

Documentation File by Group 7

**A report on E-Commerce website
“ShopForHome”**

Group Members :

**Ujjawal Rawat
Anuprash Gupta
Utsav Singh
Gaurav Joshi
Ganti Pavan Kumar**

ABSTRACT

In today's generation, most people are using technology for leading their lives and fulfilling their daily needs. In this generation most of us use E-commerce websites for shopping for clothes, groceries, home décor and electronics .

We have developed one E-commerce web application by using **MERNstack** technology as it contains **MongoDB, Express.JS framework, React.JS library, and Node.JS platform.**

This application is fully functional with different views for user and admin and it also has integrated with a payment gateway for checkout. By using this website we can buy different types of t-shirts and we can choose different styles of t-shirts based upon customer interests. In this project, we can add different products and can delete them also.

We have developed administrative functions for the website such as create a product, create categories, Admin dashboard, Manage products, Manage categories. For customers, they can quickly add their items to the cart. Based on the items in the cart then the bill gets generated and the customer can pay by using stripe.

This project is mainly divided into two main categories: The Administrators and the Customers/Users. The store manager and the staff members operate as the administrators. They can add, edit, update products or, delete products

The User is only able to browse the online shop and add a product to the cart. The user is limited to the use of the shop.

INTRODUCTION

We all know that technology has become an essential tool for online marketing these days. If we look all over the world, most of the people are showing interest to buy things online. However, we can see that there are many small shops and grocery stores selling their things offline. With this type of selling most of us will face bad experiences. For instance, in some shops the seller has the product to sell in the offer but the buyer may not know about it, or the customer may need the product urgently then he will go to the shop, but the product is out of stock, in that case, he will face a bad experience. Moreover, in online shopping customers can select a wide range of products based upon their interests and their price. Also, one can compare prices also from one store to another by using online shopping . By encountering all the problems and weaknesses of the offline shopping system, creating an Ecommerce web application is necessary for searching and shopping in each shop. These days we have seen so many e-commerce websites are created like Flipkart, Amazon, Myntra and one can easily buy their necessary products by using these websites. By using these types of websites one can buy their products by staying in their home. Eventually, we can see the difference between the prices of products as well as if we see the cost of the product will be slightly high in offline shopping when compared to online shopping. For creating these types of E-commerce web applications MERN stack will be the best option that can help us create the most effective and powerful web applications.

PROBLEM STATEMENT

The purpose of this project is to make a web application which will be easier to find interesting Home decoration items . This E-commerce web application admin can add some categories like Sofa, Dining, Bed and Mattresses etc. which will by attracting customers. Customers also can easily search for their favorite goods. They can also buy them easily by just adding to the cart and they can increase or decrease by clicking on the "+" sign and "-" sign. After adding they can check the total amount of the thing which has been added to the cart. A successful payment gateway is enabled so payment can be done by debit card, credit card, and net banking.

E-COMMERCE

E-commerce is also known as electronic commerce, it is the process of buying and selling products through the internet, and also the transfer of money and the data to complete the process of buying or selling a product . In the early days, e-commerce was not that famous but after increasing the use of mobile phones everyone was choosing the interest in buying things online so, it became popular.

Types of E-commerce: There are mainly four types of E-commerce business models

1. B2C (Business to consumer). It is a business based upon the internet, By using this model we can sell products to the end-user.
2. B2B(Business to business) It is a business that will happen between large companies, organizations, and businesses, these days most of e-commerce falls in this category.
3. C2B(Consumer to Business) This type of E-commerce allows that the individuals can sell their goods to Companies In this type of E-commerce, individuals will assign work to complete in the given time through websites

or some electronic medium, in this type, consumers can set their price tag to their work The best example of this type of work is freelancing.

4. C2C(Consumer To Consumer) This type of E-commerce connects consumers to consumers to exchange their goods and make their money by charging transactions it will motivate buyers and consumers .

ADVANTAGES:

1. A HUGE MARKET: E-commerce will give you an option to reach customers all over the world, you can buy anything that you want to from your home. Nowadays people are used to doing shopping with their mobiles only. So, it will add an advantage for E-commerce.

2. WIDE PRODUCT VARIETY: In this large world, customers can buy different types of products from different places, we can buy electronics from Russia, shoes from Japan, clothes from London and good old international products, the depth and advantage of E-commerce is uncountable.

3. TRACKING: If you order any product in e-commerce you will get the details of where the product is shipping and when it will reach you, you can cancel the product also if you don't like that product after order.

4. LESS PRICE: In E-commerce websites, we can compare the prices of products from one website to another website. In that way, we can easily know that where we will get the product for a lesser price and we will get the basic idea that how much money we can spend to buy a particular product.

5. MORE CHANCES TO “SELL”: In the physical stores' merchant can not give the full information about the product, but in the E-commerce store customer can get full information about that product and he can see the reviews of another customer who bought that product previously. In this way, if the product is good it will get more chances to get that product .

DISADVANTAGES:

- 1. SECURITY:** Even though E-commerce providing more benefits to customers. People are having fear about giving their data to website owners, so it will give security 3 International Journal for Modern Trends in Science and Technology issues when we use e-commerce websites for shopping.
- 2. TAX:** If we want to buy a product from e-commerce we will have to pay taxes like GST, it will be different from place to place based upon your distance between you and the product, it will be high when compared to buy in physical stores.
- 3. DELAYING IN DELIVERY:** If we order a product in e-commerce, the order may or may not reach you on time, this is one of the common problems in e-commerce, it will take time-based upon your distance between you and the organization from where you ordered.
- 4. TECHNOLOGY COST:** To create a website will take so much amount, one needs to spend so much money on building an e-commerce website because he needs to check all the possibilities and provide a good security system to that website.

RESEARCH AND DEVELOPMENT

There are so many applications to build a web application and, in this research, we have taken MERN technologies to use for building a web application.

MERN: MERN stands for **MongoDB Expressjs Reactjs Nodejs**.

These are the four technologies that help us to construct or to build this web application.

MongoDB:

It is an open-source cross-platform program. It comes under the NoSQL

database classification. It was a document-oriented database. It uses JSON format documents with optional Schemas.

- * Data Flexibility available means we can store every data in a separate file
- * Large data can be distributed into several connected applications
- * High speed of fetching of data possible because it only depends on indexing.
- * It is a horizontally scalable database so it can handle the data and make it easy to distribute to several machines.

NodeJS:

NodeJS is a runtime javascript environment that works outside the web page. It is mainly used for server-side applications. * NodeJS is open source and it is free of cost.

- * NodeJS uses asynchronous programming by default.
- * NodeJS will always store the data in only JSON format.

ExpressJS:

It is a famous Library in node.js used for routing. It has some methods like a router which help to do curd operations like put, get, post, and delete request

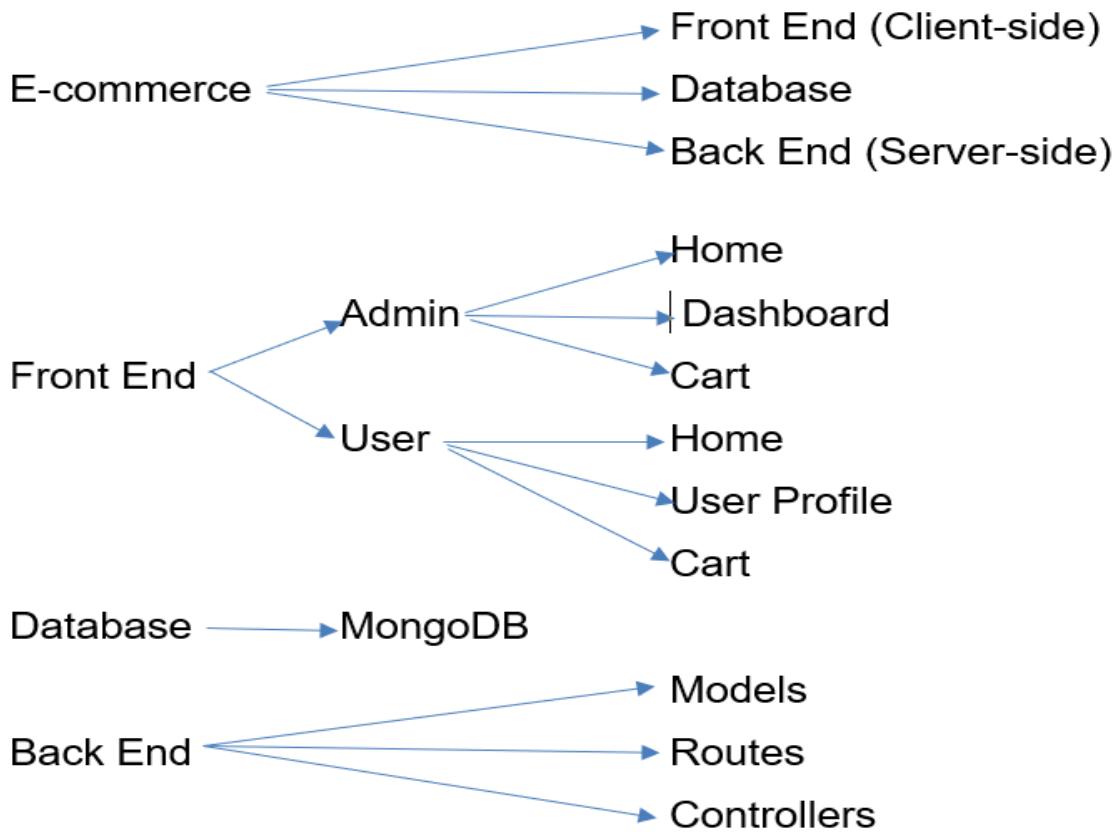
- * Robust routing
- * It will focus on high performance
- * It is an HTTP helper like it will redirect, catching.

ReactJS:

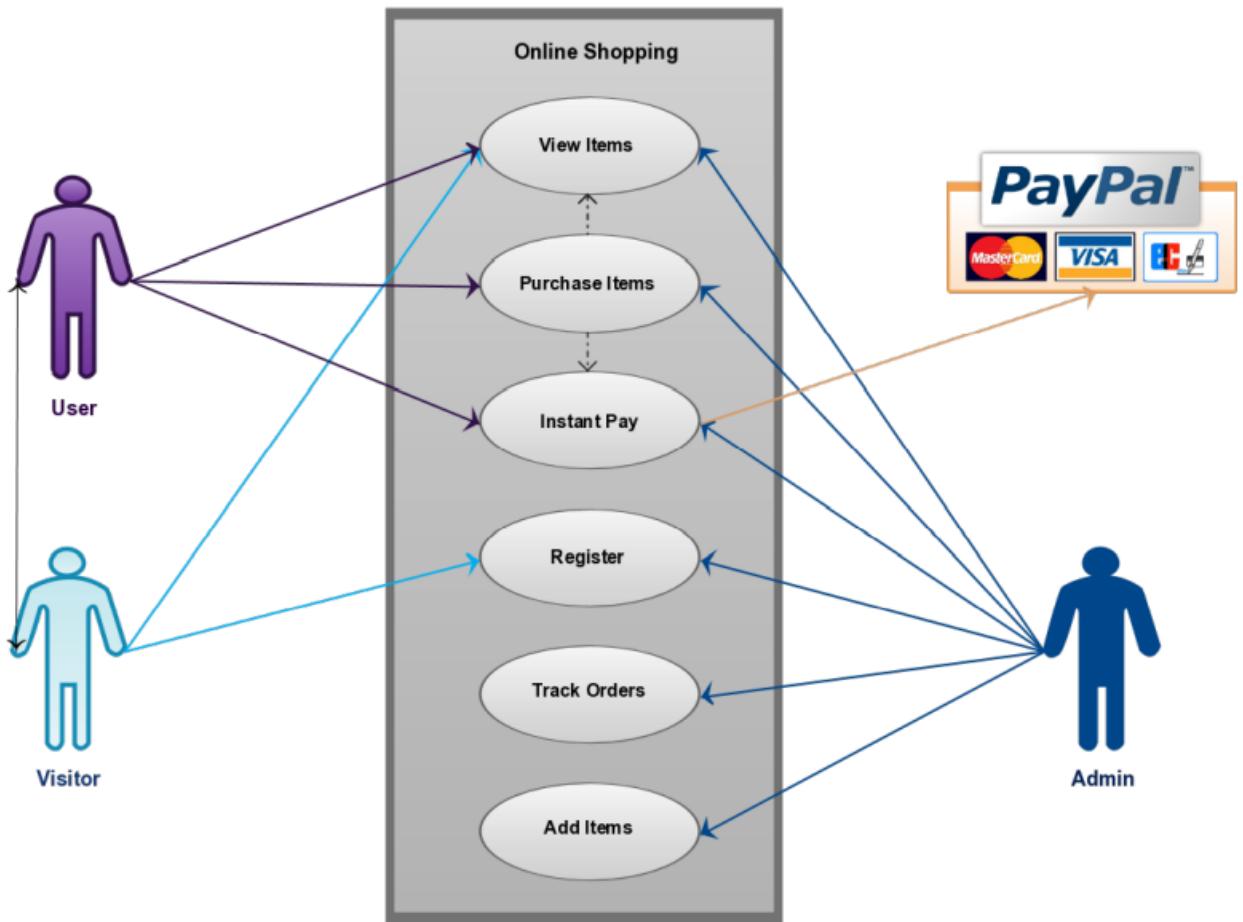
React is a library of javascript which is created by Facebook. React is mainly used as a UI which means Maximum it is used for client-side applications. There are so many libraries like React-dom, React-router-dom, and many more to help to build the frontend of any application.

- * The rendering of the page will be much faster than others because it has virtual Dom.
- * uses a stable code.
- * It has a strong community following .

WEB APPLICATION STRUCTURE:



Use Case Diagram:



OVERALL DESCRIPTION

APPLICATION AND DESCRIPTION

Overview of the Various Parts

- Any member can register and view available products.
- Only registered members can purchase multiple products regardless of quantity.
- There are two roles available: User and Admin.
- Visitors can view available products
- Users can view and purchase products.
- An Admin has some extra privilege including all privileges of visitor and user. Admin can add products, edit product information and add/remove products.
- Admin can add users, edit user information and can remove users.

Administrators Detailed Attribute

- **Admin register:** The administrator needs to register before they can have access to the core data of the shop.
- **Admin login:** The admin logs in and can view, add products, manage customers.
- **Admin Edit:** The Admin can make changes to the shop such as delete customers, add a customer or upload new products.
- **Manage Customer:** The admin has the authority to delete or add a customer
- **Bulk Upload:** The admin can upload a CSV file for products details
- **Discount Coupons:** The admin can generate random discount coupons for the users.

Customer Detailed Attribute

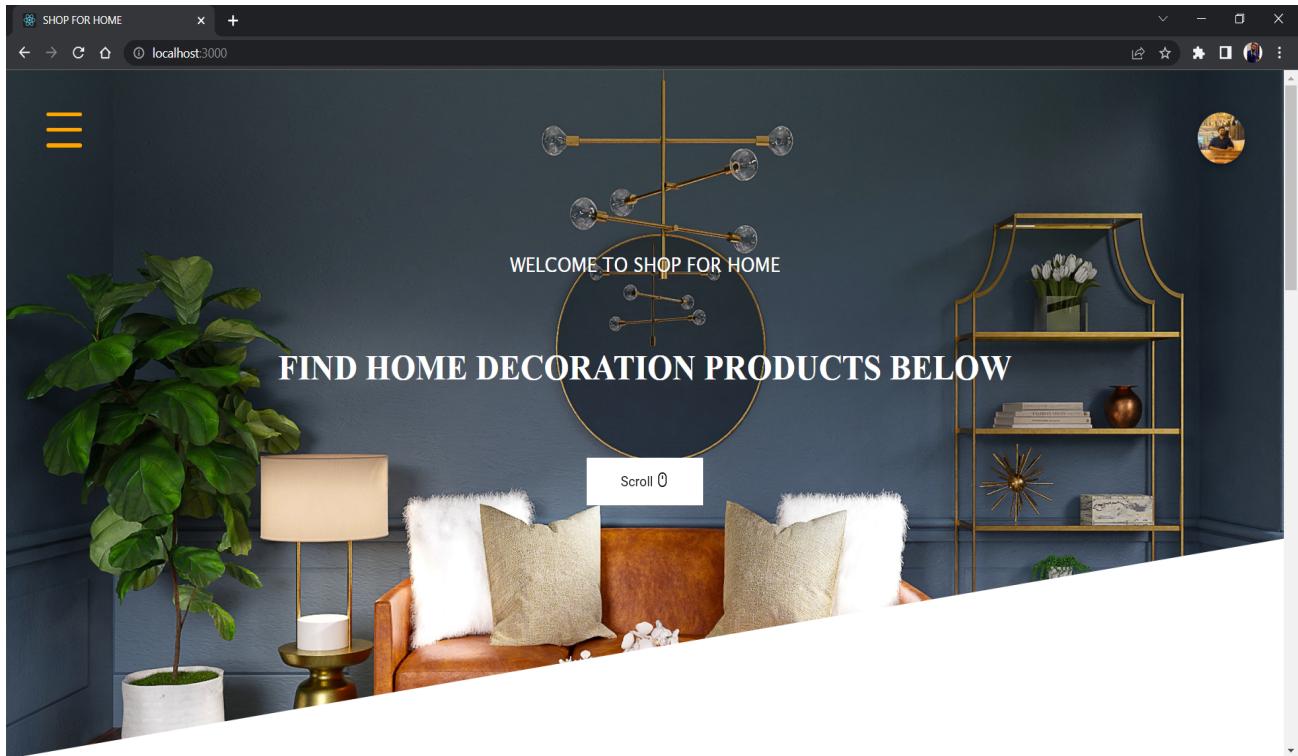
- Sign up: This refers to registering as a customer. The registered member has a lot of privileges associated with the shop when one becomes a customer.
- Login: After the user has registered, the user becomes a customer, and he or she can log in with their personal information.
- View: The customer can see all the products in the catalog and be able to look at the products and some features on the homepage.
- Update Cart: This refers to putting or removing products from a shopping cart

FRONT-END:

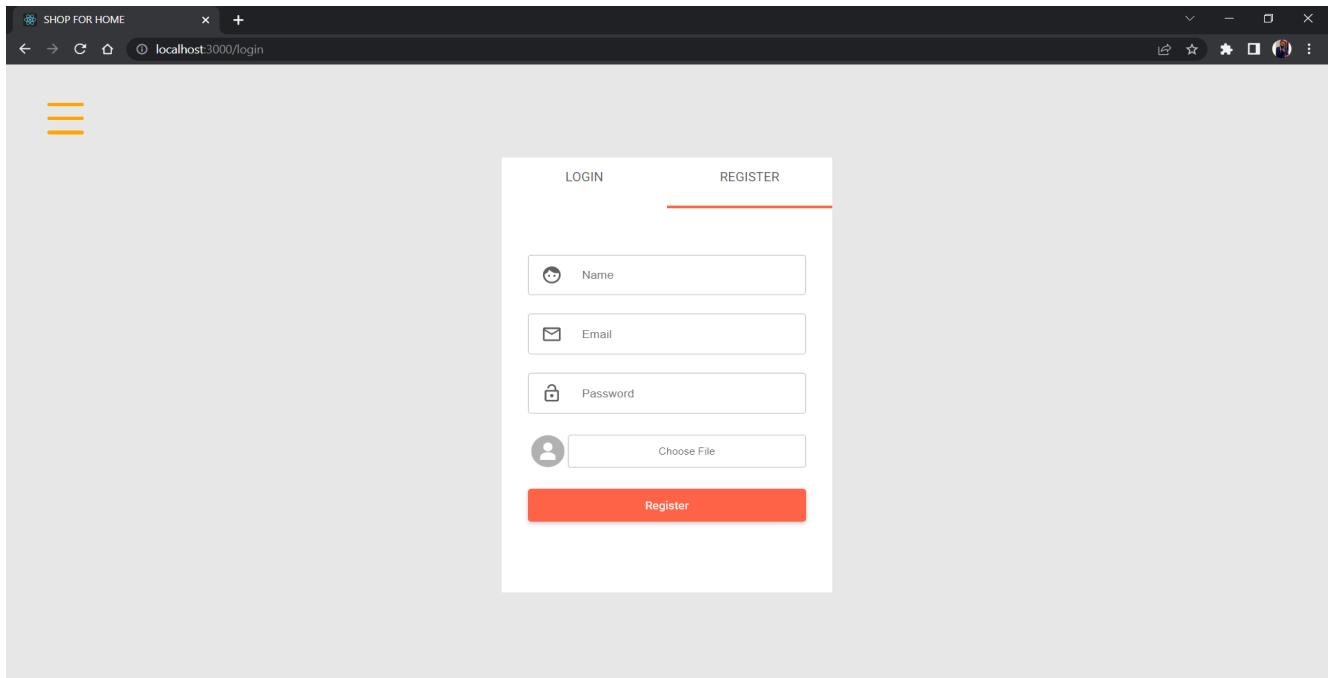
Consumer refers to customers and non-customers. These are individuals who visit the shop either to buy products or to browse. There are two categories of interfaces namely the Consumer and the Administrator interface. The administrator has higher authority over the customer in the shop. The admin can 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, 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 images, names, prices, product categories and product brands. The webpage has a registration link, login link, cart, company contact information.

Functionality For the Users:

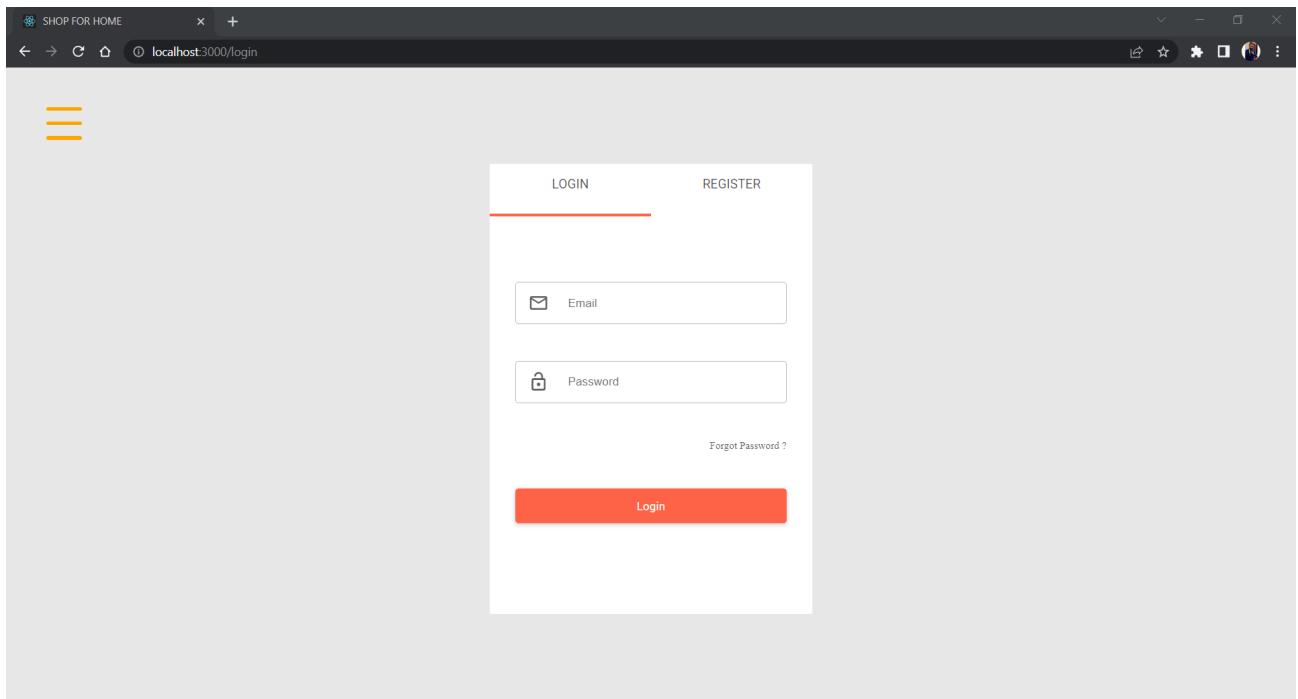
Home Page: Below is the Home of our Website which is also the landing page when the project is run.



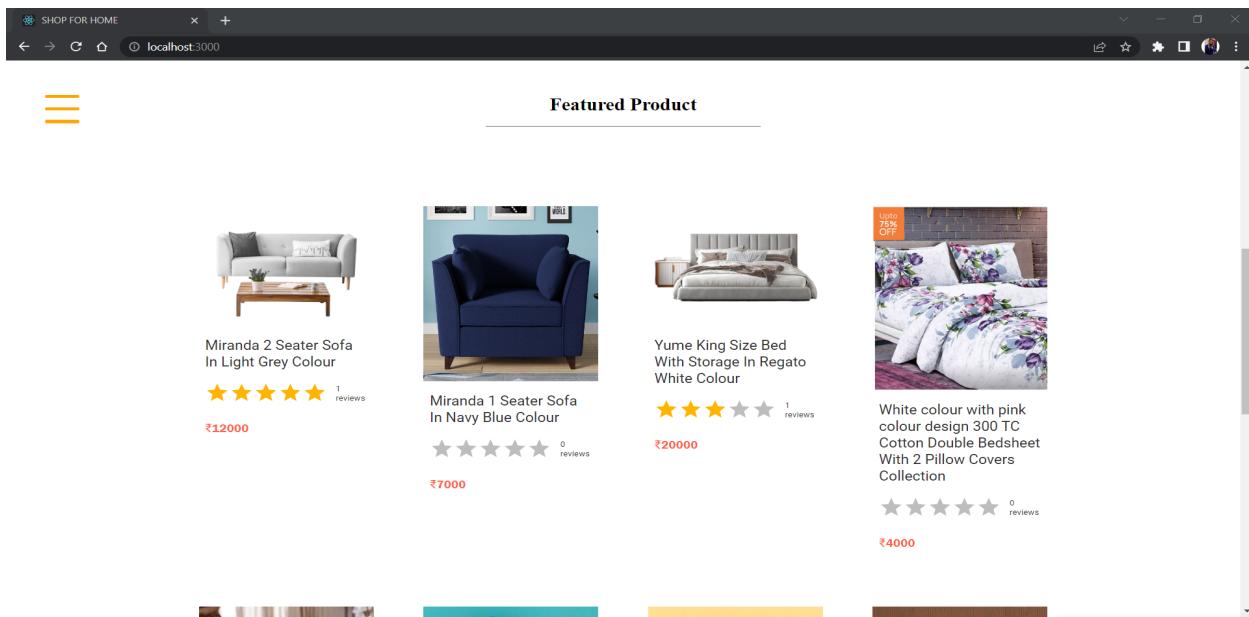
Register Page: User have to first register themselves from the register page shown below



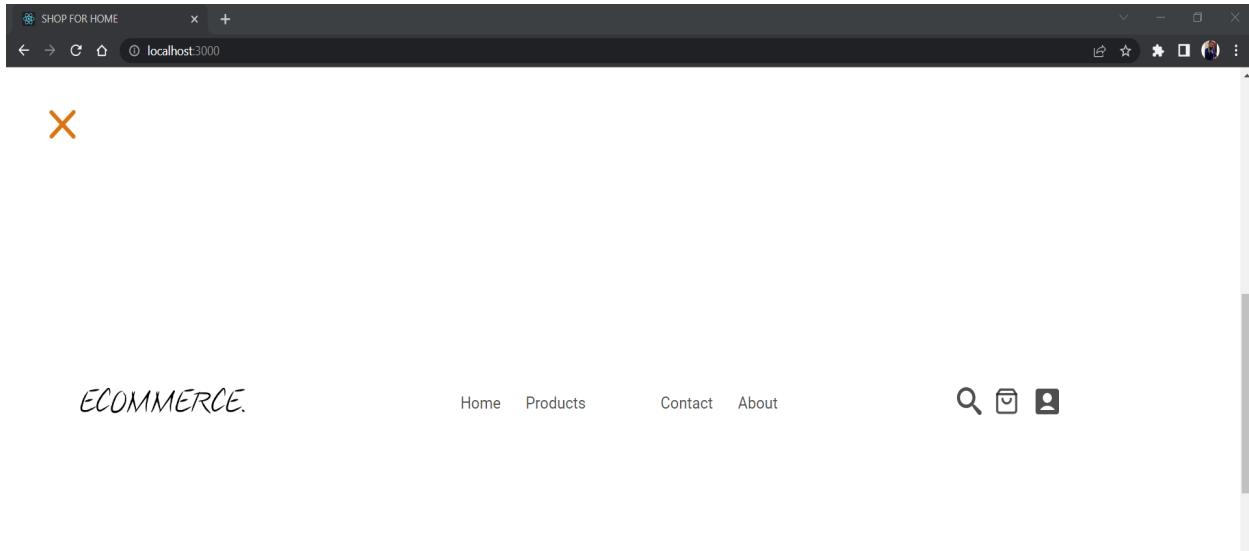
Login Page: If User is Register already than it will login with their email and password



Featured Products: In this the products which are main highlight



Navbar: You can use Navbar for rendering in products,home ,about pages etc.



Footer:



Products and Filters: In this All products are shown and in left hand side their is filters so user can filter according to their need like by rating, price and categories.

Products

Price

Categories

- Sofa
- Table and Chair
- Beds and Mattresses
- Dinning
- Wall Accents
- Decor
- Lighting
- Carpets

Rating Above

Miranda 2 Seater Sofa In Light Grey Colour

5 stars 1 reviews

₹12000

Miranda 1 Seater Sofa In Navy Blue Colour

4 stars 0 reviews

₹7000

Yume King Size Bed With Storage In Regato White Colour

4 stars 1 reviews

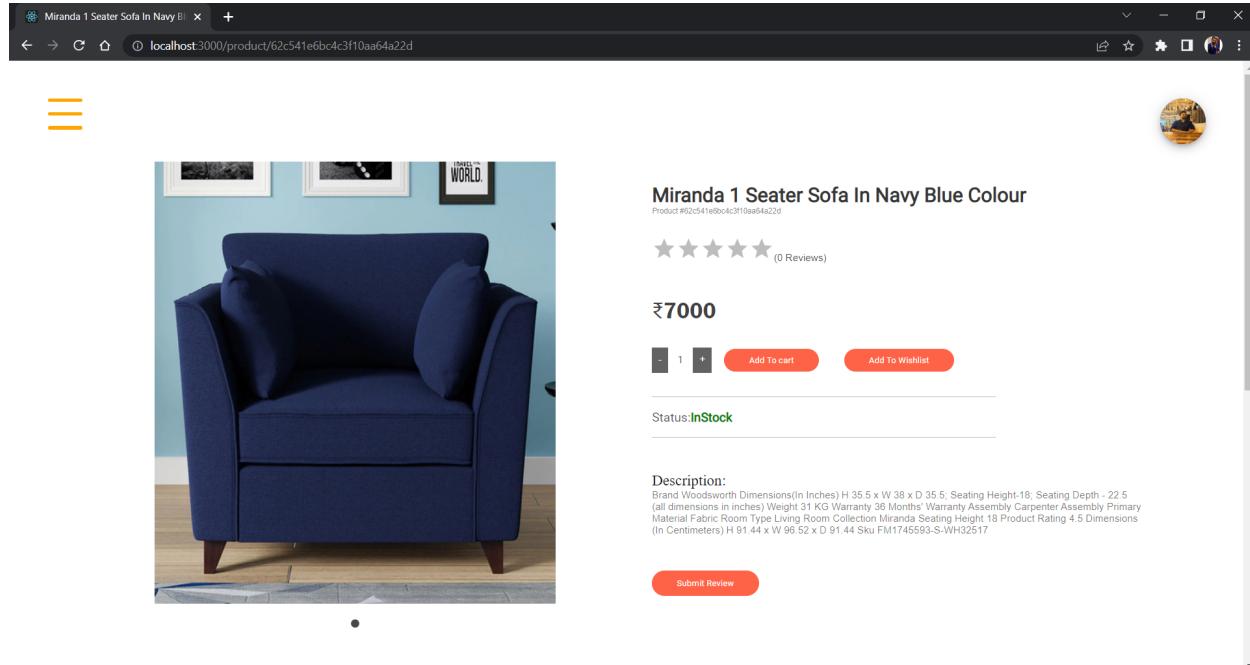
₹20000

White colour with pink colour design 300 TC Cotton Double Bedsheet With 2 Pillow Covers Collection

5 stars 0 reviews

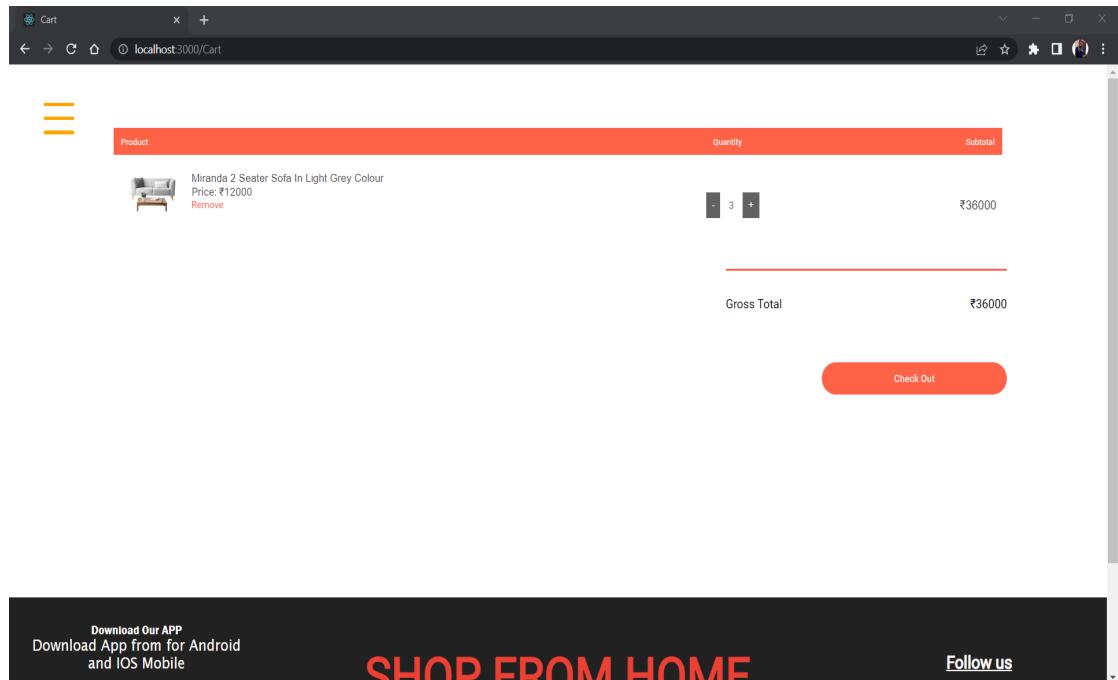
₹4000

Product Details: User can see product details by clicking on particular product and they can review the product also and see the reviews of other users also.



The screenshot shows a product page for a 'Miranda 1 Seater Sofa In Navy Blue Colour'. At the top, there's a navigation bar with a search icon, user profile, and cart icon. Below the header, a large image of the sofa is displayed against a blue wall with framed pictures. To the right of the image, the product title 'Miranda 1 Seater Sofa In Navy Blue Colour' is shown along with its product ID. A star rating section indicates 0 reviews. The price is listed as ₹7000. Below the price are quantity selection buttons (minus, plus, and a value of 1) and 'Add To cart' and 'Add To Wishlist' buttons. A status indicator shows 'Status: InStock'. The description section provides brand information, dimensions (H 35.5 x W 38 x D 35.5), weight (31 KG), warranty (36 Months), assembly (Carpenter Assembly), material (Primary Material Fabric), room type (Living Room), collection (Miranda), seating height (18 cm), product rating (4.5), and dimensions (H 91.44 x W 96.52 x D 91.44 cm). A 'Submit Review' button is located at the bottom of the description.

Cart: From the product page users can add their favorite products to the cart for further process.



The screenshot shows a shopping cart page with a single item: 'Miranda 2 Seater Sofa In Light Grey Colour' priced at ₹12000. The quantity is set to 3, resulting in a subtotal of ₹36000. The page includes a 'Check Out' button. At the bottom, there's a dark footer bar with links for 'Download Our APP', 'Download App from for Android and IOS Mobile', 'SHOP FROM HOME' in large red text, and 'Follow us' with social media icons.

Wishlist: User can add and see their favorite products in the wishlist.

The screenshot shows a browser window titled "Wishlist" at the URL "localhost:3000/wishlist". The page has a header with a profile picture and a sidebar menu. The main content area displays two products in a grid format. Each product card includes an image, the product name, price, a "Remove" button, and a quantity selector (minus, plus, current value 1). The products listed are "Yume King Size Bed With Storage In Regato White Colour" and "Black Glazed 300ml (Set of 6) Coffee Mug".

Checkout: In checkout user will fill their shipping address and confirm the order then fill the payment details so the product is order.

The screenshot shows a browser window titled "Shipping Details" at the URL "localhost:3000/shipping". The page has a header with a profile picture and a sidebar menu. It features three tabs: "Shipping Details", "Confirm Order", and "Payment". The "Shipping Details" tab is active, showing a form with fields for Address, City, Pin Code, Phone Number, Country (set to India), and State (set to Uttarakhand). A "Continue" button is located at the bottom of the form.

Discount coupon: User can add the coupon for discount.

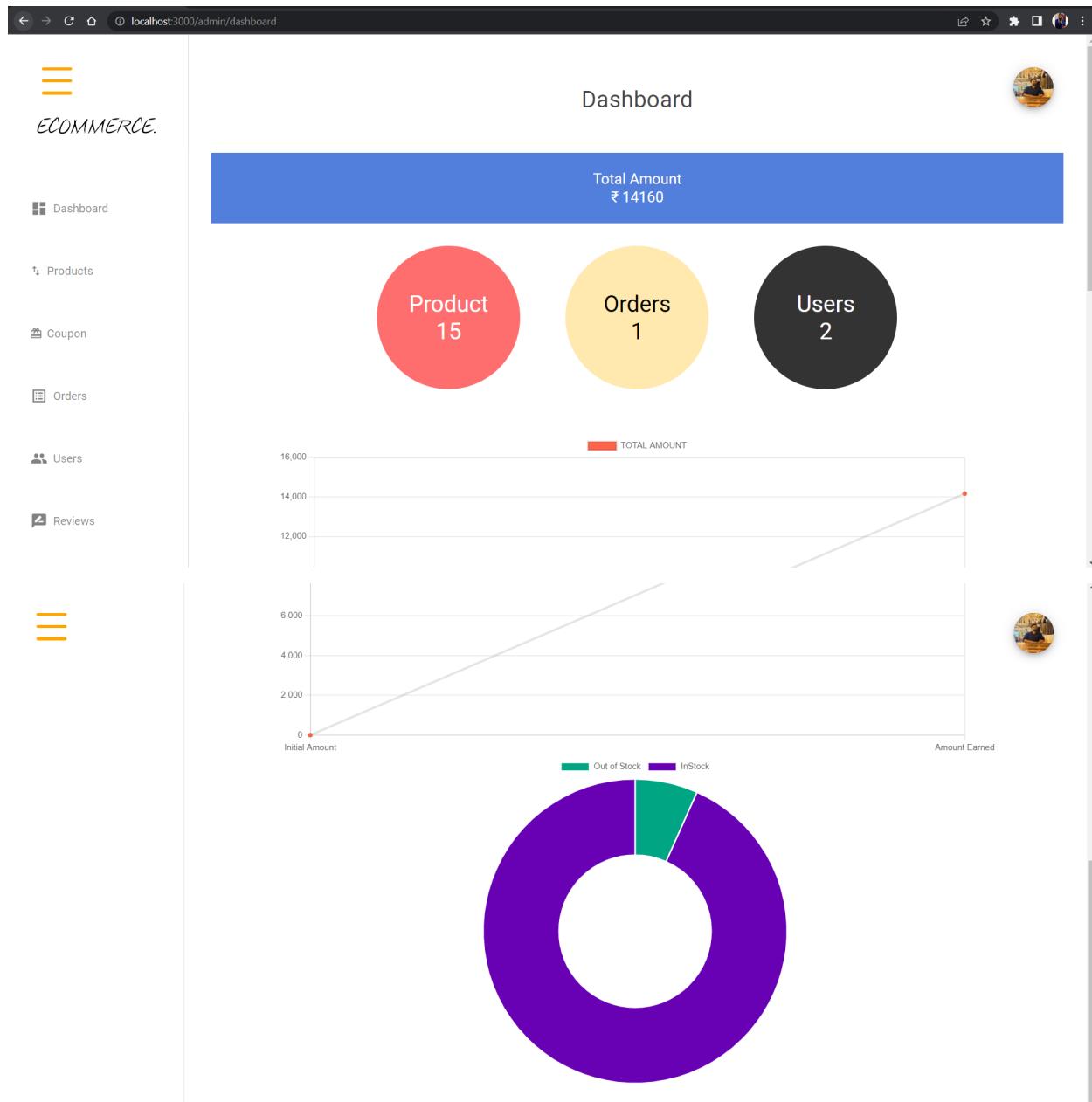
The screenshot shows a web browser window titled "Confirm Order" with the URL "localhost:3000/order/confirm". The page is divided into several sections:

- Shipping Details:** Shows a delivery truck icon and a placeholder for "Shipping Details".
- Confirm Order:** Shows a checkmark icon and a placeholder for "Confirm Order".
- Payment:** Shows a credit card icon and a placeholder for "Payment".
- User Profile:** Shows a circular profile picture of a person.
- Shipping Info:** Displays the following shipping details:
 - Name: Ujjawal Rawat
 - Phone: 7060866159
 - Address: lane no-09 someshwar nagar ,rishikesh, , lane no -09, Rishikesh, UT, 249201, IN
- Your Cart Items:** Shows an image of a light-colored sofa with two pillows (one dark grey, one white with a geometric pattern). Below the image, the text "Miranda 2 Seater" and "1 X" is displayed.
- Order Summary:** A table showing the breakdown of costs:

SubTotal:	₹12000
Discount:	- ₹500
Shipping Charges:	₹0
GST:	₹2160
Total:	₹13660
- Coupon Input:** A text input field containing "SHOP50" and a red "Apply" button.
- Proceed To Payment:** A red button at the bottom right.

Admin functionalities:-

Admin Dashboard: In admin dashboard , admin has full control for the stocks and sales report and also the CRUD operation to products and user. And also admin can upload bulk product and can create coupons which user will apply at the time of payment.



CRUD IN PRODUCTS BY ADMIN

Screenshot of the Admin Panel showing the 'All Products' page.

The URL is localhost:3000/admin/products.

The sidebar menu includes:

- Dashboard
- Products
 - All
 - Create
 - Bulk Upload
- Coupon
- Orders
- Users

The main content area displays a table titled "All Products".

Product ID	Name	Stock	Price	Action
62c54146bc4c3f10aa64a219	Miranda 2 Seater Sofa In Light Grey Colour	7	₹12000	
62c541e6bc4c3f10aa64a22d	Miranda 1 Seater Sofa In Navy Blue Colour	7	₹7000	
62c54293bc4c3f10aa64a237	Yume King Size Bed With Storage In Regato White Colour	6	₹20000	
62c54305bc4c3f10aa64a241	White colour with pink colour design 300 TC Cotton Double Bedsh...	0	₹4000	
62c54363bc4c3f10aa64a24b	Black Glazed 300ml (Set of 6) Coffee Mug	5	₹1000	
62c543cebc4c3f10aa64a255	Ebony Black 7 Inch (Set of 4) Ceramic Quarter Plate By Vareesha	5	₹600	
62c54431bc4c3f10aa64a25f	Florito Solid Wood Book Shelf in Brown Colour	4	₹3000	
62c5448cbc4c3f10aa64a269	Sian Golden Abstract Gold Mild Steel Decorative Wall Mirror	10	₹3000	
62c544ccbcb4c3f10aa64a273	Line Metal Wall Clock (1X1ft) By WallCentre	6	₹4000	
62c5452abc4c3f10aa64a27d	Calus Ray Smart Wifi Rgb Ledstrip	10	₹2000	

Pagination: 1-10 of 15

CRUD USERS BY ADMIN:

Screenshot of the Admin Panel showing the 'All Users' page.

The URL is localhost:3000/admin/users.

The sidebar menu includes:

- Dashboard
- Products
- Coupon
- Orders
- Users
- Reviews

The main content area displays a table titled "All Users".

User ID	Email	Name	Role	Actions
62c5358afbc035edecebe	ujjwalrawat2000@gmail.com	Ujjawal Rawat	admin	
62caabeb42ff7edbfbe10bc	anuprashgupta2000@gmail.com	Anuprash	user	

MANAGE COUPON BY ADMIN:

The screenshot shows a web browser window titled "ALL COUPONS - Admin" with the URL "localhost:3000/admin/coupons". On the left, there is a sidebar with a logo and the text "ECOMMERCE.". Below the logo are several navigation items: "Dashboard", "Products", "Coupon" (with "All" and "Create" sub-options), "Orders", "Users", and "Reviews". The main content area is titled "All Coupons" and contains a table with two rows of data. The columns are "Coupon ID", "Code", "Type", "Discount", and "Action". The first row has a Coupon ID of "62caab50b42ff7edbfbe10ab" and a Code of "SHOP50", with Type "flat" and Discount ₹500. The second row has a Coupon ID of "62caab7ab42ff7edbfbe10b4" and a Code of "FIRST10", with Type "flat" and Discount ₹4000. Each row has a delete icon in the "Action" column.

Coupon ID	Code	Type	Discount	Action
62caab50b42ff7edbfbe10ab	SHOP50	flat	₹500	
62caab7ab42ff7edbfbe10b4	FIRST10	flat	₹4000	

MANAGE ORDERS BY ADMIN: Admin Can manage the orders like by setup the status of the order like shipped and deliver.

The screenshot shows a web browser window titled "ALL ORDERS - Admin" with the URL "localhost:3000/admin/orders". On the left, there is a sidebar with a logo and the text "ECOMMERCE.". Below the logo are several navigation items: "Dashboard", "Products", "Coupon", "Orders", "Users", and "Reviews". The main content area is titled "All Orders" and contains a table with two rows of data. The columns are "Order ID", "Status", "Items Qty", "Amount", and "Actions". The first row has an Order ID of "62c6961653feed4472303096" and a Status of "Processing", with Items Qty 1 and Amount ₹14160. The second row has an Order ID of "62cab3d6471026953138aad5" and a Status of "Processing", with Items Qty 1 and Amount ₹14160. Each row has edit and delete icons in the "Actions" column.

Order ID	Status	Items Qty	Amount	Actions
62c6961653feed4472303096	Processing	1	₹14160	
62cab3d6471026953138aad5	Processing	1	₹14160	

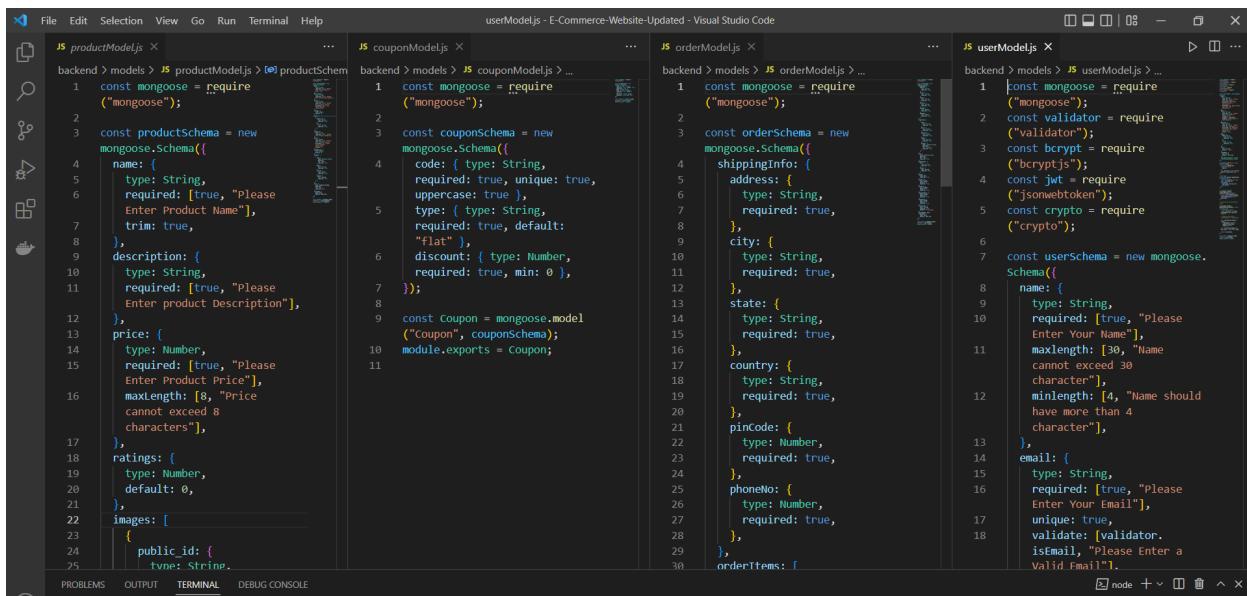
BACKEND:

Models:

Here we define the structure of the data that should in the database. By using some models which help to store the data in the database like mongoose, it is one of the famous libraries in NodeJS. Creating the schemas by mongoose with can mentions the names and type of the data.

Here we are showing the four models for our database:-

- a). productModel.js
- b). orderModel.js
- c). userModel.js
- d). couponModel.js



```

productModel.js
backend > models > JS productModel.js > (1) productSchema
1 const mongoose = require('mongoose');
2
3 const productSchema = new mongoose.Schema({
4   name: {
5     type: String,
6     required: [true, "Please Enter Product Name"],
7     trim: true,
8   },
9   description: {
10    type: String,
11    required: [true, "Please Enter product Description"],
12  },
13   price: {
14    type: Number,
15    required: [true, "Please Enter Product Price"],
16    maxLength: [8, "Price cannot exceed 8 characters"],
17  },
18   ratings: {
19    type: Number,
20    default: 0,
21  },
22   images: [
23     {
24       public_id: {
25         type: String,
26       }
27     }
28   ]
29 })
30 module.exports = mongoose.model('Product', productSchema);

couponModel.js
backend > models > JS couponModel.js > ...
1 const mongoose = require('mongoose');
2
3 const couponSchema = new mongoose.Schema({
4   code: { type: String,
5     required: true, unique: true,
6     uppercase: true },
7   type: { type: String,
8     required: true, default: "flat" },
9   discount: { type: Number,
10    required: true, min: 0 },
11 })
12 const Coupon = mongoose.model('Coupon', couponSchema);
13 module.exports = Coupon;

orderModel.js
backend > models > JS orderModel.js > ...
1 const mongoose = require('mongoose');
2
3 const orderSchema = new mongoose.Schema({
4   shippingInfo: {
5     address: {
6       type: String,
7       required: true,
8     }
9   },
10  city: {
11    type: String,
12    required: true,
13  },
14  state: {
15    type: String,
16    required: true,
17  },
18  country: {
19    type: String,
20    required: true,
21  },
22  pincode: {
23    type: Number,
24    required: true,
25  },
26  phoneNo: {
27    type: Number,
28    required: true,
29  }
30 }, {
31   orderItems: [
32     {
33       item: {
34         type: mongoose.Schema.Types.ObjectId,
35         ref: 'Product'
36       },
37       quantity: {
38         type: Number,
39         required: true
40       }
41     }
42   ]
43 })
44 module.exports = mongoose.model('Order', orderSchema);

userModel.js
backend > models > JS userModel.js > ...
1 const mongoose = require('mongoose');
2 const validator = require('validator');
3 const bcrypt = require('bcryptjs');
4 const jwt = require('jsonwebtoken');
5 const crypto = require('crypto');
6
7 const userSchema = new mongoose.Schema({
8   name: {
9     type: String,
10    required: [true, "Please Enter Your Name"],
11    maxLength: [30, "Name cannot exceed 30 character"],
12    minLength: [4, "Name should have more than 4 character"]
13  },
14   email: {
15     type: String,
16     required: [true, "Please Enter Your Email"],
17     unique: true,
18     validate: [validator.isEmail, "Please Enter a Valid Email"]
19   }
20 })
21 module.exports = mongoose.model('User', userSchema);

```

Routers:

All the work related to the routing of the pages was done here. ExpressJS is a popular library for routing. CRUD operations and routing-related code are saved in this folder.

Here we are showing the four routesfor our database:-

- a). productRoute.js
- b). orderRoute.js
- c). userRoute.js
- d). paymentRoutes.js

```

JS couponRoute.js
backend > routes > JS couponRoute.js > ...
1 const express = require('express');
2 const { newCoupon, getAllcoupons, deleteCoupon } = require('../controllers/couponController');
3 const router = express.Router();
4
5 const { isAuthenticatedUser, authorizedRoles } = require('../middleware/auth');
6
7 router.route('/coupons/:id').get(isAuthenticatedUser, getSingleCoupon);
8 router
9 .route('/admin/coupons')
10 .get(isAuthenticatedUser, authorizedRoles('admin'), getAllcoupons)
11 .post(isAuthenticatedUser, authorizedRoles('admin'), newCoupon);
12 router.route('/admin/coupons/:id').delete(isAuthenticatedUser, authorizedRoles('admin'), deleteCoupon);
13
14 module.exports = router;
15
16
17
18
19
20
21
22

JS orderRoute.js
backend > routes > JS orderRoute.js > ...
1 const express = require('express');
2 const {
3   newOrder,
4   getSingleOrder,
5   myOrders,
6   getAllOrders,
7   updateOrder,
8   deleteOrder,
9 } = require('../controllers/orderController');
10 const router = express.Router();
11
12 const { isAuthenticatedUser, authorizedRoles } = require('../middleware/auth');
13
14 router.route('/order/new').post(isAuthenticatedUser, newOrder);
15
16 router.route('/order/:id').get(isAuthenticatedUser, getSingleOrder);
17 router.route('/orders/me').get(isAuthenticatedUser, myOrders);
18 router
19 .route('/admin/orders')
20 .get(isAuthenticatedUser, authorizedRoles('admin'), getAllOrders);
21 router
22 .route('/admin/order/:id')

JS paymentRoute.js
backend > routes > JS paymentRoute.js > ...
1 const express = require('express');
2 const {
3   processPayment,
4   sendStripeApiKey,
5 } = require('../controllers/paymentController');
6
7 const router = express.Router();
8 const { isAuthenticatedUser } = require('../middleware/auth');
9
10 router.route('/payment/post').post(isAuthenticatedUser, processPayment);
11 router.route('/stripeapikey').get(isAuthenticatedUser, sendStripeApiKey);
12
13 module.exports = router;
14
15
16
17
18
19
20
21
22

JS productRoute.js
backend > routes > JS productRoute.js > ...
1 const express = require('express');
2 const {
3   getAllProducts,
4   createProduct,
5   updateProduct,
6   deleteProduct,
7   getSingleProduct,
8   createProductReview,
9   getProductReviews,
10  deleteReview,
11  getAdminProducts,
12  bulkUpload,
13 } = require('../controllers/productController');
14 const { isAuthenticatedUser, authorizedRoles } = require('../middleware/auth');
15
16 const router = express.Router();
17
18 router.route('/products').get(getAllProducts);
19
20 router.route('/admin/products').get(isAuthenticatedUser, authorizedRoles('admin'), getAdminProducts);
21
22 router.route('/admin/product/new').post(isAuthenticatedUser, authorizedRoles('admin')).

```

The screenshot shows four tabs in Visual Studio Code: 'couponRoute.js', 'orderRoute.js', 'paymentRoute.js', and 'productRoute.js'. Each tab displays code for handling specific routes. The 'couponRoute.js' file handles routes for coupons, including admin management. The 'orderRoute.js' file handles order creation and retrieval. The 'paymentRoute.js' file handles payment processing and Stripe API key management. The 'productRoute.js' file handles product management, including reviews and bulk uploads.

Controllers:

In controllers, the definitions of the functions which are declared in the routing will be stored and also the codes of the middleware are stored in this folder. In the controller phase, the function definitions of the functions which are declared in the Routers will be done. We are having some middleware defined here.

The screenshot shows the Visual Studio Code interface with three tabs open:

- auth.js**: Contains code for user authentication, including JWT verification and role-based access control.
- catchAsyncErrors.js**: A middleware function that catches errors and returns them as promises.
- error.js**: A file containing various error handling logic, including MongoDB ID errors, duplicate errors, and JSON Web Token errors.

```

auth.js
1 const ErrorHandler = require("../utils/errorhandler");
2 const catchAsyncErrors = require("./catchsyncErrors");
3 const jwt = require("jsonwebtoken");
4 const User = require("./models/userModel");
5
6 const isAuthenticatedUser = catchAsyncErrors(
7   async (req, res, next) => {
8     const { token } = req.cookies;
9
10    if (!token) {
11      return next(new ErrorHandler("please Login to access this resource", 401));
12    }
13
14    const decodedData = jwt.verify(token, process.env.JWT_SECRET);
15    req.user = await User.findById(decodedData.id);
16    next();
17  });
18
19 const authorizedRoles = (...roles) => {
20   return (req, res, next) => {
21     if (!roles.includes(req.user.role)) {
22       return next(
23         new ErrorHandler(
24           `Role: ${req.user.role} is not allowed to access this resource.`,
25           403
26         )
27       );
28     }
29   };
30
31 module.exports = {
32   isAuthenticatedUser,
33   authorizedRoles
34 };
  
```

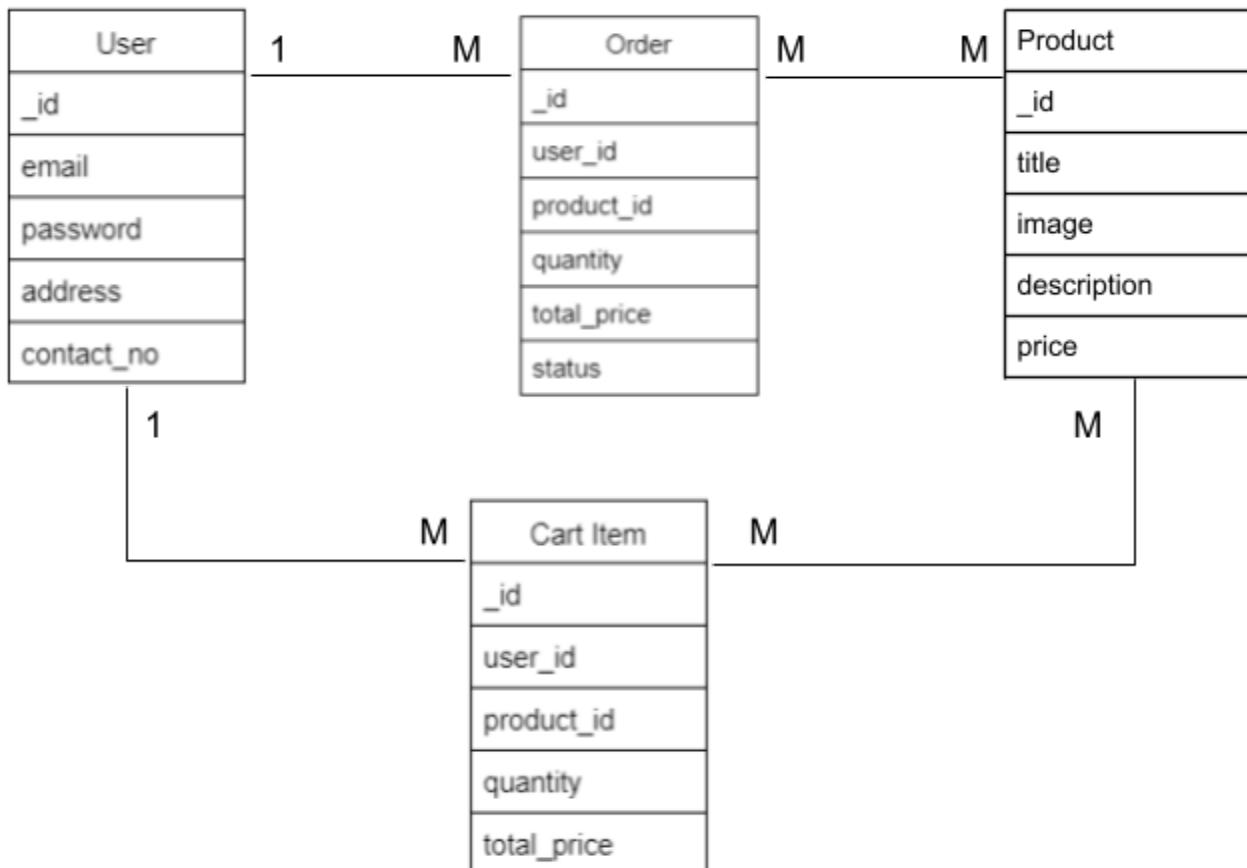
```

catchAsyncErrors.js
1 module.exports = (theFunc) => (req, res, next) => {
2   Promise.resolve(theFunc(req, res, next))
3     .catch(next);
4 };
  
```

```

error.js
1 const ErrorHandler = require("../utils/errorhandler");
2
3 module.exports = (err, req, res, next) => {
4   err.statusCode = err.statusCode || 500;
5   err.message = err.message || "Internal Server Error";
6
7   // Wrong MongoDB Id error
8
9   if (err.name === "castError") {
10     const message = `Resource not found. Invalid: ${err.path}`;
11     err = new ErrorHandler(message, 400);
12   }
13
14   // Mongoose duplicate error
15
16   if (err.code === 11000) {
17     const message = `Duplicate ${Object.keys(
18       err.keyValue
19     ).Entered}`;
20     err = new ErrorHandler(message, 400);
21   }
22
23   // Wrong JWT Error
24
25   if (err.name === "JsonWebTokenError") {
26     const message = `Json Web Token is invalid, try again`;
27     err = new ErrorHandler(message, 400);
28   }
  
```

ER- Diagram



Microservices

An e-commerce company that has a microservices-based architecture can build and deploy whatever it wants, whenever it wants and as much as it wants, unconstrained by fixed deployments

It can deliver code within a service independently, without impacting other pieces, and without restarting the entire platform. Adding and testing new site features becomes easy.

Characteristics of microservice architecture

1. Flexibility
2. Scalability
3. Agility
4. Cost efficiency



The screenshot shows a code editor window with a dark theme. The title bar says "JS couponRoute.js X". Below the title bar, the file path is shown as "backend > routes > JS couponRoute.js > ...". The main area contains the following code:

```
1 const express = require("express");
2 const { newCoupon, getSingleCoupon, getAllCoupons, deleteCoupon } = require("../controllers/couponController");
3 const router = express.Router();
4
5 const { isAuthenticatedUser, authorizedRoles } = require("../middleware/auth");
6
7 router.route("/coupons/:id").get(isAuthenticatedUser, getSingleCoupon);
8 router
9   .route("/admin/coupons")
10    .get(isAuthenticatedUser, authorizedRoles("admin"), getAllCoupons)
11    .post(isAuthenticatedUser, authorizedRoles("admin"), newCoupon);
12 router.route("/admin/coupons/:id").delete(isAuthenticatedUser, authorizedRoles("admin"), deleteCoupon);
13
14 module.exports = router;
15
```

GITHUB REPOSITORY

Here, we are providing the link of our GitHub repository -

<https://github.com/capstoneProjectTeam-group-7>

CONCLUSION

The main theme is to build an e-commerce HOME-decoration selling web application with all three i.e., Front end, back end, and database. This web application is a fully fledged working web application right from the login authentication, admin authorization, add items to cart,wishlist, and payment gateway. It can be used by any home-decor industry on either a small scale or a larger scale. The web application is easy for them to access and without any effort categories can be created and products can be added by them. It will be very attractive for the customer to see the products by sitting at home or office. It will be very helpful for the small-scale industries without selling to wholesalers, large retails mediators they can directly sell to the customer by saving money for both.