BookStore & Library

A PROJECT REPORT
for
Project (KCA451)
Session (2023-24)

Submitted by

Bittu Jaiswal: 2200290140048

Hritik Srivastava: 2200290140074

Dushyant Kaushik: 2200290140058

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Vipin Kumar
(Associate Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

CERTIFICATE

Certified that **Bittu Jaiswal (2200290140048)**, **Hritik Srivastava (2200290140074), Dushyant Kaushik (2200290140058)** has carried out the project work having "**BookStore & Library**" (**Project KCA451**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Bittu Jaiswal (2200290140048)
Hritik Srivastava (2200290140074)
Dushyant Kaushik (2200290140058)

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

Date:

Dr. Vipin Kumar	**Dr. Arun Tripathi**
Associate Professor	**Head**
Department of Computer Applications	**Department of Computer Applications**
KIET Group of Institutions, Ghaziabad	**KIET Group of Institutions, Ghaziabad**

ABSTRACT

The Project Bookstore and Library aims to bridge the gap between commercial and public access to literature by creating a unified platform that combines the functionalities of both a traditional bookstore and a public library. This project will leverage technology to enhance the user experience, providing a seamless interface for purchasing, borrowing, and accessing a diverse range of books and other media. The platform will feature an integrated catalog system, user accounts with personalized recommendations, and community engagement tools such as book clubs and discussion forums. By fostering a synergistic relationship between bookstores and libraries, the project seeks to promote literacy, support local businesses, and increase the accessibility of educational and recreational reading materials. This initiative will also incorporate sustainability practices by encouraging the digital circulation of books and the responsible management of physical inventories. Ultimately, Project Bookstore and Library envisions a future where literature is more accessible, communities are more connected, and the love for reading is universally cultivated.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Vipin Kumar** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

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

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

Bittu Jaiswal

Hritik Srivastava

Dushyant Kaushik

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

In the modern digital age, the accessibility and organization of books and educational resources are pivotal for fostering a culture of learning and literacy. With the advent of technology, traditional bookstores and libraries are evolving into more dynamic and user-friendly platforms. This project leverages the MERN stack (MongoDB, Express.js, React, Node.js) to develop an integrated online bookstore and library system, enhancing the user experience and operational efficiency.

1.2 Project Overview

This project involves the development of an integrated platform for a book store and a library using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The platform aims to combine the commercial aspects of a book store with the community and educational services of a library, providing users with a comprehensive resource for purchasing, borrowing, and exploring a wide range of books and multimedia resources.

1.3 Objective

The objective of the Book Store and Library project using the MERN stack is to develop an integrated, user-friendly platform that optimizes the management and operation of both a book store and a library. This system aims to:

1. **Enhance Operational Efficiency:**
    - Streamline the management of book inventories, user profiles, and transactions to reduce manual efforts and errors.
    - Automate routine processes such as inventory updates, sales tracking, book borrowing, and returns.
2. **Improve User Experience:**

- Provide an intuitive and responsive user interface for both customers of the book store and members of the library.
- Implement robust search and recommendation features to help users easily find and discover new books.

3. **Ensure Data Security and Integrity:**
    - Implement strong authentication and authorization mechanisms to protect user data and ensure privacy.
    - Maintain accurate and up-to-date records of inventory, user activities, and financial transactions.

1.4 Key Features

1. User Management:
    - Registration and Profiles: Enable user sign-ups, profile creation, and profile management.
    - Authentication: Secure login and authentication mechanisms for users and administrators.
2. Sales and Transactions (Book Store):
    - Purchase Processing: Facilitate the purchasing process, including generating receipts.
    - Sales History: Maintain a record of sales transactions for analysis and reporting.
3. Security:
    - Data Protection: Implement robust data security measures to protect user information and transaction data.
    - Access Control: Ensure proper authorization mechanisms to control access to sensitive information and functionalities.
4. User-Friendly Interface:
    - Responsive Design: Ensure the platform is accessible and user-friendly across various devices, including desktops, tablets, and smartphones.
    - Intuitive Navigation: Provide easy-to-navigate interfaces for both administrators and end-users.

5. Scalability:

   - Design the system to efficiently handle increasing amounts of data and a growing number of users.
   - Ensure the architecture supports future expansion and the addition of new features.

1.5 Scope of the project

The scope of BookStore & Library extends to the following:

User Interface: A responsive and user-friendly interface for end users.

Scalability: The Application is designed to adapt and scale as per the evolving needs of the user.

This chapter sets the stage for a detailed exploration of the BookStore & Library. The subsequent chapters will highlight the technical aspects, functionalities and implementation details, providing a comprehensive understanding of this innovative solution

CHAPTER 2

PROBLEM IDENTIFICATION & FEASIBILITY STUDY

2.1 Problem Identification

The motivation behind developing the BookStore arises from the acknowledgment of enduring issues in conventional manual image retrieval processes. Manual systems frequently result in inefficiencies, imprecise image recognition, delayed search processing, and a dearth of transparency in retrieving visual content. These challenges not only hinder the operational flow of image searches but also contribute to a suboptimal user experience

2.2 Feasibility Study

Prior to initiating the development of the BookStore & library, an exhaustive feasibility study was undertaken to evaluate the viability and practicability of the envisioned system. This study comprehensively covers technical, operational, and economic feasibility aspects pertinent to the implementation of an advanced book search retrieval platform.

2.2.1 Technical Feasibility

The technical feasibility examination guaranteed that the envisaged BookStore and Library system could be constructed utilizing the chosen technologies—React, MongoDB, Express.js, and Node.js. It scrutinized the accessibility of necessary software and hardware resources, the adaptability of the system with the current infrastructure, and the requisite technical proficiency for the successful implementation of the book management and retrieval platform.

2.2.2 Operational Feasibility

Operational feasibility scrutinized the degree to which the BookStore and Library system aligns with the current operations of book management and retrieval. This entailed evaluating the seamless integration into daily routines, the adaptability of personnel to the new system, and the overarching influence on operational processes within the context of book search and management functionality.

2.2.3 Economic Feasibility

Economic feasibility delved into the cost-effectiveness of implementing the BookStore and Library system. This encompassed evaluating development costs, continual maintenance expenses, and potential savings or revenue generation attributed to enhanced efficiency and user satisfaction in book management and retrieval processes. A meticulous cost-benefit analysis played a pivotal role in ascertaining the financial viability of the book management project.

In conclusion, the problem identification phase highlighted the deficiencies inherent in existing book management systems, paving the way for a focused solution. The subsequent feasibility study, likewise, affirmed that the envisioned BookStore and Library system is not merely a theoretically sound concept but also a pragmatic and economically viable undertaking.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Introduction

Requirement analysis serves as a pivotal phase in the development lifecycle of the BookStore & Library. This chapter aims to meticulously gather, document, and analyze the functional and non-functional requirements that will shape the design and implementation of the BookStore & Library.

3.2 Functional Requirements

3.2.1 User Module

The User Module is fundamental to the BookStore & Library, encompassing functionalities such as user registration, login, and profile management. End-users should be able to search book, purchases book.

3.3 NON-FUNCTIONAL REQUIREMENTS

3.3.1 PERFORMANCE

The BookStore & Library should be responsive, ensuring swift book search, purchases book. Response times should be optimized to enhance user satisfaction.

3.3.2 SECURITY

Data security is paramount. User authentication, secure transmission of sensitive information, and access controls must be implemented to safeguard user data.

3.3.3 SCALABILITY

The system should be designed to accommodate an increasing number of users.

3.3.4 USABILITY

A user-friendly interface is essential for end-users. The bookstore & Library should be intuitive, should be easy to use

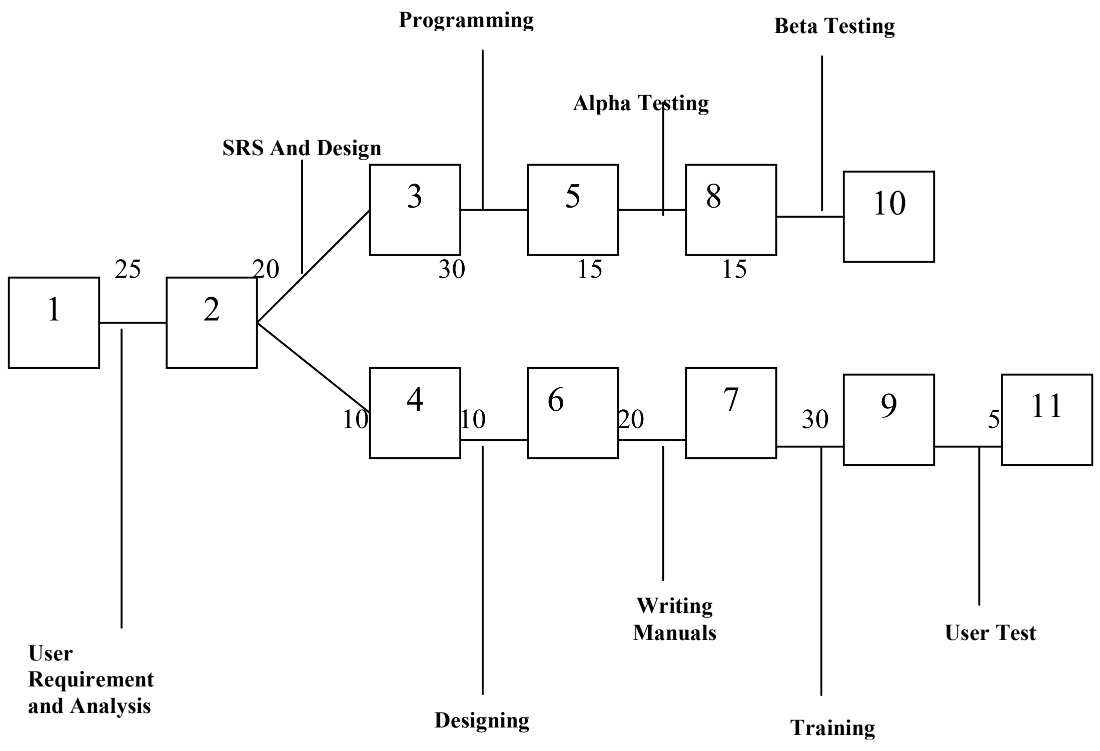
CHAPTER 4

PROJECT PLANNING AND SCHEDULING

4.1 Pert Chart:

A PERT chart is a project management tools used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project as network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project.

The direction of the arrows on the lines indicates the sequence of tasks.



CHAPTER 5

HARDWARE & SOFTWARE SPECIFICATION

5.1 Hardware Specification

The Bookstore and Library will be designed and deployed on a hardware infrastructure that guarantees optimal performance and reliability. The recommended hardware specifications are as follows:

Server:

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 256 GB SSD or higher

Database Server:

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 256 GB SSD or higher

Network Interface: Gigabit Ethernet

Client Machines:

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 128 GB SSD or higher

Network Interface: 100 Mbps Ethernet or Wi-Fi

5.2 Software Specification

The BookStore & Library will be developed using a combination of server-side and client-side technologies. The development and deployment environment will be facilitated by MERN, which provides a comprehensive stack for web application development. The software specifications include:

Server-Side Technologies:

Operating System: Windows Server 2016 or later

Web Server: Node.js

Database Management System: MongoDB

Server-Side Scripting Language: JavaScript (Node.js)

Client-Side Technologies:

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge

Client-Side Scripting: JavaScript

Development Tools:

Integrated Development Environment (IDE): Visual Studio Code.

Version Control:

Git: Version control for collaborative development

Security:

SSL/TLS: Ensure secure data transmission over the network

Firewall: Implement firewall rules to restrict unauthorized access

Anti-malware Software: Regularly updated anti-malware software on server and client machines

CHAPTER 6

CHOICE OF TOOLS & TECHNOLOGY

6.1 React

React is a JavaScript library used for building user interfaces. React allows developers to describe the UI's appearance at any point in time, and it automatically updates and renders the right components when the data changes.

React organizes the UI into reusable components, making it easier to manage and maintain complex applications. React uses a virtual DOM to optimize and update the actual DOM efficiently, improving performance by minimizing unnecessary re-rendering.

React uses JSX, a syntax extension that looks similar to XML or HTML, to describe the structure of UI components in a more concise and readable way**.** React components can manage their internal state, and data can be passed to components through props, allowing for dynamic and interactive UIs

6.2 MangoDB

MongoDB is a NoSQL (non-relational) database management system that stores data in a flexible, JSON-like format known as BSON (Binary JSON). It is designed to handle large volumes of unstructured or semi-structured data, making it particularly suitable for applications with evolving and dynamic data requirements.

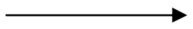
6.3 Data Flow Diagram

The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It’s also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD: -

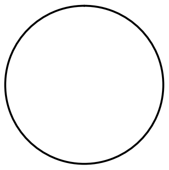
In the DFD, four symbols are used and they are as follows.

1. A square defines a source (originator) or destination of system data.

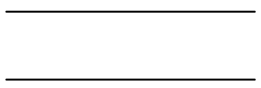
2. An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.



3. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.

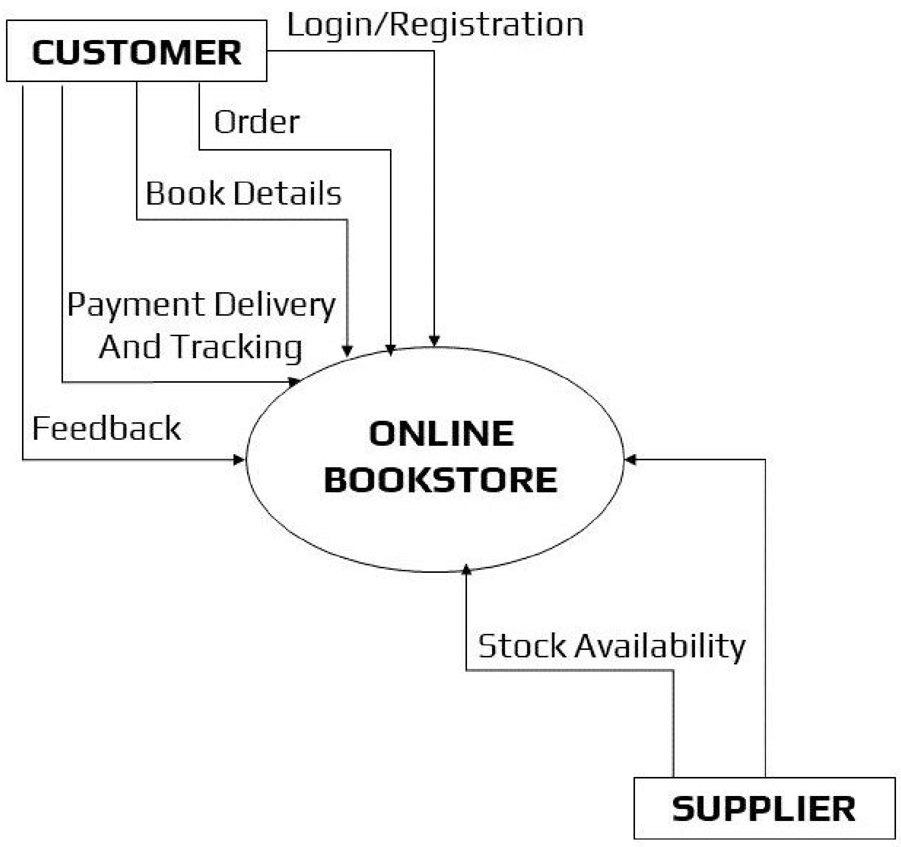


4. An open rectangle is a data store-data at rest, or a temporary repository of data.



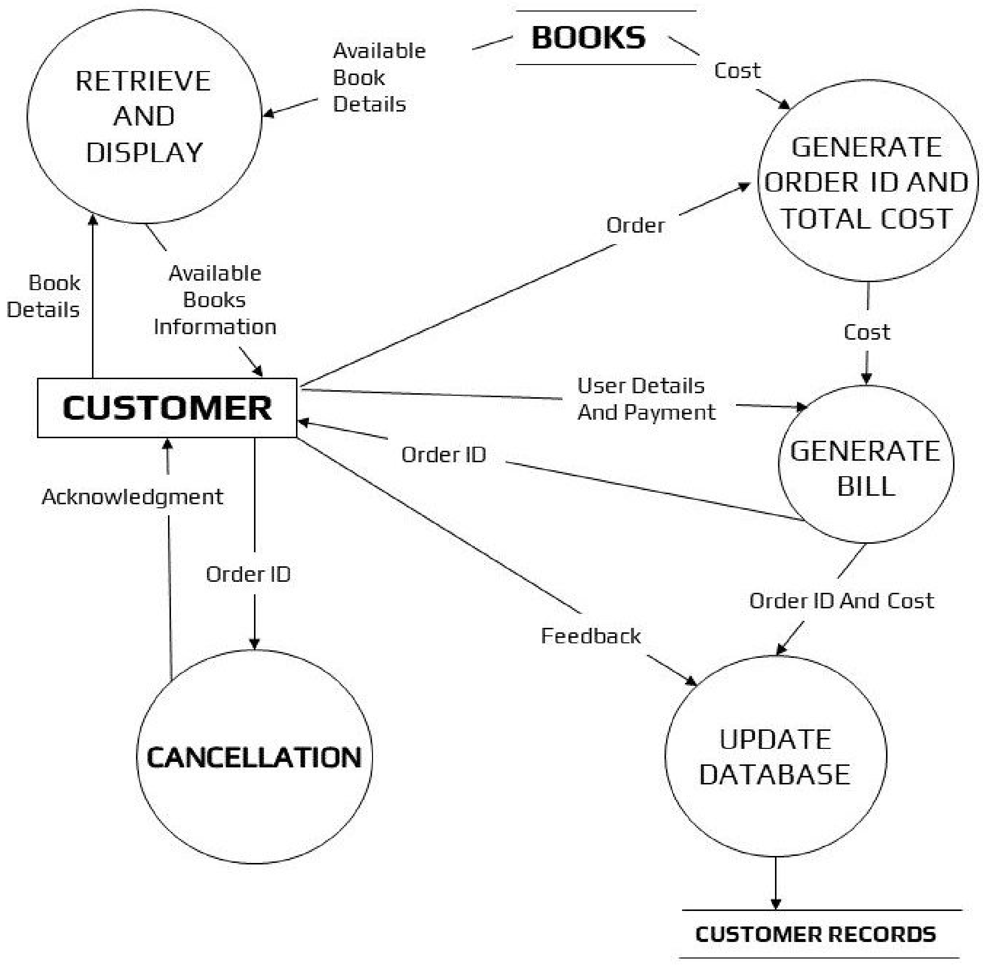
6.4 Context Level Diagram

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Canteen Management System is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.



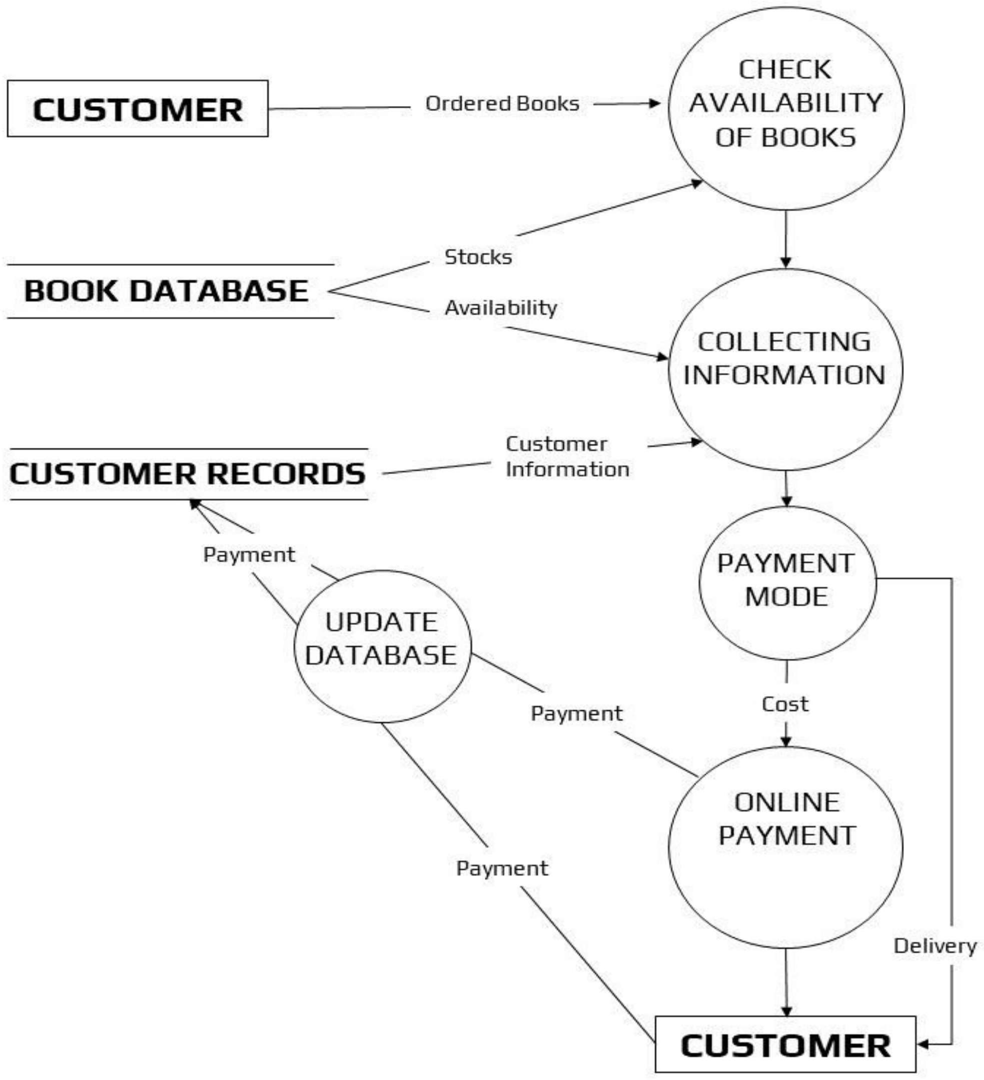
Zero Level Data flow Diagram

Fig 6.1



First Level Data flow Diagram

Fig 6.2



Second Level Data flow Diagram

Fig 6.3

CHAPTER 7

ER-DIAGRAM

7.1 Entity-relationship model: -

The entity-relationship model or entity-relationship diagram (ERD) is a data model or diagram for high-level descriptions of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams.

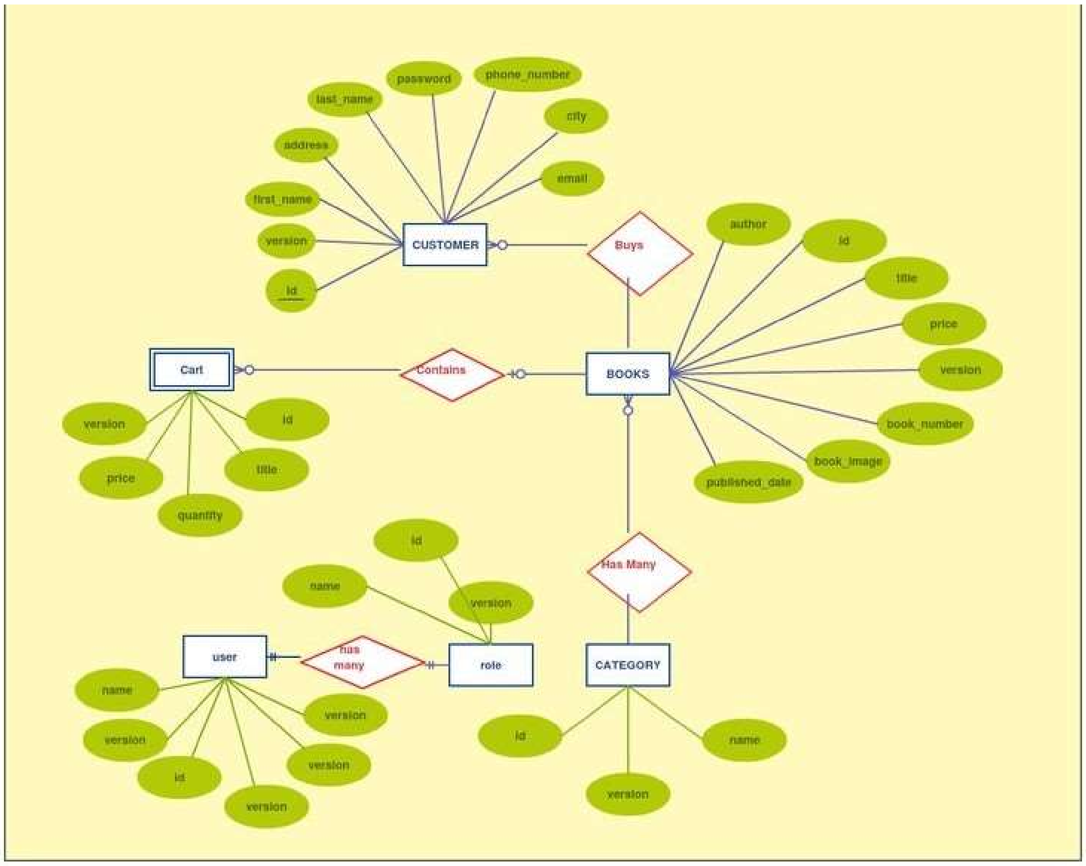


Fig 7.1

7.2 Class Diagram: -

Authentication:

- Classification: Weak Class
- Description: Represents user authentication details, including username and password. This class is responsible for user login functionality.

User:

- Classification: Strong Class
- Description: Represents the users of the system.

Canteen Item:

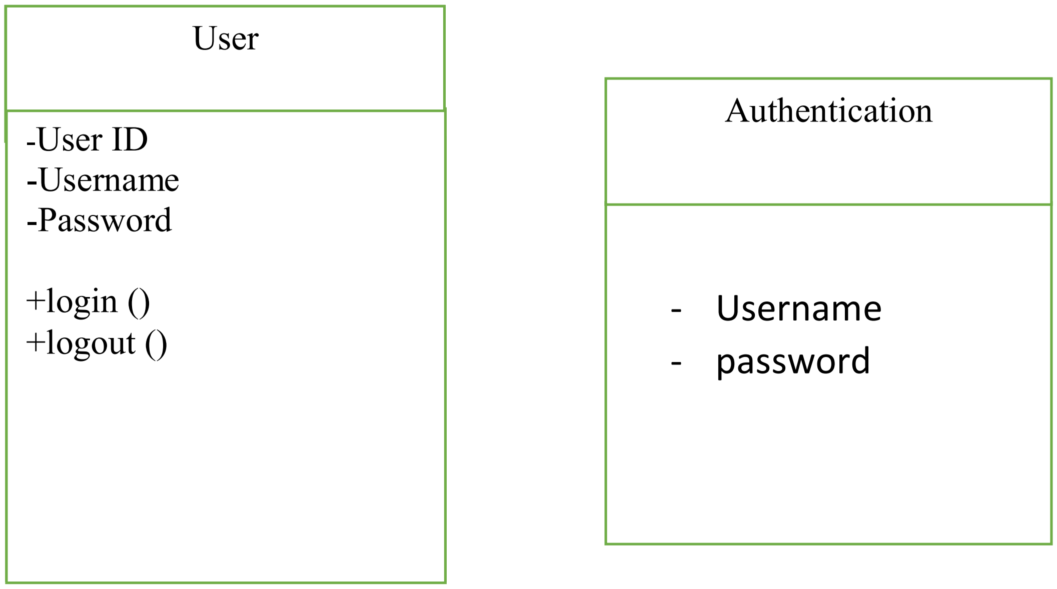
- Classification: Strong Class

• Description: Represents the items available in the canteen, including details such as item ID, name, price, and stock quantity.

Stock:

• Classification: Strong Class

• Description: Represents the stock of items in the canteen, including details such as item ID, quantity, and reorder level.



CHAPTER 8

FORM DESIGN

8.1 Home

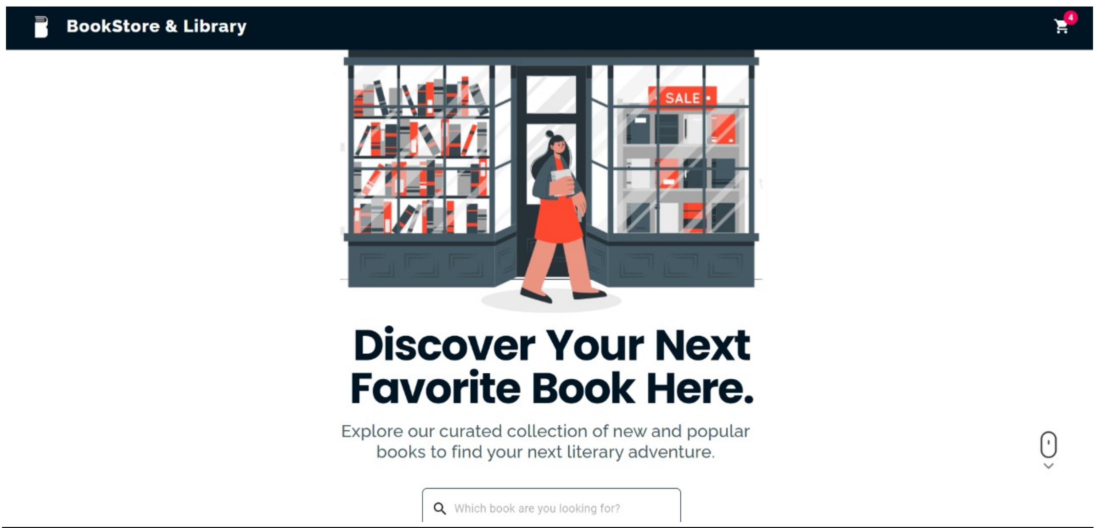
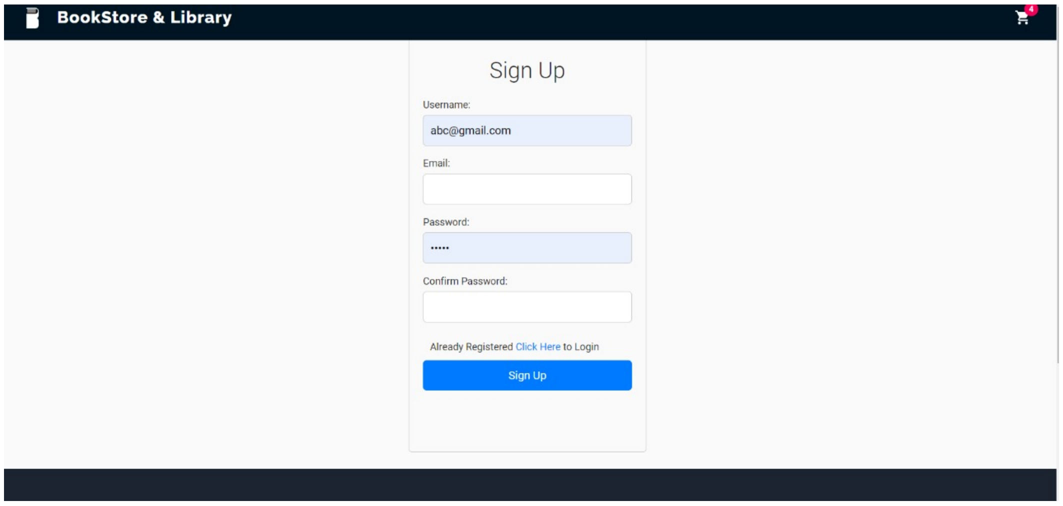

BookStore & Library
SALE
Discover Your Next Favorite Book Here.
Explore our curated collection of new and popular books to find your next literary adventure.
Which book are you looking for?

Fig. 8.1

The home page of bookstore & library application serves as the initial point of interaction for users and typically aims to provide a user-friendly and intuitive experience. A navigation menu present, offering links to different sections of the application, such as carts, search book, user account settings.

8.2 Signup



Username: abc@gmail.com

Email:

Password: •••••

Confirm Password:

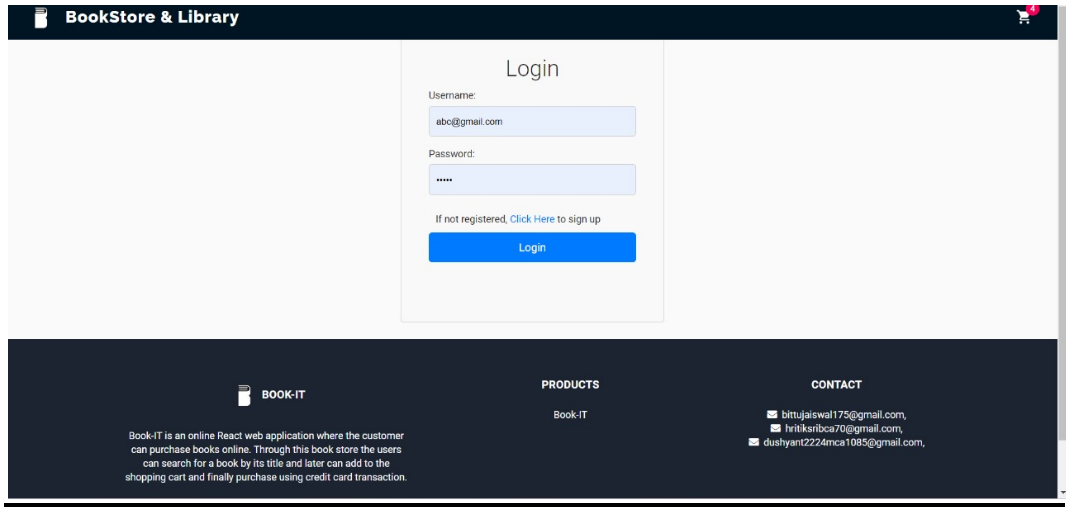
Fig. 8.2

The primary element of the signup page is the registration form, where users provide essential information to create an account. This typically includes fields for a username, email address, password, and possibly additional details for personalization.

A prominent "Sign Up" or "Create Account" button allows users to submit their information and proceed with the registration process.

Upon successful registration, a confirmation message is displayed, welcoming users to the platform and providing any additional instructions, such as logging in to access the application.

8.3 Signin



Book-IT is an online React web application where the customer can purchase books online. Through this book store the users can search for a book by its title and later can add to the shopping cart and finally purchase using credit card transaction.

bittujaiswal175@gmail.com,
hritiksribca70@gmail.com,
dushyant2224mca1085@gmail.com,

Fig 8.3

The login module in the BookStore & Library is a pivotal component responsible for securely authenticating users and regulating access to the system. This module verifies user identity by validating entered credentials, typically consisting of a username and password. Following successful authentication user can search book. Session management is initiated to maintain user states throughout their interactions within the BookStore & Library.

Security measures, including password hashing and salting, are implemented to safeguard user credentials. The login module may also include features such as an account lockout mechanism to counter multiple failed login attempts, logging and auditing for monitoring user activities, and password recovery options for forgotten passwords.

Overall, the login module is of paramount importance as it serves as the initial checkpoint, ensuring that only authorized users gain access to the BookStore & Library while upholding security and user-friendly practices.

8.4 Search module

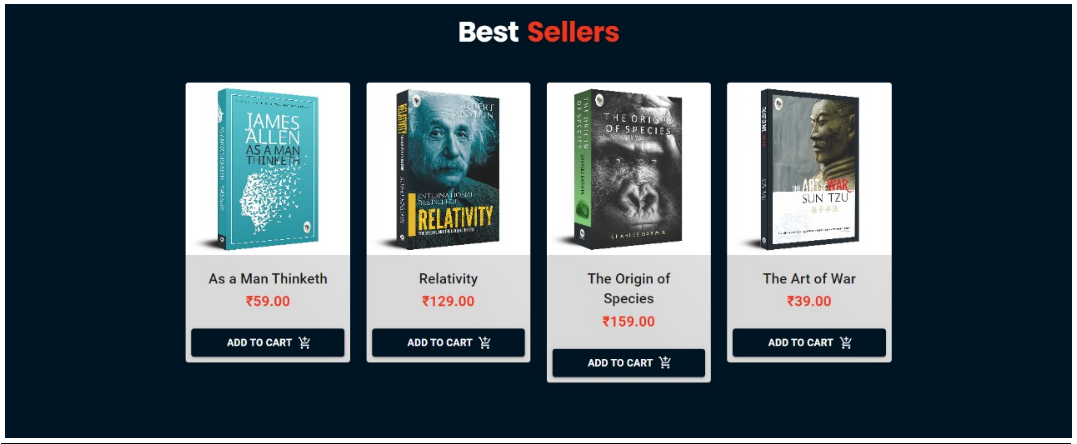


Fig 8.4

Search is in an BookStore & Library typically refers to the functionality or component within the application that allows users to input their search queries and retrieve relevant results.

8.5 Result

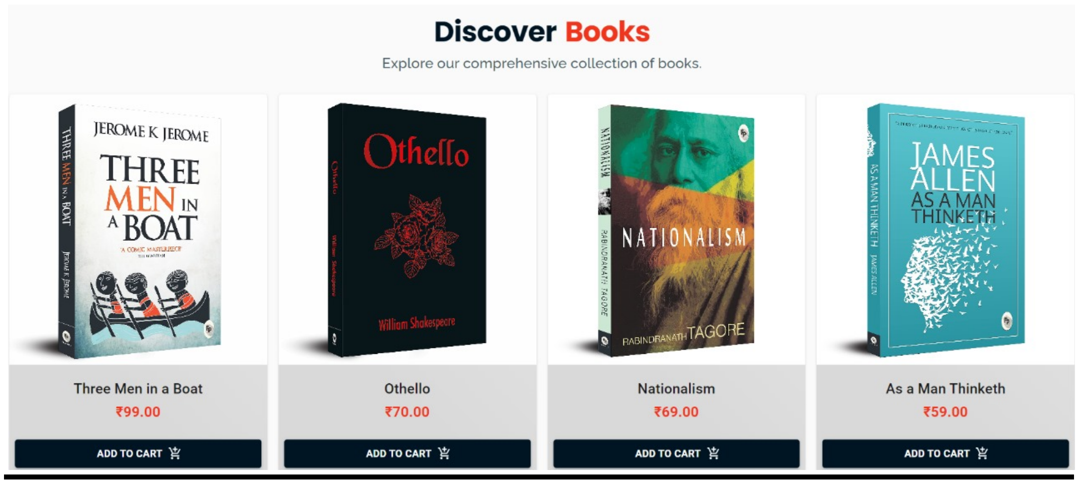


Fig 8.5

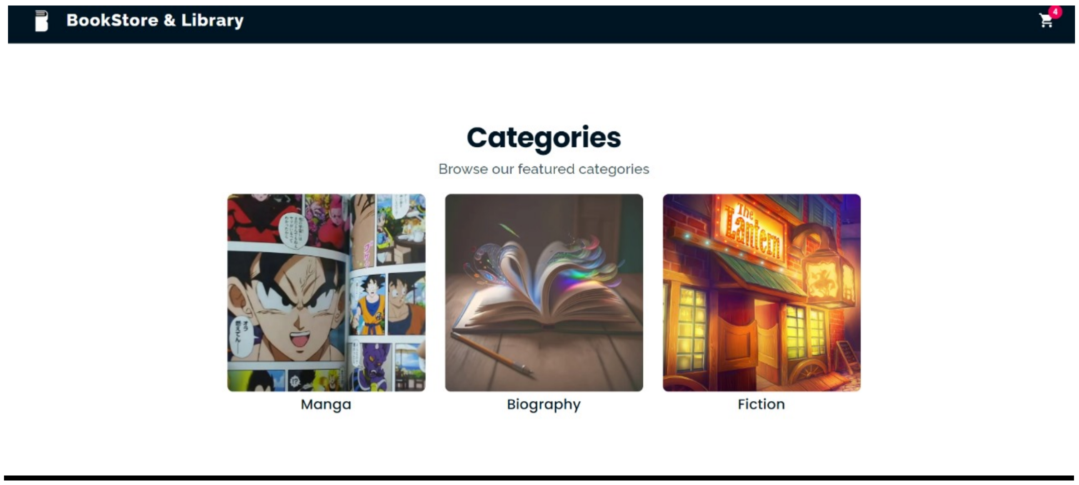
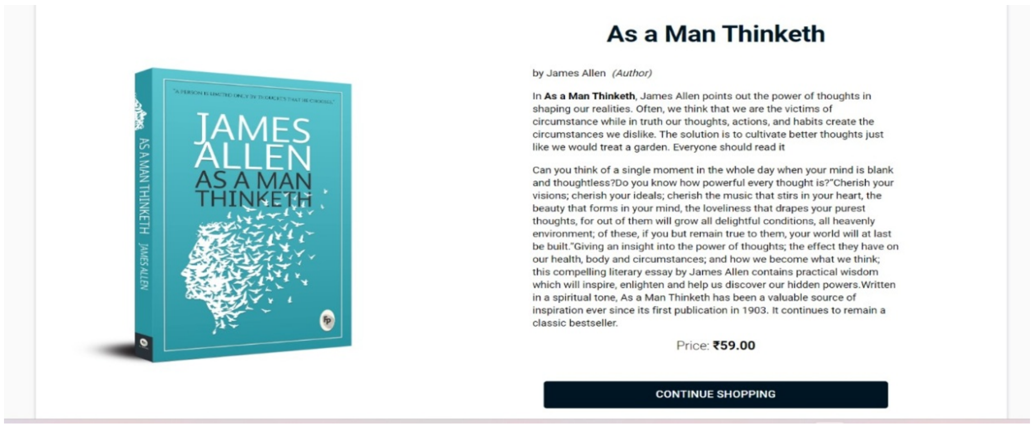


Fig 8.6


As a Man Thinketh

by James Allen *(Author)*

In **As a Man Thinketh**, James Allen points out the power of thoughts in shaping our realities. Often, we think that we are the victims of circumstance while in truth our thoughts, actions, and habits create the circumstances we dislike. The solution is to cultivate better thoughts just like we would treat a garden. Everyone should read it

Can you think of a single moment in the whole day when your mind is blank and thoughtless?Do you know how powerful every thought is?"Cherish your visions; cherish your ideals; cherish the music that stirs in your heart, the beauty that forms in your mind, the loveliness that drapes your purest thoughts, for out of them will grow all delightful conditions, all heavenly environment; of these, if you but remain true to them, your world will at last be built."Giving an insight into the power of thoughts; the effect they have on our health, body and circumstances; and how we become what we think; this compelling literary essay by James Allen contains practical wisdom which will inspire, enlighten and help us discover our hidden powers.Written in a spiritual tone, As a Man Thinketh has been a valuable source of inspiration ever since its first publication in 1903. It continues to remain a classic bestseller.

Price: **₹59.00**

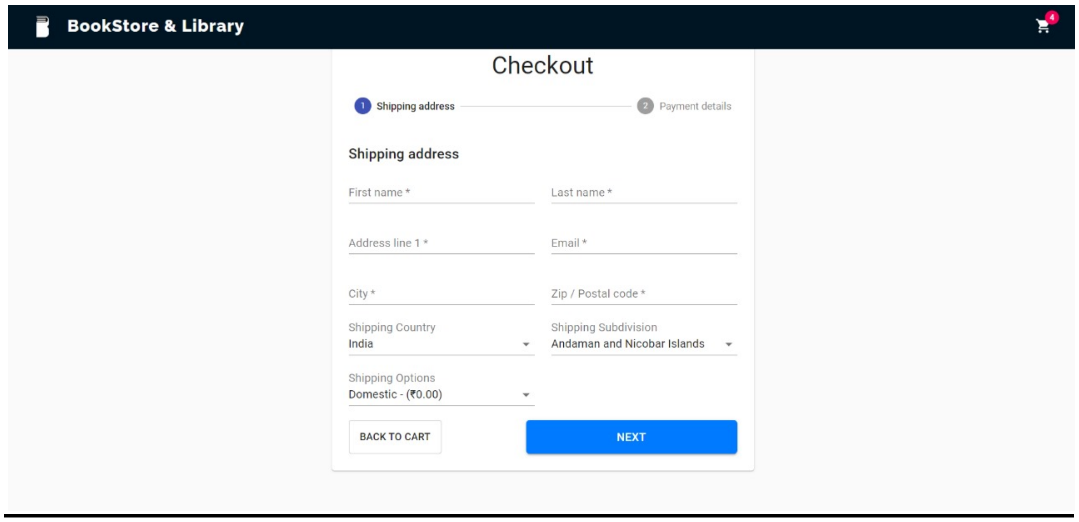
Fig 8.7

When users initiate a search in BookStore & Library, the system employs sophisticated algorithms to analyze and match the query against a vast database of indexed book. The results are then displayed to the user in the form of a visually appealing grid or list of books.

Relevant metadata, such as book name, size, and additional information, may be provided alongside the results. This helps users assess the suitability of each book before clicking on it.

BookStore & Library often feature a responsive design, enabling users to view results seamlessly across various devices, from smartphones to desktops, enhancing accessibility and user experience.

Checkout



In an online bookstore, the cart is a crucial element of the user interface, facilitating the shopping process. Here are its key features and functions:

- **Selection and Storage**: Users can add books to their cart as they browse. The cart stores these selections until the user is ready to purchase.
- **Quantity Adjustment**: Users can specify the number of copies for each book.
- **Price Calculation**: The cart provides a running total of the selected items, including discounts, taxes, and shipping costs.
- **Wishlist and Save for Later**: Users can move items to a wishlist or save for later if they are not ready to purchase immediately.

CHAPTER 9

CODING

Package.json

```
{
  "name": "ebook",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@chec/commerce.js": "^2.3.0-beta1",
    "@material-ui/core": "^4.11.0",
    "@material-ui/icons": "^4.9.1",
    "@material-ui/lab": "^4.0.0-alpha.56",
    "@stripe/react-stripe-js": "^1.1.2",
    "@stripe/stripe-js": "^1.11.0",
    "@testing-library/jest-dom": "^4.2.4",
    "@testing-library/react": "^9.3.2",
    "@testing-library/user-event": "^7.1.2",
    "bootstrap": "^5.0.1",
    "gh-pages": "^3.2.0",
    "mdbreact": "^4.6.1",
    "react": "^16.13.1",
    "react-bootstrap": "^1.6.1",
    "react-dom": "^16.13.1",
    "react-hook-form": "^6.11.5",
    "react-hot-toast": "^2.4.1",
    "react-responsive-carousel": "^3.2.18",
    "react-router": "^5.2.0",
    "react-router-dom": "^5.3.4",
    "react-scripts": "^5.0.1",
    "react-stripe-elements": "^6.1.2",
    "semantic-ui-react": "^2.0.1"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "tailwindcss": "^3.4.3"
  }
}
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
    <App />,
    document.getElementById('root')
    );
```

App.js

```
import React, { useState, useEffect } from "react";
import { CssBaseline } from "@material-ui/core";
import { commerce } from "./lib/commerce";
import Products from "./components/Products/Products";
import Navbar from "./components/Navbar/Navbar";
import Cart from "./components/Cart/Cart";
import Checkout from "./components/CheckoutForm/Checkout/Checkout";
import ProductView from "./components/ProductView/ProductView";
import Manga from "./components/Manga/Manga";
import Footer from "./components/Footer/Footer";
import { BrowserRouter as Router, Switch, Route } from "react-
router-dom";
import "bootstrap/dist/css/bootstrap.min.css";
import "mdbreact/dist/css/mdb.css";
import "@fortawesome/fontawesome-free/css/all.min.css";
import loadingImg from "./assets/loader.gif";
import "./style.css";
import Fiction from "./components/Fiction/Fiction";
import Biography from "./components/Bio/Biography";
import Login from "./components/login/Login";
import SignUp from "./components/signUp/SignUp";
import { Toaster } from "react-hot-toast";

const App = () => {
  const [mobileOpen, setMobileOpen] = React.useState(false);
  const [products, setProducts] = useState([]);
  const [mangaProducts, setMangaProducts] = useState([]);
  const [fictionProducts, setFictionProducts] = useState([]);
  const [bioProducts, setBioProducts] = useState([]);
  const [featureProducts, setFeatureProducts] = useState([]);
  const [cart, setCart] = useState({});
  const [order, setOrder] = useState({});
  const [errorMessage, setErrorMessage] = useState("");

  const fetchProducts = async () => {
    const { data } = await commerce.products.list();

    setProducts(data);
  };

  const fetchMangaProducts = async () => {
    const { data } = await commerce.products.list({
      category_slug: ["manga"],
    });

    setMangaProducts(data);
  };
  const fetchFeatureProducts = async () => {
    const { data } = await commerce.products.list({
      category_slug: ["featured"],
    });

    setFeatureProducts(data);
  };

  const fetchFictionProducts = async () => {
    const { data } = await commerce.products.list({
      category_slug: ["fiction"],
    });

    setFictionProducts(data);
  };

  const fetchBioProducts = async () => {
    const { data } = await commerce.products.list({
      category_slug: ["biography"],
    });

    setBioProducts(data);
  };

  const fetchCart = async () => {
    setCart(await commerce.cart.retrieve());
  };

  const handleAddToCart = async (productId, quantity) => {
    const item = await commerce.cart.add(productId, quantity);

    setCart(item.cart);
  };

  const handleUpdateCartQty = async (lineItemId, quantity) => {
    const response = await commerce.cart.update(lineItemId, {
quantity });

    setCart(response.cart);
  };

  const handleRemoveFromCart = async (lineItemId) => {
    const response = await commerce.cart.remove(lineItemId);
    setCart(response.cart);
  };

  const handleEmptyCart = async () => {
    const response = await commerce.cart.empty();

    setCart(response.cart);
  };

  const refreshCart = async () => {
    const newCart = await commerce.cart.refresh();

    setCart(newCart);
  };

  const handleCaptureCheckout = async (checkoutTokenId, newOrder)
=> {
    try {
      const incomingOrder = await commerce.checkout.capture(
        checkoutTokenId,
        newOrder
      );

      setOrder(incomingOrder);

      refreshCart();
    } catch (error) {
      setErrorMessage(error.data.error.message);
    }
  };

  useEffect(() => {
    fetchProducts();
    fetchFeatureProducts();
    fetchCart();
    fetchMangaProducts();
    fetchFictionProducts();
    fetchBioProducts();
  }, []);

  const handleDrawerToggle = () => setMobileOpen(!mobileOpen);

  return (
    <div>
      {products.length > 0 ? (
        <>
          <Router>
            <div style={{ display: "flex" }}>
              <CssBaseline />
              <Navbar
                totalItems={cart.total_items}
                handleDrawerToggle={handleDrawerToggle}
              />
              <Switch>
                <Route exact path="/">
                  <Products
                    products={products}
                    featureProducts={featureProducts}
                    onAddToCart={handleAddToCart}
                    handleUpdateCartQty
                  />
                </Route>
                <Route exact path="/cart">
                  <Cart
                    cart={cart}
                    onUpdateCartQty={handleUpdateCartQty}
                    onRemoveFromCart={handleRemoveFromCart}
                    onEmptyCart={handleEmptyCart}
                  />
                </Route>
                <Route path="/checkout" exact>
                  <Checkout
                    cart={cart}
                    order={order}
                    onCaptureCheckout={handleCaptureCheckout}
                    error={errorMessage}
                  />
                </Route>
                <Route path="/product-view/:id" exact>
                  <ProductView />
                </Route>
                <Route path="/manga" exact>
                  <Manga
                    mangaProducts={mangaProducts}
                    onAddToCart={handleAddToCart}
                    handleUpdateCartQty
                  />
```

```
                </Route>
                <Route path="/fiction" exact>

                  <Fiction
                    fictionProducts={fictionProducts}
                    onAddToCart={handleAddToCart}
                    handleUpdateCartQty
                  />
                </Route>
                <Route path="/login" exact component={Login}/>
                <Route path="/signup" exact><SignUp/></Route>
                <Route path="/biography" exact>
                  <Biography
                    bioProducts={bioProducts}
                    onAddToCart={handleAddToCart}
                    handleUpdateCartQty
                  />
                </Route>
              </Switch>
            </div>
          </Router>
          <Footer />
        </>
      ) : (
        <div className="loader">
          <img src={loadingImg} alt="Loading" />
        </div>
      )}
      <Toaster/>
    </div>
  );
};

export default App;
```

<u>style.css</u>

```
.loader {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

@media (max-width: 960px) {
  .loader img {
    height: 320px;
  }
}
```

index.html

```
import logo from "../assets/1280.jpg";



function Home() {
    return (
      <div className='grid place-items-center text-richblack-100
text-3xl h-full'>
      <img src={logo} height={200} width={700} loading="lazy"
alt="Logo" />
      </div>
    );
  }

  export default Home;
```

Biography.js

```
mport React from "react";
import { Grid } from "@material-ui/core";
import Product from "../Products/Product/Product.js";
import useStyles from "../Products/styles.js";
import "react-responsive-carousel/lib/styles/carousel.min.css";
import "../ProductView/style.css";

const Biography = ({ onAddToCart, bioProducts }) => {
  const classes = useStyles();

  return (
    <>
      <main className={classes.mainPage}>
        <div className={classes.toolbar} />

        <>
          <div className={classes.categorySection}>
            <h3 className={classes.categoryHeader}>Biographies</h3>
            <h3 className={classes.categoryDesc}>
              Browse our Biographies Collection
            </h3>
            <Grid
              className={classes.categoryFeatured}
              container
              justify="center"
              spacing={1}
            >
              {bioProducts.map((product) => (
                <Grid
                  className={classes.categoryFeatured}
                  item
                  xs={8}
                  sm={5}
                  md={3}
                  lg={2}
                  id="pro"
                >
                  <Product product={product}
onAddToCart={onAddToCart} />
                </Grid>
              ))}
            </Grid>
          </div>
        </>
      </main>
    </>
  );
};
```

```
export default Biography;
```

CartItem.jsx

```
import React from 'react';
import { Typography, Button, Card, CardActions, CardContent,
CardMedia } from '@material-ui/core';

import useStyles from './styles';

const CartItem = ({ item, onUpdateCartQty, onRemoveFromCart }) => {
  const classes = useStyles();

  const handleUpdateCartQty = (lineItemId, newQuantity) =>
onUpdateCartQty(lineItemId, newQuantity);

  const handleRemoveFromCart = (lineItemId) =>
onRemoveFromCart(lineItemId);

  return (
    <Card className="cart-item">
      <CardMedia image={item.media.source} alt={item.name}
className={classes.media} />
      <CardContent className={classes.cardContent}>
        <Typography variant="h6">{item.name}</Typography>
        <Typography variant="h6" color='secondary'
>{item.line_total.formatted_with_symbol}</Typography>
      </CardContent>
      <CardActions className={classes.cardActions}>

        <div className={classes.buttons}>

          <Button type="button" size="small" onClick={() =>
handleUpdateCartQty(item.id, item.quantity - 1)}>-</Button>

          <Typography>&nbsp;{item.quantity}&nbsp;</Typography>

          <Button type="button" size="small" onClick={() =>
handleUpdateCartQty(item.id, item.quantity + 1)}>+</Button>

        </div>

        <Button className={classes.button} variant="contained"
type="button" color='secondary' onClick={() =>
handleRemoveFromCart(item.id)}>Remove</Button>

      </CardActions>

    </Card>

  );

};


export default CartItem;
```

style.js

```
import { makeStyles } from '@material-ui/core/styles';

export default makeStyles(() => ({
  media: {
    height: 0,
    paddingTop: '100%',
  },
  cardContent: {
    display: 'flex',
    justifyContent: 'space-between',
  },
  cartActions: {
    justifyContent: 'space-between',
  },
  buttons: {
    display: 'flex',
    alignItems: 'center',
  },
  button: {
    color: 'white',
    width: '100%',
    height: '40px',

},
}));
```

Cart.js

```
import React from 'react';
import { Container, Typography, Button, Grid } from '@material-
ui/core';
import { Link } from 'react-router-dom';

import CartItem from './CartItem/CartItem';
import useStyles from './styles';

const Cart = ({ cart, onUpdateCartQty, onRemoveFromCart,
onEmptyCart }) => {
  const classes = useStyles();

  const handleEmptyCart = () => onEmptyCart();

  const renderEmptyCart = () => (
    <Typography variant="subtitle1">You have no items in your
shopping cart,
      <Link className={classes.link} to="/"> start adding
some</Link>!
    </Typography>
  );

  if (!cart.line_items) return 'Loading';

  const renderCart = () => (
    <>
      <Grid container spacing={4}>
        {cart.line_items.map((lineItem) => (
          <Grid item xs={12} sm={4} key={lineItem.id}>
            <CartItem item={lineItem}
onUpdateCartQty={onUpdateCartQty}
onRemoveFromCart={onRemoveFromCart} />
          </Grid>
        ))}
      </Grid>
      <div className={classes.cardDetails}>
      <Typography variant="h5" >Subtotal: <b
>{cart.subtotal.formatted_with_symbol}</b></Typography>
        <div>
          <Button className={classes.emptyButton} size="large"
type="button" variant="contained" color="secondary"
onClick={handleEmptyCart}>Empty cart</Button>
          <Button className={classes.checkoutButton}
component={Link} to="/checkout" size="large" type="button"
variant="contained" >Checkout</Button>
        </div>
      </div>
    </>
  );

  return (
    <Container>
      <div className={classes.toolbar} />
      <Typography className={classes.title} variant="h5"
gutterBottom><b>Your Shopping Cart</b></Typography>
      <hr/>
      { !cart.line_items.length ? renderEmptyCart() : renderCart()
}
    </Container>
  );
};

export default Cart;
```

<u>style.js</u>

```
import { makeStyles } from "@material-ui/core/styles";

export default makeStyles((theme) => ({
  toolbar: theme.mixins.toolbar,
  title: {
    marginTop: "3%",
  },
  emptyButton: {
    minWidth: "150px",
    [theme.breakpoints.down("xs")]: {
      marginBottom: "5px",
    },
    [theme.breakpoints.up("xs")]: {
      marginRight: "20px",
    },
  },
  checkoutButton: {
    minWidth: "150px",
    background: "#001524",
    color: "white",
    height: "40px",

    "&:hover": {
      backgroundColor: "#2a344a",
      boxShadow: "none",
      color: "white",
    },
  },
  link: {
    textDecoration: "none",
  },
  cardDetails: {
    display: "flex",
    marginTop: "7%",
    width: "100%",
    justifyContent: "space-between",
  },
}));
```

Checkout.jsx

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

import { CssBaseline, Paper, Stepper, Step, StepLabel, Typography,
CircularProgress, Divider, Button } from '@material-ui/core';

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


import { commerce } from '../../../lib/commerce';

import AddressForm from '../AddressForm';

import PaymentForm from '../PaymentForm';

import useStyles from './styles';


const steps = ['Shipping address', 'Payment details'];


const Checkout = ({ cart, onCaptureCheckout, order, error }) => {

  const [checkoutToken, setCheckoutToken] = useState(null);

  const [activeStep, setActiveStep] = useState(0);

  const [shippingData, setShippingData] = useState({});

  const classes = useStyles();

  const history = useHistory();


  const nextStep = () => setActiveStep((prevActiveStep) => prevActiveStep
+ 1);

  const backStep = () => setActiveStep((prevActiveStep) => prevActiveStep
- 1);


  useEffect(() => {

    if (cart.id) {

      const generateToken = async () => {

        try {

          const token = await commerce.checkout.generateToken(cart.id, {
type: 'cart' });
```

```
          setCheckoutToken(token);
        } catch {
          if (activeStep !== steps.length) history.push('/');
        }
      };

      generateToken();
    }
  }, [cart]);

  const test = (data) => {
    setShippingData(data);

    nextStep();
  };

  let Confirmation = () => (order.customer ? (
    <>
      <div>
        <Typography variant="h5">Thank you for your purchase,
{order.customer.firstname} {order.customer.lastname}!</Typography>
        <Divider className={classes.divider} />
        <Typography variant="subtitle2">Order ref:
{order.customer_reference}</Typography>
      </div>
      <br />
      <Button component={Link} variant="outlined" type="button"
to="/">Back to home</Button>
    </>
  ) : (
    <div className={classes.spinner}>
      <CircularProgress />
    </div>
  ));

  if (error) {
    Confirmation = () => (
      <>
        <Typography variant="h5">Error: {error}</Typography>
        <br />
        <Button component={Link} variant="outlined" type="button"
to="/">Back to home</Button>
      </>
    );
  }

  const Form = () => (activeStep === 0
    ? <AddressForm checkoutToken={checkoutToken} nextStep={nextStep}
setShippingData={setShippingData} test={test} />
    : <PaymentForm checkoutToken={checkoutToken} nextStep={nextStep}
backStep={backStep} shippingData={shippingData}
onCaptureCheckout={onCaptureCheckout} />);

  return (
    <>
      <CssBaseline />
      <div className={classes.toolbar} />
      <main className={classes.layout}>
        <Paper className={classes.paper}>
          <Typography variant="h4" align="center">Checkout</Typography>
          <Stepper activeStep={activeStep} className={classes.stepper}>
            {steps.map((label) => (
              <Step key={label}>
                <StepLabel>{label}</StepLabel>
              </Step>
            ))}
          </Stepper>
          {activeStep === steps.length ? <Confirmation /> : checkoutToken
&& <Form />}
        </Paper>
      </main>
    </>
  );
};

export default Checkout;
```

style.jsx

```
import { makeStyles } from '@material-ui/core/styles';

export default makeStyles((theme) => ({
  appBar: {
    position: 'relative',
  },
  toolbar: theme.mixins.toolbar,
  layout: {
    marginTop: '5%',
    width: 'auto',
    marginLeft: theme.spacing(2),
    marginRight: theme.spacing(2),
    [theme.breakpoints.up(600 + theme.spacing(2) * 2)]: {
      width: 600,
      marginLeft: 'auto',
      marginRight: 'auto',
    },
  },
  paper: {
    marginTop: theme.spacing(3),
    marginBottom: theme.spacing(3),
    padding: theme.spacing(2),
    [theme.breakpoints.down('xs')]: {
      width: '100%',
      marginTop: 60,
    },
    [theme.breakpoints.up(600 + theme.spacing(3) * 2)]: {
      marginTop: theme.spacing(6),
      marginBottom: theme.spacing(6),
      padding: theme.spacing(3),
    },
  },
  stepper: {
    padding: theme.spacing(3, 0, 5),
  },
  buttons: {
    display: 'flex',
    justifyContent: 'flex-end',
  },
  button: {
    marginTop: theme.spacing(3),
    marginLeft: theme.spacing(1),
  },
  divider: {
    margin: '20px 0',
  },
  spinner: {
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
  },
}));
```

AddressForm.jsx

```
import React, { useState, useEffect } from 'react';
import { InputLabel, Select, MenuItem, Button, Grid, Typography }
from '@material-ui/core';
import { useForm, FormProvider } from 'react-hook-form';
import { Link } from 'react-router-dom';

import { commerce } from '../../lib/commerce';
import FormInput from './CustomTextField';

const AddressForm = ({ checkoutToken, test }) => {
  const [shippingCountries, setShippingCountries] = useState([]);
  const [shippingCountry, setShippingCountry] = useState('');

  const [shippingSubdivisions, setShippingSubdivisions] =
useState([]);

  const [shippingSubdivision, setShippingSubdivision] =
useState('');

  const [shippingOptions, setShippingOptions] = useState([]);

  const [shippingOption, setShippingOption] = useState('');

  const methods = useForm();


  const fetchShippingCountries = async (checkoutTokenId) => {

    const { countries } = await
commerce.services.localeListShippingCountries(checkoutTokenId);


    setShippingCountries(countries);

    setShippingCountry(Object.keys(countries)[0]);

  };


  const fetchSubdivisions = async (countryCode) => {

    const { subdivisions } = await
commerce.services.localeListSubdivisions(countryCode);


    setShippingSubdivisions(subdivisions);

    setShippingSubdivision(Object.keys(subdivisions)[0]);

  };


  const fetchShippingOptions = async (checkoutTokenId, country,
stateProvince = null) => {

    const options = await
commerce.checkout.getShippingOptions(checkoutTokenId, { country,
region: stateProvince });


    setShippingOptions(options);

    setShippingOption(options[0].id);
  };

  useEffect(() => {
    fetchShippingCountries(checkoutToken.id);
  }, []);

  useEffect(() => {
    if (shippingCountry) fetchSubdivisions(shippingCountry);
  }, [shippingCountry]);

  useEffect(() => {
    if (shippingSubdivision) fetchShippingOptions(checkoutToken.id,
shippingCountry, shippingSubdivision);
  }, [shippingSubdivision]);

  return (
    <>
      <Typography variant="h6" gutterBottom>Shipping
address</Typography>
      <FormProvider {...methods}>
        <form onSubmit={methods.handleSubmit((data) => test({
...data, shippingCountry, shippingSubdivision, shippingOption }))}>
          <Grid container spacing={3}>
            <FormInput required name="firstName" label="First name"
/>
            <FormInput required name="lastName" label="Last name"
/>
            <FormInput required name="address1" label="Address line
1" />
            <FormInput required name="email" label="Email" />
            <FormInput required name="city" label="City" />
            <FormInput required name="zip" label="Zip / Postal
code" />
```

```
            <Grid item xs={12} sm={6}>
              <InputLabel>Shipping Country</InputLabel>
              <Select value={shippingCountry} fullWidth
onChange={(e) => setShippingCountry(e.target.value)}>
                {Object.entries(shippingCountries).map(([code,
name]) => ({ id: code, label: name })).map((item) => (
                  <MenuItem key={item.id} value={item.id}>
                    {item.label}
                  </MenuItem>
                ))}
              </Select>
            </Grid>
            <Grid item xs={12} sm={6}>
              <InputLabel>Shipping Subdivision</InputLabel>
              <Select value={shippingSubdivision} fullWidth
onChange={(e) => setShippingSubdivision(e.target.value)}>
                {Object.entries(shippingSubdivisions).map(([code,
name]) => ({ id: code, label: name })).map((item) => (
                  <MenuItem key={item.id} value={item.id}>
                    {item.label}
                  </MenuItem>
                ))}
              </Select>
            </Grid>
            <Grid item xs={12} sm={6}>
              <InputLabel>Shipping Options</InputLabel>
              <Select value={shippingOption} fullWidth
onChange={(e) => setShippingOption(e.target.value)}>
                {shippingOptions.map((sO) => ({ id: sO.id, label:
`${sO.description} - (${sO.price.formatted_with_symbol})`
})).map((item) => (
                  <MenuItem key={item.id} value={item.id}>
                    {item.label}
                  </MenuItem>
                ))}
              </Select>
            </Grid>
          </Grid>
          <br />
          <div style={{ display: 'flex', justifyContent: 'space-
between' }}>
            <Button component={Link} variant="outlined"
to="/cart">Back to Cart</Button>
            <Button type="submit" variant="contained"
color="primary">Next</Button>
          </div>
        </form>
      </FormProvider>
    </>
  );
};
export default AddressForm;
```

CustomTextField.jsx

```
import React from 'react';
import { useFormContext, Controller } from 'react-hook-form';
import { TextField, Grid } from '@material-ui/core';

function FormInput({ name, label, required }) {
  const { control } = useFormContext();
  const isError = false;

  return (
    <Grid item xs={12} sm={6}>
      <Controller
        as={TextField}
        name={name}
        control={control}
        label={label}
        fullWidth
        required={required}
        error={isError}
      />
    </Grid>
  );
}

export default FormInput;
```

PaymentForm.jsx

```
import React from "react";
import { Typography, Button, Divider } from "@material-ui/core";
import {
  Elements,
  CardElement,
  ElementsConsumer,
} from "@stripe/react-stripe-js";
import { loadStripe } from "@stripe/stripe-js";

import Review from "./Review";

const stripePromise =
loadStripe(process.env.REACT_APP_STRIPE_PUBLIC_KEY);

const PaymentForm = ({
  checkoutToken,
  nextStep,
  backStep,
  shippingData,
  onCaptureCheckout,
}) => {
  const handleSubmit = async (event, elements, stripe) => {
    event.preventDefault();

    if (!stripe || !elements) return;

    const cardElement = elements.getElement(CardElement);
    const { error, paymentMethod } = await
stripe.createPaymentMethod({
      type: "card",
      card: cardElement,
    });

    if (error) {
      console.log("[error]", error);
    } else {
      const orderData = {
        line_items: checkoutToken.live.line_items,
        customer: {
          firstname: shippingData.firstName,
          lastname: shippingData.lastName,
          email: shippingData.email,
        },
        shipping: {
          name: "International",
          street: shippingData.address1,
          town_city: shippingData.city,
          county_state: shippingData.shippingSubdivision,
          postal_zip_code: shippingData.zip,
          country: shippingData.shippingCountry,
        },
        fulfillment: { shipping_method: shippingData.shippingOption
},
        payment: {
          gateway: "stripe",
          stripe: {
            payment_method_id: paymentMethod.id,
          },
        },
      };

      onCaptureCheckout(checkoutToken.id, orderData);

      nextStep();
    }
  };

  return (
    <>
      <Review checkoutToken={checkoutToken} />
      <Divider />
      <Typography variant="h6" gutterBottom style={{ margin: "20px
0" }}>
        Payment method
      </Typography>
      <Elements stripe={stripePromise}>
        <ElementsConsumer>
          {({ elements, stripe }) => (
            <form onSubmit={(e) => handleSubmit(e, elements,
stripe)}>
              <CardElement />
              <br /> <br />
              <div style={{ display: "flex", justifyContent:
"space-between" }}>
                <Button variant="outlined" onClick={backStep}>
                  Back
                </Button>
                <Button
                  type="submit"
                  variant="contained"
                  disabled={!stripe}
                  style={{ backgroundColor: "#001524", color:
"#FFFF" }}
                >
                  Pay
{checkoutToken.live.subtotal.formatted_with_symbol}
                </Button>
              </div>
            </form>
          )}
        </ElementsConsumer>
      </Elements>
    </>
  );
};

export default PaymentForm;
```

Review.jsx

```
import React from 'react';
import { Typography, List, ListItem, ListItemText } from
'@material-ui/core';

const Review = ({ checkoutToken }) => (
  <>
    <Typography variant="h6" gutterBottom>Order
summary</Typography>
    <List disablePadding>
      {checkoutToken.live.line_items.map((product) => (
        <ListItem style={{ padding: '10px 0' }} key={product.name}>
          <ListItemText primary={product.name}
secondary={`Quantity: ${product.quantity}`} />
          <Typography
variant="body2">{product.line_total.formatted_with_symbol}</Typogra
phy>
        </ListItem>
      ))}
      <ListItem style={{ padding: '10px 0' }}>
        <ListItemText primary="Total" />
        <Typography variant="subtitle1" style={{ fontWeight: 700
}}>
          {checkoutToken.live.subtotal.formatted_with_symbol}
        </Typography>
      </ListItem>
    </List>
  </>
);

export default Review;
```

Fiction.jsx

```
import React from "react";
import { Grid } from "@material-ui/core";
import Product from "../Products/Product/Product.js";
import useStyles from "../Products/styles.js";
import "react-responsive-carousel/lib/styles/carousel.min.css";
import "../ProductView/style.css";

const Fiction = ({ onAddToCart, fictionProducts }) => {
  const classes = useStyles();
  return (
    <>
      <main className={classes.mainPage}>
        <div className={classes.toolbar} />

        <>
          <div className={classes.categorySection}>
            <h3 className={classes.categoryHeader}>
              <span style={{ color: "#f1361d"
}}>Fictional&nbsp;</span>Books
            </h3>
            <h3 className={classes.categoryDesc}>
              Browse our Fictional books Collection
            </h3>
            <Grid
              className={classes.categoryFeatured}
              container
              justify="center"
              spacing={3}
            >
              {fictionProducts.map((product) => (
                <Grid
                  className={classes.categoryFeatured}
                  item
                  xs={8}
                  sm={5}
                  md={3}
                  lg={2}
                  id="pro"
                >
                  <Product product={product}
onAddToCart={onAddToCart} />
                </Grid>
              ))}
            </Grid>
          </div>
        </>
      </main>
    </>
  );
};

export default Fiction;
```

Footer.js

```
import React from "react";
import { MDBCol, MDBContainer, MDBRow, MDBFooter } from "mdbreact";
import logo from "../../assets/circles.png";

const Footer = () => {
  return (
    <MDBFooter color="unique-color-dark" className="font-medium pt-
4 mt-4">
      <MDBContainer className="text-center text-md-left">
        <MDBRow className="text-center text-md-left mt-3 pb-3">
          <MDBCol md="3" lg="3" xl="4" className="mx-auto mt-3">
            <h6 className="text-uppercase mb-4 font-weight-bold">
              <img src={logo} alt="Book Store App" height="50px" />
              <strong>Book-IT</strong>
            </h6>
            <p>
              Book-IT is an online React web application where the
customer can
              purchase books online. Through this book store the
users can
              search for a book by its title and later can add to
the shopping
              cart and finally purchase using credit card
transaction.
            </p>
          </MDBCol>
          <hr className="w-100 clearfix d-md-none" />
          <MDBCol md="2" lg="2" xl="2" className="mx-auto mt-3">
            <h6 className="text-uppercase mb-4 font-weight-bold">
              <strong>Products</strong>
            </h6>
            <p>
              <a href="#">Book-IT</a>
            </p>

          </MDBCol>

          <hr className="w-100 clearfix d-md-none" />
          <MDBCol md="4" lg="3" xl="3" className="mx-auto mt-3">
            <h6 className="text-uppercase mb-4 font-weight-bold">
              <strong>Contact</strong>
            </h6>
            <p>
              <i className="fa fa-envelope mr-3" />
&nbsp;bittujaiswal175@gmail.com, <br/>
              <i className="fa fa-envelope mr-3" />
&nbsp;hritiksribca70@gmail.com, <br/>
              <i className="fa fa-envelope mr-3" />
&nbsp;dushyant2224mca1085@gmail.com,
            </p>
          </MDBCol>
        </MDBRow>
        <hr />
        <MDBRow className="d-flex align-items-center">
          <MDBCol md="8" lg="8">
            <p className="text-center text-md-left grey-text">
              &copy; {new Date().getFullYear()} Made by
              <a href=""> Bittu Jaiswal, Hritik Srivastava,
Dushyant Kaushik </a>
            </p>
          </MDBCol>
        </MDBRow>
      </MDBContainer>
    </MDBFooter>
  );
};

export default Footer;
```

Login.js

```
import React, { useState } from 'react';
import { Link, useHistory } from 'react-router-dom';
import './styles.css';
import toast from 'react-hot-toast';

function Login() {
    const history = useHistory();
    // State to hold username and password input values
    const [username, setUsername] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');

    // Function to handle form submission
    const handleSubmit = async (e) => {
        e.preventDefault();

        try {
            // Send POST request to login API route
            const response = await
fetch('http://localhost:5000/login', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({ email: username, password
}), // Assuming email is used for login
            });

            if (!response.ok) {
                const errorMessage = await response.text();
                throw new Error(errorMessage || 'Login failed');
            }

            // Reset the form after successful login
            setUsername('');
            setPassword('');
            setError('');
            console.log('Login successful');
            history.push("/");
            // Redirect to dashboard or any other page after
successful login
        } catch (error) {
            console.error('Login error:', error);
            toast.error("Error in Login")
            setError(error.message || 'Login failed');
        }
    };

    return (
        <div className="login-container">
            <h2>Login</h2>
            <form onSubmit={handleSubmit}>
                <div className="form-group">
                    <label htmlFor="username">Username:</label>
                    <input
                        type="text"
                        id="username"
                        value={username}
                        onChange={(e) =>
setUsername(e.target.value)}
                        required
                    />
                </div>
                <div className="form-group">
                    <label htmlFor="password">Password:</label>
                    <input
                        type="password"
                        id="password"
                        value={password}
                        onChange={(e) =>
setPassword(e.target.value)}
                        required
                    />
                </div>
                <div style={{ padding: '10px' }}>
                    If not registered,{' '}
                    <Link to="/signup">Click Here</Link> to sign up
                </div>
                <button type="submit">Login</button>
            </form>
        </div>
    );
}
export default Login;
```

style.css

```
.login-container {
    margin-top: 60px;
    margin:auto;
    padding: 40px;
    border: 1px solid #ccc;
    border-radius: 5px;
    background-color: #f9f9f9;
```

```
  /* Center vertically */
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  height: 65vh; /* Adjust as needed */
}

.login-container h2 {
  margin-bottom: 10px;
  text-align: center;
}

.form-group {
  margin-bottom: 15px;
}

.form-group label {
  display: block;
  margin-bottom: 5px;
}

.form-group input {
  width: 100%;
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

button[type="submit"] {
  width: 100%;
  padding: 10px;
  font-size: 16px;
  color: #87c746;
  background-color: #007bff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button[type="submit"]:hover {
  background-color: #0056b3;
  }
```

Manga.js

```
import React from "react";
import { Grid } from "@material-ui/core";
import Product from "../Products/Product/Product.js";
import useStyles from "../Products/styles.js";
import "react-responsive-carousel/lib/styles/carousel.min.css";
import "../ProductView/style.css";

const Manga = ({ onAddToCart, mangaProducts }) => {
  const classes = useStyles();

  return (
    <>
      <main className={classes.mainPage}>
        <div className={classes.toolbar} />

        <>
          <div className={classes.categorySection}>
            <h3 className={classes.categoryHeader}>
              Anime <span style={{ color: "#f1361d" }}>Manga</span>
            </h3>
            <h3 className={classes.categoryDesc}>
              Browse our handpicked selection of manga series
            </h3>
            <Grid
              className={classes.categoryFeatured}
              container
              justify="center"
              spacing={1}
            >
              {mangaProducts.map((product) => (
                <Grid
                  className={classes.categoryFeatured}
                  item
                  xs={6}
                  sm={5}
                  md={3}
                  lg={2}
                  id="pro"
                >
                  <Product product={product}
onAddToCart={onAddToCart} />
                </Grid>
              ))}
            </Grid>
          </div>
        </>
      </main>
    </>
  );
};

export default Manga;
```

Navbar.js

```
import React from "react";
import {
  AppBar,
  Toolbar,
  IconButton,
  Badge,
  Typography,
} from "@material-ui/core";
import { ShoppingCart } from "@material-ui/icons";
import { Link } from "react-router-dom";
import logo from "../../assets/circles.png";
import useStyles from "./styles";

const Navbar = ({ totalItems }) => {
  const classes = useStyles();

  return (
    <div>
      <AppBar position="fixed" className={classes.appBar}
color="inherit">
        <Toolbar>
          <Typography
            component={Link}
```

```
              to="/"
              variant="h5"
              className={classes.title}
              color="inherit"
            >
              <img
                src={logo}
                alt="Book Store App"
                height="50px"
                className={classes.image}
              />
              <div>BookStore & Library</div>
            </Typography>

            <div className={classes.grow} />
            <div className={classes.button}>
              <IconButton
                component={Link}
                to="/cart"
                aria-label="Show cart items"
                color="inherit"
              >
                <Link to=''></Link>
                <Badge badgeContent={totalItems} color="secondary">
                  <ShoppingCart />
                </Badge>
              </IconButton>
            </div>
          </Toolbar>
        </AppBar>
      </div>
    );
  };

export default Navbar;
```

Style.js

```
import { makeStyles } from "@material-ui/core/styles";

const drawerWidth = 0;
```

```
export default makeStyles((theme) => ({
  appBar: {
    color: "white",
    boxShadow: "none",
    background: "#001524",
    borderBottom: "1px solid rgba(0, 0, 0, 0.12)",
    [theme.breakpoints.up("sm")]: {
      width: `calc(100% - ${drawerWidth}px)`,
      marginLeft: drawerWidth,
    },
  },
  title: {
    flexGrow: 1,
    alignItems: "center",
    display: "flex",
    fontFamily: "Raleway",
    fontWeight: 600,
    letterSpacing: 1,
    textDecoration: "none",
    "&:hover": {
      color: "#ffff",
      boxShadow: "none",
    },
  },
  cartt: {
    "&:hover": {
      color: "#ffff",
      boxShadow: "none",
    },
  },
  image: {
    marginRight: "10px",
  },
  menuButton: {
    marginRight: theme.spacing(2),
    [theme.breakpoints.up("sm")]: {
      display: "none",
    },
  },
  grow: {
    flexGrow: 1,
  },
  inputRoot: {
    color: "inherit",
  },
  inputInput: {
    padding: theme.spacing(1, 1, 1, 0),
    paddingLeft: `calc(1em + ${theme.spacing(4)}px)`,
    transition: theme.transitions.create("width"),
    width: "100%",
    [theme.breakpoints.up("md")]: {
      width: "20ch",
    },
  },
}));
```

Product.js

```
import React from "react";
import {
  Card,
  CardMedia,
  CardContent,
  CardActions,
  Typography,
  Button,
  CardActionArea,
} from "@material-ui/core";
import { AddShoppingCart } from "@material-ui/icons";
import { Link } from "react-router-dom";
import useStyles from "./styles";

const Product = ({ product, onAddToCart }) => {
  const classes = useStyles();
  return (
    <Card className={classes.root}>
      <Link to={`product-view/${product.id}`}>
        <CardActionArea>
          <CardMedia
            className={classes.media}
            image={product.media.source}
            title={product.name}
          />
        </CardActionArea>
      </Link>
      <CardContent>
        <div className={classes.cardContent}>
          <p className={classes.cardContentName}>
{product.name}</p>
        </div>
        <div className={classes.cardContent}>
          <p className={classes.cardContentPrice}>
            <b>{product.price.formatted_with_symbol}</b>
          </p>
        </div>
      </CardContent>
      <CardActions disableSpacing className={classes.cardActions}>
        <Button
          variant="contained"
          className={classes.button}
          endIcon={<AddShoppingCart />}
          onClick={() => onAddToCart(product.id, 1)}
        >
          <b>ADD TO CART</b>
        </Button>
      </CardActions>
    </Card>
  );
};

export default Product;
```

<u>Style.js</u>

```
import { makeStyles } from "@material-ui/core/styles";

export default makeStyles(() => ({
  root: {
    maxWidth: "100%",
    background: "linear-gradient(45deg, #D9D9D9 30%, #E6E6E6 90%)",
  },
  media: {
    height: 0,
    paddingTop: "105%",
    "&:hover": {
      backgroundColor: "#2a344a",
      boxShadow: "none",
    },
  },
  cardActions: {
    display: "flex",
    justifyContent: "flex-end",
  },
  cardContent: {
    display: "flex",
    justifyContent: "center",
  },
  button: {
    background: "#001524",
    color: "white",
    width: "100%",
    height: "40px",

    "&:hover": {
      backgroundColor: "#2a344a",
      boxShadow: "none",
    },
  },
  cardContentName: {
    fontSize: 20,
    textAlign: "center",
    margin: "4px !important",
    fontWeight: 500,
  },
  cardContentPrice: {
    fontSize: 20,
    color: "#F1361D",
    margin: "0 !important",
  },
  "@media (max-width: 700px)": {
    cardContentName: {
      fontSize: 14,
      textAlign: "center",
    },
    cardContentPrice: {
      fontSize: 16,
    },
  },
}));
```

Product.jsx

```
import React, { useState, useRef } from "react";
import { Grid, InputAdornment, Input } from "@material-ui/core";
import SearchIcon from "@material-ui/icons/Search";
import Product from "./Product/Product.js";
import useStyles from "./styles";
import logo1 from "../../assets/Bookshop.gif";
import scrollImg from "../../assets/scroll.gif";
import "../ProductView/style.css";
import { Link } from "react-router-dom";
import mangaBg from "../../assets/maxresdefault.jpg";
import bioBg from "../../assets/biography.jpg";
import fictionBg from "../../assets/fiction.jpg";
import "react-responsive-carousel/lib/styles/carousel.min.css";
import { Carousel } from "react-responsive-carousel";

const Products = ({ products, onAddToCart, featureProducts }) => {
  const classes = useStyles();

  const [searchTerm, setSearchTerm] = useState("");

  const sectionRef = useRef(null);

  const handleInputClick = () => {
    // Scrolls to the section when the input is clicked
    sectionRef.current.scrollIntoView({ behavior: "smooth" });
  };

  return (
    <main className={classes.mainPage}>
      <div className={classes.toolbar} />
      <img src={scrollImg} className={classes.scrollImg} />
      <div className={classes.hero}>
        <img className={classes.heroImg} src={logo1} height="720px"
/>

        <div className={classes.heroCont}>
          <h1 className={classes.heroHeader}>
            Discover Your Next Favorite Book Here.
          </h1>
          <h3 className={classes.heroDesc} ref={sectionRef}>
            Explore our curated collection of new and popular books
to find your
            next literary adventure.
          </h3>
          <div className={classes.searchs}>
            <Input
              className={classes.searchb}
              type="text"
              placeholder="Which book are you looking for?"
              onClick={handleInputClick}
              onChange={(event) => {
                setSearchTerm(event.target.value);
              }}
              startAdornment={
                <InputAdornment position="start">
                  <SearchIcon />
                </InputAdornment>
              }
            />
          </div>
        </div>
      </div>

      {searchTerm === "" && (
        <div className={classes.categorySection}>
          <h1 className={classes.categoryHeader}>Categories</h1>
          <h3 className={classes.categoryDesc}>
            Browse our featured categories
          </h3>
          <div className={classes.buttonSection}>
            <div>
              <Link to="manga">
                <button
                  className={classes.categoryButton}
                  style={{ backgroundImage: `url(${mangaBg})` }}
                ></button>
              </Link>
              <div className={classes.categoryName}>Manga</div>
            </div>
            <div>
              <Link to="biography">
                <button
                  className={classes.categoryButton}
                  style={{ backgroundImage: `url(${bioBg})` }}
                ></button>
              </Link>
              <div className={classes.categoryName}>Biography</div>
            </div>
            <div>
              <Link to="fiction">
                <button
                  className={classes.categoryButton}
                  style={{ backgroundImage: `url(${fictionBg})` }}
                ></button>
              </Link>
              <div className={classes.categoryName}>Fiction</div>
            </div>
          </div>
        </div>
      )}

      <div className={classes.carouselSection}>
        <Carousel
          showIndicators={false}
          autoPlay={true}
          infiniteLoop={true}
          showArrows={true}
          showStatus={false}
        >
          <div>
            <Link to="manga">
              <button
                className={classes.categoryButton}
                style={{ backgroundImage: `url(${mangaBg})` }}
              ></button>
            </Link>
            <div className={classes.categoryName}>Manga</div>
          </div>
          <div>
            <Link to="biography">
              <button
                className={classes.categoryButton}
                style={{ backgroundImage: `url(${bioBg})` }}
              ></button>
            </Link>
            <div className={classes.categoryName}>Biography</div>
          </div>
          <div>
            <Link to="fiction">
              <button
                className={classes.categoryButton}
                style={{ backgroundImage: `url(${fictionBg})` }}
              ></button>
            </Link>
            <div className={classes.categoryName}>Fiction</div>
          </div>
        </Carousel>
      </div>

      {searchTerm === "" && (
        <>
          <div>
            <h3 className={classes.contentHeader}>
              Best <span style={{ color: "#f1361d"
}}>Sellers</span>
            </h3>
            <Grid
              className={classes.contentFeatured}
              container
              justify="center"
              spacing={1}
            >
              {featureProducts.map((product) => (
                <Grid
                  className={classes.contentFeatured}
                  item
                  xs={6}
                  sm={5}
                  md={3}
                  lg={2}
                  id="pro"
                >
                  <Product product={product}
onAddToCart={onAddToCart} />
                </Grid>
              ))}
            </Grid>
          </div>
        </>
      )}
```

```
        <div>
          {searchTerm === "" && (
            <>
              <h1 className={classes.booksHeader}>
                Discover <span style={{ color: "#f1361d"
}}>Books</span>
              </h1>
              <h3 className={classes.booksDesc}>
                Explore our comprehensive collection of books.
              </h3>
            </>
          )}
          <div className={classes.mobileSearch}>
            <div className={classes.mobSearchs}>
              <Input
                className={classes.mobSearchb}
                type="text"
                placeholder="Search for books"
                onChange={(event) => {
                  setSearchTerm(event.target.value);
                }}
                startAdornment={
                  <InputAdornment position="start">
                    <SearchIcon />
                  </InputAdornment>
                }
              />
            </div>
          </div>
          <Grid
            className={classes.content}
            container
            justify="center"
            spacing={2}
          >
            {products
              .filter((product) => {
                if (searchTerm === "") {
                  return product;
                } else if (
                  product.name
                    .toLowerCase()
                    .includes(searchTerm.toLocaleLowerCase())
                ) {
                  return product;
                }
              })
              .map((product) => (
                <Grid
                  className={classes.content}
                  item
                  xs={6}
                  sm={6}
                  md={4}
                  lg={3}
                  id="pro"
                >
                  <Product product={product}
  onAddToCart={onAddToCart} />
                </Grid>
              ))}
          </Grid>
        </div>
      </main>
    );
  };

  export default Products;
```

Style.js

```
import { makeStyles } from "@material-ui/core/styles";

export default makeStyles((theme) => ({
  toolbar: theme.mixins.toolbar,
  mainPage: { flexGrow: 1, overflowX: "hidden", overflowY: "hidden"
},
  content: {
    flexGrow: 1,
    backgroundColor: theme.palette.background.default,
    padding: theme.spacing(20),
    paddingTop: theme.spacing(2),
  },
  hero: {
    flexDirection: "column",
    height: "90vh",
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    padding: theme.spacing(20),
    backgroundColor: "white",
  },
  heroHeader: {
    textAlign: "center",
    color: "#001524",
    fontSize: 68,
    fontFamily: "Poppins",
    fontWeight: "800",
    letterSpacing: -3,
    lineHeight: 0.9,
    wordSpacing: 8,
    width: 660,
    paddingBottom: 8,
  },
  heroDesc: {
    textAlign: "center",
    color: "#455A64",
    fontSize: 24,
    fontFamily: "Raleway",
    paddingBottom: 28,
    width: 584,
  },

  contentHeader: {
    textAlign: "center",
    color: "#FFF",
    fontSize: 40,
    fontFamily: "Poppins",
    fontWeight: "bolder",
    paddingTop: theme.spacing(5),
    backgroundColor: "#001524",
    margin: "0 !important",
    letterSpacing: "-.8px",
    wordSpacing: "4px",
  },

  contentFeatured: {
    gap: 15,
    padding: theme.spacing(5),
    paddingTop: theme.spacing(6),
    paddingBottom: theme.spacing(10),
    backgroundColor: "#001524",
  },
  categoryFeatured: {
    gap: 15,
    padding: theme.spacing(5),
    paddingTop: theme.spacing(6),
    paddingBottom: theme.spacing(10),
    backgroundColor: "#FFF",
  },

  carouselSection: {
    display: "none",
  },

  buttonSection: {
    display: "flex",
    justifyContent: "center",
    alignItems: "center",
    textAlign: "center",
    backgroundColor: "#FFF",
    gap: 28,
    paddingBottom: 28,
  },

  categorySection: {
    backgroundColor: "#FFF",
    paddingTop: theme.spacing(8),
    paddingBottom: theme.spacing(8),
  },

  categoryName: {
    fontFamily: "Poppins",
    color: "#001524",
    fontSize: 20,
    fontWeight: 500,
  },

  categoryButton: {
    fontFamily: "Poppins",
    width: 280,
    height: 280,
    color: "#FFF",
    borderRadius: 8,
    fontSize: 60,
    border: "none",
    backgroundSize: "cover",
    backgroundPosition: "center",
    "&:hover": { opacity: 0.8, transition: "ease-in-out .4s" },
  },

  categoryHeader: {
    textAlign: "center",
    color: "#001524",
    fontSize: 40,
    fontFamily: "Poppins",
    fontWeight: "bolder",
    letterSpacing: "-.8px",
    wordSpacing: "4px",
  },

  categoryDesc: {
    textAlign: "center",
    color: "#455A64",
    fontSize: 20,
    paddingBottom: theme.spacing(2),
    fontFamily: "Raleway",
  },

  root: {
    flexGrow: 1,
  },
  searchs: {
    justifyContent: "center",
    display: "flex",
  },
  searchb: {
    backgroundColor: "white",
    height: "80%",
    width: "60%",
    padding: "12px",
    borderRadius: "6px",
    border: "1px solid #001524",
  },
  booksHeader: {
    textAlign: "center",
    color: "#001524",
    fontSize: 40,
    fontFamily: "Poppins",
    fontWeight: "bolder",
    paddingTop: theme.spacing(10),
    letterSpacing: "-.8px",
    wordSpacing: "4px",
  },
  booksDesc: {
    textAlign: "center",
    color: "#455A64",
    fontSize: 20,
    paddingBottom: theme.spacing(2),
    fontFamily: "Raleway",
  },
  scrollImg: {
    position: "absolute",
    right: 0,
    bottom: 40,
    height: 100,
  },
  mobileSearch: {
    display: "none",
  },
  "@media (max-width: 1600px)": {
    hero: {
      flexDirection: "column",
      height: "100vh",
      gap: 0,
      paddingTop: 0,
      justifyContent: "center",
    },
    heroHeader: {
      textAlign: "center",
      color: "#001524",
      fontSize: 60,
      fontFamily: "Poppins",
      fontWeight: "800",
      letterSpacing: -2,
      lineHeight: 1,
      wordSpacing: 4,
      width: 600,
      paddingBottom: 8,
    },
    heroDesc: {
      textAlign: "center",
      color: "#455A64",
      fontSize: 24,
      fontFamily: "Raleway",
      paddingBottom: 28,
      width: 584,
    },
    searchs: {
      justifyContent: "center",
    },
    heroImg: {
      height: 480,
    },
    content: {
      padding: 8,
    },
  },
  "@media (max-width: 700px)": {
    hero: {
      flexDirection: "column",
      height: "100vh",
      gap: 20,
      paddingTop: 0,
    },
    heroCont: {
      display: "flex",
      justifyContent: "center",
      alignItems: "center",
      flexDirection: "column",
    },
    heroHeader: {
      textAlign: "center",
      color: "#001524",
      fontSize: 32,
      fontFamily: "Poppins",
      fontWeight: "800",
      letterSpacing: -1.2,
      lineHeight: 1,
      wordSpacing: 4,
      width: 332,
      paddingBottom: 8,
    },
    heroDesc: {
```

```
    textAlign: "center",
    color: "#455A64",
    fontSize: 14,
    fontFamily: "Raleway",
    paddingBottom: 28,
    width: 320,
  },
  contentHeader: {
    fontSize: 32,
  },
  booksDesc: {
    fontSize: 12,
  },
  booksHeader: {
    fontSize: 32,
  },
  searchs: {
    display: "none",
  },
  mobileSearch: {
    display: "block",
    padding: 32,
    paddingTop: 20,
  },
  mobSearchs: {
    justifyContent: "center",
    display: "flex",
  },
  mobSearchb: {
    backgroundColor: "white",
    height: "80%",
    width: "80%",
    padding: "12px",
    borderRadius: "5px",
    border: "1px solid #001524",
  },
  heroImg: {
    height: 280,
  },
  content: {
    padding: 4,
  },
  contentFeatured: {
    gap: 0,
      padding: theme.spacing(0),
      paddingTop: theme.spacing(4),
      paddingBottom: theme.spacing(7),
    },
    categoryFeatured: {
      gap: 0,
      padding: theme.spacing(0),
      paddingTop: theme.spacing(4),
      paddingBottom: theme.spacing(7),
    },
    scrollImg: {
      position: "absolute",
      textAlign: "center",
      margin: "auto",
      left: 0,
      right: 0,
      bottom: 20,
      height: 100,
    },
    carouselSection: {
      display: "block",
      backgroundColor: "#FFF",
    },
    buttonSection: {
      display: "none",
    },
    categoryHeader: {
      fontSize: 32,
    },

    categoryDesc: {
      fontSize: 14,
    },
  },
}));
```

ProductView.js

```
import React from "react";
import { Container, Grid, Button, Typography } from "@material-
ui/core";
import { Link } from "react-router-dom";
import { commerce } from "../../lib/commerce";
import { useState, useEffect } from "react";
import "./style.css";

const createMarkup = (text) => {
  return { __html: text };
};

const ProductView = () => {
  const [product, setProduct] = useState({});

  const fetchProduct = async (id) => {
    const response = await commerce.products.retrieve(id);
    console.log({ response });
    const { name, price, media, quantity, description } = response;
    setProduct({
      name,
      quantity,
      description,
      src: media.source,
      price: price.formatted_with_symbol,
    });
  };

  useEffect(() => {
    const id = window.location.pathname.split("/");
    fetchProduct(id[2]);
  }, []);

  return (
    <Container className="product-view">
      <Grid container>
        <Grid item xs={12} md={6} className="image-wrapper">
          <img src={product.src} alt={product.name} />
        </Grid>
        <Grid item xs={12} md={5} className="text">
          <Typography variant="h2">
            <b>{product.name}</b>
          </Typography>
          <Typography
            variant="p"
dangerouslySetInnerHTML={createMarkup(product.description)}
          />
          <Typography variant="h3" color="secondary">
            Price: <b> {product.price} </b>
          </Typography>
          <br />
          <Grid container spacing={0}>
            <Grid item xs={12}>
              <Button
                size="large"
                className="custom-button"
                component={Link}
                to="/"
              >
                Continue Shopping
              </Button>
            </Grid>
          </Grid>
        </Grid>
      </Grid>
    </Container>
  );
};

export default ProductView;
```

Style.css

```
.product-view {
  background-color: white;
  border-radius: 8px;
  box-shadow: rgba(50, 50, 93, 0.25) 0px 2px 5px -1px,
    rgba(0, 0, 0, 0.3) 0px 1px 3px -1px;
}

.MuiContainer-root.product-view {
  margin-top: 100px;
  display: flex;
  justify-content: center;
  align-items: center;
```

```css
}

.product-view .image-wrapper img {
  display: flex;
  justify-content: center;
  align-items: center;
  padding-top: 50px;
  width: 80%;
}

.product-view .text {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  color: #000;
}

.product-view .text h2 {
  color: #001524;
  font-size: 35px;
  padding-bottom: 30px;
}

.product-view .text h3 {
  color: #000;
  font-size: 18px;
  font-weight: 300;
  margin-bottom: 20px;
}

.MuiContainer-root.product-view .custom-button {
  color: white;
  background-color: #001524;
  display: flex;
  justify-content: center;
  align-items: center;
}

.MuiContainer-root.product-view .custom-button:hover {
  color: white;
  background-color: #2a344a;
}
```

```css
.product-view .MuiGrid-root .MuiGrid-item {
  padding: 5px 15px;
}

.carousel-caption {
  top: 45%;
}

.d-block {
  height: 50vh;
  opacity: 90%;
}

.carousel-control-next {
  display: none !important;
}

.carousel-control-prev {
  display: none !important;
}

.carousel-indicators .active {
  display: none !important;
}

@media (min-width: 960px) {
  .product-view {
    height: 100vh;
  }
}
```

Signup.js

```javascript
import React, { useState } from 'react';
import './styles.css'
import {Link} from 'react-router-dom'
import toast from 'react-hot-toast';
function Signup() {
    const [username, setUsername] = useState('');
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [confirmPassword, setConfirmPassword] = useState('');
    const [error, setError] = useState('');
    const handleSubmit = async (e) => {
      e.preventDefault();
      // Perform form validation
      if (!username || !email || !password || !confirmPassword) {
        setError('All fields are required');
        toast.error("All fields are required")
        return;
      }
      if (password !== confirmPassword) {
        setError("Passwords don't match");
        toast.error("Passwords don't match")
        return;
      }

      try {
        // Send registration data to the backend API
        const response = await
fetch('http://localhost:5000/signup', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
          body: JSON.stringify({ username, email, password }),
        });

        // Check if the request was successful
        if (response.ok) {
          // Reset form fields and errors after successful
registration
          setUsername('');
          setEmail('');
          setPassword('');
          setConfirmPassword('');
          setError('');
          toast.success("User registered successfully")
        } else {
          // If the request fails, throw an error
          throw new Error('Registration failed');
        }
      } catch (error) {
        console.error('Error registering user:', error);
        setError('Registration failed. Please try again.');
      }
    };
```

```
  return (
    <div className="signup-container">
      <h2>Sign Up</h2>
      {error && <div className="error">{error}</div>}
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="username">Username:</label>
          <input
            type="text"
            id="username"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="email">Email:</label>
          <input
            type="email"
            id="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="password">Password:</label>
          <input
            type="password"
            id="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="confirmPassword">Confirm
Password:</label>
          <input
            type="password"
            id="confirmPassword"
            value={confirmPassword}
            onChange={(e) => setConfirmPassword(e.target.value)}
            required
          />
        </div>

        <div style={{padding:"10px"}}>Already Registered <Link
to='/login'>Click Here </Link> to Login</div>
        <button type="submit">Sign Up</button>
      </form>
    </div>
  );
}

export default Signup;
```

Style.css

```
.signup-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    margin-top: 30px;
    height: 90vh; /* Adjust height as needed */
    margin: 0 auto; /* Center-align along the X-axis */
    border: 1px solid #ccc; /* Add border */
    border-radius: 5px; /* Add border radius */
    padding: 20px; /* Add padding for spacing */
  }

  .signup-container h2 {
    margin-bottom: 20px;
  }

  .form-group {
    margin-bottom: 15px;
  }

  .form-group label {
    display: block;
    margin-bottom: 5px;
  }
  .form-group input {
    width: 300px; /* Adjust width as needed */
    padding: 10px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 5px;
  }

  button[type="submit"] {
    width: 300px; /* Adjust width as needed */
    padding: 10px;
    font-size: 16px;
    color: #fff;
    background-color: #007bff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
  }

  button[type="submit"]:hover {
    background-color: #0056b3;
  }
```

Commerce.js

```
import Commerce from '@chec/commerce.js';

export const commerce = new
Commerce(process.env.REACT_APP_CHEC_PUBLIC_KEY, true);
```

userModel.js

```
// user.js

const mongoose = require('mongoose');
```

```
// Define user schema
const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
});

// Create and export User model
const User = mongoose.model('User', userSchema);

module.exports = User;
```

server.js

```
const express = require('express');
const mongoose = require('mongoose');
const User = require('./models/userModel'); // Import User model
const cors=require('cors')
const app = express();
app.use(cors());
// Middleware for JSON parsing
app.use(express.json());

// Connect to MongoD
mongoose.connect('mongodb://localhost:27017/my_database', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const db = mongoose.connection;
```

```
db.on('error', console.error.bind(console, 'MongoDB connection
error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

// Signup route
app.post('/signup', async (req, res) => {
  try {
    const { username, email, password } = req.body;

    // Check if email already exists
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: 'Email already exists'
});
    }

    // Create new user
    const newUser = new User({ username, email, password });
    await newUser.save();

    res.status(201).json({ message: 'User registered successfully'
});
  } catch (error) {
    console.error('Error registering user:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
});

app.post('/login', async (req, res) => {
    try {
      const { email, password } = req.body;

      // Check if user with the provided email exists
      const user = await User.findOne({ email });
      if (!user) {
        return res.status(404).json({ message: 'User not found' });
      }

      // Check if the provided password matches the user's password
      if (password !== user.password) {
        return res.status(401).json({ message: 'Invalid password'
});
      }

      // If authentication is successful, send a success response
      res.status(200).json({ message: 'Login successful' });
    } catch (error) {
      console.error('Error logging in user:', error);
      res.status(500).json({ message: 'Internal server error' });
    }
  });


// Start the server
app.listen(5000, () => {
  console.log('Server is running on port 5000');
});
```

CHAPTER 10

TESTING

9.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

9.2 Types of Testing

9.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

9.2.2 Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

9.2.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT

- Further enhancements of this project will be user can also sell the old book
- Another enhancement would be user give feedback.

CONCLUSION & REFERNCES

The Bookstore and Library Project using the MERN stack aims to modernize the way books are managed and accessed, offering a comprehensive solution that meets the needs of both users and administrators. By integrating advanced technologies, this project promises to deliver a state-of-the-art platform that supports learning and literacy in the digital age.

Keeping in view these facts we will develop successfully. Developing the project will help us some experience on website development.

References

Coding phase: **-**

1. React

Referenced Sites:

www.w3school. com

www. https://react.dev/learn