

**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 |
|  | Sartaj |
|  |  |
|  |  |

**Table of Contents**

| **Section**  **No** | **Content** | **Page No** |
| --- | --- | --- |
| 1 | Introduction | 3 |
| 2 | Functional Requirements | 5 |
| 3 | User Manual |  |
| 4 | Frontend Development |  |
| 5 | Backend Development |  |
| 6 | Technology (Framework, Languages) |  |
| 7 | Github Repo Link |  |
| 8 | Individual Contribution |  |

**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.

1. Logout & Session Management:

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

1. 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

1. 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.

1. 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.

1. 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.

1. 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.

1. Product pagination & Search:

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

1. 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.

1. Add review/reply/comment on product

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

1. 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.

1. Online Payment Gateway:

* Users should be able to make online payments securely.

1. 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.

1. Review:

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

1. Delete or remove cart items:

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

1. 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.

1. 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.

1. Order Management:

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

1. 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.

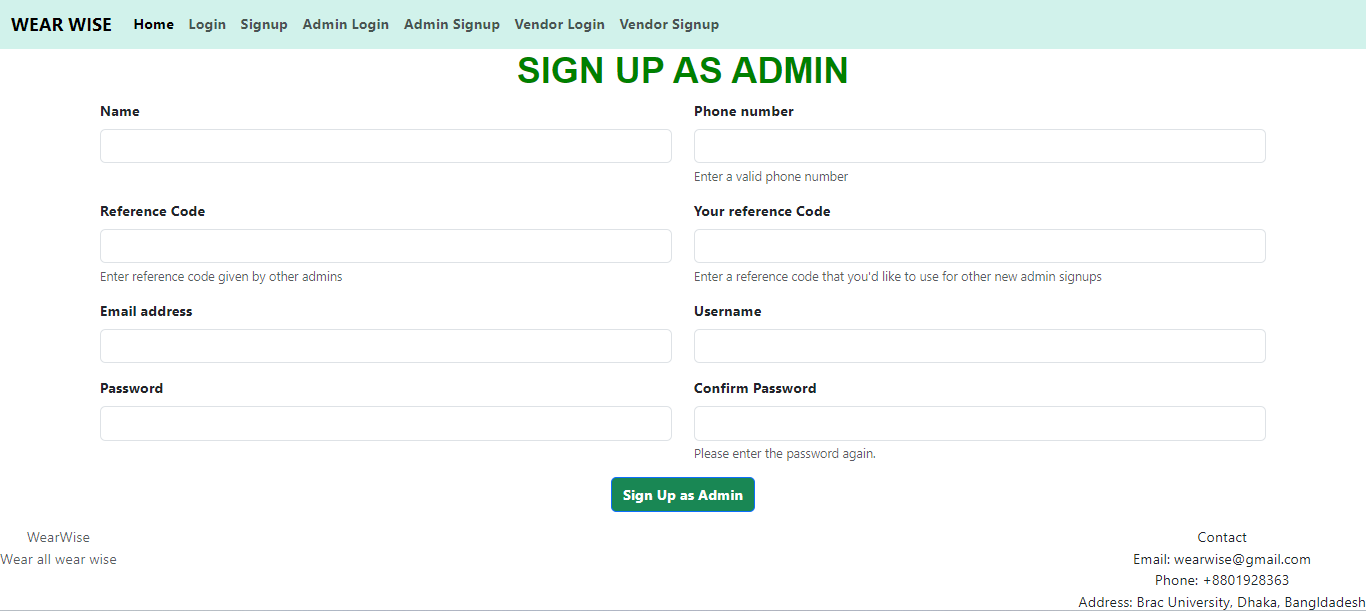
**User Manual**

Demo (Chapter 4: Page 18-29): <https://aquibazmain.github.io/documents/ReportShworolipi.pdf>

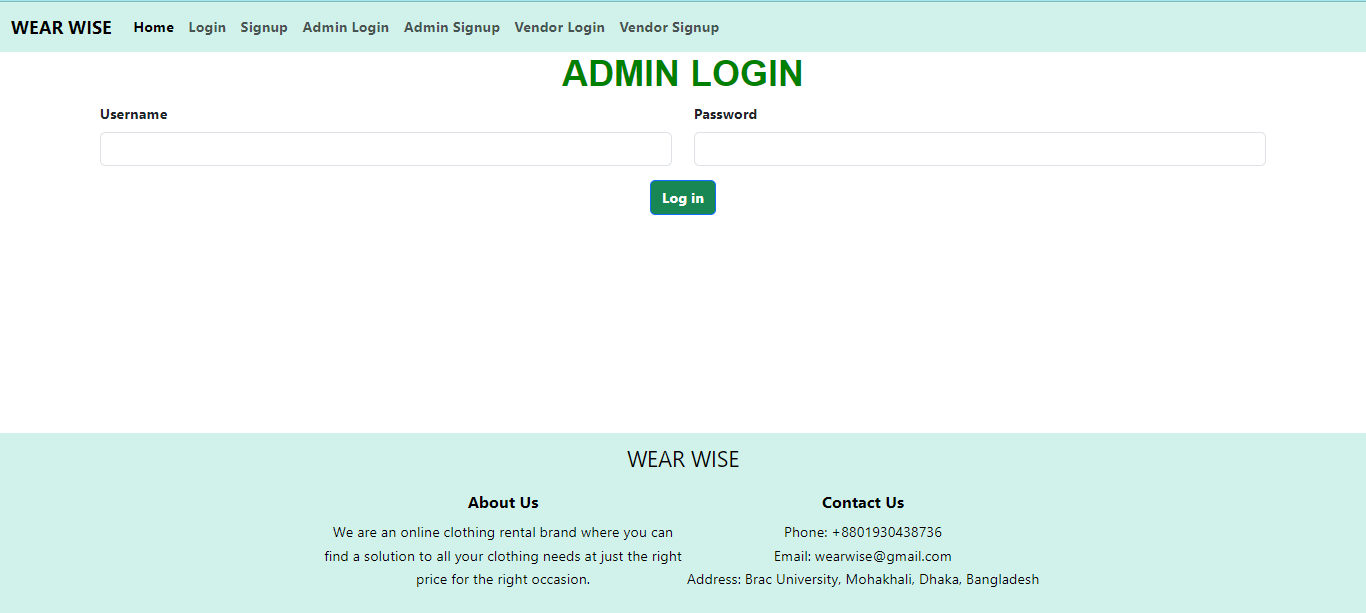
**Admin panel:**

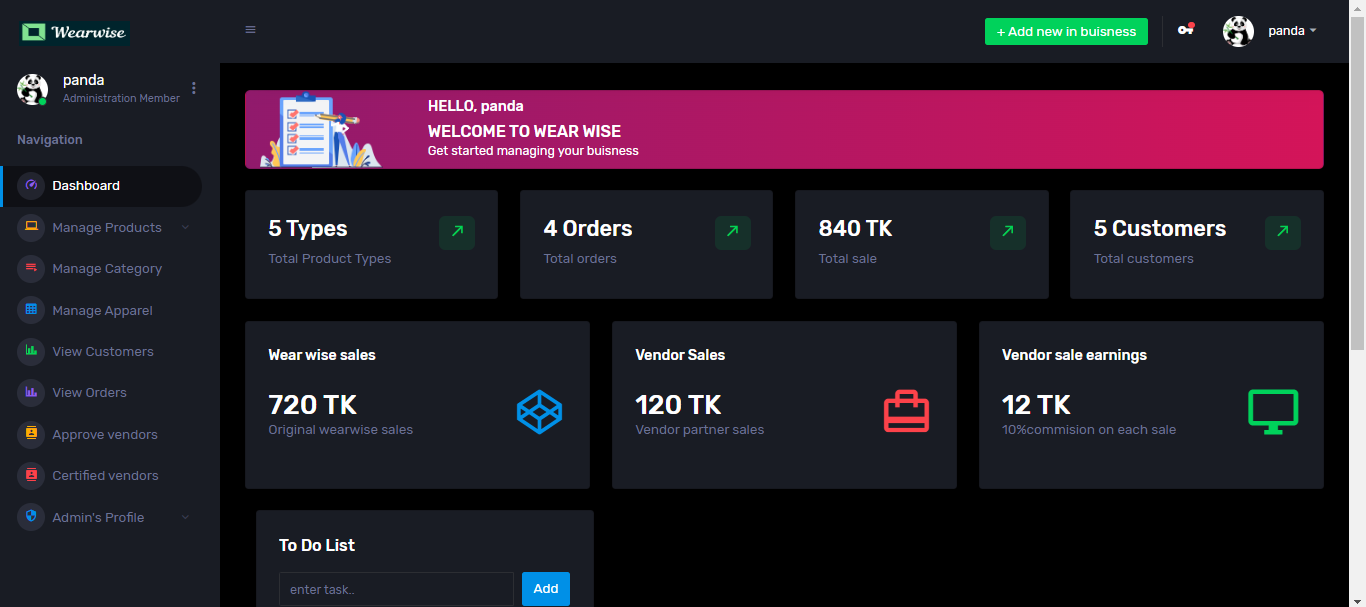
There are 11 major features for the admin panel. Those are, Login & Registration, logout & session management on all pages, product management, category management, apparel management,admin sidebar, header,dashboard features design, view orders,view Customers.

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

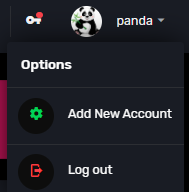
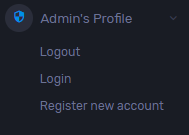


1. Login: An admin can login to see an admin dashboard with username and password if both the username and password matches in the database. Admin will see a dashboard with many necessary fields of information of the business to understand current status.

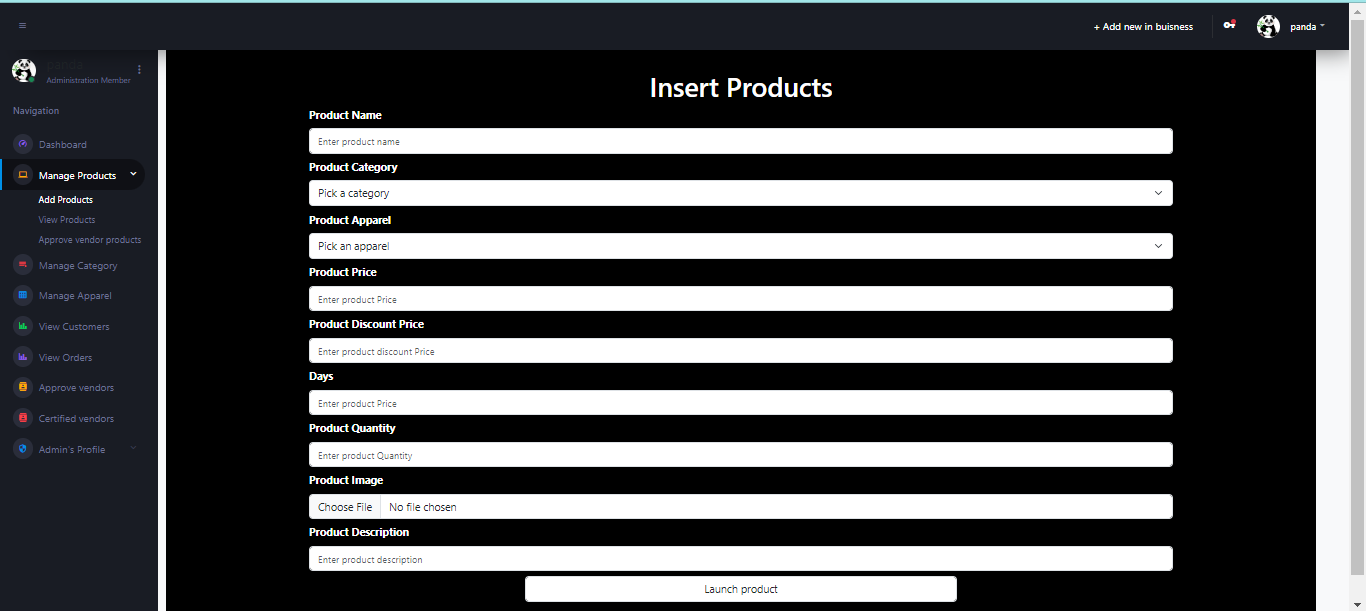




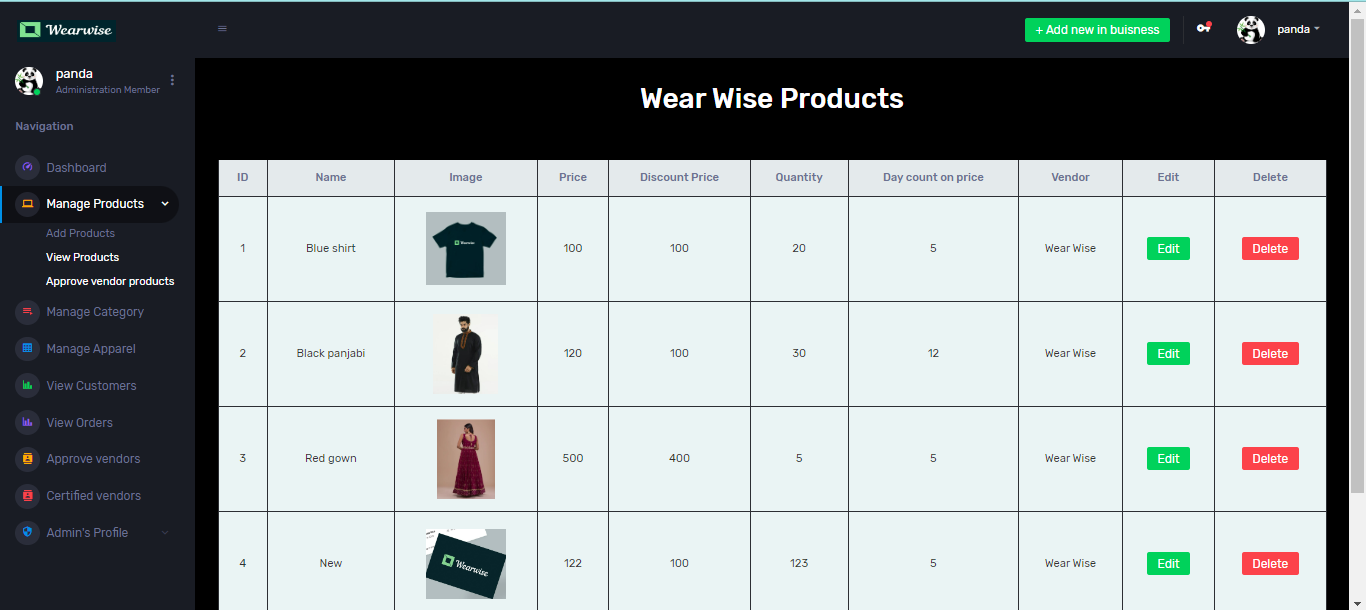
1. Logout: Admin can log out using the logout button that can be found in the admin dashboard.



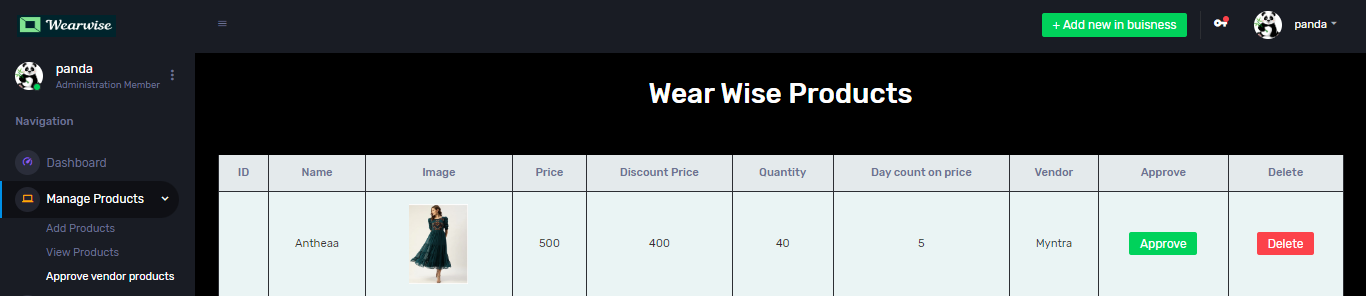
1. Add product: An admin can add products to the website under name Wear wise brand by going through Manage products-> Add products-> fill up the form and launch the product.



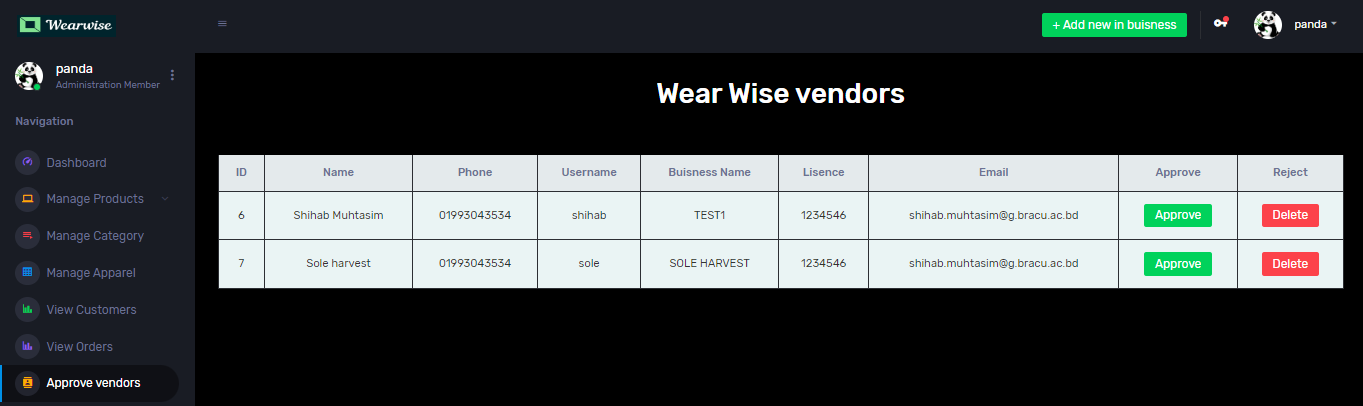
1. View product: This button takes admin to see all the products with option to edit any details or delete those.



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

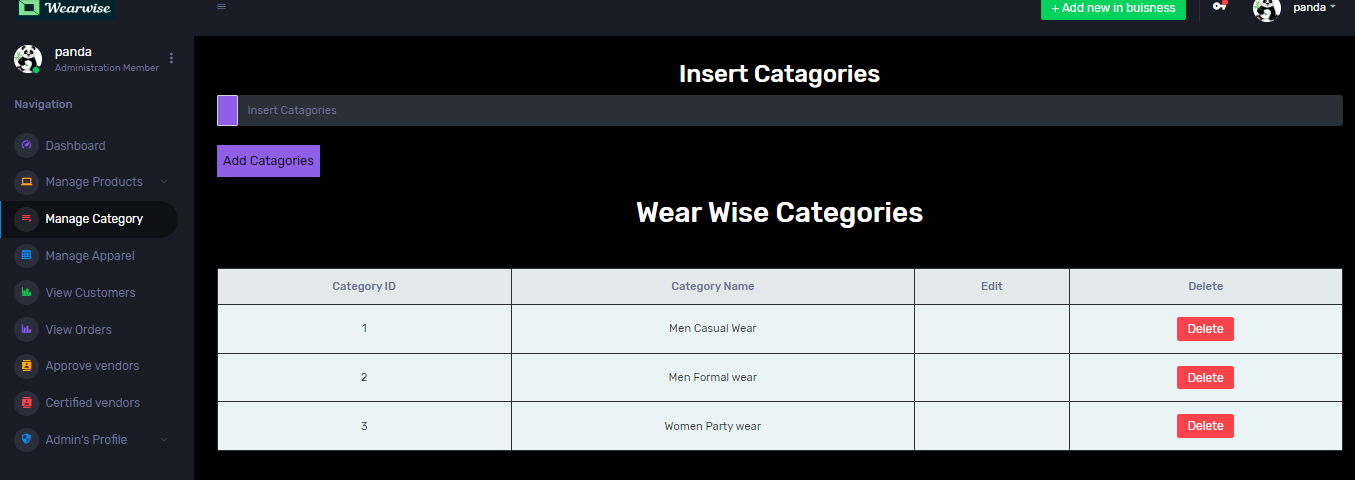
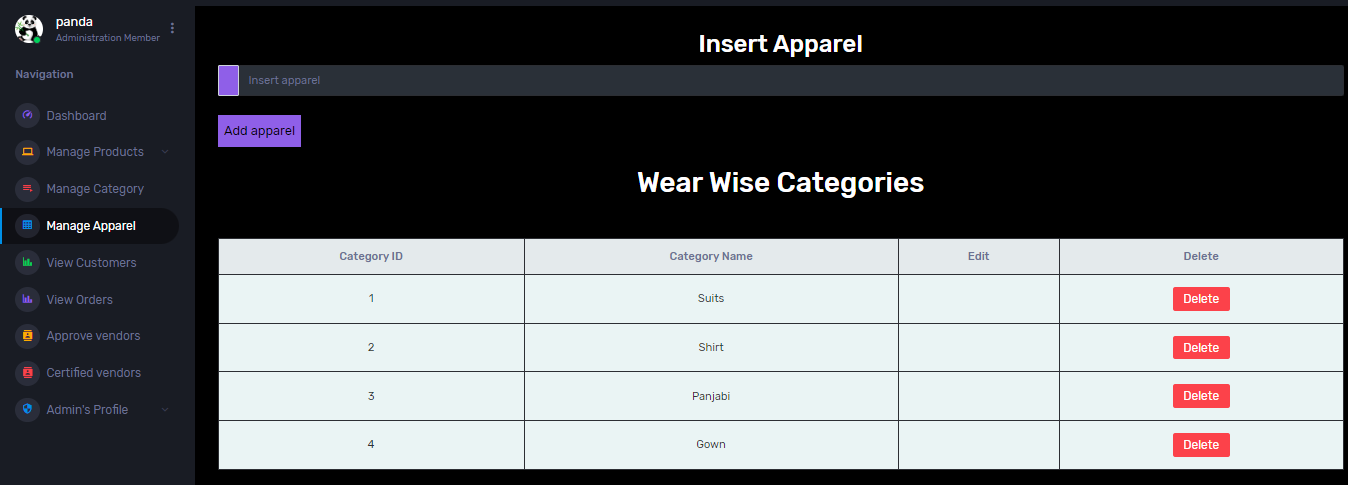
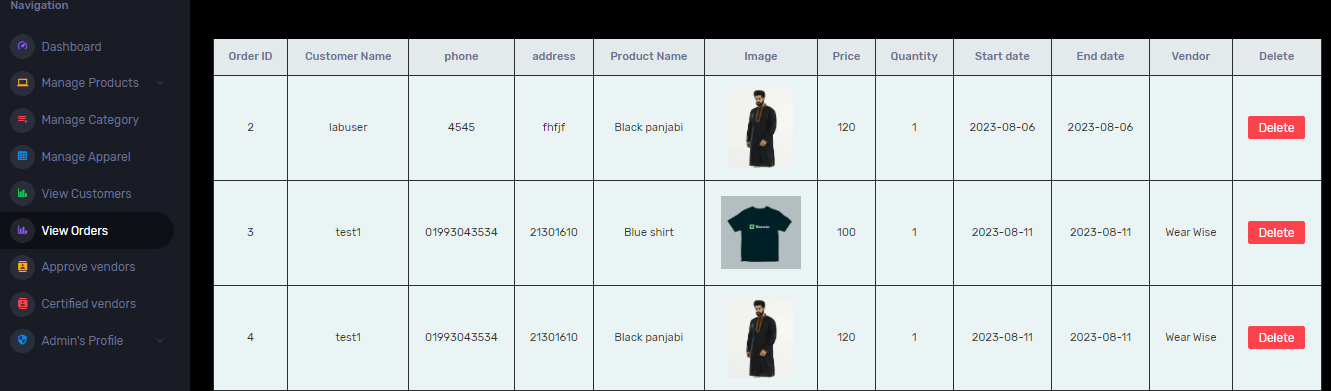
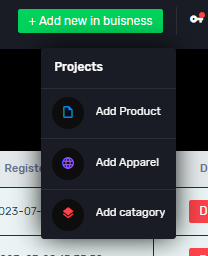
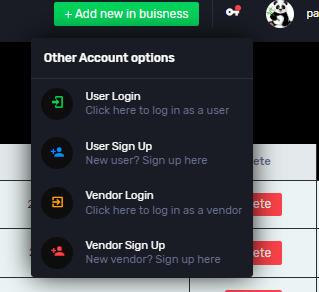


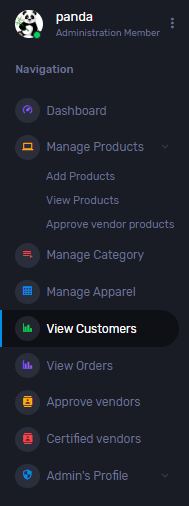
1. Approve vendors: Admin can see all requests for vendors and he can approve or delete those by selecting these buttons.



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



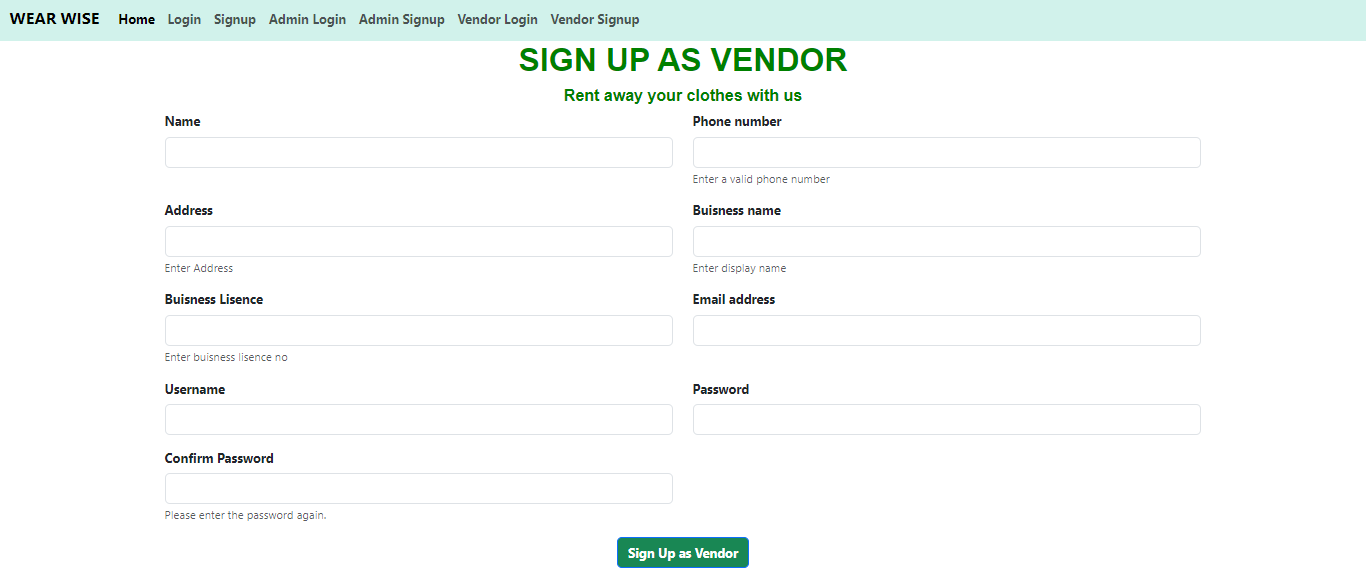
1. Manage category: This button shows all categories available along with options to add, edit or delete any.
2. Manage apparel: This button shows all apparels available along with options to add, edit or delete any.
3. View orders: Admin can view all orders and delete any by selecting the red delete button.
4. View customers: Admin can view all customers and delete any by selecting the red delete button.
5. 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.
6. 
7. Sidebar: Admin can find all options for the actions of admin in the sidebar.



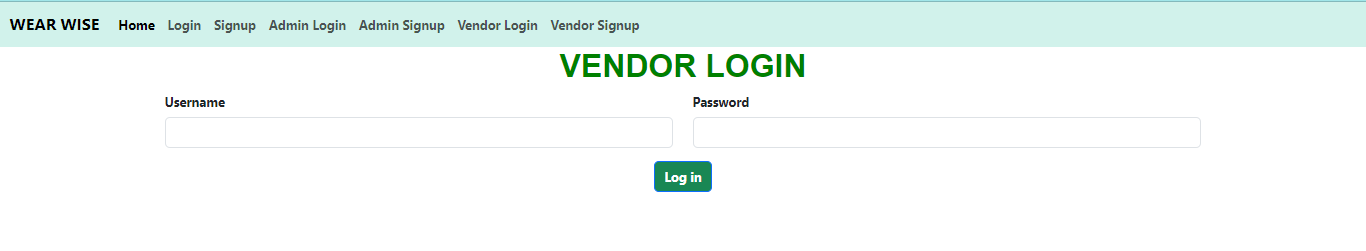
**Vendor Panel:**

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.

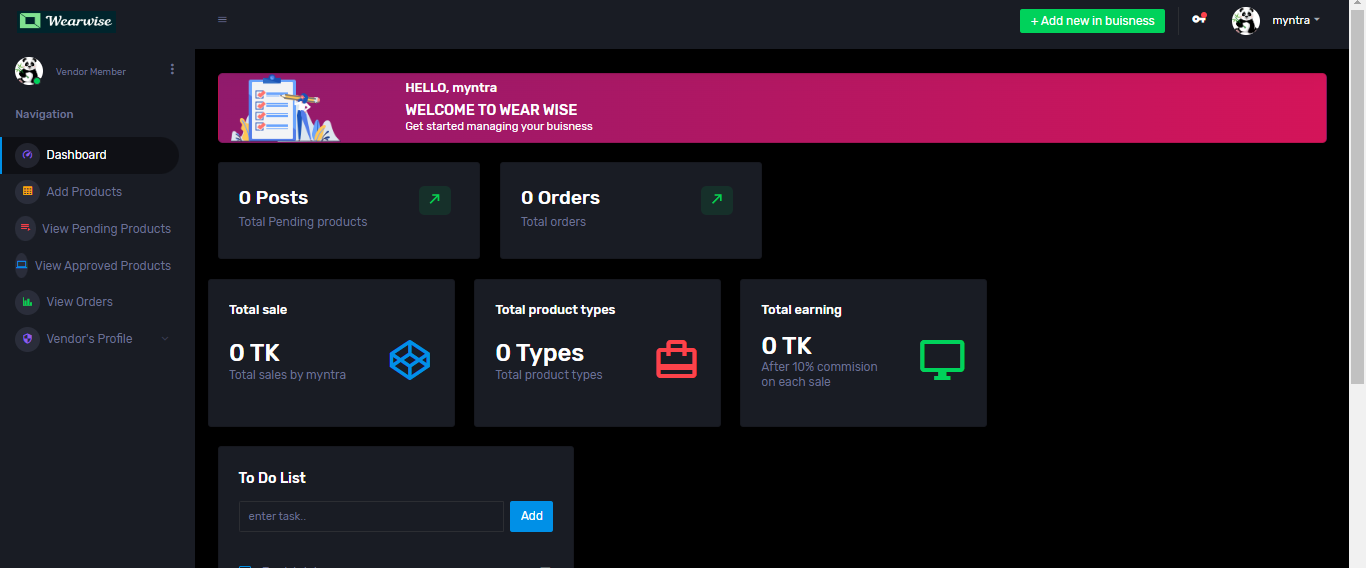
1. Registration: Prospective vendors can initiate the registration process by completing a form. The form requires a distinct username, and passwords that match. This ensures secure access to the vendor panel upon approval.



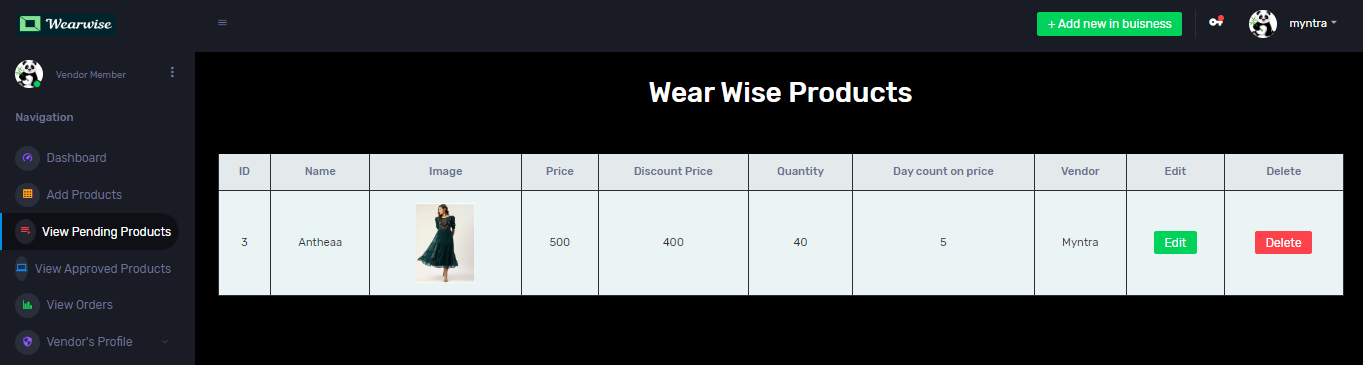
1. Login: Upon successful registration and approval, vendors can log in using their designated username and password. This will grant access to the Vendor Dashboard, where crucial insights into product management and order tracking are provided.



1. 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.



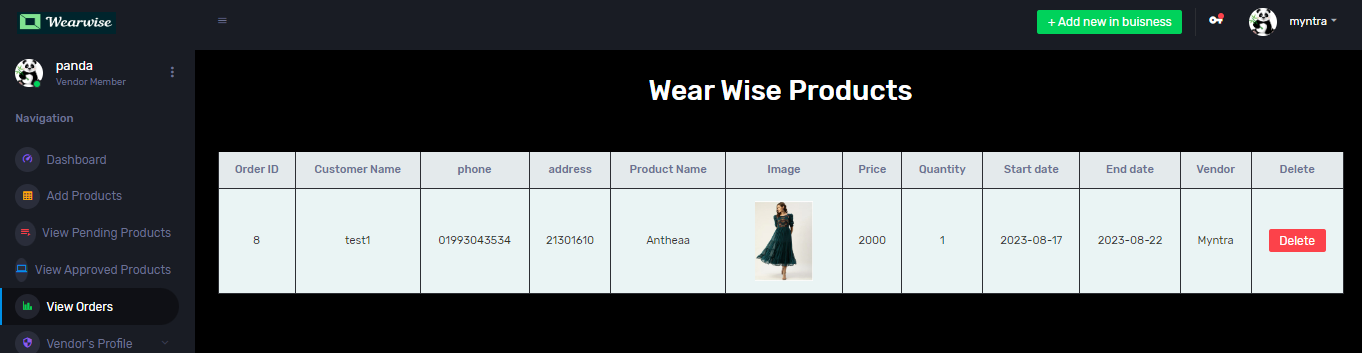
1. Logout: vendors can securely log out of the Vendor Panel by selecting the "Logout" option available within the dashboard.
2. Add Product: Vendors looking to list new products on Wearwise can utilize the "Add Product" feature. This involves populating a form with product-specific details. The submitted product will be subject to admin approval before becoming visible on the platform.
3. 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.



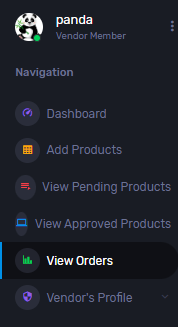
1. View Approved Products: The "Approved Products" section showcases products that have received admin approval. This overview allows vendors to access detailed product information and offers the option to remove listings as needed.



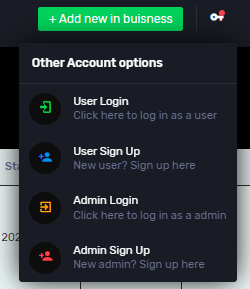
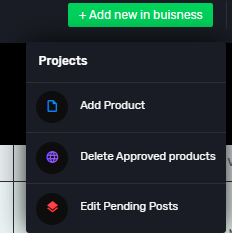
1. View Orders: Vendors can monitor orders that pertain to their specific product offerings within the "View Orders" section. This display provides essential order information, including customer details and order status.



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



1. 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 Page: When you navigate to the Products page, you'll find a comprehensive catalog of items available for rent. Each product listing includes essential details such as size, description, price, rental duration in days, and the name of the vendor offering the product.
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.
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.
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.
9. 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.

**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.
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.
3. Order Confirmation: Receive order confirmations for successful transactions, ensuring that your rental choices are accurately recorded.
4. Edit Profile: Need to update your profile information? The Edit Profile feature allows you to modify your details as necessary.
5. View Orders: Stay informed about your rental history by accessing the View Orders section. This provides an overview of your previous rental transactions.
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.
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.
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.

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**

Briefly discuss about Frontend Development and add Screenshots

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.

1. **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->catagory\_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;" class="form-label fw-bold">Product Description</label>

<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.

1. 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>

<img src="/added\_products/{{$product\_data->image}}" class="product-image">

</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.

1. **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>

<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.

1. 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>

<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.

1. **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">

<img class="img-xs rounded-circle" src="admin/assets/images/faces/face15.jpg" alt="">

<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>

1. 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>

<img src="/added\_products/{{$product\_data->image}}" class="product-image">

</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.

**Vendor Panel:**

1.Dashboard:

@php

$prodCount = $order\_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">{{$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>

<img src="/added\_products/{{$product\_data->image}}" class="product-image">

</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 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.

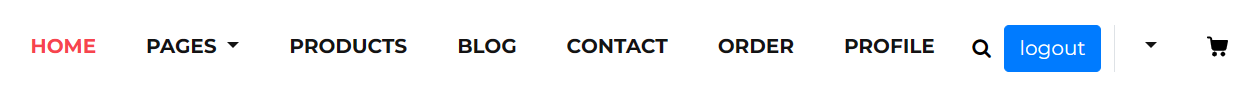
3. Rest of the pages are similar to that of Admin panels

* HTML and CSS: HTML and CSS
* JavaScript Interactivity: JavaScript Interactivity
* Responsive Design: Responsive Design
* Framework Usage (React): React Framework

N**eed to edit**

**Home**

1. Header for user: 

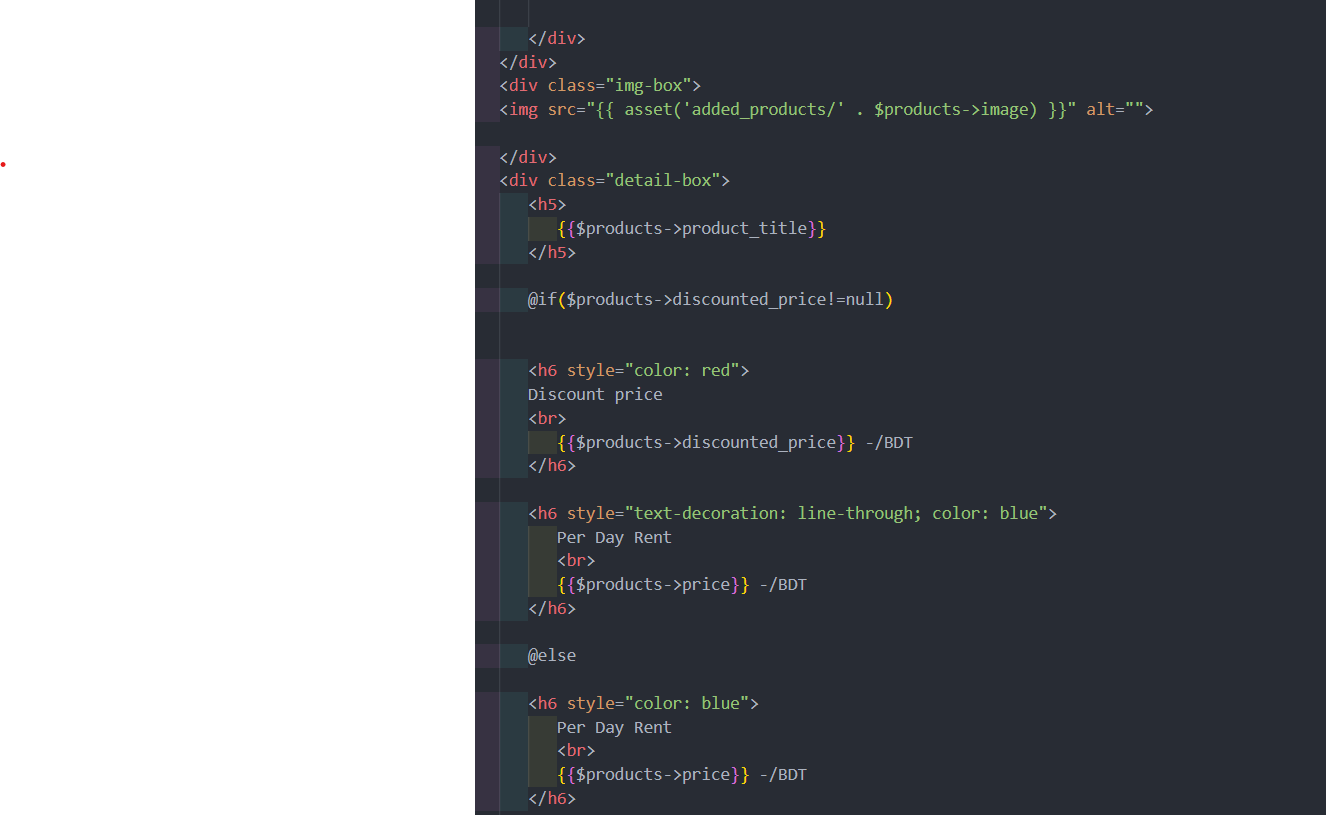


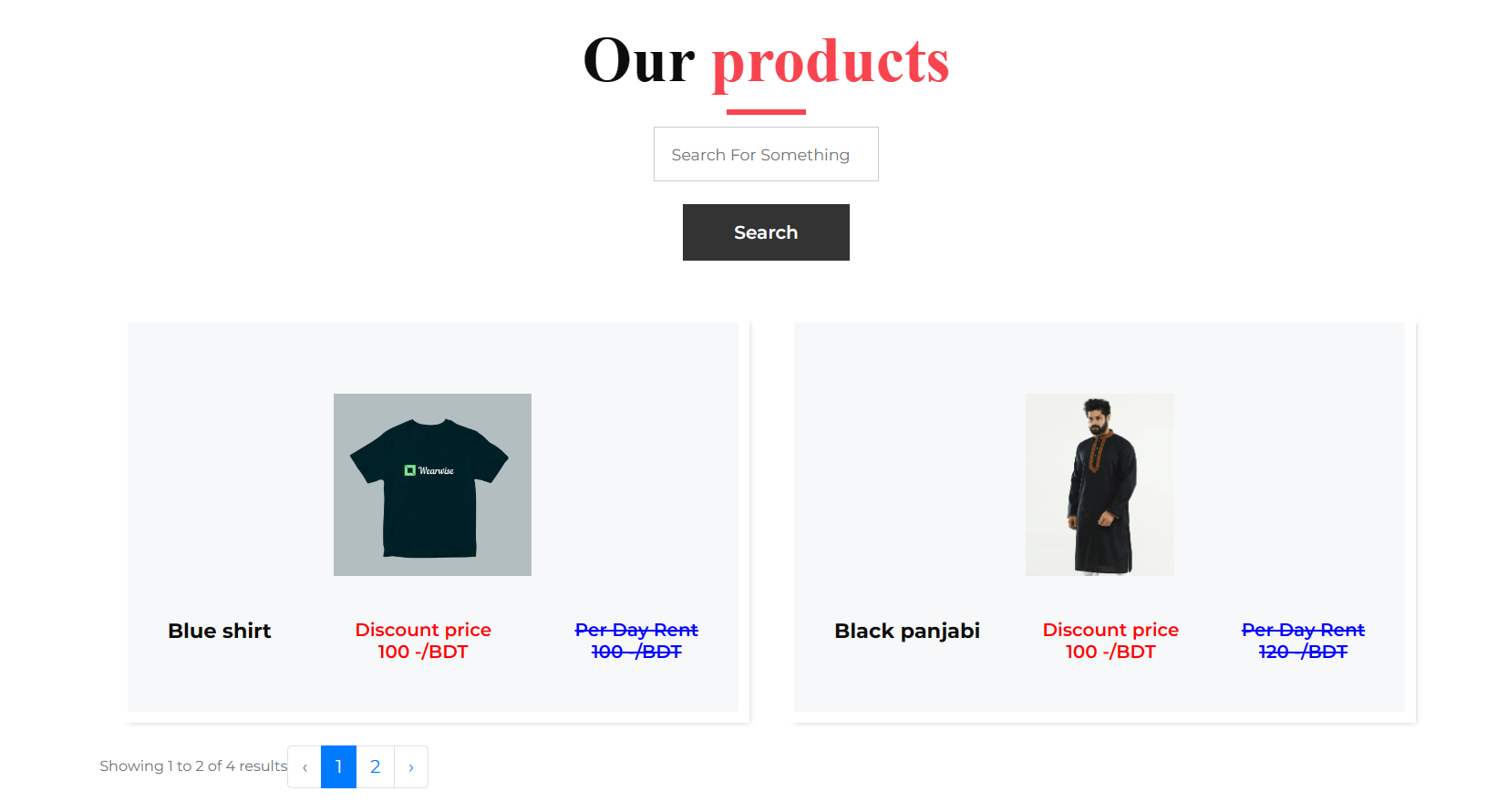
1. Header for guest user: 

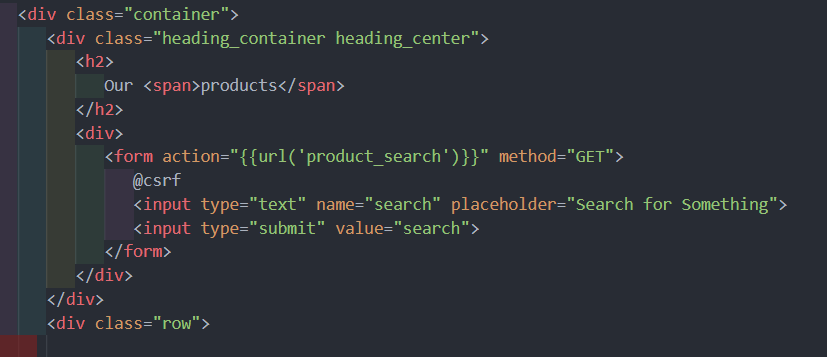


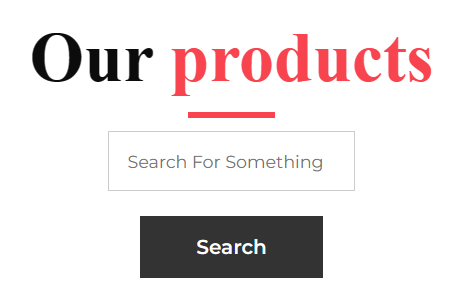
1. Slider:



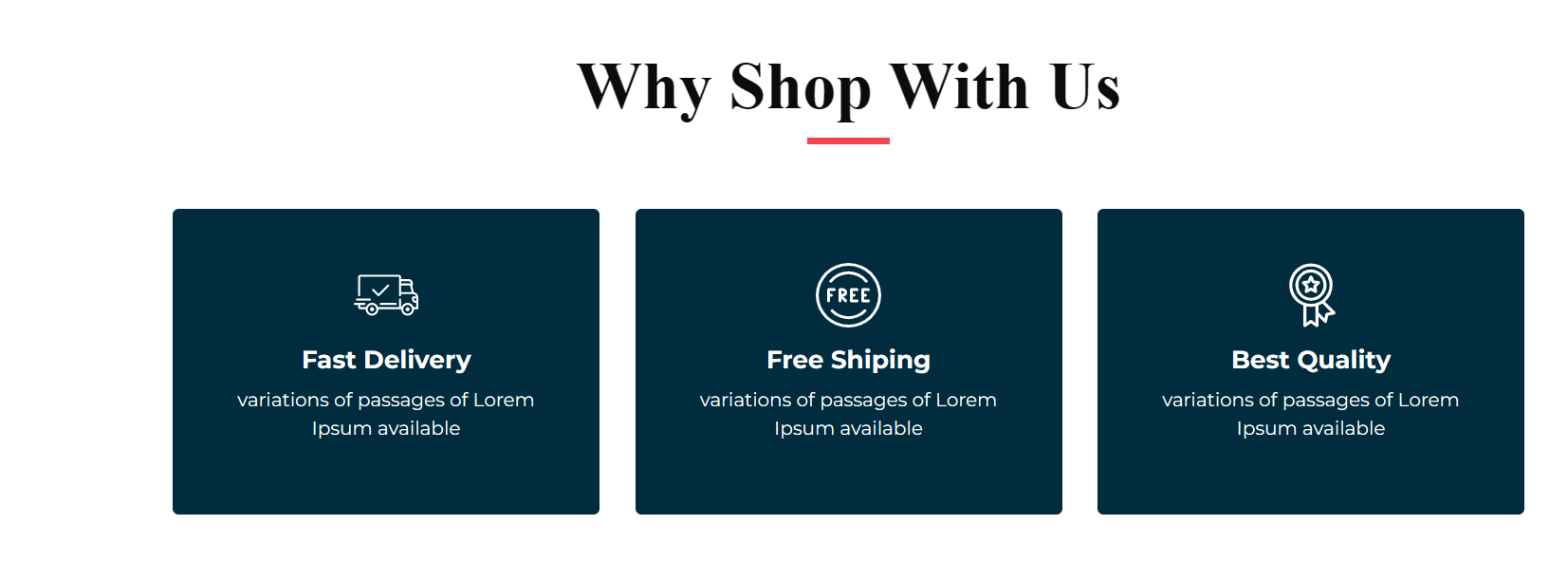
1. Product display with pagination : 

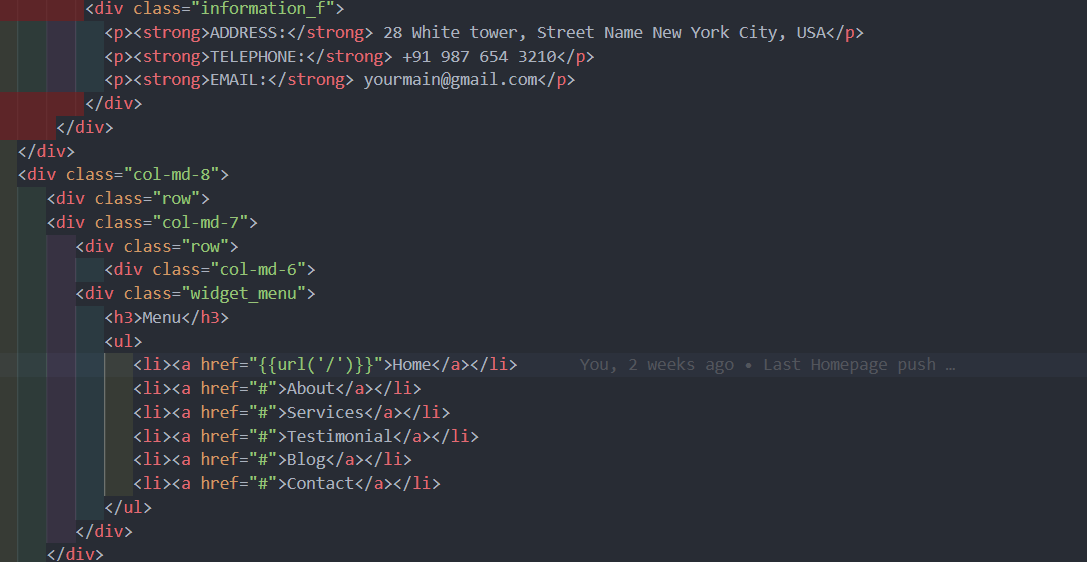


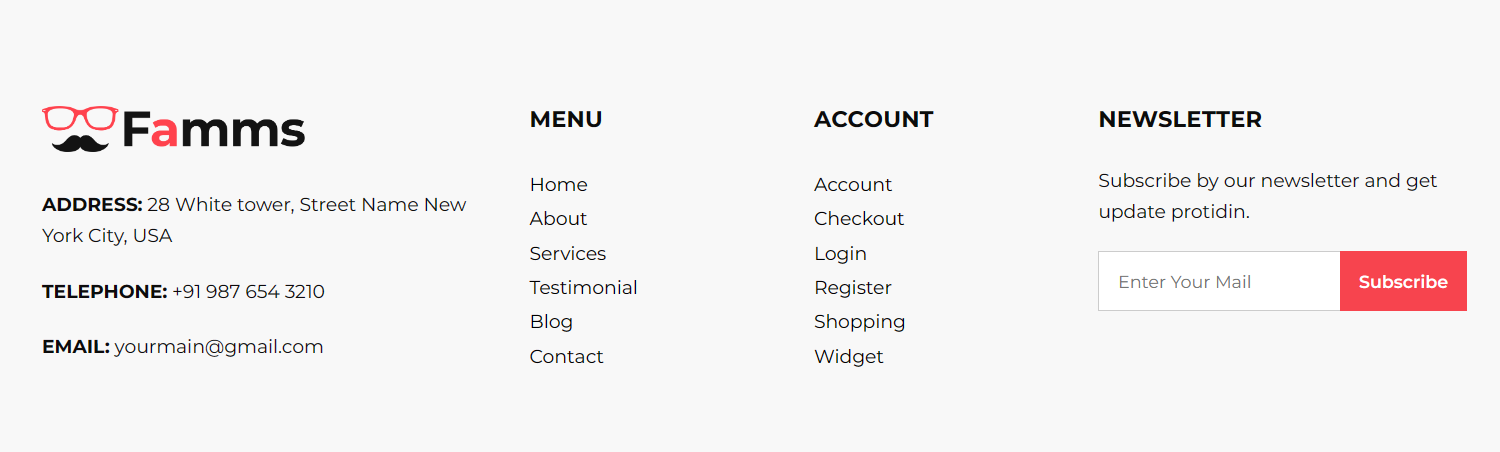
1. Search for products: 

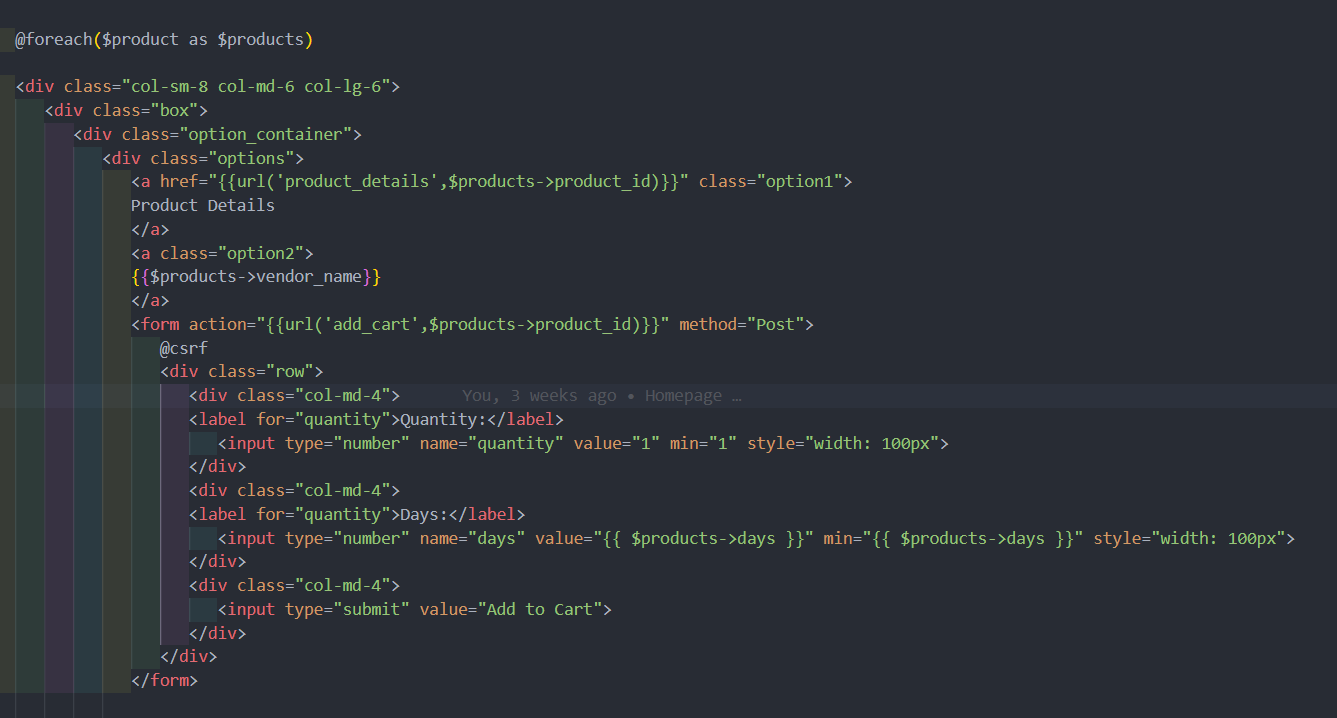


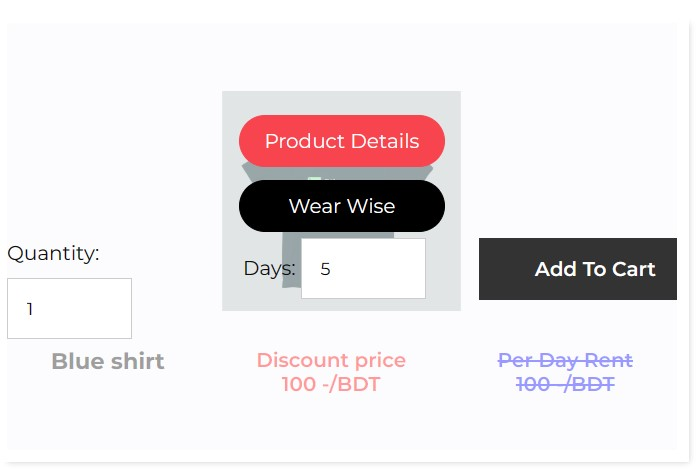
1. Why shop with us :

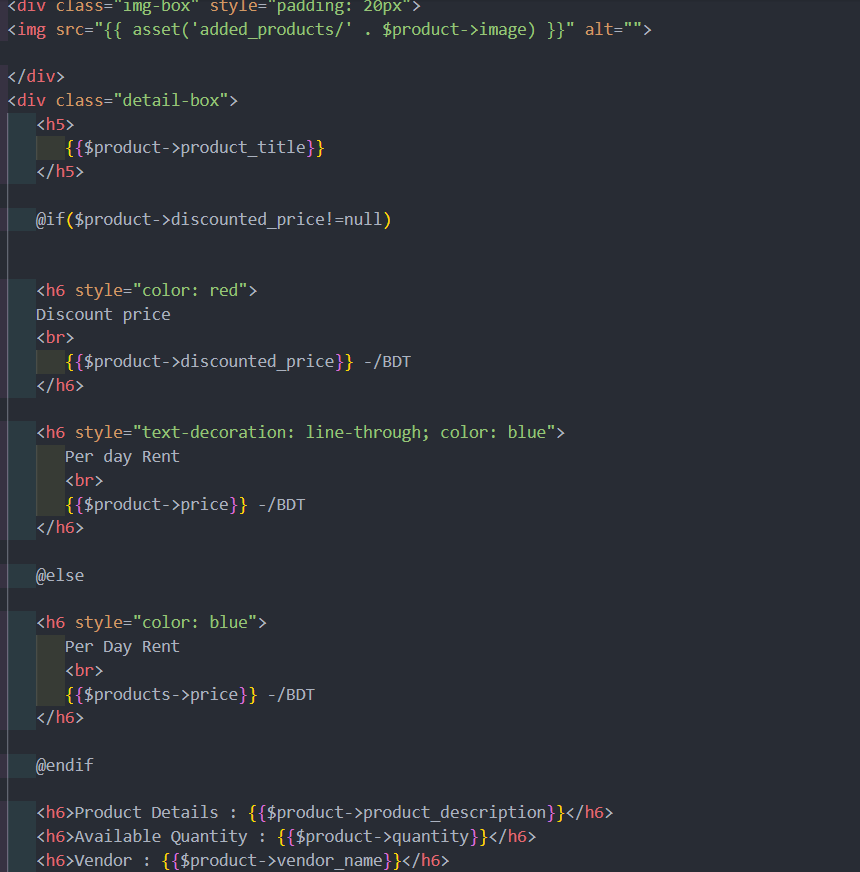


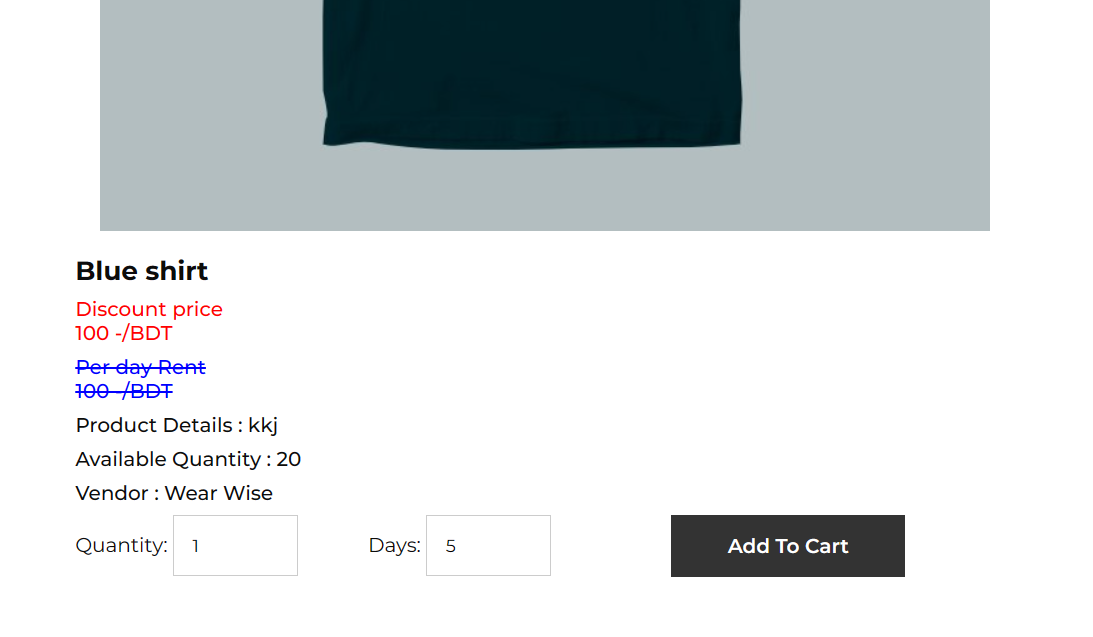
1. Footer : 



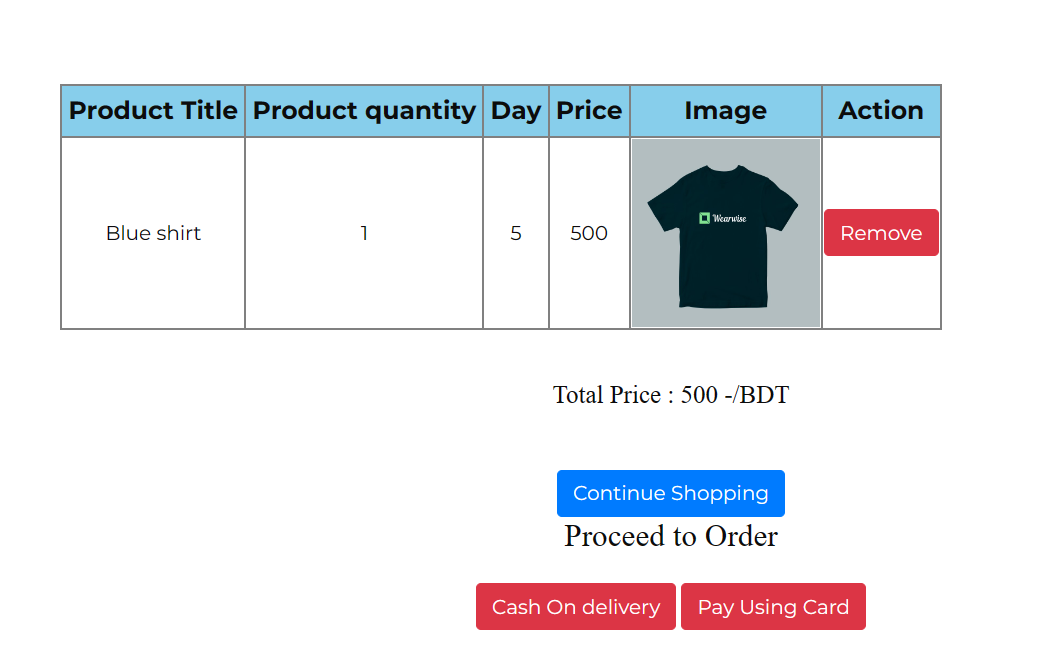
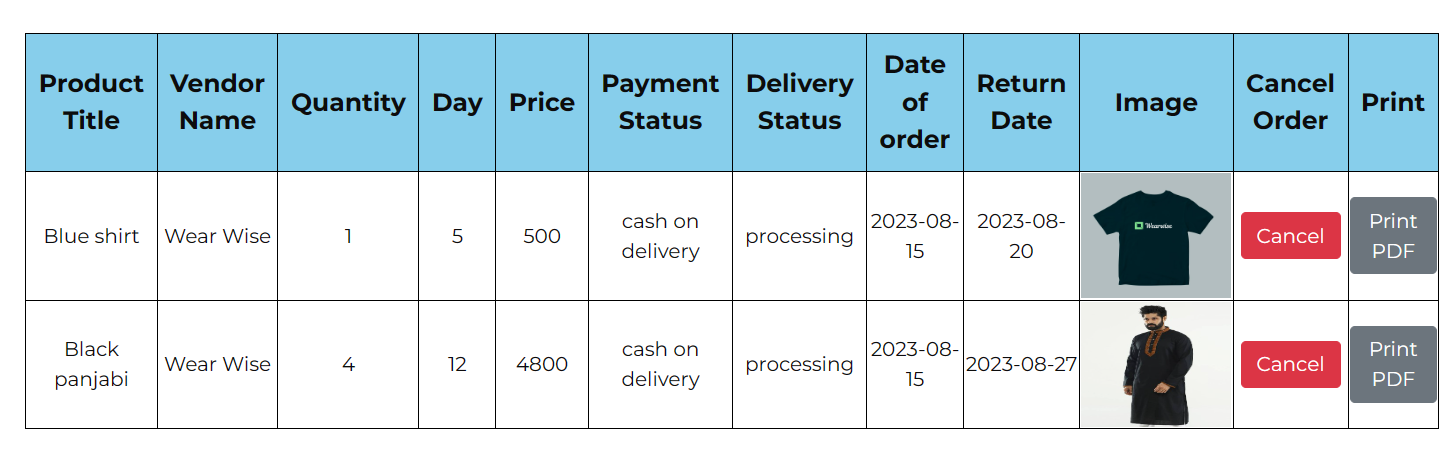
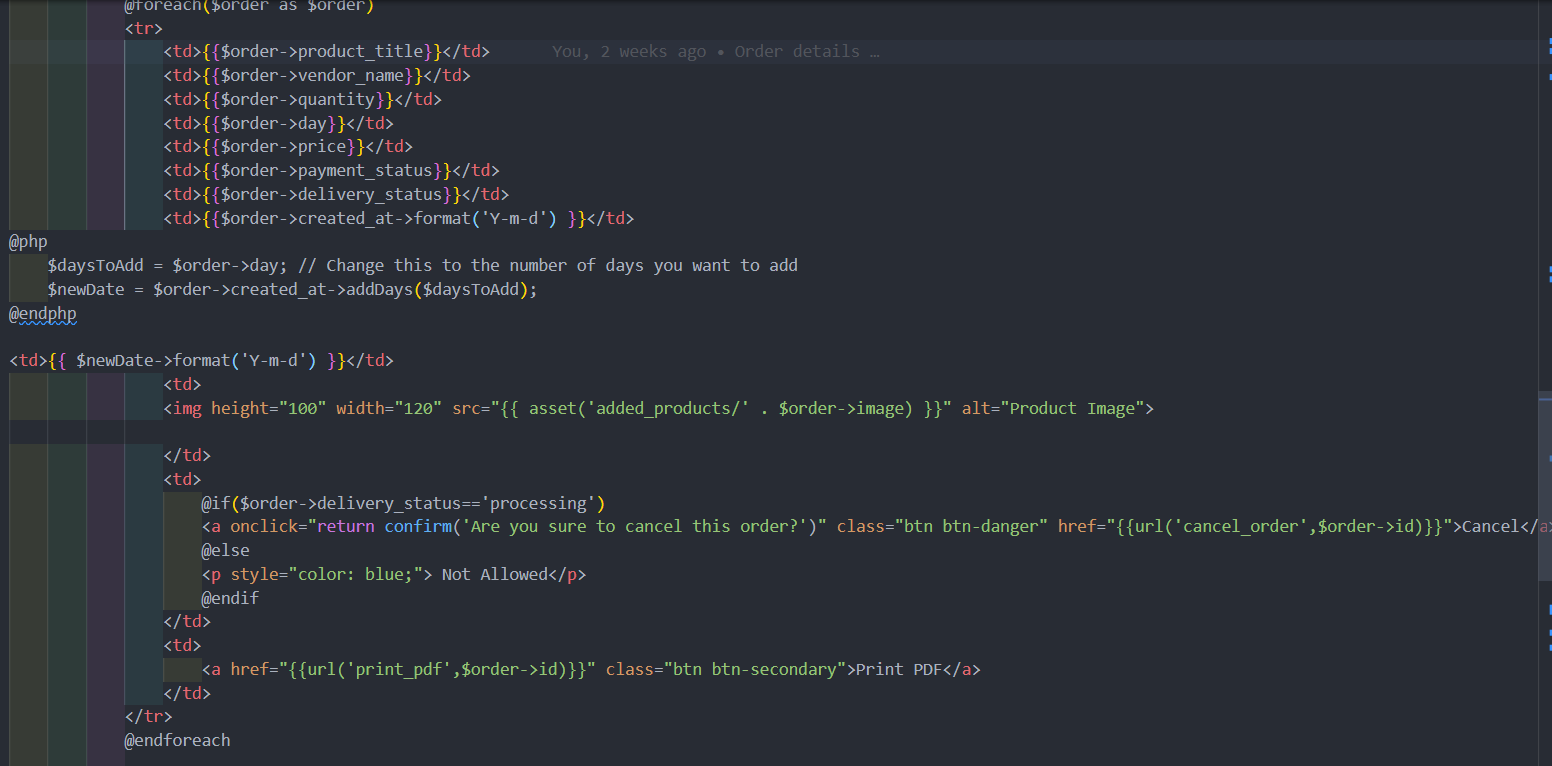
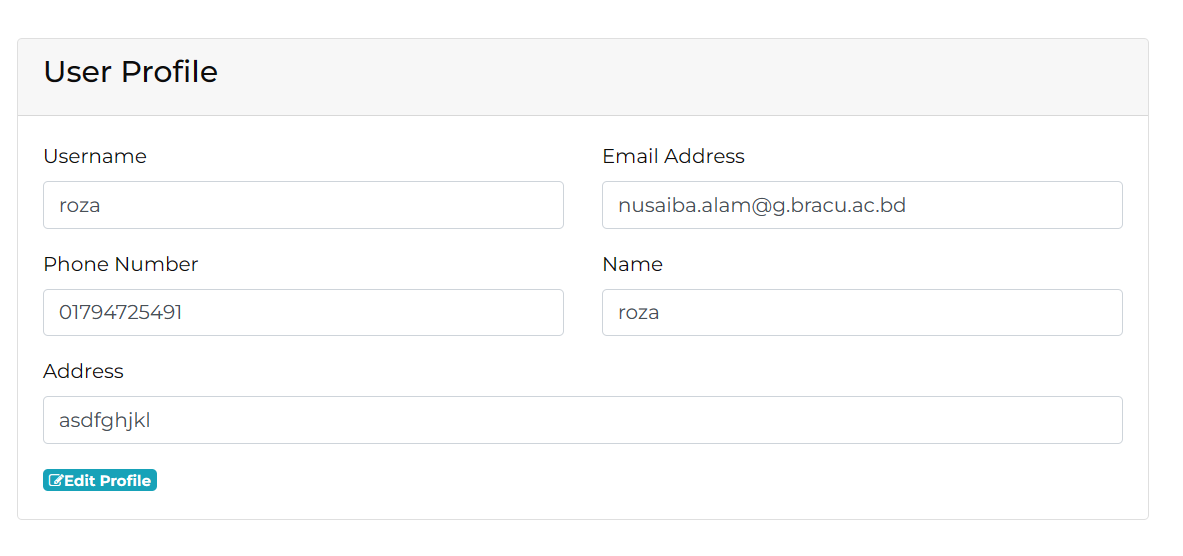
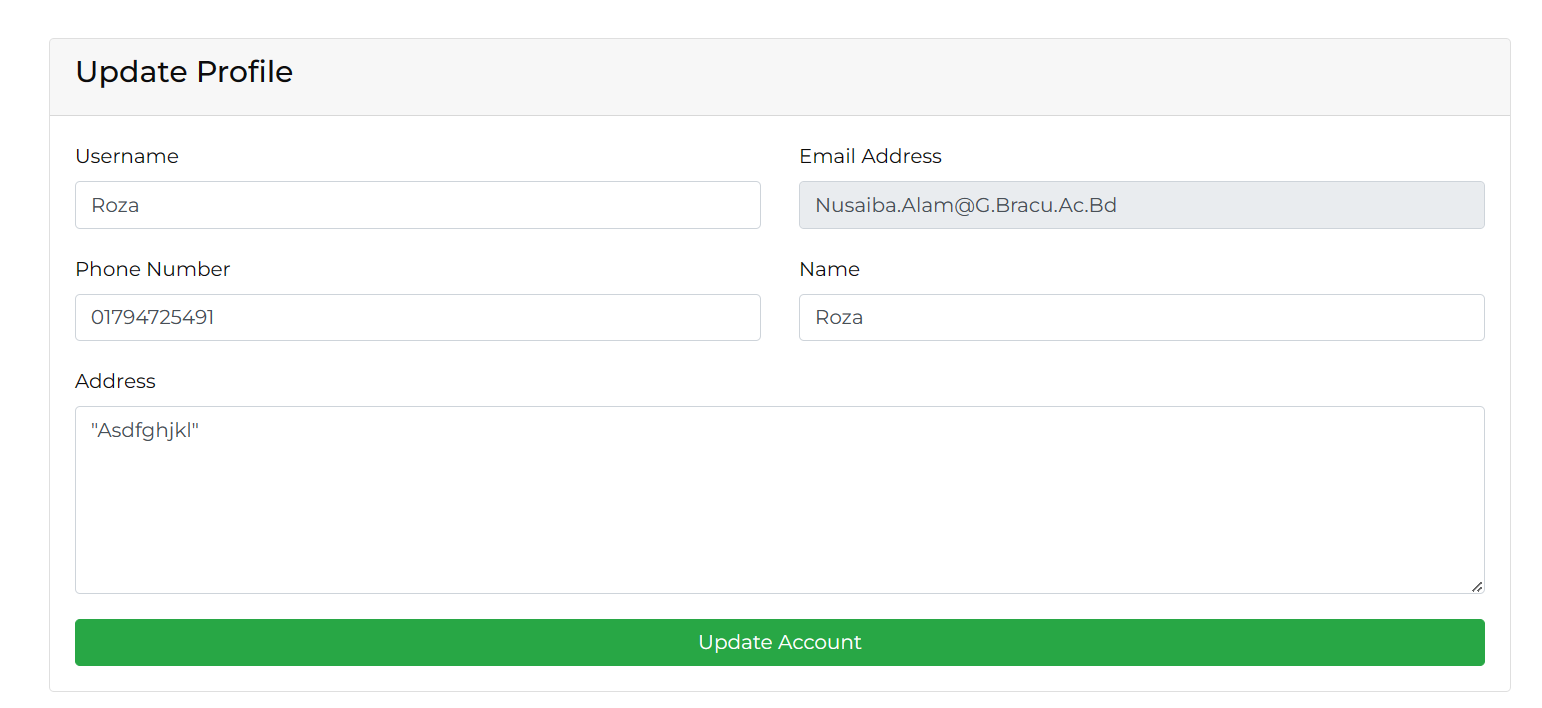
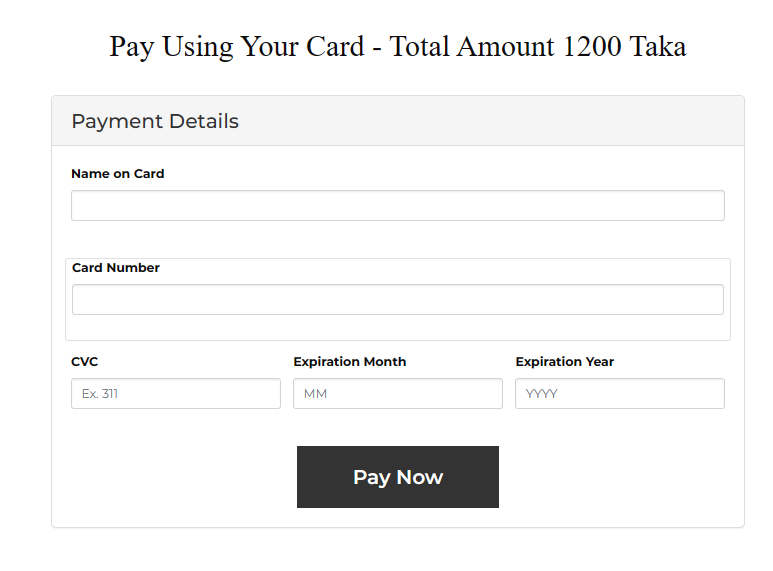
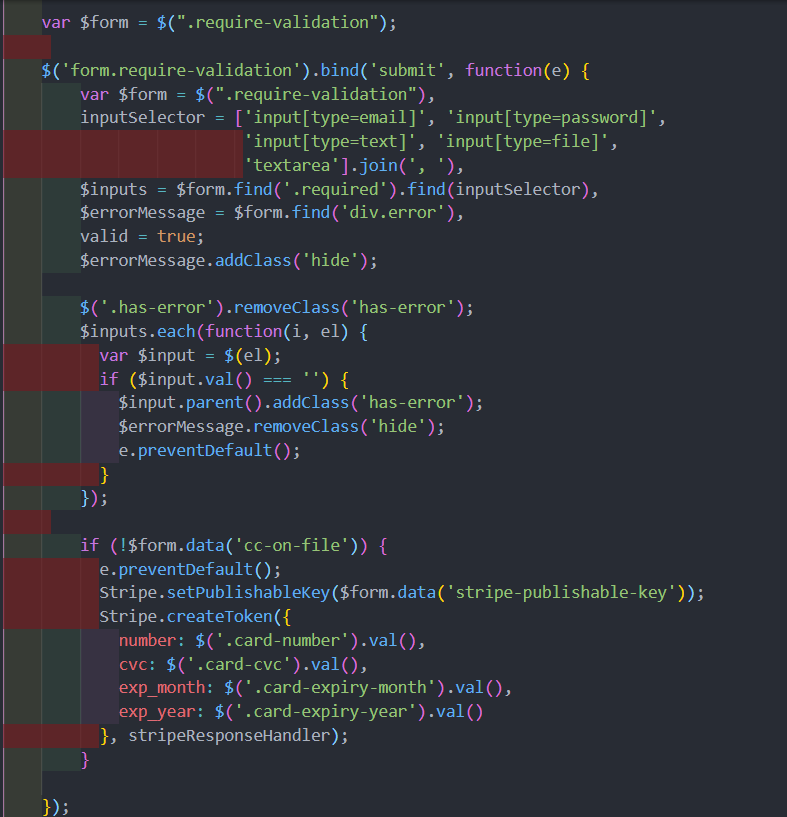
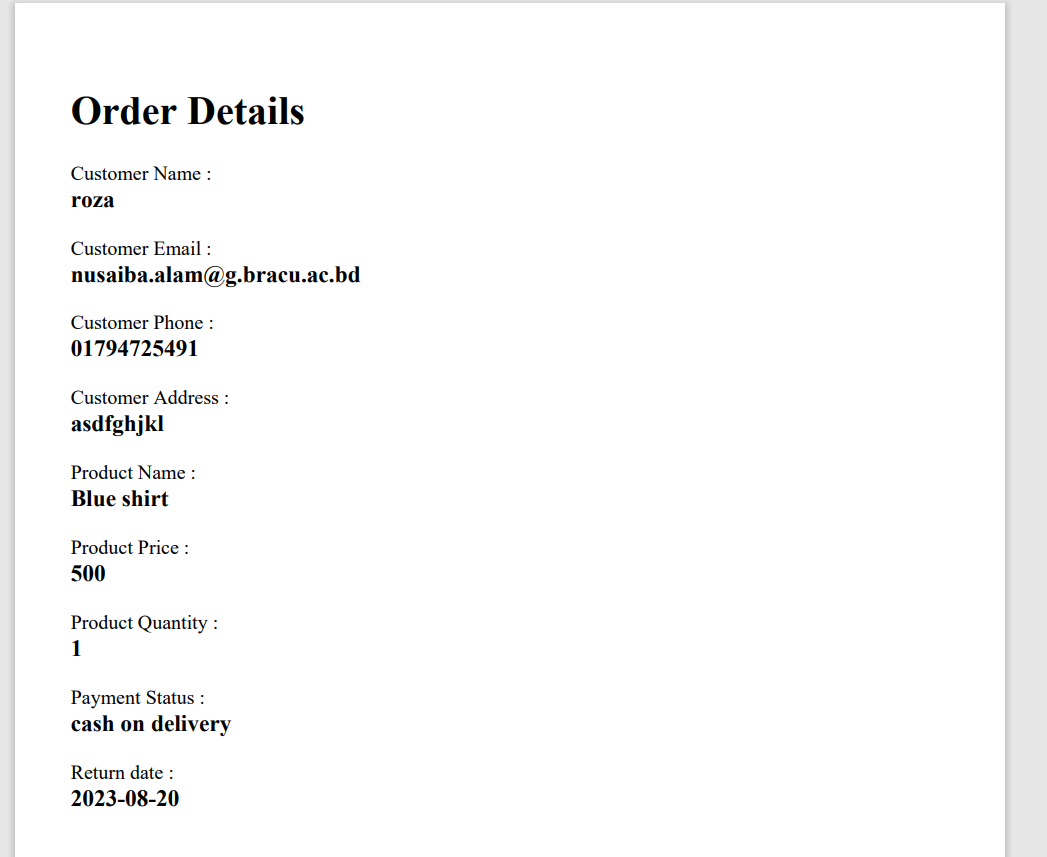
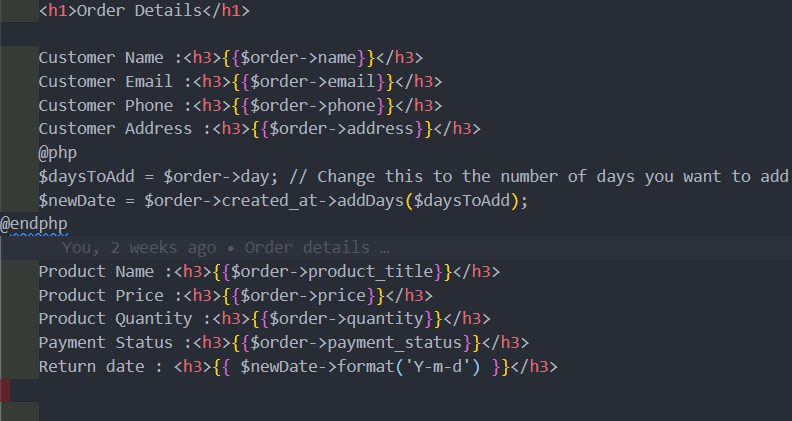
1. Product description :



1. Product Details : 



**Module 3: User Dashboard**

1. Add to Cart : 
2. Order Details : 
3. Profile view: 
4. Profile Edit : 
5. Payment gateway: 
6. Print PDF: 

**Backend Development**

Briefly discuss about Backend Development and add Screenshots

**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().

1. **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().

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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.

**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 | **Module 1: Admin Panel**   1. Login & Registration, logout & session management on all pages 2. Add product 3. View product + Edit product + Delete product 4. Add, Edit, Delete & View Category & Apparel 5. Admin sidebar, header 6. Dashboard features design & show data from database 7. View orders   **Module 2: Home /Shop**   1. User login, logout 2. User register 3. Session management on all pages   **Module 4: Vendor management:**   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 7. Vendor Dashboard features design & show data from database 8. sidebar, navbar   ***Admin panel:***   1. View Vendor account requests– approve, delete 2. View certified vendors & remove those vendors 3. View Vendor product post requests & approve / reject- (approved products go to main products table) |
|  | Nusaiba Alam | **Module 2: Home /Shop**   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   **Module 3 User dashboard:**   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 |
|  | Sartaj | 1. Admin panel view customer & delete 2. Product review |