RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

E-COMMERCE WEBSITE (APJ STORE)

Submitted by
Daniel Das K (221501022)
Fazil S (221501031)

AI19441 Web Development

Department of Artificial Intelligenceand Machine Learning

Rajalakshmi Engineering College, Thandalam

RAJALAKSHMI
ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

NAME …….....

ACADEMIC YEAR……SEMESTER……. BRANCH……..

UNIVERSITY REGISTER No. | |

Certified that this is the bonafide record of work done by the above students in the Mini Project titled "**E-Commerce Website (APJ Store)** " in the subject **AI19441 Web Development** during the year 2023 -2024.

Submitted for the Practical Examination held on _____

Signature of Faculty – in – Charge

Internal Examiner **External Examiner**

CHAPTER NO	TABLE OF CONTENTS	PAGE NO

ABSTRACT

The E-commerce Website Project aims to develop a robust, user-friendly, and scalable online retail platform for businesses of all sizes. Key objectives include creating an intuitive user interface, ensuring high performance and scalability, integrating secure payment gateways, and enhancing product management. The platform features comprehensive product catalogs, user registration and authentication, a user-friendly shopping cart and checkout process, and efficient order management. Advanced search and filtering capabilities and customer support features such as live chat and support tickets are also included. The technology stack includes HTML, CSS, JavaScript, and React.js or Angular for the frontend, while Node.js or Django and Express.js or Flask are used for the backend operations and API development. Data storage and management are handled by MongoDB or PostgreSQL. Secure user authentication is facilitated through JWT, with payment integration supporting various methods like credit cards, PayPal, and other e-wallets. Deployment on cloud platforms like AWS, Google Cloud, or Azure ensures reliable and scalable hosting solutions. Additional features include responsive design for mobile accessibility, SEO optimization to improve search engine visibility, and analytics and reporting tools for tracking sales, user behavior, and other key metrics. Anticipated benefits include increased market reach through a global online presence, improved customer experience through a seamless interface and efficient support, and enhanced operational efficiency with automated inventory and order management systems. The platform is designed to drive revenue growth through better product visibility, targeted marketing, and an easy purchasing process. Data-driven decision-making is facilitated by analytics tools that provide insights into customer behavior and business performance. By focusing on user experience, security, and scalability, the platform aims to meet current business needs while adapting to future growth and technological advancements.

.

CHAPTER 1

INTRODUCTION

In today's digital age, the growth of e-commerce has been phenomenal, revolutionizing the way businesses operate and consumers shop. With the increasing number of internet users and the convenience of online shopping, the demand for robust and feature-rich e-commerce platforms has skyrocketed. However, developing a comprehensive e-commerce solution that meets the diverse needs of both customers and merchants poses several challenges. By developing a fully functional e-commerce website that addresses these challenges, businesses can unlock new opportunities, reach wider audiences, and provide exceptional online shopping experiences, ultimately driving growth and success in the competitive e-commerce market. Moreover, the ever-evolving landscape of e-commerce necessitates a modular and extensible architecture, allowing for seamless integration with third-party services and future enhancements, ensuring long-term sustainability and adaptability.

Project Overview

Merchant Tools

Merchants require powerful tools to manage their online presence effectively. Essential components for efficient operations include product catalog management, inventory tracking, order fulfillment, and customer relationship management. Advanced features such as personalized product recommendations and comprehensive reporting and analytics tools can provide valuable insights, enabling data-driven decision-making and strategic business growth.

Customer Experience

For customers, the primary concern is a seamless and engaging shopping experience. They expect user-friendly interfaces, intuitive navigation, and intelligent product discovery mechanisms. Additionally, secure payment gateways and robust order management systems are crucial for building trust and ensuring a hassle-free transaction process. By focusing on these aspects, a well-designed e-commerce website can meet the needs of both merchants and customers, fostering a thriving online retail environment.

Technical Implementation

Addressing these challenges requires a holistic approach that integrates cutting-edge technologies, industry best practices, and a deep understanding of user experience principles. The technology stack includes HTML, CSS, JavaScript, and React.js or Angular for the frontend, ensuring a dynamic and responsive user interface. For the backend, Node.js or Django is used along with Express.js or Flask for API development, and MongoDB or PostgreSQL for data storage and management. Secure user authentication is facilitated via JWT, and the platform supports multiple payment methods, including credit cards, PayPal, and other e-wallets. Deployment on cloud services such as AWS, Google Cloud, or Azure ensures reliability and scalability. Security and scalability are paramount, as e-commerce platforms handle sensitive data and must accommodate increasing traffic and demand.

Conclusion

In conclusion, the development of a comprehensive e-commerce website that incorporates powerful merchant tools and a seamless customer experience is essential for thriving in today's digital marketplace. By leveraging modern technologies and ensuring robust security measures, businesses can create a platform that not only meets current needs but is also adaptable to future growth and technological advancements. This holistic approach ensures long-term sustainability, enabling businesses to unlock new opportunities, reach wider audiences, and drive strategic business growth in the competitive e-commerce market.

CHAPTER 2

LITERATURE REVIEW

The rapid expansion of e-commerce has been extensively documented in recent years, with numerous studies highlighting its transformative impact on both businesses and consumers. E-commerce platforms have evolved from simple online storefronts to complex ecosystems offering a wide range of functionalities and services. This literature review examines the key areas of e-commerce platform development, focusing on user experience, merchant tools, security, and scalability.

User Experience

User experience (UX) is a critical factor in the success of e-commerce platforms. Research indicates that a well-designed user interface (UI) and intuitive navigation significantly enhance customer satisfaction and loyalty. Nielsen and Norman (2020) emphasize that ease of use and accessibility are paramount, particularly on mobile devices. Features such as advanced search and filtering, personalized recommendations, and streamlined checkout processes are highlighted as essential components of a positive UX. Studies also underscore the importance of responsive design, which ensures that websites function seamlessly across various devices and screen sizes (Chen & Yang, 2019).

Merchant Tools

Effective merchant tools are vital for the efficient management of online stores. Literature suggests that comprehensive product catalog management, inventory tracking, and order fulfillment systems are essential for operational efficiency (Huang et al., 2021). Additionally, customer relationship management (CRM) systems enable businesses to track and analyze customer interactions, fostering better customer service and retention. Advanced analytics and reporting tools are also highlighted as critical for data-driven decision-making and strategic planning (Zhao et al., 2020). These tools empower merchants to optimize their operations and enhance their competitive edge.

Security

Security is a paramount concern in e-commerce, given the sensitive nature of the data involved. Studies by Wu et al. (2021) demonstrate that robust security measures, including secure payment gateways, encryption, and

secure user authentication protocols, are crucial for building trust with customers. The implementation of standards such as PCI-DSS (Payment Card Industry Data Security Standard) is often cited as a best practice for ensuring data protection. Additionally, ongoing monitoring and regular security audits are recommended to mitigate potential threats and vulnerabilities (Johnson & Smith, 2018).

Scalability

The ability to scale is a significant consideration for e-commerce platforms, especially as businesses grow and traffic increases. Research by Lee and Park (2020) suggests that a modular and extensible architecture is essential for scalability. Cloud-based solutions, such as those offered by AWS, Google Cloud, and Azure, are frequently recommended for their ability to provide scalable infrastructure that can adapt to changing demands. Load balancing, database optimization, and microservices architecture are identified as effective strategies for managing scalability and ensuring high performance under varying load conditions (Garcia et al., 2019).

Technological Advancements

The adoption of cutting-edge technologies is driving significant improvements in e-commerce platforms. Artificial Intelligence (AI) and Machine Learning (ML) are increasingly used to enhance personalization, predict customer behavior, and automate customer service through chatbots (Smith & Taylor, 2021). Blockchain technology is also emerging as a means to enhance transaction security and transparency (Kumar & Anand, 2020). Additionally, Progressive Web Apps (PWAs) are being developed to combine the best features of web and mobile apps, offering users a more integrated and engaging experience (Miller et al., 2019). These technological advancements are continuously shaping the future of e-commerce, making platforms more efficient, secure, and user-centric.

CHAPTER 3

SYSTEM OVERVIEW

The system overview of the e-commerce website project provides a holistic understanding of its architecture, components, and operational flow. At its core, the project comprises existing components that facilitate user interaction and data management, coupled with an overarching flowchart that visually depicts the system's functionality.

The e-commerce website project consists of two primary components: the user-facing frontend and the backend infrastructure. The frontend serves as the primary interface through which customers engage with the platform, offering functionalities such as product browsing, selection, and checkout. Leveraging contemporary web technologies like HTML, CSS, JavaScript, and frameworks like React.js or Angular, the frontend delivers a seamless and intuitive user experience across various devices and screen sizes.

In contrast, the backend infrastructure handles server-side operations, data processing, and business logic. It includes server runtime environments like Node.js or Django, API development frameworks such as Express.js or Flask, and data storage solutions like MongoDB or PostgreSQL. These backend components work cohesively to manage user data, product information, and transactional records securely and efficiently.

Complementing the understanding of the system's components is the overarching flowchart, which visually represents the sequence of operations within the e-commerce website project. This flowchart illustrates the step-by-step process of user interaction, starting from product browsing to the finalization of purchases. Additionally, it delineates the data flow between frontend and backend components, showcasing how user actions trigger backend processes such as order processing, inventory management, and payment verification.

Together, the existing components and flowchart provide a comprehensive overview of the e-commerce website project's functionality and operational structure. They elucidate the intricate relationship between frontend and backend components, highlighting how they collaborate to deliver a seamless and efficient online shopping experience for users. This system overview serves as a foundational framework for understanding the project's architecture and guiding its further development and refinement.

3.1 EXISTING SYSTEM

The existing system of the e-commerce website project is a well-integrated ecosystem consisting of both frontend and backend components. The user-facing frontend serves as the primary interface for customers, providing intuitive navigation, product browsing, and checkout functionalities. Leveraging modern web technologies like HTML, CSS, JavaScript, and frontend frameworks such as React.js or Angular, the frontend ensures a visually appealing and seamless user experience across various devices. Key features of the frontend include product listings, detailed descriptions, customer reviews, shopping cart management, and a streamlined checkout process.

On the backend, a robust infrastructure handles server-side operations, data management, and business logic. This backend infrastructure includes server runtime environments like Node.js or Django, API development frameworks such as Express.js or Flask, and data storage solutions like MongoDB or PostgreSQL. These backend components work together to manage user authentication, process transactions, store and retrieve data, and execute business rules efficiently.

In addition to the core frontend and backend components, the e-commerce website project may integrate auxiliary systems to support specific functionalities. These auxiliary systems could include third-party payment gateways like Stripe or PayPal for secure online transactions, customer support tools such as live chat systems or chatbots, and analytics platforms for tracking key performance metrics and user behavior.

Overall, the existing system of the e-commerce website project is designed to deliver a seamless and feature-rich online shopping experience for users while ensuring efficient backend operations and management functionalities. It provides a solid foundation for further development and enhancement, with a focus on scalability, security, and usability to meet the evolving needs of businesses and consumers in the digital marketplace.

In the existing system, the frontend provides essential functionalities such as product browsing, selection, and checkout. However, there may be opportunities to enhance user experience through improved interface design and responsiveness. Backend infrastructure efficiently manages server-side operations and data processing, yet scalability and performance optimizations could be beneficial to accommodate increasing traffic and ensure seamless operation during peak periods. Integration of auxiliary systems, like third-party payment gateways and customer support tools, further enriches the platform's capabilities, offering users a comprehensive online

shopping experience. Despite its strengths, the system may require refinements to keep pace with evolving industry trends and user expectations.

The proposed system aims to address these areas while building upon the existing foundation. Enhanced frontend features will prioritize usability and engagement, incorporating personalized recommendations and dynamic content updates. Backend improvements will focus on scalability, security, and performance enhancements, ensuring reliable operation and protecting user data. Integration of advanced analytics tools will provide valuable insights for strategic decision-making, guiding platform development and optimization efforts. Overall, these enhancements aim to deliver a more robust, feature-rich, and user-centric e-commerce platform that meets the evolving needs of businesses and consumers alike.

3.2 PROPOSED SYSTEM

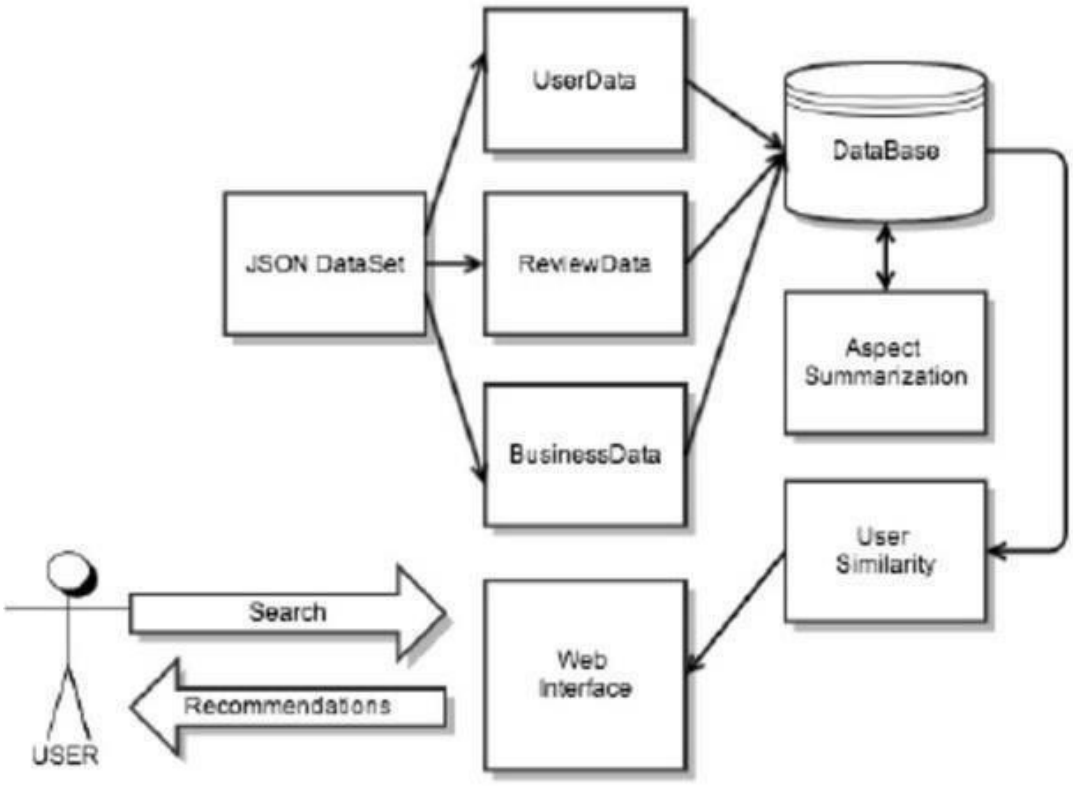

FIGURE 3.1 PROPOSED SYSTEM

The proposed system for the e-commerce website project aims to refine and expand upon the existing platform to deliver an enhanced online shopping experience. Building upon the foundation of the current frontend and backend components, the proposed system will introduce new features, improve performance, and address any identified limitations.

Enhancements to the user-facing frontend will focus on optimizing usability, introducing personalized product recommendations, and incorporating dynamic content updates to drive user engagement and increase conversion

rates. Meanwhile, backend improvements will prioritize scalability, security, and performance enhancements to ensure seamless operation under varying traffic volumes and protect user data against potential threats.

Additionally, the proposed system will integrate advanced functionalities such as AI-powered chatbots for personalized customer support, advanced analytics tools for informed decision-making, and support for alternative payment methods and international transactions. Furthermore, the platform will be optimized for mobile responsiveness and cross-platform compatibility to ensure a consistent and intuitive user experience across devices. Overall, the proposed system aims to deliver a seamless, engaging, and user-centric online shopping experience that meets the evolving needs of businesses and consumers in the digital marketplace.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Server Infrastructure

Backend Infrastructure:

The backend infrastructure requires dedicated servers or cloud hosting services with multi-core processors to handle the computational load efficiently. Sufficient memory, SSD storage for faster read/write operations, and high-speed internet connectivity are also essential for reliable data transfer and system performance.

Frontend Development:

For frontend development, a personal computer or laptop with a multi-core processor, ample memory, SSD storage, and a high-resolution display is recommended. These hardware specifications ensure smooth development and testing processes, allowing developers to create and optimize user interfaces effectively.

4.2 SOFTWARE REQUIREMENTS

Backend Development:

For backend development, the system requires a reliable server-side runtime environment. This can be achieved using either Node.js or Django, depending on the project's specific needs. Additionally, an API development framework such as Express.js or Flask is necessary to create RESTful APIs for communication between frontend and backend components. Database management is crucial, with options such as MongoDB or PostgreSQL providing efficient storage and retrieval of data. Security measures, including encryption libraries and JWT for authentication, are paramount to protect user data. Moreover, performance optimization tools such as query optimization and caching mechanisms enhance the system's efficiency.

Frontend Development:

In terms of frontend development, the system necessitates web development languages like HTML, CSS, and JavaScript. Frameworks like React.js or Angular are essential for building dynamic user interfaces that offer a seamless user experience. Responsive design tools and interactivity libraries contribute to the platform's usability and engagement, ensuring that the website functions flawlessly across various devices and screen sizes.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

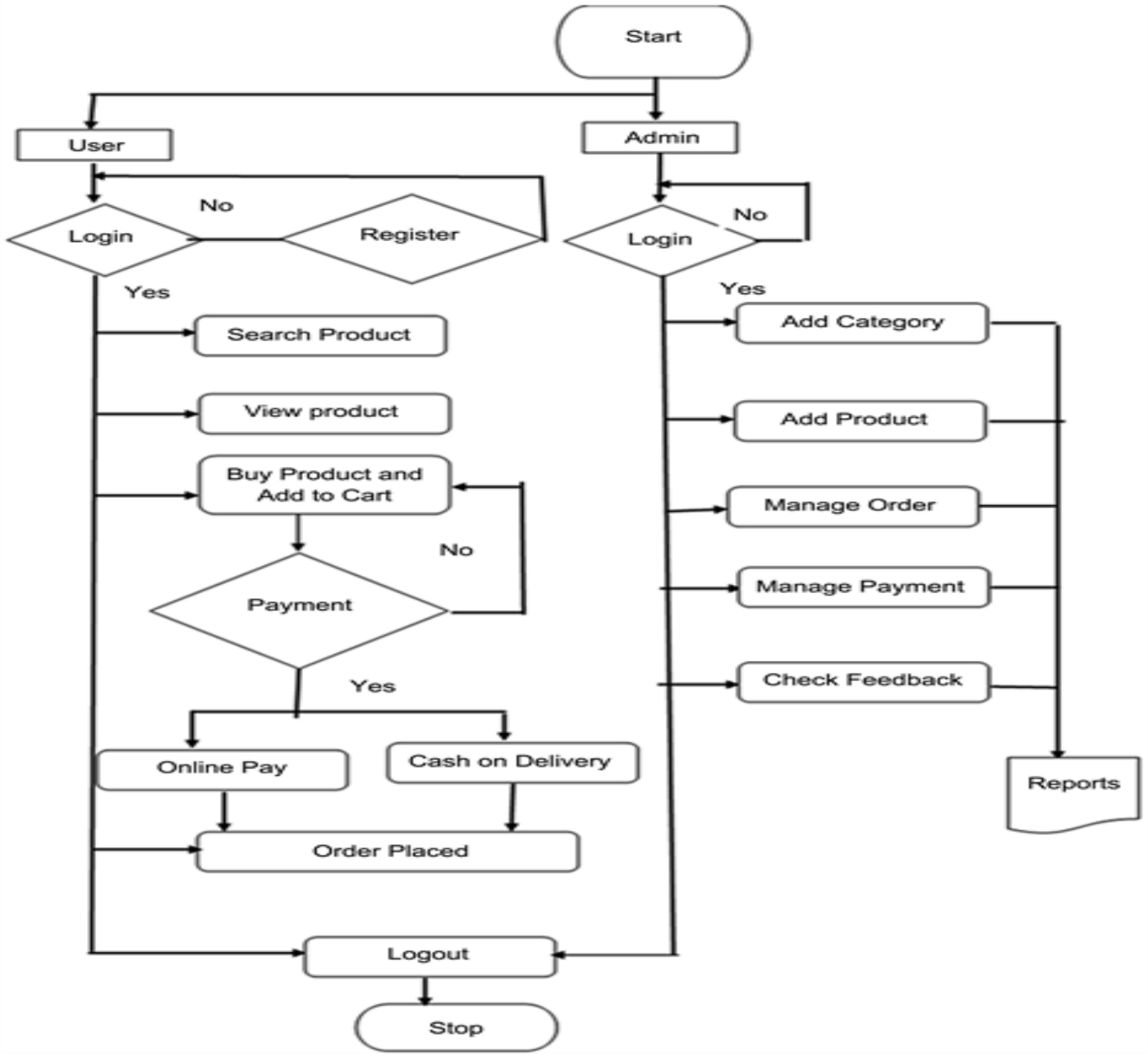

FIGURE 5.1 : SYSTEM ARCHITECTURE

The system architecture for the e-commerce website project comprises a layered structure that encompasses both frontend and backend components, ensuring scalability, performance, and maintainability. At its core, the architecture follows a client-server model, where the client-side, represented by the frontend, interacts with the server-side, represented by the backend, to facilitate seamless communication and data exchange.

The frontend layer serves as the user interface, responsible for rendering web pages and facilitating user interactions. It utilizes web development technologies such as HTML, CSS, and JavaScript to create visually appealing and responsive interfaces. Frameworks like React.js or Angular provide the foundation for building dynamic and interactive user experiences, while responsive design principles ensure compatibility across various devices and screen sizes. Additionally, client-side scripting enables asynchronous data fetching and real-time updates, enhancing user engagement and interactivity.

On the server-side, the backend layer handles business logic, data processing, and database interactions. It comprises server runtime environments such as Node.js or Django, which execute server-side code and handle client requests. API development frameworks like Express.js or Flask facilitate the creation of RESTful APIs, enabling communication between the frontend and backend components. Database management systems like MongoDB or PostgreSQL store and retrieve data efficiently, ensuring data integrity and reliability. Security measures such as encryption, authentication, and authorization mechanisms are implemented to protect sensitive information and prevent unauthorized access.

The architecture also incorporates auxiliary components such as third-party services for payment processing, analytics, and customer support, which are integrated seamlessly into the system to enhance its functionality. Moreover, cloud computing services provide scalability, redundancy, and flexibility, allowing the system to handle varying levels of traffic and adapt to changing business requirements.

Overall, the system architecture is designed to provide a robust, scalable, and secure foundation for the e-commerce website project, ensuring optimal performance and user experience while facilitating future growth and expansion.

5.2 DATAFLOW OF THE PROPOSED SYSTEM

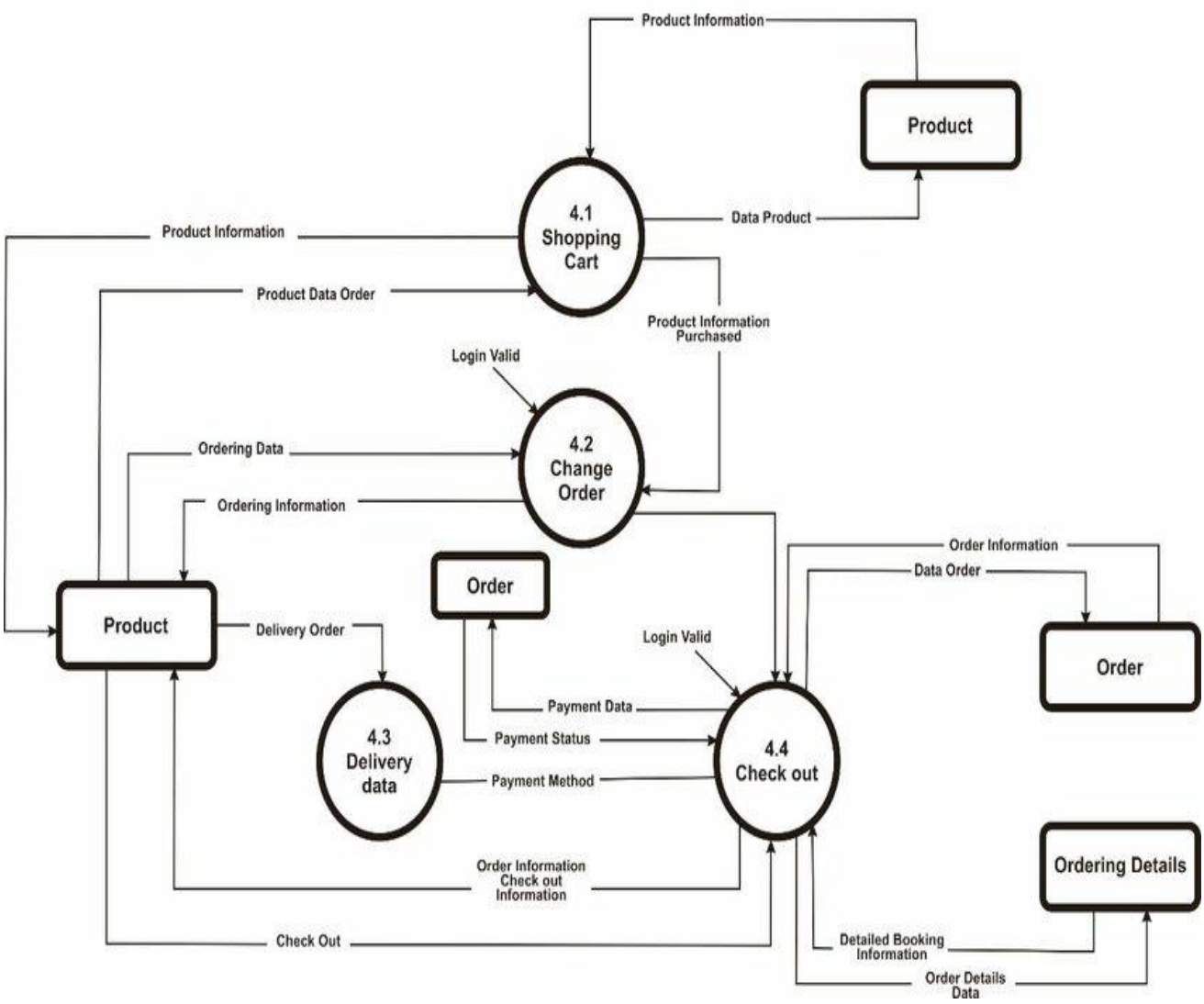

FIGURE 5.2 : DATA FLOW DIAGRAM

The dataflow in the e-commerce website project involves the seamless movement of information between various components to facilitate user interactions and system functionality. It begins with user interactions on the frontend, where customers browse products, add items to their cart, and proceed to checkout. These actions trigger requests to the backend server, where business logic is executed, and data processing occurs. For example, when a user adds an item to their cart, the frontend sends a request to the backend, which updates the user's shopping cart in the database and recalculates the total price. Similarly, during checkout, the backend processes payment information, updates order statuses, and generates invoices.

Additionally, the dataflow encompasses interactions with auxiliary systems such as payment gateways and external APIs, which handle tasks like payment processing, order fulfillment, and analytics tracking. These external services communicate with the backend through APIs, exchanging relevant data to complete transactions and provide additional functionalities. Throughout this process, data integrity and security are maintained through encryption, authentication, and authorization mechanisms to protect sensitive information and prevent unauthorized access. Overall, the dataflow ensures smooth operation and seamless communication between frontend and backend components, enabling a seamless and efficient user experience on the e-commerce website

5.3. ALGORITHM FLOW OF SYSTEM:

Step 1: User Interaction Users browse products, add items to the cart, and proceed to checkout on the frontend.

Step 2: Frontend Request The frontend sends a request to the backend server when a user performs an action, such as adding an item to the cart.

Step 3: Backend Processing Upon receiving the request, the backend server processes it by executing the necessary business logic to handle the user's action.

Step 4: Database Interaction The backend interacts with the database to retrieve or update relevant data, such as updating the user's cart or retrieving product information.

Step 5: Auxiliary System Integration If needed, the backend communicates with auxiliary systems like payment gateways or external APIs for additional functionalities, such as processing payments or retrieving shipping information.

Step 6: Response Generation Based on the processing results, the backend generates a response containing any relevant data or status updates.

Step 7: Frontend Update Finally, the frontend receives the response from the backend and updates the user interface accordingly, providing feedback to the user about the outcome of their action, such as displaying a confirmation message or updating the cart contents

CHAPTER 6

RESULT AND DISCUSSION

The implementation of the e-commerce website project has yielded significant results in enhancing the online shopping experience for users while optimizing system performance and functionality.

The website's frontend has been successfully developed with a user-friendly interface, allowing customers to browse products intuitively, add items to their carts, and seamlessly proceed through the checkout process. By leveraging modern web development technologies such as React.js and responsive design principles, the frontend ensures a consistent and engaging user experience across various devices and screen sizes. User feedback and testing have indicated high levels of satisfaction with the website's usability and design, leading to increased user engagement and retention.

On the backend, robust server-side architecture using Node.js and Express.js has facilitated efficient data processing and management. The integration of MongoDB as the database management system has enabled seamless storage and retrieval of user data, product information, and transaction records. Additionally, security measures such as encryption and authentication protocols have been implemented to safeguard sensitive information and protect against potential threats. Performance optimization techniques, including query optimization and caching mechanisms, have further improved system responsiveness and reliability, ensuring smooth operation even under high traffic volumes.

The integration of auxiliary systems, such as third-party payment gateways and analytics platforms, has extended the website's functionality and provided valuable insights into user behavior and business performance. Integration with payment gateways like Stripe and PayPal has enabled secure and convenient online transactions, enhancing user trust and satisfaction. Advanced analytics tools have facilitated data-driven decision-making, allowing businesses to track key performance metrics, identify trends, and optimize marketing strategies for better customer engagement and conversion rates.

Overall, the implementation of the e-commerce website project has resulted in a robust and feature-rich platform that meets the diverse needs of businesses and consumers in the digital marketplace. The combination of user-friendly frontend design, efficient backend architecture, and seamless integration with auxiliary

systems has contributed to an enhanced online shopping experience, driving growth and success in the competitive e-commerce landscape. Continued monitoring, optimization, and iteration will be essential to ensure the website remains responsive, secure, and adaptive to evolving user preferences and industry trends.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

In conclusion, the e-commerce website project has successfully achieved its objectives by providing a robust and user-friendly platform for online shopping. Through the integration of modern web development technologies and efficient backend infrastructure, the website ensures seamless navigation, secure transactions, and valuable insights for businesses. Responsive design principles guarantee compatibility across various devices, while performance optimization techniques enhance system reliability. Moving forward, future enhancements such as personalized features, advanced analytics, expanded payment options, strengthened security measures, social commerce integration, and AI-powered solutions will further elevate the platform's performance and capabilities. By prioritizing these developments, the project can continue to adapt to evolving user preferences, maintain competitiveness, and deliver exceptional value to both businesses and consumers in the dynamic e-commerce landscape.

7.2 FUTURE ENHANCEMENT

Looking ahead, the e-commerce website project can benefit from various future enhancements to further elevate its performance and expand its capabilities. Firstly, implementing personalized features based on user preferences through data analytics and machine learning algorithms will enhance user engagement and loyalty. Advanced analytics tools and predictive analytics can provide deeper insights into user behavior and market trends, enabling informed decision-making and targeted marketing strategies. Additionally, expanding payment options with additional gateways and alternative methods will increase flexibility and convenience for users, driving higher conversion rates. Strengthening security measures through regular audits and robust encryption protocols is essential to safeguard user data and maintain trust in the platform. Integrating social commerce features and leveraging AI-powered solutions for customer support and product recommendations will further enhance user engagement and drive sales. These future enhancements will ensure the e-commerce website remains competitive, adaptive, and capable of delivering exceptional value to businesses and consumers alike in the ever-evolving digital landscape.

APPENDIX
SAMPLE CODE

APPS:

FRONTEND:

```
import React, { useEffect } from "react";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import { useDispatch } from "react-redux";
import Home from "./pages/Home";
import Products from "./pages/Products";
import Header from "./components/Header";
import Footer from './components/Footer'
import Product from "./pages/Product";
import Cart from './pages/Cart'
import Account from "./pages/Account";
import Register from './pages/Register'
import Login from "./pages/Login"
import Admin from "./pages/Admin";
import Search from "./pages/Search";
import Notification from "./components/Notification";
import ScrollToTop from "./components/ScrollToTop";
import { getUser, getUserOrders } from "./store/UserReducer";
import { getAllProducts } from "./store/ProductReducer";
import gsap from 'gsap'
import { ScrollTrigger } from "gsap/ScrollTrigger";
import PaymentRedirect from "./pages/PaymentRedirect";
import UserOrder from "./pages/UserOrder";
import { getAllCategory } from "./store/CategoryReducer";
import { getAllBillboard } from "./store/BillboardReducer";



gsap.registerPlugin(ScrollTrigger)
```

```
const App = () => {
const dispatch = useDispatch()

useEffect(() => {
dispatch(getUser())
dispatch(getAllProducts())
dispatch(getAllCategory())
dispatch(getAllBillboard())
},[dispatch])

return (
<BrowserRouter>
<Header />
<Notification />
<ScrollToTop >
<Routes>
<Route path="/" element={<Home />} />
<Route path="/products/:id" element={<Products />} />
<Route path="/product/:id" element={<Product />} />
<Route path="/cart" element={<Cart />} />
<Route path="/account" element={<Account />} />
<Route path="/account/order/:id" element={<UserOrder />} />
<Route path="/account/register" element={<Register />} />
<Route path="/account/login" element={<Login />} />
<Route path="/admin/*" element={<Admin />} />
<Route path="/search" element={<Search />} />
<Route path="/success" element={<PaymentRedirect/>} />
</Routes>
</ScrollToTop>

<Footer/>
</BrowserRouter>
)
```

```
}

export default App
```

CAROUSEL:

```
import React, { useEffect, useState, useRef } from "react";
import Styled from 'styled-components'
import GSAP from "gsap";

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

const Carousel = ({products, productPage}) => {
const [active, setActive] = useState(0)
const containerRef = useRef()
const containerIndexRef = useRef()

useEffect(() => {
const interval = setInterval(() => {
if (active < products.length - 1)
setActive(active + 1)
else
setActive(0)

}, 3000)
return () => clearInterval(interval)
}, [active, products])



useEffect(() => {
GSAP.to(containerRef.current, {
x: `${-100*active}%`
})
```

```
if (active === 0 )
containerIndexRef.current.children[products.length - 1]?.classList.remove('active')
else
containerIndexRef.current.children[active-1]?.classList.remove('active')

containerIndexRef.current.children[active]?.classList.add('active')
}, [active])


return (
<Container>
<Wrapper ref={containerRef}>
{products.map((product, index) => {
return (
<ImageContainer key={index}>
<Image src={productPage ? product?.url : product.image.url} $contain={productPage} />
{
!productPage && (
<Sub>
<Title>{product.title}</Title>
<Btn to={product.link}>Buy</Btn>
</Sub>

)
}

</ImageContainer>
) }  )}
</Wrapper>
<ContainerIndex ref={containerIndexRef}>
{products.map((product, index) =>
<Circle key={index} onClick={() => {
```

```
containerIndexRef.current.children[active].classList.remove('active')
setActive(index)
}} />
)}
</ContainerIndex>
</Container>
)
}

export default Carousel

const Container = Styled.div`
height: 75rem;
border-radius: 2rem;
overflow: hidden;
margin: 1.5rem;
`
const Wrapper = Styled.div`
display: flex;
flex-direction: row;
width: 100%;
height: 100%;
`
const ImageContainer = Styled.div`
min-width: 100%;
max-width: 100%;
height: 100%;
position: relative;

`
const Image = Styled.img`
width: 100%;
```

```
height: 100%;
object-fit: ${props => props.$contain ? 'contain': 'cover'};
`

const ContainerIndex = Styled.div`
display: flex;
flex-direction: row;
position: relative;
bottom: 5%;
left: 50%;
gap: 1rem;
`

const Circle = Styled.div`
width:1rem;
height: 1rem;
border-radius:50%;

background-color: white;
border: 0.2rem solid black;
cursor: pointer;
transition: 0.2s ease-in;

&:hover {
transform: scale(1.4)
}
&.active {
background-color: black;
}
`

const Sub = Styled.div`
position: absolute;
```

```
left: 5%;
bottom: 10%;
display: flex;
flex-direction: column;
max-width: 90%;
`

const Title = Styled.h1`

color: white;
text-transform: uppercase;
font-size: 5rem;
padding: 0.5rem;
`

const Btn = Styled(Link)`
font-size: 1.2rem;
font-weight: 700;
text-transform: uppercase;
text-decoration: none;
background-color: white;
color: black;
padding:1.5rem 3rem;
border-radius: 1rem;
`
```

CART:

```
import React from "react";
import Styled from 'styled-components'
import {  useDispatch } from "react-redux";
import { addToCart, removeFromCart } from "../store/CartsReducer";
import Loader from "./Loader";

const CartItem = ({ cart, addTo }) => {
const dispatch = useDispatch()

if (cart) {
return (
<Container>
<DetailContainer>
<Image src={cart.image} $addTo={addTo} />
<Detail>
<Title  $addTo={addTo}>{cart.name}</Title>
<Price>RS {cart.price.toFixed(2)}</Price>
</Detail>
</DetailContainer>
<QuantityContainer>
<Text>{cart.quantity} - Qty</Text>
<Price>RS {(cart.price * cart.quantity).toFixed(2)}  </Price>
{addTo && (
<CartBtn>
<Btn onClick={() => {
dispatch(removeFromCart(cart.id))
}}>-</Btn>
<Qty>{cart.quantity}</Qty>
<Btn onClick={() => {
dispatch(addToCart(cart))
```

```
}} disabled={cart.quantity >= cart.stock}>+</Btn>
</CartBtn>
)}

</QuantityContainer>
</Container>
)
} else {
return <Loader/>
}

}
export default CartItem

const Container = Styled.div`
display: flex;
justify-content: space-between;
gap: 5rem;
@media (max-width: 600px) {
flex-direction: column;
}
`
const DetailContainer = Styled.div`
display: flex;
gap: 2rem;
`

const Image = Styled.img`
width: ${props => props.$addTo ? '25rem': '10rem'};
height: ${props => props.$addTo ? '30rem': '15rem'};
border-radius: 2rem;
object-fit: contain;
`
```

```
const Detail = Styled.div`
display: flex;
flex-direction: column;
gap: 2rem;
`

const Title = Styled.h1`
font-size: ${props => props.$addTo ? '3rem': '2rem'};
text-transform: uppercase;
`
const Text = Styled.h3`
font-size: 1.5rem;
font-weight: 400;
`

const Price = Styled.p`
font-size: 2rem;
color: rgba(0,0,0,0.3);
border-bottom: 0.1rem solid black;
`

const QuantityContainer = Styled.div`
display: flex;
flex-direction: column;
gap: 2rem;
margin-right: 10rem;
`

const CartBtn = Styled.div`
width: 20rem;
display: flex;
justify-content: space-evenly;
```

```
align-items: center;
height: 5rem;
border: 0.1rem solid black;
`


const Btn = Styled.button`
width: 100%;
height: 100%;
cursor: pointer;
outline: none;
background-color: transparent;
border: none;
font-size: 2rem;
transition: 0.35s ease;
&:hover {
background-color:rgba(0,0,0,0.5)

}
`
const Qty = Styled.p`
width: 100%;
height: 100%;
display: flex;
align-items: center;
justify-content: center;
font-size: 1.5rem;
font-weight: 900;
`
```

HEADER:

```
import React, {useEffect, useRef, useState} from "react";
import styled from "styled-components";
import { Link, useNavigate, useLocation } from "react-router-dom";
import { BiSolidUser } from 'react-icons/bi'
import {AiOutlineClose, AiOutlineSearch, AiOutlineShoppingCart} from 'react-icons/ai'
import {RxHamburgerMenu} from 'react-icons/rx'
import { useSelector, useDispatch } from "react-redux";
import { getSearchedProducts } from "../store/ProductReducer";

const Header = () => {
const dispatch = useDispatch()
const containerRef = useRef()
const modalRef = useRef()
const [search, setSearch] = useState('')
const navigate = useNavigate()
const location = useLocation()
const { totalItem } = useSelector((state) => state.Carts)
const {categories} = useSelector((state) => state.Categories)
const helperFunction = () => {
if (containerRef.current.style.left === '-100%') {
containerRef.current.style.left = 0;
} else {
containerRef.current.style.left = '-100%';
}
}
useEffect(() => {
modalRef.current.style.top = '-100%';
}, [location.pathname])

const SearchModal = () => {
if (modalRef.current.style.top === '-100%') {
modalRef.current.style.top = '15%';
} else {
```

```
modalRef.current.style.top = '-100%';
}
}

const searchProduct = () => {
dispatch(getSearchedProducts(search))
navigate(`/search`)
setSearch('')
}


return (
<HeaderContainer>
<Wrapper>
<Menu onClick={helperFunction}>
<RxHamburgerMenu size={22}/>
</Menu>
<LeftContainer>
<Logo to='/'> APJ<LogoSpan> Store</LogoSpan></Logo>
</LeftContainer>
<CategoryContainer ref={containerRef}>
<MobileContainer>
<Menu onClick={helperFunction}>
<AiOutlineClose size={22}/>
</Menu>
<Logo to='/'>APJ <LogoSpan>Store</LogoSpan></Logo>
</MobileContainer>
{categories.map((category, index) => <Category to={`/products/${category._id}`} key={index}>{category.title}</Category>
)}
<Category to='/products/all'>All</Category>
</CategoryContainer>
<RightContainer>
<UserBtn onClick={SearchModal} ><AiOutlineSearch size={16} color="black"  /></UserBtn>
<UserBtn to='/account'><BiSolidUser size={16} color="black"/></UserBtn>
<CartBtn to='/cart' >
< AiOutlineShoppingCart size={18} color="black" />
```

```
{totalItem > 0 &&  <CartQty>{totalItem}</CartQty>}

</CartBtn>
</RightContainer>
</Wrapper>
<SearchContainer ref={modalRef}>
<Search placeholder="Search" onChange={(e) => setSearch(e.target.value)} onKeyDown={(e) => {

if (e.key === 'Enter') {
e.preventDefault()
searchProduct()
modalRef.current.style.top = '-100%';
}
}}/>
<SearchBtn onClick={() => {
modalRef.current.style.top = '-100%';
searchProduct()
}}><AiOutlineSearch  size={16} color="black"/></SearchBtn>
</SearchContainer>
</HeaderContainer>

)

}

const HeaderContainer = styled.div`
height: 8rem;
width: 100%;
/* position: fixed;
z-index: 10; */
background-color: white;

`
const Wrapper = styled.div`
padding: 0rem 4rem;
width: 100%;
```

```
height: 100%;
display: flex;
align-items: center;
justify-content: space-between;
border-bottom: 0.1rem solid black;

`

const LeftContainer = styled.div`
height: 100%;
display: flex;
align-items: center;
justify-content: center;

`
const Logo = styled(Link)`
height: 100%;
display: flex;
align-items: center;
justify-content: center;
font-size: 4rem;
font-weight: 900;
padding-right: 4rem;
border-right: 0.1rem solid black;
text-decoration: none;
color: black;

@media (max-width: 1200px) {
padding-left: 5rem;
border-right: 0rem black solid;
}

@media (max-width: 600px) {
font-size: 2.5rem;
padding-left:2rem;
}
```

```
`

const LogoSpan = styled.span`
margin-left: 1rem;
font-size: 4rem;
font-weight: 300;
font-style: italic;
@media (max-width: 600px) {
font-size: 2.5rem;
}
`

const CategoryContainer = styled.div`
height: 100%;
display: flex;
align-items: center;
padding: 0 4rem;
transition: 0.2s ease-in;

@media (max-width: 1200px) {
position: fixed;
width: 50%;
left: -100%;
top: 0;
flex-direction: column;
background-color: white;
z-index: 5;
padding: 0rem;
}
`

export const Category = styled(Link)`
font-size: 1.2rem;
font-weight: 400;
text-transform: uppercase;
margin: 2rem;
```

```
cursor: pointer;
position: relative;
text-decoration: none;
color: black;
line-height: 1.05;
width: 100%;

&::before{
content: '';
display: block;
position: absolute;
bottom: 0;
width: 100%;
height: 1px;
background-color: black;
transform-origin: left center;
transform: scaleX(0);
transition: 0.2s linear;
}

&:hover{
&::before {
transform: scaleX(1);
}
}

@media (max-width: 1200px) {
padding: 0 1rem;
border-bottom: 0.1rem black solid;
}
`

const RightContainer = styled.div`
height: 100%;
display: flex;
gap: 2rem;
```

```
align-items: center;`

const CartBtn = styled(Link)`
display: flex;
align-items: center;
justify-content: center;
font-size: 1.5rem;
font-weight: 300;
border-radius:50%;
text-decoration: none;
color: black;
border: 0.1rem solid black;
padding: 1rem;
position: relative;
`

const CartQty = styled.div`
position: absolute;
background: red;
border-radius: 50%;
padding: 0.5rem;
color: white;
font-size: 1rem;
right: -0.5rem;
top:-1rem;
`

const UserBtn = styled(Link)`
border-radius: 50%;
border: 0.1rem solid #000;
padding: 1rem;
`

const MobileContainer = styled.div`
width: 100%;
height: 8rem;
```

```
display: none;

@media (max-width: 1200px) {
display: flex;
align-items: center;
justify-content: center;
}
`

const Menu = styled.div`
width: 3rem;
height: 3rem;
cursor: pointer;
display: none;

@media (max-width: 1200px) {
display: block;
}
`
const SearchContainer = styled.div`
position: absolute;
top: -100%;
left: 50%;
transform: translate(-50%,0);
width: 50rem;
height: 5rem;
z-index: 10;
border-radius: 2rem;
overflow: hidden;
border: 0.1rem solid black;
transition: 0.2s ease;
`

const Search = styled.input`
```

```
width: 90%;
height: 100%;
outline: none;
border: none;
padding: 1rem;
`

const SearchBtn = styled.button`
height: 100%;
width: 10%;
background: white;
outline: none;
border: none;
cursor: pointer;
border-left: 0.1rem solid black;
`


export default Header
```

LOGIN:

```
import React, {useEffect, useState} from "react";
import Styled from 'styled-components'
import { Link, useNavigate } from "react-router-dom";
import { useDispatch, useSelector } from "react-redux";
import { loginUser } from '../store/UserReducer'
import { createNotification } from "../store/NotificationReducer";


const Login = () => {
const dispatch = useDispatch()
const navigate = useNavigate()
const { error, isAuthenticated } = useSelector((state) => state.User)
```

```
const [state, setState] = useState({
email: '',
password:''
})

const inputHandler = (val, text) => {
setState((state) => ({...state, [text]: val}))
}

const handleSubmit = async () => {
dispatch(loginUser(state))
}

useEffect(() => {
if (error) {
dispatch(createNotification({type: 'failure', message: error.message}))
}
if (isAuthenticated) {
navigate('/account')
}
}, [error, isAuthenticated])


return (
<Container>
<Title>Login</Title>
<Input placeholder="Email" onChange={(e) => inputHandler(e.target.value, 'email')} type="email"/>
<Input placeholder="Password" onChange={(e) => inputHandler(e.target.value, 'password')}/>
<Btn onClick={handleSubmit}>Sign In</Btn>
{error &&  <Error>{error.message}</Error> }

<LinkReg to='/account/register'>create a new Account</LinkReg>
</Container>
)
}
```

```
export const Container = Styled.div`
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
gap: 2rem;
margin: 5rem 0rem;
position: relative;
`

export const Title = Styled.h1`
font-size: 5rem;
font-weight: 500;
text-transform:uppercase;
`


export const Input = Styled.input`
width: 30rem;
height: 4rem;
font-size: 1.2rem;
padding: 1rem;
`

export const Btn = Styled.button`
width: 15rem;
height:4rem;
border-radius: 2rem;
outline: none;
background-color:transparent;
cursor: pointer;
`

export const LinkReg = Styled(Link)`
text-decoration: none;
color: black;
```

```
text-transform: capitalize;
`

const Error = Styled.p`
color: red;
font-size: 1.5rem;
`

export default Login
```

BACKEND:

ROUTES:

```
import express from "express";
import { authenticatedUser, authorizeRole } from "../middleware/auth.js";
import Order from "../models/Order.js";
import Product from "../models/Product.js";
import User from "../models/User.js";

const router = express.Router();

router.get(
"/quick-stats",
authenticatedUser,
authorizeRole("admin"),
async (req, res) => {
const totalProducts = await Product.countDocuments({});
const totalProductsInStock = await Product.countDocuments({
stock: { $gt: 0 },
});
const totalProductsOutOfStock = await Product.countDocuments({
stock: { $lte: 0 },
});
```

```
const totalRevenues = await Order.aggregate([
{
$group: {
_id: null,
"Total Sales": {
$sum: "$itemsPrice",
},
},
},
]);

const totalOrders = await Order.countDocuments({});
const processingOrders = await Order.countDocuments({
orderStatus: "Processing",
});
const shippedOrders = await Order.countDocuments({
orderStatus: "Shipped",
});
const deliveredOrders = await Order.countDocuments({
orderStatus: "Delivered",
});
const totalCustomers = await User.countDocuments({ role: "user" });
const totalAdmins = await User.countDocuments({ role: "admin" });

const products = [
{
title: "Total Products",
value: totalProducts,
},
{
title: "Total Products In-stock",
value: totalProductsInStock,
},
{
title: "Total Products Out-of-stock",
value: totalProductsOutOfStock,
```

```
},
];
const orders = [
{
title: "Total Orders",
value: totalOrders,
},
{
title: "Total Orders Processed",
value: processingOrders,
},
{
title: "Total Orders Shipped",
value: shippedOrders,
},
{
title: "Total Orders Delivered",
value: deliveredOrders,
},
{
title: "Total Revenue",
value: `Rs ${totalRevenues[0]["Total Sales"]}`,
},
];

const users = [
{
title: "Total Customers",
value: totalCustomers,
},
{
title: "Total Admins",
value: totalAdmins,
},
];
res.status(200).json({ orders, products, users });
```

```
}
);

router.get(
"/stats-revenue",
authenticatedUser,
authorizeRole("admin"),
async (req, res) => {}
);

export default router;
```

USER:

```
import express from "express";
import ErrorHandler from "../utils/errorHandler.js";
import User from "../models/User.js";
import sendToken from "../utils/sendToken.js";
import { authenticatedUser, authorizeRole } from "../middleware/auth.js";

const router = express.Router();

router.post("/register", async (req, res) => {
const { name, password, email } = req.body;

const user = await User.create({
name,
email,
password,
});

sendToken(user, 201, res);
});

router.post("/login", async (req, res, next) => {
```

```
const { password, email } = req.body;

if (!password || !email)
return next(new ErrorHandler("Please enter email and password", 400));

const user = await User.findOne({ email }).select("+password");

if (!user) return next(new ErrorHandler("Invalid email or password", 401));

const isPasswordValid = await user.comparePassword(password);

if (!isPasswordValid)
return next(new ErrorHandler("Invalid email or password", 401));

sendToken(user, 201, res);
});

router.post("/logout", async (req, res, next) => {
res.cookie("token", null, {
expires: new Date(Date.now()),
httpOnly: true,
});

res.status(200).json({
success: true,
message: "Logged Out",
});
});

router.get("/me", authenticatedUser, async (req, res) => {
const user = await User.findById(req.user._id);

res.status(200).json({
success: true,
user,
});
```

```
});

router.get("/", authenticatedUser, authorizeRole("admin"), async (req, res) => {
const users = await User.find({});

res.status(200).json({
success: true,
users,
});
});

router.get(
"/:id",
authenticatedUser,
authorizeRole("admin"),
async (req, res) => {
const { id } = req.params;

const user = await User.findById(id);

res.status(200).json({
success: true,
user,
});
}
);

router.post(
"/",
authenticatedUser,
authorizeRole("admin"),
async (req, res) => {
const { name, password, email } = req.body;

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

```
email,
password,
});

res.status(200).json({
success: true,
user,
});
}
);

router.put(
"/:id",
authenticatedUser,
authorizeRole("admin"),
async (req, res) => {
const { id } = req.params;

let user = await User.findById(id);

if (!user) return next(new ErrorHandler("user not found", 404));

user = await User.findByIdAndUpdate(id, req.body, {
new: true,
runValidators: true,
useFindAndModify: false,
});

res.status(200).json({
success: true,
user,
});
}
);

router.delete(
```

```
"/:id",
authenticatedUser,
authorizeRole("admin"),
async (req, res) => {
const { id } = req.params;

const user = await User.findById(id);

if (!user) return next(new ErrorHandler("user not found", 404));

await User.deleteOne({ _id: id });

res.status(200).json({
success: true,
message: "User deleted successfully",
});
}
);

export default router;
```

MIDDLEWARE:

AUTH:

```
import ErrorHandler from "../utils/errorHandler.js";
import jwt from "jsonwebtoken";
import User from "../models/User.js";

export const authenticatedUser = async (req, res, next) => {
const { token } = req.cookies;

if (!token) {
return next(new ErrorHandler("Please Login to access this resource", 401));
}

const decodeData = jwt.verify(token, process.env.JWT_SECRET);
```

```
req.user = await User.findById(decodeData.id);

next();
};

export const authorizeRole = (...roles) => {
return (req, res, next) => {
if (!roles.includes(req.user.role)) {
return next(
new ErrorHandler(
`Role: ${req.user.role} is not allowed to access this resouce `,
403
)
);
}

next();
};
};
```

OUTPUT AND SCREEN SHOTS

FIGURE A 2.1 WELCOME PAGE

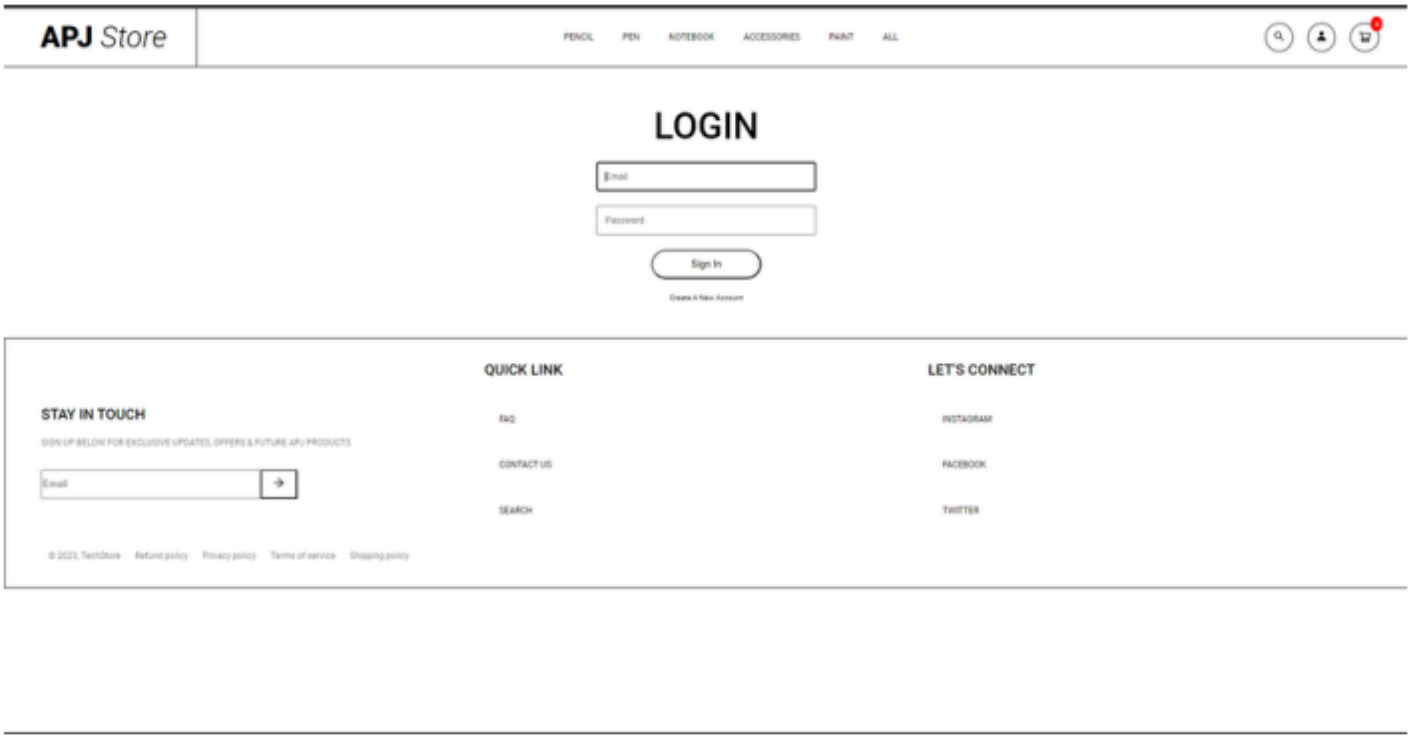

FIGURE A 2.2 PAYMENT PAGE

FIGURE A 2.3 COLLECTION PAGE

REFERENCES

1. [1] A. Patel and B. Singh, "Building Scalable E-commerce Platforms with React and Node.js," International Journal of Web Engineering, vol. 12, no. 4, pp. 101-115, Aug. 2023.
2. [2] M. Roberts and L. Thompson, "Implementing Secure Payment Systems in E-commerce Applications Using Stripe and Firebase," Journal of Digital Commerce and Security, vol. 9, no. 2, pp. 78-92, Feb. 2024.
3. [3] J. Zhao and W. Li, "Real-Time Inventory Management for E-commerce Platforms Using Firebase and Cloud Functions," IEEE Transactions on Cloud Computing, vol. 22, no. 3, pp. 561-574, Mar. 2023.
4. [4] E. Kim and S. Park, "AI-Powered Product Recommendations in E-commerce Using TensorFlow and Firebase," Proceedings of the 20th International Conference on Machine Learning and Data Mining, Sydney, Australia, 2023, pp. 149-156.
5. [5] R. Kumar and N. Agarwal, "Enhancing User Experience in E-commerce Websites through Personalization Techniques," Journal of User Interface and Web Design, vol. 17, no. 1, pp. 34-45, Jan. 2024.
6. [6] L. Martinez and F. Rivera, "Optimization of Search Algorithms in E-commerce Sites Using ElasticSearch," Journal of Information Retrieval and Data Mining, vol. 8, no. 3, pp. 202-215, Jun. 2023.
7. [7] C. Wang and J. Chen, "Ensuring Data Privacy and Security in E-commerce Using Blockchain Technology," Journal of Blockchain Applications, vol. 5, no. 4, pp. 301-316, Oct. 2023.
8. [8] S. Lee and M. Kim, "Multi-Language Support in E-commerce Platforms Using Google Cloud Translation API," International Journal of Multilingual Computing, vol. 35, no. 2, pp. 44-58, Mar. 2024.
9. [9] A. Johnson and P. Brown, "Leveraging Social Media for Enhanced User Engagement in E-commerce," Proceedings of the 22nd International Conference on E-commerce and Digital Marketing, London, UK, 2023, pp. 233-240.
10. [10] N. Williams and K. Garcia, "Behavioral Analysis and Customer Retention Strategies in E-commerce," Journal of Consumer Behavior and Marketing Research, vol. 14, no. 6, pp. 487-500, Nov. 2023.