

# Capstone Project

By Group:-10

GROUP MEMBERS:

1. Ananth kumar dakoji
2. Prateek Raj Saroj
3. Divyansh Gupta
4. D. Karthik Vardhan
5. Swati Mallappa Awate

## Table Contents

Sr. No.	Topics	Page No.
1	Introduction	3
2	Problem Statement	4
3	Database Schema	5
4	Implementation	6
5	Project Output	7-14
6	Backend API	15-17
7	Conclusion	18

# Introduction

It is true that technology has become an essential tool for online marketing nowadays. However, there are numerous small shops and grocery stores with mostly offline business model in Vietnam recently. With this commerce model, it will bring a lot of bad experiences for both buyers and sellers. For instance, the seller has the product want to offer but the buyer may not know it, or the buyer may urgently need to purchase something, but the store is out of stock. Electronic commerce in the world is becoming an increasingly popular form of trade. Most shoppers start looking for products, descriptions and quality features online before buying a product. In order to provide customers with more convenience, more and more companies and existing stores are setting up their own online stores where a person can buy at a convenient time, even at night when regular stores are no longer working. Online stores allow you to save time spent by a person searching for a particular product and driving through shops. However, customers still find it difficult to choose the products they want because of the large variety of products on these sites and not focus on specific things. Moreover, the sellers have to spend a high amount of money on marketing or paying for fees. From there disadvantages, implement an online e-commerce web application for small grocery stores helps retailers can manage products on their own systems and not depend on the 3rd party website. For the customers, they can quickly search the products if it is available and come to store to pick it up and they can contact directly to the shop owner to learn more about the products that they are looking for. In order to make a website that can acquire the needs of both customers and retailers, MERN (MongoDB, Express.js framework, ReactJS library, NodeJS platform) is one of the powerful stacks that can help us to develop an e-commerce web application.

# Problem Statement:-

## User Stories –

1. As a user I should be able to login, Logout and Register into the application.
2. As a user I should be able to see the products in different categories.
3. As a user I should be able to sort the products.
4. As a user I should be able to add the products into the shopping cart
5. As a user I should be able to increase or decrease the quantity added in the cart.
6. As a user I should be able to add n number of products in the cart.
7. As a user I should be able to get the Wishlist option where I can add those products which I want but don't want to order now.
8. As a user I should get different discount coupons.

## Admin Stories –

1. As an Admin I should be able to login, Logout and Register into the application.
2. As an Admin I should be able to perform CRUD on Users.
3. As an Admin I should be able to Perform CRUD on the products.
4. As an Admin I should be able to get bulk upload option to upload a csv for products
5. As an Admin I should be able to get the stocks.
6. As an Admin I should be able to mail if any stock is less than 10.
7. As an Admin I should be able to get the sales report of a specific duration.
8. As an Admin I should be able to set the discount coupons for the specific set of users



# Database Schema



## Implementation:

There are two people who can handle the application:

- Admin
- Users

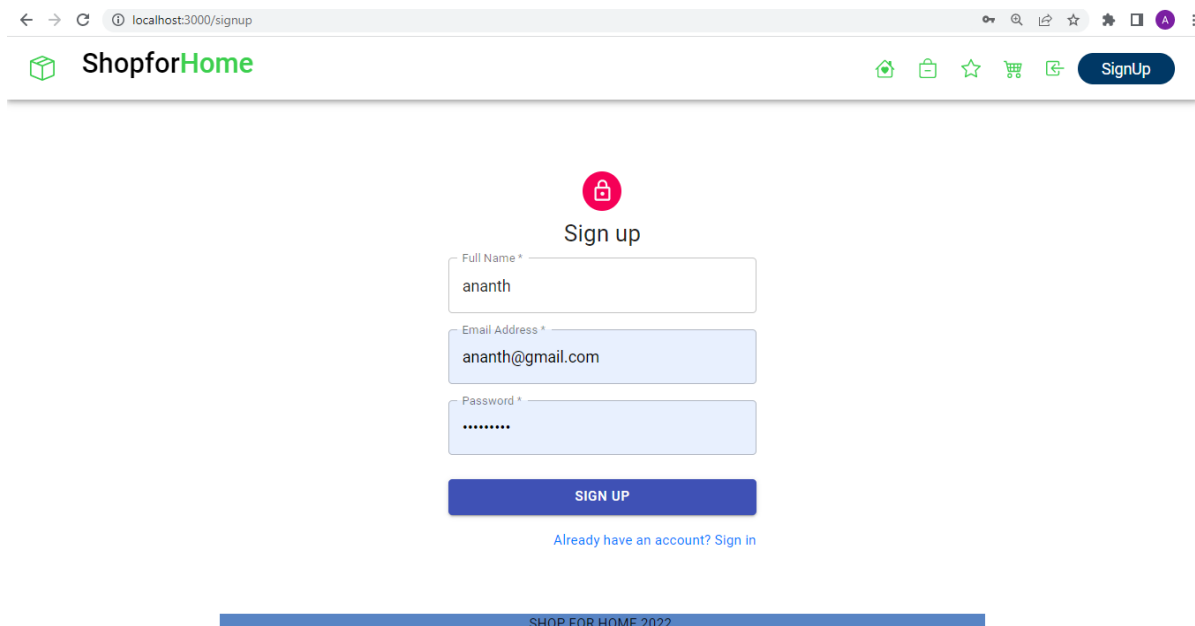
Admin:

1. Admin can Login Logout and Register on the application.
2. Admin can perform CRUD on users.
3. Admin can perform CRUD on products.
4. Admin can able to get bulk upload option to upload a csv for products details.
5. Admin will be getting Stock report.
6. If stock of any product is less than 10 Admin can send mail
7. Admin can analyse Sales Report. 8. Admin can give discount to specific users.

Users:

1. User can Login Logout and Register on the application.
2. User can see different products in shop option.
3. User can filter products based on their requirements.
4. User can add the product to cart.
5. User can increase or decrease the quantity they want to order.
6. User can add n number of product he wants to order
7. User can add product to wishlist.
8. User will be getting discount.

- Sign In Page



The screenshot shows a web browser at localhost:3000/signup. The ShopforHome logo is in the top left, and navigation icons (home, cart, wishlist, checkout, login) are in the top right. The main content area features a 'Sign up' heading with a lock icon. Below it are three input fields: 'Full Name \*' with the value 'ananth', 'Email Address \*' with the value 'ananth@gmail.com', and 'Password \*' with masked characters. A blue 'SIGN UP' button is positioned below the fields. A link 'Already have an account? Sign in' is located below the button. At the bottom of the page, a blue banner reads 'SHOP FOR HOME 2022'.

localhost:3000/signup

ShopforHome

Sign up

Full Name \*  
ananth

Email Address \*  
ananth@gmail.com

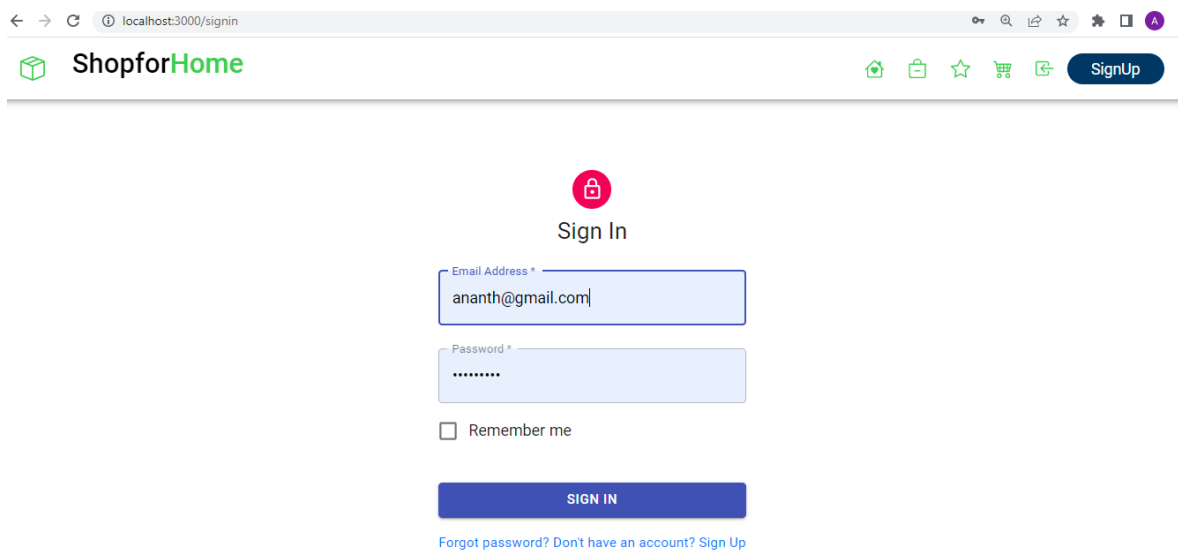
Password \*  
\*\*\*\*\*

SIGN UP

Already have an account? Sign in

SHOP FOR HOME 2022

- Sign Up Page



The screenshot shows a web browser at localhost:3000/signin. The ShopforHome logo is in the top left, and navigation icons (home, cart, wishlist, checkout, login) are in the top right. The main content area features a 'Sign In' heading with a lock icon. Below it are two input fields: 'Email Address \*' with the value 'ananth@gmail.com' and 'Password \*' with masked characters. A checkbox labeled 'Remember me' is located below the password field. A blue 'SIGN IN' button is positioned below the fields. A link 'Forgot password? Don't have an account? Sign Up' is located below the button.

localhost:3000/signin

ShopforHome

Sign In

Email Address \*  
ananth@gmail.com

Password \*  
\*\*\*\*\*

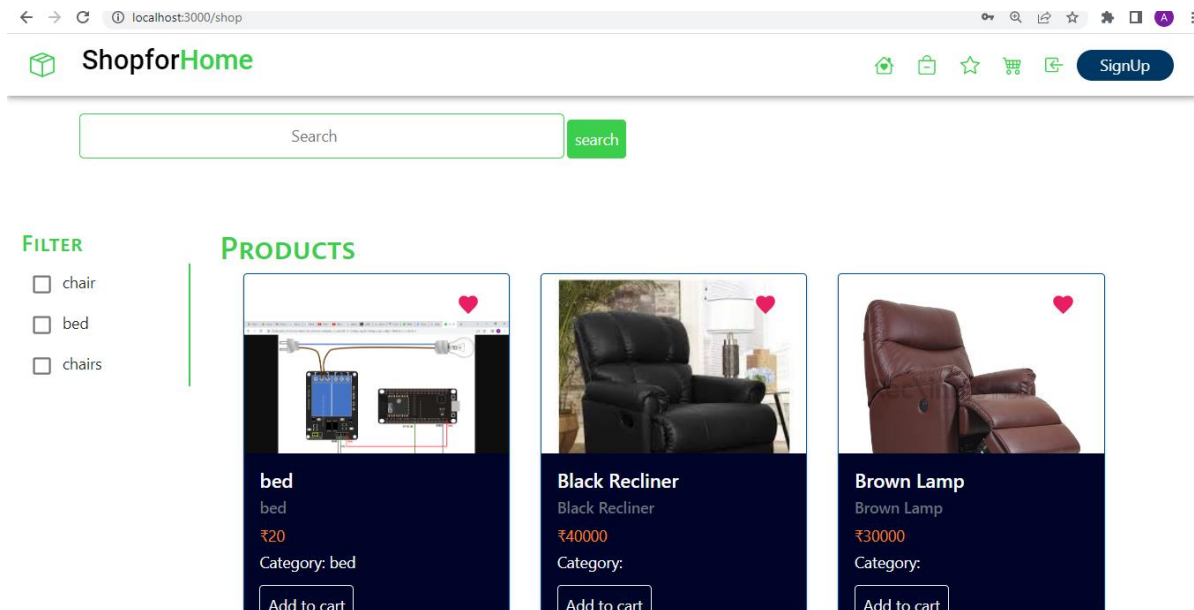
☐ Remember me

SIGN IN

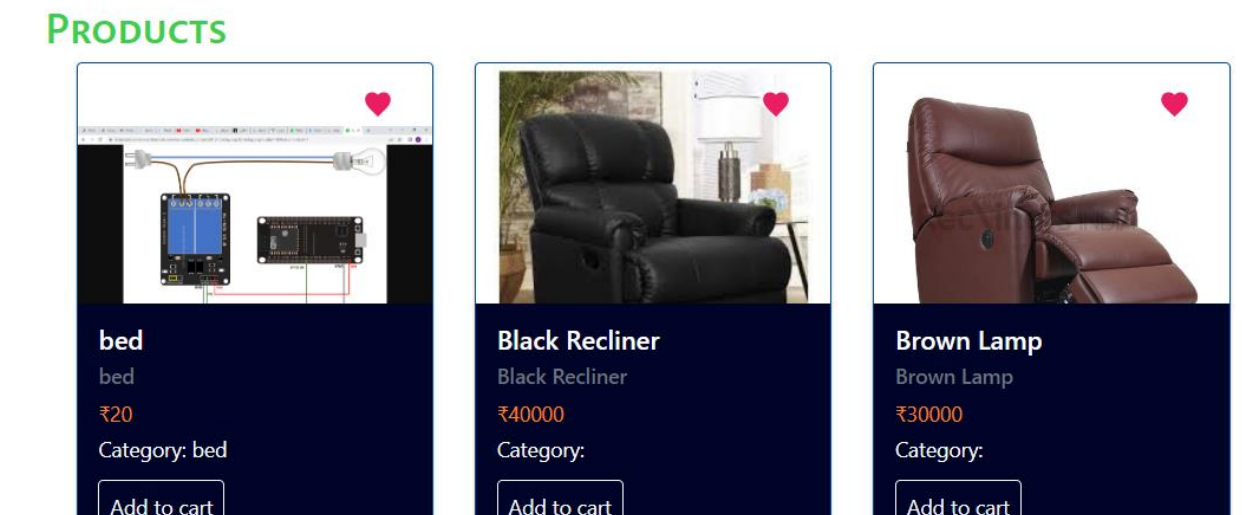
Forgot password? Don't have an account? Sign Up



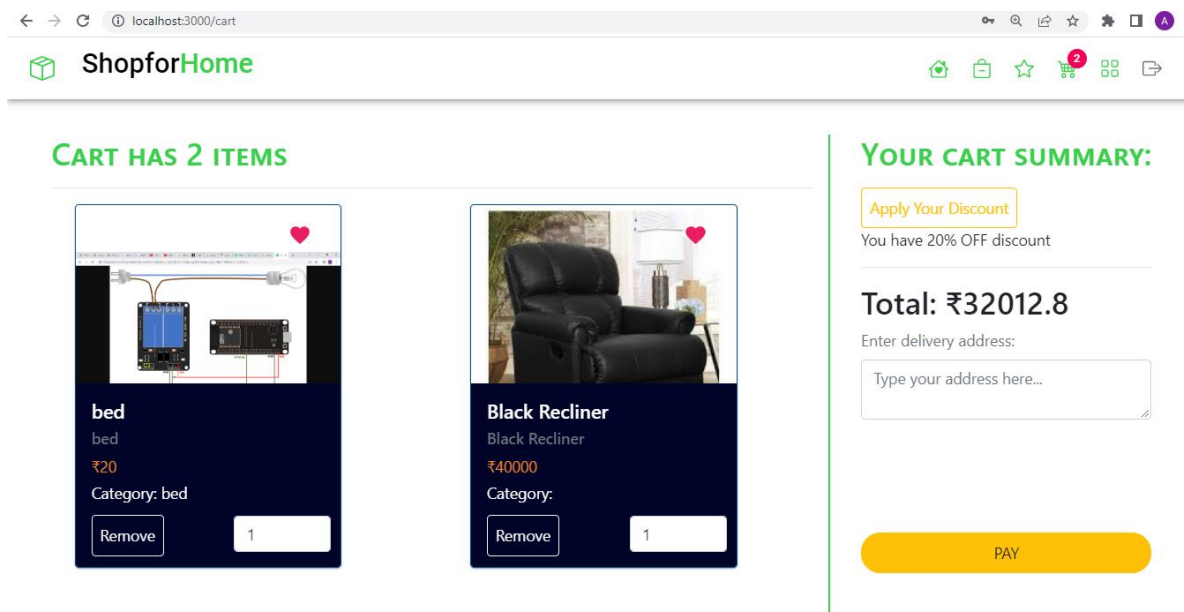
- Home Page



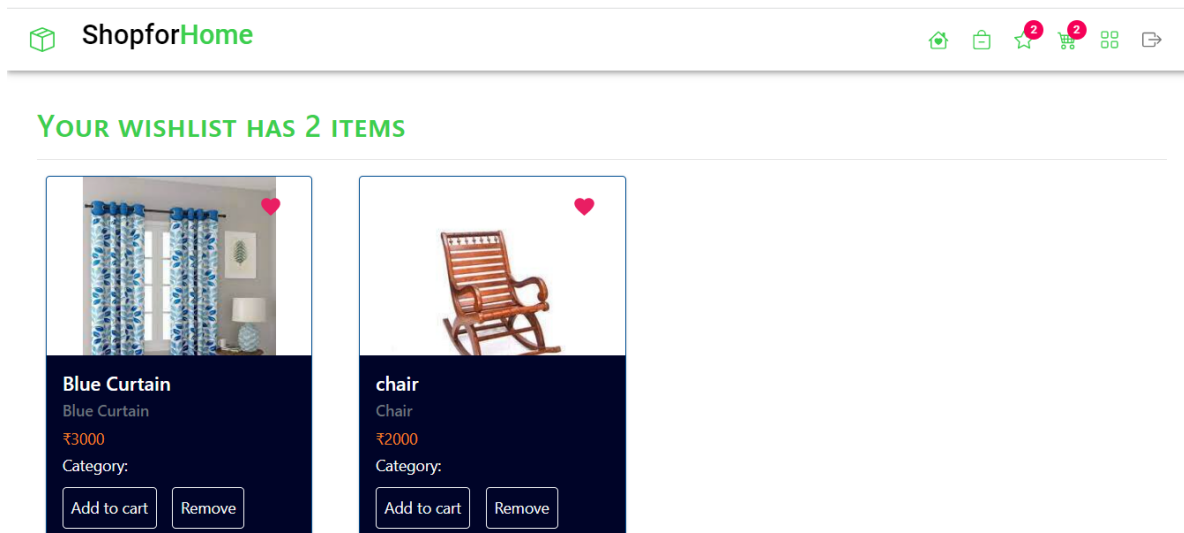
- Available Products



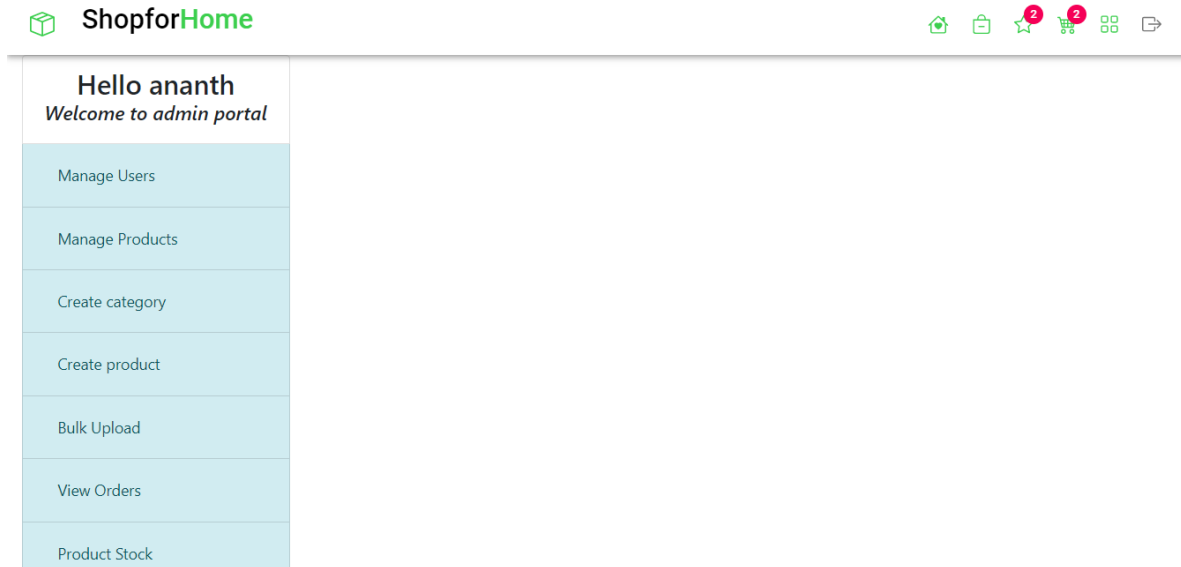
- Cart



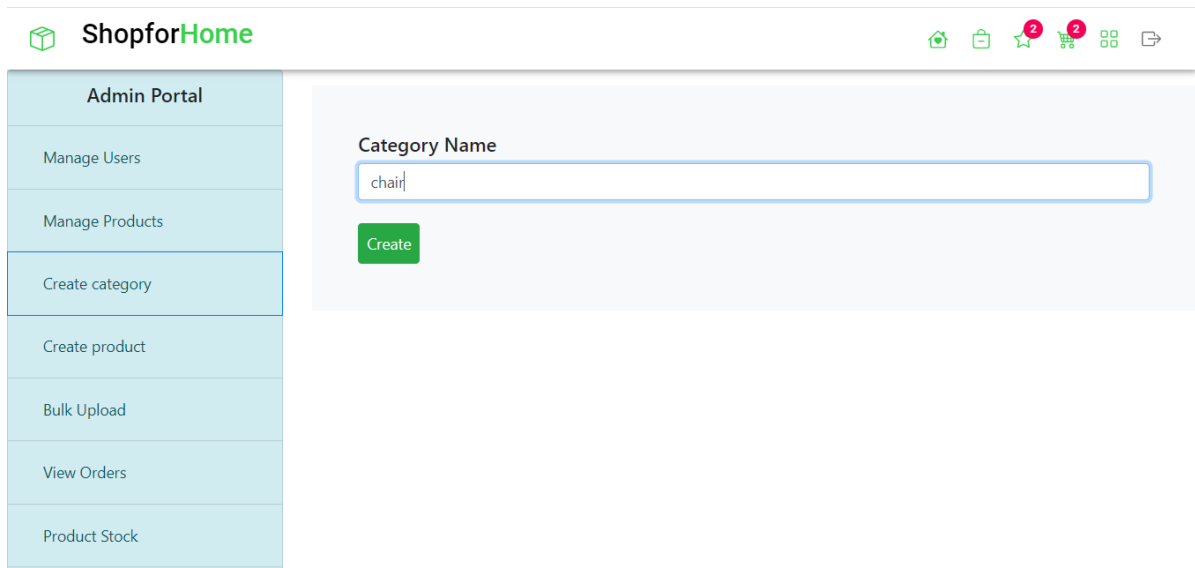
- Wishlist



- Admin Dashboard



- Adding Category



- Adding Product

ShopforHome

2

2

Create product

Bulk Upload

View Orders

Product Stock

Discount Section

Sale Report

black chair

Price600

ShippingYes

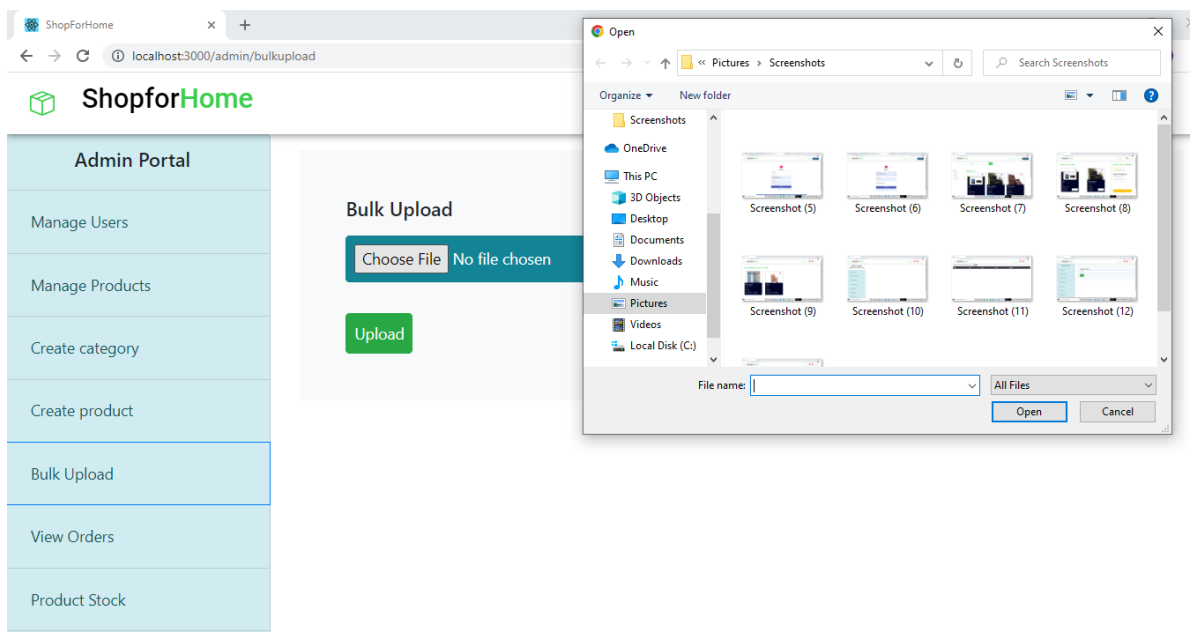
Quantity20

Upload Photo


Choose FileNo file chosen







Create Product

- Bulk Upload



## • View Orders



ShopforHome

Admin Portal	<div> Total orders: 11  <b>Order ID:</b>  <b>63028a33add0ec2bc4cafd61</b> </div> <div> Transaction ID:  Amount: ₹1996  Ordered on: 2 days ago  Delivery address: jimidipeta </div> <div> <table> <tr> <td>Product name</td> <td>chair</td> </tr> <tr> <td>Product price</td> <td>499</td> </tr> <tr> <td>Product total</td> <td>4</td> </tr> <tr> <td>Product Id</td> <td>63028996add0ec2bc4cafd60</td> </tr> </table> </div>	Product name	chair	Product price	499	Product total	4	Product Id	63028996add0ec2bc4cafd60
Product name		chair							
Product price		499							
Product total		4							
Product Id		63028996add0ec2bc4cafd60							
Manage Users									
Manage Products									
Create category									
Create product									
Bulk Upload									
View Orders									
Product Stock									

## • Manage Products


ShopforHome















Admin Portal	Brown Chair	 
Manage Users	chair	 
Manage Products	Blue Curtain	 
Create category	bed	 
Create product	chair	 
Bulk Upload	dsvv	 
View Orders		
Product Stock		

- Product Stock


 ShopforHome









Admin Portal	<table><tr><th>Products Name</th><th>Stocks</th></tr><tr><td>bed</td><td>30</td></tr><tr><td>Brown Chair</td><td>200</td></tr><tr><td>chair</td><td>5</td></tr><tr><td>Blue Curtain</td><td>60</td></tr><tr><td>Brown Lamp</td><td>500</td></tr><tr><td>Black Recliner</td><td>600</td></tr><tr><td>chair</td><td>20</td></tr></table>	Products Name	Stocks	bed	30	Brown Chair	200	chair	5	Blue Curtain	60	Brown Lamp	500	Black Recliner	600	chair	20
Products Name	Stocks																
bed	30																
Brown Chair	200																
chair	5																
Blue Curtain	60																
Brown Lamp	500																
Black Recliner	600																
chair	20																
Manage Users																	
Manage Products																	
Create category																	
Create product																	
Bulk Upload																	
View Orders																	
Product Stock																	

localhost:3000/admin/stocks

- Sales Report





 ShopforHome



From:  To:

Name	sold	Price	Total	quantity
------	------	-------	-------	----------

- Manage Users

Admin Portal	ananth	 
Manage Users	user1	 
Manage Products		
Create category		
Create product		
Bulk Upload		
View Orders		
Product Stock		

## • Discount Section

Admin Portal	ananth	<a href="#">Add Discount Coupon %</a>
Manage Users		
Manage Products		
Create category		
Create product		
Bulk Upload		
View Orders		
Product Stock		

## • Sign-Up, Sign-In and Sign-Out

```
1 const User = require('../models/user');
2 const jwt = require('jsonwebtoken'); // to generate signed token
3 const expressJwt = require('express-jwt'); // for auth check
4 const { errorHandler } = require('../helpers/dbErrorHandler');
5 const Product = require('../models/product');
6
7 require('dotenv').config();
8
9 exports.signup = (req, res) => {
10   // console.log('req.body', req.body);
11   const user = new User(req.body);
12   user.save((err, user) => {
13     if (err) {
14       return res.status(400).json({
15         error: errorHandler(err),
16       });
17     }
18     user.salt = undefined;
19     user.hashed_password = undefined;
20     res.json({
21       user,
22     });
23   });
24 };
25
26 exports.signin = (req, res) => {
27   const { email, password } = req.body;
28   User.findOne({ email }, (err, user) => {
29     if (err || !user) {
30       return res.status(400).json({
31         error: "User with that email doesn't exist. Please signup.",
32       });
33     }
34
35     if (!user.authenticate(password)) {
36       return res.status(401).json({
37         error: "Email and password didn't match",
38       });
39     }
40
41     const { _id: user_id,
42           process.env.JWT_SECRET
43         };
44     res.cookie('t', token, { expire: new Date() + 9999 });
45     const { _id, name, email, role } = user;
46     return res.json({ token, user: { _id, email, name, role, } });
47   });
48 };
49
50 exports.signout = (req, res) => {
51   res.clearCookie('t');
52   res.json({ message: 'Signout success' });
53 };
54
55 exports.requireSignin = expressJwt({
56   secret: process.env.JWT_SECRET,
57   userProperty: 'auth',
58 });
59
60 exports.isAuth = (req, res, next) => {
61   let user = req.profile && req.auth && req.profile_id == req.auth._id;
62   if (!user) {
63     return res.status(403).json({
64       error: 'Access denied',
65     });
66   }
67   next();
68 };
69
70 exports.isAdmin = (req, res, next) => {
71   if (req.profile.role == 0) {
72     return res.status(403).json({
73       error: 'Admin resource! Access denied',
74     });
75   }
76   next();
77 };
78
79
80
```

```
43   { _id: user_id,
44     process.env.JWT_SECRET
45   });
46   res.cookie('t', token, { expire: new Date() + 9999 });
47   const { _id, name, email, role } = user;
48   return res.json({ token, user: { _id, email, name, role, } });
49 };
50
51 exports.signout = (req, res) => {
52   res.clearCookie('t');
53   res.json({ message: 'Signout success' });
54 };
55
56 exports.requireSignin = expressJwt({
57   secret: process.env.JWT_SECRET,
58   userProperty: 'auth',
59 });
60
61 exports.isAuth = (req, res, next) => {
62   let user = req.profile && req.auth && req.profile_id == req.auth._id;
63   if (!user) {
64     return res.status(403).json({
65       error: 'Access denied',
66     });
67   }
68   next();
69 };
70
71 exports.isAdmin = (req, res, next) => {
72   if (req.profile.role == 0) {
73     return res.status(403).json({
74       error: 'Admin resource! Access denied',
75     });
76   }
77   next();
78 };
79
80
```



- Create, Update and Remove User

```
controllers > JS users.js > ...
1  const User = require('../models/user');
2  const { Order } = require('../models/order');
3  const { errorHandler } = require('../helpers/dbErrorHandler');
4
5  exports.userById = (req, res, next, id) => {
6    User.findById(id).exec((err, user) => {
7      if (err || !user) {
8        return res.status(400).json({
9          error: 'User not found',
10        });
11      }
12      req.profile = user;
13      next();
14    });
15  };
16
17  exports.read = (req, res) => {
18    req.profile.hashed_password = undefined;
19    req.profile.salt = undefined;
20    return res.json(req.profile);
21  };
22
23  exports.update = (req, res) => {
24
25    User.findOneAndUpdate(
26      { _id: req.profile._id },
27      { $set: req.body },
28      { new: true },
29      (err, user) => {
30        if (err) {
31          return res.status(400).json({
32            error: 'You are not authorized to perform this action',
33          });
34        }
35        user.hashed_password = undefined;
36        user.salt = undefined;
37        res.json(user);
38      }
39    );
40  };
41
```

```
controllers > JS users.js > ...
44  exports.addOrderToUserHistory = (req, res, next) => {
45    let history = [];
46
47    req.body.order.products.forEach((item) => {
48      history.push({
49        _id: item._id,
50        name: item.name,
51        description: item.description,
52        category: item.category,
53        quantity: item.count,
54        transaction_id: req.body.order.transaction_id,
55        amount: req.body.order.amount,
56      });
57    });
58
59    User.findOneAndUpdate(
60      { _id: req.profile._id },
61      { $push: { history: history } },
62      { new: true },
63      (error, data) => {
64        if (error) {
65          return res.status(400).json({
66            error: 'Could not update user purchase history',
67          });
68        }
69        next();
70      }
71    );
72  };
73
74  exports.purchaseHistory = (req, res) => {
75    Order.find({ user: req.profile._id })
76      .populate('user', '_id name')
77      .sort('-created')
78      .exec((err, orders) => {
79        if (err) {
80          return res.status(400).json({
81            error: errorHandler(err),
82          });
83        }
84      });
85  };
86
```

```

80         return res.status(400).json({
81             error: errorHandler(err),
82         });
83     }
84     res.json(orders);
85 });
86 };
87
88 exports.getUser = (req, res) => {
89     User.find()
90     .exec((err, user) => {
91         if (err) {
92             return res.status(400).json({
93                 error: errorHandler(err),
94             });
95         }
96         res.json(user);
97     });
98 };
99
100 exports.deleteUser = (req, res) => {
101     req.params.userId
102     console.log(req.params.userId)
103     User.findByIdAndDelete(req.params.userId)
104     .exec((err, user) => {
105         if (err) {
106             return res.status(400).json({
107                 error: errorHandler(err),
108             });
109         }
110         res.json("deleted succesfully");
111     });
112 };

```

# Conclusion

The achievement of the thesis is researching the basic components of MERN stack technology: MongoDB, ExpressJS framework, ReactJS library and NodeJS platform. Using MERN stack technology in conjunction with Braintree to build an e-commerce web application with payment gateway. The advantages are performing the basic functions of a product search website for customers, making it easy for customers to find categories that have the products they are looking for. Gives small stores a platform to store information and promote their products. Password data of accounts when logging in to the system is stored in a secure database. The management interface, statistics of the user and admin are easy to use for everyone.

Since the purpose of the thesis is the e-commerce application, the understanding about MERN technologies and applying it to this app is the most important. Overcome current shortcomings, listen to customers' comments and making improvements, helping users have a great experience in the future.

## **Git Repository Link:**

<https://github.com/Ananthdakoji2001/shopforhome>

