**A PROJECT REPORT ON**

**Comprehensive E-Commerce Platform Website**

**For online book sales**

**A Report Submitted to**

**Blackbuck Engineers Pvt. Ltd**

**Submitted By**

**B. Yogeswari (228X1A0518)**

**N. Mohan (Roll No: 208X1A0576)**

**B. Sudharshan (Roll No: 208X1A0508)**

**P. Dhanush (Roll No: 208X1A0579)**

**B. Surendra (Roll No: 208X1A0516)**

**G. Mounika (Roll No: 208X1A0540)**

**S. Prabhu Kumar (Roll No: 218X5A0506)**



**Blackbuck Engineers Pvt. Ltd**

**Road no 36, Jubilee Hills,**

**Hyderabad.**

**ACKNOWLEDGEMENT**

We extend our heartfelt appreciation to Black Bucks Academy and Ram Kumar Sir for their invaluable training and mentorship in MERN Stack development. Their expertise and guidance have been instrumental in equipping our team with the skills needed to bring project “Tourism Places” to life.

We are also deeply thankful to Ranjith Reddy Sir, our placement officers, for providing us with the opportunity to undertake this training Your support, dedication to our career growth, and belief in our vision have been crucial in turning this endeavor into a reality.

This project would not have been possible without the knowledge and opportunities provided by Black Bucks Academy, the guidance of Ram Kumar Sir, and the trust and support of Placement Officer. We are sincerely grateful for their contributions to the success of our project on “Toursim Places”

**ABSTRACT**

E-commerce has revolutionized the way people shop, offering convenience, accessibility, and a wide array of products at their fingertips. "E-Commerce360" aims to create a comprehensive e-commerce platform to cater to the evolving needs of online shoppers and businesses alike. This full stack project integrates various functionalities to provide a seamless shopping experience while empowering merchants with robust management tools.

**KEY FEATURES:**

**User Management:**

Registration and login for customers, enabling personalized experiences.

Profile management to update personal information, addresses, and payment methods.

**Product Catalog:**

Extensive product listings categorized for easy navigation.

Advanced search and filtering options for precise product discovery.

**Shopping Cart and Checkout:**

Secure shopping cart to add and manage selected items.

Smooth checkout process with multiple payment gateways for flexibility.

**Order Management:**

Order history and tracking to monitor the status of past and current orders.

Integration with shipping providers for real-time shipping updates.

**Merchant Dashboard:**

Registration and onboarding for sellers with verification processes.

Product management tools to add, edit, and remove listings efficiently.

**Analytics and Reporting:**

Sales analytics dashboard for both customers and merchants.

Detailed reports on sales, inventory, and customer behavior for informed decision-making.

**Customer Support:**

Help desk functionality for customer queries and issue resolution.

Integration with live chat and ticketing systems for real-time assistance.

**Technology Stack:**

**Frontend:** HTML, CSS, JavaScript, React.js

**Backend:** Node.js, Express.js, MongoDB

**Authentication**: JSON Web Tokens (JWT)

**Customer Support:** LiveChat API, Zendesk Integration

**Payment** Integration: Stripe, PayPal

**Analytics**: Google Analytics, Custom Dashboard Development

INTRODUCTION

E-commerce is fast gaining ground as an accepted and used business paradigm. More and more business houses are implementing web sites providing functionality for performing commercial transactions over the web. It is reasonable to say that the process of shopping on the web is becoming commonplace. The objective of this project is to develop a generalpurpose e-commerce store where any product (such as books, CDs, computers, mobile phones, electronic items, and home appliances) can be bought from the comfort of home through the Internet. However, for implementation purposes, this paper will deal with an online book store.

An online store is a virtual store on the Internet where customers can browse the catalog and select products of interest. The selected items may be collected in a shopping cart. At checkout time, the items in the shopping cart will be presented as an order. At that time, more information will be needed to complete the transaction. Usually, the customer will be asked to fill or select a billing address, a shipping address, a shipping option, and payment information such as credit card number. An e-mail notification is sent to the customer as soon as the order is placed.

Our project is an online e-commerce platform, harmonizing modern technologies like HTML, CSS, JavaScript, and Bootstrap to provide a seamless and user-friendly shopping experience. It grants customers the freedom to explore an extensive array of furniture selections, all from the comfort of their homes.

**Advantages of a full-stack E-Commerce platform:**

**1.Customization**: Tailor-made to match business needs and branding.

**2.Scalability**: Easily grows with business, handling increased traffic and transactions.

**3.Integration**: Seamless integration with third-party services enhances functionality.

**4.Performance**: Optimization ensures fast loading times and smooth user interactions.

**5.Security**: Robust measures at every layer protect against threats and ensure safe transactions.

**6.Flexibility**: Ability to choose the best technologies for each layer of the application stack.

**7.Control**: Full control over development process, timeline, and feature prioritization.

**8.Cost-effectiveness**: Long-term savings compared to pre-built solutions with recurring fees.

**9.Ownership**: Complete control of codebase, data, and intellectual property.

**10.Competitive advantage**: Offers differentiation, unique features, and adaptability in the market.Top of Form

**The potential disadvantages of a full-stack E-Commerce Platform:**

**1.Complexity**: Multi-layered development increases multiple

Complexity.

**2.Time-consuming**: Custom development takes longer than pre-built solutions.

**3.Higher upfront costs**: Initial investment in resources can be significant.

**4.Maintenance overhead**: Requires ongoing effort to maintain and update.

**5.Security risks**: Managing security across layers is critical.

**6.Dependency on expertise**: Reliance on internal developers for maintenance.

**7.Feature gaps**: May lack some features compared to pre-built solutions.

**8.Limited support**: Internal resources handle troubleshooting.

**9.Longer time to market**: Development may take more time to launch.

**10. Risk of over-engineering**: Potential for unnecessary complexity and costs.

**SOFTWARE AND HARDWARE REQUIREMENTS**

**Software Requirements:**

**Operating System:**

Linux, preferably Ubuntu Server or CentOS, for hosting the web server.

**Web Server:**

Apache or Nginx to serve web pages to users.

**Database Management System**:

MySQL or PostgreSQL for storing product information, user data, and order details.

**Programming Languages and Frameworks:**

**Frontend:**

HTML, CSS, JavaScript (ES6) for building user interfaces.

Frontend Frameworks like React.js or Vue.js for dynamic web application development.

**Backend:**

Node.js, Python (with Django or Flask), or Ruby (with Ruby on Rails) for server-side logic.

**Database:**

Sequelize (for Node.js), Django ORM (for Python), or ActiveRecord (for Ruby) for interacting with the database.

**Payment Gateway Integration:**

Integration with payment gateways like PayPal, Stripe, or Square for processing online payments securely.

**Security Measures:**

Implementation of SSL/TLS certificates for secure data transmission (HTTPS).

User authentication and authorization mechanisms.

Input validation and sanitization to prevent common security vulnerabilities like SQL injection and cross-site scripting (XSS).

**Content Delivery Network (CDN):**

Integration with CDN services like Cloudflare or AWS CloudFront for faster content delivery and DDoS protection.

**Version Control:**

Git for version control and collaboration among developers.

**Hardware Requirements:**

**Server Hardware:**

Virtual Private Server (VPS) or Dedicated Server with sufficient computing resources:

**CPU:** least a dual-core processor (4 cores or more recommended for high traffic).

**RAM:** Minimum 2 GB (4 GB or more recommended).

**Storage:** SSD storage for faster data access.

Alternatively, cloud-based services like AWS EC2, Google Cloud Compute Engine, or Microsoft Azure VMs can be used.

**Database Server:**

Separate server for hosting the database, especially for larger-scale applications.

Similar specifications as the web server but with higher RAM allocation depending on the database size and traffic.

**Backup Solutions:**

Regular backups of the database and website files to ensure data integrity and disaster recovery.

**Load Balancer (Optional):**

For distributing incoming traffic across multiple servers to improve performance and reliability, especially for high-traffic websites.

**Monitoring Tools:**

Monitoring software like Prometheus, Grafana, or New Relic for tracking server performance, uptime, and application health.

**Scalability Considerations:**

Architecture designed for horizontal scalability to accommodate increasing traffic and user demand.

Ensure that the chosen hardware and software components meet the scalability, security, and performance requirements of the e-commerce platform for online book sales. Regular maintenance, updates, and security audits are essential for maintaining the system's integrity and protecting user data.

This section describes the software and hardware requirements of the system.

**Online registration for students.**

Individual member accounts for accessing information.

Comprehensive book details management, including authors, number of copies, availability status, etc.

Management of issued books, return dates, and fines for delays.

Administrator privileges for adding and updating books.

Reduced time consumption, improved accuracy, and reliability.

**MOTIVATION**

"HomeBlissFurniture is born from a profound appreciation for the transformative power of furniture in our homes and a deep desire to share the joy of creating spaces that truly resonate with our customers. Furniture has the unique ability to turn houses into homes, where cherished memories are made, and lifestyles are expressed. This project is a dedication to enhancing your everyday life, a commitment to convenience, and a testament to our passion for creating innovative online experiences.

We aim to simplify the furniture-buying journey, inspire creativity, and connect people with the potential of their living spaces. HomeBlissFurniture isn't just a website; it's a digital canvas for your lifestyle, crafted with love and unwavering dedication to enrich your home. Welcome to a world where furniture becomes a reflection of you."

**LITERATURE REVIEW**

**1.E-commerce Market Dynamics:**

The e-commerce industry has seen remarkable growth and transformation, influenced by shifting consumer preferences, technological advancements, and the impact of the digital economy (eMarketer). An understanding of these dynamics is crucial for effectively positioning your furniture e-commerce business in a competitive market.

**2.Responsive Web Design and User Experience:**

Research emphasizes the importance of responsive web design to ensure seamless and userfriendly experiences across various devices, including mobile, which is increasingly becoming the preferred platform for online shopping (Nielsen Norman Group). Adhering to these 1principles can enhance your website's accessibility and user satisfaction.

**3.Product Visualization and Information:**

Studies show that high-quality product images, 3D visualization, and comprehensive product descriptions significantly influence the purchasing decisions of online shoppers. A well-designed product presentation can provide a more immersive and informed shopping experience (Forrester Research). 4.User-Generated Content and Social Proof: Integrating customer reviews, ratings, and testimonials into your website can build trust and credibility. Research indicates that shoppers often rely on the experiences and opinions of others when making purchase decisions (BrightLocal). Encouraging and showcasing user-generated content can bolster your customers' confidence in their furniture selections.

**Existing System:**

* **Traditional Furniture Shopping:**

In the existing system, customers rely on physical furniture stores to browse and purchase furniture. They need to visit the store in person, which may involve time-consuming trips and geographic constraints.

* **Limited Product Selection:**

Physical stores often have limited floor space, which restricts the variety of furniture available for customers to choose from. This limitation can result in customers not finding the exact pieces they desire.

* **Manual Inventory Management:**

Traditional furniture stores often manage their inventory manually, which can lead to discrepancies, stockouts, and challenges in keeping track of product availability.

* **Limited Online Presence:**

Many traditional furniture stores lack a strong online presence, making it difficult for customers to explore their offerings, check prices, or make informed decisions online.

* **Seasonal Sales Fluctuations:**

Traditional furniture stores may experience seasonal fluctuations in sales, leading to inventory management challenges and the need to offer discounts during off-peak seasons.

**Proposed System:**

The proposed system for the HouseBlissFurniture e-commerce project encompasses the following key elements:

**User-Friendly Website:** The central component is a user-friendly e-commerce website designed to provide customers with an intuitive and enjoyable shopping experience. It should feature an attractive and responsive design, making it accessible from various devices and browsers.

**Product Catalog:** The website will have a comprehensive catalog of various flower varieties, arrangements, and related products. Each product listing should include high-quality images, detailed descriptions, and pricing information.

**Secure Checkout Process:** The checkout process should be secure, simple, and efficient. It should include options for payment methods, delivery addresses, and delivery date selection. Integration with secure payment gateways is essential for handling transactions.

**Delivery and Shipping:** Implement a system for setting delivery zones, calculating shipping costs, and providing estimated delivery times. Customers should receive order tracking information.

**Customer Support:** Offer multiple channels for customer support, including live chat, email, or a dedicated support page. Promptly address customer inquiries and issues.

**Keywords and Definitions:**

**1. E-commerce (Electronic Commerce):** E-commerce refers to the buying and selling of goods or services over the internet. It encompasses various online activities, including online shopping, electronic payments, and online marketplaces.

**2. Website:** A website is a collection of web pages accessible through the internet. It serves as the online storefront for e-commerce businesses, providing information about products or services and facilitating transactions.

**3. Online Shopping:** Online shopping is the process of purchasing products or services through a website or app. It allows customers to browse, select, and buy items without visiting a physical store.

**1. Product Catalog:** A product catalog is an organized listing of the products or services offered by an e- commerce business. It includes product descriptions, prices, and images.

**2. User Account:** A user account is a personal and secure online account that customers create on an e- commerce website. It allows them to store personal information, track orders, and receive updates.

**3. Inventory Management:** Inventory management involves tracking and controlling the quantity and availability of products in stock. E-commerce businesses use software to manage their inventory efficiently.

**Implementation:**

**● HTML, CSS, JavaScript:** Build the user interface using these fundamental web technologies for layout, styling, and interactivity.

**● Responsive Design:** Ensure the website is responsive to different screen sizes and devices.

**● Bootstrap:** Utilize Bootstrap or similar frameworks for a responsive and consistent design.

**● Client-Side Frameworks:** Consider using JavaScript frameworks like React or Vue.js for dynamic user interfaces.

**Challenges Faced:**

* Limited market reach and customer base due to physical store limitations.
* Inability to provide a wide variety of flowers and convenience to customers.
* Manual processes for inventory management and order processing.

**System Testing:**

Limited marketing and online presence, missing out on the growing e-commerce market. Software testing is an important element of software quality assurance and represents the ultimate review of specification, design and coding. It increasing visibility of software as a system element and the costs associates with a software failure are motivating forces for all well planned through testing. The system is tested with giving wrong information. Cascade deletion and, the software developer checks updating. Testing and debugging are different activities, but debugging must be accommodated in any testing strategy.

**UNIT TESTING:**

Unit testing is the important and major part of the project. So errors can be rectified easily in each module and program clarity can be increased. In this project, the entire system is divided into several modules and is developed individually. Hence, unit testing is conducted to individual modules.

**INTEGRATION TESTING:**

Integration testing is the systematic technique for constructing the program structure while conducting tests to uncover errors associated with integrating. After the unit test, each module is gradually integrated to form one final system. All the modules when unit tested will work properly but after integrating the data can cause error one module can have an inadvertent, adverse effect on another; sub functions when combined may not produce the desired major function; global data structures can cause problems, etc. In this project the integration testing is performed by combing login, civil registration and status modules are generated the report.

**PERFORMANCE TESTING:**

A type of Physical test covering a wide range of engineering or functional evaluations where a material, product, or system is not specified by detailed material or component specifications: rather, emphasis is on the final measurable performance characteristics. Testing can be a qualitative or quantitative procedure.

**ACCEPTANCE TESTING:**

The User Acceptance testing focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CODING**

**SEARCH FOR BOOKS.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Online Bookstore</title>

<style>

table {

width: 100%;

border-collapse: collapse;

}

th, td {

border: 1px solid #dddddd;

text-align: left;

padding: 8px;

}

th {

background-color: #f2f2f2;

}

add-to-cart-button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 8px 16px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 14px;

margin: 4px 2px;

cursor: pointer;

border-radius: 4px;

}

search-container {

margin-bottom: 10px;

}

nav ul {

list-style-type: none;

padding: 0;

margin: 0;

background-color: #f2f2f2;

overflow: hidden;

}

nav li {

float: left;

}

nav li a {

display: block;

color: #333;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

nav li a: hover {

background-color: #ddd;

}

</style>

</head>

<body>

<nav>

<ul>

<li><a href="#">Home</a></li>

<li><a href="#">Books</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

<h1>Online Bookstore</h1>

<div class="search-container">

<input type="text" id="searchInput" onkeyup="searchBooks ()" placeholder="Search for book titles...">

</div>

<table id="booksTable">

<thead>

<tr>

<th>Book Title</th>

<th>Author</th>

<th>Quantity Available</th>

<th>Price</th>

<th>Add to Cart</th>

</tr>

</thead>

<tbody>

<tr>

<td>Book 1</td>

<td>Author 1</td>

<td>10</td>

<td>$10.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>Book 2</td>

<td>Author 2</td>

<td>8</td>

<td>$12.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>Book 3</td>

<td>Author 3</td>

<td>5</td>

<td>$8.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>Book 4</td>

<td>Author 4</td>

<td>12</td>

<td>$14.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>Book 5</td>

<td>Author 5</td>

<td>7</td>

<td>$9.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

</tbody>

</table>

**SEARCH FOR BOOKS.JS**

<script>

function search Books () {

var input, filter, table, tr, td, i, txtValue;

input = document.get ElementById("searchInput");

filter = input.value.to UpperCase ();

table = document. getElement ById("booksTable");

tr = table. Get Elements ByTagName("tr");

for (i = 0; i < tr. length; i++) {

td = tr[i]. get Elements ByTagName("td") [0];

if (td) {

txtValue = td.text Content || td. Inner Text;

if (txtValue.toUpperCase(). Index Of(filter) > -1) {

tr[i]. style. display = "";

} else {

tr[i]. style. display = "none";

}

}

}

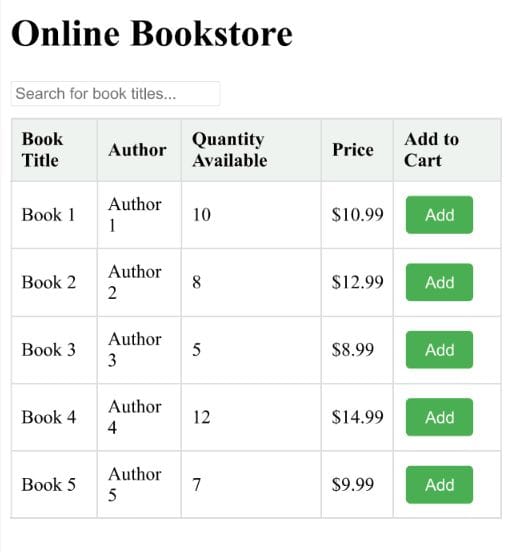
}

</script>

</body>

</html>

**OUTPUT**



**ONLINE BOOK STORE.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Online Bookstore</title>

<style>

table {

width: 100%;

border-collapse: collapse;

}

th, td {

border: 1px solid #dddddd;

text-align: left;

padding: 8px;

}

th {

background-color: #f2f2f2;

}

add-to-cart-button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 8px 16px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 14px;

margin: 4px 2px;

cursor: pointer;

border-radius: 4px;

}

search-container {

margin-bottom: 10px;

}

</style>

</head>

<body>

<h1>Online Bookstore</h1>

<div class="search-container">

<input type="text" id="isbnInput" placeholder="Search by ISBN...">

<input type="text" id="titleInput" placeholder="Search by book title...">

<input type="text" id="authorInput" placeholder="Search by author...">

<button onclick="searchBooks()">Search</button>

</div>

<table id="booksTable">

<thead>

<tr>

<th>ISBN</th>

<th>Book Title</th>

<th>Author</th>

<th>Quantity Available</th>

<th>Price</th>

<th>Add to Cart</th>

</tr>

</thead>

<tbody>

<tr>

<td>1234567890</td>

<td>Book 1</td>

<td>Author 1</td>

<td>10</td>

<td>$10.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>2345678901</td>

<td>Book 2</td>

<td>Author 2</td>

<td>8</td>

<td>$12.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>3456789012</td>

<td>Book 3</td>

<td>Author 3</td>

<td>5</td>

<td>$8.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>4567890123</td>

<td>Book 4</td>

<td>Author 4</td>

<td>12</td>

<td>$14.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

<tr>

<td>5678901234</td>

<td>Book 5</td>

<td>Author 5</td>

<td>7</td>

<td>$9.99</td>

<td><button class="add-to-cart-button">Add</button></td>

</tr>

</tbody>

</table>

**ONLINE BOOK STORE.JS**

<script>

function search Books () {

var isbnInput, titleInput, authorInput, isbnFilter, titleFilter, authorFilter, table, tr, td, i, isbnValue, titleValue, authorValue;

isbnInput = document. getElement ById("isbnInput");

titleInput = document. Get Element ById("titleInput");

authorInput = document.get Element ById("authorInput");

isbnFilter = isbn Input. value.to Upper Case ();

titleFilter = titleInput. value.to Upper Case ();

authorFilter = author Input.value.to Upper Case ();

table = document.get ElementById("booksTable");

tr = table.get Elements ByTagName("tr");

for (i = 0; i < tr. length; i++) {

isbnValue = tr[i]get ElementsByTagName("td") [0]. Text Content || tr[i].

get Elements ByTagName("td") [0]. Inner Text;

titleValue = tr[i].

get ElementsByTagName("td") [1].text Content || tr[i].

getElementsBy TagName("td") [1]. innerText;

authorValue = tr[i]

getElementsBy TagName("td") [2].text Content || tr[i].

getElementsByTagName("td") [2]. innerText;

if (isbnValue.toUpperCase().

indexOf(isbnFilter) > -1 &&

titleValue.toUpperCase(). Index Of(titleFilter) > -1 &&

authorValue.toUpperCase().

indexOf (authorFilter) > -1) {

tr[i]. style display = "";

} else {

tr[i]. style. Display = "none";

}

}

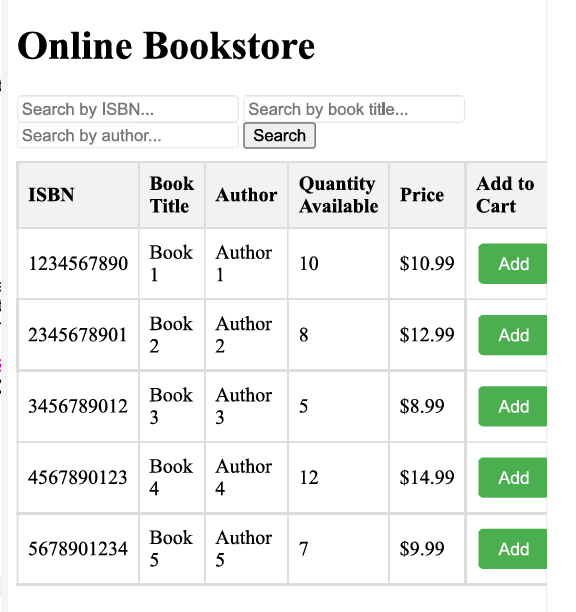
}

</script>

</body>

</html>

**OUTPUT**



SHOPPING CART FOR THE USER.HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Online Bookstore</title>

<style>

table {

width: 100%;

border-collapse: collapse;

}

th, td {

border: 1px solid #dddddd;

text-align: left;

padding: 8px;

}

th {

background-color: #f2f2f2;

}

add-to-cart-button {

background-color: #4CAF50; /\* Green \*/

border: none;

color: white;

padding: 8px 16px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 14px;

margin: 4px 2px;

cursor: pointer;

border-radius: 4px;

}

search-container {

margin-bottom: 10px;

}

</style>

</head>

<body>

<h1>Online Bookstore</h1>

<div class="search-container">

<input type="text" id="isbnInput" placeholder="Search by ISBN...">

<input type="text" id="titleInput" placeholder="Search by book title...">

<input type="text" id="authorInput" placeholder="Search by author...">

<button onclick="searchBooks()">Search</button>

</div>

<table id="booksTable">

<thead>

<tr>

<th>ISBN</th>

<th>Book Title</th>

<th>Author</th>

<th>Quantity Available</th>

<th>Price</th>

<th>Add to Cart</th>

</tr>

</thead>

<tbody>

<tr>

<td>1234567890</td>

<td>Book 1</td>

<td>Author 1</td>

<td>10</td>

<td>$10.99</td>

<td><button class="add-to-cart-button" onclick="addToCart ('1234567890', 'Book 1', 'Author 1', 10, 10.99)">Add</button></td>

</tr>

<tr>

<td>2345678901</td>

<td>Book 2</td>

<td>Author 2</td>

<td>8</td>

<td>$12.99</td>

<td><button class="add-to-cart-button" onclick="addToCart ('2345678901', 'Book 2', 'Author 2', 8, 12.99)">Add</button></td>

</tr>

<tr>

<td>3456789012</td>

<td>Book 3</td>

<td>Author 3</td>

<td>5</td>

<td>$8.99</td>

<td><button class="add-to-cart-button" onclick="addToCart ('3456789012', 'Book 3', 'Author 3', 5, 8.99)">Add</button></td>

</tr>

<tr>

<td>4567890123</td>

<td>Book 4</td>

<td>Author 4</td>

<td>12</td>

<td>$14.99</td>

<td><button class="add-to-cart-button" onclick="addToCart ('4567890123', 'Book 4', 'Author 4', 12, 14.99)">Add</button></td>

</tr>

<tr>

<td>5678901234</td>

<td>Book 5</td>

<td>Author 5</td>

<td>7</td>

<td>$9.99</td>

<td><button class="add-to-cart-button" onclick="addToCart ('5678901234', 'Book 5', 'Author 5', 7, 9.99)">Add</button></td>

</tr>

</tbody>

</table>

<h2>Shopping Cart</h2>

<ul id="shoppingCart">

<! - -Shopping cart items will be added here dynamically -->

</ul>

<button onclick="placeOrder ()">Place Order</button>

**SHOPPING CART FOR THE USER.JS**

<script>

var cartItems = [];

function addToCart (isbn, title, author, quantity, price) {

cartItems.push({isbn: isbn, title: title, author: author, quantity: quantity, price: price});

render Cart ();

}

function renderCart () {

var cartList = document.get Element ById("shoppingCart");

cartList.innerHTML = "";

var totalPrice = 0;

cartItems.forEach(function(item) {

var li = document. createElement("li");

li.textContent =item. title+ " by " + item. Author + " - Quantity: " + item. quantity+ " - Price: $" + (item. quantity\* item. Price). To Fixed (2);

cartList.appendChild(li);

totalPrice += item. quantity\* item. Price;

});

var totalLi = document. createElement("li");

totalLi.textContent = "Total Price: $" + totalPrice.toFixed(2);

cartList.appendChild(totalLi);

}

function place Order () {

// Here you can implement the logic to place the order, such as sending a request to a server.

// For demonstration purposes, this function simply clears the shopping cart.

cartItems = [];

render Cart ();

alert ("Your order has been placed!");

}

function searchBooks () {

// Implement your book search logic here

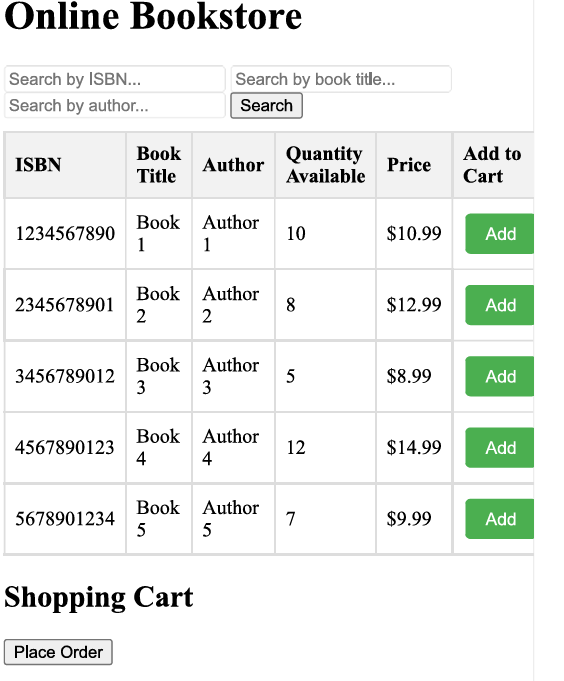
}

</script>

</body>

</html>

**OUTPUT**



**REGISTRATION.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Online Bookstore</title>

<style>

/\* CSS styles \*/

form-row {

margin-bottom: 10px;

}

</style>

</head>

<body>

<div id="loginContainer">

<h1>Registration</h1>

<div id="registrationForm">

<div class="form-row">

<label for="userId">User ID:</label>

<input type="text" id="userId" placeholder="User ID">

</div>

<div class="form-row">

<label for="password">Password:</label>

<input type="password" id="password" placeholder="Password">

</div>

<div class="form-row">

<label for="confirmPassword">Confirm Password:</label>

<input type="password" id="confirmPassword" placeholder="Confirm Password">

</div>

<div class="form-row">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" placeholder="First Name">

</div>

<div class="form-row">

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" placeholder="Last Name">

</div>

<div class="form-row">

<label for="address">Address:</label>

<input type="text" id="address" placeholder="Address">

</div>

<div class="form-row">

<label for="city">City:</label>

<input type="text" id="city" placeholder="City">

</div>

<div class="form-row">

<label for="zip">Zip:</label>

<input type="text" id="zip" placeholder="Zip">

</div>

<div class="form-row">

<label for="state">State:</label>

<select id="state">

<option value="AL">Alabama</option>

<option value="AK">Alaska</option>

<option value="AZ">Arizona</option>

<! - - Add more states as needed -->

</select>

</div>

<div class="form-row">

<label for="email">Email:</label>

<input type="email" id="email" placeholder="Email">

</div>

<div class="form-row">

<label for="phone">Phone:</label>

<input type="tel" id="phone" placeholder="Phone">

</div>

<div class="form-row">

<button onclick="register()">Submit</button>

<button type="reset">Reset</button>

</div>

</div>

</div>

<div id="bookstoreContainer" style="display: none;">

<! - - Bookstore content goes here -->

</div>

<script>

// Function to simulate user registration

function register () {

// Retrieve values from input fields and perform validation

var userId = documentGetElement ById("userId").value;

var password = document. Get Element ById("password").value;

var confirmPassword = document.get ElementById("confirmPassword").value;

var firstName = document.get ElementById("firstName").value;

var lastName = document.get ElementById("lastName").value;

var address = document.get ElementById("address").value;

var city = document.get ElementById("city").value;

var zip = document.get ElementById("zip").value;

var state = document.get ElementById("state").value;

var email = document.get ElementById("email").value;

var phone = document.get ElementById("phone").value;

// Check if passwords match

if (password! == confirmPassword) {

alert ("Passwords do not match.");

return;

}

// Additional validation can be added here

// Display registration success message

Alert ("Registration successful!");

// Clear form fields

document.get ElementById("registrationForm"). Reset ();

}

</script>

</body>

</html>

**OUTPUT**



**LOGIN.HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Online Bookstore</title>

<style>

/\* CSS styles \*/

form-row {

margin-bottom: 10px;

}

</style>

</head>

<body>

<div id="loginContainer">

<h1>Registration</h1>

<div id="registrationForm">

<div class="form-row">

<label for="userId">User ID:</label>

<input type="text" id="userId" placeholder="User ID">

</div>

<div class="form-row">

<label for="password">Password:</label>

<input type="password" id="password" placeholder="Password">

</div>

<div class="form-row">

<label for="confirmPassword">Confirm Password:</label>

<input type="password" id="confirmPassword" placeholder="Confirm Password">

</div>

<div class="form-row">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" placeholder="First Name">

</div>

<div class="form-row">

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" placeholder="Last Name">

</div>

<div class="form-row">

<label for="address">Address:</label>

<input type="text" id="address" placeholder="Address">

</div>

<div class="form-row">

<label for="city">City:</label>

<input type="text" id="city" placeholder="City">

</div>

<div class="form-row">

<label for="zip">Zip:</label>

<input type="text" id="zip" placeholder="Zip">

</div>

<div class="form-row">

<label for="state">State:</label>

<select id="state">

<option value="AL">Alabama</option>

<option value="AK">Alaska</option>

<option value="AZ">Arizona</option>

<! - - Add more states as needed -->

</select>

</div>

<div class="form-row">

<label for="email">Email:</label>

<input type="email" id="email" placeholder="Email">

</div>

<div class="form-row">

<label for="phone">Phone:</label>

<input type="tel" id="phone" placeholder="Phone">

</div>

<div class="form-row">

<button onclick="register()">Submit</button>

<button type="reset">Reset</button>

</div>

</div>

</div>

<div id="bookstoreContainer" style="display: none;">

<! - - Bookstore content goes here -->

</div>

**LOGIN.JS**

<script>

// Function to simulate user registration

function register () {

// Retrieve values from input fields and perform validation

var userId = document.get ElementById("userId").value;

var password = document.get ElementById("password").value;

var confirmPassword = document.get ElementById("confirmPassword").value;

var firstName = document.get ElementById("firstName").value;

var lastName = document.get ElementById("lastName").value;

var address = document.get ElementById("address").value;

var city = document.get ElementById("city").value;

var zip = document.get ElementById("zip").value;

var state = document.get ElementById("state").value;

var email = document.get ElementById("email").value;

var phone = document.get ElementById("phone").value;

// Check if passwords match

if (password! = = confirmPassword) {

alert ("Passwords do not match.");

return;

}

// Additional validation can be added here

// Display registration success message

Alert ("Registration successful!");

// Clear form fields

document.get ElementById("registrationForm") reset ();

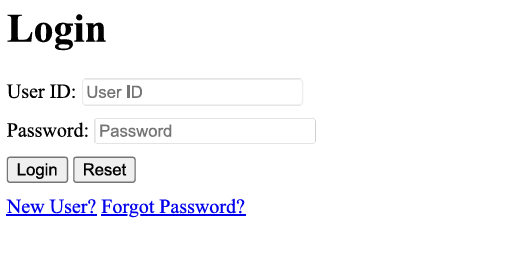
}

</script>

</body>

</html>

**OUTPUT**



**BACKEND DEVELOPMENT**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const nodemailer = require('nodemailer');

const app = express ();

const port = 3000;

// Connect to MongoDB

mongoose. Connect ('mongodb://localhost:27017/myapp'

{use New UrlParser: true, useUnifiedTopology: true});

// Define user schema and model

const userSchema = new mongoose. Schema ({

username: String,

password: String,

email: String

});

const User = mongoose. Model ('User', userSchema);

// Middleware to parse JSON requests

app.use (bodyParser.json());

// Secret key for JWT token

const secretKey = 'your-secret-key';

// Email configuration

const transporter = nodemailer. createTransport ({

service: 'gmail',

auth: {

user: 'your-email@gmail.com',

pass: 'your-email-password'

}

});

// User registration endpoint

app.post ('/register', async (req, res) => {

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

try {

// Check if user already exists

const existingUser = await User.findOne({username});

if (existingUser) {

return res. Status (400). Json ({message: 'User already exists'});

}

// Hash password

const hashedPassword = await bcrypt. Hash (password, 10);

// Create new user

const newUser = new User ({username, password: hashedPassword, email});

await newUser.save();

res. Json ({message: 'User registered successfully'});

} catch (error) {

console. Error ('Error registering user:', error);

res. status (500) json ({message: 'Internal server error'});

}

});

// User login endpoint

app.post ('/login', async (req, res) => {

const {username, password} = req. body;

try {

// Find user by username

const user = await User.findOne({username});

if (! user) {

return res. Status (401) json ({message: 'Invalid username or password'});

}

// Compare hashed password

const passwordMatch = await bcrypt. Compare (password, user. Password);

if (! password Match) {

return res. status (401). Json ({message: 'Invalid username or password'});

}

// Generate JWT token

const token = jwt. Sign ({userId: user.id}, secretKey, {expiresIn: '1h'});

res. Json ({message: 'Login successful', token});

} catch (error) {

console. Error ('Error logging in user:', error);

res. status (500). Json ({message: 'Internal server error'});

}

});

// Forgot password endpoint

app.post ('/forgot-password', async (req, res) => {

const {email} = req. body;

try {

// Check if email exists

const user = await User.findOne({email});

if (! user) {

return res. Status (404). Json ({message: 'User not found'});

}

// Generate password reset token

const token = jwtsign ({userId: user.id }, secretKey, { expiresIn: '1h' });

// Send password reset link to user's email

const mailOptions = {

from: 'your-email@gmail.com',

to: email,

subject: 'Password Reset',

text: Click the following link to reset your password: http://localhost:3000/reset-password/${token}

};

await transporter. Send Mail(mailOptions);

res. Json ({message: 'Password reset link sent to your email'});

} catch (error) {

console. Error ('Error sending password reset link:', error);

res. status (500). Json ({message: 'Internal server error'});

}

});

// Reset password endpoint

app. Post ('/reset-password/: token', async (req, res) => {

const {token} = req. params;

const {new Password} = req. body;

try {

// Verify token

const decoded = jwt. Verify (token, secretKey);

// Find user by decoded user ID

const user = await User.findById(decoded. userId);

if (! user) {

return res. status (404). Json ({message: 'User not found'});

}

// Hash new password

const hashedPassword = await bcrypt. Hash (newPassword, 10);

// Update user's password

user. Pass word = hashedPassword;

await user. Save ();

res. Json ({message: 'Password reset successful'});

} catch (error) {

console. Error ('Error resetting password:', error);

res. status (500) json ({message: 'Internal server error'});

}

});

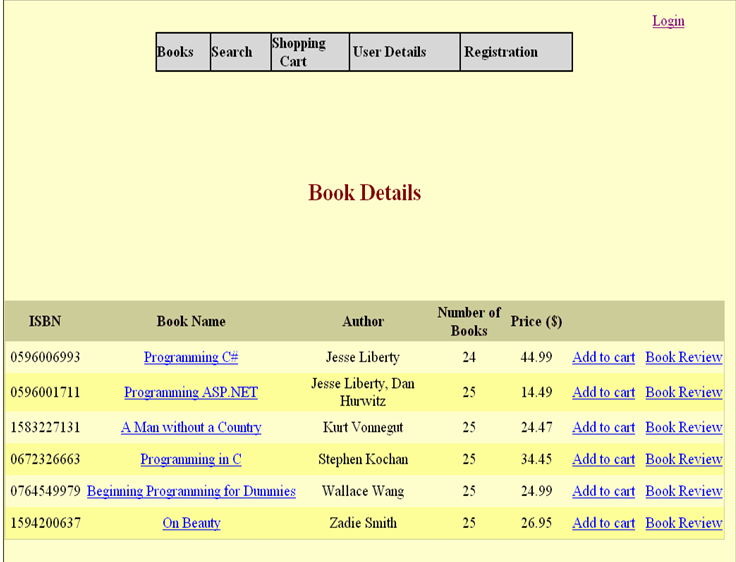
// Start server

app. listen (port, () => {

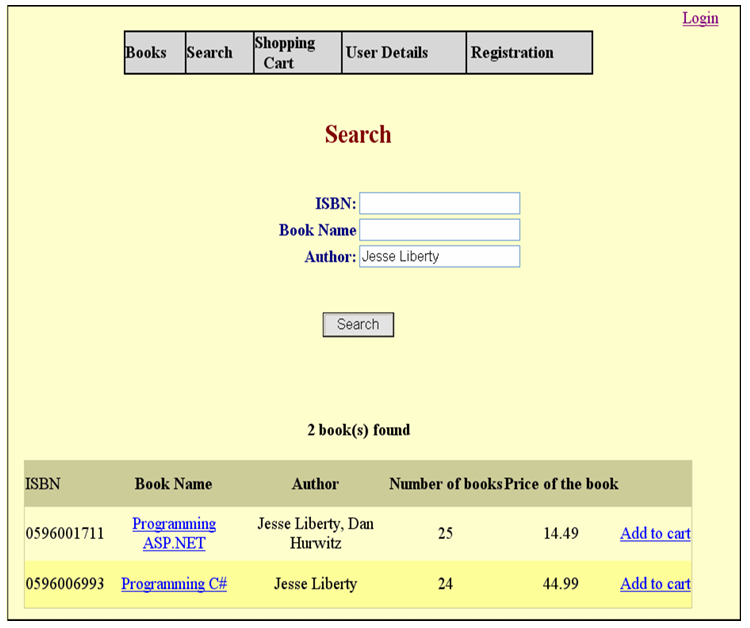
console.log (Server running at http://localhost:${port});

});

**RESULTS**



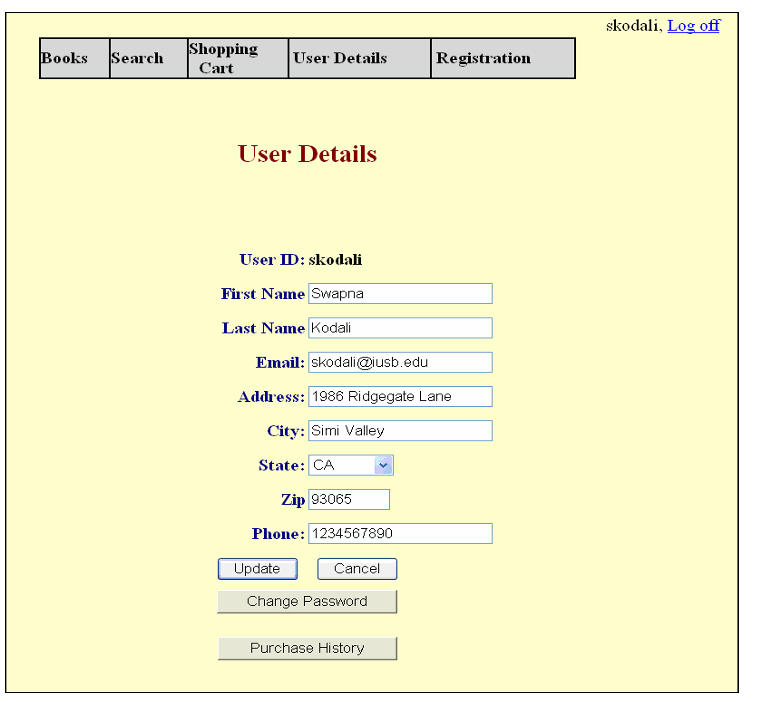
**FIGURE:1**



**FIGURE:2**

****

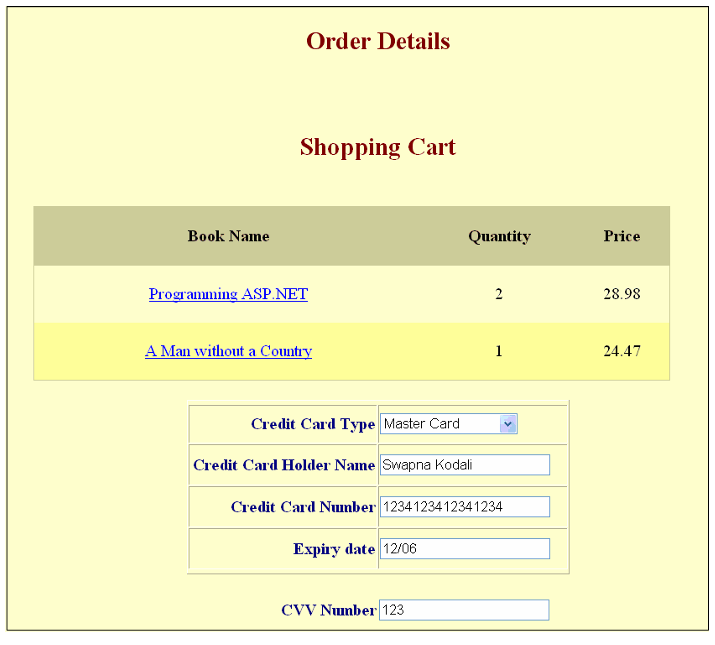
**FIGURE:3**



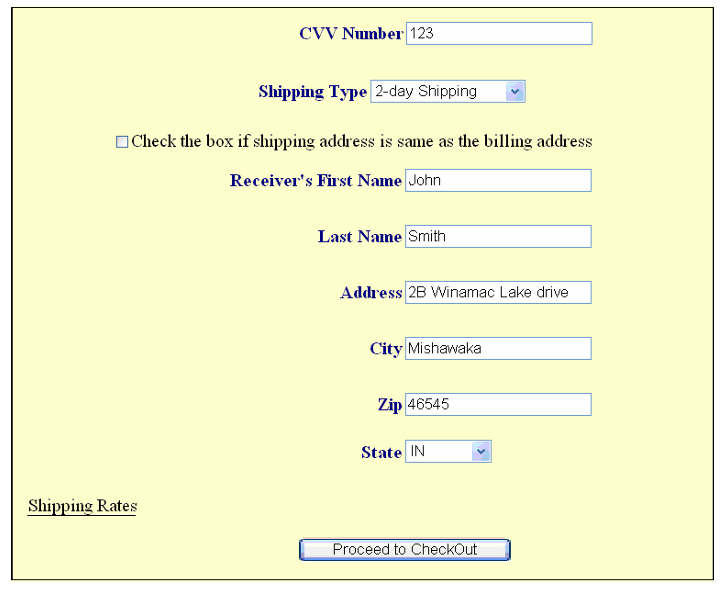
**FIGURE:4**



**FIGURE:5**

****

**FIGURE:6**

****

**FIGURE:7**

****

**FIGURE:8**

**CONCLUSION**

E-Commerce industry is that force which cannot be ignored by any element of the economy. Coronavirus pandemic proved that one of the major tools that can help consumers during crisis is e-commerce. In order to maintain social distancing and self quarantine the consumers have become more reliable on the e-commerce industry.

COVID-19 pandemic catalyzed a significant surge in e-commerce activity, with consumers increasingly turning to online shopping for its safety and convenience. This rapid adoption of e-commerce highlighted both opportunities and challenges within the industry. Businesses adapted by enhancing their online presence and implementing measures such as contactless delivery. However, supply chain disruptions and cybersecurity risks posed notable challenges. Despite these obstacles, the pandemic accelerated the digital transformation of retail, with long-term implications expected to shape the e-commerce landscape well beyond the pandemic's end.

"The 'Comprehensive e commerce platform on online book sales ' web application represents a significant advancement over traditional furniture shopping methods, eliminating manual work and the associated challenges. This platform offers a seamless and efficient way to access extensive information on the wide array of furniture available in the market. It not only enhances the quality of information but also streamlines the entire furniture shopping experience.

While we take pride in the progress we've made, we recognize that there's always room for improvement. Our commitment to excellence drives us to continually refine and enhance our platform, ensuring it provides even better services and information to our customers.

Primarily, 'Online Book Sales' excels in delivering relevant and comprehensive details, empowering customers to make informed choices. It serves as a digital solution for cataloging and accessing information about furniture, as well as the task management needs of both customers and employees within our organization.

We've embarked on this journey with the vision of reducing reliance on manual processes, simplifying complexities, and enriching the furniture shopping experience. In doing so, we aim to empower our customers and create an environment where the perfect furniture choices bring joy and harmony to their homes.