**Ecommerce Application**

**Important Instructions:**

1. Please read the document thoroughly before you code.
2. Develop this application using ReactJs with the following softwares

NodeJS

Visual Studio Code

Internet explorer/Google Chrome

**Business Scenario**

It’s a well-known fact that e-commerce has eaten up the mainstream industry. Applications for it has made it easier and better for people to involve themselves in buying and selling online. The idea here is to create a simple e-commerce application that will be industry-specific and not a big marketplace. Expose the product list to the user. The user will be able to add/remove products from a product list to/from a shopping cart and to place an order and view bill.

**Application UI Layout**

App Component

Menu Component

Placeholder for displaying other components

Figure 1: Application UI Layout

**Architecture**

Json file

REST API

React Front-End

Figure 2: Architecture Diagram

# **Project structure**

1. Create a new ReactJS application
2. Create a sub-folder with-in src named as **“components”**
3. Each component must be created in a separate **<componentname.js>** file with-in **“component”** sub-folder.
4. Create a sub-folder within src names as “**services**”
5. In App.js, add the routing for the components.
6. Create a service called **EcommerceService in Ecommerce.js** file through which the components will interact with each other.
7. **Products.json** file will be provided which contains the product details.

**EcommerceService**

Add the following methods in the EcommerceService.js

**getProducts()**

This method is to fetch the product details from json file and display to the user.

**addProduct(id)**

This method is to add the product to the cart selected by the user.

**removeProduct(id)**

To remove the product fromm the cart

**viewCart()**

To view the cart items of the user.

**showBill()**

To display the bill amount.

**placeOrder()**

To place order.

# **MenuComponent**

1. Create MenuComponent inside **“menucomponent.js”**, in the components sub-folder.
2. MenuComponent shouldn’t have any state and must be created as a functional component instead of class-based component.
3. MenuComponent will be used for display a list of selection to the user.
4. On clicking each menu item an appropriate react component must be displayed below the menu component inside the placeholder
5. For an user menu component will display the following menu items
   * 1. Products
     2. Cart
     3. Bill
6. Along side the menu items menu component will also display the application name
7. Use <nav> , bootstrap navbar classes and <Link> to create the given menu.

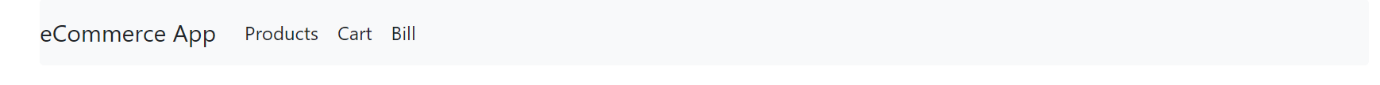


Figure 3: Menu Component

# **ProductListComponent**

1. Create **ProductListComponent** with in **components** folder inside the **“ProductListComponent.js”** file
2. ProductListComponent should be a class component. It can receive the required data through the props and maintain its own state.

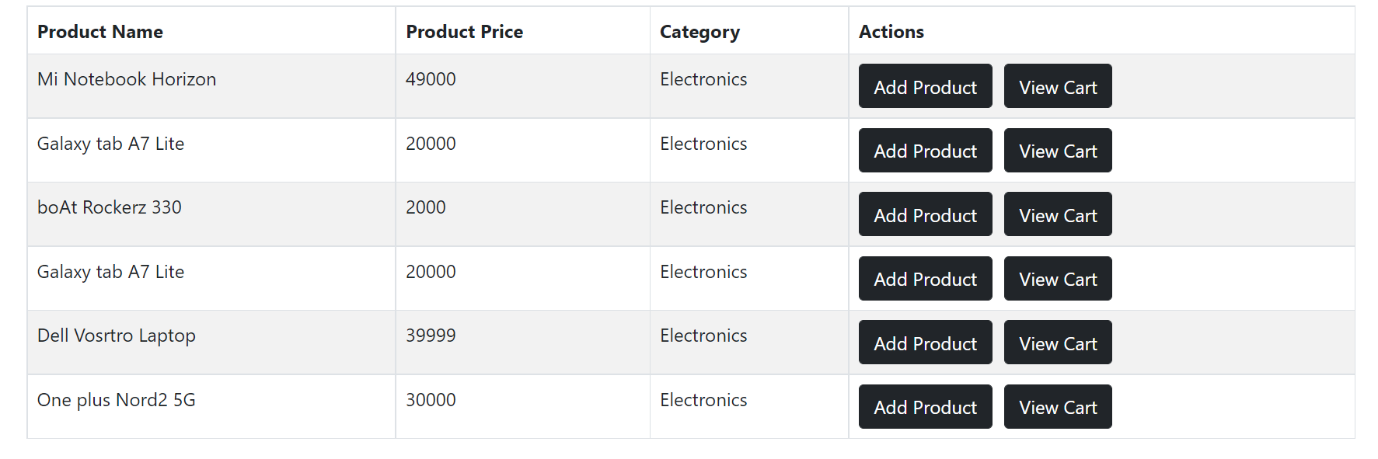


Figure 5: ProductListComponent

1. ProductListComponent will call getProducts() method of Ecommerce Service and fetch the product data from products.json file and render it as a list .
2. Display the list using map method.
3. Ensure that each item in the list has a unique key and list is created using the map method.
4. With the list of products, Add Product and View Cart buttons are displayed.
5. If user clicks Add Product button, the product should be added to the cart. This can be implemented by calling the addProduct(id) method of EcommerceService where selected product id is passed as parameter.
6. If user clicks, View Cart button, cart items should be displayed. This can be implemented by calling viewCart() method of EcommerceService.

# **CartComponent**

1. Create a new component inside **components** folder with the name as CartComponent in a file named CartComponent**.js.**
2. CartComponent will call the **viewCart()** method of EcommerceServce to get the list of products added to cart of the user.

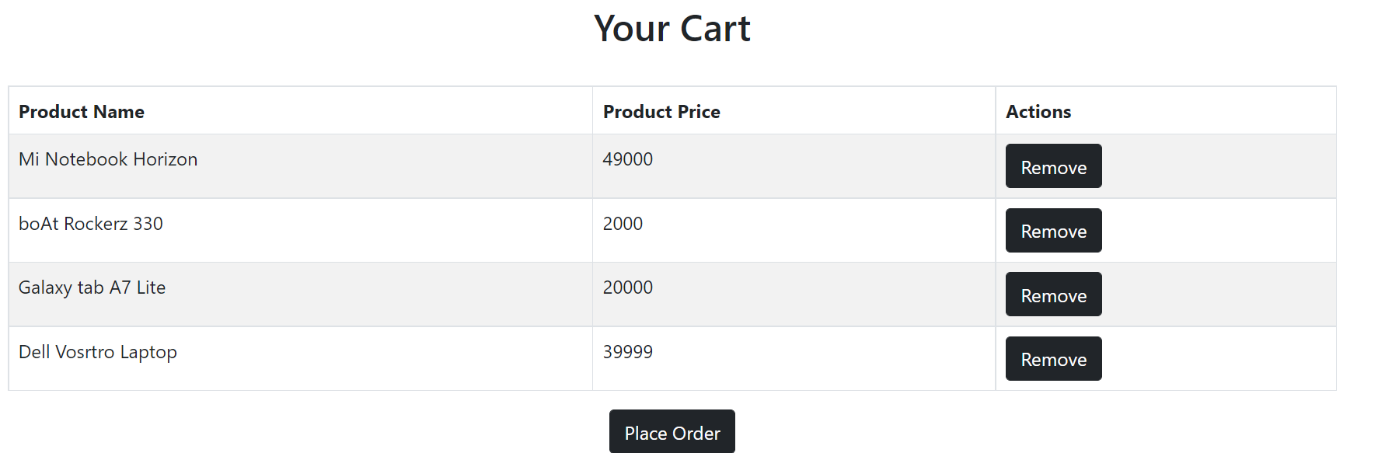


Figure 6: CartComponent

1. CartComponent will have a “Remove” button which will allow the current user to remove the items from the cart list.
2. On clicking the Remove button inside CartComponent, call to viewCart() method of EcommerceService must be made.
3. After successful removal of item, “Item removed successfully” should be displayed to the user.
4. To confirm the order, user should click the Place Order button.
5. Once order is placed, cart items should be updated.

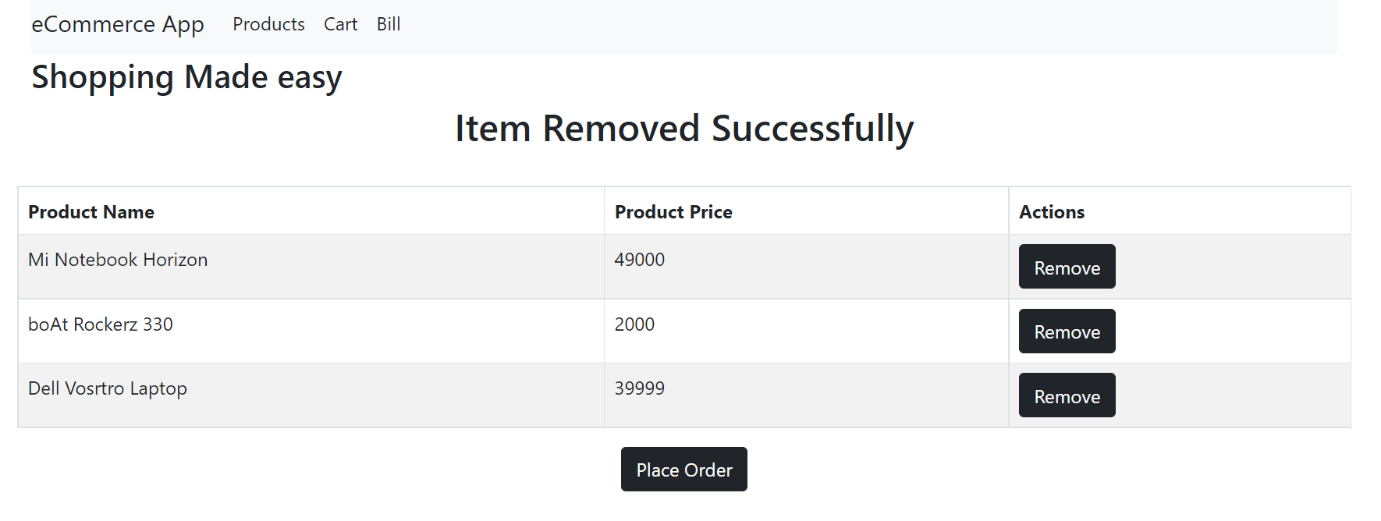


Figure 7: CartComponent

1. If there are no products added to the cart list display the message as “No items added to the cart”

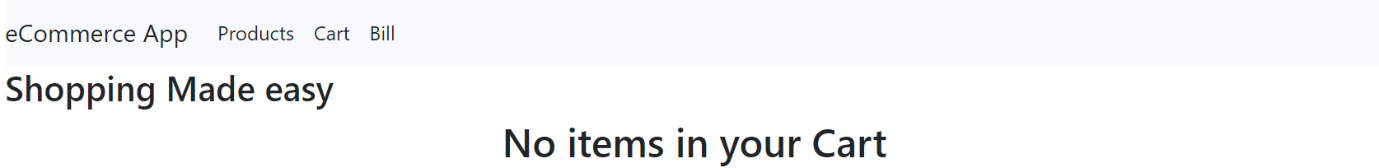


Figure 8: CartComponent

# **BillComponent**

1. Go to **components** folder and create a new component named **BillComponent** inside the file **BillComponent.js**
2. This component will render the bill for the user.To do this, call showbill() method of EcommerceService.
3. Now the user can make payment.

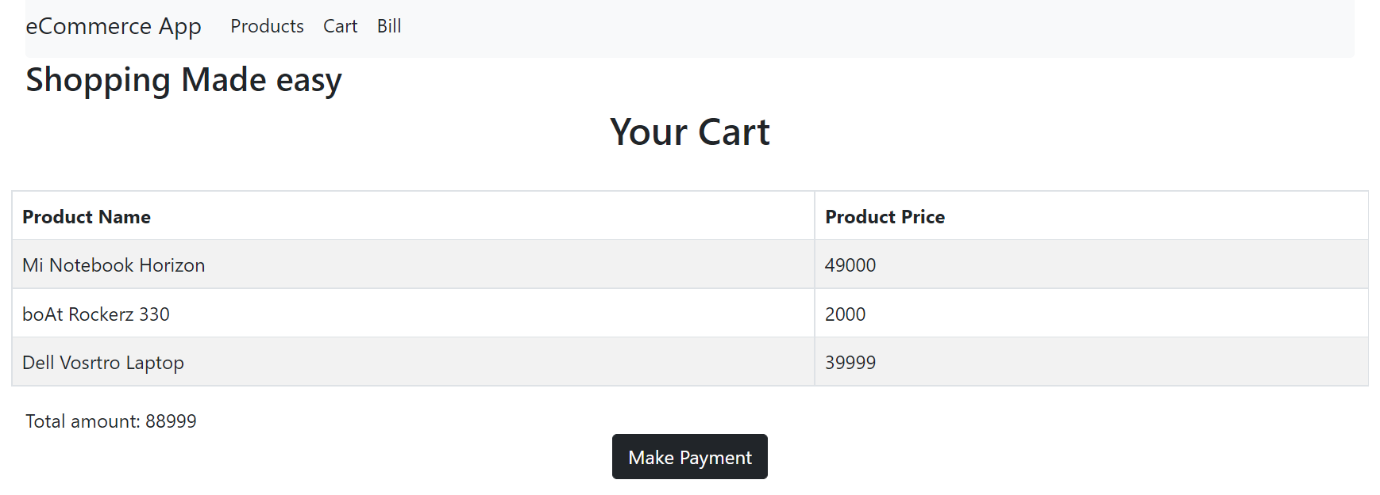


Figure 9: BillComponent

1. If Make Payment button is clicked, display the message “Payment made” to the user.