

A PROJECT REPORT
ON
E-Commerce Website

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY
(ELECTRONICS AND COMMUNICATION ENGINEERING)

Submitted by:-
Avinash Kumar

University roll no.:-
1903711

DAV INSTITUTE OF ENGINEERING AND TECHNOLOGY

June 2023



IKG PUNJAB TECHNICAL UNIVERSITY
KAPURTHALA, PUNJAB

DAV Institute of Engineering and Technology Jalandhar

Abstract

React: React is an open-source JavaScript library used for building user interfaces (UIs) and UI components. It was developed by Facebook and has gained significant popularity in the web development community. React follows a component-based architecture, where UIs are built using reusable components that manage their own state and efficiently update the UI when the underlying data changes. React provides a declarative syntax and uses a virtual DOM (Document Object Model) to efficiently update the actual DOM, resulting in better performance.

Abstract: "Abstract" is a general term used to refer to a concept or idea that represents something in a simplified or generalized manner. In the context of software development, an abstract concept typically refers to a high-level representation or generalization of a specific implementation. Abstract classes and interfaces are examples of abstract concepts in object-oriented programming languages like Java or C#. These constructs define a set of methods or properties that must be implemented by their derived classes, providing a blueprint for creating objects. However, it's important to note that JavaScript, the language React is built with, doesn't have the concept of abstract classes or interfaces.

Acknowledgement

We would like to express our deepest gratitude and appreciation to everyone involved in the development and launch of our React-based e-commerce website.

First and foremost, we extend our heartfelt thanks to our development team for their exceptional skills, dedication, and hard work throughout the entire project. Their expertise in React and their ability to tackle complex challenges were instrumental in bringing our vision to life. Without their tireless efforts, this website wouldn't have been possible.

We are also immensely grateful to our design team for their creative input and meticulous attention to detail. Their innovative designs and user-friendly interfaces have enhanced the overall shopping experience on our website, ensuring a visually appealing and intuitive platform for our customers.

Our sincere appreciation goes out to our project managers and coordinators for their exceptional organizational skills and commitment. Their ability to effectively communicate, plan, and prioritize tasks played a crucial role in ensuring the project's smooth execution and timely delivery.

We are incredibly proud of what we have accomplished together, and we look forward to the continued success and growth of our e-commerce platform.

Thank you once again to everyone involved. Your contributions have made this project a resounding success.

Sincerely,

[Dav Institute of Engineering & Technology]

TABLE OF CONTENT

S.N NO	Contents	
2	Introduction	5
3	Objective	6
4	Project Category	7-9
5	Hardware and Software Requirement	9
6	Software Requirement Specification	10-13
7	Problem and Requirement Specification	14-15
8	Project Planning and Scheduling	16-18
9	Analysis	19-20
10	Modules of the System	21-22
11	System Flow Chart	23
12	Coding	24-30
13	Output Of Project	31-36

Introduction

The purpose of this project report is to provide an overview of the development and implementation of our React-based e-commerce website. This report will cover the project's objectives, scope, key features, technologies used, challenges faced, and future recommendations.

One of the key advantages of e-commerce shopping websites is the wide range of products and services they offer. From clothing and electronics to groceries and furniture, these platforms provide an extensive catalog that caters to diverse consumer needs. Shoppers can easily search for specific items, filter their results based on various criteria, and find the best deals without having to physically visit multiple stores.

Moreover, e-commerce websites provide a seamless and personalized shopping experience. They utilize sophisticated algorithms to analyze customer preferences and behavior, allowing them to recommend relevant products and provide a tailored user interface. Additionally, these platforms often offer features such as customer reviews, detailed product descriptions, and multiple images, enabling shoppers to make informed decisions and have a clear understanding of what they are purchasing.

Another significant advantage of e-commerce shopping websites is the convenience they offer. With just a few clicks, consumers can complete their purchases and have their items delivered to their doorstep. This eliminates the need to spend time and effort commuting to physical stores, searching for parking, and waiting in long checkout lines. E-commerce websites also provide flexible payment options, including credit cards, digital wallets, and even cash on delivery, making transactions hassle-free.

In addition to benefiting consumers, e-commerce shopping websites also provide opportunities for businesses. Small and large retailers alike can reach a global audience, expand their customer base, and increase their sales revenue through online platforms. These websites often offer tools and analytics that help businesses track their performance, manage inventory, and optimize their marketing strategies.

Objectives

- Develop a robust and user-friendly e-commerce website using React as the frontend framework.
- Create an intuitive and seamless shopping experience for customers.
- Implement essential features such as product listing, search functionality, shopping cart, user authentication, and payment integration.
- Optimize the website for performance, scalability, and security.
- Launch the website within the specified timeline and budget.
- The primary objective of the React e-commerce website is to provide a robust and efficient online shopping solution that caters to the needs of both customers and business owners. By harnessing the capabilities of React, the website delivers a smooth and dynamic user experience, enhancing engagement and driving customer satisfaction.

Project Category

1. Technologies Used:

- React: JavaScript library for building user interfaces.
- HTML/CSS/JavaScript: Core web technologies for structuring, styling, and adding interactivity to the application.
- Strapi: NoSQL database for storing product information, user data, and order details.
- RESTful APIs: Integration with external services such as payment gateways and shipping providers.
- Responsive Web Design: Techniques and frameworks to ensure optimal user experience across different devices.

2. Key Features:

- Product catalog with detailed product information, images, and customer reviews.
- Search functionality with filtering and sorting options for efficient product discovery.
- Shopping cart management for adding, removing, and updating items.
- Secure payment processing and integration with popular payment gateways.
- Order management system for tracking and managing customer orders.
- Responsive design to ensure seamless user experience across desktop and mobile devices.
- Admin dashboard for managing products, inventory, and user data.

3. Challenges:

- Handling complex product catalog structures and efficient data retrieval.
- Implementing secure user authentication and data protection measures.
- Integrating with external APIs and ensuring seamless communication.
- Optimizing performance and scalability to handle high traffic and large datasets.

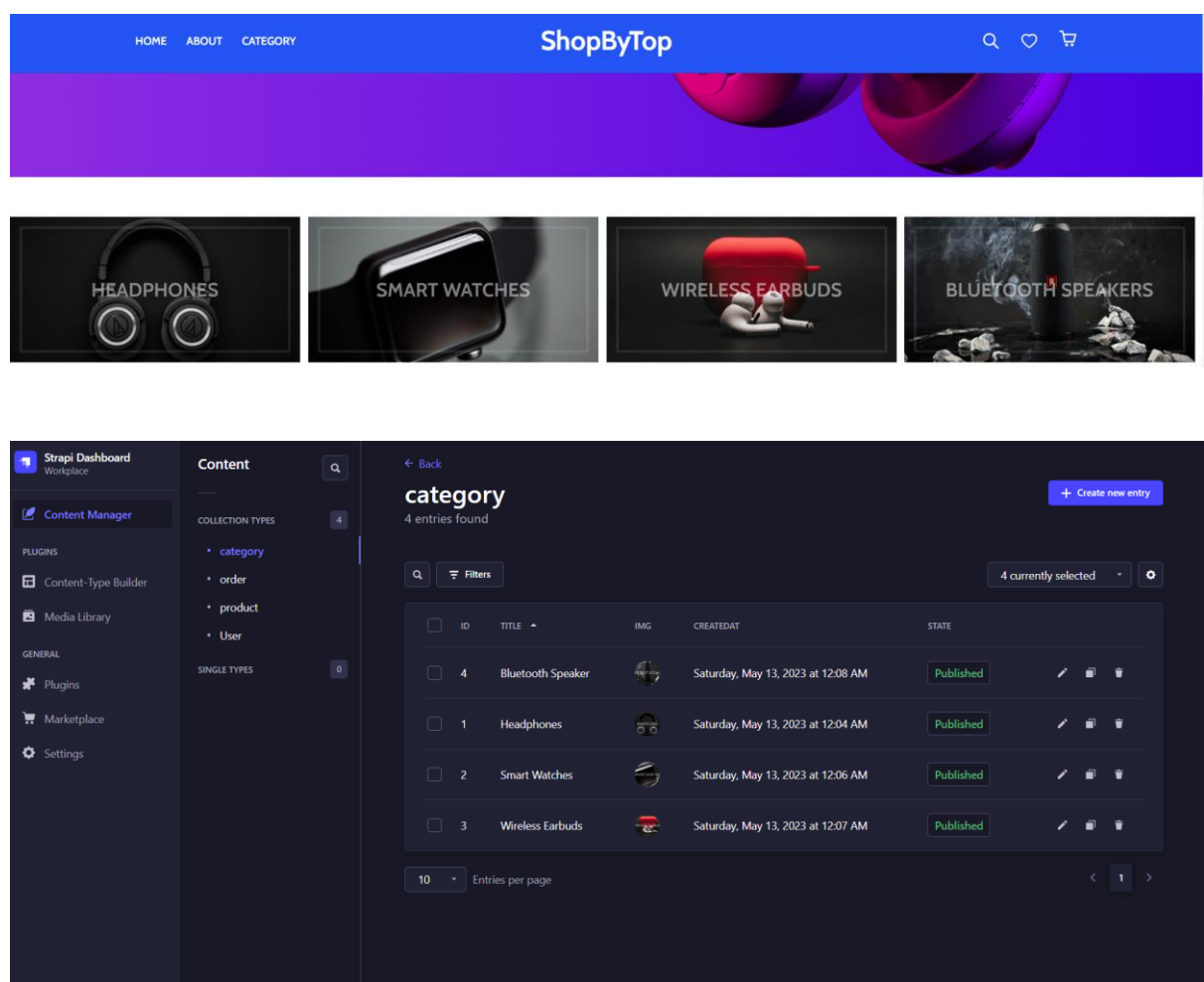
4. Conclusion:

The e-commerce application developed using React is aimed at providing a comprehensive platform for online buying and selling. By leveraging the power of React and integrating essential e-commerce functionalities, The project scope included the following key components:

5. Frontend development using React, HTML, CSS, and JavaScript.

6. Backend integration for data management and processing.
7. Integration with third-party APIs for payment gateway and shipping services.
8. Responsive web design for optimal user experience across various devices.
9. Implementation of user authentication and authorization.
10. Testing, bug fixing, and quality assurance.

User Interface Of Category



Hardware and Software Requirement of react website

Hardware and software requirements for a React website can vary depending on the specific project's scale and complexity. However, here are some general hardware and software requirements to consider:

Hardware Requirements:

Computer: A desktop or laptop computer capable of running modern web browsers and development tools.

Processor: A multi-core processor (e.g., Intel Core i5 or higher) for efficient development and testing.

Memory (RAM): At least 8GB of RAM to handle the memory-intensive tasks of development environments and browser testing.

Storage: Adequate storage capacity to accommodate the project files, dependencies, and related tools.

Display: A monitor with a suitable resolution for comfortable coding and design work.

Internet Connection: A reliable internet connection to access documentation, resources, and collaborate with team members.'

Software Requirements:

Operating System: The choice of operating system depends on personal preference and development environment compatibility. Common options include Windows, macOS, or Linux.

Code Editor or Integrated Development Environment (IDE): Popular choices include Visual Studio Code, WebStorm, or Atom, offering features like syntax highlighting, code completion, and debugging capabilities.

Node.js: A JavaScript runtime environment required to execute JavaScript on the server-side. Install the latest stable version of Node.js (<https://nodejs.org>).

Package Manager: npm (Node Package Manager) is usually installed with Node.js and allows easy installation and management of project dependencies.

Version Control System: Git is commonly used for version control, allowing team collaboration, code tracking, and managing different code versions.

Web Browser: Modern web browsers like Google Chrome, Mozilla Firefox, or Safari are necessary for testing and debugging the website.

React and Supporting Libraries: Install React and related dependencies using npm. Additionally, you may need tools like Webpack or Babel for building and transpiling React code.

Backend Technologies: If your React website requires server-side functionality, you may need to install additional software like Express.js or a database management system such as MongoDB.

Software Requirement Specification (SRS) for a React Website

1. Introduction: The Software Requirement Specification (SRS) outlines the functional and non-functional requirements for the development of a React website. It provides a comprehensive description of the software system, its purpose, scope, and the specific features and functionalities it should possess.
2. Purpose: The purpose of the React website is to create an interactive and user-friendly online platform for e-commerce or any other specific application. The website will utilize React as the frontend framework and may integrate with backend services for data management, authentication, and other required functionalities.
3. Scope: The scope of the project includes the following:
 - Development of a React-based website.
 - Implementation of core features such as product listing, search functionality, user authentication, shopping cart, payment integration, and order management.
 - Integration with backend services and APIs, if applicable.
 - Responsive design to ensure optimal user experience across different devices.
 - Compliance with security and performance standards.
4. Functional Requirements:
 - 4.1. User Management: - User registration and authentication. - User profile management.
 - 4.2. Product Catalog: - Displaying products with details, images, and pricing. - Categorization and filtering of products. - Search functionality.
 - 4.3. Shopping Cart: - Adding, removing, and updating items in the cart. - Calculating totals and applying discounts. - Proceeding to checkout.
 - 4.4. Payment Integration: - Integration with payment gateway(s) for secure transactions. - Handling different payment methods.
 - 4.5. Order Management: - Order placement and confirmation. - Tracking orders and providing status updates. - Order history and management for users and administrators.
 - 4.6. Admin Dashboard: - Managing products, categories, and inventory. - User and order management.
 - 4.7. Responsive Design: - Adapting the website layout for optimal viewing on various devices. - Ensuring usability and readability.
5. Non-functional Requirements:
 - 5.1. Performance: - Fast and responsive website with quick page load times. - Efficient data retrieval and processing.
 - 5.2. Security: - Secure user authentication and password storage. - Encryption of sensitive data. - Protection against common web vulnerabilities.
 - 5.3. Scalability: - Ability to handle increasing user traffic and data growth. - Scalable architecture and database design.
 - 5.4. Usability: -

Intuitive user interface with clear navigation and ease of use. - Consistent design and visual appeal. 5.5. Compatibility: - Compatibility with modern web browsers (e.g., Chrome, Firefox, Safari). - Responsive design for different screen sizes and resolutions.

5.6. Documentation: - Comprehensive documentation for developers and administrators.

6. Constraints:

- Compatibility with existing backend systems or APIs.
- Compliance with industry regulations and standards.
- Budget and time constraints.

7. Assumptions and Dependencies:

- Availability of necessary hardware and software resources.
- Availability of APIs or third-party services for specific functionalities (payment gateway, shipping, etc.).

The Software Requirement Specification serves as a guide for the development team, ensuring a clear understanding of the project requirements and facilitating effective implementation and testing. It provides a foundation for communication between stakeholders, developers, and other project participants throughout the development lifecycle.

Problem and Requirement Specification

1. Problem Statement: The problem is the absence of an efficient and user-friendly online platform to facilitate e-commerce transactions and provide a seamless shopping experience. Existing websites may lack responsiveness, interactivity, and modern features, leading to a subpar user experience and decreased customer satisfaction. This a need for a React website that addresses these issues and fulfills the requirements of a successful e-commerce platform.
2. User Requirements:
 - 2.1. User-Friendly Interface: - The website should have an intuitive and visually appealing interface that is easy to navigate. - Users should be able to browse products, search for specific items, and view detailed product information.
 - 2.2. Seamless Shopping Experience: - Users should be able to add products to a shopping cart, manage quantities, and proceed to checkout effortlessly. - The checkout process should be secure, easy to understand, and support various payment methods.
 - 2.3. Personalization: - Users should be able to create accounts, log in securely, and access personalized features such as order history, wishlists, and saved payment methods.
 - 2.4. Mobile Responsiveness: - The website should be responsive and adapt well to different screen sizes, ensuring a seamless experience on mobile devices.
3. Functional Requirements:
 - 3.1. Product Catalog: - The website should have a comprehensive product catalog with categories, subcategories, and filtering options. - Products should be displayed with images, descriptions, pricing, and customer reviews.
 - 3.2. Search Functionality: - The website should provide a robust search feature to allow users to find specific products based on keywords, filters, or attributes.
 - 3.3. Shopping Cart: - Users should be able to add products to a shopping cart, update quantities, remove items, and view a summary of their cart.
 - 3.4. Secure Checkout: - The website should integrate with secure payment gateways to ensure the confidentiality of users' sensitive information during the checkout process.
 - 3.5. User Authentication: - Users should be able to create accounts, log in securely, and manage their profiles and personal information.
 - 3.6. Order Management: - Users should have access to an order management system to track their orders, view shipping details, and request returns or exchanges if necessary.

4. Non-Functional Requirements:
- 4.1. Performance: - The website should have fast loading times and responsive interactions to provide a smooth user experience. - It should handle high traffic and concurrent user sessions efficiently.
 - 4.2. Security: - The website should implement secure authentication mechanisms to protect user accounts and personal information. - It should utilize encryption and secure protocols for data transmission during payment processing.
 - 4.3. Scalability: - The website should be scalable to accommodate increasing numbers of users, products, and orders.
 - 4.4. Compatibility: - The website should be compatible with popular web browsers and ensure consistent functionality across different devices and platforms.
5. Constraints:
- The development of the React website should adhere to project budget and timeline constraints.
 - Compliance with legal and regulatory requirements for e-commerce, such as data protection and privacy laws.
6. Assumptions and Dependencies:
- Availability of necessary hardware and software resources for development and hosting.
 - Integration with external services such as payment gateways and shipping providers.
- The Problem and Requirement Specification provide a clear understanding of the issues faced by users in existing e-commerce platforms and the specific requirements that need to be addressed in the development of a React website. It serves as a foundation for designing, developing, and testing

Project Planning and Scheduling

1. Define Project Goals and Objectives:

- Clearly define the goals and objectives of the React website project, such as developing a user-friendly e-commerce platform with specific features and functionalities.
- Identify key milestones and deliverables to track progress throughout the project.

2. Breakdown of Tasks:

- Identify all the tasks required to complete the project, such as design, frontend development, backend development, integration, testing, and deployment.
- Break down each task into smaller sub-tasks for better manageability.

3. Task Dependencies:

- Determine dependencies between tasks to understand which tasks are dependent on others and which can be executed concurrently.
- Identify critical path tasks that directly impact the project timeline.

4. Estimate Task Durations:

- Estimate the time required for each task based on the complexity and resources available.
- Consider factors such as development effort, testing, review, and potential revisions.

5. Resource Allocation:

- Identify the resources required for each task, including developers, designers, testers, and project managers.
- Allocate resources based on their availability and expertise.

6. Create a Project Schedule:

- Use a project management tool, such as a Gantt chart or project scheduling software, to create a visual representation of the project timeline.
- Assign start and end dates to each task, considering task dependencies and resource availability.
- Define milestones and deliverables within the schedule.

7. Risk Assessment and Mitigation:

- Identify potential risks and challenges that may impact the project timeline, such as changes in requirements, resource unavailability, or technical issues.

- Develop mitigation strategies to address these risks, such as contingency plans, alternative resources, or communication protocols.
8. Communication and Collaboration:
- Establish effective communication channels and collaboration tools to ensure seamless coordination among team members.
 - Schedule regular meetings, stand-ups, or status updates to track progress, address challenges, and make necessary adjustments.
9. Monitoring and Tracking:
- Continuously monitor the progress of tasks and compare it with the project schedule.
 - Update the project schedule as needed, considering any changes or delays.
10. Project Contingency:
- Allow buffer time for unexpected delays or challenges that may arise during the project.
 - Assess the impact of any changes on the project timeline and adjust accordingly.
11. Documentation:
- Maintain documentation of the project plan, schedule, and any changes or updates made throughout the development process.
 - Document lessons learned and best practices for future reference.
12. Regular Evaluation and Review:
- Conduct regular evaluations and reviews of the project progress, milestones, and deliverables to ensure adherence to the project plan and identify areas for improvement.

Analysis

Analyzing a React website involves assessing various aspects of its design, functionality, performance, and user experience. Here are key areas to consider during the analysis:

1. User Interface and Design:
 - Evaluate the overall user interface design, including layout, color scheme, typography, and visual appeal.
 - Assess the consistency and intuitiveness of navigation elements, menus, buttons, and forms.
 - Check for responsive design, ensuring the website adapts well to different screen sizes and devices.
2. Functionality and Features:
 - Review the core functionality of the website, such as product listing, search, shopping cart, checkout, and user authentication.
 - Assess the implementation of additional features like product filtering, sorting, wishlist, order history, and social sharing.
 - Check for seamless integration with external services, such as payment gateways, shipping providers, or APIs.
3. Performance:
 - Evaluate the website's performance, considering factors like page load speed, responsiveness, and smooth interactions.
 - Check for efficient data fetching and rendering, optimizing network requests, and minimizing unnecessary re-renders.
 - Assess the handling of large data sets, including pagination or lazy loading to improve performance.
4. Scalability and Maintainability:
 - Assess the website's architecture and code structure, considering its ability to scale as the user base and data grow.
 - Evaluate the modularity and reusability of code components, promoting maintainability and future development.
 - Consider the use of state management libraries like Redux or context API for efficient data management.
5. Accessibility:

- Evaluate the website's accessibility features, ensuring it conforms to web accessibility standards (e.g., WCAG 2.0 or 2.1).
 - Assess the use of appropriate HTML semantics, alt text for images, keyboard navigation support, and proper focus management.
6. Security:
- Review the implementation of security measures to protect user data, including secure authentication and encryption of sensitive information.
 - Check for protection against common web vulnerabilities like Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), or SQL injection.
7. Browser Compatibility:
- Test the website on different web browsers (e.g., Chrome, Firefox, Safari, Edge) to ensure consistent functionality and rendering.
 - Check for any browser-specific issues or discrepancies in rendering or user experience.
8. User Experience (UX):
- Evaluate the overall user experience, considering factors like ease of use, intuitive navigation, and clear communication of actions.
 - Assess the flow of user interactions, ensuring a logical progression through the website's features and functionalities.
 - Solicit user feedback or conduct usability testing to gain insights into user satisfaction and identify areas for improvement.
9. Mobile Responsiveness:
- Test the website on various mobile devices and screen sizes to ensure it provides a seamless experience across different platforms.
 - Check for responsive design elements, optimized touch interactions, and mobile-specific considerations.
10. Analytics and Tracking:
- Assess the integration of analytics tools to track user behavior, site performance, and conversion rates.
 - Check for proper implementation of event tracking, goal tracking, and conversion funnels to gain insights into website performance.

By conducting a thorough analysis of a React website across these dimensions, you can identify areas of strength and areas for improvement, leading to an enhanced user experience and overall success of the website.

Modules of the System

The modules of a React website can vary depending on the specific requirements and functionality of the system. However, here are some common modules that are often found in a typical React website:

1. User Authentication:

- This module handles user registration, login, and authentication processes.
- It manages user credentials, session management, and password reset functionalities.

2. Product Management:

- This module deals with the management of products available on the website.
- It includes features like product listing, adding new products, editing existing products, and managing product categories and attributes.

3. Shopping Cart:

- The shopping cart module allows users to add products to their cart, view the cart contents, update quantities, and proceed to the checkout process.
- It handles calculations such as subtotal, taxes, discounts, and total amount.

4. Order Management:

- This module is responsible for managing customer orders.
- It includes features like order placement, order tracking, order history, generating invoices, and handling order cancellations or returns.

5. Payment Gateway Integration:

- This module integrates with third-party payment gateways to facilitate secure online payment processing.
- It handles the communication between the website and the payment gateway, ensuring smooth and secure transactions.

6. Search and Filtering:

- This module provides search functionality for users to search for products based on keywords, categories, or attributes.
- It may include advanced filtering options to narrow down search results.

7. User Profile and Settings:

- This module allows users to manage their profiles, update personal information, and configure settings such as communication preferences, shipping addresses, and saved payment methods.

8. Wishlist:

- The wishlist module enables users to create and manage a list of products they want to save for future reference or purchase.
- It allows users to add or remove items from their wishlist and view their saved products.

9. Content Management System (CMS):

- This module provides a backend interface for managing website content.
- It allows administrators to create and edit static pages, manage banners or promotional content, and update other website-related information.

10. Analytics and Reporting:

- This module integrates with analytics tools to track website performance, user behavior, and conversion rates.
- It generates reports and visualizations to provide insights into website usage, popular products, and other key metrics.

11. Admin Dashboard:

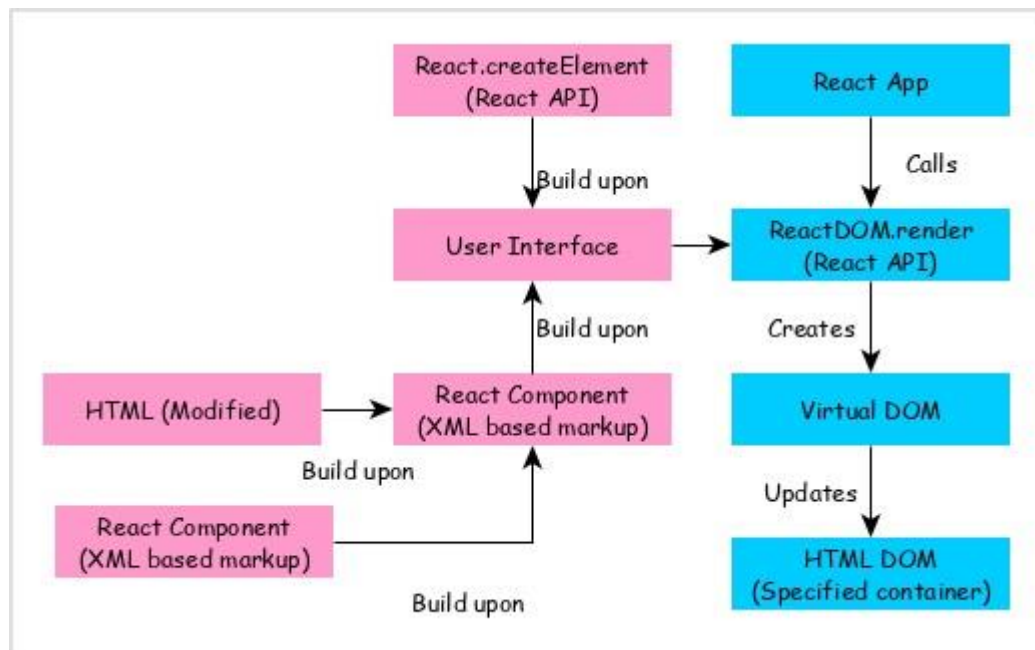
- The admin dashboard module provides a centralized interface for website administrators.
- It includes functionalities for managing users, orders, products, content, and other administrative tasks.

12. Social Media Integration:

- This module integrates social media platforms to enable social sharing of products, login via social media accounts, or other social interactions.

These modules can be further divided into smaller components and implemented using React components, state management libraries (such as Redux), and other relevant tools and technologies. The specific modules and their complexity may vary based on the requirements and scale of the React website.

System Flow Chart



Coding

Homepage

```
import { useEffect,useContext } from "react";
import './Home.scss';
import Banner from './Banner/Banner';
import Category from './Category/Category';
import Products from './Products/Products';
import { fetchDataFromApi } from '../utils/api'
import { Context } from '../utils/context';

const Home = () => {
  const { products,setProducts,categories,setCategories }=useContext(Context);
  useEffect(()=>{
    getProducts()
    getCategories()
  },[])
  const getProducts=()=>{
    fetchDataFromApi("/api/products?populate=*").then((res)=>{
      console.log(res);// for categories call
      setProducts(res);
    });
  };
  const getCategories=()=>{
    fetchDataFromApi("/api/categories?populate=*").then((res) =>{
      console.log(res);
      setCategories(res);
    });
  }
  return (<div>
    <Banner/>
    <div className="main-content">
    <div className="layout"></div>
```

```
</div>
<Category categories={categories}/>
<Products headingText="Popular Products" products={products}/>
</div>
);
};
```

```
export default Home;
```

```
Home.scss
```

```
@import "../css-config/mixins.scss";
```

```
.main-content{
max-width: calc(100% - 20px);//for mobile
margin: 0 auto;
@include md{
max-width: 1200px;
}
}
```

Footer.js

```
import { useEffect, useS, useContext, useState } from "react";
import { useNavigate } from "react-router-dom";
import { TbSearch } from "react-icons/tb";
import { CgShoppingCart } from "react-icons/cg";
import { AiOutlineHeart } from "react-icons/ai";
import Search from "../Search/Search";
import Cart from "../Cart/Cart";
import { Context } from "../../utils/context";
import "../Header.scss";
const Header = () => {
  const [scrolled, setScrolled] = useState(false);
  const [showCart, setShowCart] = useState(false);
  const [showSearch, SetShowSearch] = useState(false);
  const { cartCount } = useContext(Context)
  const navigate = useNavigate();
  const handleScroll = () => {
    const offset = window.scrollY;
    console.log(offset); //for scroll bar
    if (offset > 200) {
      setScrolled(true);
    }
    else {
      setScrolled(false)
    }
  };
  useEffect(() => {
    window.addEventListener("scroll", handleScroll)
  } //for scroll bar
  , [])
  return (
    <
```



```

<header className={`main-header ${ scrolled ? "sticky-header" : "" }`>
<div className="header-context">
<ul className="left">
<li onClick={()=>navigate("/")}>Home</li>
<li onClick={()=>navigate("/footer")} >About</li>
<li onClick={()=>navigate("/category")}>Category</li>
</ul>
<div className="center" onClick={()=>navigate("/")}>ShopByTop
</div>
<div className="right">
< TbSearch onClick={()=> SetShowSearch(true)} />
< AiOutlineHeart />

<span className="cart-icon" onClick={()=>setShowCart(true)}>
< CgShoppingCart />
{ !!cartCount && <span>{ cartCount } </span> }
</span>
</div>
</div>
</header>
{ showCart && <Cart setShowCart={ setShowCart } /> }
{
showSearch &&

<Search SetShowSearch={ SetShowSearch } /> }
</>
);
};

```

```
export default Header;
```

```
Footer.scss
```

```
@import "../css-config/mixins.scss";
```

```

.main-header {
width: 100%;
padding: 0 20px;
background-color: #2455f4;
color: white;
border-bottom: 1px solid rgba(0, 0, 0, 0.1);
z-index: 99;
@include md {
padding: 0 40px;
}
.header-context {
display: flex;
justify-content: space-between;
align-items: center;
height: 50px;
max-width: 1200px;
margin: 0 auto;
@include md {
height: 80px;
}
.left {
list-style-type: none;
align-items: center;
display: none;
gap: 25px;
@include md {
display: flex;
}
li {
font-size: 14px;
font-weight: 600;
text-transform: uppercase;

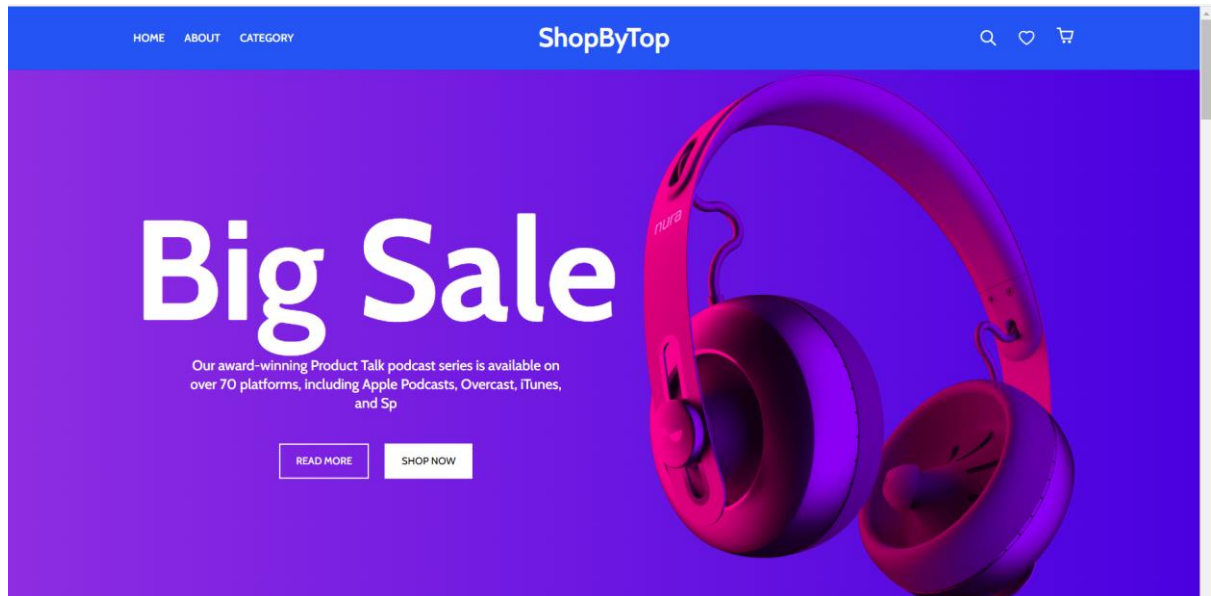
```

```
cursor: pointer;
}
}
.center {
font-size: 22px;
font-weight: 700;
cursor: pointer;
@include md {
font-size: 34px;
position: absolute;
left: 50%;
transform: translateX(-50%);
}
}
.right {
display: flex;
align-items: center;
gap: 20px;
@include md {
gap: 25px;
}
svg {
font-size: 20px;
cursor: pointer;
@include md {
font-size: 24px;
}
}
.cart-icon {
position: relative;
span {
min-width: 20px;
text-align: center;
```

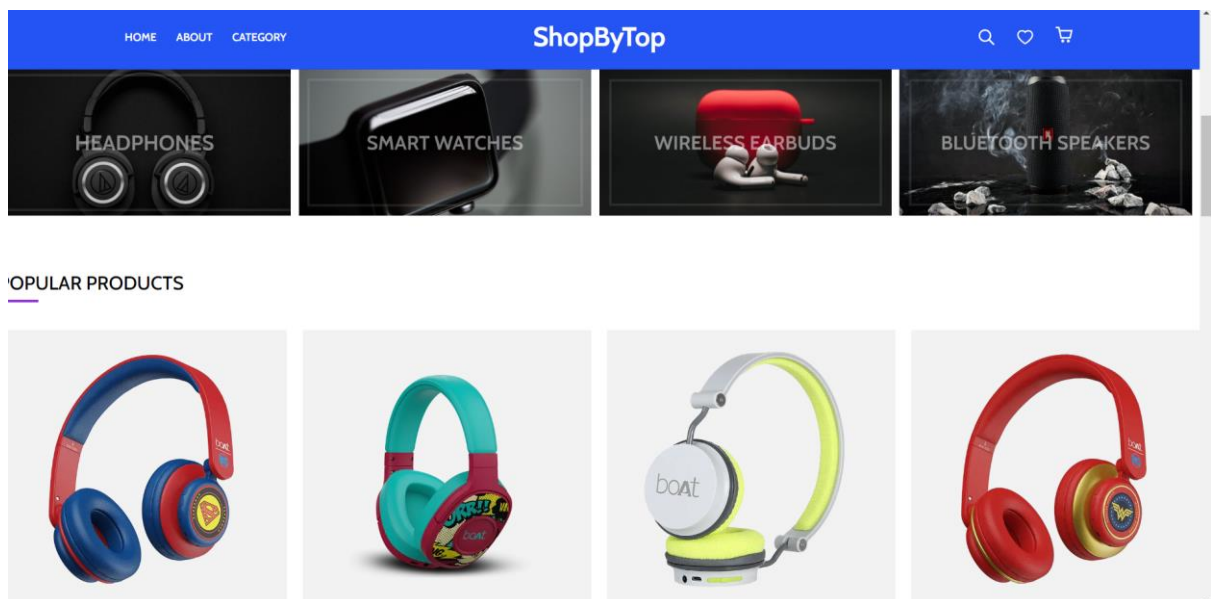
```
background-color: #8e2de2;
padding: 2.5px;
position: absolute;
top: -5px;
right: -12px;
font-size: 12px;
line-height: 1;
border-radius: 10px;
}
}
}
}
&.sticky-header {
position: sticky;
top: 0;
animation: stickyHeader 0.3s ease forwards;
}
}
@keyframes stickyHeader {
0% {
transform: translateX(-80px);
}
100% {
transform: translateX(0);
}
}
```

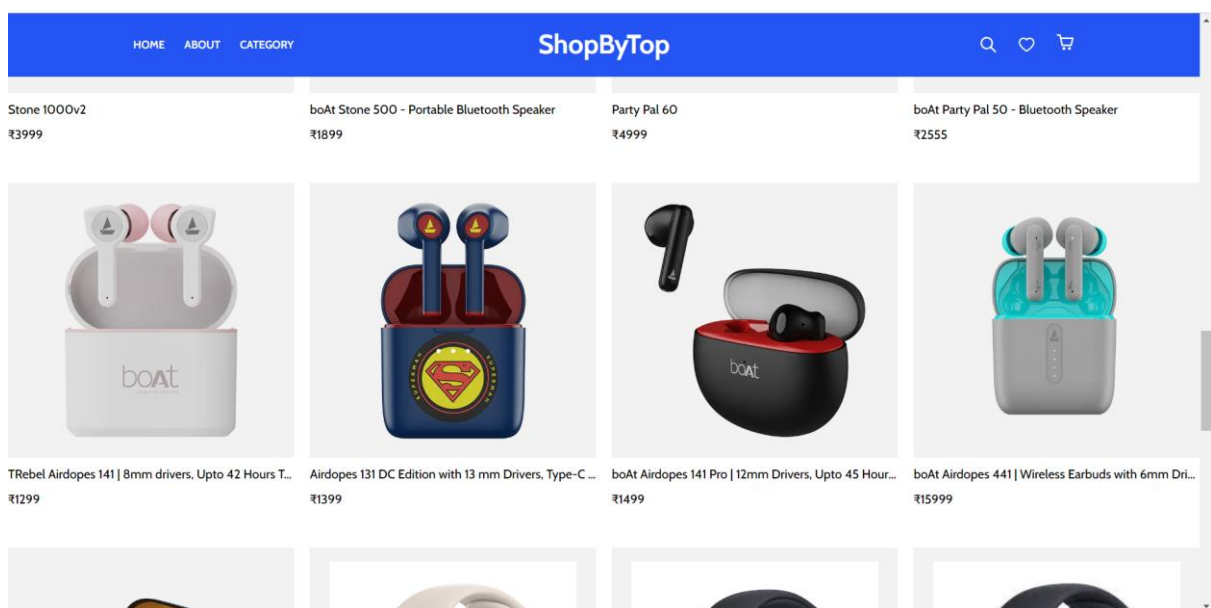
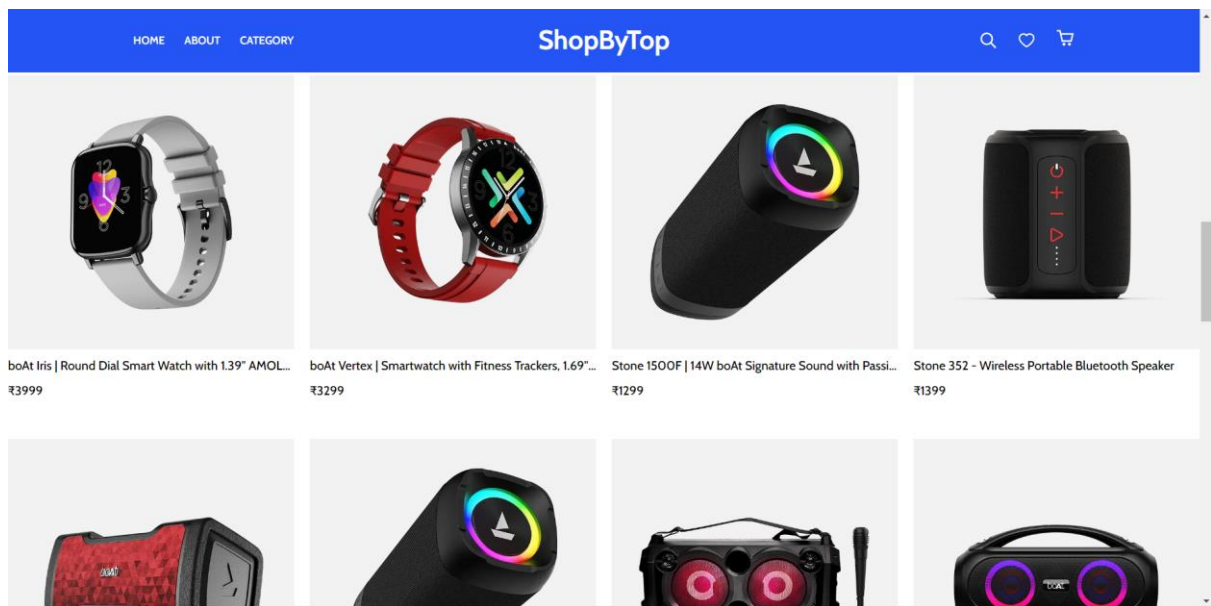
Output of Project

Home page




Product Page






Search page


BOA




boAt Rockerz 450 DC edition | Wireless Headphone with 40mm Dynamic Driver




boAt Iris | Round Dial Smart Watch with 1.39" AMOLED Display, Multiple Watch Faces




boAt Vertex | Smartwatch with Fitness Trackers, 1.69" HD Display, Sleep Tracking, 100+ Watch Faces




Stone 1500F | 14W boAt Signature Sound with Passive Bass Radiator




boAt Stone 500 - Portable Bluetooth Speaker



boAt Party Pal 50 - Bluetooth Speaker



boAt Airdopes 141 Pro | 12mm Drivers, Upto 45 Hours Playback, Quad Mics with ENx™ Technology



boAt Airphones 441 | Wireless Earbuds with 6mm Driver For Immersive Sound

ShopByTop

HOME ABOUT CATEGORY

Apple Watch Starlight Aluminium Case with Sport Band

₹29990

The aluminium case is lightweight and made from 100 per cent recycled aerospace-grade alloy. The Sport Band is made from a durable yet surprisingly soft high-performance fluoroelastomer, with an innovative pin-and-tuck closure.

- 1 +

ADD TO CART

Category Smart Watches

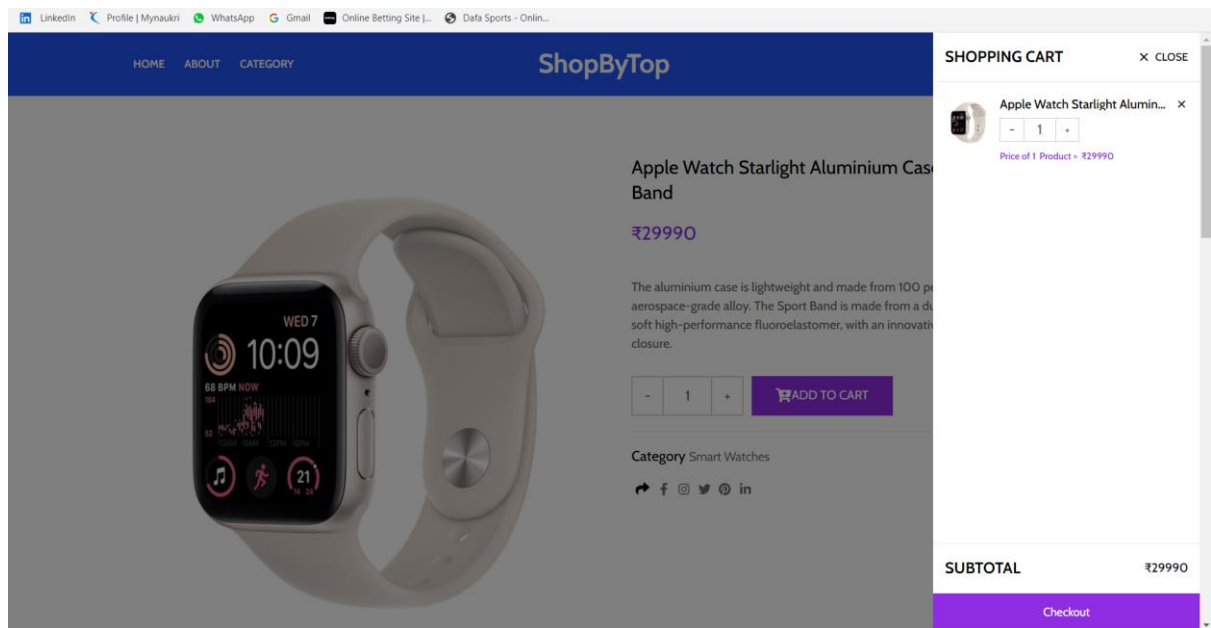
f

@

in

Page 31 of 33

Cart Page



Payment Page

Ecommerce

Apple Watch Starlight Aluminium Case with Sport Band

₹29,990.00

Powered by [stripe](#) | [Terms](#) | [Privacy](#)

Pay with card

Shipping information

Email

deepukumarlaldev5@gmail.com

Shipping address

Deepak Singh

India

Kabir Nagar Jalandhar

Clear

Kabir Nagar Jalandhar



Jalandhar City 144008


Punjab

Payment details

Card information

1234 1234 1234 1234

VISA  

MM / YY CVC 

☒ Billing address is same as shipping

Pay

Footer Page

