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

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.

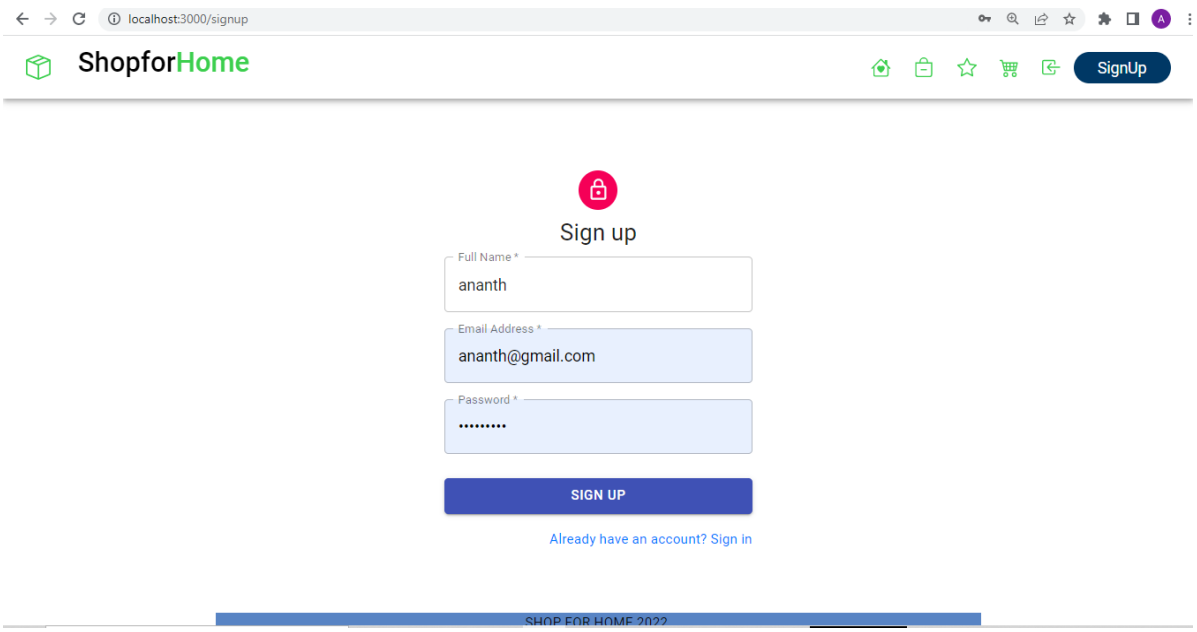
5.User can increase or decrease the quantity they want to order.

6. User can add n number of product he wants to order

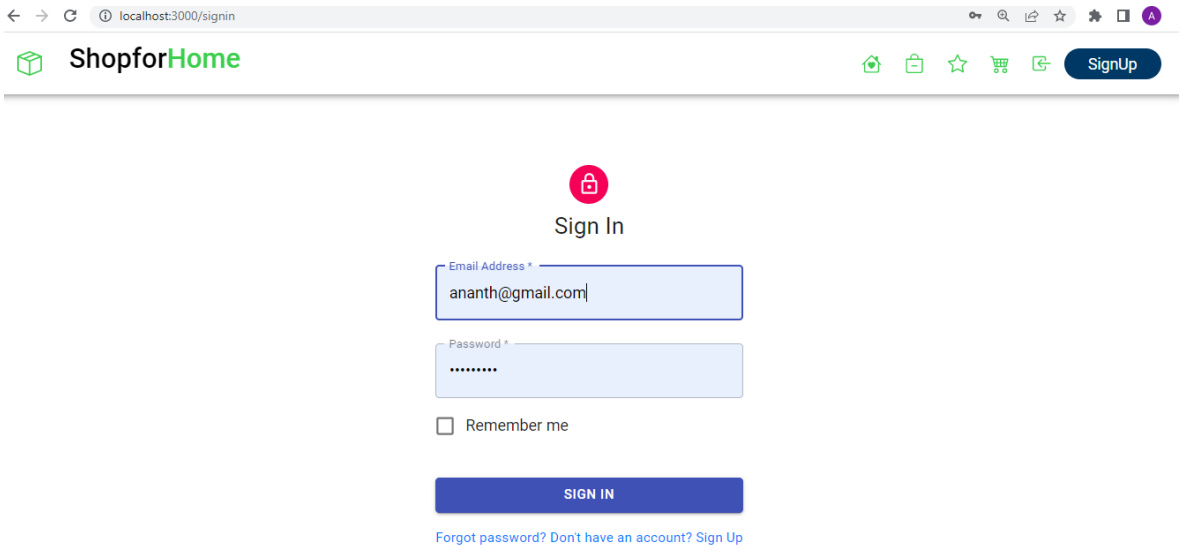
7. User can add product to wishlist.

8. User will be getting discount.

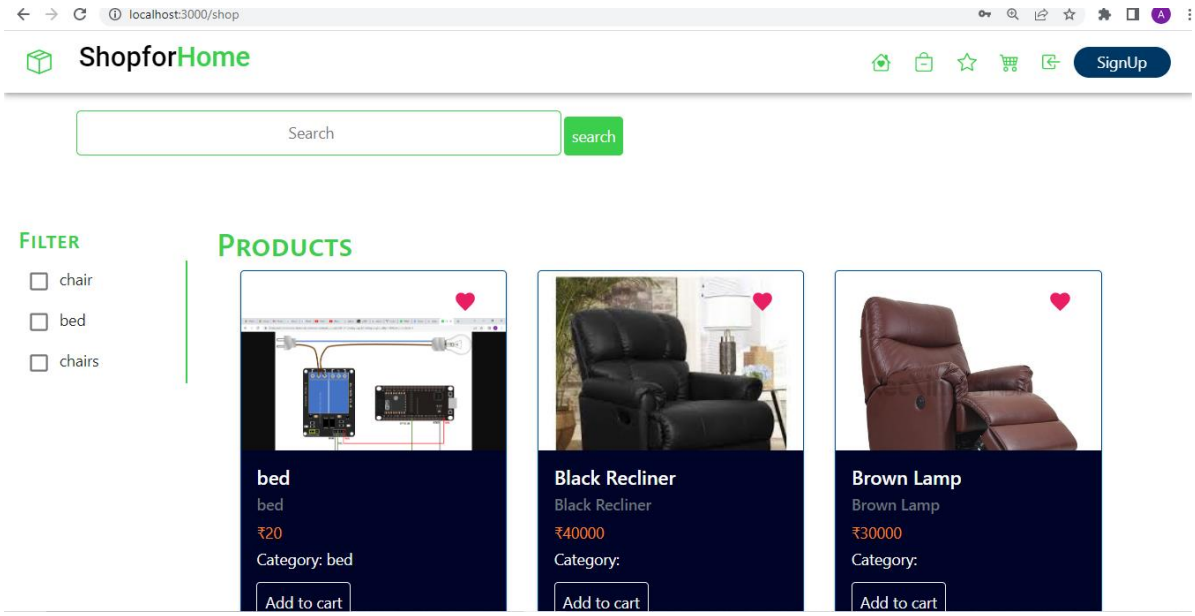
- **Sign In Page**



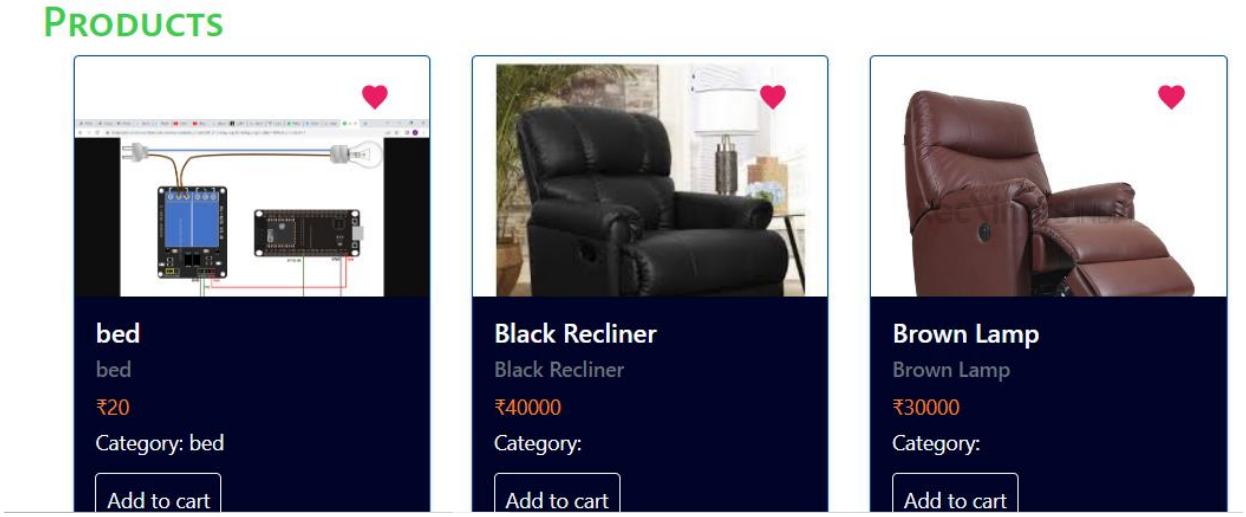
- **Sign Up Page**



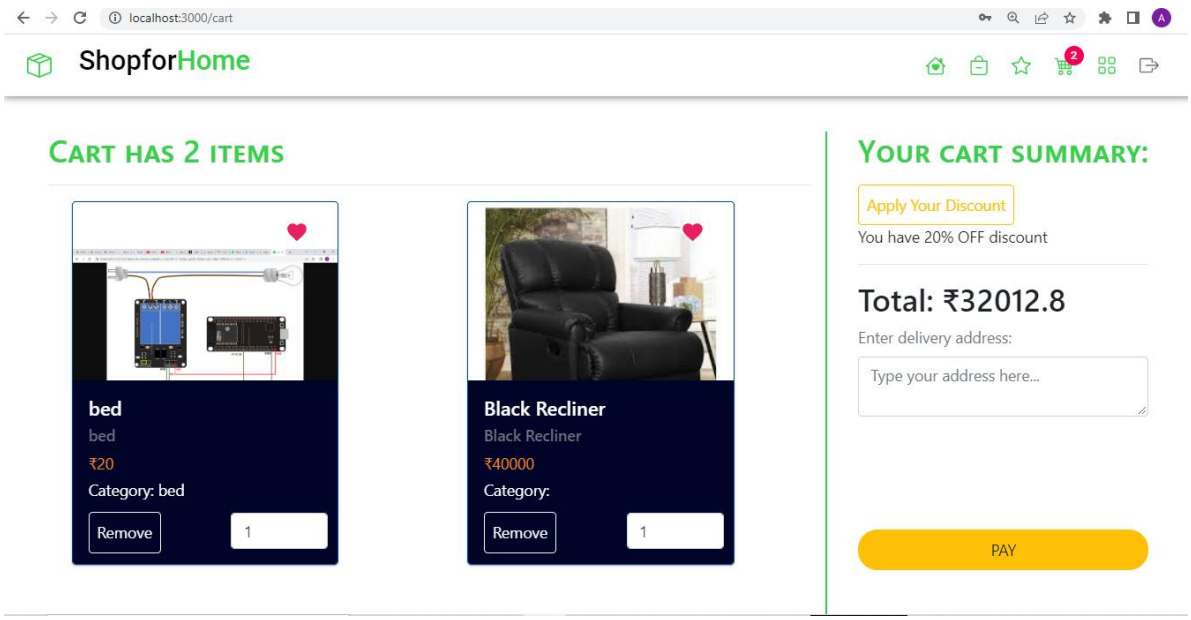
- **Home Page**



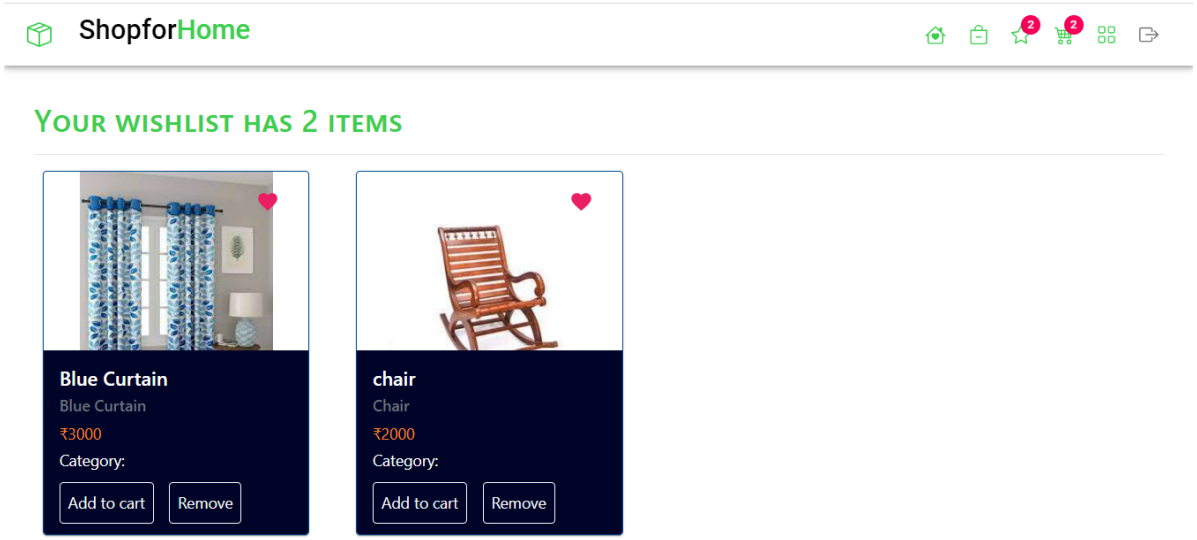
- **Available Products**



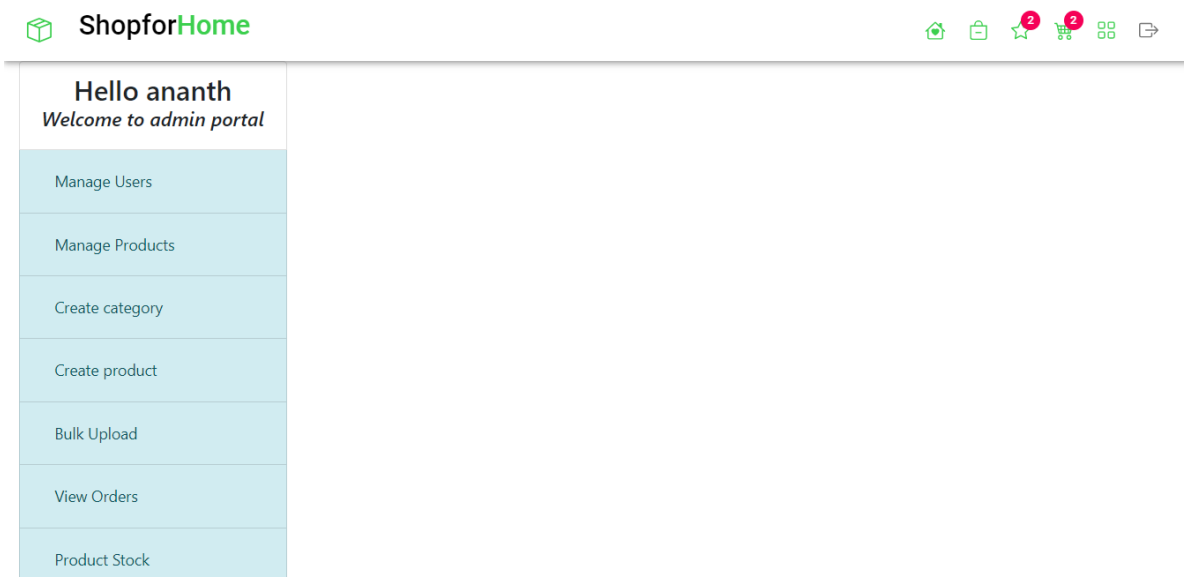
- **Cart**



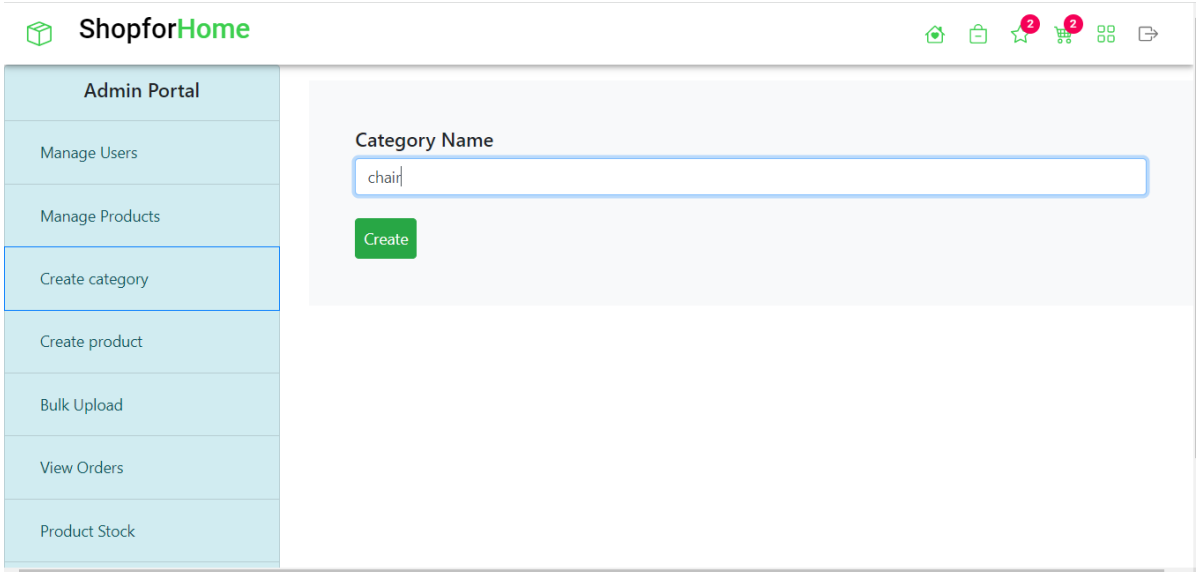
- **Wishlist**



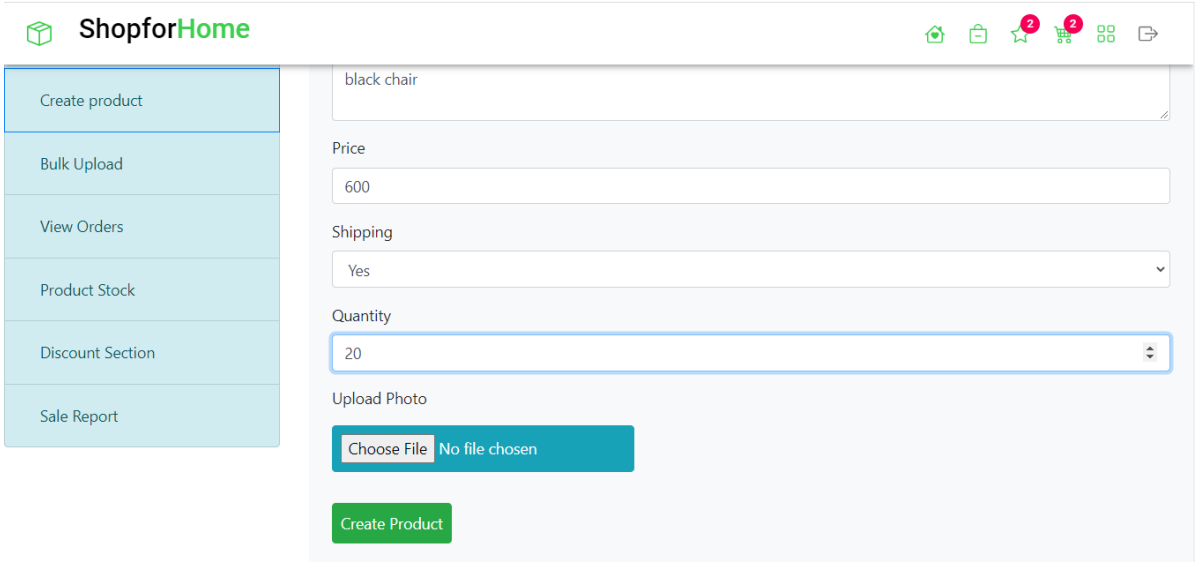
- **Admin Dashboard**



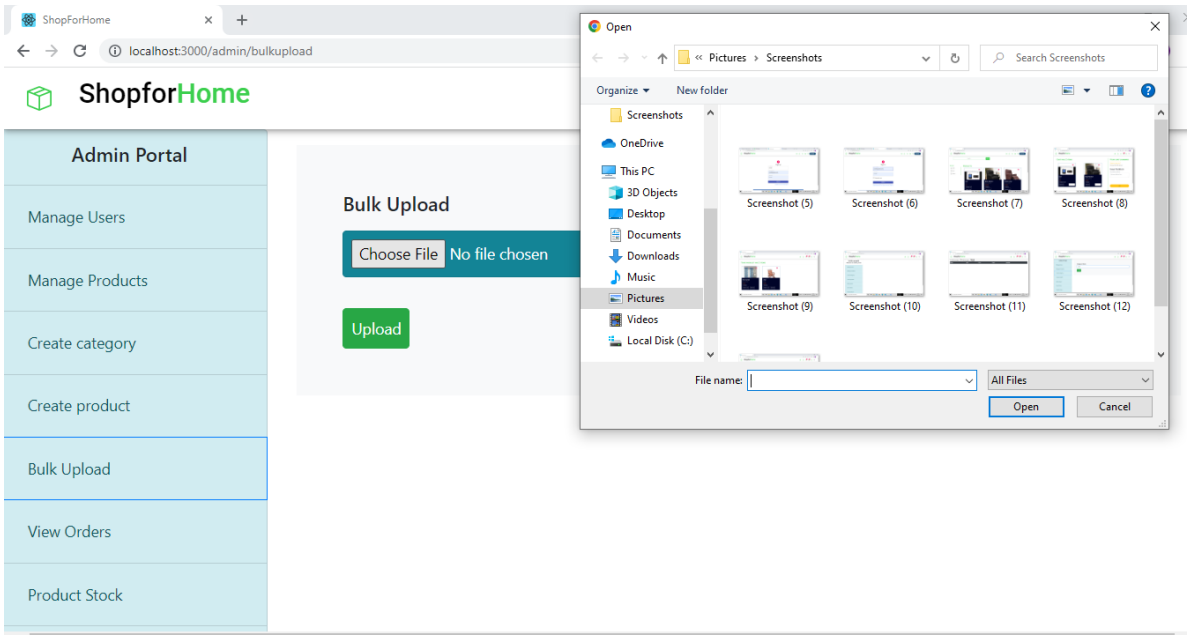
- **Adding Category**



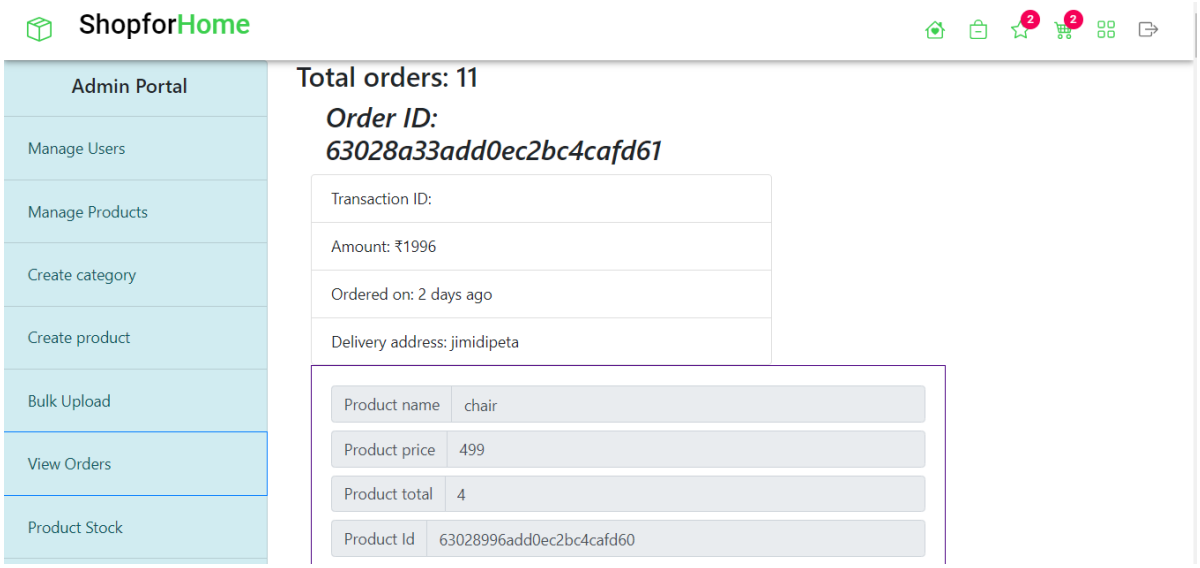
- **Adding Product**


ShopforHome
Create product
Bulk Upload
View Orders
Product Stock
Discount Section
Sale Report
black chair
Price
600
Shipping
Yes
Quantity
20
Upload Photo
Choose File
No file chosen
Create Product

- **Bulk Upload**

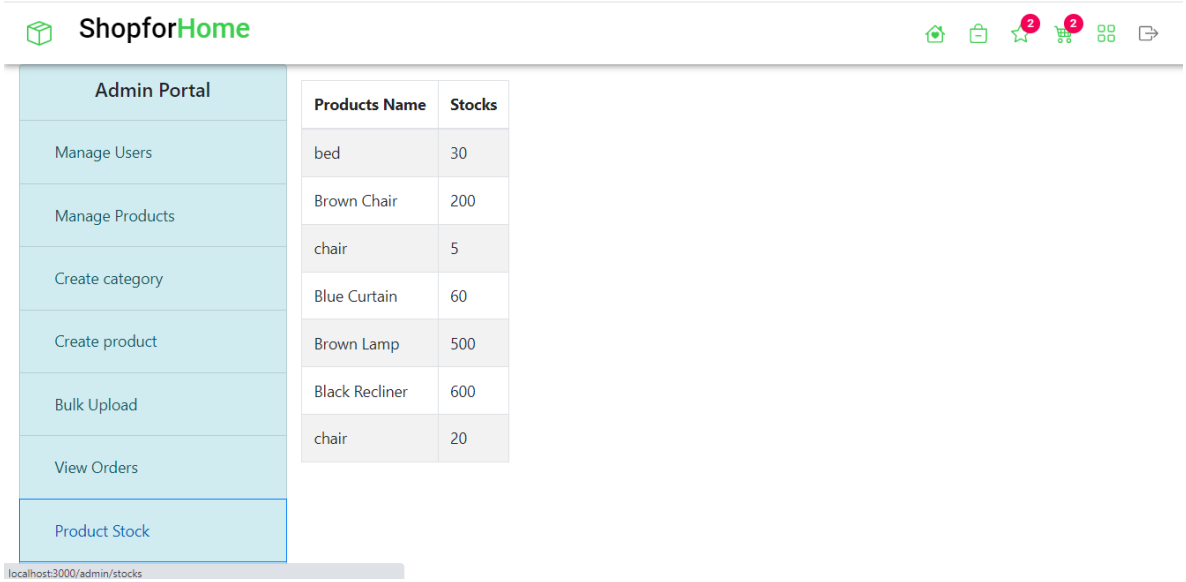

ShopForHome
localhost:3000/admin/bulkupload
ShopforHome
Admin Portal
Manage Users
Manage Products
Create category
Create product
Bulk Upload
View Orders
Product Stock
Bulk Upload
Choose File
No file chosen
Upload
Open
Pictures › Screenshots
Search Screenshots
Organize
New folder
Screenshots
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Screenshot (5)
Screenshot (6)
Screenshot (7)
Screenshot (8)
Screenshot (9)
Screenshot (10)
Screenshot (11)
Screenshot (12)
File name:
All Files
Open
Cancel

- **View Orders**

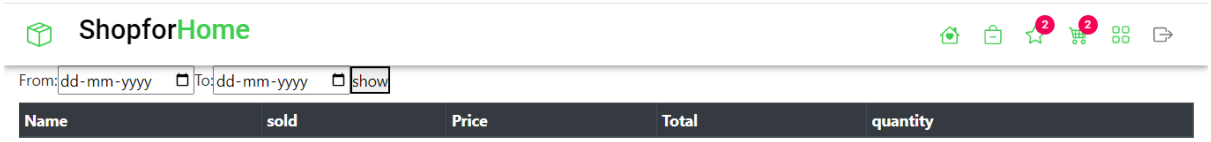


- **Manage Products**

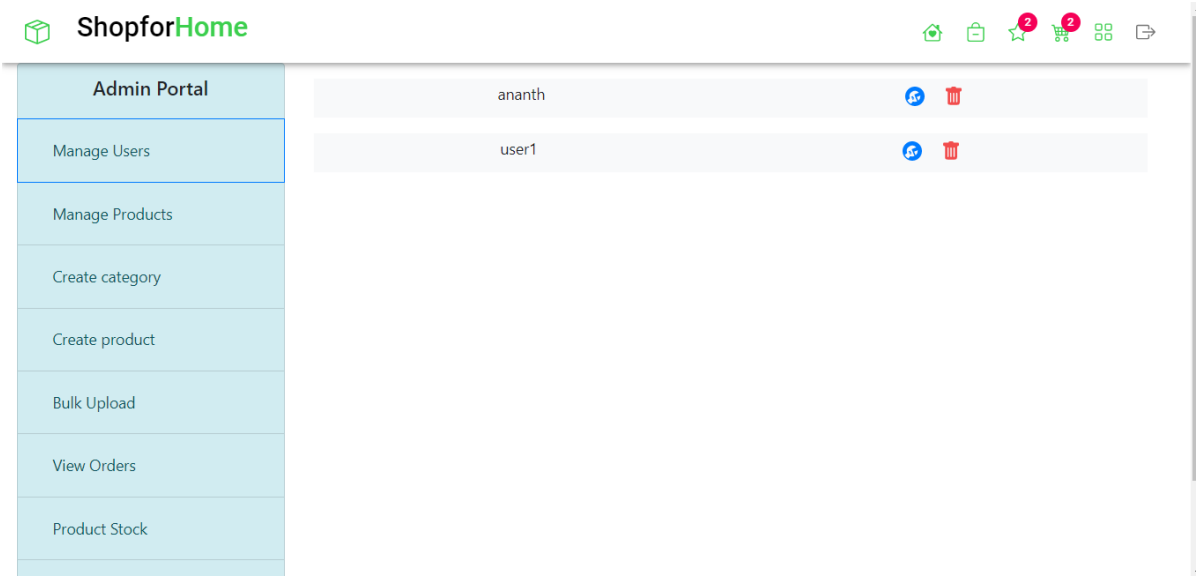
- **Product Stock**



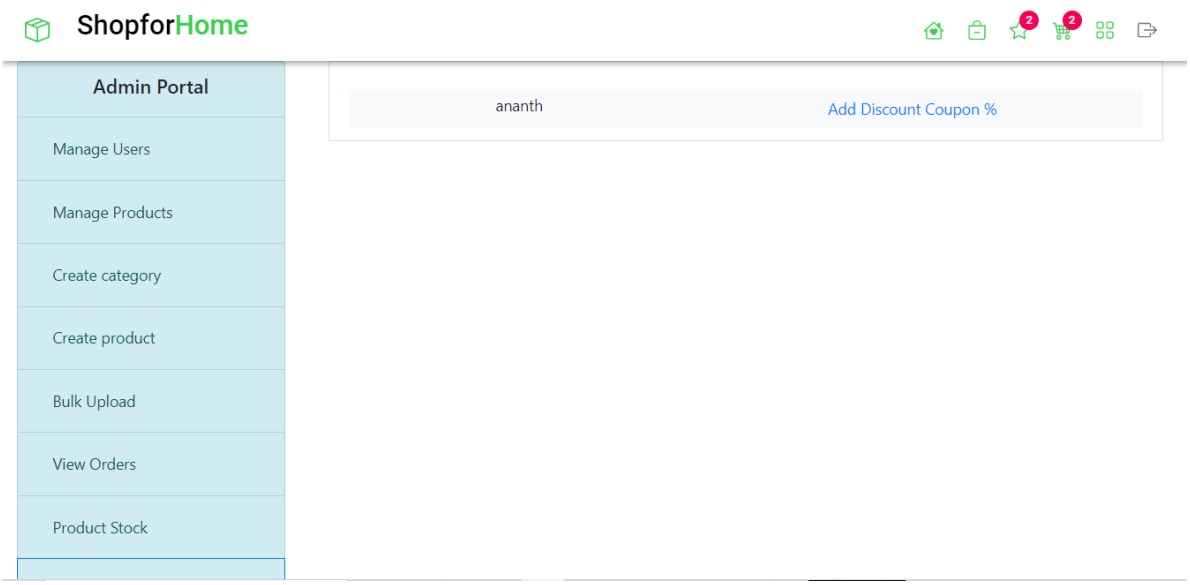
- **Sales Report**



- **Manage Users**



- **Discount Section**



- **Sign-Up, Sign-In and Sign-Out**

```
const User = require('../models/user');
const jwt = require('jsonwebtoken'); // to generate signed token
const expressJwt = require('express-jwt'); // for auth check
const { errorHandler } = require('../helpers/dbErrorHandler');
const Product = require('../models/product');

require('dotenv').config();

exports.signup = (req, res) => {
  // console.log('req.body', req.body);
  const user = new User(req.body);
  user.save((err, user) => {
    if (err) {
      return res.status(400).json({
        err: errorHandler(err),
      });
    }
    user.salt = undefined;
    user.hashed_password = undefined;
    res.json({
      user,
    });
  });
};

exports.signin = (req, res) => {

  const { email, password } = req.body;
  User.findOne({ email }, (err, user) => {
    if (err || !user) {
      return res.status(400).json({
        error: "User with that email doesn't exist. Please signup.",
      });
    }

    if (!user.authenticate(password)) {
      return res.status(401).json({
        error: "Email and password didn't match",
      });
```

```
        { _id: user._id },
        process.env.JWT_SECRET
    );
    res.cookie('t', token, { expire: new Date() + 9999 });
    const { _id, name, email, role } = user;
    return res.json({ token, user: { _id, email, name, role, } });
  });
};

exports.signout = (req, res) => {
  res.clearCookie('t');
  res.json({ message: 'Signout success' });
};

exports.requireSignin = expressJwt({
  secret: process.env.JWT_SECRET,

  userProperty: 'auth',
});

exports.isAuth = (req, res, next) => {
  let user = req.profile && req.auth && req.profile._id == req.auth._id;
  if (!user) {
    return res.status(403).json({
      error: 'Access denied',
    });
  }
  next();
};

exports.isAdmin = (req, res, next) => {
  if (req.profile.role === 0) {
    return res.status(403).json({
      error: 'Admin resource! Access denied',
    });
  }
  next();
};
```

- **Create, Update and Remove User**

controllers > JS user.js > ...

```javascript
const User = require('../models/user');
const { Order } = require('../models/order');
const { errorHandler } = require('../helpers/dbErrorHandler');

exports.userById = (req, res, next, id) => {
  User.findById(id).exec((err, user) => {
    if (err || !user) {
      return res.status(400).json({
        error: 'User not found',
      });
    }
    req.profile = user;
    next();
  });
};

exports.read = (req, res) => {
  req.profile.hashed_password = undefined;
  req.profile.salt = undefined;
  return res.json(req.profile);
};

exports.update = (req, res) => {

  User.findOneAndUpdate(
    { _id: req.profile._id },
    { $set: req.body },
    { new: true },
    (err, user) => {
      if (err) {
        return res.status(400).json({
          error: 'You are not authorized to perform this action',
        });
      }
      user.hashed_password = undefined;
      user.salt = undefined;
      res.json(user);
    }
  );
```

controllers > JS user.js > ...

```javascript
exports.addOrderToUserHistory = (req, res, next) => {
  let history = [];

  req.body.order.products.forEach((item) => {
    history.push({
      _id: item._id,
      name: item.name,
      description: item.description,
      category: item.category,
      quantity: item.count,
      transaction_id: req.body.order.transaction_id,
      amount: req.body.order.amount,
    });
  });

  User.findOneAndUpdate(
    { _id: req.profile._id },
    { $push: { history: history } },
    { new: true },
    (error, data) => {
      if (error) {
        return res.status(400).json({
          error: 'Could not update user purchase history',
        });
      }
      next();
    }
  );
};

exports.purchaseHistory = (req, res) => {
  Order.find({ user: req.profile._id })
    .populate('user', '_id name')
    .sort('-created')
    .exec((err, orders) => {
      if (err) {
        return res.status(400).json({
          error: errorHandler(err),
```

```
79        if (err) {
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