

**MVIT:An online marketplace for used items
for VITians**

PROJECT REPORT

Submitted in fulfilment for the J-Component of CSE3002 – Internet and Web Programming

CSE-3002-INTERNET AND WEB PROGRAMMING
in

B.Tech. (Computer Science)

by

Anuj Mishra(20BCE2934)

Iyashi Pal(20BCI0295)

Mediseti Varaguna SreeChandana(20BCE0977)

Tarang Gupta(20BCI0310)

Prince Panjiyar(20BCE2775)

Under the guidance of

Prof. Jayakumar Sadhasivam

SCOPE



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Fall Semester 2022-2023

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	Abstract	3
2.	Introduction	3
3.	Problem Statement	4
4.	Process Flow Diagram	4
5.	Modules	5
6.	ER Diagram	7
7.	Database Design	7
8.	Conclusion	7
9.	References	8
10.	Snapshots	8
11.	Sample Source Code	11

Title:

MVIT (An online marketplace for used items for Vitians using MERN Stack)

Abstract:

MVIT is an online shopping website which allows users to buy and sell used items and it is specially designed for Vitians. It makes selling and buying of things easier and accessible within the campus of VIT.

These days online shopping is very popular and many websites are being made but it is difficult to recognize the fraud also. This website is made only for Vitians and takes proper login information to access the portal. This project is developed using ReactJS and MongoDB.

The sellers can put an advertisement on the portal after creating an account and registering to the website. The buyers can access the portal to see the items and price of the product then if they wish to buy the payments can be done using UPI QR code. By using this website, the buyers need not buy a completely new product, they can get a used product at a cheaper price.

The main users of the website are the customer, sellers and the admin. The admin of the system has full control over the system like removing the products and updating information about the seller or website etc.

Introduction

The purpose of the project called "MVIT" is to make buying and selling of used items within VIT easier and much more accessible to students inside the premises.

The project has been developed using ReactJS as the front-end with MongoDB for the database. An online shopping site that will allow VIT students to sell their new or used items to other people inside of VIT and continue with the payments online as well using UPI QR Code. Sellers here can put up their posts for advertisement and sell their items using the portal. This would permit a lot of students to make their choices available using much less amount than buying an item completely new from somewhere else.

The objective of this project is to create an e-commerce web portal with a content management system that would allow product information to be updated by adding up the product individually by creating an account on the website. The web portal has an online interface in the form of an e-commerce website that will allow users to buy goods from the sellers. There are two types of users available in the project. The first one is Customer and the second one is Admin or the seller. Customers/users have limited access rights to access the system while the admin users have full control over the system like removing products from the website of all the users and updating the information about the seller on the website.

We have used token authentication for the user login in the system directly for secure authentication.

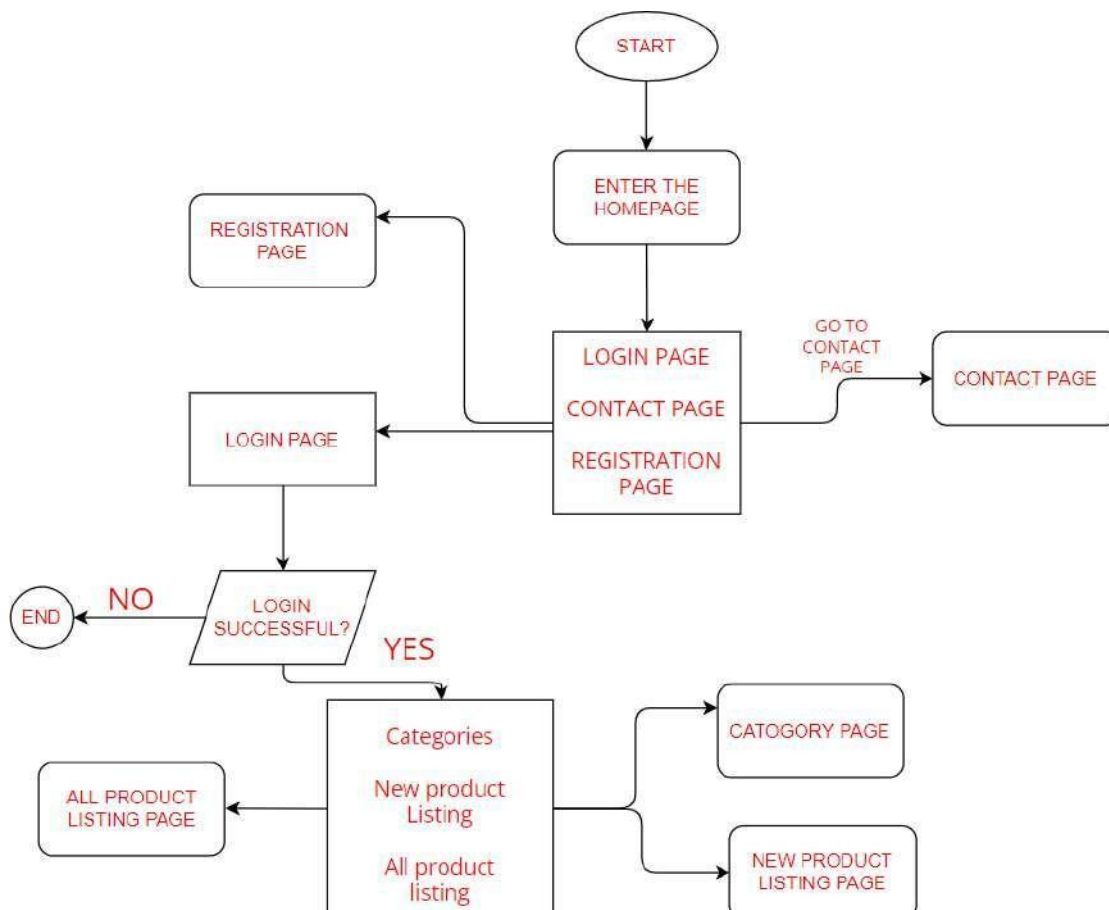
Problem Statement:

These days people have started throwing the used items when they are no longer needed even when they are in working condition. But these things may be useful to other people. So MVIT gives a platform to the people who want to sell their used items through our website and also for the people who want to buy the used items. By using the MVIT website, the sellers are trying to get some money for their used items and the buyers are able to buy the things they need for less cost. In this way both the seller and the buyer are benefitted and the wastage is reduced.

Reason for choosing this topic:

We chose the topic of MVIT: An online market in Vit because it is a very relatable problem in Vit today. Even many of the alumni of vit came across this problem during their time and now mostly seniors and freshers are facing this issue. When seniors leave the college most of their stuff which they bought during joining the college is simply wasted as they leave the college. Similarly, Freshers when they join Vit they have to buy a lot of stuff at high price which could be solved if they get a place to buy second hand goods from their seniors. This will also help students by selling their goods to other students for some price.

Process Flow Diagram:



Modules:

A. Frontend

We have used the JavaScript framework, ReactJS for the frontend work on our project. ReactJS allows us to divide the various web pages in our website into functional components. Each of the functional components can act independently of the web page. What this allows us to do is that we can only refresh that one component instead of reloading the whole web page. Breaking a complex web page into multiple components helps improve the user experience. It also takes comparatively less time to load, because components like the navbar and footer are common to all the pages. We also used State Management and Hooks to manage the variables, tokens and user details on the frontend. We used this to store and manage user form data, authentication tokens returned by the backend to the frontend. In some cases, we have also set the status returned by the backend and based on the status, we have displayed various components, like the logout button on the navbar. It only appears after the user has successfully logged in and the backend returns a token to the frontend. The token is then stored as a cookie. Whenever we wanted to check if the user is logged in, we checked if the cookie existed.

Web Pages

a. Homepage

This is the landing page of our website; we display recently posted products and all the available categories of products on this page. We also have links to the registration page and the login page on our homepage. Registration allows users to register on the website with basic information like name, email and password.

b. Login and Registration

Login needs the user's email address and password. For a user to register on our website, they just have to enter their name, email and their chosen password.

c. Edit Profile Page

Upon registering based on their basic information, the users have a chance to update their profile with important information before they go on and sell or buy products. The users can add or edit information like username, the address of the user, their primary mobile number, their alternate mobile number and their UPI ID for payments. We do not allow email changes on our website as we use it as the entity which uniquely identifies a user on our website. We think that adding functionality to change a user's email address will be too complicated in terms of uniquely identifying the users.

d. Posting Items for Sale

All users on the website have a chance to post items for sale on our website. They can do so by filling a form on the “add new product” page. The users have to enter the product name, description, price, category into which the product falls, its SKU and the condition of the product.

e. All Products Page

This page displays all the products that have been posted on our page. Users can click on the respective product card if they are interested in buying the product. They will be redirected to the seller profile page; they can then scan the UPI ID QR code there to make the payment or just type in the UPI ID on their payment app and send the money.

f. Contact Us Page

This page is for the users to ask questions or post in queries, if any, about the website. They can also register complaints about sellers if they run into any problems.

g. Support Requests Viewing Page

This page is for the admin users to view all the incoming support requests. The admins can then click on the “Answer Request” button to answer a specific request. They can type the answer on the page they are redirected.

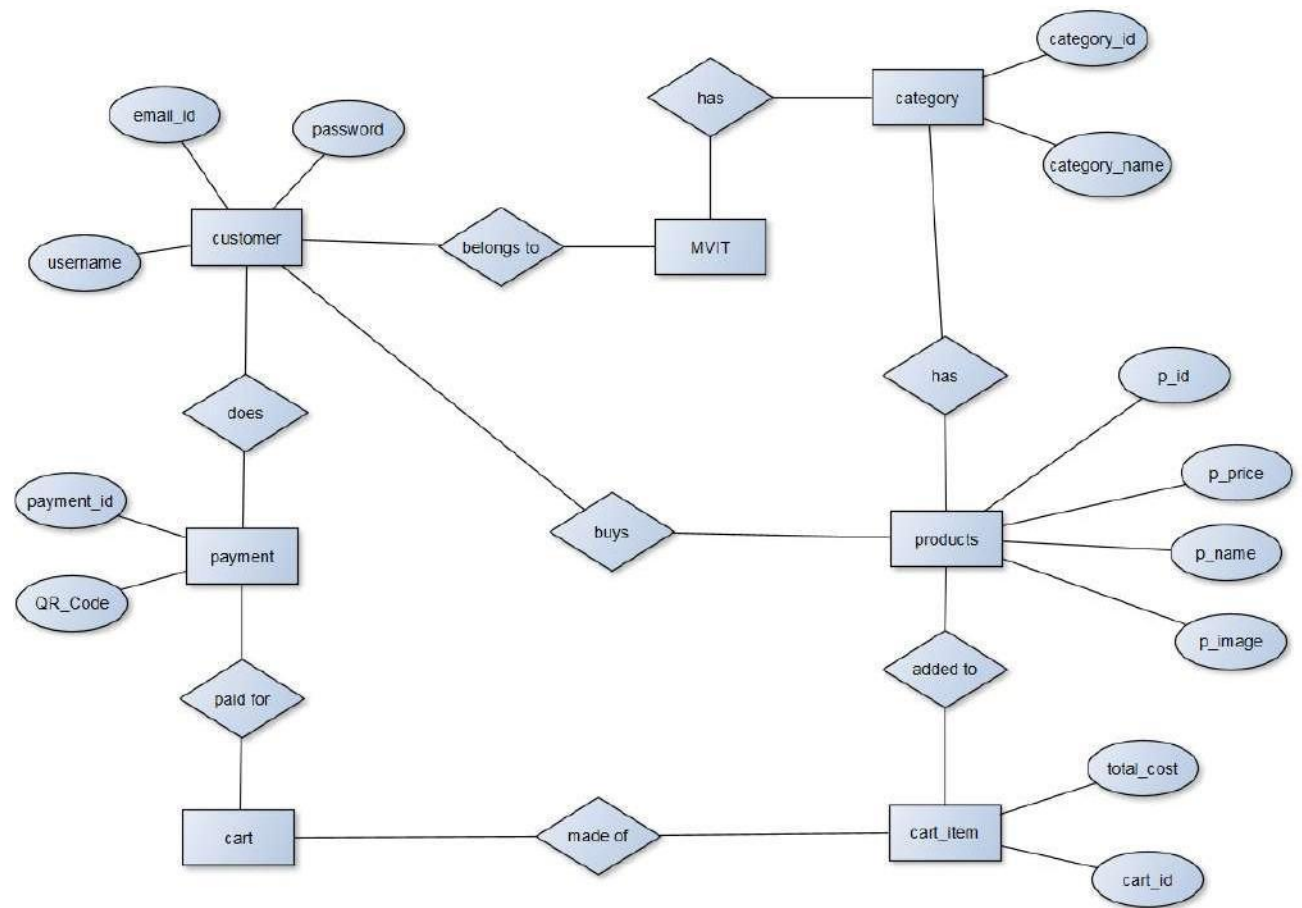
h. Support Requests Answering Page

This page is used by the administrator to answer all the queries which are viewed in the support requests page.

Backend

In our project, for the backend, we have used ExpressJS and NodeJS Frameworks of the JavaScript language. We used MongoDB Atlas as the database. It is hosted on a cloud platform. We deployed our backend on Heroku and frontend on Vercel. In the next section, we briefly describe the various models we used for the database in our project.

ER Diagram



Database Design

We are using MongoDB as the database for the project. In the database we have customer, payment, cart, MVIT, products, cart_item, category as the entities and they have different attributes whose values are stored in the database. In the database we are storing the username, password and the email id of the customer and this username and password is used as login credentials for the website. We also store the data about different products and they are categorized into different categories. In payments we store payment id and QR code details.

Conclusion:

Our MVIT website aims to make buying and selling of used items within VIT easier and much more accessible to students inside the premises. These items may include daily purpose things like mattresses, books and bags as well as more valuable things like laptops, mobile phones and other electronics. Through this project, we will be able to minimize wastage and avoid throwing away the used items that are no longer useful to us. Instead, they can be sold to other people which will be beneficial for everyone. We will also be able to help people who do not wish to spend a large amount of money to buy brand new products.

References:

[1] Online Shopping Management System

Department of Computer Science and Engineering, Brac University (2017)

[2] Building The Electronic Store Website Vietnam-Korea University Of Information And Communication Technology (2021)

[3] Laudon, Kenneth C., and Carol Guercio Traver. E-commerce. Boston, MA: Pearson, 2013.

[4] Sila, Ismail. "Factors affecting the adoption of B2B e-commerce technologies." Electronic commerce research 13.2 (2013): 199-236.

[5] Singh, Mohini. "E-services and their role in B2C e-commerce." Managing Service Quality: An International Journal (2002).

[6] Singh, M., 2002. E-services and their role in B2C e-commerce. Managing Service Quality: An International Journal.

[7] Wu, Qinglie, Jing Ma, and Zhong Wu. "Consumer-Driven E-commerce: A Study on C2B applications." 2020 International Conference on E-Commerce and Internet Technology (ECIT). IEEE, 2020.

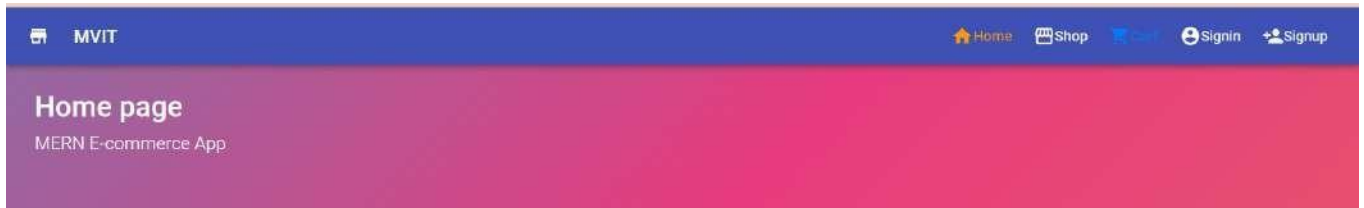
[8] Wu, Fan, Hsiao-Hui Li, and Yo-Hsin Kuo. "Reputation evaluation for choosing a trustworthy counterparty in C2C e-commerce." Electronic Commerce Research and Applications 10.4 (2011): 428-436.

[9] Sila, Ismail. "Factors affecting the adoption of B2B e-commerce technologies." Electronic commerce research 13.2 (2013): 199-236.

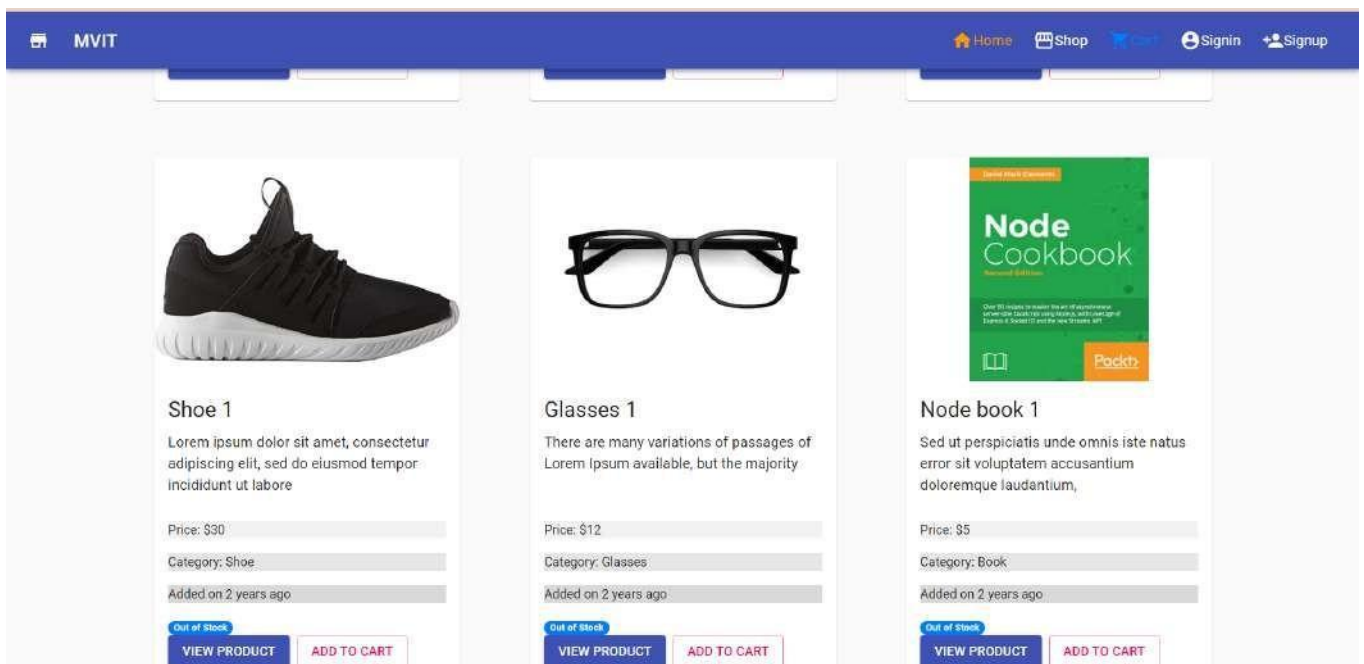
[10] Zeng, Y. E., Wen, H. J., & Yen, D. C. (2003). Customer relationship management (CRM) in business-to-business (B2B e-commerce. Information Management & Computer Security.

Complete Front-end Design

1. Home Page:



New Arrivals



Best Sellers



T-shirt 1

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, to

Price: \$15

Category: T-shirt

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Formal Shirt

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, to

Price: \$41

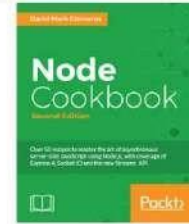
Category: Formal Shirt

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Node book 1

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium,

Price: \$5

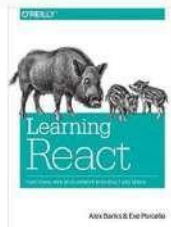
Category: Book

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Redux Book

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis.

Price: \$10

Category: Book

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Glasses 1

There are many variations of passages of Lorem Ipsum available, but the majority

Price: \$12

Category: Glasses

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Mouse

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

Price: \$22

Category: Accessories

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)

Shop Page

MVIT

Home

Shop

Cart

Signin

Signup

Shop page

Search and find books

Select

Search by name

SEARCH

Filter by categories

☐ Book

☐ T-shirt


☐ Glasses


☐ Shoe


☐ Formal Shirt

☐ Accessories

Products









Shoe 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore

Price: \$30

Category: Shoe

Added on 2 years ago

Out of Stock

VIEW PRODUCT

ADD TO CART



Glasses 1

There are many variations of passages of Lorem ipsum available, but the majority

Price: \$12

Category: Glasses

Added on 2 years ago

Out of Stock

VIEW PRODUCT

ADD TO CART



Node book 1

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium,

Price: \$5

Category: Book

Added on 2 years ago

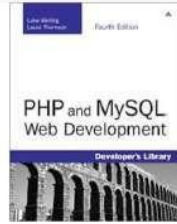
Out of Stock

VIEW PRODUCT

ADD TO CART

LOAD MORE

Copyright © MVIT 2022



PHP book

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium,

Price: \$30

Category: Book

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Python Book

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis.

Price: \$20

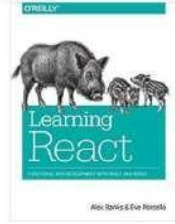
Category: Book

Added on 2 years ago

In Stock

[VIEW PRODUCT](#)

[ADD TO CART](#)



Redux Book

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis.

Price: \$10

Category: Book

Added on 2 years ago

Out of Stock

[VIEW PRODUCT](#)


[ADD TO CART](#)

Sign In page

MVIT

HomeShopCartSigninSignup

Signin to MERN E-commerce App



Sign in

Email Address *

Password *

☐ Remember me

SIGN IN

[Forgot password?](#) [Don't have an account? Sign Up](#)


Copyright © Anuj Mishra 2022.

Sign Up page

MVIT

HomeShopCartSigninSignup

Signup to MERN E-commerce App



Sign up

Full Name *

Email Address *

Password *

SIGN UP

[Already have an account? Sign in](#)

Copyright © Anuj Mishra 2022.

Dashboard Page

BRAND

HomeShopCartDashboardSignout

Dashboard

Anuj Mishra

User links

My cart

Update profile

User information

Anuj Mishra

anujmishra2234@gmail.com

Registered user

Purchase history

Shopping cart and Payment page


BRAND

HomeShopCartDashboardSignout

Shopping Cart

Manage your cart items. Add remove checkout or continue shopping.

Your cart has 1 items



T-shirt 1

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, to

Your cart summary

Total: \$15

Delivery address:

Type your delivery address here...

Choose a way to pay

Card

PayPal
Developer Error: Something went wrong. Check the console for details.

Pay

Code:

index.js

```
import { API } from '../config';

export const signup = async (user) =>
  { try {
    const response = await
fetch(`${process.env.REACT_APP_DB_URL}/${process.env.REACT_APP_ADD_USE
R_URL}`, {
  method: 'POST',
  headers: {
    Accept: 'application/json',
    'Content-Type':
    'application/json',
  },
  body: JSON.stringify(user),
});
    return await response.json();
  } catch (err) {
    console.log(err);
  }
};

// export const signin = async (user) => {
//   // console.log(name, email, password);
//   try {
//     const response = await
fetch(`${process.env.REACT_APP_DB_URL}/${process.env.REACT_APP_GET_US
E_R_EMAIL_URL}/${user.email}`, {
//     method: 'GET',
//     headers: {
//       Accept: 'application/json',
//       'Content-Type': 'application/json',
//     },
//     body: JSON.stringify(user),
//   });
//   return await response.json();
// } catch (err) {
//   console.log(err);
}
```

```

//    }
// };

export const signin = async (user) => {
  // console.log(name, email, password);
  console.log(process.env.REACT_APP_DB_URL);
  return await
fetch(`${process.env.REACT_APP_DB_URL}/${process.env.REACT_APP_GET_USE
R_EMAIL_URL}/${user.email}`, {
  method: 'GET',
  headers: {
    Accept: 'application/json',
    'Content-Type':
    'application/json',
  }
})
  .then((response) => {
    return
    response.json();
  })
  .catch((err) => {
    console.log(err);
  });
};

export const authenticate = (data, next) => {
  if (typeof window !== 'undefined') {
    localStorage.setItem('jwt', JSON.stringify(data));
    next();
  }
};

export const signout = (next) => {
  if (typeof window !== 'undefined')
  { localStorage.removeItem('jwt');
  next();
  return fetch(`${API}/signout`, {
    method: 'GET',
  })
}

```



```
.then((response) => {  
  console.log('signout',  
    response);  
})
```

```

    ))
    .catch((err) => console.log(err));
  }
};

export const isAuthenticated = () =>
{ if (typeof window ===
'undefined') {
  return false;
}

  if (localStorage.getItem('jwt')) {
    return JSON.parse(localStorage.getItem('jwt'));
  } else {
    return false;
  }
};

```

Routes.js

```

import React from 'react';
import { BrowserRouter, Switch, Route } from
'react-router-dom'; import Signup from './user/Signup';
import Signin from './user/Signin';
import Home from './core/Home';
import PrivateRoute from './auth/PrivateRoute';
import Dashboard from './user/UserDashboard';
import AdminRoute from './auth/AdminRoute';
import AdminDashboard from
'./user/AdminDashboard'; import AddCategory from
'./admin/AddCategory'; import AddProduct from
'./admin/AddProduct';
import Shop from './core/Shop';
import Product from
'./core/Product'; import Cart from
'./core/Cart'; import Orders from
'./admin/Orders'; import Profile
from './user/Profile';
import ManageProducts from './admin/ManageProducts';
import UpdateProduct from './admin/UpdateProduct';
import NotFound from './core/NotFound';

```

```
const Routes = () => {
```

```

return (
  <BrowserRouter>
    <Switch>
      <Route path="/" component={Home} exact />
      <Route path="/shop" component={Shop} exact />
      <Route path="/signin" component={Signin} exact />
      <Route path="/signup" component={Signup} exact />
      <PrivateRoute path="/user/dashboard" component={Dashboard}
exact />
      <AdminRoute path="/admin/dashboard" component={AdminDashboard}
exact />
      <AdminRoute path="/create/category" component={AddCategory}
exact />
      <AdminRoute path="/create/product" component={AddProduct}
exact />
      <Route path="/product/:productId" component={Product} exact />
      <Route path="/cart" component={Cart} exact />
      <AdminRoute path="/admin/orders" component={Orders} exact />
      <PrivateRoute path="/profile/:userId" component={Profile}
exact />
      <AdminRoute path="/admin/products" component={ManageProducts}
exact />
      <AdminRoute
        path="/admin/product/update/:productId"
        component={UpdateProduct}
        exact
      />
      <Route component={NotFound} />
    </Switch>
  </BrowserRouter>
);
};

export default Routes;

```

Server.js

```

const express = require('express');
const mongoose =
require('mongoose'); const morgan =
require('morgan');

```

```
const bodyParser = require('body-parser');
const cookieParser =
require('cookie-parser'); const cors =
require('cors');
const path = require('path');
const expressValidator =
require('express-validator'); const crypto =
require("crypto"); require('dotenv').config();
// import routes
const authRoutes =
require('./routes/auth'); const userRoutes
= require('./routes/user');
const categoryRoutes = require('./routes/category');
const productRoutes = require('./routes/product');
const braintreeRoutes =
require('./routes/braintree'); const orderRoutes =
require('./routes/order');
const utils = require("./utils");
const db = require("./db");

// app
const app = express();

// db connection
const connectDB = async () =>
{ try {
  await mongoose.connect(
    `${process.env.MONGODB_URI}/${process.env.DB_NAME}`,
    {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      useCreateIndex: true,
      useFindAndModify: false,
    }
  );
  console.log('MongoDB Connected');
} catch (err) {
  console.error(err.message);
  // exit process with failure
  process.exit(1);
}
```



```
// middlewares
app.use(morgan('dev'));
app.use(bodyParser.json());
app.use(cookieParser());
app.use(expressValidator());
app.use(cors());

// routes middleware
app.use('/api', authRoutes);
app.use('/api', userRoutes);
app.use('/api', categoryRoutes);
app.use('/api', productRoutes);
app.use('/api', braintreeRoutes);
app.use('/api', orderRoutes);

// Server static assets if in production
if (process.env.NODE_ENV === 'production') {
  // Set static folder
  app.use(express.static('client/build'));

  app.get('*', (req, res) => {
    res.sendFile(path.resolve(__dirname, 'client',
'build', 'index.html'));
  });
}
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});

// GET Methods
app.get("/get_user/id/:userID", async (req, res) => {
  utils.print_process(`GET request to get user:
${req.params.userID}`);

  let user = await db.get_user_by_id(req.params.userID);

  if(user == null) {
```

```

        utils.print_error(`Could not retrieve user with id:
${req.params.userID}`);
        return res.json({"status": "Could not retrieve user"});
    } else {
        utils.print_success(`Successfully retrieved user details: `);
        console.log(user);
        return res.json(user);
    }
})

app.get("/get_user/email/:userEmail", async (req, res) =>
{
    utils.print_process(`GET request to get user:
${req.params.userEmail}`);
    let user = await db.get_user_by_email(req.params.userEmail);

    if(user == null) {
        utils.print_error(`Could not retrieve user with email:
${req.params.userEmail}`);
        return res.json({"status": "Could not retrieve user"});
    } else {
        utils.print_success(`Successfully retrieved user details: `);
        console.log(user);
        return res.json(user);
    }
})

// POST Methods
app.post('/add_user', async (req, res) => {
    // let result = await
    db.get_user_by_email(req.body.email); let result = await
    db.add_user(req.body)

    if(result.email == req.body.email) {
        console.log('[+] Successfully added
user');
        res.json({"response": "Successfully added user"});
    } else {
        console.log("[-] Could not add user");
        res.json({"response": "Could not add
user"});
    }
})

```


product.js

```
const mongoose = require('mongoose');
const { ObjectId } = mongoose.Schema;

const productSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      trim: true,
      required: true,
      maxlength: 32,
    },
    description: {
      type: String,
      required: true,
      maxlength:
        2000,
    },
    price: {
      type: Number,
      trim: true,
      required: true,
      maxlength: 32,
    },
    category: {
      type: ObjectId,
      ref: 'Category',
      required: true,
    },
    quantity: {
      type:
        Number,
    },
    sold: {
      type: Number,
      default: 0,
    },
    photo: {
      data: Buffer,
```

```
    contentType:  
    String,  
  },  
}
```

```

    shipping: {
      required: false,
      type: Boolean,
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model('Product', productSchema);

```

user.js

```

const mongoose =
require('mongoose'); const crypto =
require('crypto');

const userSchema = new mongoose.Schema(
{
  name: {
    type: String, trim:
      true,
    required:
      true,
    maxLength:
      32,
  },
  email: {
    type: String, trim:
      true,
    required:
      true, unique:
      true,
  },
  password: {
    type:
      String,
    required: true,
  },
  about: {
    type: String,

```

```

    { timestamps: true }
);

// virtual field

// userSchema
//   .virtual('password')
//   .set(function (password) {
//     this._password = password;
//     this.salt = uuidv1();
//     this.hashd_password = this.encryptPassword(password);
//   })
//   .get(function () {
//     return [this.salt, this._password];
//   });

// userSchema.methods = {
//   authenticate: function(plainText) {
//     return this.encryptPassword(plainText)
// == this.hashd_password;
//   },

//   encryptPassword: function (password) {
//     if (!password) return '';
//     try {
//       return crypto
//         .createHmac('sha1', this.salt)
//         .update(password)
//         .digest('hex');
//     } catch (err) {
//       return '';
//     }
//   },
// };

userSchema.methods.setPassword = function (password)
{
  this.salt =
  crypto.randomBytes(16).toString('hex');
  this.hash =
  crypto.pbkdf2Sync(password, this.salt,
    1000, 64, `sha512`).toString(`hex`);

```



```
userSchema.methods.validPassword = function (password)
{
  var hash = crypto.pbkdf2Sync(password,
    this.salt, 1000, 64, `sha512`).toString(`hex`);
  return this.hash === hash;
};

module.exports = mongoose.model('users', userSchema);
```