();
    $appa= Apparel::all();
    return view('vendor.v_edit_products',compact('data', 'cata', 'appa'));
}
```

`show_vendor_products` Function: Retrieves all temporary vendor products and displays them in the '`show_vendor_products`' view.

`v_delete_product` Function: Deletes a temporary vendor product entry based on the provided product ID, then redirects back with a success message.

`v_edit_product` Function: Fetches the details of a specific temporary vendor product based on the provided product ID, along with all available categories and apparels. Renders the '`v_edit_products`' view with the fetched data and options for editing.

4. Approved products of vendor:

```
public function show_approved_vendor_products() {
```

```

    $log = session('vendor');

$row = final_vendors::where('username', $log)->first();

if ($row) {
    $vendorName = $row->buisness_name;
}

$product_data= products:: where('vendor_name', $vendorName)->get();
//dd($product_data);
//return view('vendor.test',compact('product_data'));

return
view('vendor.show_approved_vendor_products',compact('product_data'));
}

```

show_approved_vendor_products Function: Retrieves all products that are approved and associated with the currently logged-in vendor. It uses the vendor's username to fetch the business name from the final_vendors table and then retrieves all products with the matching vendor name from the products table. The retrieved product data is then displayed in the 'show_approved_vendor_products' view.

5. Show vendor specific orders & delete:

```

public function v_orders(){
    $log = session('vendor');

$row = final_vendors::where('username', $log)->first();

if ($row) {
    $vendorName = $row->buisness_name;
}

$product_data= Order:: where('vendor_name', $vendorName)->get();

return view('vendor.v_orders',compact('product_data'));
}

public function delete_v_orders($product_id){

$data=Order::where('id', $product_id);
$data->delete();
}

```

```
        return redirect()->back()->with('message', 'Order Deleted  
successfully');  
    }  
  
}
```

v_orders Function: Retrieves orders associated with the currently logged-in vendor. It uses the vendor's username to fetch the business name from the final_vendors table and then retrieves all orders with the matching vendor name from the Order table. The retrieved order data is then displayed in the 'v_orders' view.

delete_v_orders Function: Deletes an order based on the provided product ID. It locates the order using the ID and deletes it from the Order table. Upon successful deletion, the function redirects back to the previous page with a success message.

User Dashboard

1. Add to Cart :

```

    @foreach($cart as $cart)

        <tr>
            <td>{{$cart->product_title}}</td>
            <td>{{$cart->quantity}}</td>
            <td>{{$cart->day}}</td>
            <td>{{$cart->price}}</td>
            <td></td>
            <td>
                <a class="btn btn-danger" onclick="confirmation(event)" href="{{url('remove_cart',$cart->id)}}>Remove</a>
            </td>
        </tr>
        <?php $totalprice=$totalprice + $cart->price ?>
    @endforeach
</table>
<div>
<h1 class="total_deg">Total Price : {{$totalprice}} -/BDT</h1>
</div>

<div class="buttons-container">
    <a class="btn btn-primary" href="{{ url('/') }}>continue Shopping</a> You, 2 weeks ago • Last Homepage push ...
</div>

<div>
    <h1 style="font-size: 25px; padding-bottom: 15px;">Proceed to Order</h1>
    <a href="{{url('cash_order')}}" class="btn btn-danger">Cash On delivery</a>
    <a href="{{url('stripe',$totalprice)}}" class="btn btn-danger">Pay Using Card</a>
</div>
</div>

```

```

public function add_cart(Request $request,$product_id)
{
    $store = session('user'); // Assuming $store holds the username
    $user = UserSignup::where('username', $store)->first();
    $product = products::where('product_id', $product_id)->first(); You, 3 weeks ago • Last Homepage push ...

    $cart=new cart;
    $cart->name=$user->name;
    $cart->email=$user->email;
    $cart->phone=$user->phone;
    $cart->address=$user->address;
    $cart->user_id=$user->id;
    $cart->username=$user->username;

    $cart->product_title=$product->product_title;
    $cart->vendor_name=$product->vendor_name;
    $cart->Product_id=$product->product_id;
    $cart->image=$product->image;
    $cart->quantity=$request->quantity;
    $cart->day=$request->days;
    if($product->discounted_price!=null)
    {
        $cart->price=$product->discounted_price * $request->quantity * $request->days;
    }
    else
    {
        $cart->price=$product->price * $request->quantity * $request->days;
    }
}

```

```
$cart->save();

Alert::success('Product Added Successfully','We have added product to the cart');

return redirect()->back();
}

public function show_cart()
{
    $username = session('user');
    $cart=cart::where('username', '=', $username)->get();
    return view('home.showcart',compact('cart'));
}

public function remove_cart($product_id)
{
    $cart=cart::find($product_id);
    $cart->delete();
    return redirect()->back();
}
```

The backend development code is a responsive e-commerce template featuring a cart display with product details, dynamic total price calculation, and options for continuing shopping or proceeding to checkout. It utilizes SweetAlert for alerts and offers "Cash On Delivery" and "Pay Using Card" payment methods. The code ensures a seamless user experience for managing cart items and initiating orders.

2. Order Details :

```

public function show_order()
{
    $username = session('user');
    $order = order::where('username', '=', $username)->get();
    return view('home.order', compact('order'));
}

public function cancel_order($id)
{
    $order=order::find($id);
    $order->delivery_status='Your order have been canceled';
    $order->save();
    return redirect()->back();

}

@foreach($order as $order)
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| {{ $order->product_title}}</td>  {{ $order->vendor_name}}</td>  {{ $order->quantity}}</td>  {{ $order->day}}</td>  {{ $order->price}}</td>  {{ $order->payment_status}}</td>  {{ $order->delivery_status}}</td>  {{ $order->created_at->format('Y-m-d') }}</td>  @php $daysToAdd = $order->day; // Change this to the number of days you want to add $newDate = $order->created_at->addDays($daysToAdd); @endphp  <td> {{ $newDate->format('Y-m-d') }}</td> <td> Product ImageCancel @else  Not Allowed</p> @endif <td> Print PDF </td>  @endforeach | | | | | | | |

```

This backend development code generates a web page displaying an order history table for this platform. The table includes product details, vendor information, quantity, days, price, payment and delivery status, order and return dates, and product image. Users can cancel orders in the 'processing' delivery status and print PDF receipts. The page features responsive design, uses SweetAlert for confirmations, and includes navigation and styling components for a complete user experience.

3. Profile view:

```

public function profile()
{
    $username = session('user');
    $user = UserSignup::where('username', $username)->first();

    return view('home.profile', compact('user'));
}

<div class="row">
    <div class="col-md-6">
        <div class="mb-3">
            <label>Username</label>
            <div class="form-control">
                {{$user->username}}
            </div>
        </div>
    </div>
    <div class="col-md-6">
        <div class="mb-3">
            <label>Email Address</label>
            <div class="form-control">
                {{$user->email}}
            </div>
        </div>
    </div>
    <div class="col-md-6">
        <div class="mb-3">
            <label>Phone Number</label>
            <div class="form-control">
                {{$user->phone}}
            </div>
        </div>
    </div>
    <div class="col-md-6">
        <div class="mb-3">
            <label>Name</label>
            <div class="form-control">
                {{$user->name}}
            </div>
        </div>
    </div>
    <div class="col-md-12">
        <div class="mb-3">
            <label>Address</label>
            <div class="form-control">
                ...
            </div>
        </div>
    </div>

```

The page includes user details such as username, email address, phone number, name, and address, displayed in a structured form layout. Users can view their profile information and click on an "Edit Profile" button to make

changes. The template also includes a header section with navigation, and it features responsive design for compatibility with different devices. It utilizes Bootstrap for styling and includes JavaScript libraries for interactivity and functionality.

```
Route::middleware(['user.auth'])->group(function () {  
  
    Route :: get('/test1',[admin_content_controller::class,'view_test1']);  
    route::post('/add_cart/{product_id}',[HomeController::class,'add_cart']);  
    Route::get('/', [HomeController::class, 'index']);  
    route::get('/show_cart',[HomeController::class,'show_cart']);  
    route::get('/remove_cart/{id}',[HomeController::class,'remove_cart']);  
    route::get('/cash_order',[HomeController::class,'cash_order']);  
    route::get('/stripe/{totalprice}',[HomeController::class,'stripe']);  
    Route::post('stripe/{totalprice}',[HomeController::class,'stripePost'])->name('stripe.post');  
    route::get('/show_order',[HomeController::class,'show_order']);  
    route::get('/cancel_order/{id}',[HomeController::class,'cancel_order']);  
    route::get('/print_pdf/{id}',[HomeController::class,'print_pdf']);  
    route::get('/profile',[HomeController::class,'profile']);  
    route::get('/profile_edit/{id}',[HomeController::class,'profile_edit']);  
    route::post('/profile_update/{id}',[HomeController::class,'profile_update']);  
  
});  
  
Route :: get('/userlogout', function(){  
    if (session()->has('user')) {  
        session()->pull('user');  
        return redirect('userlogin');  
    }  
});
```

4. Profile Edit :

```

public function profile_edit()
{
    $username = session('user');
    $user = UserSignup::where('username', $username)->first();
    return view('home.profile_edit',compact('user'));
}

public function profile_update(Request $request,$id)
{
    $username = session('user');
    $user = UserSignup::where('username', $username)->first();
    $user->username = $request->username;
    $user->name = $request->name;

    $user->phone = $request->phone;
    $user->address = $request->address;
    $user->save();
    return redirect('profile');
}

```



The screenshot shows a profile edit form. It includes fields for Username, Email Address (readonly), Phone Number, Name, and Address. The form is styled using Bootstrap's responsive design, with columns for input fields.

The page allows users to edit and update their profile information. It includes a form with fields for username, email address (readonly), phone number, name, and address. Users can input their updated information and submit the form to update their account details. The template features responsive design, utilizes Bootstrap for styling, and includes header

navigation from the main template. The provided code allows users to interactively update their profile information.

5. Payment gateway:

```

public function stripe($totalprice)
{
    return view('home.stripe', compact('totalprice'));
}
public function stripePost(Request $request,$totalprice)
{
    Stripe\Stripe::setApiKey(env('STRIPE_SECRET'));

    Stripe\Charge::create([
        "amount" => $totalprice * 100,
        "currency" => "usd",
        "source" => $request->stripeToken,
        "description" => "Test payment from itsolutionstuff.com."
    ]);

    $store = session('user'); // Assuming $store holds the username
    $data = cart::where('username','=', $store)->get();

    foreach($data as $data)
    {
        $order=new order;
        $order->name=$data->name;
        $order->email=$data->email;
        $order->phone=$data->phone;
        $order->address=$data->address;
        $order->user_id=$data->user_id;
        $order->username=$data->username;
        $order->product_title=$data->product_title;
        $order->price=$data->price;
        $order->quantity=$data->quantity;
        $order->day=$data->day;
        $order->image=$data->image;
        $order->product_id=$data->Product_id;
    }
}

```

```

public function cash_order()
{
    $store = session('user'); // Assuming $store holds the username
    $data = cart::where('username', '=', $store)->get();

    foreach($data as $data)
    {
        $order=new order;
        $order->name=$data->name;
        $order->email=$data->email;
        $order->phone=$data->phone;
        $order->address=$data->address;
        $order->user_id=$data->user_id;
        $order->username=$data->username;
        $order->product title=$data->product_title;
        $order->price=$data->price;
        $order->quantity=$data->quantity;
        $order->day=$data->day;
        $order->image=$data->image;
        $order->product id=$data->Product_id;
        $order->vendor_name=$data->vendor_name;

        $order->payment_status='cash on delivery';
        $order->delivery_status='processing';
        $order->save();

        $cart_id=$data->id;
        $cart=cart::find($cart_id);
        $cart->delete();
    }
}

var $form = $(".require-validation");

$('form.require-validation').bind('submit', function(e) {
    var $form = $(".require-validation"),
        inputSelector = ['input[type=email]', 'input[type=password]',
                        'input[type=text]', 'input[type=file]',
                        'textarea'].join(', '),
        $inputs = $form.find('.required').find(inputSelector),
        $errorMessage = $form.find('div.error'),
        valid = true;
    $errorMessage.addClass('hide');

    $('.has-error').removeClass('has-error');
    $inputs.each(function(i, el) {
        var $input = $(el);
        if ($input.val() === '') {
            $input.parent().addClass('has-error');
            $errorMessage.removeClass('hide');
            e.preventDefault();
        }
    });
    if (!$form.data('cc-on-file')) {
        e.preventDefault();
        Stripe.setPublishableKey($form.data('stripe-publishable-key'));
        Stripe.createToken({
            number: $('.card-number').val(),
            cvc: $('.card-cvc').val(),
            exp_month: $('.card-expiry-month').val(),
            exp_year: $('.card-expiry-year').val()
        }, stripeResponseHandler);
    }
});

```

This HTML template creates a payment page for processing credit card payments using the Stripe API. The page allows users to enter their payment details, including card number, CVV, expiration date, and name on the card. It calculates and displays the total amount to be paid. Upon successful payment, a success message is displayed, and if there are any errors during the payment process, an error message is shown.

The template is integrated with the Bootstrap CSS framework and includes jQuery and Stripe's JavaScript library for handling the payment process. The form data is submitted to the server for processing the payment using the Stripe API. It features responsive design and includes header navigation from the main template. The template enables users to securely make payments using their credit cards through Stripe's payment gateway.

6. Print PDF:

```

public function print_pdf($id)
{
    $order=order::find($id);
    $pdf=PDF::loadView('home.pdf',compact('order'));
    return $pdf->download('order_details.pdf');
}

<h1>Order Details</h1>

Customer Name :<h3>{$order->name}</h3>
Customer Email :<h3>{$order->email}</h3>
Customer Phone :<h3>{$order->phone}</h3>
Customer Address :<h3>{$order->address}</h3>
@php
$daysToAdd = $order->day; // Change this to the number of days you want to add
$newDate = $order->created_at->addDays($daysToAdd);
@endphp
You, 2 weeks ago • Order details ...
Product Name :<h3>{$order->product_title}</h3>
Product Price :<h3>{$order->price}</h3>
Product Quantity :<h3>{$order->quantity}</h3>
Payment Status :<h3>{$order->payment_status}</h3>
Return date : <h3>{$newDate->format('Y-m-d')}</h3>

```

This HTML template generates an order details PDF document. It displays information related to a customer's order, including their name, email, phone number, and address. Additionally, it provides details about the purchased product, such as its name, price, and quantity. The template also includes the payment status of the order and the calculated return date based on the provided number of days.

The template is designed to be used for generating order-related PDFs and is intended to be dynamically populated with data from the server. It does not include any styling or formatting beyond the basic structure required to present the order information in a clear and organized manner.

Home:

1. Homepage:

```
class HomeController extends Controller
{
    public function index()
    {
        $product=products::paginate(2);
        return view('home.userpage',compact('product'));
    }

    public function index2()
    {
        $product=products::paginate(3);
        return view('home.guestuser',compact('product'));
    }
}
```

The 'HomeController' class manages different views for users and guest users on a website. In the 'index()' method, it retrieves a paginated list of products (2 products per page) from a database table named "products" and passes this data to the "home.userpage" view for rendering. This view is likely intended for authenticated users and displays the products.

In the 'index2()' method, it retrieves another paginated list of products (3 products per page) from the same "products" table and sends this data to the "home.guestuser" view. This view is probably designed for guest users and presents the products to them.

Both methods utilize the 'products' model to fetch data from the database and make it available to the corresponding views, enabling the display of products based on the user's status (authenticated or guest) while providing a paginated browsing experience.

2. Header:

```

        </li>
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">About</a>
            <ul class="dropdown-menu">
                <li><a href="about.html">About</a></li>
                <li><a href="testimonial.html">Testimonial</a></li>
            </ul>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{url('product_show')}}">Products</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="blog_list.html">Blog</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="contact.html">Contact</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{url('show_order')}}">Order</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{url('profile')}}">Profile</a>
        </li>
        <form class="form-inline">
            <button class="btn my-2 my-sm-0 nav_search-btn" type="submit">
                <i class="fa fa-search" aria-hidden="true"></i>
            </button>
        </form>

        <li class="nav-item">
            <a class="btn btn-primary" id="logincss" href="{{url('/userlogout')}}">Logout</a>
        </li>
    
```

The backend development for the given code involves creating and managing the functionality of a navigation bar in an e-commerce website. The code utilizes a Laravel framework to dynamically generate navigation links and dropdown menus based on user roles and actions. The backend logic includes retrieving and displaying links for pages such as Home, About, Testimonial, Products, Blog, Contact, Orders, Profile, and Cart. It also handles user authentication, including user login, user registration, vendor login, and vendor registration, with corresponding routes and redirections. The backend development ensures proper routing, user authentication, session management, and data retrieval to populate the navigation bar and provide a seamless user experience. Additionally, the backend supports canceling orders, printing PDFs, and handling logout actions. The code showcases efficient routing and data management to create a comprehensive and interactive navigation menu.

3. Header for guest user :

```

        <a class="nav-link" href="{{url('/')}}">Home <span class="sr-only" style="font-weight: bold;">(current)</span></a>
    </li>
    <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="true"> <span class="sr-only" style="font-weight: bold;">About</span>
        <ul class="dropdown-menu">
            <li><a href="about.html">About</a></li>
            <li><a href="testimonial.html">Testimonial</a></li>
        </ul>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="{{url('product_show')}}">Products</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="blog_list.html">Blog</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="contact.html">Contact</a>
    </li>
    <form class="form-inline">
        <button class="btn my-2 my-sm-0 nav-search-btn" type="submit">
            <i class="fa fa-search" aria-hidden="true"></i>
        </button>
    </form>

    <li class="nav-item">
        <a class="btn btn-primary" id="logincss" href="{{url('/userlogin')}}">Login</a>
    </li>
    <li class="nav-item">
        <a class="btn btn-primary" id="logincss" href="{{url('/userssignup')}}">Register</a>
    </li>

```

The backend development involves implementing the server-side functionality and data management for the website. This includes handling user authentication, managing product data, processing orders, and generating dynamic content. User authentication is implemented to securely manage user logins and registrations, enabling access to personalized features. The product management system allows administrators to add, edit, and display products on the website. Orders are processed and stored in a database, with details like product information, quantities, and payment status recorded. Additionally, dynamic content generation is achieved through server-side scripting, enabling the display of user-specific data and real-time updates. The backend development ensures smooth data flow, secure interactions, and efficient management of website operations, contributing to a seamless user experience.

4. Product

```

@foreach($product as $products)

    <div class="col-sm-8 col-md-6 col-lg-6">
        <div class="box">
            <div class="option_container">
                <div class="options">
                    <a href="{{url('product_details',$products->product_id)}}" class="option1">
                        Product Details
                    </a>
                    <a class="option2">
                        {{ $products->vendor_name }}
                    </a>
                <form action="{{url('add_cart',$products->product_id)}}" method="Post">
                    @csrf
                    <div class="row">
                        <div class="col-md-4">
                            <label for="quantity">Quantity:</label>
                            <input type="number" name="quantity" value="1" min="1" style="width: 100px">
                        </div>
                        <div class="col-md-4">
                            <label for="days">Days:</label>
                            <input type="number" name="days" value="{{ $products->days }}" min="{{ $products->days }}" style="width: 100px">
                        </div>
                        <div class="col-md-4">
                            <input type="submit" value="Add to Cart">
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
<div class="img_box">
    
</div>
<div class="detail-box">
    <h5>
        {{$products->product_title}}
    </h5>
    @if($products->discounted_price!=null)

        <h6 style="color: red">
            Discount price
        <br>
        {{$products->discounted_price}} -/BDT
        </h6>

        <h6 style="text-decoration: line-through; color: blue">
            Per Day Rent
        <br>
        {{$products->price}} -/BDT
        </h6>
    @else

        <h6 style="color: blue">
            Per Day Rent
        <br>
        {{$products->price}} -/BDT
        </h6>
    @endif
</div>

```



```
You, 3 weeks ago • Homepage
@endif

        </div>
    </div>
</div>
@endforeach

<span style="padding-top: 20px;">

{!!$product->withQueryString()->links('pagination::bootstrap-5')!!}
</span>
</div>
</section>
```

The backend development for the product section involves handling product data and user interactions. The server fetches product information from a database and dynamically generates product listings on the website. The search functionality is implemented using a form that sends search queries to the server, enabling users to search for specific products. Each product is displayed with its details, including title, vendor name, image, price, and discount (if applicable). Users can select quantities and rental days for products and add them to the cart. The "Add to Cart" functionality is achieved through a form submission that updates the user's cart in the backend. Pagination is implemented to display a limited number of products per page and allow users to navigate through the product list. The backend processes these user interactions, retrieves and updates product data, and manages cart items, enhancing the user experience and providing a seamless product browsing and selection process.

5. Product-Details:

```

![Product image]({{ asset('added_products/' . $product->image) }})

```

{\$product->product_title}

Discount price

 {\$product->discounted_price} -/BDT

Per day Rent

 {\$product->price} -/BDT

Product Details : {\$product->product_description}

Available Quantity : {\$product->quantity}

Vendor : {\$product->vendor_name}

Quantity :

Days :

Add to Cart

```

<form action="{{url('add_cart',$product->product_id )}}" method="Post">
    @csrf
    <div class="row">
        <div class="col-md-4">
            <label for="quantity">Quantity:</label>
            <input type="number" name="quantity" value="1" min="1" style="width: 100px;">
        </div>
        <div class="col-md-4">
            <label for="days">Days:</label>
            <input type="number" name="days" value="5" min="5" style="width: 100px;">
        </div>
        <div class="col-md-4">
            <input type="submit" value="Add to Cart">
        </div>
    </div>
</form>

```

```
public function product_details($product_id)
{
    $product=products::find($product_id);

    return view('home.product_details',compact('product'));
}
```

The backend development for this product detail page involves retrieving and displaying dynamic product information from a database based on the product ID. It handles logic to show discounted prices if available, processes user input for adding items to the cart, and manages cart functionality. Additionally, it includes common header and footer templates for a consistent layout. The backend ensures data accuracy, security, and smooth user interactions for an engaging and functional shopping experience.

6. Footer :

```

<div class="col-md-6">
    <div class="widget_menu">
        <h3>Account</h3>
        <ul>
            <li><a href="#">Account</a></li>
            <li><a href="#">Checkout</a></li>
            <li><a href="#">Login</a></li>
            <li><a href="#">Register</a></li>
            <li><a href="#">Shopping</a></li>
            <li><a href="#">Widget</a></li>
        </ul>
    </div>
</div>
</div>
</div>
<div class="col-md-5">
    <div class="widget_menu">
        <h3>Newsletter</h3>
        <div class="information_f">
            <p>Subscribe by our newsletter and get update protidin.</p>
        </div>
        <div class="form_sub">
            <form>
                <fieldset>
                    <div class="field">
                        <input type="email" placeholder="Enter Your Mail" name="email" />
                        <input type="submit" value="Subscribe" />
                    </div>
            </form>
        </div>
    </div>
</div>

```

The backend part of this footer involves rendering dynamic content such as the company's address, telephone number, and email address. It also generates the menu items for navigation, both for the main menu and the account-related links. Additionally, the backend handles the newsletter subscription functionality, capturing user email input and managing the subscription process. Overall, the backend ensures that the footer's information and functionality are dynamically generated and responsive to user interactions.

Technology (Framework, Languages)

Technology Used in Implementation

The development of Wearwise, the Online Clothing Rental Service Website, leveraged a comprehensive technology stack, with Laravel as the core framework, to build a robust and dynamic platform tailored to meet user needs.

Framework:

Laravel Framework: Laravel, a powerful and popular PHP web application framework, formed the foundation of the project. It provided a structured and efficient environment for developing feature-rich web applications. Leveraging Laravel's elegant syntax, built-in security features, and extensive ecosystem, Wearwise was able to deliver a scalable and maintainable solution.

Client-Side Technologies:

HTML (Hyper Text Markup Language): HTML was utilized to create the structure of web pages, defining content elements and ensuring semantic consistency across the site.

CSS (Cascading Style Sheets): CSS3 empowered the platform's visual presentation, enabling responsive design, layouts, and consistent styling.

Bootstrap (Front-end Framework): Bootstrap streamlined front-end development, offering pre-designed components, grid systems, and responsive features that ensured a seamless user experience on various devices.

Server-Side Technologies:

PHP: As a server-side scripting language, PHP powered dynamic content generation, data interaction, and database management, ensuring a robust back-end for the website.

SQL (Structured Query Language): SQL queries, managed through Laravel's Eloquent ORM, handled data transactions, supporting essential functions like user profiles, product information, and order management.

Implementation Tools:

Visual Studio Code (VS Code): VS Code provided a feature-rich source code editor with extensions and version control integration, enhancing developer productivity and code quality.

XAMPP (Cross-Platform Web Server Solution Stack): XAMPP created a local development environment, bundling components such as the Apache HTTP Server, MariaDB database, PHP interpreter, and Perl, facilitating testing and deployment.

Apache HTTP Server: Apache served as the web server software, handling HTTP requests and hosting the website, providing a reliable foundation for the platform.

MySQL (Relational Database Management System): MySQL efficiently manages structured data, ensuring data integrity and supporting essential features like stored procedures and query optimization.

The combination of Laravel as the core framework, alongside the aforementioned technologies and tools, enabled Wearwise to successfully implement a sophisticated online platform, delivering a user-friendly experience while leveraging the power of modern web development techniques. The adoption of Laravel, known for its flexibility and developer-friendly features, played a pivotal role in achieving the project's goals.

Github Repository

Link: <https://github.com/shihabmuhtasim/wearwise> (Accessible to our lab faculties only)

Individual Contribution

ID	Name	Contribution
21301610	Shihab Muhtasim	<p>Module 1: Admin Panel</p> <ul style="list-style-type: none"> 1. Login , registration & logout 2. session management on all pages 3. Add product 4. View product , Edit product & Delete product 5. Add, Edit, Delete & View Category 6. Add, Edit, Delete & View Apparel 7. Admin sidebar, header 8. Dashboard features design & show data from database 9. View orders, delete orders <p>Module 2: Home /Shop</p> <ul style="list-style-type: none"> 1. User login, logout 2. User register 3. Session management on all pages <p>Module 4: Vendor management:</p> <ul style="list-style-type: none"> 1. Vendor register - (approval from admin) 2. Vendor login, logout & session management 3. Post product for approval 4. View pending products, edit those, and delete those 5. View approved products, delete those 6. View orders of vendor

		<p>7. Vendor Dashboard features design & show data from database 8. sidebar, navbar</p> <p>Admin panel:</p> <p>9. View Vendor account requests— approve, delete. 10. View certified vendors & remove those vendors. 11. Delete products of removed vendors. 12. View Vendor product post requests & approve / reject- (approved products go to main products table)</p>
21301274	Nusaiba Alam	<p>Module 2: Home /Shop</p> <p>1. Homepage products display 2. Products page- description, price, days, vendor name display 3. Sort products by pagination 4. Search products by keywords 5. Cart- update, delete, continue shopping</p> <p>Module 3 User dashboard:</p> <p>1. View profile 2. Online Payment gateway using card implementation 3. order confirmation 4. Edit profile 5. View orders of user 6. Cart- checkout 7. Cancel order from order page 8. Download pdf of order details</p>
20301326	Sartaj Emon Prattoy	<p>Module 1: Admin Panel</p> <p>1. Admin panel view customer 2. Delete Customer</p>

		<p>3. View Order list</p> <p>Module 3: User Dashboard</p> <p>4. Product review: Comment</p> <p>5. Reply.</p>
--	--	--