HARDWARE STORE

**A PROJECT REPORT**

**Submitted By: MANVENDRA PRATAP SINGH**

**(University Roll No - 2000290140066 )**

**PRAKHAR VARSHNEY**

**(University Roll No - 2000290140088 )**

**Submitted in partial fulfillment of the Requirements for the Degree of**

MASTER OF COMPUTER APPLICATIONS

**Under the Supervision of**

**Neelam Rawat**

**Associate Professor**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206**

**(June 2022)**

# DECLARATION

I hereby declare that the work presented in this report entitled “**HARDWARE STORE**", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name : Manvendra Pratap Singh Name: Prakhar Varshney

Roll. No. : 2000290140066 Roll.No. 2000290140088

Branch: MCA Branch: MCA

(Candidate Signature) (Candidate Signature)

# CERTIFICATE

Certified that **Manvendra Pratap Singh University Roll No - 2000290140066, Prakhar Varshney University Roll No – 2000290140088** have carried out the project work having “**HARDWARE STORE”** for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:** 13 JAN 2022

**Manvendra Pratap Singh (2000290140066) Prakhar Varshney (2000290140088)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

**Neelam Rawat**

**Associate Professor**

**Department of Computer Applications KIET Group of Institutions, Ghaziabad**

**Signature of Internal Examiner Signature of External Examiner**

**Dr. Ajay Shrivastava**

**Head, Department of Computer Applications KIET Group of Institutions, Ghaziabad**

# ABSTRACT

Hardware Store is process of doing business through computer networks. A person sitting on his chair in front of a computer can access all the facilities of the Internet to buy or sell the products. Unlike traditional commerce that is carried out physically with effort of a person to go & get products, ecommerce has made it easier for human to reduce physical work and to save time. E-Commerce which was started in early 1990’s has taken a great leap in the world of computers, but the fact that has hindered the growth of e-commerce is security. Security is the challenge facing ecommerce today & there is still a lot of advancement made in the field of security. The main advantage of e-commerce over traditional commerce is the user can browse online shops, compare prices and order merchandise sitting at home on their PC. For increasing the use of e-commerce in developing countries the B2B e-commerce is implemented for improving access to global markets for firms in developing countries. For a developing country advancement in the field of e­commerce is essential. The research strategy shows the importance of the e-commerce in developing countries for business applications. The main objective of the E-commerce Website is to manage the details of Products,Customer,Shipping,Payment,Category. It manages all the information about Products, Sales, Category, Products. The project is totally built at administrative end and thus only the administrator is guaranteed the access. The purpose of the project is to build an application program to reduce the manual work for managing the Products Customer, Sales, Shipping. It tracks all the details about the Shipping,Payment,Category

# ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Neelam Rawat mam** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Manvendra Pratap Singh Prakhar Varshney**

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| Declaration | 2 |
| Certificate | 3 |
| Abstract | 4 |
| Acknowledgements | 5 |
| Table of Content | 6-7 |
| List of Tables | 8 |
| List of Figures | 9 |
| Chapter 1 - Introduction   * 1. Project Introduction   2. Aim   3. Existing System   4. Proposed System   5. Hardware & Software Requirements   Chapter 2 Feasibility Study 13-14   * 1. Feasibility Study   2. Operational Feasibility   3. Technical Feasibility   4. Economical Feasibility   Chapter 3 Design & Planning SDLC Model   * 1. General Overview   2. Use Case Diagram   3. Sequence Diagram   4. Activity Diagram | 10-12  13-15  15-26 |

3.6 DFD

[Chapter 4 Form Design 27-28](#_TOC_250003)

4.1 Input / Output Form (Screenshot)

[Chapter 5 Coding 29-48](#_TOC_250002)

[Chapter 6 Testing 49-52](#_TOC_250001)

* 1. Test Case-1
  2. Test Case-2

[Chapter 7 Conclusion 53](#_TOC_250000)

Bibliography 54

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Name of Table** | **Page** |
| 2.1 | Hardware Specification | 12 |
| 2.2 | Software Specification | 12 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **Figure No.** | **Page No.** |
| 3.1.1 Waterfall model | 16 |
| 3.2 General Overview | 17 |
| 3.3 Use Case Diagram | 18 |
| 3.4.1 Login | 19 |
| 3.4.2 Search | 20 |
| 3.4.3 Add new product | 21 |
| 3.4.4 Buy product | 22 |
| 3.5.1 User Side | 23 |
| 3.5.2 For admin | 23 |
| 3.6.1 Zero level DFD | 24 |
| 3.6.2 First Level DFD | 25 |
| 3.6.3 Second Level DFD | 26 |
| 4.1.1 Login form | 27 |
| 4.1.2Input output form | 28 |

**CHAPTER 1**

**INTRODUCTION**

### INTRODUCTION

Customer get many benefits via online shopping this helps e-commerce companies to build long lasting and profitable relationship with their customers. For making strong relationship with these users it is very important to focus on the customer as a whole and making sense of a flood of real time information that goes well beyond demographics or shopping behaviour. There are two entities who will have the access to the system. One is the admin and another one will be the registered user.

Admin can add product details, view all the order details and can also view the sales of the products. User need to register with basic registration details to generate a valid username and password. After the user logins, it can view all the products that are recommended on the homepage compiled by the system based on user’s information. From the recommended products, the user can even further view its details and then if interested to buy, the system gives add to cart option for purchasing the product. The system even has an AI bot with the help of which the user can get answers to queries like features, warranty, price etc. details of the products. This AI Bot even converts text to speech. After selecting the product, user can do payment for the particular product online. Users can view their order history of their purchased product.

### AIM

The main aim of HARDWARE STORE development is to sell products to users. The most successful websites are carefully optimized to achieve a high percentage of purchases. To achieve success ecommerce websites need to integrate all of the latest online closing & upsell techniques available which have been proven to increase chances that a visitor will purchase. There are many important elements that go into building a successful e-commerce website such as removing friction during the purchasing process, making the checkout smooth and easy, making the website fast and attractive, up selling users on related products, incentivizing buyers, reducing cart abandonment, nurturing past buyers to buy again, remarketing to past visitors who haven’t yet purchased, using the proper payment options, having a mobile ready design and many more things which are needed to develop and e-commerce website.

### EXISTING SYSTEM

This existing system of buying goods has several disadvantages. It requires lots of time to travel to the particular shop to buy the goods. It is having lots of manual work. Since everyone is leading busy life now a days, time means a lot to everyone. Also there are expenses for travelling from house to shop. It is less user-friendly. In current system user must go to shop and order products. It is difficult to identify the required product. More over the shop from where we would like to buy something may not be open 24\*7\*365. Hence we have to adjust our time with the shopkeeper’s time or vendor’s time. In current e-commerce system user have to go shop to view the description of the product. It is unable to generate different kinds of report.

### PROPOSED SYSTEM

The proposed system helps in building a website to buy, sell products or goods online using internet connection. Unlike traditional commerce that is carried out physically with effort of a person to go and get products, eCommerce has made it easier for human to reduce physical work and to save time. The basic concept of the application is to allow the customer to shop virtually using the Internet and allow customers to buy the items and articles of their desire from the store.E-commerce is fast gaining ground as an accepted and used business paradigm.

### HARDWARE & SOFTWARE REQUIREMENTS SPECIFICATION

* + 1. **Hardware Requirements**

|  |  |
| --- | --- |
| **Number** | **Description** |
| 1 | PC with 250 GB or more Hard disk. |
| 2 | PC with 2 GB RAM. |
| 3 | PC with Pentium 1 and above. |

* + 1. **Software Requirements**

|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **Type** |
| 1 | Operating System | Windows |
| 2 | Language | JavaScript , HTML |
| 3 | Database | MongoDB |
| 4 | IDE | VS Code |
| 5 | Browser | Chrome, Firefox, Edge |

# CHAPTER 2 FEASIBILITY STUDY

## FEASIBILITY STUDY

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it’s worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

* + 1. Technical Feasibility
    2. Economical Feasibility
    3. Operational Feasibility

## OPERATIONAL FEASIBILITY

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realised. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

### TECHNICAL FEASIBILITY

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on outline design of system requirements in terms of input, processes, output, fields, programs and procedures. This can be qualified in terms of volume of data, trends, frequency of updating inorder to give an introduction to the technical system. The application is the fact that it has been developed on windows XP platform and a high configuration of 1GB RAM on Intel Pentium Dual core processor. This is technically feasible .The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

## ECONOMICAL FEASIBILITY

Establishing the cost-effectiveness of the proposed system i.e. if the benefits do not outweigh the costs then it is not worth going ahead. In the fast paced world today there is a great need of online social networking facilities. Thus the benefits of this project in the current scenario make it economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

# CHAPTER 3

# DESIGN & PLANNING

### SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

* + 1. **WATERFALL MODEL**

The waterfall model was selected as the SDLC model due to the following reasons:

* + - * Technology was adequately understood.
      * Simple and easy to understand and use.
      * There were no ambiguous requirements.
      * Easy to manage due to the rigidity of the model.
      * Each phase has specific deliverables and a review process.
      * Clearly defined stages.
      * Well understood milestones.
      * Easy to arrange tasks.

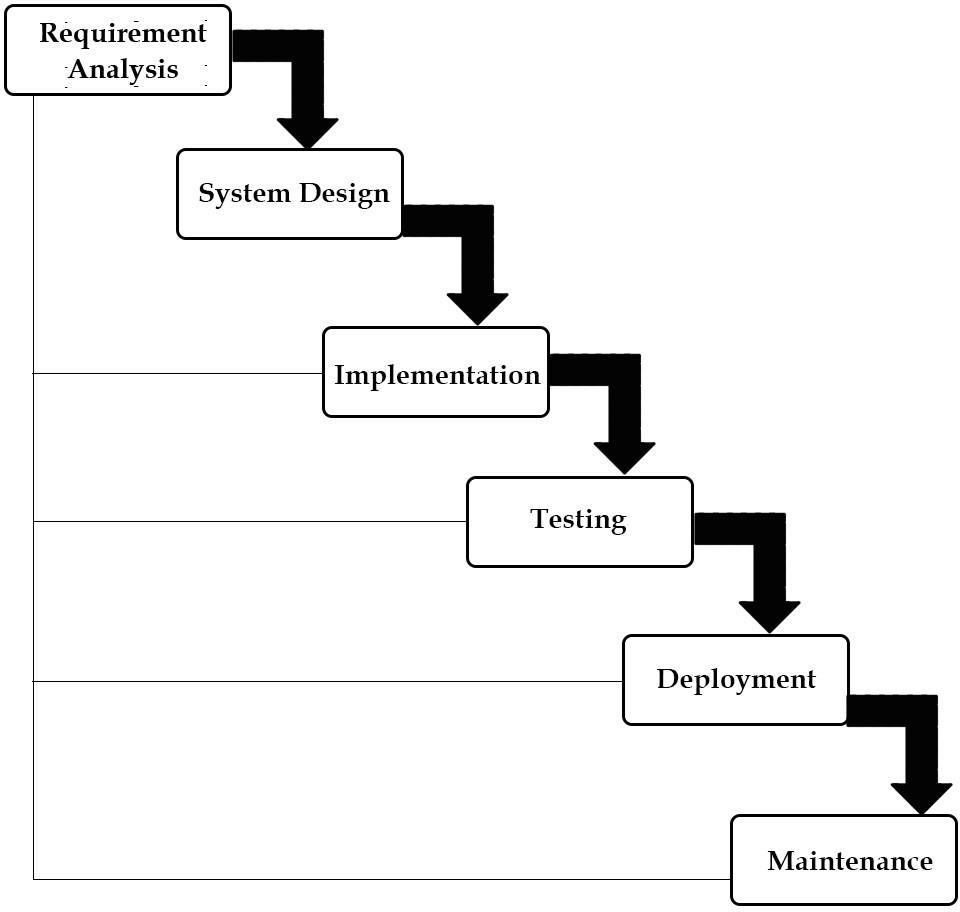


Figure : SDLC Phases

* 1. **GENERAL OVERVIEW**

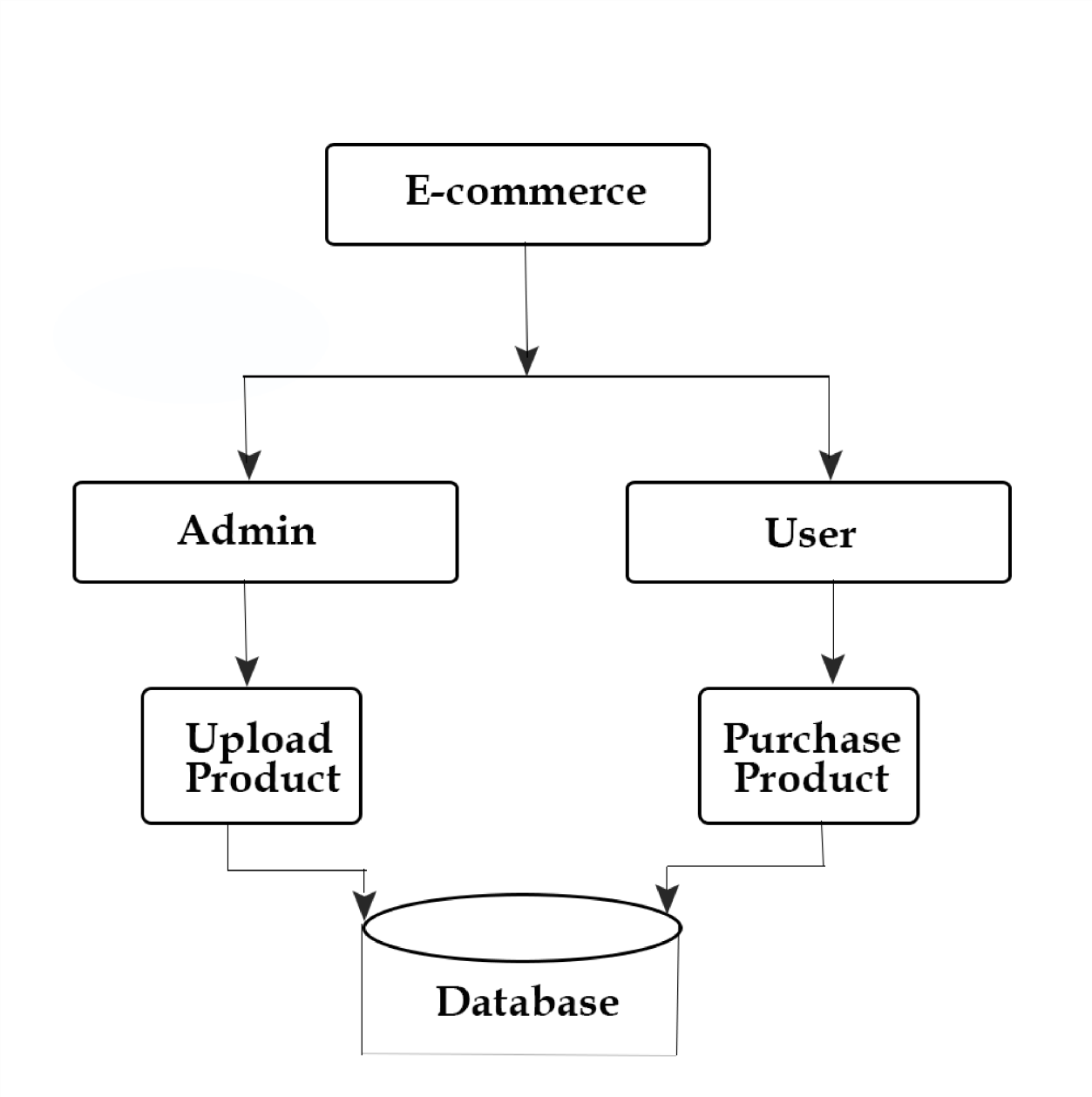


Figure : Block Diagram of E-commerce

* 1. **USE CASE DIAGRAM**

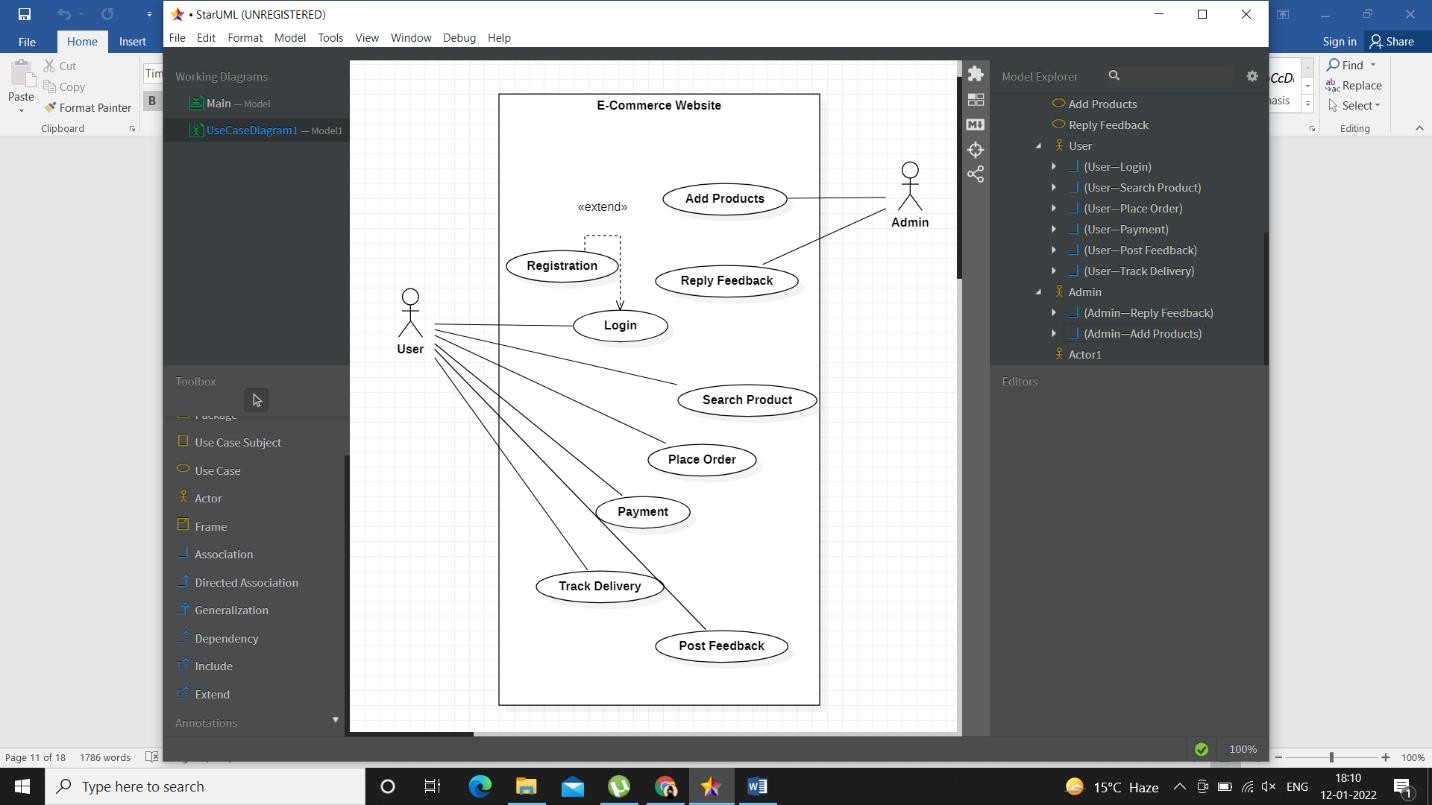
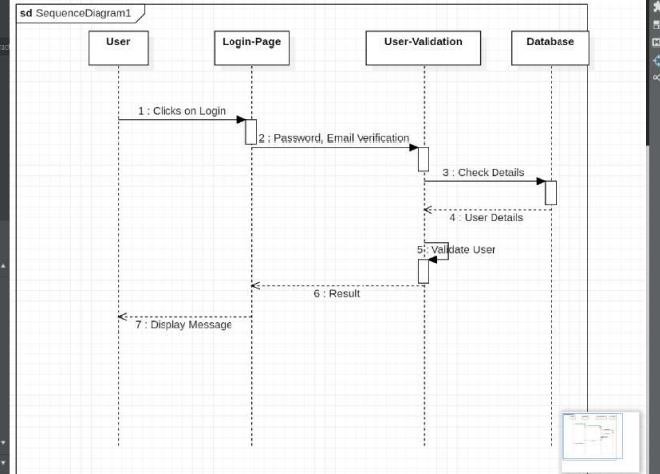
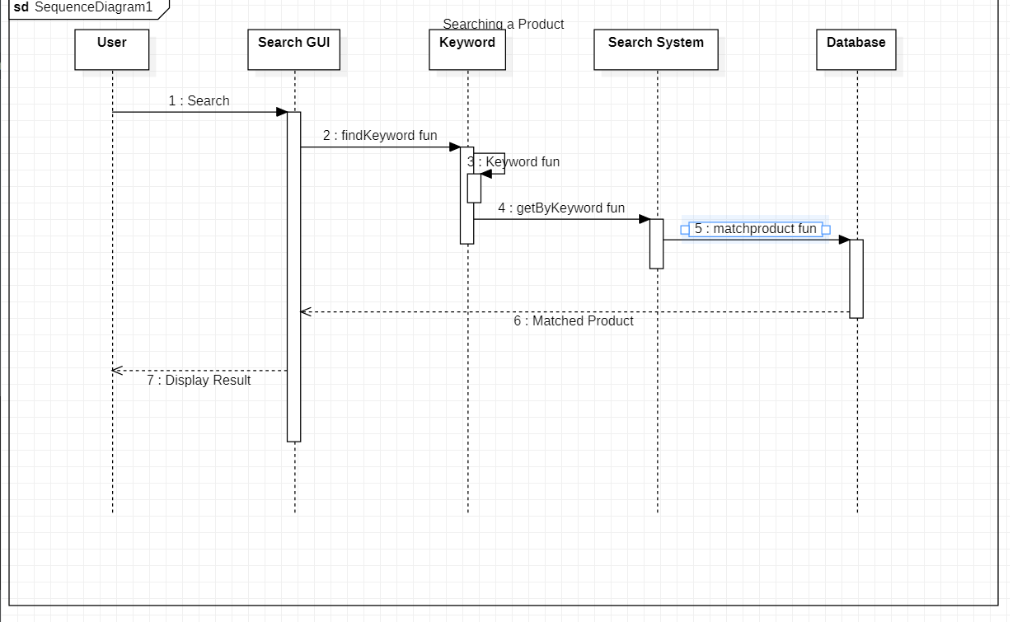


Figure : Use Case Diagram of E-commerce

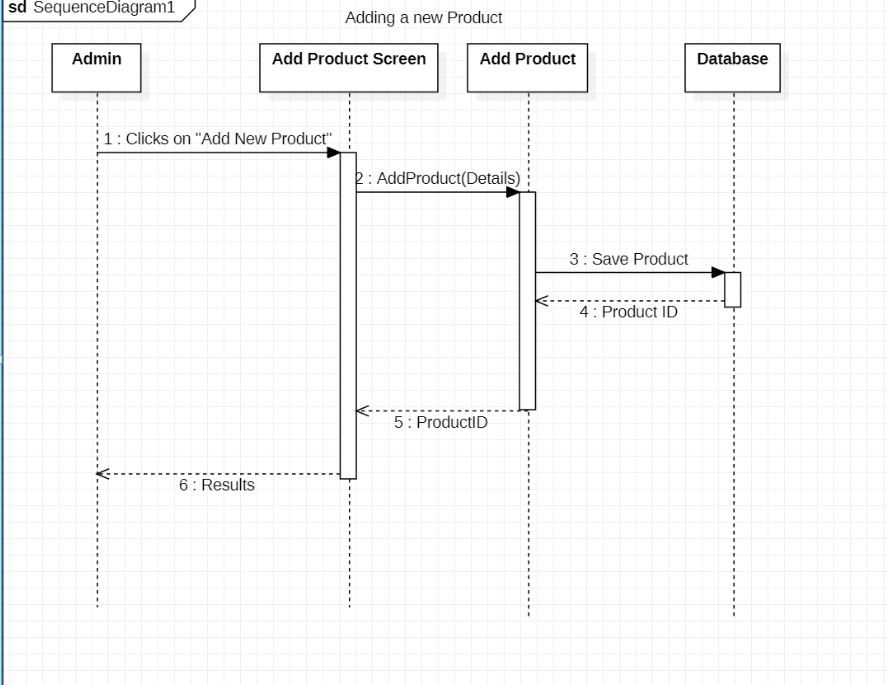
* 1. **SEQUENCE DIAGRAM**
     1. **Login**



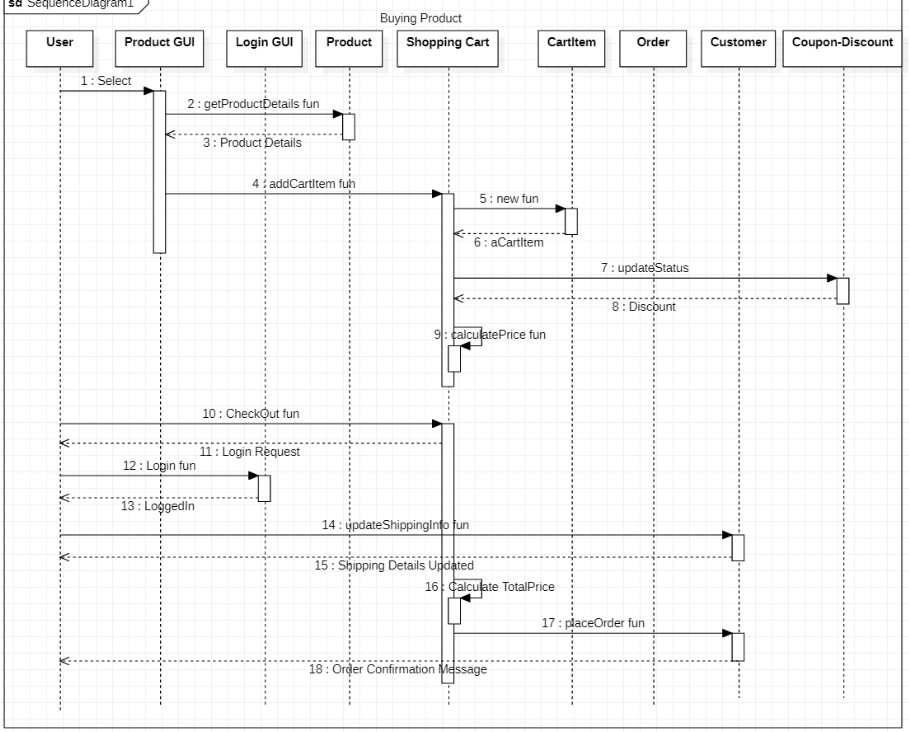
* + 1. **Search**



* + 1. **Add New Product**

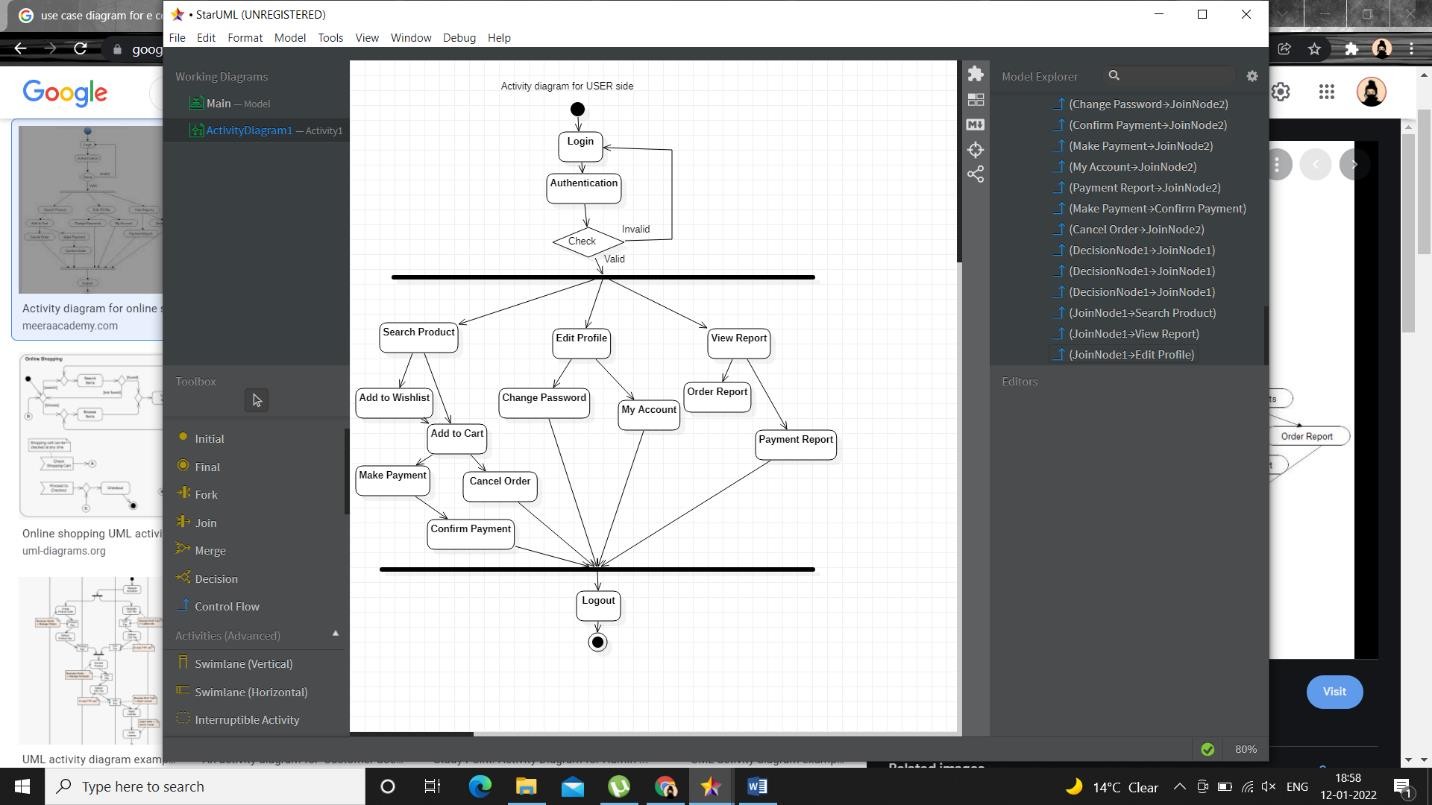


* + 1. **Buy Product**

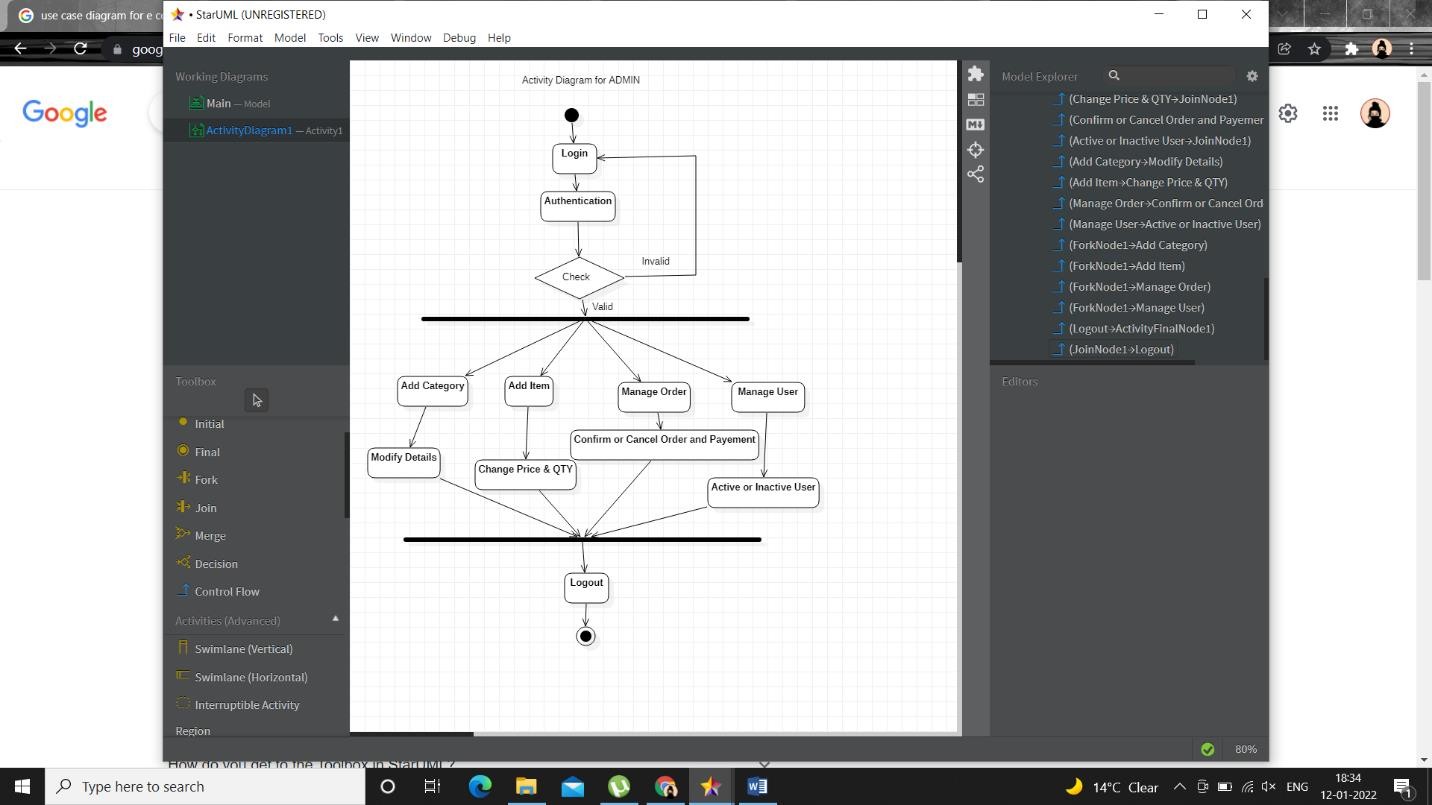


### ACTIVITY DIAGRAM

* + 1. **User Side**

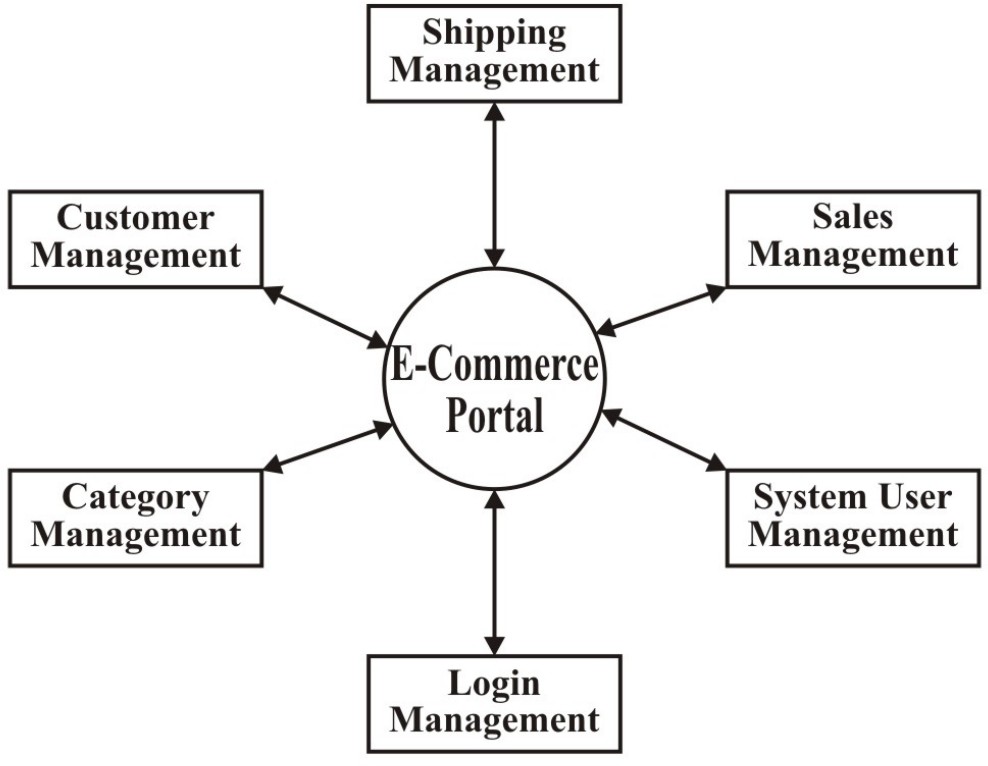


* + 1. **For Admin**

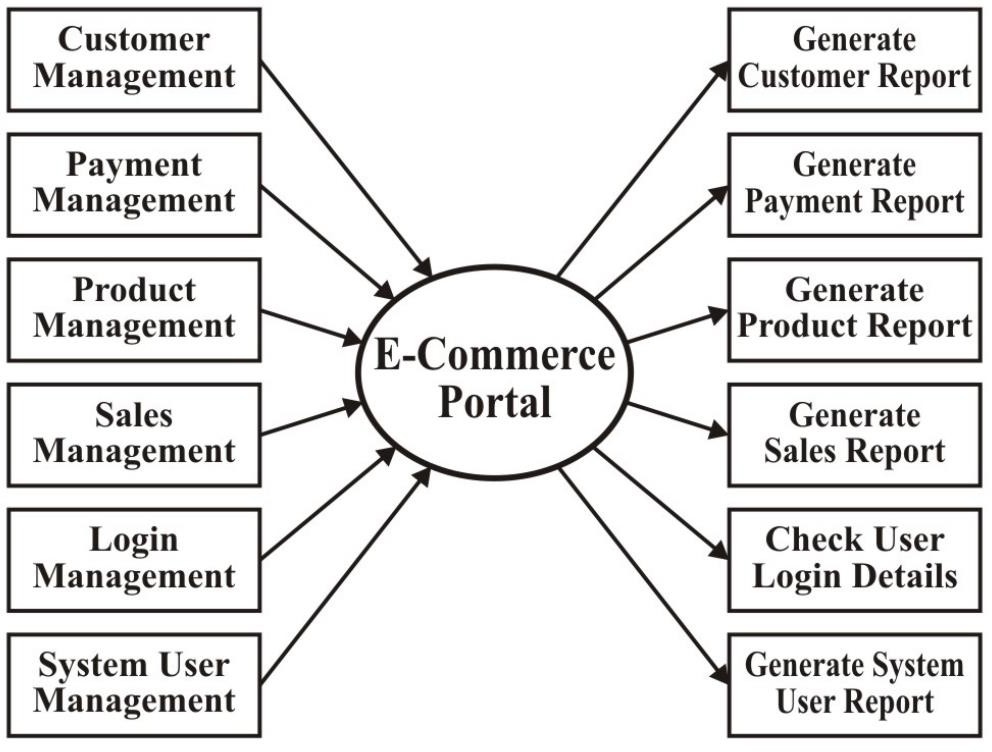


### DFD

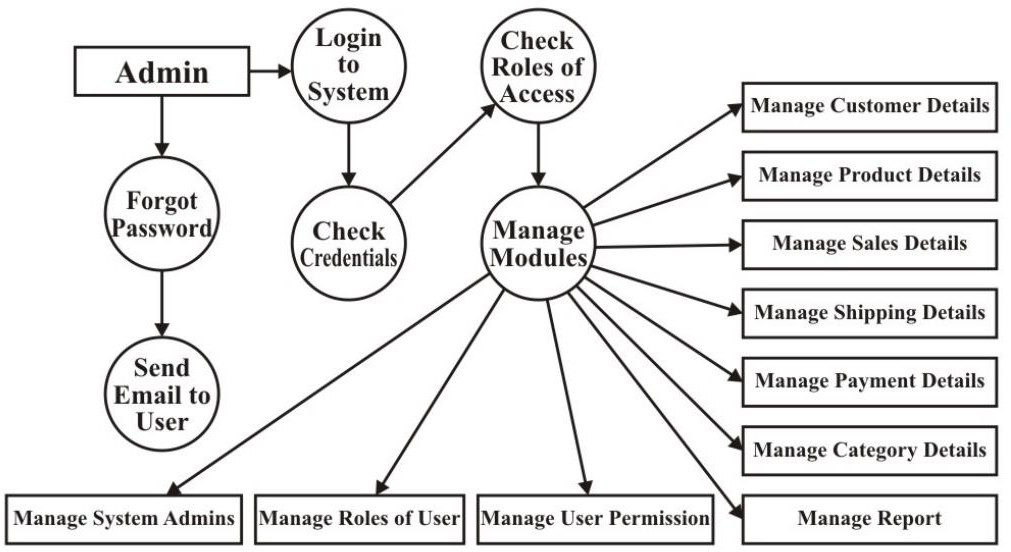
* + 1. **Zero-Level DFD Diagram**



* + 1. **First-Level DFD Diagram**



* + 1. **Second-Level DFD Diagram**



# 

# CHAPTER 4

# FORM DESIGN

### 4.1 LOGIN/REGISTER FORM

Graphical user interface, website

Description automatically generated

Figure :4.1.1- Login For

Graphical user interface, website

Description automatically generated

Figure : 4.1.2- Register User Form

Graphical user interface, website

Description automatically generated

Fig: 4.1.3- Invalid Login

Graphical user interface, website

Description automatically generated

Fig: 4.1.4- Logout User

4.2 HOME PAGE

A screenshot of a bathroom

Description automatically generated

4.3 PRODUCTS PAGE

Graphical user interface, website

Description automatically generated

4.4 SHOPPING CART

Graphical user interface, application

Description automatically generated

4.5 USER PROFILE

Graphical user interface, website

Description automatically generated

4.6 EDIT PROFILE

Graphical user interface, website

Description automatically generated

4.7 UPDATE PASSWORD

Graphical user interface, website

Description automatically generated

# 4.8 SEARCH PRODUCT

# A picture containing graphical user interface Description automatically generated

# CHAPTER 5

# CODING

### INDEX.HTML (HTML FILE)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta

name="description"

content="Web site created using create-react-app"

/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<title>Hardware Store</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

</body>

</html>

### APP.JS (JAVASCRIPT FILE)

import './App.css';

import Header from "./component/layout/Header/Header.js";

import {BrowserRouter as Router, Route} from "react-router-dom";

import WebFont from "webfontloader";

import React, { useEffect, useState } from "react";

import Footer from "./component/layout/Footer/Footer.js";

import Home from "./component/Home/Home.js";

import ProductDetails from "./component/Product/ProductDetails.js"

import Products from "./component/Product/Products.js";

import Search from "./component/Product/Search.js";

import LoginSignUp from './component/User/LoginSignUp';

import store from "./store";

import { loadUser } from './actions/userAction';

import UserOptions from "./component/layout/Header/UserOptions.js";

import { useSelector } from 'react-redux';

import Profile from "./component/User/Profile.js";

import ProtectedRoute from './component/Route/ProtectedRoute';

import UpdateProfile from "./component/User/UpdateProfile.js";

import UpdatePassword from "./component/User/UpdatePassword.js";

import ForgotPassword from "./component/User/ForgotPassword.js";

import ResetPassword from "./component/User/ResetPassword.js";

import Cart from "./component/Cart/Cart.js";

import ConfirmOrder from "./component/Cart/ConfirmOrder.js";

import axios from 'axios';

import Payment from "./component/Cart/Payment.js";

import {Elements} from "@stripe/react-stripe-js";

import {loadStripe} from "@stripe/stripe-js";

import OrderSuccess from "./component/Cart/OrderSuccess.js"

function App() {

const {isAuthenticated, user} = useSelector(state => state.user);

const [stripeApiKey, setStripeApiKey] = useState("");

async function getStripeApiKey() {

const {data} = await axios.get("/api/v1/stripeapikey");

setStripeApiKey(data.stripeApiKey)

}

useEffect(()=>{

WebFont.load({

google: {

families: ["Roboto","Droid Sans", "Chilanka"]

}

})

store.dispatch(loadUser())

getStripeApiKey()

}, [])

return (

<div>

<Router>

<Header />

{isAuthenticated && <UserOptions user = {user} />}

<Route exact path = "/" component = {Home}></Route>

<Route exact path = "/product/:id" component = {ProductDetails}></Route>

<Route exact path = "/products" component = {Products}></Route>

<Route path = "/products/:keyword" component = {Products}></Route>

<Route exact path = "/search" component = {Search}></Route>

<ProtectedRoute exact path = "/account" component = {Profile} />

<ProtectedRoute exact path = "/me/update" component = {UpdateProfile} />

<ProtectedRoute exact path = "/password/update" component = {UpdatePassword} />

<Route exact path = "/password/forgot" component = {ForgotPassword} ></Route>

<Route exact path = "/password/reset/:token" component = {ResetPassword} ></Route>

<Route exact path = "/login" component = {LoginSignUp}></Route>

<Route exact path = "/cart" component = {Cart}></Route>

<ProtectedRoute exact path = "/order/confirm" component = {ConfirmOrder} />

{stripeApiKey && (

<Elements stripe={loadStripe(stripeApiKey)}>

<ProtectedRoute exact path = "/process/payment" component = {Payment} />

</Elements>

)}

<ProtectedRoute exact path = "/success" component = {OrderSuccess} />

<Footer />

</Router>

</div>

);

}

export default App;

### LOGINSIGNUP.JS

import React, { Fragment, useRef, useState, useEffect } from 'react';

import "./LoginSignUp.css";

import Loader from '../layout/Loader/Loader';

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

import MailOutlineIcon from "@material-ui/icons/MailOutline";

import LockOpenIcon from "@material-ui/icons/LockOpen";

import FaceIcon from "@material-ui/icons/Face";

import { useDispatch, useSelector } from 'react-redux';

import {clearErrors, login, register} from "../../actions/userAction";

import {useAlert} from "react-alert";

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

const dispatch = useDispatch();

const alert = useAlert();

const {error, loading, isAuthenticated } = useSelector(state => state.user);

const loginTab = useRef(null);

const registerTab = useRef(null);

const switcherTab = useRef(null);

const [loginEmail, setLoginEmail] = useState("");

const [loginPassword, setLoginPassword] = useState("");

const [user, setUser] = useState({

name: "",

email: "",

password: "",

});

const {name, email, password} = user;

const [avatar, setAvatar] = useState();

const [avatarPreview, setAvatarPreview] = useState("/Profile.png");

const loginSubmit = (e) =>{

e.preventDefault();

dispatch(login(loginEmail, loginPassword));

}

const registerSubmit = (e) => {

e.preventDefault();

const myForm = new FormData();

myForm.set("name", name);

myForm.set("email", email);

myForm.set("password", password);

myForm.set("avatar", avatar);

dispatch(register(myForm));

}

const registerDataChange = (e) => {

if(e.target.value === "avatar"){

const reader = new FileReader();

reader.onload = () =>{

if(reader.readyState === 2){

setAvatarPreview(reader.result);

setAvatar(reader.result);

}

};

reader.readAsDataURL(e.target.files[0]);

} else {

setUser({...user, [e.target.name]: e.target.value})

}

}

const redirect = location.search ? location.search.split("=")[1] : "/account";

useEffect(() => {

if(error){

alert.error(error);

dispatch(clearErrors());

}

if(isAuthenticated){

history.push(redirect);

}

},[dispatch, error, alert, history, isAuthenticated,redirect]);

const switchTabs = (e, tab) =>{

if(tab==="login"){

switcherTab.current.classList.add("shiftToNeutral");

switcherTab.current.classList.remove("shitfToRight");

registerTab.current.classList.remove("shiftToNeutralForm");

loginTab.current.classList.remove("shiftToLeft");

}

if(tab==="register"){

switcherTab.current.classList.add("shiftToRight");

switcherTab.current.classList.remove("shitfToNeutral");

registerTab.current.classList.add("shiftToNeutralForm");

loginTab.current.classList.add("shiftToLeft");

}

}

return (

<Fragment>

{loading ? <Loader /> : (

<Fragment>

<div className='LoginSignUpContainer'>

<div className='LoginSignUpBox'>

<div>

<div className='login\_signUp\_toggle'>

<p onClick={(e) => switchTabs(e, "login")}>LOGIN</p>

<p onClick={(e) => switchTabs(e, "register")}>REGISTER</p>

</div>

<button ref={switcherTab}></button>

</div>

<form className='loginForm' ref={loginTab} onSubmit={loginSubmit}>

<div className='loginEmail'>

<MailOutlineIcon />

<input

type = "email"

placeholder='Email'

required

value = {loginEmail}

onChange = {(e) => setLoginEmail(e.target.value)}

/>

</div>

<div className='loginPassword'>

<LockOpenIcon />

<input

type = "password"

placeholder='Password'

required

value = {loginPassword}

onChange = {(e) => setLoginPassword(e.target.value)}

/>

</div>

<Link to = "/password/forgot">Forgot Password ?</Link>

<input type = "submit" value = "Login" className='loginBtn' />

</form>

<form className='signUpForm'

ref={registerTab}

encType= "multipart/form-data"

onSubmit={registerSubmit}>

<div className='signUpName'>

<FaceIcon />

<input

type = "text"

placeholder='Name'

required

name = "name"

value = {name}

onChange = {registerDataChange}

/>

</div>

<div className='signUpEmail'>

<MailOutlineIcon />

<input

type = "email"

placeholder='Email'

required

name='email'

value = {email}

onChange = {registerDataChange}

/>

</div>

<div className='signUpPassword'>

<LockOpenIcon />

<input

type = "password"

placeholder='Password'

required

name='password'

value = {password}

onChange = {registerDataChange}

/>

</div>

<div id='registerImage'>

<img src = {avatarPreview} alt = "Avatar Preview" />

<input

type = "file"

name = "avatar"

accept='image/\*'

onChange={registerDataChange} />

</div>

<input

type = "submit"

value = "Register"

className='signUpBtn'

/>

</form>

</div>

</div>

</Fragment>

)}

</Fragment>

)

}

export default LoginSignUp

### HOME.JS

**import React, { Fragment, useEffect } from 'react';**

**import "./Home.css";**

**import ProductCard from "./ProductCard.js";**

**import MetaData from '../layout/MetaData';**

**import {clearErrors, getProduct} from "../../actions/productAction";**

**import {useSelector, useDispatch} from "react-redux";**

**import Loader from "../layout/Loader/Loader";**

**import { useAlert } from 'react-alert';**

**const Home = () => {**

**const alert = useAlert();**

**const dispatch = useDispatch();**

**const {loading, error, products} = useSelector(state=>state.products);**

**useEffect(()=>{**

**if(error){**

**alert.error(error);**

**dispatch(clearErrors);**

**}**

**dispatch(getProduct());**

**}, [dispatch, error,alert])**

**return (**

**<Fragment>**

**{loading ? (<Loader />): (<Fragment>**

**<MetaData title="Hardware Store" />**

**<div className="banner">**

**<p>Welcome to Ecommerce</p>**

**<h1>FIND AMAZING PRODUCTS BELOW</h1>**

**<a href="#container">**

**<button>**

**Scroll**

**</button>**

**</a>**

**</div>**

**<h2 className = "homeHeading"> Featured Products</h2>**

**<div className="container" id = "container">**

**{products && products.map((product)=> (<ProductCard key={product.\_id} product = {product} />))}**

**</div>**

**</Fragment>**

**)}**

**</Fragment>**

**)**

**}**

**export default Home**

### ABOUT\_US.JS

import React from 'react'

import pv from './resume\_image.png' import monu from './monu.jpg' import ankit from './ankit.jpeg'; function AboutUs() {

return (

<div>

<section class="our-webcoderskull padding-lg">

<div class="container">

<div class="row heading heading-icon">

<h2>Our Team</h2>

</div>

<ul class="row"> <li class="col-12 col-md-6 col-lg-3">

</li>

<li class="col-12 col-md-6 col-lg-3">

<div class="cnt-block equal-hight about-style">

<figure><img src={pv} class="img-responsive" alt=""/></figure>

<h3><a href=["http://www.w](http://www.webcoderskull.com/)e[bcoderskull.com/"](http://www.webcoderskull.com/)>Prakhar Varshney</a></h3>

<p>Student</p>

<ul class="follow-us clearfix">

<li><a href="#"><i class="fa fa-facebook" aria-hidden="true"></i></a></li>

<li><a href="#"><i class="fa fa-twitter" aria-hidden="true"></i></a></li>

<li><a href="#"><i class="fa fa-linkedin" aria-hidden="true"></i></a></li>

</ul>

</div>

</li>

<li class="col-12 col-md-6 col-lg-3">

<div class="cnt-block equal-hight about-style">

<figure><img src={monu} class="img-responsive" alt=""/></figure>

<h3><a href="#">Manvendra Pratap Singh </a></h3>

<p>Student</p>

<ul class="follow-us clearfix">

<li><a href="#"><i class="fa fa-facebook" aria-hidden="true"></i></a></li>

<li><a href="#"><i class="fa fa-twitter" aria-hidden="true"></i></a></li>

<li><a href="#"><i class="fa fa-linkedin" aria-hidden="true"></i></a></li>

</ul>

</div>

</li>

</ul>

</div>

</section>

</div>

)

}

export default AboutUs

### CONTACT.JS

import React from 'react' import './custom.css'; function Contact() {

return (

<div>

<div class="mystyle">

<section class="p-5 mystyle" >

<h1 class="custom-h1"><b>Contact Us</b></h1><br></br>

<h2 class="custom-h1"><b>We love to hear from you!</b></h2><br></br>

<form class="mb-5 mx-md-5" action="">

<div class="row">

<div class="col-md-4 mb-4">

<input type="text" id="name" class="form-control" placeholder="Name"/>

</div>

<div class="col-md-4 mb-4">

<input type="email" id="email" class="form-control" placeholder="Email"/>

</div>

<div class="col-md-4 mb-4">

<input type="number" id="phone" class="form-control" placeholder="Phone"/>

</div>

</div>

<div class="row">

<div class="col-md-12 mb-4">

<input type="text" id="subject" class="form-control" placeholder="Subject"/>

</div>

</div>

<div class="row">

<div class="col-md-12">

<div class="form-group mb-4">

<textarea class="form-control rounded" id="message" rows="3" placeholder="How can we help?"></textarea>

</div>

<div class="text-center">

<input type="submit" class="form-control" name = "Submit"></input>

</div>

</div>

</div>

</form>

</section>

</div>

</div>

)

}

export default Contact

### SERVER.JS (BACKEND FILE)

**const app = require("./app");**

**const cloudinary = require("cloudinary");**

**const dotenv = require("dotenv");**

**const connectDatabase = require("./config/database")**

**//Hnadling uncaught exception**

**process.on("uncaughtException", (err)=>{**

**console.log(`Error: ${err.message}`);**

**console.log("shutting down the srver duee to uncaught exception");**

**process.exit(1);**

**})**

**//config**

**dotenv.config({path:"backend/config/config.env"});**

**//connecting to database**

**connectDatabase()**

**cloudinary.config({**

**cloud\_name: process.env.CLOUDINARY\_NAME,**

**api\_key: process.env.CLOUDINARY\_API\_KEY,**

**api\_secret: process.env.CLOUDINARY\_API\_SECRET,**

**})**

**app.listen(process.env.PORT,()=>{**

**console.log(`server is working on http://localhost:${process.env.PORT}`)**

**})**

**//Unhandled promise rejection**

**process.on("unhandledRejection", (err)=>{**

**console.log(`Error: ${err.message}`);**

**console.log("shutting down the srver duee to unhandeled promise rejection");**

**ServiceWorkerRegistration.close(()=>{**

**process.exit(1);**

**});**

**});**

### APP.CSS (CSS FILE)

\*{

margin: 0;

scroll-behavior: smooth;

}

/\* Chrome, safari, edge, opera \*/

input ::-webkit-outer-spin-button,

input ::-webkit-inner-spin-button{

-webkit-appearance: none;

margin: 0;

}

/\* Firefox\*/

input[type="number"] {

-moz-appearance: textfield;

}

.greenColor{

color: green;

}

.redColor{

color: red;

}

### LOGINSIGNUP.CSS

**.LoginSignUpContainer {**

**width: 100vw;**

**height: 100vh;**

**max-width: 100%;**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**background-color: rgb(231, 231, 231);**

**position: fixed;**

**top: 0%;**

**left: 0;**

**}**

**.LoginSignUpBox {**

**background-color: white;**

**width: 25vw;**

**height: 70vh;**

**box-sizing: border-box;**

**overflow: hidden;**

**}**

**.login\_signUp\_toggle {**

**display: flex;**

**height: 3vmax;**

**}**

**.login\_signUp\_toggle > p {**

**color: rgba(0, 0, 0, 0.678);**

**font: 300 1vmax "Roboto";**

**transition: all 0.5s;**

**cursor: pointer;**

**display: grid;**

**place-items: center;**

**width: 100%;**

**}**

**.login\_signUp\_toggle > p:hover {**

**color: tomato;**

**}**

**.LoginSignUpBox > div > button {**

**background-color: tomato;**

**height: 3px;**

**width: 50%;**

**border: none;**

**transition: all 0.5s;**

**}**

**.loginForm,**

**.signUpForm {**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**margin: auto;**

**padding: 2vmax;**

**justify-content: space-evenly;**

**height: 70%;**

**transition: all 0.5s;**

**}**

**.signUpForm {**

**transform: translateY(-100%) translateX(-100vmax);**

**}**

**.loginForm > div,**

**.signUpForm > div {**

**display: flex;**

**width: 100%;**

**align-items: center;**

**}**

**.loginForm > div > input,**

**.signUpForm > div > input {**

**padding: 1vmax 4vmax;**

**padding-right: 1vmax;**

**width: 100%;**

**box-sizing: border-box;**

**border: 1px solid rgba(0, 0, 0, 0.267);**

**border-radius: 4px;**

**font: 300 0.9vmax cursive;**

**outline: none;**

**}**

**.loginForm > div > svg,**

**.signUpForm > div > svg {**

**position: absolute;**

**transform: translateX(1vmax);**

**font-size: 1.6vmax;**

**color: rgba(0, 0, 0, 0.623);**

**}**

**.loginForm > a {**

**color: rgba(0, 0, 0, 0.651);**

**text-decoration: none;**

**align-self: flex-end;**

**transition: all 0.5s;**

**font: 500 0.8vmax "Gill Sans";**

**}**

**.loginForm > a:hover {**

**color: black;**

**}**

**#registerImage > img {**

**width: 3vmax;**

**border-radius: 100%;**

**}**

**#registerImage > input {**

**display: flex;**

**padding: 0%;**

**}**

**#registerImage > input::file-selector-button {**

**cursor: pointer;**

**width: 100%;**

**z-index: 2;**

**height: 5vh;**

**border: none;**

**margin: 0%;**

**font: 400 0.8vmax cursive;**

**transition: all 0.5s;**

**padding: 0 1vmax;**

**color: rgba(0, 0, 0, 0.623);**

**background-color: rgb(255, 255, 255);**

**}**

**#registerImage > input::file-selector-button:hover {**

**background-color: rgb(235, 235, 235);**

**}**

**.loginBtn,**

**.signUpBtn {**

**border: none;**

**background-color: tomato;**

**color: white;**

**font: 300 0.9vmax "Roboto";**

**width: 100%;**

**padding: 0.8vmax;**

**cursor: pointer;**

**transition: all 0.5s;**

**border-radius: 4px;**

**outline: none;**

**box-shadow: 0 2px 5px rgba(0, 0, 0, 0.219);**

**}**

**.loginBtn:hover,**

**.signUpBtn:hover {**

**background-color: rgb(179, 66, 46);**

**}**

**.shiftToLeft {**

**transform: translateX(-100%);**

**}**

**.shiftToNeutral {**

**transform: translateX(0%);**

**}**

**.shiftToNeutralForm {**

**transform: translateX(0%) translateY(-100%);**

**}**

**.shiftToRight {**

**transform: translateX(100%);**

**}**

**@media screen and (max-width: 600px) {**

**.LoginSignUpContainer {**

**background-color: white;**

**}**

**.LoginSignUpBox {**

**width: 100vw;**

**height: 95vh;**

**}**

**.login\_signUp\_toggle {**

**height: 5vmax;**

**}**

**.login\_signUp\_toggle > p {**

**font: 300 1.5vmax "Roboto";**

**}**

**.loginForm,**

**.signUpForm {**

**padding: 5vmax;**

**}**

**.loginForm > div > input,**

**.signUpForm > div > input {**

**padding: 2.5vmax 5vmax;**

**font: 300 1.7vmax cursive;**

**}**

**.loginForm > div > svg,**

**.signUpForm > div > svg {**

**font-size: 2.8vmax;**

**}**

**.loginForm > a {**

**font: 500 1.8vmax "Gill Sans";**

**}**

**#registerImage > img {**

**width: 8vmax;**

**border-radius: 100%;**

**}**

**#registerImage > input::file-selector-button {**

**height: 7vh;**

**font: 400 1.8vmax cursive;**

**}**

**.loginBtn,**

**.signUpBtn {**

**font: 300 1.9vmax "Roboto";**

**padding: 1.8vmax;**

**}**

**}**

### PRODUCTDETAILS.JS

import React, { Fragment, useEffect, useState } from 'react'

import Carousel from "react-material-ui-carousel";

import "./ProductDetails.css";

import { useDispatch, useSelector} from "react-redux";

import { clearErrors, getProductDetails } from '../../actions/productAction';

import ReactStars from 'react-rating-stars-component';

import ReviewCard from "./ReviewCard.js";

import Loader from "../layout/Loader/Loader";

import {useAlert} from "react-alert";

import MetaData from '../layout/MetaData';

import {addItemsToCart} from "../../actions/cartAction";

const ProductDetails = ({match}) => {

const dispatch = useDispatch();

const alert = useAlert();

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

const addToCartHandler = () => {

dispatch(addItemsToCart(match.params.id, quantity));

alert.success("Item Added to Cart");

}

useEffect(()=>{

if(error){

alert.error(error);

dispatch(clearErrors());

}

dispatch(getProductDetails(match.params.id))

}, [dispatch, match.params.id, error, alert])

const options = {

edit:false,

color: "rgba(20,20,20,0.1)",

activeColor: "tomato",

size: window.innerWidth <600 ? 20 : 25,

value: product.ratings,

isHalf: true,

};

const [quantity, setQuantity] = useState(1);

const increasequantity = () => {

if(product.stock <= quantity) return;

const qty = quantity + 1;

setQuantity(qty);

}

const decreasequantity = () => {

if(quantity >=1) return ;

const qty = quantity - 1;

setQuantity(qty);

}

return (

<Fragment>

{loading? <Loader /> : (<Fragment>

<MetaData title = {`${product.name} -- HARDWARE STORE`}/>

<div className="ProductDetails">

<div>

<Carousel>

{product.images && product.images.map((item, i)=>(

<img className="CarouselImage" key= {i}

src = {item.url}

alt = {`${i} Slide`} />

))}

</Carousel>

</div>

<div>

<div className = "detailsBlock-1">

<h2>{product.name}</h2>

<p>Product # {product.\_id}</p>

</div>

<div className = "detailsBlock-2">

<ReactStars {...options} />

<span>({product.numOfReviews} Reviews)</span>

</div>

<div className="detailsBlock-3">

<h1>{`₹${product.price}`}</h1>

<div className="detailsBlock-3-1">

<div className="detailsBlock-3-1-1">

<button onClick={decreasequantity}>-</button>

<input readOnly type="number" value= {quantity} />

<button onClick={increasequantity}>+</button>

</div>

<button onClick={addToCartHandler}>

Add to Cart

</button>

</div>

<p>

Status:

<b className={product.Stock < 1 ? "redColor" : "greenColor"}>

{product.Stock < 1 ? "OutOfStock" : "InStock"}

</b>

</p>

</div>

<div className="detailsBlock-4">

Description : <p>{product.description}</p>

</div>

<button className="submitReview">

Submit Review

</button>

</div>

</div>

<h3 className='reviewsHeading'>REVIEWS</h3>

{ product.reviews && product.reviews[0] ? (

<div className='reviews'>

{product.reviews && product.reviews.map((review)=> <ReviewCard key = {review.\_id} review = {review}/>)}

</div>

) : (

<p className='noReviews'>No Reviews Yet</p>

) }

</Fragment>)}

</Fragment>

)

}

export default ProductDetails

**5.11 PRODUCTDETAILS.CSS**

.ProductDetails {

background-color: rgb(255, 255, 255);

width: 100vw;

max-width: 100%;

padding: 6vmax;

box-sizing: border-box;

display: flex;

}

.ProductDetails > div {

width: 100%;

display: flex;

flex-direction: column;

justify-content: space-evenly;

align-items: center;

padding: 2vmax;

box-sizing: border-box;

border: 1px solid white;

}

.ProductDetails > div:last-child {

align-items: flex-start;

}

.CarouselImage {

width: 20vmax;

}

.detailsBlock-1 > h2 {

color: rgb(54, 54, 54);

font: 600 1.6vmax "Roboto";

}

.detailsBlock-1 > p {

color: rgba(54, 54, 54, 0.582);

font: 200 0.6vmax cursive;

}

.detailsBlock-2 {

display: flex;

justify-content: flex-start;

align-items: center;

border-top: 1px solid rgba(0, 0, 0, 0.205);

border-bottom: 1px solid rgba(0, 0, 0, 0.205);

width: 70%;

padding: 1vmax 0;

}

.detailsBlock-2-span {

font: 200 0.8vmax cursive;

color: rgba(0, 0, 0, 0.699);

}

.detailsBlock-3 {

width: 70%;

}

.detailsBlock-3 > h1 {

color: rgba(17, 17, 17, 0.795);

font: 400 1.8vmax "Franklin Gothic Medium";

margin: 1vmax 0;

}

.detailsBlock-3-1 {

display: flex;

align-items: center;

}

.detailsBlock-3-1-1 > button {

border: none;

background-color: rgba(0, 0, 0, 0.616);

padding: 0.5vmax;

cursor: pointer;

color: white;

transition: all 0.5s;

}

.detailsBlock-3-1-1 > button:hover {

background-color: rgba(0, 0, 0, 0.767);

}

.detailsBlock-3-1-1 > input {

border: none;

padding: 0.5vmax;

width: 1vmax;

text-align: center;

outline: none;

font: 400 0.8vmax "Roboto";

color: rgba(0, 0, 0, 0.74);

}

.detailsBlock-3-1 > button:last-child {

border: none;

cursor: pointer;

color: white;

transition: all 0.5s;

background-color: tomato;

font: 500 0.7vmax "Roboto";

border-radius: 20px;

padding: 0.5vmax 2vmax;

margin: 1vmax;

outline: none;

}

.detailsBlock-3-1 > button:last-child:hover {

background-color: rgb(214, 84, 61);

}

.detailsBlock-3 > p {

border-top: 1px solid rgba(0, 0, 0, 0.205);

border-bottom: 1px solid rgba(0, 0, 0, 0.205);

padding: 1vmax 0;

color: rgba(0, 0, 0, 0.651);

font: 400 1vmax "Roboto";

margin: 1vmax 0;

}

.detailsBlock-4 {

color: rgba(0, 0, 0, 0.897);

font: 500 1.2vmax sans-serif;

}

.detailsBlock-4 > p {

color: rgba(0, 0, 0, 0.534);

font: 300 0.8vmax sans-serif;

}

.submitReview {

border: none;

background-color: tomato;

font: 500 0.7vmax "Roboto";

border-radius: 20px;

padding: 0.6vmax 2vmax;

margin: 1vmax 0;

color: white;

cursor: pointer;

transition: all 0.5s;

outline: none;

}

.submitReview:hover {

background-color: rgb(197, 68, 45);

transform: scale(1.1);

}

.submitDialog {

display: flex;

flex-direction: column;

}

.submitDialogTextArea {

border: 1px solid rgba(0, 0, 0, 0.082);

margin: 1vmax 0;

outline: none;

padding: 1rem;

font: 300 1rem "Roboto";

}

.reviewsHeading {

color: #000000be;

font: 500 1.4vmax "Roboto";

text-align: center;

border-bottom: 1px solid rgba(0, 0, 0, 0.226);

padding: 1vmax;

width: 20vmax;

margin: auto;

margin-bottom: 4vmax;

}

.reviews {

display: flex;

overflow: auto;

}

.reviewCard {

flex: none;

box-shadow: 0 0 5px rgba(0, 0, 0, 0.226);

border: 1px solid rgba(56, 56, 56, 0.116);

width: 30vmax;

display: flex;

flex-direction: column;

align-items: center;

margin: 1vmax;

padding: 3vmax;

}

.reviewCard > img {

width: 5vmax;

}

.reviewCard > p {

color: rgba(0, 0, 0, 0.836);

font: 600 0.9vmax "Roboto";

}

.reviewCardComment {

color: rgba(0, 0, 0, 0.445);

font: 300 0.8vmax cursive;

}

.noReviews {

font: 400 1.3vmax "Gill Sans";

text-align: center;

color: rgba(0, 0, 0, 0.548);

}

@media screen and (max-width: 600px) {

.ProductDetails {

flex-direction: column;

height: unset;

}

.ProductDetails > div:last-child {

align-items: center;

}

.detailsBlock-1 > h2 {

font-size: 2.8vmax;

text-align: center;

}

.detailsBlock-1 > p {

text-align: center;

font-size: 1vmax;

}

.detailsBlock-2 {

justify-content: center;

}

.detailsBlock-2 > span {

font-size: 1.5vmax;

}

.detailsBlock-3 > h1 {

font: 700 3vmax "Franklin Gothic Medium";

text-align: center;

}

.detailsBlock-3-1 {

flex-direction: column;

}

.detailsBlock-3-1-1 {

padding: 2vmax 0;

}

.detailsBlock-3-1-1 > button {

padding: 1.2vmax;

width: 4vmax;

}

.detailsBlock-3-1-1 > input {

padding: 1.5vmax;

width: 3vmax;

font: 400 1.8vmax "Roboto";

}

.detailsBlock-3-1 > button:last-child {

font: 500 1.7vmax "Roboto";

padding: 1.5vmax;

width: 20vmax;

margin: 3vmax 0;

}

.detailsBlock-3 > p {

padding: 2.5vmax 0;

text-align: center;

font: 400 2vmax "Roboto";

}

.detailsBlock-4 {

font: 500 2.5vmax sans-serif;

}

.detailsBlock-4 > p {

font: 300 1.8vmax sans-serif;

}

.submitReview {

font: 500 1.7vmax "Roboto";

padding: 1.5vmax;

width: 20vmax;

margin: 3vmax 0;

}

.reviewCard > p {

font: 600 3vw "Roboto";

}

.reviewCardComment {

font: 300 5vw cursive;

}

}

**5.12 SEARCH.JS**

import React, { useState, Fragment } from "react";

import MetaData from "../layout/MetaData";

import "./Search.css";

const Search = ({ history }) => {

const [keyword, setKeyword] = useState("");

const searchSubmitHandler = (e) => {

e.preventDefault();

if (keyword.trim()) {

history.push(`/products/${keyword}`);

} else {

history.push("/products");

}

};

return (

<Fragment>

<MetaData title="Search A Product -- HARDWARE STORE" />

<form className="searchBox" onSubmit={searchSubmitHandler}>

<input

type="text"

placeholder="Search a Product ..."

onChange={(e) => setKeyword(e.target.value)}

/>

<input type="submit" value="Search" />

</form>

</Fragment>

);

};

export default Search;

**5.13 SEARCH.CSS**

.searchBox {

width: 100vw;

height: 100vh;

max-width: 100%;

display: flex;

justify-content: center;

align-items: center;

background-color: rgb(231, 231, 231);

position: fixed;

top: 0%;

left: 0;

}

.searchBox > input[type="text"] {

box-shadow: 0 0 5px rgba(0, 0, 0, 0.274);

background-color: white;

border: none;

color: rgba(0, 0, 0, 0.637);

padding: 1vmax 2vmax;

width: 50%;

outline: none;

border-radius: 0%;

font: 300 1.1vmax cursive;

box-sizing: border-box;

height: 8%;

}

.searchBox > input[type="submit"] {

height: 8%;

border-radius: 0%;

background-color: tomato;

border: none;

padding: 1vmax;

width: 10%;

font: 300 1.1vmax "Roboto";

cursor: pointer;

color: white;

transition: all 0.5s;

}

.searchBox > input[type="submit"]:hover {

background-color: rgb(55, 97, 214);

}

@media screen and (max-width: 600px) {

.searchBox > input[type="text"] {

width: 100%;

font: 300 4vw cursive;

height: 10%;

}

.searchBox > input[type="submit"] {

height: 10%;

width: 30%;

font: 300 4vw "Roboto";

}

}

**5.14 CART.JS**

import React, { Fragment } from 'react'

import "./Cart.css";

import CartItemCard from "./CartItemCard.js";

import { useDispatch, useSelector } from 'react-redux';

import { addItemsToCart, removeItemsFromCart } from '../../actions/cartAction';

import { Typography } from '@material-ui/core';

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

import RemoveShoppingCartIcon from "@material-ui/icons/RemoveShoppingCart";

const Cart = (history) => {

const dispatch = useDispatch();

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

const increaseQuantity = (id, quantity, stock) => {

const newQty = quantity + 1;

if(stock <= quantity){

return;

}

dispatch(addItemsToCart(id, newQty));

}

const decreaseQuantity = (id, quantity) => {

const newQty = quantity + 1;

if(quantity<=1){

return;

}

dispatch(addItemsToCart(id, newQty));

}

const deleteCartItems = (id) => {

dispatch(removeItemsFromCart(id));

}

const checkoutHandler = () => {

history.push("/login?redirect=shipping");

}

return (

<Fragment>

{cartItems.length === 0 ? (

<div className='emptyCart'>

<RemoveShoppingCartIcon />

<Typography> No Product in your Cart</Typography>

<Link to = "/products">View Products</Link>

</div>

) : (

<Fragment>

<div className='cartPage'>

<div className='cartHeader'>

<p>Product</p>

<p> Quantity</p>

<p> Subtotal</p>

</div>

{cartItems && cartItems.map((item) => (

<div className='cartContainer' key={item.product}>

<CartItemCard item = {item} deleteCartItems = {deleteCartItems} />

<div className='cartInput'>

<button onClick={decreaseQuantity(item.product, item.price)}>-</button>

<input type = "number" value={item.quantity} readOnly/>

<button onClick={increaseQuantity(item.product, item.quantity, item.stock)}>+</button>

</div>

<p className='cartSubtotal'>{`Rs${item.price \* item.quantity}`}</p>

</div>

))}

<div className='cartGrossProfit'>

<div></div>

<div className='cartGrossProfitBox'>

<p>Gross Total</p>

<p>{`Rs${cartItems.reduce(

(acc, item) => acc + item.quantity \* item.price,

0

)}`}</p>

</div>

<div></div>

<div className='checkOutBtn'>

<button onClick={checkoutHandler}>Check Out</button>

</div>

</div>

</div>

</Fragment>)}

</Fragment>

)

}

export default Cart

**5.15 CART.CSS**

.emptyCart {

margin: auto;

text-align: center;

padding: 10vmax;

height: 50vh;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

}

.emptyCart > svg {

font-size: 5vmax;

color: tomato;

}

.emptyCart > p {

font-size: 2vmax;

}

.emptyCart > a {

background-color: rgb(51, 51, 51);

color: white;

border: none;

padding: 1vmax 3vmax;

cursor: pointer;

font: 400 1vmax "Roboto";

text-decoration: none;

}

.cartPage {

padding: 5vmax;

}

.cartHeader {

background-color: tomato;

width: 90%;

box-sizing: border-box;

margin: auto;

color: white;

display: grid;

grid-template-columns: 4fr 1fr 1fr;

font: 300 0.7vmax "Roboto";

}

.cartHeader > p {

margin: 10px;

}

.cartHeader > p:last-child {

text-align: end;

}

.cartContainer {

width: 90%;

margin: auto;

display: grid;

grid-template-columns: 4fr 1fr 1fr;

}

.cartInput {

display: flex;

align-items: center;

height: 8vmax;

}

.cartInput > button {

border: none;

background-color: rgba(0, 0, 0, 0.616);

padding: 0.5vmax;

cursor: pointer;

color: white;

transition: all 0.5s;

}

.cartInput > button:hover {

background-color: rgba(0, 0, 0, 0.767);

}

.cartInput > input {

border: none;

padding: 0.5vmax;

width: 1vmax;

text-align: center;

outline: none;

font: 400 0.8vmax "Roboto";

color: rgba(0, 0, 0, 0.74);

}

.cartSubtotal {

display: flex;

padding: 0.5vmax;

height: 8vmax;

align-items: center;

box-sizing: border-box;

font: 300 1vmax cursive;

justify-content: flex-end;

color: rgba(0, 0, 0, 0.753);

}

.cartGrossProfit {

display: grid;

grid-template-columns: 2fr 1.2fr;

}

.cartGrossProfitBox {

border-top: 3px solid tomato;

margin: 1vmax 4vmax;

box-sizing: border-box;

padding: 2vmax 0;

font: 300 1vmax "Roboto";

display: flex;

justify-content: space-between;

}

.checkOutBtn {

display: flex;

justify-content: flex-end;

}

.checkOutBtn > button {

background-color: tomato;

color: white;

border: none;

padding: 0.8vmax 4vmax;

width: 50%;

font: 300 0.8vmax "Roboto";

margin: 1vmax 4vmax;

cursor: pointer;

border-radius: 30px;

}

@media screen and (max-width: 600px) {

.cartPage {

padding: 0;

min-height: 60vh;

}

.cartHeader {

width: 100%;

font: 300 1.7vmax "Roboto";

grid-template-columns: 3fr 1fr 1fr;

}

.cartContainer {

width: 100%;

grid-template-columns: 3fr 1fr 1fr;

}

.cartInput {

height: 20vmax;

}

.cartInput > button {

padding: 1.5vmax;

}

.cartInput > input {

width: 2vmax;

padding: 1.5vmax;

font: 400 1.8vmax "Roboto";

}

.cartSubtotal {

padding: 1.5vmax;

height: 20vmax;

font: 300 2vmax "Roboto";

}

.cartGrossProfit {

display: grid;

grid-template-columns: 0fr 2fr;

}

.cartGrossProfitBox {

padding: 2vmax;

font: 300 2vmax "Roboto";

}

.checkOutBtn > button {

padding: 2vmax 4vmax;

width: 100%;

font: 300 2vmax "Roboto";

}

}

5.16 PAYMENT.JS

import React, { Fragment, useEffect, useRef } from "react";

import CheckoutSteps from "./Cart/CheckoutSteps";

import { useSelector, useDispatch } from "react-redux";

import MetaData from "../layout/MetaData";

import { Typography } from "@material-ui/core";

import { useAlert } from "react-alert";

import {

CardNumberElement,

CardCvcElement,

CardExpiryElement,

useStripe,

useElements,

} from "@stripe/react-stripe-js";

import axios from "axios";

import "./payment.css";

import CreditCardIcon from "@material-ui/icons/CreditCard";

import EventIcon from "@material-ui/icons/Event";

import VpnKeyIcon from "@material-ui/icons/VpnKey";

import { createOrder, clearErrors } from "../../actions/orderAction";

const Payment = ({ history }) => {

const orderInfo = JSON.parse(sessionStorage.getItem("orderInfo"));

const dispatch = useDispatch();

const alert = useAlert();

const stripe = useStripe();

const elements = useElements();

const payBtn = useRef(null);

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

const { user } = useSelector((state) => state.user);

const { error } = useSelector((state) => state.newOrder);

const paymentData = {

amount: Math.round(orderInfo.totalPrice \* 100),

};

const order = {

shippingInfo,

orderItems: cartItems,

itemsPrice: orderInfo.subtotal,

taxPrice: orderInfo.tax,

shippingPrice: orderInfo.shippingCharges,

totalPrice: orderInfo.totalPrice,

};

const submitHandler = async (e) => {

e.preventDefault();

payBtn.current.disabled = true;

try {

const config = {

headers: {

"Content-Type": "application/json",

},

};

const { data } = await axios.post(

"/api/v1/payment/process",

paymentData,

config

);

const client\_secret = data.client\_secret;

if (!stripe || !elements) return;

const result = await stripe.confirmCardPayment(client\_secret, {

payment\_method: {

card: elements.getElement(CardNumberElement),

billing\_details: {

name: user.name,

email: user.email,

address: {

line1: shippingInfo.address,

city: shippingInfo.city,

state: shippingInfo.state,

postal\_code: shippingInfo.pinCode,

country: shippingInfo.country,

},

},

},

});

if (result.error) {

payBtn.current.disabled = false;

alert.error(result.error.message);

} else {

if (result.paymentIntent.status === "succeeded") {

order.paymentInfo = {

id: result.paymentIntent.id,

status: result.paymentIntent.status,

};

dispatch(createOrder(order));

history.push("/success");

} else {

alert.error("There's some issue while processing payment ");

}

}

} catch (error) {

payBtn.current.disabled = false;

alert.error(error.response.data.message);

}

};

useEffect(() => {

if (error) {

alert.error(error);

dispatch(clearErrors());

}

}, [dispatch, error, alert]);

return (

<Fragment>

<MetaData title="Payment" />

<CheckoutSteps activeStep={2} />

<div className="paymentContainer">

<form className="paymentForm" onSubmit={(e) => submitHandler(e)}>

<Typography>Card Info</Typography>

<div>

<CreditCardIcon />

<CardNumberElement className="paymentInput" />

</div>

<div>

<EventIcon />

<CardExpiryElement className="paymentInput" />

</div>

<div>

<VpnKeyIcon />

<CardCvcElement className="paymentInput" />

</div>

<input

type="submit"

value={`Pay - ₹${orderInfo && orderInfo.totalPrice}`}

ref={payBtn}

className="paymentFormBtn"

/>

</form>

</div>

</Fragment>

);

};

export default Payment;

**5.17 PAYMENT.CSS**

.paymentContainer {

display: grid;

place-items: center;

background-color: rgb(255, 255, 255);

height: 65vh;

margin: 2vmax;

}

.paymentForm {

width: 22%;

height: 100%;

}

.paymentForm > p {

font: 400 2vmax "Roboto";

color: rgba(0, 0, 0, 0.753);

border-bottom: 1px solid rgba(0, 0, 0, 0.13);

padding: 1vmax 0;

text-align: center;

width: 50%;

margin: auto;

}

.paymentForm > div {

display: flex;

align-items: center;

margin: 2vmax 0;

}

.paymentInput {

padding: 1vmax 4vmax;

padding-right: 1vmax;

width: 100%;

box-sizing: border-box;

border: 1px solid rgba(0, 0, 0, 0.267);

border-radius: 4px;

outline: none;

}

.paymentForm > div > svg {

position: absolute;

transform: translateX(1vmax);

font-size: 1.6vmax;

color: rgba(0, 0, 0, 0.623);

}

.paymentFormBtn {

border: none;

background-color: tomato;

color: white;

font: 300 0.9vmax "Roboto";

width: 100%;

padding: 0.8vmax;

cursor: pointer;

transition: all 0.5s;

outline: none;

}

.paymentFormBtn:hover {

background-color: rgb(179, 66, 46);

}

@media screen and (max-width: 600px) {

.paymentForm {

width: 90%;

}

.paymentForm > p {

font: 400 8vw "Roboto";

padding: 4vw 0;

width: 60%;

}

.paymentForm > div {

margin: 10vw 0;

}

.paymentInput {

padding: 4vw 10vw;

}

.paymentForm > div > svg {

font-size: 6vw;

}

.paymentFormBtn {

font: 300 4vw "Roboto";

padding: 4vw;

}

}

# CHAPTER 6 TESTING

### : TESTING

* + 1. **INTRODUCTION**

Testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* + - * meets the requirements that guided its design and development,
      * responds correctly to all kinds of inputs,
      * performs its functions within an acceptable time,
      * it is sufficiently usable,
      * can be installed and run in its intended environments, and
      * Achieves the general result its stakeholder’s desire.
    1. **Static vs. dynamic testing:**

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred to as dynamic testing.

Static testing is often implicit, like proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Typical techniques for these are either using stubs/drivers or execution from a debugger environment.

| **Static Testing** | **Dynamic Testing** |
| --- | --- |
| Static testing is a type of white box testing that is performed early in the development life cycle. | Dynamic testing, on the other hand, occurs later in the development process. |
| It is less expensive | It is more expensive |
| Static testing covers more statements in less time than dynamic testing. | There are fewer statement stages in dynamic testing because it just covers a small area of code |
| It is completed prior to the deployment of the code. | It is completed after the deployment of the code. |
| It is carried out at the Verification Stage. | It is carried out at the Validation Stage. |

* + 1. **White-box testing:**

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration, and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

* API testing – testing of the application using public and private APIs (application programming interfaces)
* Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies
* Mutation testing methods
* Static testing methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

* Function coverage, which reports on functions executed
* Statement coverage, which reports on the number of lines executed to complete the test
* Decision coverage, which reports on whether both the True and the False branch of a given tens.
* it has been executed

100% statement coverage ensures that all code paths or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly. Pseudo-tested functions and methods are those that are covered but not specified (it is possible to remove their body without breaking any test case).

* + 1. **Black-box testing:**



Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight.” Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

### TEST CASES

* + 1. **Testing shopping cart**

Some quantity of products was inserted into the shopping cart, and then we proceeded to check out. When we checked out and tested with Paypal, the cart became empty. This indicates that the cart works appropriately as it should. The "index.js" file is responsible for handling the cart of the shop. When an editor runs the file "index.js" the cart() function is called which insert products into the database. After that, the view cart() function can also be invoked to show the product in the cart. The customer can manipulates his cart such as updating the cart or, adding a product to cart.

* + 1. **Customer checking out**

The customer can only check out if he has logged in as a customer. Without being a customer, he is unable to checkout successfully.

# 

# CHAPTER 7 CONCLUSION

In general, today’s businesses must always strive to create the next best thing that consumers will want because consumers continue to desire their products, services etc. to continuously be better, faster, and cheaper. In this world of new technology, businesses need to accommodate to the new types of consumer needs and trends because it will prove to be vital to their business’ success and survival. E-commerce is continuously progressing and is becoming more and more important to businesses as technology continues to advance and is something that should be taken advantage of and implemented. From the inception of the Internet and e-commerce, the possibilities have become endless for both businesses and consumers. Creating more opportunities for profit and advancements for businesses, while creating more options for consumers. However, just like anything else, e-commerce has its disadvantages including consumer uncertainties, but nothing that can not be resolved or avoided by good decision-making and business practices.

# BIBILIOGRAPHY

* + - * https:/[/www.tutorialspoint.com/index.htm](http://www.tutorialspoint.com/index.htm)
      * [https://www.javatpoint.com](https://www.javatpoint.com/)
      * https:/[/www.w3schools.com](http://www.w3schools.com/)
      * https://html.com