A Field Project Report on

# E-COMMERCE WEBSITE

**Submitted**

*In partial fulfillment of the requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

By

| | |
|---|---|
| K. Unnathi | (231FA04760) |
| P. Manohar Reddy | (231FA04796) |
| A. Sri Keerthi Reddy | (231FA04830) |
| P. Hari Siva Sai Manikanta | (231FA04865) |

Under the Guidance of
**Mr.T. Narasimha Rao**
**Assistant Professor, CSE**

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING AND INFORMATICS**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH**
**(Deemed to be University)**
**Vadlamudi, Guntur -522213, INDIA.**

**April, 2025**

VIGNAN'S

FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

## CERTIFICATE

This is to certify that the field project entitled *"E-Commerce Website"* is being submitted by **[K. Unnathi], [231FA04760], [P.Manohar Reddy], [231FA04796], [A.Sri Keerthi Reddy], [231FA04830], and [P. Hari Manikanta], [ 231FA04865]** in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

**Guide**

**Project Review Committee**

**HoD, CSE**

HoD
Dept. of Computer Science & Engineering
VFSTR Deemed to be Univ
VADLAMUDI - 52

VIGNAN'S

FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH

(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

## DECLARATION

Date:24-04-2025

We hereby declare that the work presented in the field project titled "E-Commerce Website" is the result of our own efforts and investigations.

This project is being submitted under the supervision of Mr.T.Narasimha Rao, Assistant Professor, CSE in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

| | | |
|---|---|---|
| K. Unnathi | (231FA04760) | K. Unnathi |
| P. Manohar Reddy | (231FA04796) | P. Manohar Reddy |
| A. Sri Keerthi Reddy | (231FA04830) | A. Sri Keerthi |
| P. Hari Siva Sai Manikanta | (231FA04865) | |

# Contents

# 1. INTRODUCTION

E-commerce has revolutionized the way businesses operate, enabling seamless buying and selling of goods and services online. By leveraging advancements in technology, such as secure payment gateways, AI-driven personalization, and cloud computing, e-commerce platforms have made shopping more accessible, convenient, and efficient for consumers worldwide. These platforms connect sellers with a global audience, offering diverse products and services across industries. However, challenges like cybersecurity, logistics, and intense competition require continuous innovation. This paper explores the fundamental components, benefits, and challenges of e-commerce, highlighting its transformative impact on modern business and consumer behavior. E-commerce, or electronic commerce, is the process of buying, selling, and exchanging goods, services, or information through the internet, revolutionizing traditional business models and consumer behavior. It provides unparalleled convenience, allowing transactions to occur anytime and anywhere, breaking geographical barriers and creating a global marketplace accessible 24/7. Key features of e-commerce include its ability to offer personalized shopping experiences through advanced algorithms, cost efficiency by reducing overhead expenses like store maintenance, and the use of data analytics to gain insights into consumer behavior and market trends. E-commerce is categorized into various models, including Business-to-Consumer (B2C), where businesses sell directly to individual customers; Business-to-Business (B2B), which involves transactions between companies; Consumer-to-Consumer (C2C), where individuals sell to each other on platforms like eBay; and Consumer-to-Business (C2B), where individuals provide goods or services to businesses. While e-commerce offers significant advantages, such as broader market reach, operational efficiency, and convenience for consumers, it also faces challenges like cybersecurity risks, intense competition, and logistical complexities. The integration of advanced technologies such as artificial intelligence, augmented reality, and blockchain continues to shape the evolution of e-commerce, making it a driving force in digital transformation and the global economy.

## 1.1 Problem Definition

The rapid growth of e-commerce has brought significant advancements in how businesses interact with consumers and how goods and services are exchanged globally. One of the primary issues is ensuring cybersecurity, as online transactions and consumer data are increasingly cyberattacking and fraud. Competition in the e-commerce space is also fierce, with both large-scale and small businesses competing for consumer attention in a crowded digital marketplace, making differentiation and customer retention more difficult. Additionally, the rapid pace of technological advancement means that businesses must continuously innovate to stay relevant and meet evolving customer expectations, particularly in areas like personalization and user experience. Finally, the regulatory landscape remains complex, as e-commerce businesses must navigate various legal requirements, and data protection laws, particularly when dealing with cross-border transactions. This paper seeks to define these key problems in e-commerce and explore potential solutions to ensure its continued growth and success in the global market.

## 1.2 Existing Software

E-Commerce Platforms: Online store creation and management tools like Shopify, WooCommerce, and Magento, OpenCart allow businesses to sell products online.

- **Payment Gateways:** Secure systems like PayPal, Stripe, and square handle online payments, ensuring safe transactions for customers.
- **Logistics and Fulfilment:** Services such as Amazon, FedEx, and third- party logistics (3PL) providers manage inventory, order fulfilment, and product delivery.
- **Artificial Intelligence and Personalization**: AI technologies personalize shopping experiences by recommending products based on customer behavior and preferences.
- **Cloud Computing:** Cloud services like AWS and Google Cloud provide scalable infrastructure for e-commerce platforms, ensuring flexibility and data storage.

## 1.3 Proposed Software

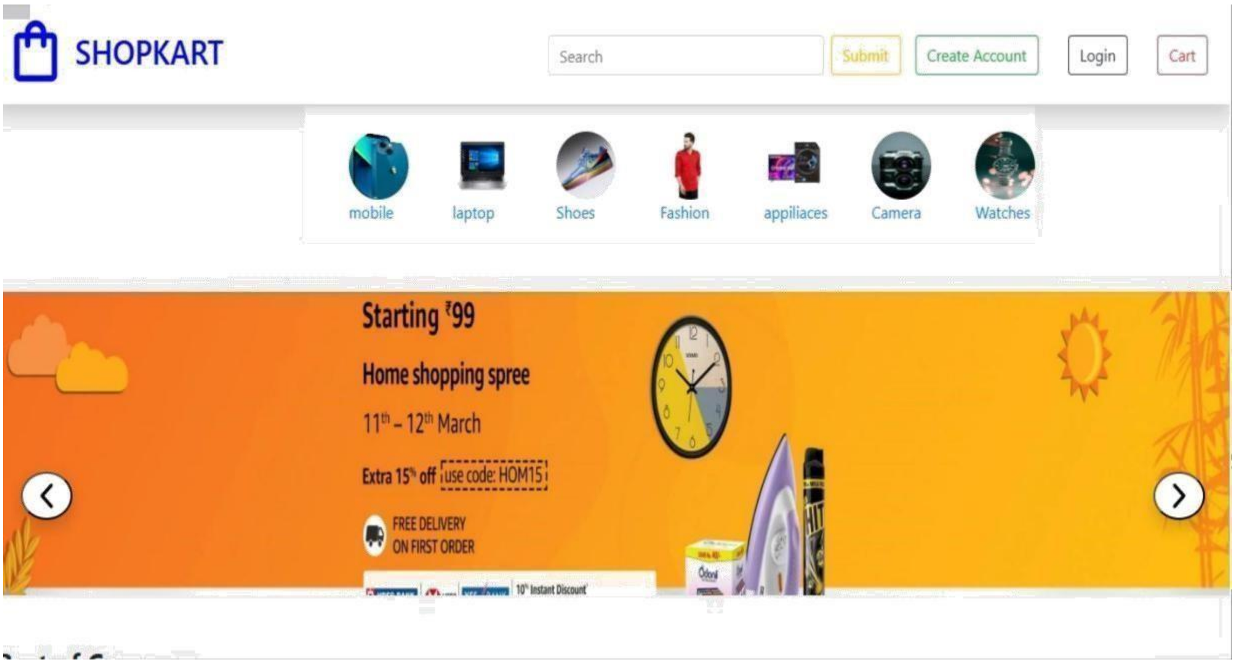


Fig-1.3.1

## 1.4 Literature Review

E-Commerce Systems:

E-commerce has rapidly transformed the business landscape, driven by technological advancements and evolving consumer behavior. This literature review highlights key studies in the development, benefits, challenges, and future trends of e-commerce systems.

**Evolution of E-Commerce:** E-commerce began in the 1990s and has evolved from simple online stores to complex business models like B2C, B2B, and C2C (Laudon & Traver, 2020). Modern e-commerce now includes personalized experiences and advanced systems for inventory management and payment processing (Hossain, 2016).

**Technological Foundations:** Key technologies include secure payment gateways, Web development tools (HTML, CSS, JavaScript), cloud computing, and big data analytics. Payment gateways ensure secure transactions (Hansen et al., 2020), while cloud computing enables scalability during high-demand periods (Marston et al., 2011). AI and machine learning are used for personalized shopping experiences and predictive analytics (Li et al., 2020).

**Consumer Behavior and Personalization**: AI and machine learning algorithms enhance personalization by offering tailored recommendations, which increase customer retention (Kumar & Shah, 2018). Personalized marketing and CRM systems are essential for engaging customers (Cai et al., 2018).

**Future Trends**: Emerging trends like augmented reality (Pantano et al., 2020), blockchain for secure transactions (Zohar et al., 2019), and voice commerce (McKinsey & Company, 2020) are set to further transform e-commerce. These technologies promise to enhance user engagement and provide more immersive shopping experiences.

# 2. SYSTEM REQUIREMENTS

**Frontend:**

HTML, CSS, JavaScript UI
Design (Figma, Sketch)
Responsive (Bootstrap)

**Backend:**

Languages (Python, PHP, Node.js, Java) Databases (MySQL, PostgreSQL, MongoDB) Server (Apache, Nginx)

**Payment Gateway:**

Stripe, PayPal, Razor pay

**Security:**

SSL, 2FA, Encryption

## 2.1 Hardware & Software Requirements

**Hardware**

| Category | Requirement |
|---|---|
| Server | Intel Xeon Quad-Core, 16 GB RAM, 500 GB SSD |
| Network | High-speed internet (1 Gbps or more) |
| Client Device | PC/Laptop/Mobile with minimum 4 GB RAM, modern browser |

**Software**

| Category | Requirement |
|---|---|
| Frontend | HTML, CSS, JavaScript, React/Angular/Vue (optional) |
| Backend | Node.js / Python / PHP / Java |
| Framework | Express / Django / Laravel / Spring Boot |
| Database | MySQL / PostgreSQL / MongoDB |
| Web Server | Apache / Nginx |
| Operating System | Linux (Ubuntu/CentOS) |
| Version Control | Git |
| Other Tools | SSL Certificate, Payment Gateway (e.g., PayPal, Stripe), Cloud Hosting (AWS/Azure) |

## 2.2 Software Requirements Specification (SRS) for E- Commerce Website

**Purpose**

This document outlines the software requirements for an e-commerce website, focusing on user-friendly shopping, secure transactions, and efficient management.

**Overall Description Product Functions:**

**User Management:** Registration, login, profile management.

**Product Management:** Browse, search, filter products.

**Shopping Cart:** Add/remove items, update quantities.

**Order Management:** Checkout, order tracking.

**Payment Integration:** Support multiple payment methods.

**Admin Dashboard:** Manage products, users, and orders.

**Constraints:**

- Comply with GDPR and PCI DSS.
- Support up to 10,000 concurrent users.

**Specific Requirements**

- Secure user authentication.
- Product search and filters by category, price, and ratings.
- Smooth checkout with order confirmation.
- Load pages within 2 seconds.
- Ensure 99.9% uptime.
- Use SSL for secure data transmission.

**External Interfaces**

- User-friendly, responsive UI.
- Integrate with payment gateways (e.g., Stripe, PayPal).

# 3. SYSTEM DESIGNS

**Architecture**

**Frontend**: Built using React or Angular for a responsive and interactive user interface.

**Backend**: Node.js or Django for handling business logic and API integration.

**Database**:

Relational (MySQL/PostgreSQL) for structured data (products, orders, users).

NoSQL (MongoDB) for unstructured data (logs, analytics).

**Hosting**: Cloud-based (AWS, Azure) with auto-scaling and load balancing.

**Key Components**

**User Management**

- Authentication via JWT or OAuth.
- Secure password storage using hashing (e.g., bcrypt).

**Product Management**

- Product catalog stored in the database with indexing for search.
- Search and filter features implemented using Elasticsearch.

**Shopping Cart**

- Session-based cart for guests.
- Persistent cart for logged-in users using database storage.

**Order Management**

- Multi-step checkout process.
- Order tracking with status updates.

**Payment Gateway**

- Integration with Stripe, PayPal, or Razorpay for secure transactions.

**Admin Panel**

- Dashboard for managing products, orders, and users.
- Analytics visualization using libraries like Chart.js or D3.js.

**Deployment**

- **CI/CD Pipeline**: Automate testing and deployment using Jenkins or GitHub

Actions.

- **Monitoring**: Tools like New Relic or Grafana for real-time performance tracking.

**Security**

- Use HTTPS for secure communication.
- Implement firewalls and regular vulnerability scanning.
- Protect against SQL injection, XSS, and CSRF attacks.

### High-Level Diagram

```
[Client Devices]
    |
 [Load Balancer]
    |
 [Application Servers]
    |
 [Database + Cache]
    |
[Payment Gateway, CDN, Search, Shipping APIs]
```

## 3.1 Modules of System

**User Module**: Registration, login, profile management, role-based access.

**Product Management**: CRUD for products, catalog browsing, search, and filters.

**Shopping Cart**: Add/remove items, update quantities, save cart.

**Order Management**: Place orders, track orders, view history, generate invoices.

**Payment Integration**: Secure payment gateways, refunds, transaction history.

**Shipping**: Real-time shipping cost, tracking via APIs.

**Admin Dashboard**: Manage users, products, orders, and view analytics.

**Notifications**: Email/SMS alerts for orders and updates.

## 3.2 UML Diagrams

**Use Case Diagram**

**Description:**

Provides a high-level overview of the system's functionalities and interactions between user roles and the system.

**Actors**:

- **Customer**: Browses products, adds to cart, places orders.
- **Admin**: Manages products, views analytics, handles user management.
- **Payment Gateway**: Processes payments.

**Use Cases**:

- Login/Logout (all actors).
- Browse Products (Customer).
- Add to Cart (Customer).
- Place Order (Customer).
- Process Payment (Payment Gateway).
- Manage Products (Admin).
- View Analytics (Admin).

**Representation**

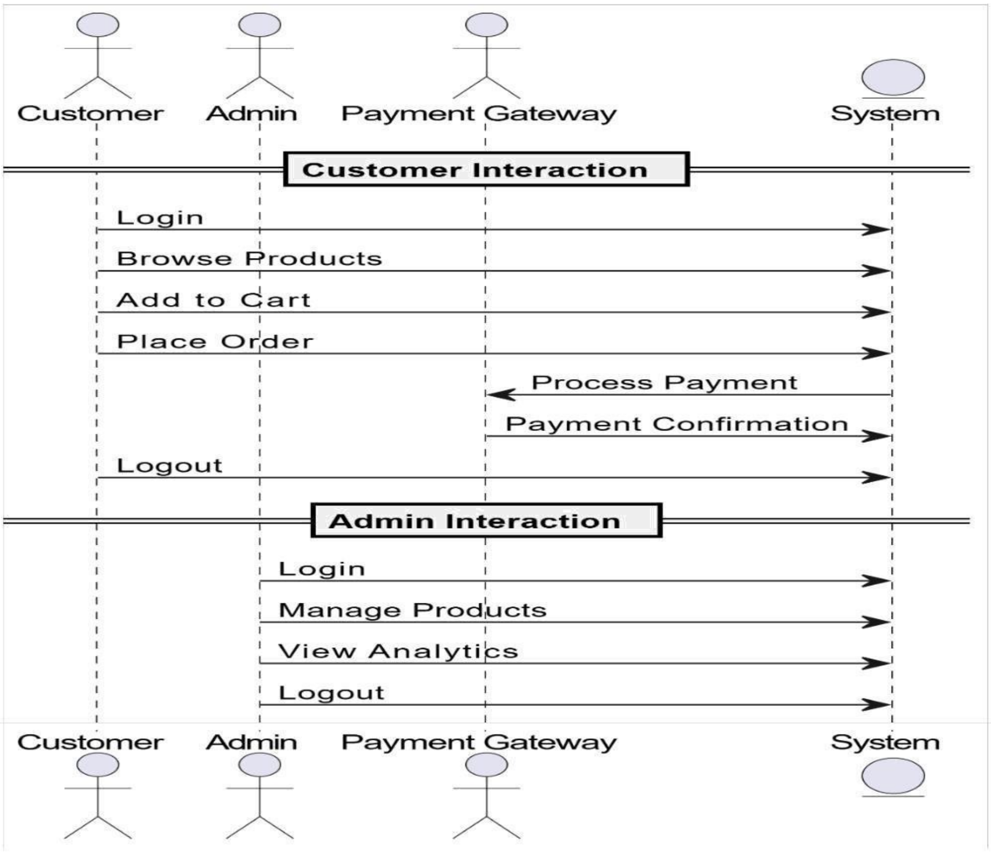


Fig:3.2.1

### Class Diagram:

### Description:

Depicts the structure of the system by representing classes, their attributes, methods, and relationships.

**Classes**:

- **User**:
  - Attributes: UserID, Name, Email, Password, Role.
  - Methods: Login (), Logout (), Register ().
- **Product**:
  - Attributes: ProductID, Name, Price, Stock.
  - Methods: AddProduct (), UpdateProduct (), DeleteProduct ().
- **Order**:
  - Attributes: OrderID, Date, Status.
  - Methods: PlaceOrder (), TrackOrder ().
- **Payment**:
  - Attributes: TransactionID, Amount, PaymentMethod.
  - Methods: ProcessPayment ().

**Representation**:

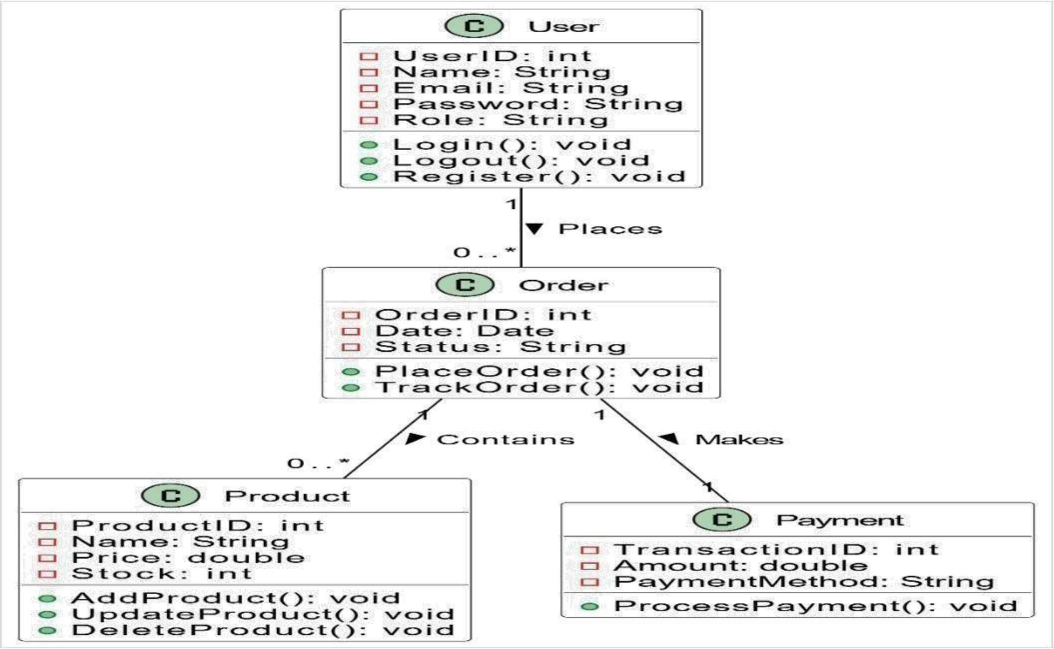


Fig:3.2.2

**Activity Diagram**

Illustrates the flow of activities in the system. **Scenario**: Checkout Process

**Representation**

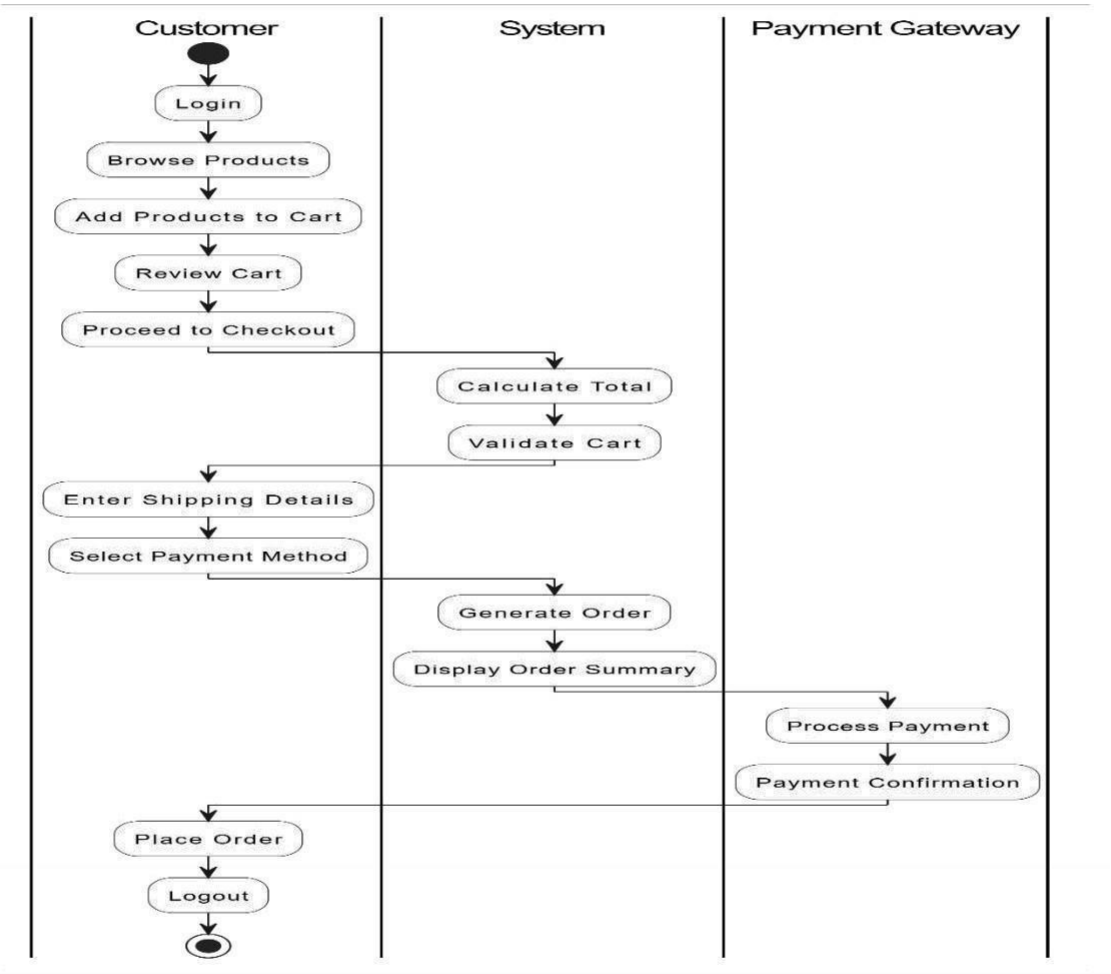


Fig:3.2.3

# 4. IMPLEMENTATION

**Frontend:**

**Languages:** HTML, CSS, JavaScript.

**Frameworks:** React.js, Angular, or Vue.js.

**Features:**

Responsive design for mobile and desktop.

User-friendly navigation for browsing and searching products.

**Backend:**

**Languages:** Node.js, Python (Django/Flask), Java (Spring Boot), or Ruby on Rails.

**Features:**

RESTful APIs for data handling.

Business logic for cart, orders, and payments.

**Database:**

Relational Databases: MySQL, PostgreSQL.

**Hosting:**

- **Cloud Providers:** AWS, Google Cloud, Azure.
- **Services:** Load balancers, auto-scaling, CDN (e.g., Cloudflare).

**Key Modules and Their Implementation**

**User Management:**

- **Features:** Registration, login/logout, profile updates.
- **Implementation:**
  - **Authentication:** Use OAuth 2.0 or JWT.
  - **Encryption:** Store passwords securely using bcrypt.

**Product Management:**

- **Features:** Add, update, delete products (Admin); search and browse (Customer).
- **Implementation:**
  - **Database schema:** Store product details, images, and stock.
  - **Search:** Use Elasticsearch or database indexing for fast queries.

**Order Management:**

- **Features:** Order placement, tracking, history.
- **Implementation:**
    - Use order status enums (e.g., Pending, Shipped, Delivered).
    - Send real-time order updates using WebSockets or notifications.

**Payment Integration:**

- **Features:** Secure payment processing, refunds.
- **Implementation:**
    - Integrate payment gateways (e.g., Stripe, PayPal).
    - Ensure PCI DSS compliance for secure transactions.

**Shipping:**

- **Features:** Real-time tracking, shipping cost calculation.
- **Implementation:**
    - Integrate third-party shipping APIs (e.g., FedEx, UPS).

**Admin Dashboard:**

- **Features:** Manage users, view sales reports, monitor activities.
- **Implementation:**
    - Use data visualization libraries (e.g., Chart.js, Recharts).

**Implementation Steps**

**Set Up Environment:**

Choose your tech stack.

Initialize repositories for version control (e.g., Git).

Configure the development environment (e.g., Node.js, Django).

**Database Design:**

Design schema for users, products, orders, and transactions.

Set up relationships and indexing for optimized performance.

**Frontend Development:**

Create a responsive UI using modern frontend frameworks.

Implement state management (e.g., Redux for React).

**Backend Development:**

Develop APIs for user authentication, product management, and order processing.

## 4.1 Sample Code

**App.css**

```
.title {
color: red;
}
.circle{
display: inline-flex;
background-color:white;
margin-right: 2px;
color: black;
}
.banner{
display:inline-flex;
background-color: white;
 width:100%;
height: 290px;
}
.title h2{
 left:2px;
}
  .all{
    background-color: #eeee;
  }
  .p1{
    display: inline-flex;
    background-color: white;
  }
  .p2{
    background-color: white;
   width:100%;
   height: 3%;
  }
  .add{
    position: absolute;
    right:-9% ;
    top:159%;
  display: inline-flex;
    width:85%;
    background-color:white;
  }
  .add img{ overflow:
    hidden;
  }
  .discount{
    display: inline-flex;
    width:90%;
  }
  .search{
    position: absolute;
    top:11%;
    left: 57%;
  }
  .mobile{ color:
    black;
    background-color:#eeee;
  }
  .mobile img{ background-
    color: aqua;
  }
  .details{
    position: absolute;
    right:20%; top:5%;
  }
```

**App.js**

```
import Sa from './jaya';
import { useState } from 'react'; import './App.css';
import { Link } from 'react-router-dom';
import { MdOutlineShoppingBag } from "react-icons/md"; function App() {
const [data, setData] = useState({ search: ''
});
const change = (e) => {
setData({ ...data, [e.target.name]: e.target.value });
};
const submit = (e) => { e.preventDefault(); console.log(data);
};
return (
<>
<div className='title fixed-top shadow p-2 bg-light d-flex align-items- center justify-content-
between'>
<h2 className='d-flex align-items-center'>
<MdOutlineShoppingBag style={{ width: "70px", height: "70px",color:'red'
}} />
<span className='ml-2'>SHOPKART</span>
</h2>
<form onSubmit={submit} className='d-flex align-items-center'>
<input type='text' name='search'
placeholder='Search' onChange={change} className='form-control mr-2' style={{ width: "300px"
}}
/>
<button className="btn btn-outline-warning" type="submit">Submit</button>
<Link to="http://localhost:3000/login/" className='btn btn-outline- success m-3'>Create
Account</Link>
<Link to="http://localhost:3000/create/" className='btn btn-outline-dark m-3'>Login</Link>
<Link to="http://localhost:3000/cart" className='btn btn-outline-danger m-3'>Cart</Link>
</form>
</div>
</>
);
}
```

```
export default App;
```

App.test.js

```
import { render, screen } from '@testing-library/react'; import App from './App';
test('renders learn react link', () => { render(<App />);
const linkElement = screen.getByText(/learn react/i); expect(linkElement).toBeInTheDocument();
});
```

**banner.jsx**

```
import { useState, useEffect } from "react"; import axios from 'axios';
import './App.css';
import { ChevronLeft, ChevronRight } from "react-feather"; import { Link } from "react-router-dom";
function Banner() {
const [data, setData] = useState([]);
const [currentIndex, setCurrentIndex] = useState(0);

const fetchData = async () => { try {
const response = await axios('http://127.0.0.1:8000/banner/'); setData(response.data);
} catch (error) {
console.error("Error fetching data:", error);
}
};
useEffect(() => { fetchData();
}, []);

useEffect(() => {
if (data.length > 0) {
const interval = setInterval(() => {
setCurrentIndex((prevIndex) => (prevIndex + 1) % data.length);
}, 4000);
return () => clearInterval(interval);
}
}, [data]);
```

```
const nextImage = () => {
setCurrentIndex((prevIndex) => (prevIndex + 1) % data.length);

};
const prevImage = () => {
setCurrentIndex((prevIndex) => (prevIndex - 1 + data.length) % data.length);
};
return (
<div className="banner overflow-hidden">
<div className="overflow-hidden flex-box">
<Link to=""></Link>
{data.length > 0 && (
<div>
<img
src={data[currentIndex]?.img} alt={`Banner ${currentIndex}`}
style={{ width: "100%", height: "360px", transform:
'translateX(0)' }}
className="flex transition-transform ease-out duration-500"
/>
<button
onClick={prevImage}
style={{ position: 'absolute', left: '2%', top: '70%' }} className="rounded-circle shadow bg-white"
>
<ChevronLeft size="40" />
</button>
<button
onClick={nextImage}
style={{ position: 'absolute', right: '2%', top: '70%' }} className="rounded-circle shadow bg-white"
>
<ChevronRight size="40" />
</button>
</div>
)}
</div>
</div>
);
}
export default Banner;
```

**cart.jsx**

```
import React, { useState, useEffect } from "react"; import axios from "axios";
import { IoMdAddCircle } from "react-icons/io";
import { IoRemoveCircleSharp } from "react-icons/io5";

function Car() {
const [cart, newCart] = useState([]);
const [no,setNo]=useState(1) useEffect(() => {
const fetchProducts = async () => { try {
const response = await axios.get("http://localhost:9000/products"); newCart(response.data);
console.log(response.data);
} catch (error) {
console.error("Error fetching products:", error);
};
fetchProducts();
}, []);
const totalprice=cart.reduce((total,item)=>total+item.price*no,0) const next=()=>{
if(no>=5){
alert('max products reached')
}
else{
setNo(no+1)
}
}
const neg=()=>{ if(no>0){
setNo(no-1)
}
}
const click=(carty)=>{ console.log(carty)
alert("product successfully buyed from cart")
}
return (
<div>
<ul>
{cart.map((carty) => (
<div key={carty.id}className="card row-md-3 bg-white">
<h1 style={{position:"fixed",top:"0px",right:"3px"}}>Your Cart price:{totalprice}</h1>
<div className="card-body row md-5">
<img className="card-img"src={carty.img} alt={carty.name} style={{ width: "160px", height:
"200px" }} />
</div>
<div className="card-body row md-2">
<h2 className=" card-title">{carty.name}</h2>
<h3>Price: ${carty.price}</h3></div>
<h3>Discount:{carty.discount}</h3>
<button style={{width:'60px',border:"0px",outline:'none'}} onClick={next}><IoMdAddCircle
style={{backgroundColor:'white',color:'orange',width:'60px',height:'40px'}}
/></button><h3 style={{color:'red',width:'70px',border:'2px solid black'}}>{no}</h3>
<button style={{width:'60px',border:"0px",outline:'none'}} onClick={neg}><IoRemoveCircleSharp
style={{backgroundColor:'white',color:'orange',width:'60px',height:'40px'}}
/></button>
<button onClick={()=>click(carty)} className="btn btn-danger col-md-2 m-2">Buynow</button>
</div>
))}

</ul>
</div>
);
}
export default Car;
```

**circle.jsx**

```
import { useState,useEffect } from "react"; import axios from 'axios'
import './App.css';
import { Link } from "react-router-dom";
function Circle(){
const [data,setData]=useState([])
const c = async () => {
const response=await axios('http://127.0.0.1:8000/circle/') setData(response.data)
}
useEffect(()=>{ c()
},[])
return(
<div className="circle mt-5 md-2">
{data.map((product) => (
<div key={product.id}>
<Link to={`/circle/${product.id}`}>
<img src={product.image} alt={product.text} class=" mt-4 mr-10 ml-5 rounded-
circle"style={{width:"65px",height:"65px"}}/>
<p className="ml-5">{product.text}</p></Link>
</div>
))}
</div>
)
}
export default Circle
```

**create.jsx**

```
import React, { useState, createContext } from "react"; import axios from "axios";
import { useNavigate } from "react-router-dom"; import './App.css';
export const Acontext = createContext(); function Create({ children }) {
const navigate = useNavigate();
const [data, setData] = useState({ name: "",
password: "",
});
const [message, setMessage] = useState(""); const [token, setToken] = useState("");
if (token) { navigate("/");
}

const submit = (e) => {
setData({ ...data, [e.target.name]: e.target.value });
};

const wsubmit = async (e) => { e.preventDefault();
```

```
try {
const response = await axios.post("http://localhost:9000/login", data); setToken(response.data.token);
} catch (error) {
console.error("No account found", error); setMessage("No Account Found");
}
};
return (
<>
<div className="bg1">
<h1 style={styles.header}>Shopsy</h1>
</div>
<div className="login-container">
{message && <p style={styles.errorMessage}>{message}</p>}
<Acontext.Provider value={{ token }}>
{children}
</Acontext.Provider>
<div className="login-card" style={styles.loginCard}>
<form onSubmit={wsubmit}>
<h3 style={styles.formTitle}>Login</h3>
<div style={styles.inputContainer}>
<label htmlFor="name">Enter Name</label>
<input
className="input" type="text"
placeholder="Enter your name" onChange={submit} name="name"
required style={styles.input}
/>
</div>
<div style={styles.inputContainer}>
<label htmlFor="password">Enter Password</label>
<input
className="input" type="password" placeholder="Enter your password" onChange={submit}
name="password"
required style={styles.input}
/>
</div>
<button className="btn" type="submit" style={styles.button}>Login</button>
</form>
</div>
</div>
</>
```

package.json

```
{
"name": "node",
"version": "1.0.0",
"main": "test.js", "scripts": {
"build": "netlify deploy --prod", "start": "node test.js",
"test": "echo \"Error: no test specified\" && exit 1", "server": "nodemon test.js"
},
"author": "",
"license": "ISC",
"description": "", "dependencies": { "bcrypt": "^5.1.1",
"body-parser": "^1.20.3",
"cli": "^1.0.1",
"cors": "^2.8.5",
"dotenv": "^16.4.7",
"express": "^4.21.2",
"jsonwebtoken": "^9.0.2",
"mongodb": "^6.12.0",
"mongoose": "^8.9.5",
"multer": "^1.4.5-lts.1",
"netlify": "^13.2.0",
"netlify-lambda": "^2.0.16",
"nodemon": "^3.1.9",
"serverless-http": "^3.2.0"

}
}
```

test.js

```
const express = require("express"); const mongoose = require("mongoose"); const models1 =
require("./model2"); const jwt = require("jsonwebtoken"); const cors = require("cors");
const multer = require("multer"); const path = require("path"); const model3=require('./model3')
const url = "mongodb://127.0.0.1:27017/new";
try {
mongoose.connect(url); console.log("Database is connected");
} catch (error) {
console.log("Not connected to MongoDB database");
}
const am = express(); am.use(cors());
am.use(express.json());
const storage = multer.diskStorage({ destination: (req, file, cb) => {
cb(null, 'uploads/');
},
filename: (req, file, cb) => {
cb(null, Date.now() + path.extname(file.originalname));
}
});
const upload = multer({ storage: storage }); am.post("/valid", async (req, res) => {
const { name, password } = req.body;

const verify1 = new models1({ name, password }); const wait = await models1.findOne({ name });
try {
if (wait) {
console.log("User  already exists");
res.status(400).json({ success: false, message: "User  already exists" });
} else {
await verify1.save();
```

```
);
}
const styles = { header: {
textAlign: 'center', fontSize: '3.5rem', fontWeight: '700', color: '#fff',
backgroundColor: '#4CAF50', padding: '40px 0',
margin: 0,
textTransform: 'uppercase',
},
loginContainer: {  display: 'flex', justifyContent: 'center', alignItems: 'center', height: '100vh',
left:"30%",
backgroundColor: '#f0f0f0', padding: '20px',
},
loginCard: { backgroundColor: '#fff', padding: '40px 30px', borderRadius: '12px',
boxShadow: '0 12px 24px rgba(0, 0, 0, 0.15)',
width: '100%', position:'absolute', left:'30%', marginTop:'10px', maxWidth: '400px', textAlign:
'center',
fontFamily: 'Arial, sans-serif',
},
formTitle: { marginBottom: '20px', fontSize: '1.5rem', fontWeight: '600',
color: '#333',
},
inputContainer: { marginBottom: '20px', textAlign: 'left',
},
input: {
width: '100%', padding: '14px', fontSize: '16px', borderRadius: '6px',
border: '1px solid #ddd', outline: 'none', marginTop: '8px',
transition: 'border-color 0.3s ease, box-shadow 0.3s ease',
},
button: {
width: '100%', padding: '14px',
backgroundColor: '#4CAF50', border: 'none',
borderRadius: '6px', fontSize: '16px', color: '#fff',
cursor: 'pointer', fontWeight: '600',
transition: 'background-color 0.3s, transform 0.3s ease',
},
errorMessage: { color: '#ff4d4d', fontSize: '16px',
marginBottom: '20px', fontWeight: 'bold', textAlign: 'center',
},
};
export default Create;

discount.jsx
import axios from "axios";
import { useState,useEffect } from "react"; import './App.css';
function Discount(){
const [data,setData]=useState([])
const a=async()=>{
const response=await axios("http://127.0.0.1:8000/discount/") setData(response.data)
}
useEffect(()=>{ a()
},[])
return(
<div>
<h3><b>Discount</b></h3>
<div className="discount card ml-3">
<div className="row fade-in-down">
{
data.map((products)=>(
<ul key={products.id}>
<div className=" card md-2 mt-3">
<img
src={products.image}alt={products.id}style={{width:'355px',height:'250px'}}cl
assName="card-img-top  "></img>
</div>
</ul>
)
}
</div>
</div>
</div>
)
}
export default Discount
```

getimages.jsx

```
import React, { useEffect, useState } from "react"; import axios from 'axios';
function Get() {
const [op, setOp] = useState([]); const fetchImages = async () => {
try {
const response = await axios("http://localhost:2000/pro"); setOp(response.data);
} catch (error) {
console.error("Error fetching images:", error);
}
};
useEffect(() => { fetchImages();
}, []);

return (
<div>
{op.map((product) => (
<div key={product._id}>
<img src={product.bannerimg} alt={product._id}/>
</div>
))}
</div>
);
}
export default Get;
```

model2.js

```
const mongoose=require("mongoose") const verify=new mongoose.Schema({
name:{
type:String, required:true
},
password:{ type:String,
require:true
}
})
module.exports=mongoose.model("authenticati",verify)
model3.js
const mongoose=require("mongoose") const Cart=new mongoose.Schema({
name:{
type:String, require:true
},
img:{ type:String, require:true
},
price:{
type:String, require:true
},
discount:{ type:String, require:true
}
})
module.exports=mongoose.model("cart",Cart)
console.log("Successfully stored");
res.status(201).json({ success: true, message: "User created successfully" });
}
} catch (error) { console.log(error);
res.status(500).json({ success: false, message: "Internal server error" });
}
});
// User login route
am.post("/login", async (req, res) => { const { name, password } = req.body;
const log = await models1.findOne({ name }); if (log && log.password === password) {
const token = jwt.sign({ name: log.name }, "hari123", { expiresIn: "1h" });
res.status(200).json({ token }); console.log(token);
} else {
res.status(400).json("Failed"); console.log("Invalid credentials");
}
});
const tokenizer = (req, res, next) => {
const token = req.headers['Authorization']?.split(' ')[1]; if (!token) {
return res.status(400).json("No token provided!");
} else {
jwt.verify(token, "hari123", (err, decoded) => { if (err) {
console.log(err);
return res.status(401).json("Unauthorized");
} else {
req.name = decoded.name; next();
}
});
```

```
}
};
```

## 4.2 Test Cases

### User Management

- UM001: Verify user registration with valid details.
- UM002: Verify login with valid credentials.
- UM003: Verify error message for invalid login.
- UM004: Test password reset functionality.

### Product Management

- PM001: Search products with a valid keyword.
- PM002: Apply filters and check results.
- PM003: Add a product to the catalog (Admin).

### Shopping Cart

- SC001: Add an item to the cart
- SC002: Remove an item from the cart.
- SC003: Update the quantity of an item.

### Checkout

- CO001: Complete checkout with valid payment.
- CO002: Handle invalid payment during checkout.
- CO003: Prevent checkout with an empty cart.

**Payment Integration**

- PI001: Process payment successfully.
- PI002: Handle payment failure with invalid details.
- PI003: Test refund functionality.
- OM001: View order history.
- OM002: Cancel an order and verify status.
- OM003: Track an active order.
- ST001: Prevent SQL injection in input fields.
- ST002: Restrict unauthorized access to admin pages.

# 5. RESULTS

## 5.1 Output Screens

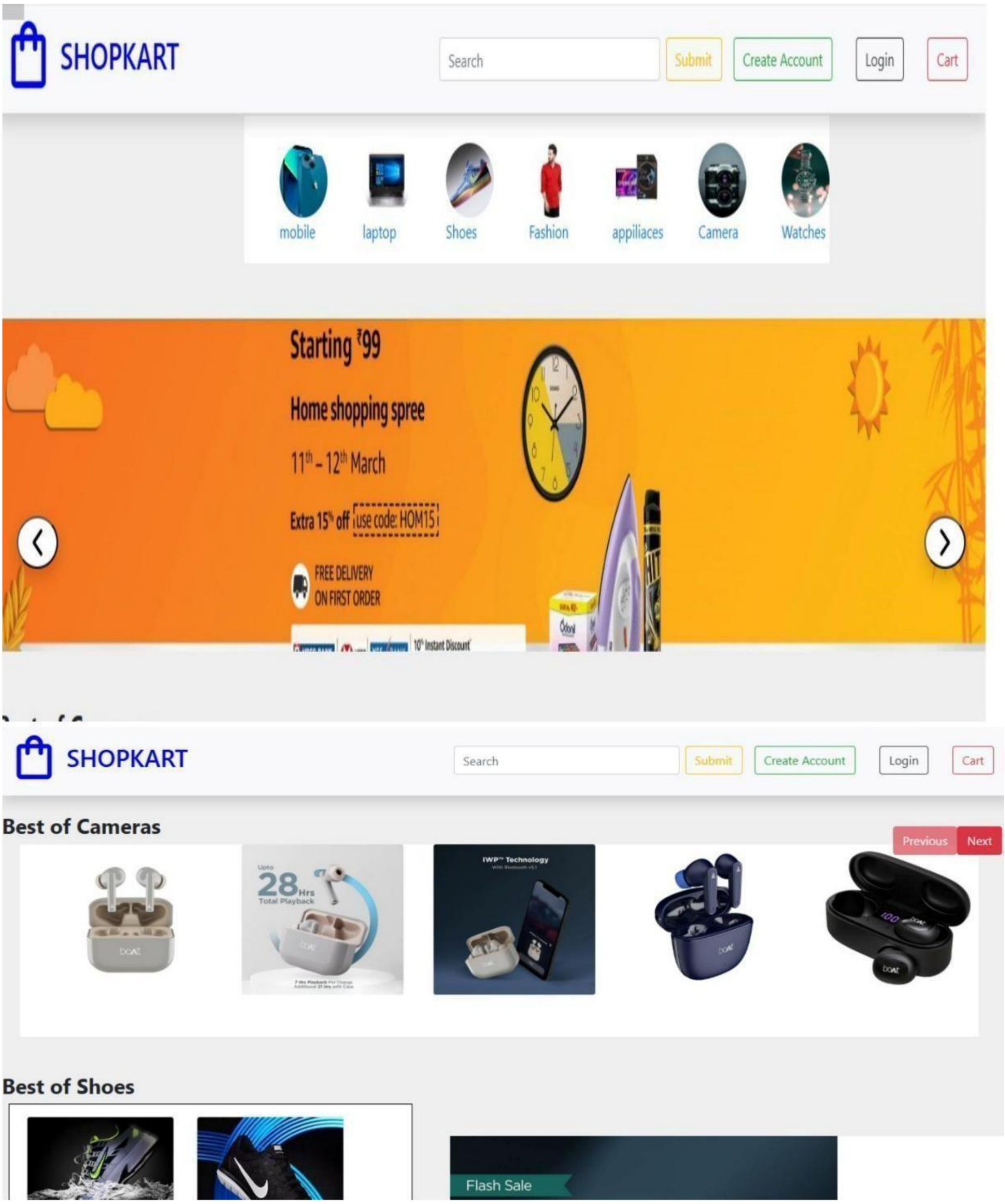
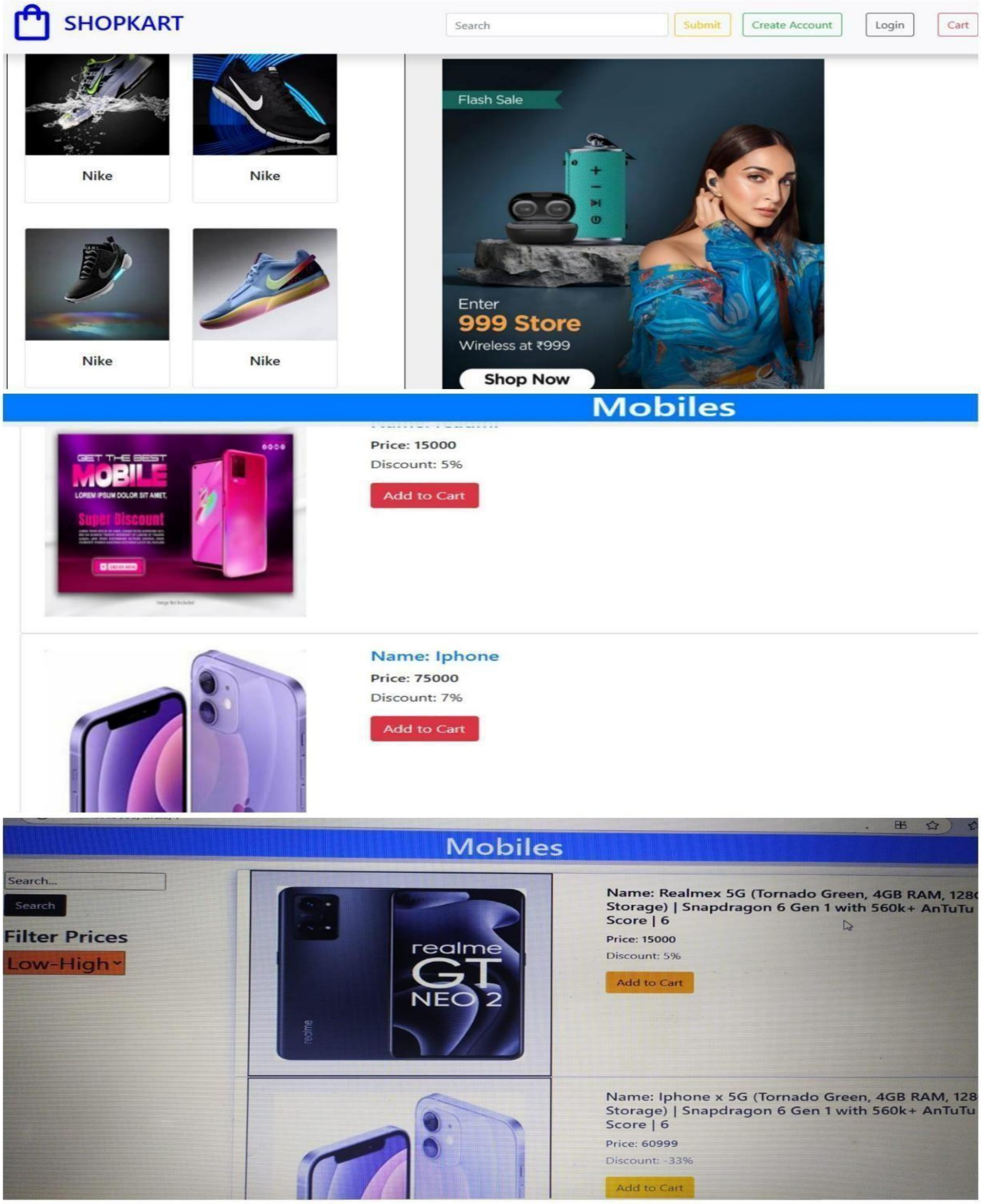


Fig:5.1.1



Price: 15000

Discount: 5%

Name: Iphone

Price: 75000

Discount: 7%

Name: Realmex 5G (Tornado Green, 4GB RAM, 128
Storage) | Snapdragon 6 Gen 1 with 560k+ AnTuTu
Score | 6

Price: 15000

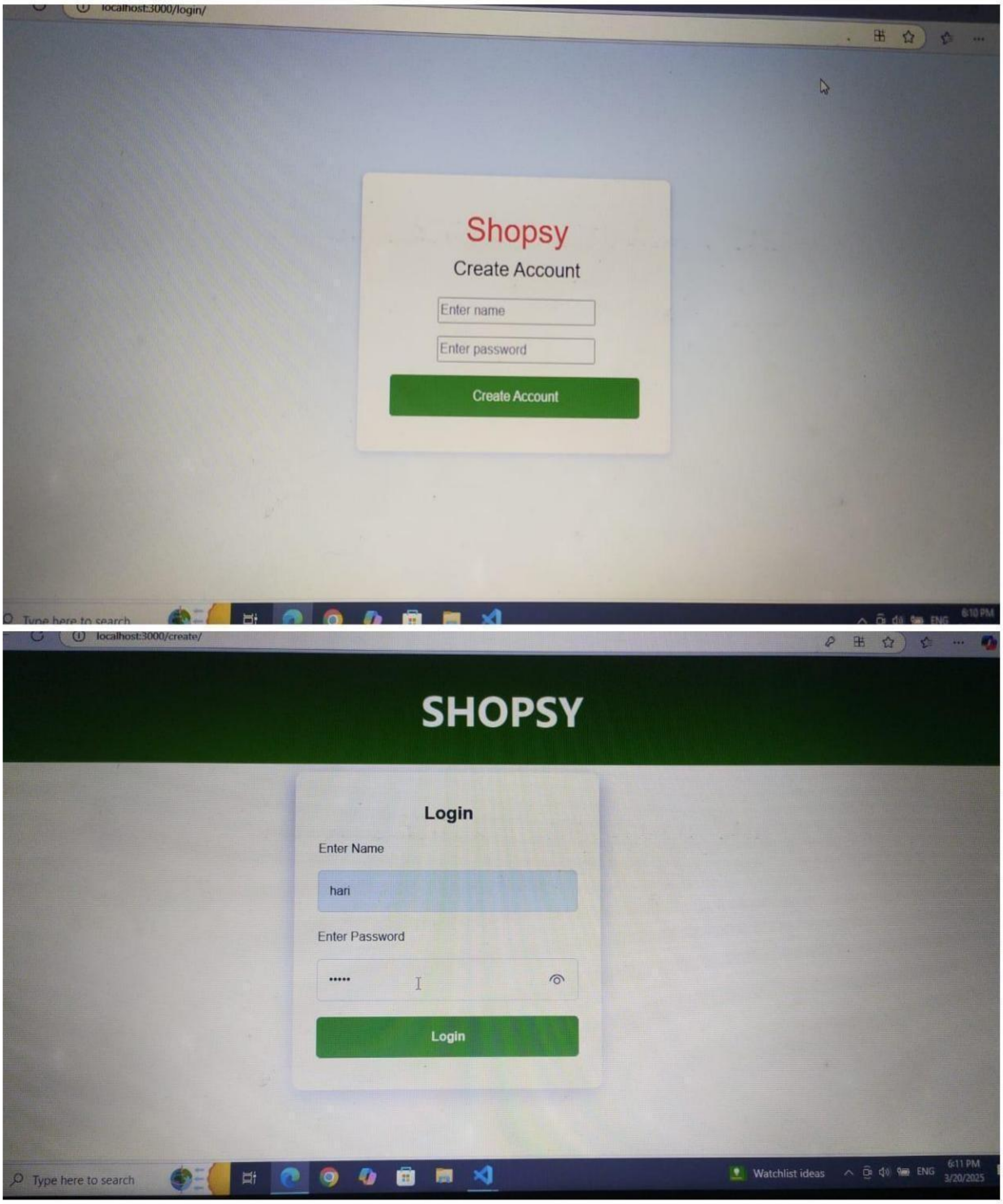
Discount: 5%

Name: Iphone x 5G (Tornado Green, 4GB RAM, 128
Storage) | Snapdragon 6 Gen 1 with 560k+ AnTuTu
Score | 6

Price: 60999

Discount: -33%

Fig:5.1.2



localhost:3000/login/

Shopsy

Create Account

Enter name

Enter password

localhost:3000/create/

SHOPSY

Login

Enter Name: hari

Enter Password: •••••

Fig:5.1.3

# 6. CONCLUSION

E-Commerce has transformed the way businesses operate and consumers shop, breaking geographical barriers and offering convenience and accessibility. It enables companies to reach global audiences while providing personalized shopping experiences through advancements like AI and secure payment systems. Despite its benefits, e-commerce faces challenges, including cybersecurity risks, intense competition, and logistical hurdles. To succeed, businesses must focus on customer trust, innovation, and efficient service delivery. Overall, e-commerce continues to redefine traditional business models, driving global economic growth and catering to the needs of modern digital consumers.

## References

1. Chaffey, D. (2015). *Digital Business and E-Commerce Management* (6th ed.). Pearson Education.
   → A comprehensive guide on eCommerce strategies, business models, and digital marketing.
2. Laudon, K. C., & Traver, C. G. (2021). *E-Commerce 2021: Business, Technology, Society* (16th ed.). Pearson.
   → Covers all foundational aspects of eCommerce including tech, legal, and societal impacts.
3. Statista. (2024). *E-commerce worldwide – statistics & facts.*
   Retrieved from: https://www.statista.com/topics/871/online-shopping/
   → Latest trends, stats, and data on global eCommerce performance.
4. Shopify. (2023). *How Gymshark grew from a UK startup to a global fitness brand.*
   Retrieved from: https://www.shopify.com/enterprise/gymshark-case- study
   → A practical example of eCommerce growth and branding strategy.