Distributed Systems - SE3020

Assignment 2 – REST API

Group Id - 2021S1_REG_WE_38

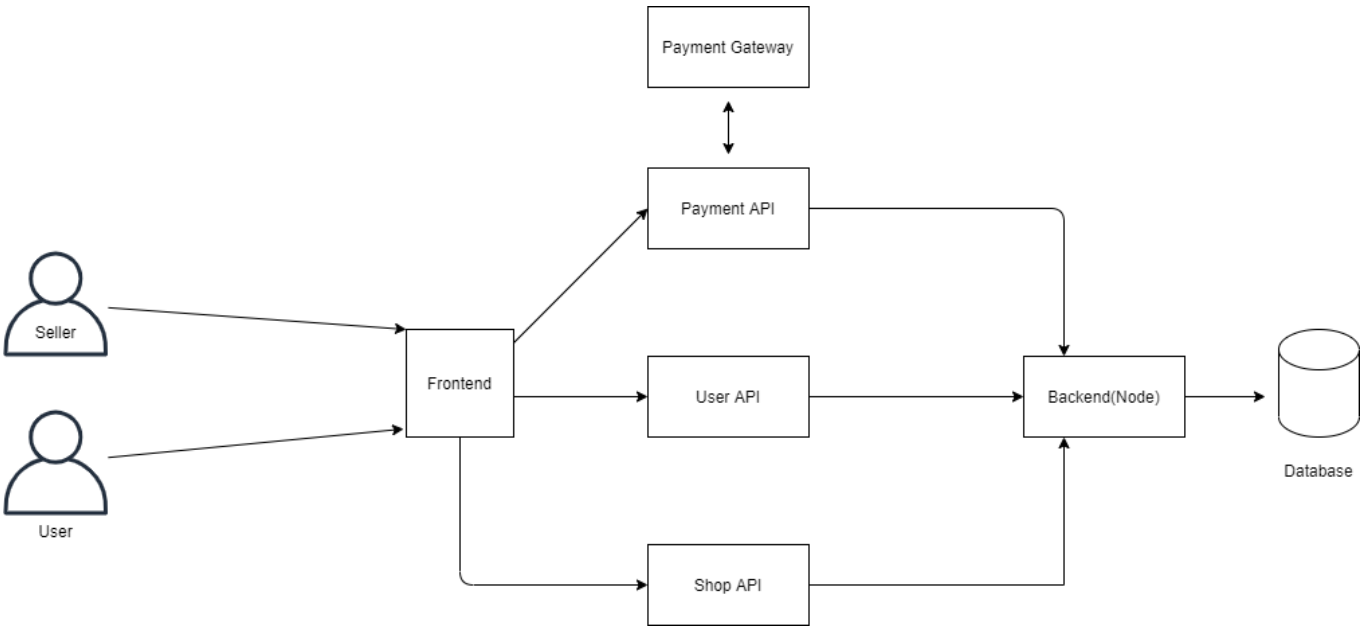
Student ID	Student Name
IT19216874	R. Y. Senevirathne
IT19217796	M.A.A.H.Ratnayake
IT19362854	S.L.Abeygunawardana
IT19105826	N.D.Meegoda

Introduction

BookMart is an online book shop web application. Customers can view book details and prices through the system without registering to the system. But if a new customer wants to buy books from BookMart, the customer must register to the system. Registered customers have to log in to the system using email and password to continue purchasing process to buy books. After login to the system customer can update profile details and view their orders from the account. A Customer can add books to the cart. After that customer can check out all items in the cart and order items by adding delivery details. Finally, the customer can pay the total bill through PayPal. Seller work as the administrator in the BookMart by managing book details and orders. The seller can add books to the system and also update, delete book details according to the requirement. We have used jwt tokens and .env file to enhance the security.

RESTful Web Service – Express and Node
Web Client – React
Database – MongoDB atlas

High Level Architecture Diagram



Implementation

User Authentication

User can login to the system by using email and password. After validating the user details user navigates to seller page or user page according to the backend validation. Also user get the bear token to continue the session in secured way. This session token is saved in the local storage and expired after some time. All user operations handled by the usercontroller.js in the backend.

From User controller.js – User Login

```
// @desc  Get the user profile
// @route GET /api/users/userAccount
// @access Private

const getUserAccount = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id)

  if (user) {
    res.json({
      _id: user._id,
      name: user.name,
      email: user.email,
      isAdmin: user.isAdmin,
    })
  } else {
    res.status(404)
    throw new Error('User cannot be found')
  }
})
```

From User controller.js – Register User

```
// @desc  Add a new user
// @route POST /api/users/
// @access Public

const addUser = asyncHandler(async (req, res) => {
  const { name, email, password } = req.body
```

```
  const chk_user_existence = await User.findOne({ email: email })

  if (chk_user_existence) {
    res.status(400)
    throw new Error(`There's a member already registered with that mail`)
  }

  const user = await User.create({
    name,
    email,
    password,
  })
  if (user) {
    res.status(201).json({
      _id: user._id,
      name: user.name,
      email: user.email,
      isAdmin: user.isAdmin,
      token: genToken(user._id),
    })
  } else {
    res.status(400)
    throw new Error('This user account cannot be created. Try again')
  }
})
```

To protect routes, shield is used to separate special roues from public routes.

From UserRoutes.js

```
import express from 'express'
import { authUser, getUserAccount, addUser, updateUserAccount, getUsers } from '../contr
ollers/userController.js'
const router = express.Router()
import {shield, admin} from '../middleware/validateTokenMiddleware.js'

router.post('/login', authUser) // to login
router.route('/account').get(shield, getUserAccount).put(shield, updateUserAccount) // t
o authorise with jwt tokens  and view user profile or update user profile
//router.post('/', addUser)// to create a new user

router.route('/').post(addUser).get(shield,admin, getUsers)
```

```
export default router
```

Buyer

Unregistered user can add items to the cart. Registered user to buy items from Bookmark. As an example to check out all the items in the care, users have to log in to the system. To place an order user have to login to the system.

From orderAction.js

```
export const AddOrder = (order) => async (dispatch, getState) => {
    try {
      dispatch({
        type: ORDER_ADDED_REQUEST,
      })

      const {
        userSignin: { userDetails },
      } = getState()



      const config = {
        headers: {
          'Content-Type': 'application/json',
          Authorization: `Bearer ${userDetails.token}`,
        },
      }

      const { data } = await axios.post(`/api/orders`, order, config)

      dispatch({
        type: ORDER_ADDED_SUCCESS,
        payload: data,
      })

    } catch (error) {
      dispatch({
        type: ORDER_ADDED_FAIL,
        payload:
```

```
          error.response && error.response.data.message
            ? error.response.data.message
            : error.message,
      })
    }
  }
```

Appendix

Frontend

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Provider} from 'react-redux'
import store from './store'
import './bootstrap.min.css' //importing bootstrap min file
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

App.js

```
import { BrowserRouter as Router, Route } from 'react-router-dom'
import { Container } from 'react-bootstrap'
import Header from './components/Header'
import Footer from './components/Footer'
import Home from './screens/Home'
```

```
import ProductView from './screens/ProductView'
import CartView from './screens/CartView'
import  SignInView from './screens/SignInView'
import AddUserView from './screens/AddUserView'
import UserProfileView from './screens/userProfileView'
import shippingprocessView from './screens/shippingprocessView'
import paymentmethodView from './screens/paymentmethodView'
import ConfirmOrder from './screens/confirmOrder'
import OrderView from './screens/orderView'
import userList from './screens/userList'
import ProductlistView from './screens/productlistView'

const App = () => {
  return (
    <Router>
      <Header />
      <main className='py-2'>
        <Container>


          <Route path='/product/:id' component={ProductView} />
          <Route path='/cart/:id?' component={CartView} />
          <Route path='/account' component={UserProfileView} />
          <Route path='/login' component={SignInView} />
          <Route path='/admin/productlist' component={ProductlistView} />
          <Route path='/orders' component={OrderView} />
          <Route path='/paymentprocess' component={paymentmethodView} />
          <Route path='/admin/userlist' component={userList} />
          <Route path='/confirmOrder' component={ConfirmOrder} />
          <Route path='/shippingprocess' component={shippingprocessView} />
          <Route path='/register' component={AddUserView} />
           {/* Here we have used route to go to homepage if path is exactly / using exac
t keyword */}
           <Route path='/' component={Home} exact />




        </Container>
      </main>
      <Footer />
    </Router>
  )
}
```

```
export default App
```

AdduserView.js

```
import React, { useState, useEffect } from 'react'
import { Link } from 'react-router-dom'
import { Form, Button, Row, Col } from 'react-bootstrap'
import FormC from '../components/Form_layout'
import { useDispatch, useSelector } from 'react-redux'
import Alertmsg from '../components/Alert'
import Buffer from '../components/buffer'
import  {AddUser}  from '../actions/userActions'

const AddUserView = ({ location, history }) => {


  const [name, setAccUName] = useState('')
  const [email, setUsername] = useState('')
  const [password, setPass] = useState('')
  const [confirmpassword, setConPass] = useState('')
  const [messagePop, setmessagePop] = useState(null)

  const dispatch = useDispatch()
  const userCreate = useSelector(state => state.userCreate)
  const { error, userDetails, loading } = userCreate

  const redirect = location.search ? location.search.split('=')[1] : '/'

  useEffect(() => {
    if (userDetails) {
      history.push(redirect)
    }
  }, [userDetails, redirect, history])

  const submitHandler = (e) => {
    e.preventDefault()
    if(password !== confirmpassword){
        setmessagePop('Passwords are not matching')
    } else {
        dispatch(AddUser(name,email, password))
    }

  }
```

```
return (
  <FormC>
    <h1>Create Account</h1>

    {messagePop && <Alertmsg variant='danger'>{messagePop}</Alertmsg>}
    {error && <Alertmsg variant='danger'>{error}</Alertmsg>}
    {loading && <Buffer />}

    <Form onSubmit={submitHandler}>

    <Form.Group controlId='name'>
        <Form.Label>Your Name</Form.Label>
        <Form.Control
          type='name'
          placeholder='Enter your name'
          value={name}
          onChange={(e) => setAccUName(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='email'>
        <Form.Label>Username</Form.Label>
        <Form.Control
          type='email'
          placeholder='Enter your username'
          value={email}
          onChange={(e) => setUsername(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='password'>
        <Form.Label>Password</Form.Label>
        <Form.Control
          type='password'
          placeholder='Enter the password'
          value={password}
          onChange={(e) => setPass(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='confirmpassword'>
        <Form.Label>Confirm Password</Form.Label>
        <Form.Control
          type='password'
          placeholder='Confirm your password'
```

```
            value={confirmpassword}
            onChange={(e) => setConPass(e.target.value)}
          ></Form.Control>
        </Form.Group>

        <Button type='submit' variant='primary'>
          Add Me
        </Button>
      </Form>

      <Row className='my=6'>
        <Col>
          Already have a account?
          <Link to={redirect ? `/login?redirect=${redirect}` : '/login'}>
            Sign In
          </Link>
        </Col>
      </Row>
    </FormC>
  )
}

export default AddUserView
```

CartView.js

```
import React, { useEffect } from 'react'
import { Link } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import {
  Row,
  Col,
  Image,
  Button,
  Card,
  ListGroup,
  FormControl,
} from 'react-bootstrap'
import { addToCart, removeFromCart } from '../actions/cartActions'

import Alertmsg from '../components/Alert'


const CartView = ({ match, location, history }) => {
```

```
const bookID = match.params.id

const stkqty = location.search ? Number(location.search.split('=')[1]) : 1

// testing purposes console.log(stkqty)

const dispatch = useDispatch()

const cart = useSelector((state) => state.cart)
const { cartItems } = cart

useEffect(() => {
  if (bookID) {
    dispatch(addToCart(bookID, stkqty))
  }
}, [dispatch, bookID, stkqty])

const removeFrmCartHandler = (id) => {
  dispatch(removeFromCart(id))
}

const checkoutHandler = () => {
  history.push('/login?redirect=shippingprocess')
}

return (
  <Row>
    <Col md={8}>
      <h1>Shopping Cart</h1>
      {cartItems.length === 0 ? (
        <Alertmsg>
          The Cart Is Empty<Link to='/'> Back to Home</Link>
        </Alertmsg>
      ) : (
        <ListGroup variant='flush'>
          {cartItems.map((item) => (
            <ListGroup.Item key={item.product}>
              <Row>
                <Col md={2}>
                  <Image src={item.image} alt={item.name} fluid rounded />
                </Col>
                <Col md={3}>
                  <Link to={`/product/${item.product}`}>{item.name}</Link>
                </Col>
                <Col md={2}>Lkr {item.price}</Col>
```

```
                <Col md={2}>
                  <FormControl
                    as='select'
                    value={item.oqty}
                    onChange={(e) =>
                      dispatch(
                        addToCart(item.product, Number(e.target.value))
                      )
                    }
                  >
                    {[...Array(item.countInStock).keys()].map((x) => (
                      <option key={x + 1} value={x + 1}>
                        {x + 1}
                      </option>
                    ))}
                  </FormControl>
                </Col>
                <Col md={2}>
                  <Button
                    type='button'
                    variant='light'
                    onClick={() => removeFrmCartHandler(item.product)}
                  >
                    <i className='fas fa-trash'></i>
                  </Button>
                </Col>
              </Row>
            </ListGroup.Item>
          ))}
        </ListGroup>
      )}
    </Col>
    <Col md={4}>
      <Card>
        <ListGroup variant='flush'>
          <ListGroup.Item>
            <h2>SubTotal ({cartItems.reduce((acc, cur) => acc + cur.oqty,0)}) Books</h
2>
            Lkr {cartItems.reduce((acc,curIt)=> acc + curIt.oqty * curIt.price,0).toFi
xed(2)}
          </ListGroup.Item>
          <ListGroup.Item>
              <Button type='button' className='btn-
block' disabled={cartItems.length=== 0} onClick={checkoutHandler}> Checkout </Button>
          </ListGroup.Item>
```

```
          </ListGroup>
        </Card>
      </Col>
    </Row>
  )
}

export default CartView
```

ConfirmOrder.js

```
import React, { useEffect } from 'react'

import { Button, Row, Col, ListGroup, Image, Card } from 'react-bootstrap'
import Alertmsg from '../components/Alert'
import { Link } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import Checkoutnavigator from '../components/checkoutnavigator'
import { AddOrder } from '../actions/orderActions'

const ConfirmOrder = ({history}) => {
  const dispatch = useDispatch()

  const cart = useSelector((state) => state.cart)

  const orderAdd = useSelector(state => state.orderAdd)
  const {order, success, error} = orderAdd
  useEffect(() => {
    if(success){
      history.push(`/order/${order._id}`)
    }
    // eslint-disable-next-line
  },[history, success])

  const confirmOrderHandler = () => {
    dispatch(
      AddOrder({
        orderItems: cart.cartItems,

        shippingAddress: cart.shippingDetails,

       paymentMethod: cart.paymentDetails,
        itemsPrice: cart.itemsCost,
```

```
      shippingPrice: cart.shippingCost,
      totalPrice: cart.totalCost,
    })
  )
}

cart.itemsCost = cart.cartItems.reduce(
  (acc, item) => acc + item.price * item.oqty,
  0
)
cart.shippingCost = cart.itemsCost > 1500 ? 0 : 150
cart.totalCost = Number(cart.itemsCost + cart.shippingCost)

return (
  <>
    <Checkoutnavigator s1 s2 s3 s4 />
    <Row>
      <Col md={8}>
        <ListGroup variant='flush'>
          <ListGroup.Item>
            <h2>Shipping to</h2>
            <p>
              <strong>Address: </strong>
              {cart.shippingDetails.address}, {cart.shippingDetails.city},{' '}
              {cart.shippingDetails.postalCode},{cart.shippingDetails.country}
              , {cart.shippingDetails.contactno}
            </p>
          </ListGroup.Item>
          <ListGroup.Item>
            <h2>Payment type</h2>
            <p>
              <strong>Through: </strong>
              {cart.paymentDetails}
            </p>
          </ListGroup.Item>
          <ListGroup.Item>
            <h2>Ordered Items</h2>
            {cart.cartItems.length === 0 ? (
              <Alertmsg> Cart is a empty one </Alertmsg>
            ) : (
              <ListGroup variant='flush'>
                {cart.cartItems.map((item, id) => (
                  <ListGroup.Item key={id}>
                    <Row>
                      <Col md={1}>
```

```
                    <Image
                      src={item.image}
                      alt={item.name}
                      width='30'
                      height='40'
                      fliud
                      rounded
                    />
                  </Col>

                  <Col>
                    <Link to={`/product/${item.product}`}>
                      {item.name}
                    </Link>
                  </Col>
                  <Col md={4}>
                    {item.oqty} X LKR {item.price} = LKR{' '}
                    {item.oqty * item.price}
                  </Col>
                </Row>
              </ListGroup.Item>
            ))}
          </ListGroup>
        )}
      </ListGroup.Item>
    </ListGroup>
  </Col>

  <Col md={4}>
    <Card>
      <ListGroup variant='flush'>
        <ListGroup.Item>
          <h3>Summary</h3>
        </ListGroup.Item>
        <ListGroup.Item>
          <Row>
            <Col>OrderCost :</Col>
            <Col>${cart.itemsCost}</Col>
          </Row>
        </ListGroup.Item>
        <ListGroup.Item>
          <Row>
            <Col>Shipping Cost :</Col>
            <Col>${cart.shippingCost}</Col>
          </Row>
```

```
              </ListGroup.Item>
              <ListGroup.Item>
                <Row>
                  <Col>Total Cost : </Col>
                  <Col>${cart.totalCost}</Col>
                </Row>
              </ListGroup.Item>
              <ListGroup.Item>
                {error && <Alertmsg variant='danger'>{error}</Alertmsg>}
              </ListGroup.Item>
              <ListGroup.Item>
                <Button
                  type='button'
                  disabled={cart.cartItems === 0}
                  onClick={confirmOrderHandler}
                >
                  Confirm Order
                </Button>
              </ListGroup.Item>
            </ListGroup>
          </Card>
        </Col>
      </Row>
    </>
  )
}

export default ConfirmOrder
```

Home.js

```
import React, { useEffect } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import { Row, Col } from 'react-bootstrap'
import Product from '../components/Product'
import Buffer from '../components/buffer'
import Alertmsg from '../components/Alert'
import { listProducts } from '../actions/productActions'

const Home = () => {
  const dispatch = useDispatch()

  const productList = useSelector((state) => state.productList)
```

```
  const { loading, error, products } = productList

  useEffect(() => {
    dispatch(listProducts())
  }, [dispatch])

  return (
    <>
      <h1> Latest Books </h1>
      {loading ? (
        <Buffer />
      ) : error ? (
        <Alertmsg variant='danger'>{error}</Alertmsg>
      ) : (
        <Row>
          {products.map((product) => (
            <Col key={product._id} sm={12} md={6} lg={4} xl={3}>
              <Product product={product} />
            </Col>
          ))}
        </Row>
      )}
    </>
  )
}

export default Home
```

OrderView.js

```
import React, { useState, useEffect } from 'react'
import { Row, Col, ListGroup, Image, Card } from 'react-bootstrap'
import axios from 'axios'
import { PayPalButton } from 'react-paypal-button-v2'
import Alertmsg from '../components/Alert'
import { Link } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import Buffer from '../components/buffer'
import { getOrderInfo, OrderPayment } from '../actions/orderActions'
import { ORDER_PAYMENT_CLEAN } from '../constants/orderConstants'

const OrderView = ({ match }) => {
  const orderId = match.params.id
```

```
const [sdkPayPal, setSDKpaypal] = useState(false)
const dispatch = useDispatch()

const orderInfo = useSelector((state) => state.orderinfo)
const { order, loading, error } = orderInfo
const orderPayment = useSelector((state) => state.orderPayment)
const { loading: loadingpay, success: successpay } = orderPayment

if(!loading){
    order.itemsCost = order.orderItems.reduce(
      (acc, item) => acc + item.price * item.oqty,
      0
    )
}

console.log(order);

useEffect(() => {
  //PAYPAL SCRIPT CONFIGURATION
  const payPal_dev_script = async () => {
    const { data: clientId } = await axios.get('/api/config/paypal')
    const script = document.createElement('script')
    script.type = 'text/javascript'
    script.src = `https://www.paypal.com/sdk/js?client-id=${clientId}`
    script.async = true
    script.onload = () => {
      setSDKpaypal(true)
    }
    document.body.appendChild(script)
  }

  if (!order || successpay) {
    if (!order || order._id !== orderId) {
      dispatch({ type: ORDER_PAYMENT_CLEAN })
      dispatch(getOrderInfo(orderId))
    }
  } else if (!order.isPaid) {
    if (!window.paypal) {
      payPal_dev_script()
    } else {
      setSDKpaypal(true)
    }
  }
}, [dispatch, order, orderId, successpay])
```

```
const successPaymenthandler = (paymentResult) => {
  console.log(paymentResult)
  dispatch(OrderPayment(orderId, paymentResult))
}
return loading ? (
  <Buffer />
) : error ? (
  <Alertmsg variant='danger'>{error}</Alertmsg>
) : (
  <>
    <h2>Order {order._id}</h2>

    <Row>
      <Col md={8}>
        <ListGroup variant='flush'>
          <ListGroup.Item>
            <h2>Shipping to</h2>
            <strong>Name:</strong> {order.user.name}
            <a href={`mailto:${order.user.email}`}>{order.user.email}</a>
            <p>
              <strong>Address: </strong>
              {order.shippingAddress.address}, {order.shippingAddress.city},{' '}
              {order.shippingAddress.postalCode},
              {order.shippingAddress.country},{' '}
              {order.shippingAddress.contactno}
            </p>
            {order.isDelivered ? (
              <Alertmsg variant='success'>
                Delivered on {order.deliveredAt}
              </Alertmsg>
            ) : (
              <Alertmsg variant='danger'>Not Delivered</Alertmsg>
            )}
          </ListGroup.Item>
          <ListGroup.Item>
            <h2>Payment type</h2>
            <p>
              <strong>Through: </strong>
              {order.paymentDetails}
            </p>
            {order.isPaid ? (
              <Alertmsg variant='success'>Paid on {order.paidAt}</Alertmsg>
            ) : (
              <Alertmsg variant='danger'>Not paid</Alertmsg>
            )}
```

```
          </ListGroup.Item>
          <ListGroup.Item>
            <h2>Ordered Items</h2>
            {order.orderItems.length === 0 ? (
              <Alertmsg> Order is a null </Alertmsg>
            ) : (
              <ListGroup variant='flush'>
                {order.orderItems.map((item, id) => (
                  <ListGroup.Item key={id}>
                    <Row>
                      <Col md={1}>
                        <Image
                          src={item.image}
                          alt={item.name}
                          width='30'
                          height='40'
                          fliud
                          rounded
                        />
                      </Col>

                      <Col>
                        <Link to={`/product/${item.product}`}>
                          {item.name}
                        </Link>
                      </Col>
                      <Col md={4}>
                        {item.oqty} X LKR {item.price} = LKR{' '}
                        {item.oqty * item.price}
                      </Col>
                    </Row>
                  </ListGroup.Item>
                ))}
              </ListGroup>
            )}
          </ListGroup.Item>
        </ListGroup>
      </Col>

      <Col md={4}>
        <Card>
          <ListGroup variant='flush'>
            <ListGroup.Item>
              <h3>Summary</h3>
            </ListGroup.Item>
```

```
              <ListGroup.Item>
                <Row>
                  <Col>OrderCost :</Col>
                  <Col>${order.itemsCost}</Col>
                </Row>
              </ListGroup.Item>
              <ListGroup.Item>
                <Row>
                  <Col>Shipping Cost :</Col>
                  <Col>${order.shippingCost}</Col>
                </Row>
              </ListGroup.Item>
              <ListGroup.Item>
                <Row>
                  <Col>Total Cost : </Col>
                  <Col>${order.totalCost}</Col>
                </Row>
              </ListGroup.Item>

              {!order.isPaid && (
                <ListGroup.Item>
                  {loadingpay && <Buffer />}
                  {!sdkPayPal ? (
                    <Buffer />
                  ) : (
                    <PayPalButton
                      amount={order.totalCost}
                      onSuccess={successPaymenthandler}
                    />
                  )}
                </ListGroup.Item>
              )}
            </ListGroup>
          </Card>
        </Col>
      </Row>
    </>
  )
}

export default OrderView
```

PaymentMethodView.js

```
import React, { useState } from 'react'
import { Form, Button, Col } from 'react-bootstrap'
import FormC from '../components/Form_layout'
import { useDispatch, useSelector } from 'react-redux'
import { recordPaymentmethod } from '../actions/cartActions'
import Checkoutnavigator from '../components/checkoutnavigator'

const PaymentmethodView = ({ history }) => {
  const cart = useSelector((state) => state.cart)
  const { shippingDetails } = cart

  if (!shippingDetails) {
    history.push('/shippingprocess')
  }

  const [paymentMethod, setPaymentmethod] = useState('PayPal')

  const dispatch = useDispatch()

  const submithandler = (e) => {
    e.preventDefault()
    dispatch(recordPaymentmethod(paymentMethod))
    history.push('/confirmOrder')
  }

  return (
    <FormC>
      <Checkoutnavigator s1 s2 s3 />
      <h2> Choose a Payment Method</h2>

      <Form onSubmit={submithandler}>
        <Form.Group>
          <Form.Label >Available Methods</Form.Label>



        <Col>

            <Form.Check type='radio' label='PayPal' id='Paypal' checked onChange={(e) =>
 setPaymentmethod(e.target.value)} name='paymentGateway' value='Paypal' ></Form.Check>
            <Form.Check type='radio' label='Stripe' id='Stripe'  onChange={(e) => setPay
mentmethod(e.target.value)} name='paymentGateway' value='Stripe' ></Form.Check>
```

```
            <Form.Check type='radio' label='Mobile' id='Mobile'  onChange={(e) => setPay
mentmethod(e.target.value)} name='paymentGateway' value='Mobile' ></Form.Check>

        </Col>

        </Form.Group>

        <Button type='submit' variant='primary'>
          Next
        </Button>
      </Form>
    </FormC>
  )
}

export default PaymentmethodView
```

ProductView.js

```
import React, { useState, useEffect } from 'react'
import { Link } from 'react-router-dom'
import { useDispatch, useSelector } from 'react-redux'
import {
  Row,
  Col,
  Image,
  ListGroup,
  Card,
  Button,
  FormControl,
} from 'react-bootstrap'
import Rating from '../components/Rating'
import Buffer from '../components/buffer'
import Alertmsg from '../components/Alert'
import { listProductInfo } from '../actions/productActions'

const ProductView = ({ history, match }) => {
  const [stk_count, set_stkCount] = useState(1)

  const dispatch = useDispatch()

  const productInfo = useSelector((state) => state.productInfo)
  const { loading, error, product } = productInfo

  useEffect(() => {
```

```
  dispatch(listProductInfo(match.params.id))
}, [dispatch, match])

const cartHandler = () => {
  history.push(`/cart/${match.params.id}?stkqty=${stk_count}`)

}

return (
  <>
    <Link className='btn btn-light my-3' to='/'>
      Back
    </Link>

    {loading ? (
      <Buffer />
    ) : error ? (
      <Alertmsg variant='danger'>{error}</Alertmsg>
    ) : (
      <Row>
        <Col md={6}>
          <Image
            src={product.b_image}
            height='450'
            width='400'
            alt={product.b_name}
            fluid
          />
        </Col>
        <Col md={3}>
          <ListGroup variant='flush'>
            <ListGroup.Item>
              <h3>{product.b_name}</h3>
            </ListGroup.Item>
            <ListGroup.Item>
              <Rating
                value={product.rating}
                text={` ${product.numReviews} reviews `}
              />
            </ListGroup.Item>
            <ListGroup.Item>Price: LKR {product.price}</ListGroup.Item>
            <ListGroup.Item>
              Description: {product.b_description}
            </ListGroup.Item>
          </ListGroup>
```

```
          </Col>
          <Col md={3}>
            <Card>
              <ListGroup variant='flush'>
                <ListGroup.Item>
                  <Row>
                    <Col>Price:</Col>
                    <Col>
                      <strong>{product.price}</strong>
                    </Col>
                  </Row>
                </ListGroup.Item>

                <ListGroup.Item>
                  <Row>
                    <Col>Status:</Col>
                    <Col>
                      {product.countInStock > 0 ? 'In Stock' : 'Out of Stock'}
                    </Col>
                  </Row>
                </ListGroup.Item>

                {/*if products are available in the stock */}

                {product.countInStock > 0 && (
                  <ListGroup.Item>
                    <Row>
                    <Col>Quantity</Col>
                      <Col>

                        <FormControl
                          as='select'
                          value={stk_count}
                          onChange={(e) => set_stkCount(e.target.value)}
                        >
                          {[...Array(product.countInStock).keys()].map((x) => (
                            <option key={x + 1} value={x + 1}>
                              {x + 1}
                            </option>
                          ))}
                        </FormControl>
                      </Col>
                    </Row>
                  </ListGroup.Item>
                )}
```

```
                <ListGroup.Item>
                  <Button
                   onClick={cartHandler}
                    className='btn-block'
                    type='button'
                    disabled={product.countInStock === 0}
                  >
                    Add To cart
                  </Button>
                </ListGroup.Item>
              </ListGroup>
            </Card>
          </Col>
        </Row>
      )}
    </>
  )
}

export default ProductView
```

ShippingProcessView.js

```
import React, { useState } from 'react'
import { Form, Button } from 'react-bootstrap'
import FormC from '../components/Form_layout'
import { useDispatch, useSelector } from 'react-redux'
import {recordShippinginfo} from '../actions/cartActions'
import Checkoutnavigator from '../components/checkoutnavigator'

const ShippingprocessView = ({ history }) => {
  const cart = useSelector((state) => state.cart)
  const { shippingDetails } = cart

  const [address, setAddress] = useState(shippingDetails.address)
  const [city, setCity] = useState(shippingDetails.city)
  const [postalCode, setPostalcode] = useState(shippingDetails.postalCode)
  const [contactno, setContactNo] = useState(shippingDetails.contactno)
  const [country, setCountry] = useState(shippingDetails.country)


  const dispatch = useDispatch()
```

```
const submithandler = (e) => {
  e.preventDefault()
  dispatch(recordShippinginfo({address, city, postalCode, contactno, country}))
  history.push('/paymentprocess')
}

return (
  <FormC>
    <Checkoutnavigator s1 s2 />
    <h2> Shipping Page</h2>

    <Form onSubmit={submithandler}>
      <Form.Group controlId='address'>
        <Form.Label>Residential Adress</Form.Label>
        <Form.Control
          type='text'
          placeholder='Enter your address'
          value={address}
          required
          onChange={(e) => setAddress(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='city'>
        <Form.Label>City</Form.Label>
        <Form.Control
          type='text'
          placeholder='Enter your city'
          value={city}
          required
          onChange={(e) => setCity(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='postalCode'>
        <Form.Label>Postal Code</Form.Label>
        <Form.Control
          type='text'
          placeholder='Enter your postalCode'
          value={postalCode}
          required
          onChange={(e) => setPostalcode(e.target.value)}
        ></Form.Control>
      </Form.Group>
```

```
        <Form.Group controlId='contactno'>
          <Form.Label>Contact No</Form.Label>
          <Form.Control
            type='text'
            placeholder='Enter your contact number'
            value={contactno}
            required
            onChange={(e) => setContactNo(e.target.value)}
          ></Form.Control>
        </Form.Group>

        <Form.Group controlId='country'>
          <Form.Label>Country</Form.Label>
          <Form.Control
            type='text'
            placeholder='Enter your country'
            value={country}
            required
            onChange={(e) => setCountry(e.target.value)}
          ></Form.Control>
        </Form.Group>

        <Button type='submit' variant='primary'>
          {' '}
          Next{' '}
        </Button>
      </Form>
    </FormC>
  )
}

export default ShippingprocessView
```

SignInView.js

```
import React, { useState, useEffect } from 'react'
import { Link } from 'react-router-dom'
import { Form, Button, Row, Col } from 'react-bootstrap'
import FormC from '../components/Form_layout'
import { useDispatch, useSelector } from 'react-redux'
import Alertmsg from '../components/Alert'
import Buffer from '../components/buffer'
import  {Sign_In}  from '../actions/userActions'
```

```
const SignInView = ({ location, history }) => {
  const [email, setUsername] = useState('')
  const [password, setPass] = useState('')

  const dispatch = useDispatch()
  const userSignin = useSelector(state => state.userSignin)
  const { error, userDetails, loading } = userSignin

  const redirect = location.search ? location.search.split('=')[1] : '/'

  useEffect(() => {
    if (userDetails) {
      history.push(redirect)
    }
  }, [userDetails, redirect, history])

  const submitHandler = (e) => {
    e.preventDefault()
    dispatch(Sign_In(email,password))
  }

  return (
    <FormC>
      <h1>Sign in to the account</h1>
      {error && <Alertmsg variant='danger'>{error}</Alertmsg>}
      {loading && <Buffer />}

      <Form onSubmit={submitHandler}>
        <Form.Group controlId='email'>
          <Form.Label>Username</Form.Label>
          <Form.Control
            type='email'
            placeholder='Enter your username'
            value={email}
            onChange={(e) => setUsername(e.target.value)}
          ></Form.Control>
        </Form.Group>

        <Form.Group controlId='password'>
          <Form.Label>Password</Form.Label>
          <Form.Control
            type='password'
            placeholder='Enter the password'
            value={password}
```

```
            onChange={(e) => setPass(e.target.value)}
          ></Form.Control>
        </Form.Group>

        <Button type='submit' variant='primary'>
          Log In
        </Button>
      </Form>

      <Row className='my=6'>
        <Col>
          Don't have a account?
          <Link to={redirect ? `/register?redirect=${redirect}` : '/register'}>
            Sign Up
          </Link>
        </Col>
      </Row>
    </FormC>
  )
}

export default SignInView
```

UserProfileView.js

```
import React, { useState, useEffect } from 'react'
import { Form, Button, Row, Col } from 'react-bootstrap'
import { useDispatch, useSelector } from 'react-redux'
import Alertmsg from '../components/Alert'
import Buffer from '../components/buffer'
import { ViewuserInfo, updateUserAccount } from '../actions/userActions'
import {USER_UPDATE_ACC_CLEAN} from '../constants/userConstants'

const UserProfileView = ({ history }) => {

  const [name, setAccUName] = useState('')
  const [email, setUsername] = useState('')
  const [password, setPass] = useState('')
  const [confirmpassword, setConPass] = useState('')
  const [messagePop, setmessagePop] = useState(null)

  const dispatch = useDispatch()
```

```
const userInfo = useSelector((state) => state.userInfo)
const { error, user, loading } = userInfo

const userSignin = useSelector((state) => state.userSignin)
const { userDetails } = userSignin

const userUpdateAcc = useSelector((state) => state.userUpdateAcc)
const { success } = userUpdateAcc

useEffect(() => {
  if (!userDetails) {
    history.push('/login')
  } else {
    if ( !user || !user.name || success) {
      dispatch({type: USER_UPDATE_ACC_CLEAN})
      dispatch(ViewuserInfo('account'))
    } else {
      setAccUName(user.name)
      setUsername(user.email)
    }
  }
}, [dispatch, userDetails, history,user, success])

const submitHandler = (e) => {
  e.preventDefault()
  if (password !== confirmpassword) {
    setmessagePop('Passwords are not matching')
  } else {
    dispatch(updateUserAccount({id:user._id, name, email, password}))
  }
}


return <Row>
  <Col md={3}>
    <h2>User Account</h2>

    {messagePop && <Alertmsg variant='danger'>{messagePop}</Alertmsg>}
    {error && <Alertmsg variant='danger'>{error}</Alertmsg>}
    {success && <Alertmsg variant='success'>Updated Successfully</Alertmsg>}
    {loading && <Buffer />}

    <Form onSubmit={submitHandler}>
      <Form.Group controlId='name'>
        <Form.Label>Your Name</Form.Label>
```

```
        <Form.Control
          type='name'
          placeholder='Enter your name'
          value={name}
          onChange={(e) => setAccUName(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='email'>
        <Form.Label>Email</Form.Label>
        <Form.Control
          type='email'
          placeholder='Enter your username'
          value={email}
          onChange={(e) => setUsername(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='password'>
        <Form.Label>Password</Form.Label>
        <Form.Control
          type='password'
          placeholder='Enter the password'
          value={password}
          onChange={(e) => setPass(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Form.Group controlId='confirmpassword'>
        <Form.Label>Confirm Password</Form.Label>
        <Form.Control
          type='password'
          placeholder='Confirm your password'
          value={confirmpassword}
          onChange={(e) => setConPass(e.target.value)}
        ></Form.Control>
      </Form.Group>

      <Button type='submit' variant='primary' >
        Update Account
      </Button>
    </Form>
  </Col>
  <Col md={9}>
    <h2> Order Table</h2>
```

```
    </Col>
  </Row>

}

export default UserProfileView
```

CartReducer.js

```
import {
  CART_ADD_BOOK,
  CART_REMOVE_BOOK,
  CART_RECORD_SHIPPING_ADDRESS,
  CART_RECORD_PAYMENT_METHOD,
} from '../constants/cartConstants'

export const cartReducer = (
  state = { cartItems: [], shippingDetails: {} },
  action
) => {
  switch (action.type) {
    case CART_ADD_BOOK:
      const item = action.payload

      const existItem = state.cartItems.find((x) => x.product === item.product)

      if (existItem) {
        return {
          ...state,
          cartItems: state.cartItems.map((x) =>
            x.product === existItem.product ? item : x
          ),
        }
      } else {
        return {
          ...state,
          cartItems: [...state.cartItems, item],
        }
      }

    case CART_REMOVE_BOOK:
      return {
        ...state,
```

```
        cartItems: state.cartItems.filter((x) => x.product !== action.payload),
      }

    case CART_RECORD_SHIPPING_ADDRESS:
      return {
        ...state,
        shippingDetails: action.payload,
      }

    case CART_RECORD_PAYMENT_METHOD:
       return {
          ...state,
          paymentDetails: action.payload,
        }


    default:
      return state
  }
}
```

OrderReducer.js

```
import {
  ORDER_ADDED_REQUEST,
  ORDER_ADDED_SUCCESS,
  ORDER_ADDED_FAIL,
  ORDER_INFO_REQUEST,
  ORDER_INFO_SUCCESS,
  ORDER_INFO_FAIL,
  ORDER_PAYMENT_FAIL,
  ORDER_PAYMENT_SUCCESS,
  ORDER_PAYMENT_REQUEST,
  ORDER_PAYMENT_CLEAN,
} from '../constants/orderConstants'

export const orderADDReducer = (state = {}, action) => {
  switch (action.type) {
    case ORDER_ADDED_REQUEST:
      return {
        loading: true,
      }
    case ORDER_ADDED_SUCCESS:
```

```
      return {
        loading: false,
        success: true,
        order: action.payload,
      }
    case ORDER_ADDED_FAIL:
      return {
        loading: false,
        error: action.payload,
      }
    default:
      return state
  }
}

export const orderInfoReducer = (
  state = { loading: true, orderItems: [], shippingAddress: {} },
  action
) => {
  switch (action.type) {
    case ORDER_INFO_REQUEST:
      return {
        ...state,
        loading: true,
      }
    case ORDER_INFO_SUCCESS:
      return {
        loading: false,
        order: action.payload,
      }
    case ORDER_INFO_FAIL:
      return {
        loading: false,
        error: action.payload,
      }
    default:
      return state
  }
}

export const orderPaymentReducer = (state = {}, action) => {
  switch (action.type) {
    case ORDER_PAYMENT_REQUEST:
      return {
        loading: true,
```

```
      }
    case ORDER_PAYMENT_SUCCESS:
      return {
        loading: false,
        success: true,
      }
    case ORDER_PAYMENT_FAIL:
      return {
        loading: false,
        error: action.payload,
      }
      case ORDER_PAYMENT_CLEAN:
        return {}

    default:
      return state
  }
}
```

ProductReducer.js

```
import {
  PRODUCT_LIST_REQUEST,
  PRODUCT_LIST_SUCCESS,
  PRODUCT_LIST_FAIL,
  PRODUCT_INFO_REQUEST,
  PRODUCT_INFO_SUCCESS,
  PRODUCT_INFO_FAIL,
} from '../constants/productConstants'

export const productListReducer = (state = { products: [] }, action) => {
  switch (action.type) {
    case PRODUCT_LIST_REQUEST:
      return { loading: true, products: [] }
    case PRODUCT_LIST_SUCCESS:
      return { loading: false, products: action.payload }
    case PRODUCT_LIST_FAIL:
      return { loading: false, error: action.payload }
    default:
      return state
  }
}
```

```
export const productInfoReducer = (
  state = { product: { reviews: [] } },
  action
) => {
  switch (action.type) {
    case PRODUCT_INFO_REQUEST:
      return { loading: true, ...state }
    case PRODUCT_INFO_SUCCESS:
      return { loading: false, product: action.payload }
    case PRODUCT_INFO_FAIL:
     return { loading: false, error: action.payload }
    default:
      return state
  }
}
```

UserReducer.js

```
import {
  USER_SIGIN_REQUEST,
  USER_SIGIN_SUCCESS,
  USER_SIGIN_FAIL,
  USER_SIGNOUT,
  USER_CREATE_REQUEST,
  USER_CREATE_SUCCESS,
  USER_CREATE_FAIL,
  USER_INFO_REQUEST,
  USER_INFO_SUCCESS,
  USER_INFO_FAIL,
  USER_UPDATE_ACC_REQUEST,
  USER_UPDATE_ACC_SUCCESS,
  USER_UPDATE_ACC_FAIL,
  USER_UPDATE_ACC_CLEAN,
  USER_LIST_REQUEST,
  USER_LIST_SUCCESS,
  USER_LIST_FAIL,
} from '../constants/userConstants'

export const userSignInReducer = (state = {}, action) => {
  switch (action.type) {
    case USER_SIGIN_REQUEST:
      return { loading: true }
```

```
    case USER_SIGIN_SUCCESS:
      return { loading: false, userDetails: action.payload }
    case USER_SIGIN_FAIL:
      return { loading: false, error: action.payload }
    case USER_SIGNOUT:
      return {}
    default:
      return state
  }
}

export const userCreateReducer = (state = {}, action) => {
  switch (action.type) {
    case USER_CREATE_REQUEST:
      return { loading: true }
    case USER_CREATE_SUCCESS:
      return { loading: false, userDetails: action.payload }
    case USER_CREATE_FAIL:
      return { loading: false, error: action.payload }
    default:
      return state
  }
}

export const userInfoReducer = (state = { user: {} }, action) => {
  switch (action.type) {
    case USER_INFO_REQUEST:
      return { ...state, loading: true }
    case USER_INFO_SUCCESS:
      return { loading: false, user: action.payload }
    case USER_INFO_FAIL:
      return { loading: false, error: action.payload }
    default:
      return state
  }
}

export const userUpdateReducer = (state = { }, action) => {
  switch (action.type) {
    case USER_UPDATE_ACC_REQUEST:
      return {loading: true }
    case USER_UPDATE_ACC_SUCCESS:
      return { loading: false, success: true, userDetails: action.payload }
    case USER_UPDATE_ACC_FAIL:
      return { loading: false, error: action.payload }
```

```
    case USER_UPDATE_ACC_CLEAN:
      return {}
    default:
      return state
  }
}

export const userListReducer = (state = {users: [] }, action) => {
  switch (action.type) {
    case USER_LIST_REQUEST:
      return {loading: true }
    case USER_LIST_SUCCESS:
      return { loading: false, users: action.payload }
    case USER_LIST_FAIL:
      return { loading: false, error: action.payload }

    default:
      return state
  }
}
```

cartConstants.js

```
export const CART_ADD_BOOK = 'CART_ADD_BOOK'
export const CART_REMOVE_BOOK = 'CART_REMOVE_BOOK'
export const CART_RECORD_SHIPPING_ADDRESS = 'CART_RECORD_SHIPPING_ADDRESS'
export const CART_RECORD_PAYMENT_METHOD = 'CART_RECORD_PAYMENT_METHOD'
```

OrderConstants.js

```
export const ORDER_ADDED_REQUEST = 'ORDER_ADDED_REQUEST'
export const ORDER_ADDED_SUCCESS = 'ORDER_ADDED_SUCESS'
export const ORDER_ADDED_FAIL = 'ORDER_ADDED_FAIL'

export const ORDER_INFO_REQUEST = 'ORDER_INFO_REQUEST'
export const ORDER_INFO_SUCCESS = 'ORDER_INFO_SUCESS'
export const ORDER_INFO_FAIL = 'ORDER_INFO_FAIL'

export const ORDER_PAYMENT_REQUEST = 'ORDER_PAYMENT_REQUEST'
export const ORDER_PAYMENT_SUCCESS = 'ORDER_PAYMENT_SUCESS'
export const ORDER_PAYMENT_FAIL = 'ORDER_PAYMENT_FAIL'
export const ORDER_PAYMENT_CLEAN = 'ORDER_PAYMENT_CLEAN'
```

ProductConsts.js

```
export const PRODUCT_LIST_REQUEST = 'PRODUCT_LIST_REQUEST'
export const PRODUCT_LIST_SUCCESS = 'PRODUCT_LIST_SUCCESS'
export const PRODUCT_LIST_FAIL = 'PRODUCT_LIST_FAIL'

export const PRODUCT_INFO_REQUEST = 'PRODUCT_INFO_REQUEST'
export const PRODUCT_INFO_SUCCESS = 'PRODUCT_INFO_SUCCESS'
export const PRODUCT_INFO_FAIL = 'PRODUCT_INFO_FAIL'
```

UserConsts.js

```
export const USER_SIGIN_REQUEST = 'USER_SIGNIN_REQUEST'
export const USER_SIGIN_SUCCESS = 'USER_SIGNIN_SUCCESS'
export const USER_SIGIN_FAIL = 'USER_SIGNIN_FAIL'
export const USER_SIGNOUT = 'USER_SIGNOUT'

export const USER_CREATE_REQUEST = 'USER_CREATE_REQUEST'
export const USER_CREATE_SUCCESS = 'USER_CREATE_SUCCESS'
export const USER_CREATE_FAIL = 'USER_CREATE_FAIL'

export const USER_INFO_REQUEST = 'USER_INFO_REQUEST'
export const USER_INFO_SUCCESS = 'USER_INFO_SUCCESS'
export const USER_INFO_FAIL = 'USER_INFO_FAIL'

export const USER_UPDATE_ACC_REQUEST = 'USER_UPDATE_ACC_REQUEST'
export const USER_UPDATE_ACC_SUCCESS = 'USER_UPDATE_ACC_SUCCESS'
export const USER_UPDATE_ACC_FAIL = 'USER_UPDATE_ACC_FAIL'
export const USER_UPDATE_ACC_CLEAN = 'USER_UPDATE_ACC_CLEAN'

export const USER_LIST_REQUEST = 'USER_LIST_REQUEST'
export const USER_LIST_SUCCESS = 'USER_LIST_SUCCESS'
export const USER_LIST_FAIL = 'USER_LIST_FAIL'
```

CartActions.js

```
import axios from 'axios'
import { CART_ADD_BOOK, CART_REMOVE_BOOK, CART_RECORD_SHIPPING_ADDRESS, CART_RECORD_PAYM
ENT_METHOD } from '../constants/cartConstants'
```

```
export const addToCart = (id, oqty) => async (dispatch, getState) => {
  const { data } = await axios.get(`/api/products/${id}`)

  dispatch({
    type: CART_ADD_BOOK,
    payload: {
      product: data._id,
      name: data.b_name,
      image: data.b_image,
      price: data.price,
      countInStock: data.countInStock,
      oqty,
    },
  })

  localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems))
}

export const removeFromCart = (id) =>  (dispatch, getState) => {
  dispatch({
    type: CART_REMOVE_BOOK,
    payload: id,
  })

  localStorage.setItem('cartItems', JSON.stringify(getState().cart.cartItems))
}

export const recordShippinginfo = (data) =>  (dispatch) => {
  dispatch({
    type: CART_RECORD_SHIPPING_ADDRESS,
    payload: data,
  })

  localStorage.setItem('shippingDetails', JSON.stringify(data))
}

export const recordPaymentmethod = (data) =>  (dispatch) => {
  dispatch({
    type: CART_RECORD_PAYMENT_METHOD,
    payload: data,
  })

  localStorage.setItem('paymentDetails', JSON.stringify(data))
}
```

OrderActions.js

```
import {ORDER_ADDED_REQUEST, ORDER_ADDED_SUCCESS, ORDER_ADDED_FAIL,ORDER_INFO_REQUEST,OR
DER_INFO_SUCCESS,ORDER_INFO_FAIL, ORDER_PAYMENT_SUCCESS, ORDER_PAYMENT_REQUEST, ORDER_PA
YMENT_FAIL} from '../constants/orderConstants'
import axios from 'axios'



export const AddOrder = (order) => async (dispatch, getState) => {
    try {
      dispatch({
        type: ORDER_ADDED_REQUEST,
      })

      const {
        userSignin: { userDetails },
      } = getState()

      const config = {
        headers: {
          'Content-Type': 'application/json',
          Authorization: `Bearer ${userDetails.token}`,
        },
      }

      const { data } = await axios.post(`/api/orders`, order, config)

      dispatch({
        type: ORDER_ADDED_SUCCESS,
        payload: data,
      })

    } catch (error) {
      dispatch({
        type: ORDER_ADDED_FAIL,
        payload:
          error.response && error.response.data.message
            ? error.response.data.message
            : error.message,
      })
    }
  }
```

```
export const getOrderInfo = (id) => async (dispatch, getState) => {
  try {
    dispatch({
      type: ORDER_INFO_REQUEST,
    })

    const {
      userSignin: { userDetails },
    } = getState()

    const config = {
      headers: {

        Authorization: `Bearer ${userDetails.token}`,
      },
    }

    const { data } = await axios.get(`/api/orders/${id}`,config)

    dispatch({
      type: ORDER_INFO_SUCCESS,
      payload: data,
    })

  } catch (error) {

    console.log("aa"+id);

    dispatch({
      type: ORDER_INFO_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}


export const OrderPayment = (orderID, paymentResult) => async (dispatch, getState) =>
{
  try {
    dispatch({
      type: ORDER_PAYMENT_REQUEST,
    })
```

```
      const {
        userSignin: { userDetails },
      } = getState()

      const config = {
        headers: {
          'Content-Type': 'application/json',
          Authorization: `Bearer ${userDetails.token}`,
        },
      }

      const { data } = await axios.put(`/api/orders/${orderID}/pay`,paymentResult,config
)

      dispatch({
        type: ORDER_PAYMENT_SUCCESS,
        payload: data,
      })

    } catch (error) {
      dispatch({
        type: ORDER_PAYMENT_FAIL,
        payload:
          error.response && error.response.data.message
            ? error.response.data.message
            : error.message,
      })
    }
  }
```

ProductActions.js

```
import axios from 'axios'
import {
  PRODUCT_LIST_REQUEST,
  PRODUCT_LIST_SUCCESS,
  PRODUCT_LIST_FAIL,
  PRODUCT_INFO_REQUEST,
  PRODUCT_INFO_SUCCESS,
  PRODUCT_INFO_FAIL,
  PRODUCT_DELETE_SUCCESS,
```

```
  PRODUCT_DELETE_FAIL,
  PRODUCT_DELETE_REQUEST,
} from '../constants/productConstants'

export const listProducts = () => async (dispatch) => {
  try {
    dispatch({ type: PRODUCT_LIST_REQUEST })

    const { data } = await axios.get('/api/products')

    dispatch({
      type: PRODUCT_LIST_SUCCESS,
      payload: data,
    })
  } catch (error) {
    dispatch({
      type: PRODUCT_LIST_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}

export const listProductInfo = (id) => async (dispatch) => {
  try {
    dispatch({ type: PRODUCT_INFO_REQUEST })

    const { data } = await axios.get(`/api/products/${id}`)

    dispatch({
      type: PRODUCT_INFO_SUCCESS,
      payload: data,
    })
  } catch (error) {
    dispatch({
      type: PRODUCT_INFO_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}
```

```
export const delteProduct = (id) => async (dispatch, getState) => {
  try {
    dispatch({
      type: PRODUCT_DELETE_REQUEST,
    })

    const {
      userSignin: { userDetails },
    } = getState()

    const config = {
      headers: {

        Authorization: `Bearer ${userDetails.token}`,
      },
    }

    await axios.delete(`/api/products/${id}`,config)


    dispatch({
      type: PRODUCT_DELETE_SUCCESS,

    })

  } catch (error) {
    dispatch({
      type: PRODUCT_DELETE_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}
```

UserActions.js

```
import axios from 'axios'
import {
```

```
  USER_SIGIN_FAIL,
  USER_SIGIN_REQUEST,
  USER_SIGIN_SUCCESS,
  USER_SIGNOUT,
  USER_CREATE_REQUEST,
  USER_CREATE_SUCCESS,
  USER_CREATE_FAIL,
  USER_INFO_SUCCESS,
  USER_INFO_REQUEST,
  USER_INFO_FAIL,
  USER_UPDATE_ACC_REQUEST,
  USER_UPDATE_ACC_SUCCESS,
  USER_UPDATE_ACC_FAIL,
  USER_LIST_REQUEST,
  USER_LIST_SUCCESS,
  USER_LIST_FAIL,
  USER_LIST_CLEAN

} from '../constants/userConstants'

export const Sign_In = (email, password) => async (dispatch) => {
  try {
    dispatch({
      type: USER_SIGIN_REQUEST,
    })

    const config = {
      headers: {
        'Content-Type': 'application/json',
      },
    }

    const { data } = await axios.post(
      '/api/users/login',
      { email, password },
      config
    )

    dispatch({
      type: USER_SIGIN_SUCCESS,
      payload: data,
    })

    localStorage.setItem('userDetails', JSON.stringify(data))
  } catch (error) {
```

```
    dispatch({
      type: USER_SIGIN_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}

export const signout = () => (dispatch) => {
  localStorage.removeItem('userDetails')
  dispatch({ type: USER_SIGNOUT })
  dispatch({ type: USER_LIST_CLEAN })

}

export const AddUser = (name, email, password) => async (dispatch) => {
  try {
    dispatch({
      type: USER_CREATE_REQUEST,
    })

    const config = {
      headers: {
        'Content-Type': 'application/json',
      },
    }

    const { data } = await axios.post(
      '/api/users',
      { name, email, password },
      config
    )

    dispatch({
      type: USER_CREATE_SUCCESS,
      payload: data,
    })

    dispatch({
      type: USER_SIGIN_SUCCESS,
      payload: data,
    })
```

```
    localStorage.setItem('userDetails', JSON.stringify(data))
  } catch (error) {
    dispatch({
      type: USER_CREATE_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}

export const ViewuserInfo = (id) => async (dispatch, getState) => {
  try {
    dispatch({
      type: USER_INFO_REQUEST,
    })

    const {
      userSignin: { userDetails },
    } = getState()

    const config = {
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${userDetails.token}`,
      },
    }

    const { data } = await axios.get(`/api/users/${id}`, config)

    dispatch({
      type: USER_INFO_SUCCESS,
      payload: data,
    })
  } catch (error) {
    dispatch({
      type: USER_INFO_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}
```

```
export const updateUserAccount = (user) => async (dispatch, getState) => {
  try {
    dispatch({
      type: USER_UPDATE_ACC_REQUEST,
    })

    const {
      userSignin: { userDetails },
    } = getState()

    const config = {
      headers: {
        'Content-Type': 'application/json',
        Authorization: `Bearer ${userDetails.token}`,
      },
    }

    const { data } = await axios.put(`/api/users/account`, user, config)

    dispatch({
      type: USER_UPDATE_ACC_SUCCESS,
      payload: data,
    })

    dispatch({
      type: USER_SIGIN_SUCCESS,
      payload: data,
    })

    localStorage.setItem('userDetails', JSON.stringify(data))

  } catch (error) {
    dispatch({
      type: USER_UPDATE_ACC_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}

export const ListUsers = () => async (dispatch, getState) => {
  try {
```

```
    dispatch({
      type: USER_LIST_REQUEST,
    })

    const {
      userSignin: { userDetails },
    } = getState()

    const config = {
      headers: {

        Authorization: `Bearer ${userDetails.token}`,
      },
    }

    const { data } = await axios.get(`/api/users`, config)

    dispatch({
      type: USER_LIST_SUCCESS,
      payload: data,
    })

  } catch (error) {
    dispatch({
      type: USER_LIST_FAIL,
      payload:
        error.response && error.response.data.message
          ? error.response.data.message
          : error.message,
    })
  }
}
```

Store.js

```
import { createStore, combineReducers, applyMiddleware } from 'redux'
import thunk from 'redux-thunk'
import { composeWithDevTools } from 'redux-devtools-extension'
```

```
import {
  productListReducer,
  productInfoReducer,
  productDeleteReducer,
} from './reducers/productReducers'
import { cartReducer } from './reducers/cartReducers'
import { userSignInReducer, userCreateReducer, userInfoReducer, userUpdateReducer, userL
istReducer } from './reducers/userReducers'
import {orderADDReducer, orderInfoReducer, orderPaymentReducer} from './reducers/orderRe
ducers'

const reducer = combineReducers({
  productList: productListReducer,
  productInfo: productInfoReducer,
  cart: cartReducer,
  userSignin: userSignInReducer,
  userCreate: userCreateReducer,
  userInfo: userInfoReducer,
  userUpdateAcc: userUpdateReducer,
  orderAdd: orderADDReducer,
  orderinfo: orderInfoReducer,
  orderPayment:orderPaymentReducer,
  userListb: userListReducer,
  productDelete: productDeleteReducer,
})

const cartAddeditemsfromLocalStrg = localStorage.getItem('cartItems')
  ? JSON.parse(localStorage.getItem('cartItems'))
  : []

  const userInfoPassedfromLocalStrg = localStorage.getItem('userDetails')
  ? JSON.parse(localStorage.getItem('userDetails'))
  : null

  const shippingAdressfromLocalStrg = localStorage.getItem('shippingDetails')
  ? JSON.parse(localStorage.getItem('shippingDetails'))
  : {}

const initialState = {
  cart: { cartItems: cartAddeditemsfromLocalStrg, shippingDetails: shippingAdressfromLoc
alStrg },
  userSignin: {userDetails: userInfoPassedfromLocalStrg},

}
const middleware = [thunk]
```

```
const store = createStore(
  reducer,
  initialState,
  composeWithDevTools(applyMiddleware(...middleware))
)

export default store
```

Common – Components

Alert.js

```
import React from 'react'
import { Alert } from 'react-bootstrap'

const Alertmsg = ({ variant, children }) => {
  return <Alert variant={variant}>{children}</Alert>
}
Alertmsg.defaultProps = {
  variant: 'info',
}

export default Alertmsg
```

Buffer.js

```
import React from 'react'
import {Spinner} from 'react-bootstrap'

const buffer = () => {
    return (
        <Spinner
            animation='border'
```

```
            role= 'status'
            style = {{
                width: '150px',
                height: '150px',
                margin: 'auto',
                display: 'block',
            }}> <span className='sr-only'>Loading..</span>
        </Spinner>
    )
}

export default buffer
```

CheckoutNavigator.js

```
import React from 'react'
import { Nav } from 'react-bootstrap'
import { LinkContainer } from 'react-router-bootstrap'

const Checkoutnavigator = ({ s1, s2, s3, s4 }) => {
  return (

    <Nav className='justify-content-center mb-4'>
      <Nav.Item>
        {s1 ? (
          <LinkContainer to='/login'>
            <Nav.Link>SignIn</Nav.Link>
          </LinkContainer>
        ) : (
          <Nav.Link disabled>SignIn</Nav.Link>
        )}
      </Nav.Item>

      <Nav.Item>
        {s2 ? (
          <LinkContainer to='/shippingprocess'>
            <Nav.Link>Shipping</Nav.Link>
          </LinkContainer>
        ) : (
          <Nav.Link disabled>Shipping</Nav.Link>
        )}
```

```
      </Nav.Item>

      <Nav.Item>
        {s3 ? (
          <LinkContainer to='/paymentprocess'>
            <Nav.Link>Payment</Nav.Link>
          </LinkContainer>
        ) : (
          <Nav.Link disabled>Payment</Nav.Link>
        )}
      </Nav.Item>

      <Nav.Item>
        {s4 ? (
          <LinkContainer to='/confirmOrder'>
            <Nav.Link>Confirm Order</Nav.Link>
          </LinkContainer>
        ) : (
          <Nav.Link disabled>Confirm Order</Nav.Link>
        )}
      </Nav.Item>


    </Nav>
  )
}

export default Checkoutnavigator
```

Footer.js

```
import React from 'react'
import {Container, Row , Col} from 'react-bootstrap'

const Footer = () => {
    return (
        <footer>
            <Container>
                <Row>
                    <Col className='text-center py-3'>
                        Copyright &copy; BookMart
                    </Col>
                </Row>
            </Container>
```

```
        </footer>
    )
}

export default Footer
```

FormLayout.js

```
import React from 'react'
import {Container,Row, Col} from 'react-bootstrap'

const Form_layout = ({children}) => {
    return (
        <Container>
            <Row className='justify-content-md-center'>
                <Col xs={12} md={6}>
                    {children}
                </Col>

            </Row>

        </Container>
    )
}

export default Form_layout
```

Header.js

```
/*ES7 snippets here we will be using rafce(react arrow function export) the difference o
f this is it createsa a variable and export itdown to the
header*/

import React from 'react'
import { LinkContainer } from 'react-router-bootstrap'
```

```
import { useDispatch, useSelector } from 'react-redux'
import { Navbar, Nav, Container, NavDropdown } from 'react-bootstrap'
import {signout} from '../actions/userActions'

const Header = () => {

  const dispatch = useDispatch()

  const userSignin = useSelector((state) => state.userSignin)
  const { userDetails } = userSignin

  const signouthandler = () => {
      dispatch(signout())
  }

  return (
    <header>
      <Navbar bg='light' variant='blue' expand='lg' collapseOnSelect>
        <Container>
          {/* Instead of normal Link we have used LinkContainer*/}
          <LinkContainer to='/'>
            <Navbar.Brand>BookMart</Navbar.Brand>
          </LinkContainer>

          <Navbar.Toggle aria-controls='basic-navbar-nav' />
          <Navbar.Collapse id='basic-navbar-nav'>
            <Nav className='ms-auto'>
              <LinkContainer to='/cart'>
                <Nav.Link>
                  <i className='fas fa-shopping-cart'>CART</i>
                </Nav.Link>
              </LinkContainer>
              {userDetails ? (
                  <NavDropdown title={userDetails.name} id='accountname'><LinkContainer
to='/account'><NavDropdown.Item>Account</NavDropdown.Item></LinkContainer>
                  <NavDropdown.Item onClick={signouthandler}>Sign Out</NavDropdown.Item>
</NavDropdown>


              ) :<LinkContainer to='/login'>
                <Nav.Link>
                  <i className='fas fa-user'> SIGN IN</i>
                </Nav.Link>
              </LinkContainer>}
```

```
                {userDetails && userDetails.isAdmin && (
              <NavDropdown title='Admin' id='adminpanel'>

                  <LinkContainer to ='admin/userlist'>
                <NavDropdown.Item >Users</NavDropdown.Item>
                  </LinkContainer>

                  <LinkContainer to ='admin/productlist'>
                <NavDropdown.Item >product</NavDropdown.Item>
                  </LinkContainer>

                  <LinkContainer to ='admin/orderlist'>
                <NavDropdown.Item >Orders</NavDropdown.Item>
                  </LinkContainer>

              </NavDropdown>
                )}
            </Nav>
          </Navbar.Collapse>
        </Container>
      </Navbar>
    </header>
  )
}

export default Header
```

Product.js

```
import React from 'react'


import { Link } from 'react-router-dom'

import { Card } from 'react-bootstrap'
import Rating from '../components/Rating'

const Product = (props) => {
  return (
    <Card className='my-3 py-3 rounded'>
      <Link to={`/product/${props.product._id}`}>
        <Card.Img src={props.product.b_image} variant='top' fluid='true' />
```

```
      </Link>

      <Card.Body>
        <Link to={`/product/${props.product._id}`}>
          <Card.Title as='div'>
            <strong>{props.product.b_name}</strong>
          </Card.Title>
        </Link>

        <Card.Text as='div'></Card.Text>
        <Card.Text as='h5'>LKR {props.product.price}</Card.Text>
        <Rating
          value={props.product.rating}
          text={`${props.product.numReviews} reviews`}
        />
      </Card.Body>
    </Card>
  )
}

export default Product
```

Backend

Server.js

```
import express from 'express'
import dotenv from 'dotenv'
import colors from 'colors'
import connectDB from './config/db.js'
import { notFound, errorHandler } from './middleware/errorMiddleware.js'

import productRoutes from './routes/productRoutes.js'
import userRoutes from './routes/userRoutes.js'
```

```
import orderRoutes from './routes/orderRoutes.js'

dotenv.config()

connectDB()

const app = express()

//a middleware to accpet json request body through the server
app.use(express.json())

app.get('/', (req, res) => {
  res.send('The API is working')
})

//Routing to relavent Roters
app.use('/api/products', productRoutes)
app.use('/api/users', userRoutes)
app.use('/api/orders', orderRoutes)

app.get('api/config/paypal', (req, res) => res.send(process.env.PAYPAL_SANDBOX_CLIENT_ID
))

app.use(notFound)
app.use(errorHandler)

//calling the dotenv file
const PORT = process.env.PORT || 5000

app.listen(
  PORT,
  console.log(
    `Server running in ${process.env.NODE_ENV} mode on port ${PORT}`.yellow
  )
)
```

Db.js

```
import mongoose from 'mongoose'

const connectDB = async () => {
  try {
    const conn = await mongoose.connect(process.env.DB_CONET_STRNG, {
      useUnifiedTopology: true,
      useNewUrlParser: true,
      useCreateIndex: true,
    })
    console.log(`MongoDB Connected: ${conn.connection.host}`.cyan)
  } catch (error) {
    console.log(`Error: ${error.message}`.red.bold)
    process.exit(1)
  }
}

export default connectDB
```

OrderController.js

```
import asyncHandler from 'express-async-
handler' //importing Async handler to do error handling in routes rather than using TryC
atch
import Order from '../models/orderModel.js'

// @desc  insert new order
// @route POST /api/orders
// @access Private
const createOrder = asyncHandler(async (req, res) => {
  const {
    orderItems,
    shippingPrice,
    paymentMethod,
    itemsPrice,
    shippingAddress,
    totalPrice,
  } = req.body

  // console.log(req.body);

  if (orderItems && orderItems.length === 0) {
    res.status(400)
```

```
    throw new Error('Empty Order list')
    return
  } else {
    const order = new Order({
      user: req.user._id,
      orderItems,
      shippingPrice,
      paymentMethod,
      itemsPrice,
      shippingAddress,
      totalPrice,
    })

    const finalizeOrder = await order.save()
    res.status(201).json(finalizeOrder)
  }
})

// @desc  fetch order  by a id
// @route GET /api/orders/:id
// @access Private
const getOrderByID = asyncHandler(async (req, res) => {
 const order = await Order.findById(req.params.id).populate('user', 'name email')

 if(order) {
   res.json(order)
 }else{
   res.status(404)
   throw new Error('Invalid Order')
 }

})

// @desc  Update order to paid
// @route GET /api/orders/:id/pay
// @access Private

const updateOrdertoPaid = asyncHandler(async (req, res) => {
  const order = await Order.findById(req.params.id)

  if(order) {
    order.isPaid =true
    order.paidAt = Date.now()
    order.paymentResult = {
      id: req.body.id,
```

```
      status: req.body.status,
      update_time: req.body.update_time,
      email_address: req.body.payer.email_address
    }

    const updatedOrder = await order.save()
    res.json(updatedOrder)

  }else{
    res.status(404)
    throw new Error('Invalid Order')
  }

 })


export { createOrder, getOrderByID, updateOrdertoPaid }
```

UserController.js

```
import genToken from '../utilities/tokenGenerate.js'
import asyncHandler from 'express-async-
handler' //importing Async handler to do error handling in routes rather than using TryC
atch
import User from '../models/userModel.js'

// @desc  Fetch validate the user credentials and then send a token
// @route POST /api/users/login
// @access Public

const authUser = asyncHandler(async (req, res) => {
  const { email, password } = req.body

  const user = await User.findOne({ email })

  if (user && (await user.verifyCredentials(password))) {
    res.json({
      _id: user._id,
      name: user.name,
      email: user.email,
      isAdmin: user.isAdmin,
```

```
      token: genToken(user._id),
    })
  } else {
    res.status(401)
    throw new Error('Entered Credentials are not correct')
  }
})

// @desc  Get the user profile
// @route GET /api/users/userAccount
// @access Private

const getUserAccount = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id)

  if (user) {
    res.json({
      _id: user._id,
      name: user.name,
      email: user.email,
      isAdmin: user.isAdmin,
    })
  } else {
    res.status(404)
    throw new Error('User cannot be found')
  }
})

// @desc  Add a new user
// @route POST /api/users/
// @access Public

const addUser = asyncHandler(async (req, res) => {
  const { name, email, password } = req.body

  const chk_user_existence = await User.findOne({ email: email })

  if (chk_user_existence) {
    res.status(400)
    throw new Error(`There's a member already registered with that mail`)
  }

  const user = await User.create({
    name,
    email,
```

```
    password,
  })
  if (user) {
    res.status(201).json({
      _id: user._id,
      name: user.name,
      email: user.email,
      isAdmin: user.isAdmin,
      token: genToken(user._id),
    })
  } else {
    res.status(400)
    throw new Error('This user account cannot be created. Try again')
  }
})

// @desc  To Update user profile
// @route PUT /api/users/userAccount
// @access Private

const updateUserAccount = asyncHandler(async (req, res) => {
  const user = await User.findById(req.user._id)

  if (user) {
    user.name = req.body.name || user.name
    user.email = req.body.email || user.email
    if (req.body.password) {
      user.password = req.body.password
    }

    const updatedUser = await user.save()

    res.json({
      _id: updatedUser._id,
      name: updatedUser.name,
      email: updatedUser.email,
      isAdmin: updatedUser.isAdmin,
      token: genToken(updatedUser._id),
    })


  } else {
    res.status(404)
    throw new Error('User cannot be found')
  }
```

```
})



// @desc  Get request to all users
// @route PUT /api/users
// @access Private/ admin

const getUsers = asyncHandler(async (req, res) => {
  const users = await User.find({})
  res.json(users)


})

export { authUser, getUserAccount, addUser, updateUserAccount, getUsers }
```

ErrorMiddlerware.js

```
const notFound = (req, res, next)=> {
    const error = new Error(`Not Found - ${req.originalUrl}`)
    res.status(404)
    next(error)
  }


const errorHandler = (err,req, res, next)=> {
    const statusCode = res.statusCode === 200 ? 500: res.statusCode
    res.status(statusCode)
    res.json({
      message: err.message,
      stack: process.env.NODE_ENV === `production` ? null: err.stack,
    })
  }

  export {notFound, errorHandler}
```

ValidateTokenMiddleware.js

```
import asyncHandler from 'express-async-handler'
import jwt from 'jsonwebtoken'
import User from '../models/userModel.js'

const shield = asyncHandler(async(req,res, next) => {

    let receivedToken

    if(req.headers.authorization && req.headers.authorization.startsWith('Bearer'))
    {
       try{
         receivedToken = req.headers.authorization.split(' ')[1]
         const decodedToken = jwt.verify(receivedToken, process.env.JWT_SCRT_KEY)

         req.user = await User.findById(decodedToken.id).select('-password')


         next()
       }
       catch(error){

        console.error(error)
        res.status(401)
        throw new Error('token is broken , cannot be authroised')

       }
    }

    if(!receivedToken)
    {
        res.status(404)
        throw new Error('a token is not availabe, cannot be authorized')
    }



})

const admin = (req,res,next) => {
  if(req.user && req.user.isAdmin)
  {
    next()
  }else{
    res.status(401)
    throw new Error('Not an admin credential')
```

```
  }
}

export { shield, admin }
```

OrderModel.js

```
import mongoose from 'mongoose'

const orderSchema = mongoose.Schema(
  {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      required: true,
      ref: 'User',
    },

    orderItems: [
      {
        name: { type: String, required: true },
        oqty: { type: Number, required: true },
        image: { type: String, required: true },
        price: { type: Number, required: true },
        product: {
          type: mongoose.Schema.Types.ObjectId,
          required: true,
          ref: 'Product',
        },
      },
    ],

    shippingAddress: {
      address: { type: String, required: true },
      city: { type: String, required: true },
      postalCode: { type: String, required: true },
      country: { type: String, required: true },
      contactno: {type:String, required: true},
    },

    paymentMethod: {
      type: String,
      required: true,
```

```
    },
    paymentResult: {
      id: { type: String },
      status: { type: String },
      update_time: { type: String },
      email_address: { type: String },
    },

    shippingPrice: {
      type: Number,
      required: true,
      default: 0.0,
    },
    itemsPrice: {
      type: Number,
      required: true,
      default: 0.0,
    },
    totalPrice: {
      type: Number,
      required: true,
      default: 0.0,
    },
    isPaid: {
      type: Boolean,
      required: true,
      default: false,
    },
    paidAt: {
      type: Date,
    },
    iseDelivered: {
      type: Boolean,
      required: true,
      default: false,
    },
    deliveredAt: {
      type: Date,
    },
  },
  {
    timestamps: true,
  }
)
```

```
const Order = mongoose.model('Order', orderSchema)

export default Order
```

ProductModel.js

```
import mongoose from 'mongoose'

const reviewSchema = mongoose.Schema({
    name: {type: String, required: true},
    rating: {type: Number, required: true},
    comment:{type: String, required: true}
},{
    timestamps: true
})

const productSchema = mongoose.Schema(
  {
    user: {
      type: mongoose.Schema.Types.ObjectId,
      required: true,
      ref: 'User',
    },

    b_name: {
      type: String,
      required: true,
    },
    b_image: {
      type: String,
      required: true,
    },
    b_author: {
      type: String,
      required: true,
    },
    category: {
      type: String,
      required: true,
    },
    b_description: {
```

```
      type: String,
      required: true,
    },

    reviews: [reviewSchema],

    rating: {
      type: Number,
      required: true,
      default: 0,
    },
    numReviews: {
      type: Number,
      required: true,
      default: 0,
    },
    price: {
      type: Number,
      required: true,
      default: 0,
    },
    countInStock: {
      type: Number,
      required: true,
      default: 0,
    },
  },
  {
    timestamps: true,
  }
)

const Product = mongoose.model('Product', productSchema)

export default Product
```

UserModel.js

```
import mongoose from 'mongoose'
import bcrypt from 'bcryptjs'

const userSchema = mongoose.Schema({
  name: {
    type: String,
    required: true,
```

```
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  isAdmin: {
    type: Boolean,
    required: true,
    default: false,
  },
}, {
    timestamps: true
})

userSchema.methods.verifyCredentials = async function(inputPwd){
  return await bcrypt.compare(inputPwd, this.password)
}

//middleware function to encrypt the password before inserting to the database
userSchema.pre('save', async function (next){

  if(!this.isModified('password')){
    next()
  }

  const Salt = await bcrypt.genSalt(10)
  this.password = await bcrypt.hash(this.password, Salt)

})

const User = mongoose.model('User', userSchema)

export default User
```

OrderModel.js

```
import express from 'express'
import { createOrder, getOrderByID, updateOrdertoPaid } from '../controllers/orderContro
ller.js'
const router = express.Router()
import {shield} from '../middleware/validateTokenMiddleware.js'


router.route('/').post(shield, createOrder)
router.route('/:id').get(shield, getOrderByID)
router.route('/:id/pay').put(shield, updateOrdertoPaid)




export default router
```

ProductModel.js

```
import express from 'express'
import { getProducById, getProducts, deleteProducById } from '../controllers/productCont
roller.js'
import {shield, admin} from '../middleware/validateTokenMiddleware.js'

const router = express.Router()

router.route('/').get(getProducts)

router.route('/:id').get(getProducById).delete(shield, admin, deleteProducById).put(shie
ld, admin, )

export default router
```

UserRoutes.js

```
import express from 'express'
import { authUser, getUserAccount, addUser, updateUserAccount, getUsers } from '../contr
ollers/userController.js'
const router = express.Router()
import {shield, admin} from '../middleware/validateTokenMiddleware.js'

router.post('/login', authUser) // to login
```

```
router.route('/account').get(shield, getUserAccount).put(shield, updateUserAccount) // t
o authorise with jwt tokens  and view user profile or update user profile
//router.post('/', addUser)// to create a new user

router.route('/').post(addUser).get(shield,admin, getUsers)




export default router
```

TokenGenerate.js

```
import jwt from 'jsonwebtoken'

const genToken= (id) => {
    return jwt.sign({id}, process.env.JWT_SCRT_KEY, {
        expiresIn:'30d'
    })
}

export default genToken
```

Seeder.js

```
import mongoose from 'mongoose'
import dotenv from 'dotenv'
import colors from 'colors'
import users from './data/users.js'
import products from './data/products.js'
import User from './models/userModel.js'
import Product from './models/productModel.js'
import Order from './models/orderModel.js'
import connectDB from './config/db.js'

dotenv.config()

connectDB()
```

```
const importData = async () => {
  try {
    await Order.deleteMany()
    await Product.deleteMany()
    await User.deleteMany()

    const createdUsers = await User.insertMany(users)
    const adminUser = createdUsers[0]._id
    const sampleProducts = products.map((product) => {
      return { ...product, user: adminUser }
    })

    await Product.insertMany(sampleProducts)

    console.log('Data Imported'.green.inverse)
    process.exit()
  } catch (error) {
    console.error(`${error}`.red.inverse)
    process.exit(1)
  }
}

const destroyData = async () => {
    try {
      await Order.deleteMany()
      await Product.deleteMany()
      await User.deleteMany()


      console.log('Data Destroyed'.red)
      process.exit()
    } catch (error) {
      console.error(`${error}`.red.inverse)
      process.exit(1)
    }
  }

  if (process.argv[2] === '-d'){
      destroyData()
  }else{
      importData()
  }
```