**Multi store E-commerce Application**

**ABSTRACT**

The Document aimed to develop an online multi store shop using open source technologies (HTML5, CSS3, JavaScript, React JS, Apollo, MySQL and Strapi CMS).

This project is mainly divided into three main categories: The Administrators, Vendors and the Customers/Users. The store managers (admins) and the staff members (vendors) operate as the administrators. They can add, edit, update products or, delete products thus they able to change the names of products, change prices and, add or remove products. The customer can search for products selection, update the cart, remove products from the cart and check out from the shop. The customer is also able to update his information such as names, addresses, orders and other data.

**APPLICATION AND DESCRIPTION**

**1. Overview of the Various Applications**

This Project is divided into three different categories of applications.

1. Admin dashboard app. 2. Reseller dashboard app. 3. Store/Shop app.

Above applications are designed to meet the specific needs listed in Table. The overview of the three categories of applications.

|  |  |  |
| --- | --- | --- |
| **Admin Dashboard** | **Reseller Dashboard** | **Store/Shop App** |
| Designed for Super admin and vendors | Designed for vendors | Designed for customers/guest users |
| Login access for admin and vendors | Login access for vendors | Login access for customers/users |
| Admin and vendors are managed products, categories, carts, orders and customers data | vendors are export products, categories, orders data from admin into their stores | customers/users can view categories ,products and orders, can add/update products to cart, can make order/checkout products |
| Admin and vendors are managed payments data | vendors are make payments of customer orders to admin | customers are make payments of checkout products to vendor |

**2. Admin Dashboard Application details**

**➢ Super admin and vendors register**

The Super admin and vendors are needs to register before they can have access to the core data of the shop.

➢ **Super admin and vendor login**

The Super admin and vendors logs in and can view, add/update/delete products and categories, manage payments, customers and orders.

➢ **Manage Vendors and Customers**

The Super admin has the authority to delete or add a vendor. Vendor has the authority to delete, block and confirm a customer.

**3. Reseller Dashboard Application details**

➢ **Sign up**

This refers to registering as a vendor. The registered member has a lot of privileges associated with the shop when one becomes a vendor.

➢ **Login**

After the user has registered, the user becomes a vendor, and can log in with their personal information.

➢ **Smart Export Tool**

The vendor can see all the products in the catalog and exports these products with profit margin into their store shop by categories wise and product wise.

➢ **Edit**

The vendor can make changes to their data displayed on my account page.

➢ **Cart** **Converter**

This refers to convert customer orders to vendor cart and make payments of customer orders to Super admin by place order with vendor account.

**4. Store/Shop details**

➢ **Sign up**

This refers to registering as a customer. The registered member has a lot of privileges associated with the shop when one becomes a customer.

➢ **Login**

After the user has registered, the user becomes a customer, and customer can log in with their personal information.

➢ **Edit**

The customer can make changes to their data displayed on my account page.

➢ **Cart**

This refers to putting or removing products from a shopping cart.

➢ **Order**

After the customer/guest has added products to cart, customer/guest can check out that cart and make payment to vendor.

➢ **Order Track**

After order products, customer/guest has received a mail receipt of order. Customer/guest can know order status via order track page with the help of order Id.

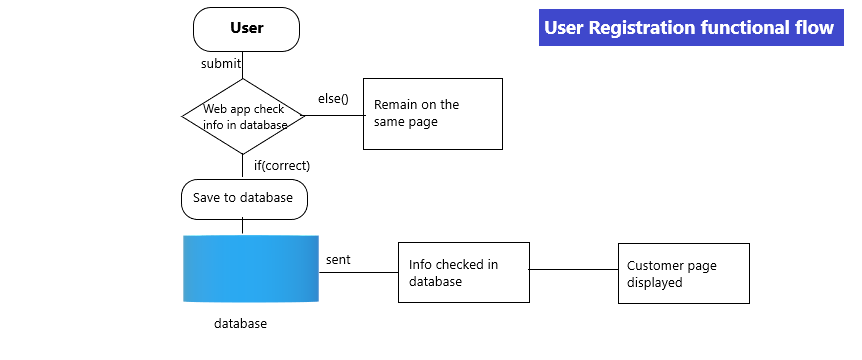
**5. The Various Management Units**

The Administrators (Super admin and vendor) play the management role. They make sure everything in the shop runs smoothly. Table lists the various management units.

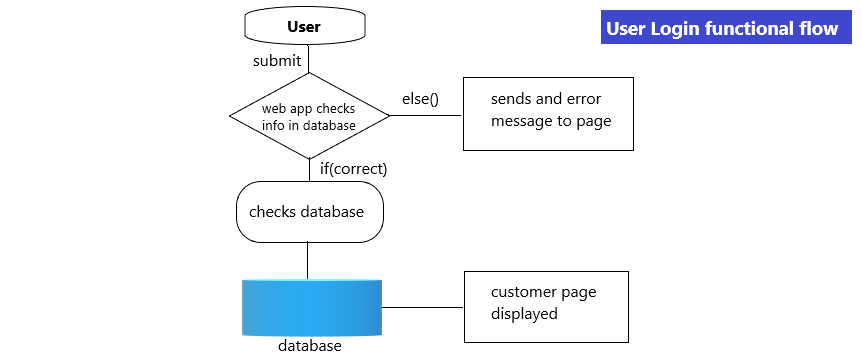
|  |
| --- |
| **Product Management Unit** |
| This is the Unit that is responsible for Keeping records, product name, description, price, SKU, product images, product details, product variants. |
| **Category Management Unit** |
| This is the Unit that is responsible for Keeping records, category name, category images, category children, category products. |
| **Order Management Unit** |
| This is the Unit that is responsible for Keeping records, order entries, ordered products, order payment. |
| **Admin Login Unit** |
| The administrator can log into management dashboard and make use of features of dashboard such as add/update/delete categories, products, orders. |
| **Admin Logout Unit** |
| The administrator will be able to logged out with this function |

**6. Customer, Vendor and Super Admin Registration**

The Customers, Vendors and Super Admin will use their exclusive information to register. After filling the form and submitting it, the Web Application (Admin dashboard/Reseller dashboard/Store App) to see if all the fields entered by the user are correct. If the area is not correctly filled the user remains on the same page but if the requirements are met the data goes to the database (table "users") and saves the information of the User. The User then becomes a customer/vendor, and customer/vendor/super admin is then directed to the home webpage of the Admin dashboard/Reseller dashboard/Store App.



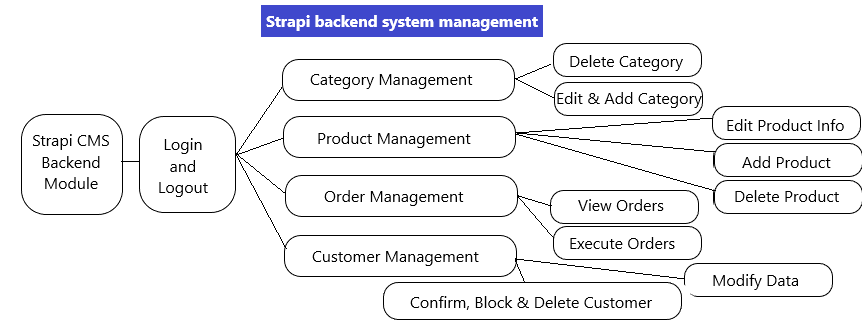
**7. Customer, Vendor and Super Admin Login**

****

The customer/vendor/super admin will use his particular data e-mail, password to log into the Web Application. After submitting the form, the Web Application (Admin dashboard/Reseller dashboard/Store App) checks if all the fields have been filled correctly. If the condition is not met the customer/vendor/super admin remains on the same page but if the fields are correctly filled the customer/vendor/super admin login information is sent to the database to check if the data entered into the areas are same as the ones used to register to the database. If it is correct, the customer/vendor/super admin is redirected to his homepage.

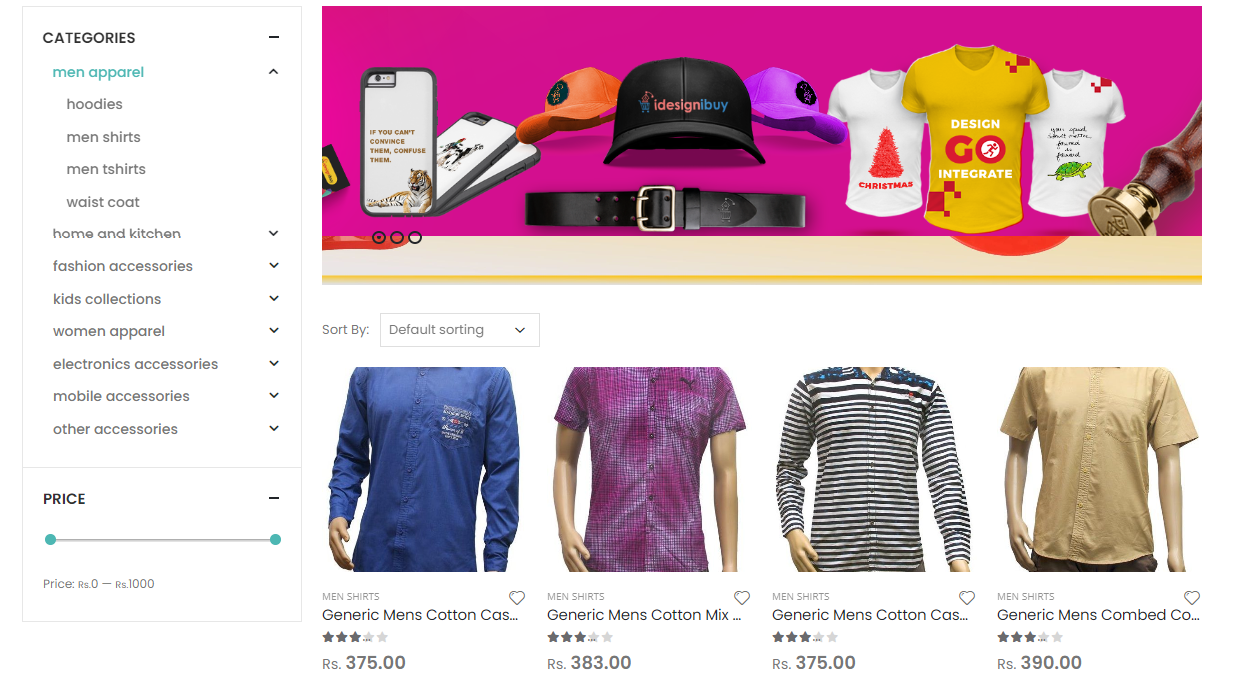
**8. Strapi Backend Module Layout**

This includes Units such as products, category, orders and, customer management modules. The back-end module/Unit is used to manage the backend of the shop. This is only available to the administrators/vendors. They can manipulate the shop to suit the conditions they have set for the shop. They also make sure that vendors have a good experience when visiting the shop by adding products, updating products, deleting products, executing orders to database, and managing customers.

****

**9. Customer/Vendor Interface layout**

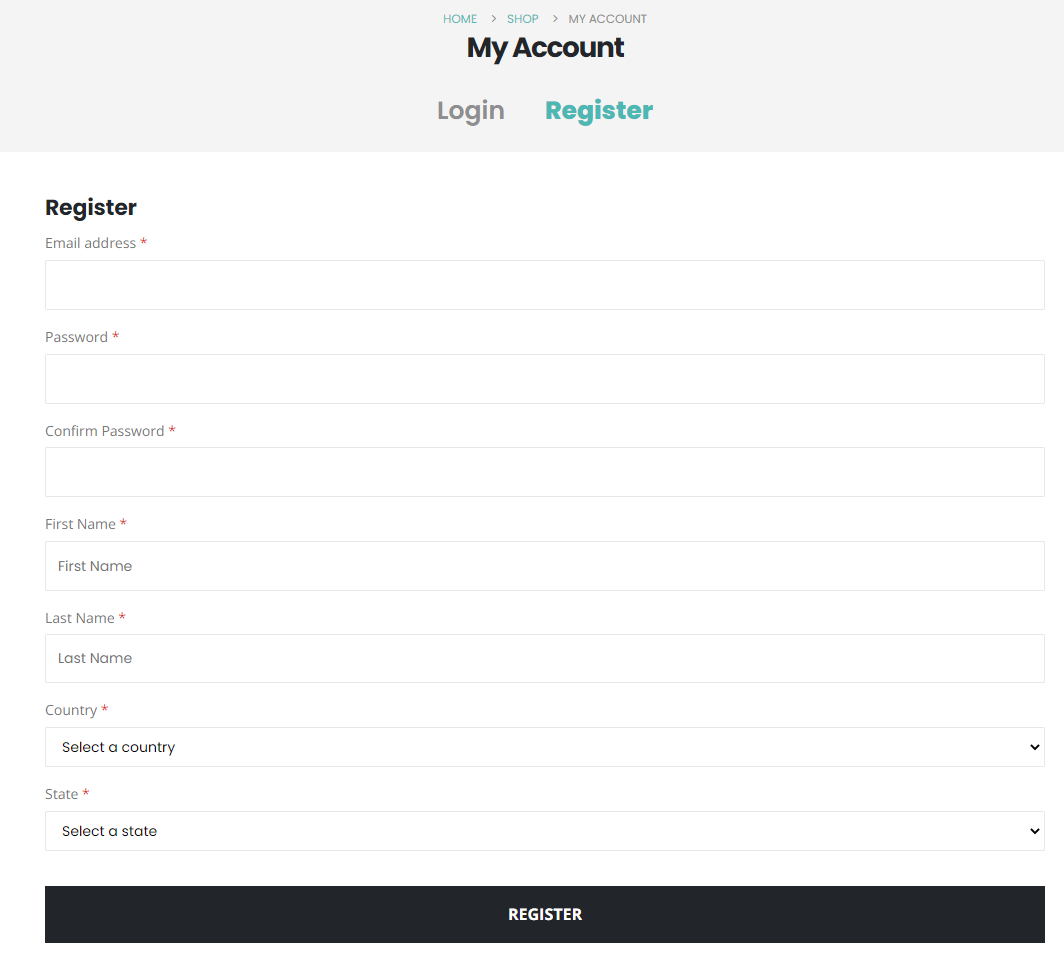
Consumer refers to customers and non-customers. These are individuals who visit the shop either to buy products or to browse. There are two categories of interfaces namely the Consumer and the Administrator interface. The administrator has higher authority over the customer in the shop. The admin can edit, replace a product and, manipulate data in the shop. The customer can browse a product, add a product to the cart, change personal information, and check shopping history and checkout or log out. The User, on the other hand, can only browse and add a product to cart. The homepage or interface is the index page of the shop so can be accessed when the address is typed into a browser. The webpage has products images, names, prices, product categories and product brands. The webpage has a registration link, login link, cart, company contact information

****

Home page of the shop

➢ **Registration web page**

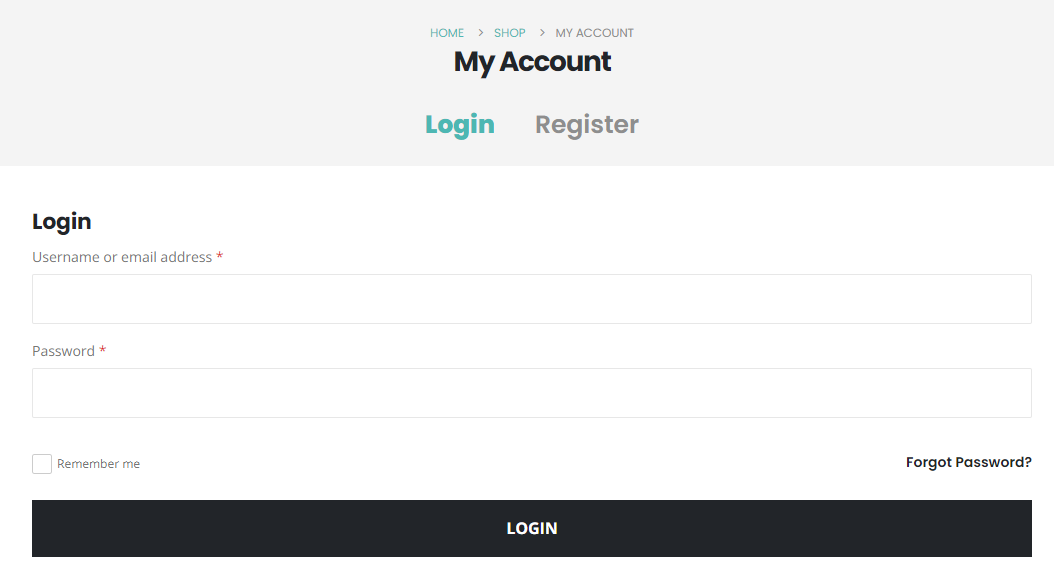
This registration page is only for users who wish to become customers/vendors. They have to meet the required filled standard for them to register successfully; otherwise, they will remain on the same page.



Customer/Vendor Registration Web page

➢ **Login web page**

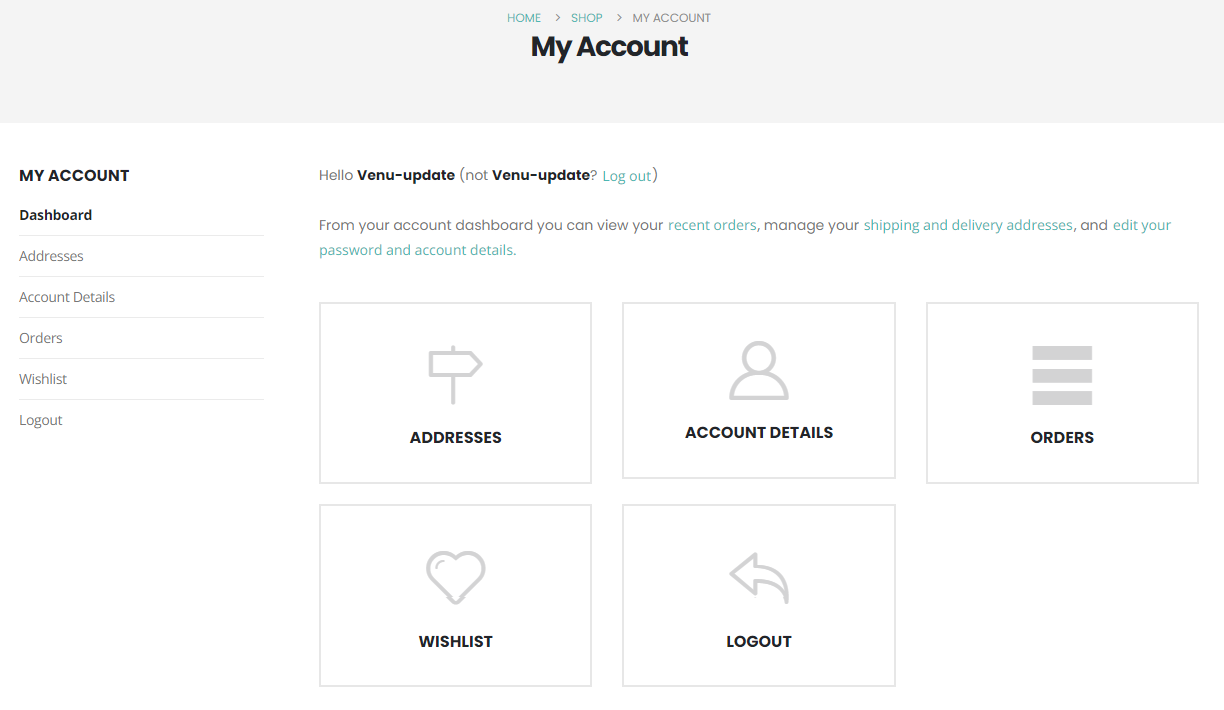
Customers/Vendors will have the right to log in with their information such as e-mail and password. The information is sent to the database to check for a match. If no match is found the customer remains on the same page, otherwise he is directed to the customer page.



Customer/Vendor Login Web page

**➢ Customer/Vendor Account page**

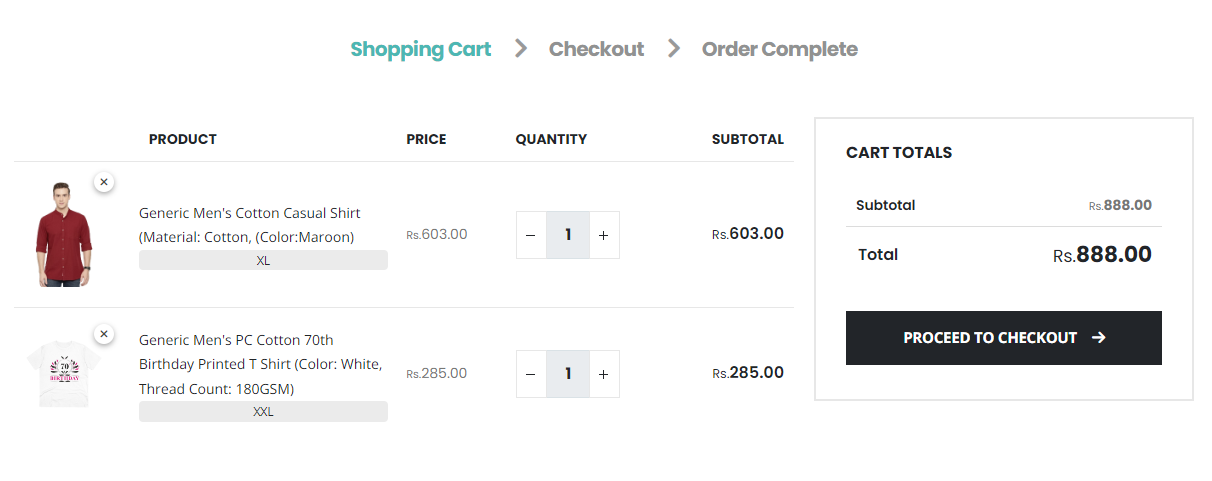
This web page can be accessed if customer/vendor information matches with the one in the database that was used to initially register. The customer is able to edit data, orders, password, and log out.



Customer/Vendor account page

**➢ Shopping Cart**

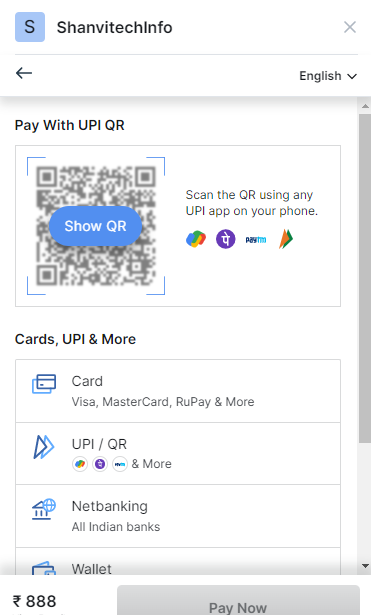
The customer can add products to the shopping cart. They can also update the shopping cart from time to time until they are done shopping. After that, they can check out by clicking on the checkout button



Shopping Cart

**➢ Payment**

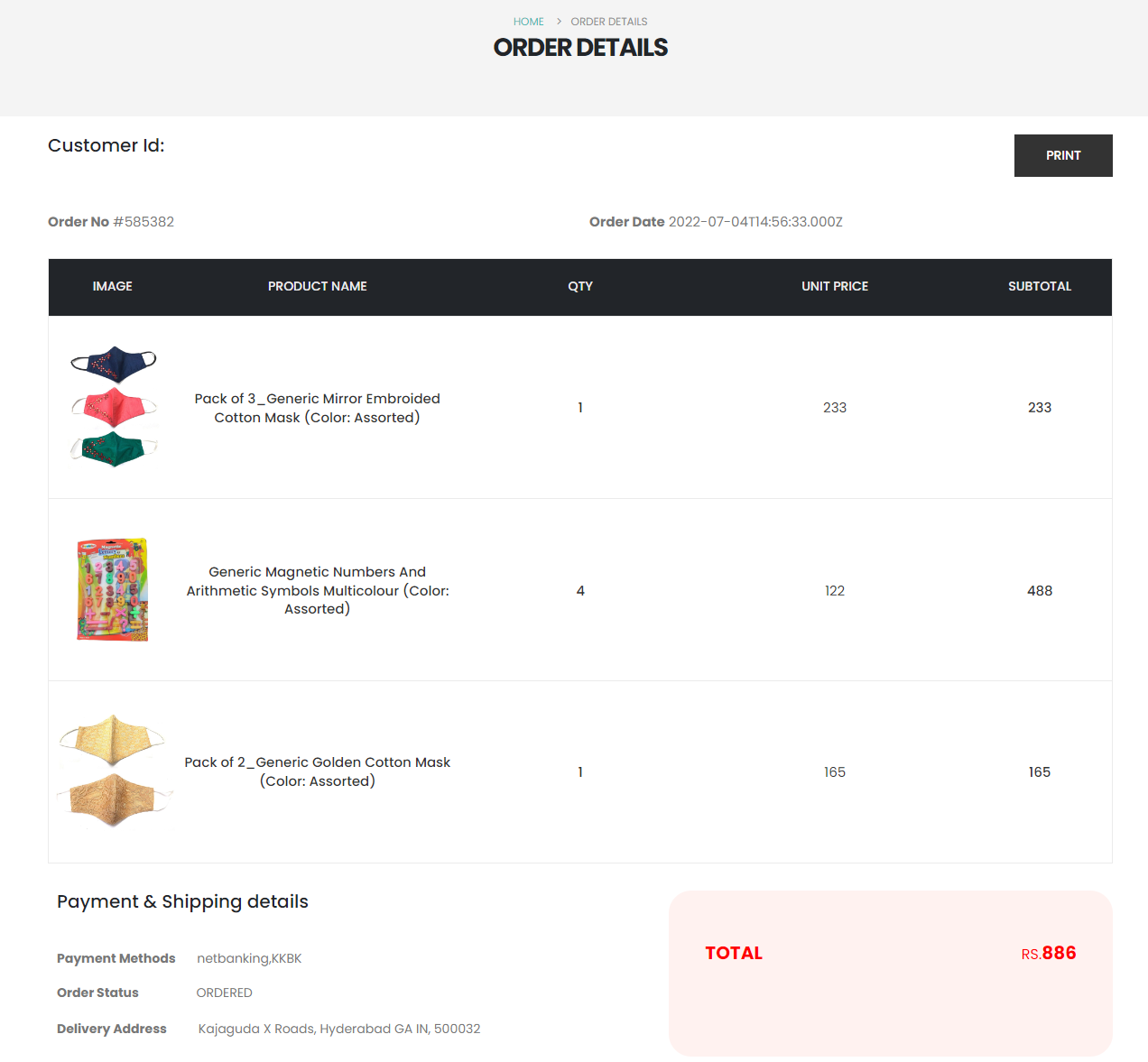
This is the final stage of the shopping experience. The customer who is satisfied with their choice of product and price can now pay with



Payment Options

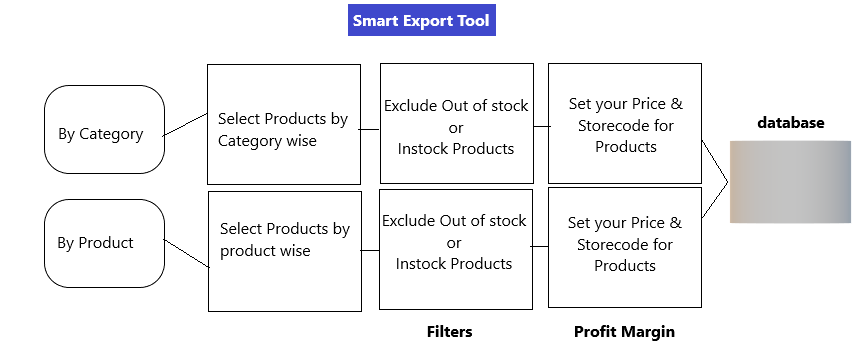
**➢ Order Track**

After Customer/Vendor made order, they can know order status with the help of order track user interface.



**➢ Smart Export Tool**

Smart Export Tool is designed and developed for Vendors. This user interface is in Reseller dashboard app for exporting products from admin shop to vendors shop. Vendors export products by categories wise and products wise. Vendors choose their profit margin for product prices and import into their shops. For using smart export tool, vendors must be register in the reseller dashboard. This tool also provide filters, these filters are exclude out of stock and in stock products.

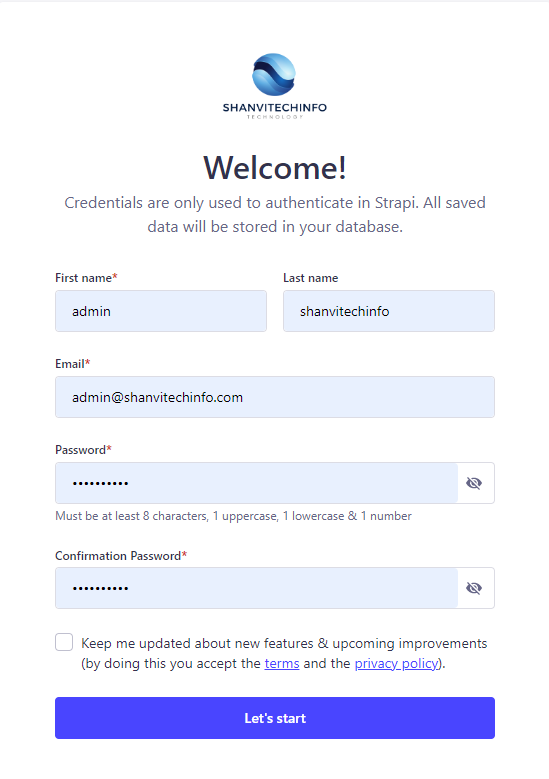


**➢ Customers Orders**

Customers orders are made by customers in store/shop app. Vendors are received these orders in their stores and again make order to admin. Vendors are getting customers order with their account information with the help of My Account dashboard user interface. Vendors are making payments of customer order to admin (Drop shipper). For this action, Place order button is provided in customer order details page. It is convert customer orders into vendor’s cart. Vendor can checkout this cart.

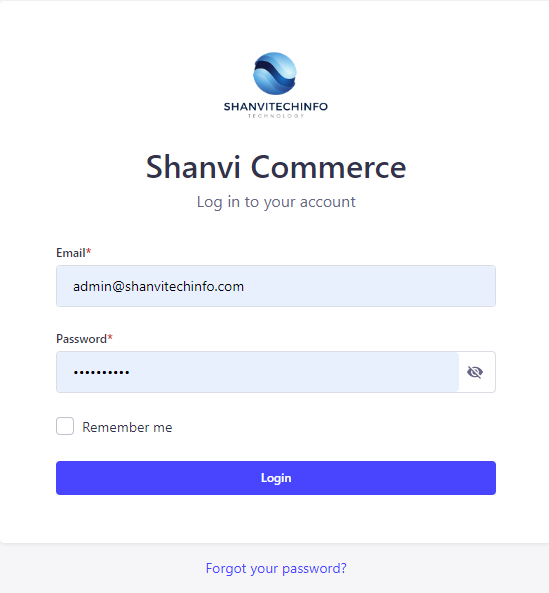
**10. Management Interface design**

The administrator must enter username, email, and password. The strapi will check if all the fields are filled. If field conditions are not met the admin remains on the same page; otherwise, information is saved in the database.

****

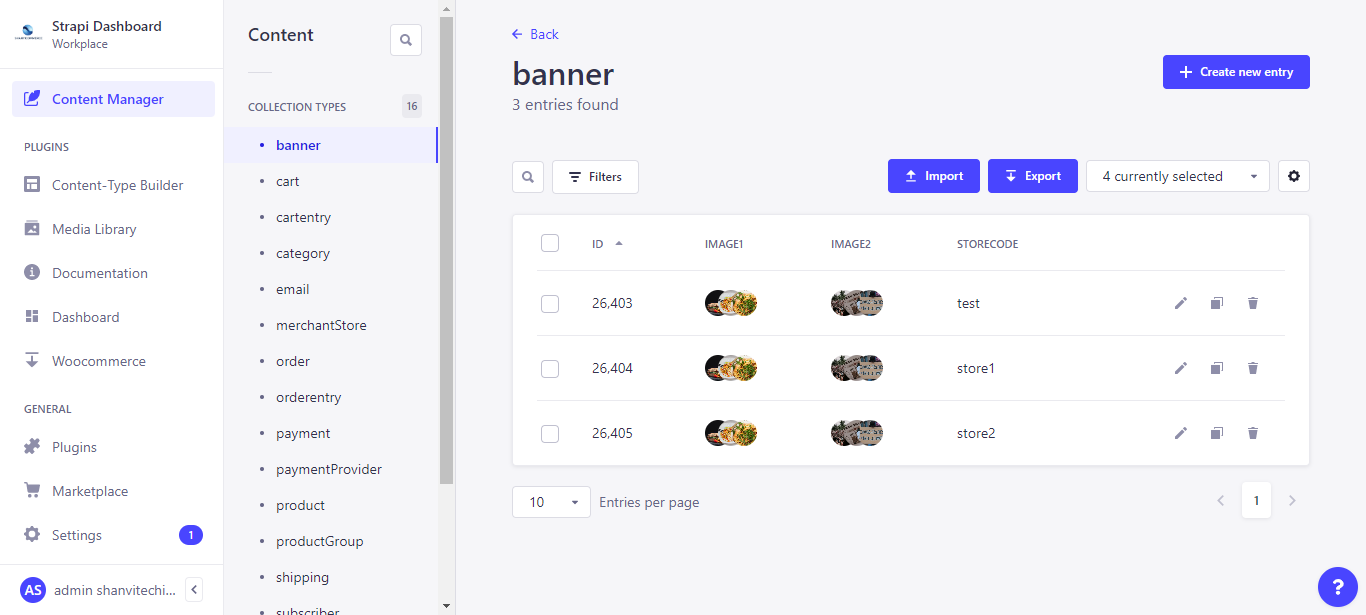
Strapi Super Admin/vendor registration page

The administrators can log in with their email and password. If there is no match for the information provided Admin will remain on the same page until the correct data are inserted into the login space.



Strapi Super Admin/Vendor Login Page

When the administrator/vendor has finally succeeded to log in the admin's content management area will emerge for the admin/vendor to have access to the webpage.



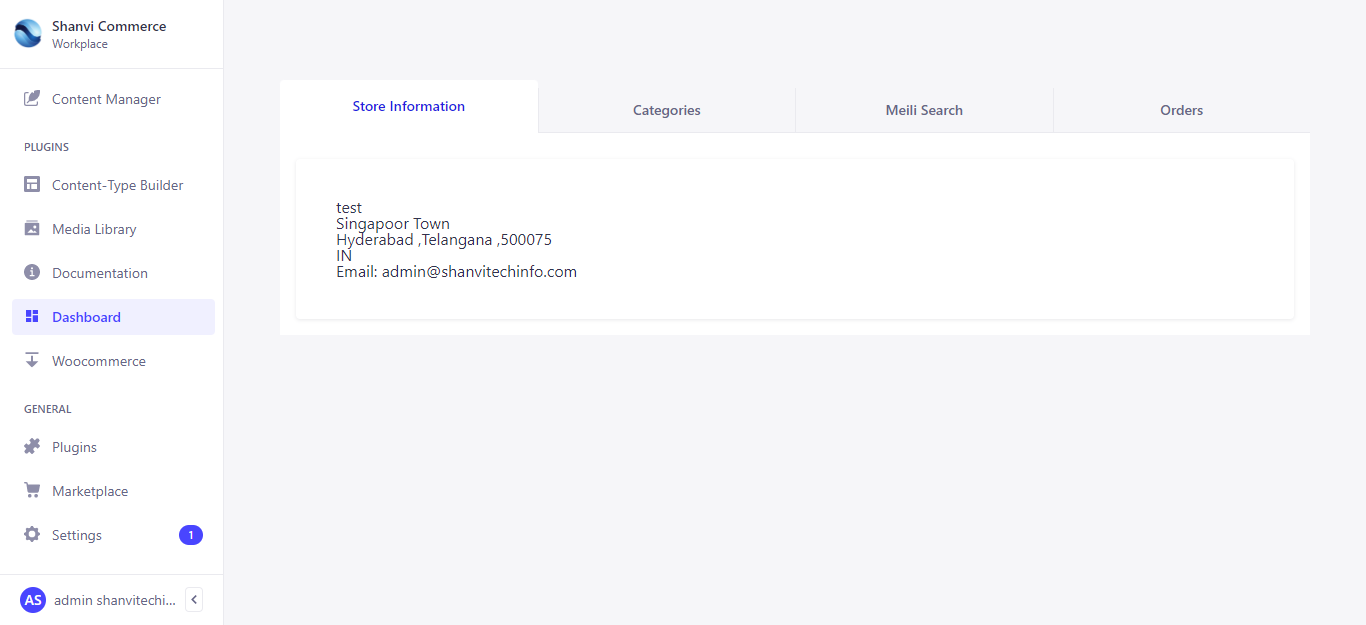
Admin/Vendor content management

**➢ Custom Plugins Development**

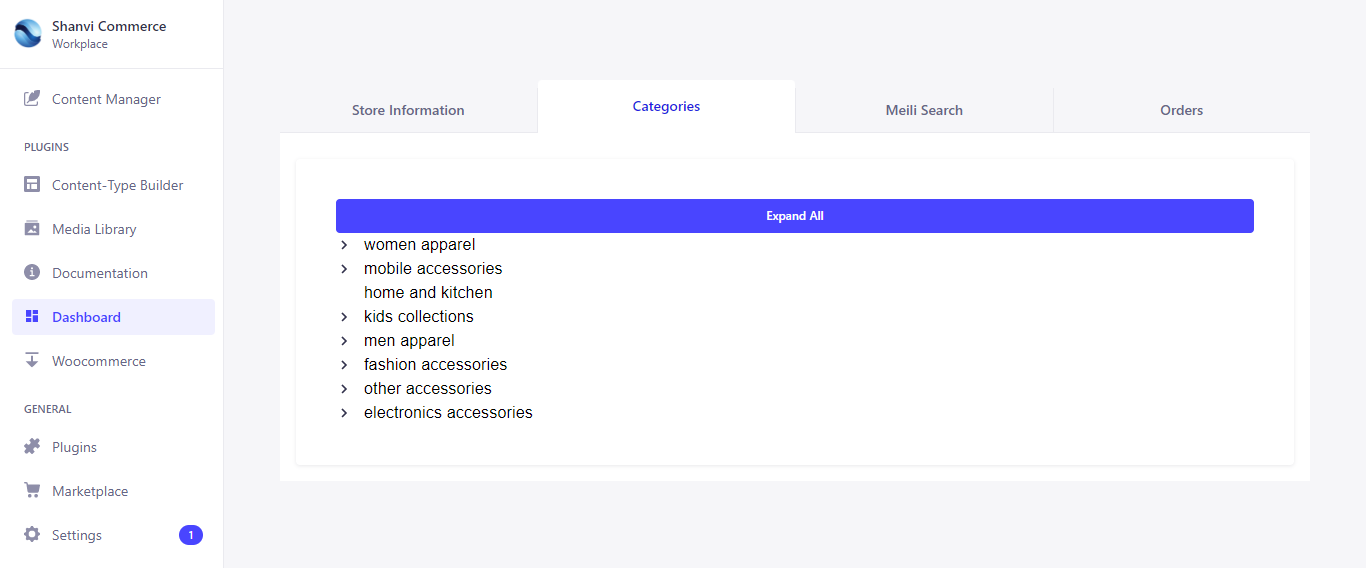
Strapi provides programmatic APIs for plugins to hook into some of Strapi's features. Plugins can register with the server and/or the admin panel, by looking for entry point files at the root of the package:

* strapi-server.js for the Server,
* strapi-admin.js for the admin panel.

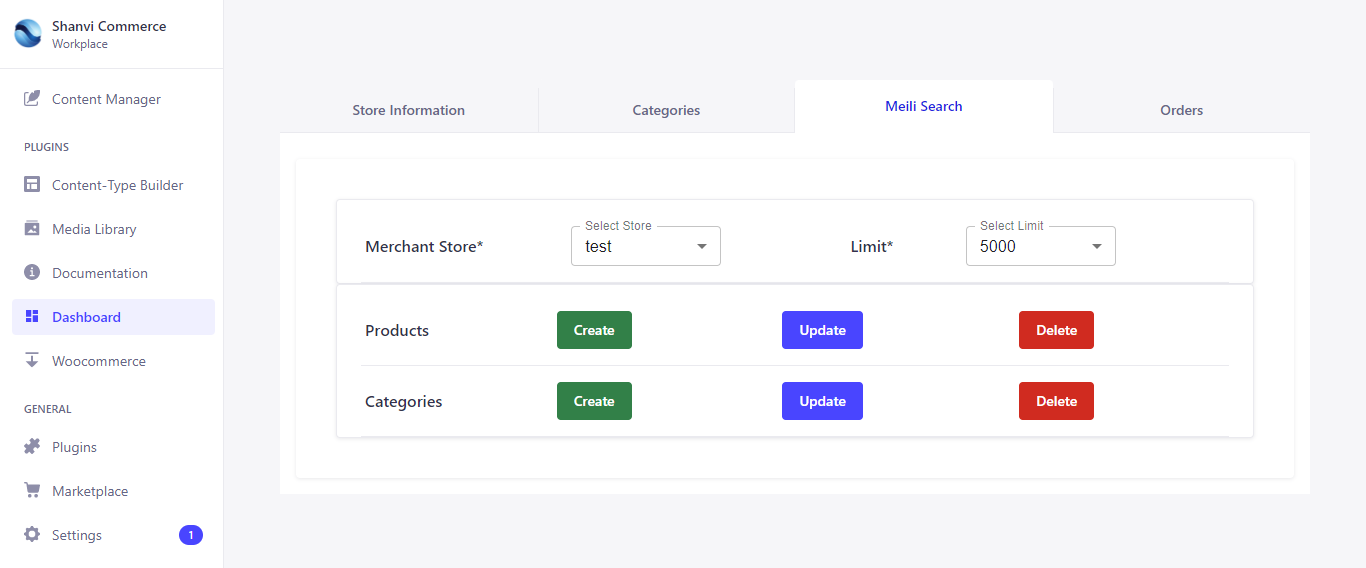
Dashboard and Woo commerce plugins are developed in Admin dashboard. Dashboard is for display store information, categories, orders and CRUD operations in Meilisearch.



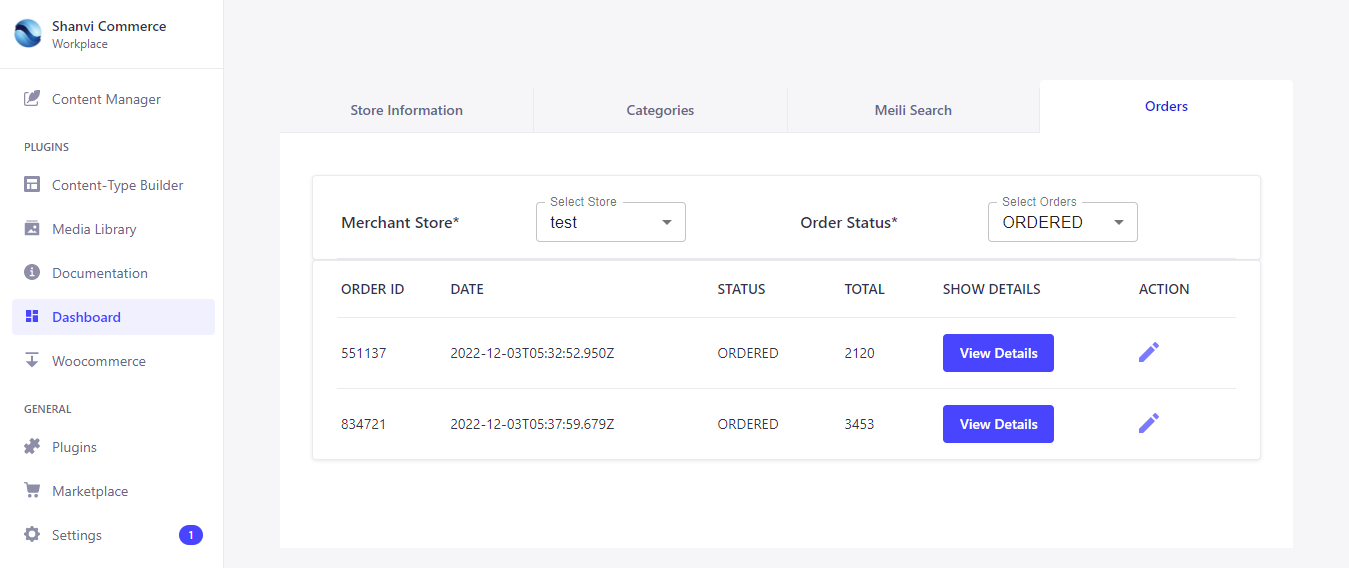
Store Information Tab



Categories Tab

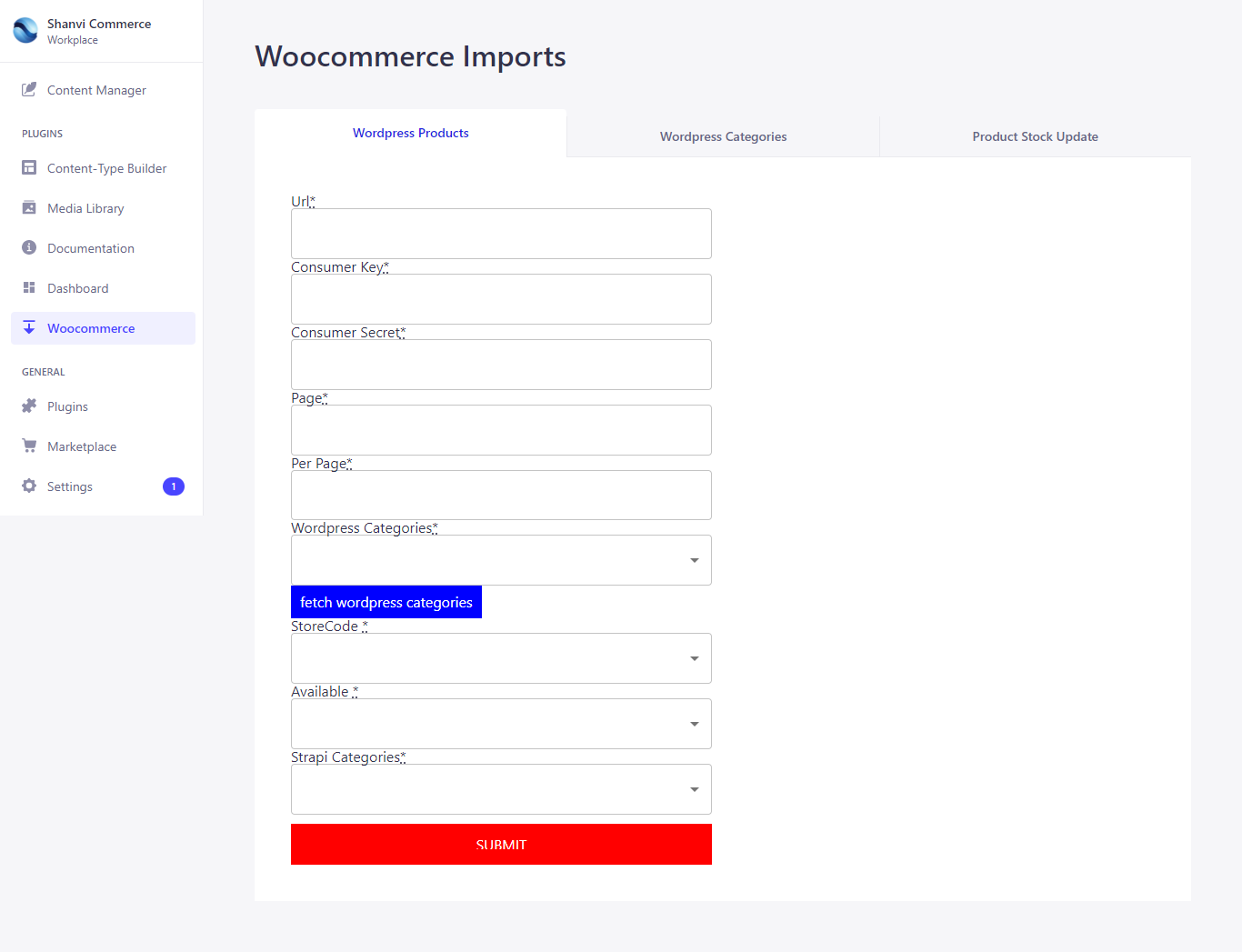
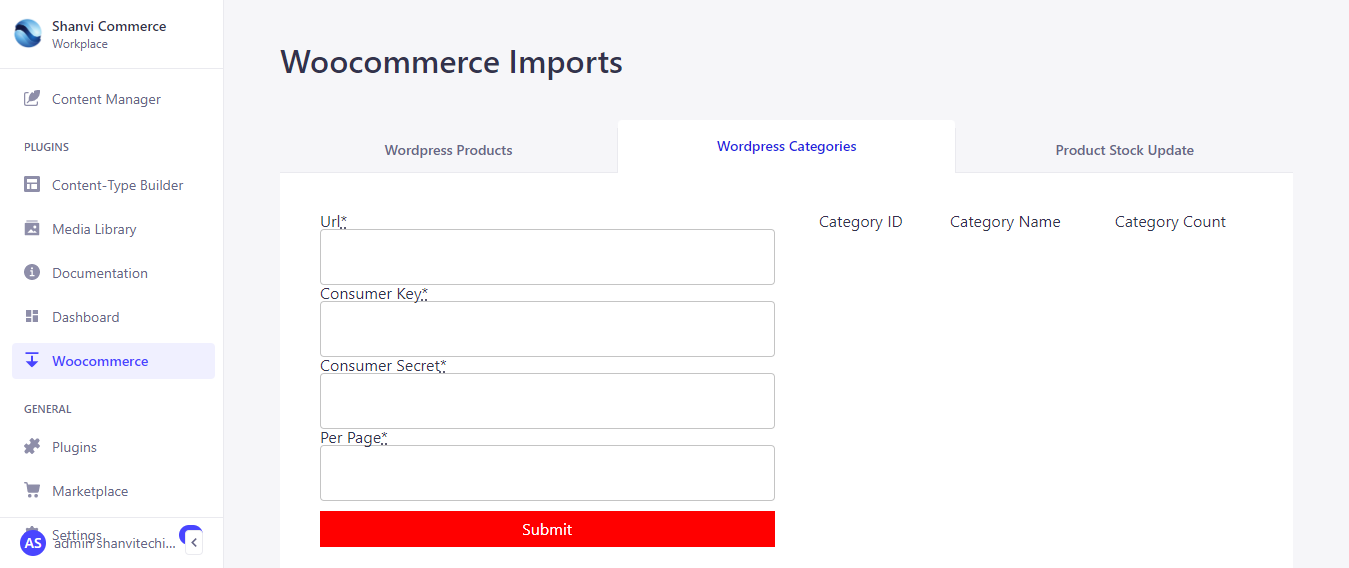


Meilisearch Tab

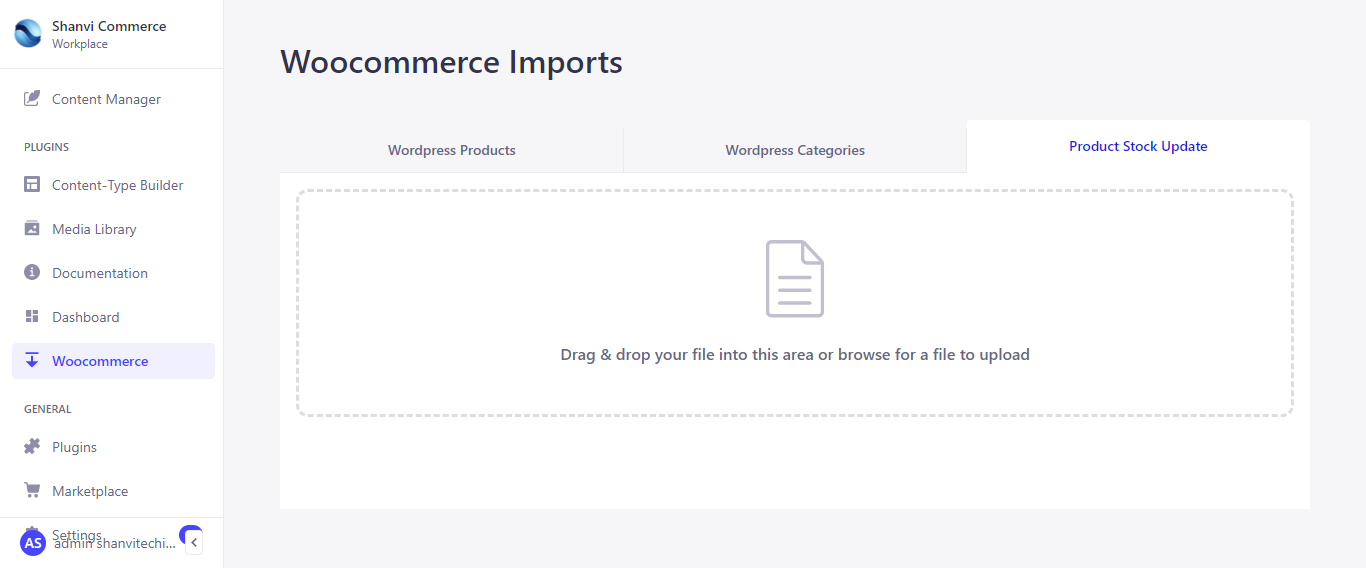


Orders Tab

Woocommerce is for import products, display categories and Update products stocks based on product sku.

Product Import based on Categories wise

Categories UI



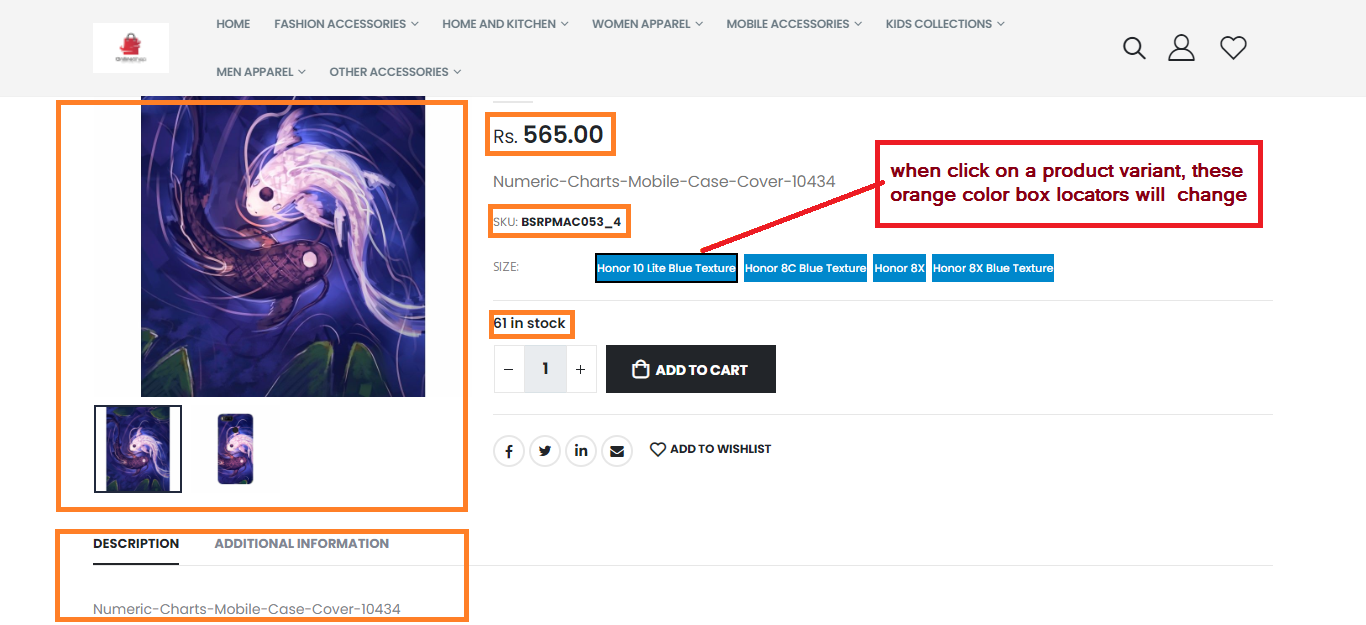
Products stocks update UI

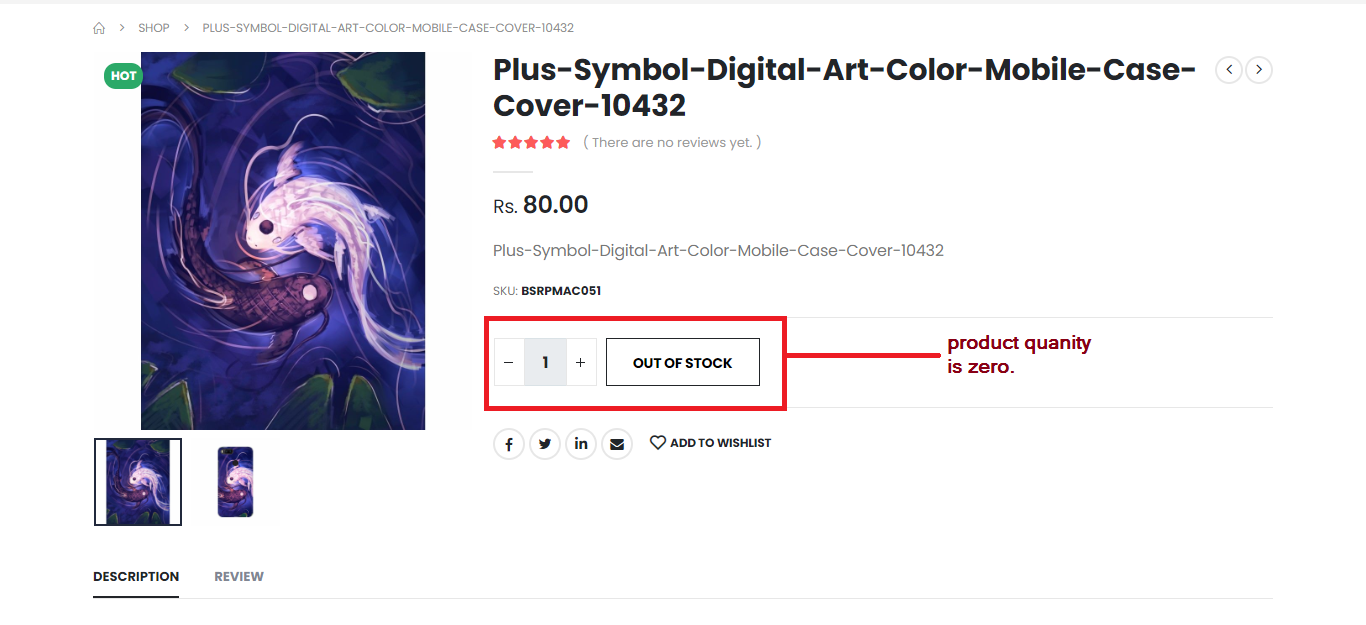
**11. Testing**

To affirm the online store assessments had been made at distinct stages of the task; I checked the reliability of all of the functions. The test is built on the customer/vendor and Management Unit. The customer test proved that an account could be created, login can be established, the cart can be loaded with products and the customer can check out when done shopping. The administrator can create an account and log in afterward. The admin can then manage all the content in the shop.

➢ **Products**

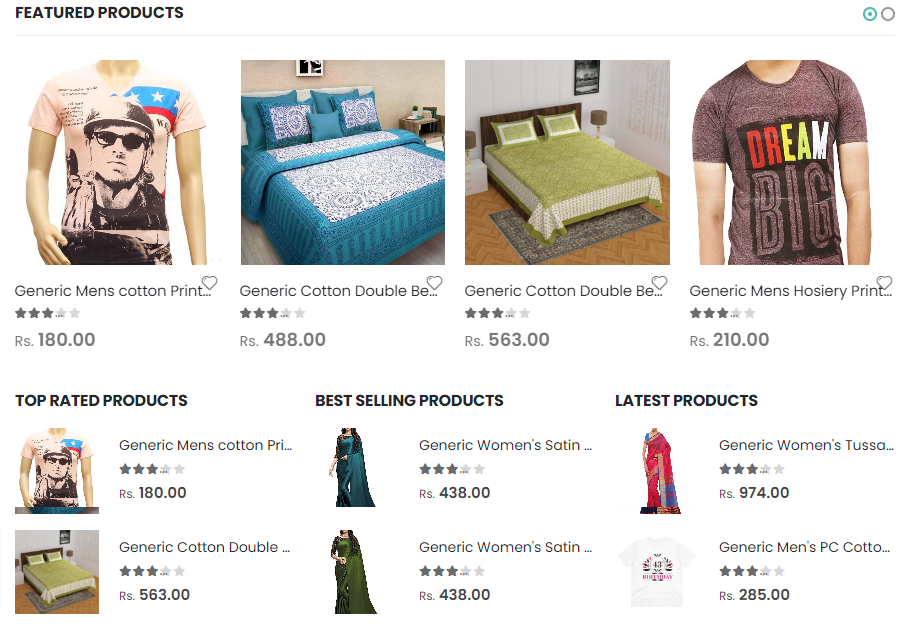
Products (single or variants) appearance on website based on available (for product) and visible (for variants) options true or false in backend. Product (single or variants) information details like price, stock, images, variants and description displayed on product details page in website. Here price, images and description displayed with respect to specific product (single or variants). If product is out of stock, it can’t add to shopping cart. Add to cart button disable, and out of stock displayed here.





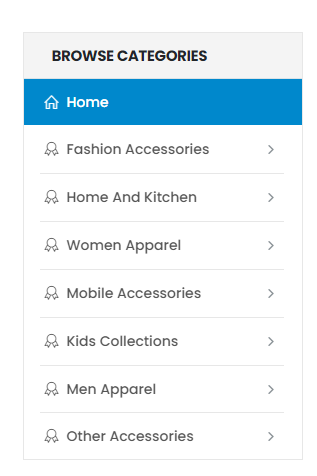
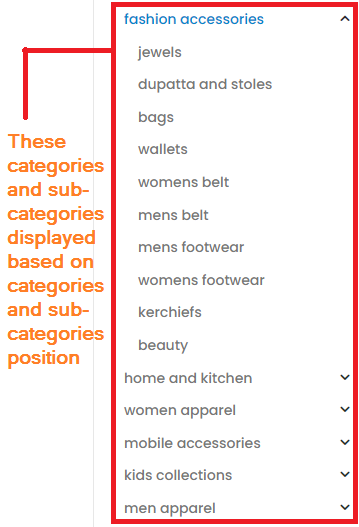
Product displayed in product details page

Products are categorized into product groups like featured products, bestselling products, latest products and top rated products. With the help of products group api, displayed on various parts of website



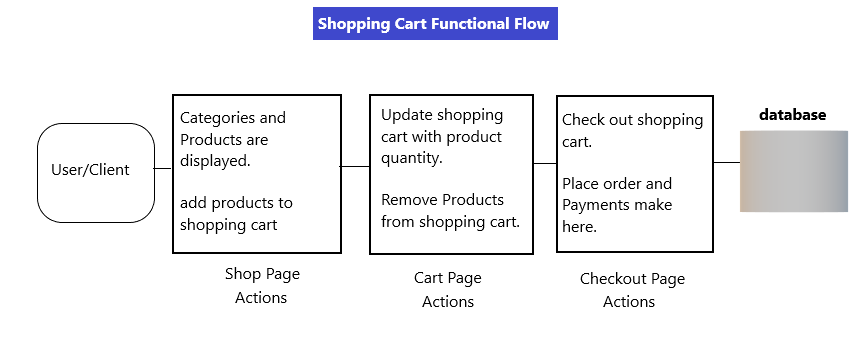
➢ **Categories**

Categories or its sub categories appearance on website based on visible options true or false in backend.



C:\Users\user\Downloads\FireShot\navcategories.png

➢ **Cart Scenarios**



1. Guest user add products (single product or product variants) with quantity to the shopping cart for first time from Feature products (home page),Shop page from Quick view or product details page.

Result: Cart record will create in backend, get cart id and other fields from API response. This cart id put into redux state. cart id is useful for if another product(single product or product variants) is add to cart , we call cart update API so that, those products are add to the same cart in backend with that same cart Id.

2. Guest user increase or decrease a specific product quantity and delete a specific product, user update these cart items in cart page.

3. Guest user fill the address form, after that Place order the cart items. Razor pay pop up will open up. On the razor pay pop up, different payment methods are available. Guest user will choose payment method and submit payment details as input. Then after check out API calls happens. After payment successfully it will return order Id.

4. Registered user add products (single product or product variants) with quantity to the shopping cart for first time from Feature products (home page),Shop page from Quick view or product details page.

Result: Cart record will create in backend, get cart id and other fields from API response. This cart id put into redux state. cart id is useful for if another product (single product or product variants) is add to cart , we call cart update API so that, those products are add to the same cart in backend with that same cart Id.

5. Registered user increase or decrease a specific product quantity and delete a specific product, user update these cart items in cart page.

6. Registered user fill the address form, after that Place order the cart items. Razor pay pop up will open. On the razor pay pop up, different payment methods are available. Guest user will choose payment method and submit payment details as input. Then after check out API calls. After payment successfully it will return order Id.

7. Guest cart to Registered user cart Items conversion (Registered user don't have cart in the backend): Guest user add products(single product or product variants) with quantity to the shopping cart from Feature products (home page),Shop page from Quick view or product details page. After login with registered email, these guest cart items (products) convert into user cart items in the backend. This scenario applicable in Login page and Check out page login.

8. Guest cart to Registered user cart Items conversion (Registered user have empty cart or cart items in the backend): Guest user add products(single product or product variants) with quantity to the shopping cart from Feature products (home page),Shop page from Quick view or product details page. After login with registered email, these guest cart items (products) convert into user cart items if user cart is empty in the backend. User cart have cart items (product), in that cart a specific product (single product or product variants)(quantity:9) exist. If Guest user add that product (single product or product variants)(quantity:3) to cart. After login with registered email, these guest cart items (products) convert into user cart items in the backend. Guest user added product (single product or product variants)(quantity:3) update in the user cart with same quantity(quantity:3). This scenario applicable in Login page and Check out page login.

9. After login, if login users have cart items in the backend, it will show cart items on mini cart. Otherwise, it will show empty mini cart.

10. Redux cart, mini cart and backend cart scenario: Specific product (single or variants) (quantity: 3) exists in the redux cart, mini cart and backend cart. if we add that same product(single or variants) from quick view or product details page with some quantity(example: 5). this quantity(5) is add to previous cart product quantity(3) (redux cart , mini cart and backend cart) and updated that product quantity(5+3=8) in carts. This scenario applicable to Guest user and Registered User shopping.

11. Registered user cart scenario: Specific product (single or variants) (quantity: 5) exists in the Registered user cart. if we add that same product(single or variants) from quick view or product details page with some quantity(example: 3) as Guest user. After Login register user, that product is update with this quantity (3) in the registered user cart.

12. Guest user adds some products (single or variants) to cart. Close that window, then that session is close. After open the window, Guest user can proceed same order management functional flow like update product quantity, remove product from cart, checkout that cart also.(in this process backend operations happen).

13. Guest user add specific product (single or variants) (quantity: 5) to cart. Close that window, then that session is close. After open the window, Guest user can add some products (single or variants) to that same cart. If Guest user add a specific product (single or variants) (quantity: 3), that product (single or variants) (quantity: 5) is already existing in the cart, cart is update with product (quantity: 5+3=8). (In this process backend operations happen).

14. Guest user or Registered user remove products from cart page or mini cart. These products removed from backend cart, redux cart and mini cart. (In this process backend operations happen).

➢ **Wishlist Scenarios**

1. Guest user add product to the wishlist for first time from Feature products (home page), Shop page from Quick view or product details page. Wishlist record will create in backend, get wishlist id and other fields from API response. This wishlist id put into redux state. wishlist id is useful for if another product is add to wishlist , we call wishlist update API so that, those products are add to the same wishlist in backend with that same wishlist Id.

2. Registered user add product to the wishlist for first time from Feature products (home page), Shop page from Quick view or product details page. Wishlist record will create in backend, get wishlist id and other fields from API response. This wishlist id put into redux state. wishlist id is useful for if another product is add to wishlist , we call wishlist update API so that, those products are add to the same wishlist in backend with that same wishlist Id.

3. Guest wishlist to registered user wishlist Items conversion: Guest user adds products to the wishlist from Feature products (Home page), Shop page from Quick view or product details page. After login with registered email, these guest wishlist items (products) convert into user wishlist items in the backend. This scenario only applicable for Login.

4. Guest user or Registered user remove products from wishlist in wishlist. These products removed from redux wishlist also.