E-commerce Application on IBM Cloud Foundry

Karthik P(420721104015)

Problem Definition:

This abstract provides a glimpse into the development and deployment of an E-commerce Application on IBM Cloud Foundry, offering insights into its key features and the advantages it offers to businesses and customers alike. The E-commerce Application on IBM Cloud Foundry represents a cutting-edge solution designed to address the complexities and challenges faced by modern online retailers.

Design Thinking:

Design thinking is a user-ences of users to create a seamless and satisfying shopping experience. Here's how design thinking can be applied to design centered approach to problem-solving and innovation that can be applied to the development of an e-commerce application on IBM Cloud Foundry. It focuses on understanding the needs and prefering an e-commerce application on IBM Cloud Foundry

INNOVATION IDEAS:

1.Augmented Reality (AR) Shopping: Integrate AR technology into e-commerce platforms to allow customers to virtually try on products, see how furniture fits in their space, or visualize how a product would look in real life. This enhances the shopping experience.

How AR Shopping Works:

- **AR App or Feature:** Customers typically need a mobile app with AR capabilities or a web platform that supports AR. They can access this through their smartphones or tablets.
- **Product Visualization:** When browsing products, users can activate the AR feature to view items in 3D or overlay them onto their real-world environment using the device's camera.
- **Virtual Try-On:** For fashion and accessories, users can virtually try on clothing, glasses, or makeup. They can see how these items look on them in real-time through their device's screen.

- **Product Interaction:** Customers can manipulate, zoom in, or rotate products to examine them from different angles, helping to make more informed purchasing decisions.

Benefits of AR Shopping:

- **Enhanced Shopping Experience:** AR shopping bridges the gap between online and in-store shopping, providing a more immersive and interactive experience for customers.
- **Innovative Marketing:** AR shopping can be part of your marketing strategy. It can serve as a unique selling point and attract media attention.
- **Reduced Showrooming:** When customers can virtually "try on" or visualize products at home, they are less likely to visit physical stores to make purchasing decisions.

2 .Voice-Based Product Reviews:

Voice-based product reviews are an innovative way to enhance the shopping experience and provide valuable insights to both consumers and businesses. Here's how voice-based product reviews can work and their potential benefits:

- **Recording:** Customers can leave voice-based reviews by recording their feedback, thoughts, and experiences with a product using their smartphone or other recording devices.
- **Integration:** E-commerce platforms or mobile apps can integrate this feature into their product pages, allowing customers to upload or record their voice reviews directly from the product page.
- **Transcription and Analysis**: Voice recordings can be transcribed into text for easier indexing and search. Advanced Natural Language Processing (NLP) and sentiment analysis can be applied to understand the content and sentiment of the reviews.
- **Accessibility**: Customers who might prefer speaking over typing can use voice reviews, making it more inclusive for people with varying abilities.

.

3.Sustainable and Eco-friendly Shopping:

Offer eco-friendly product options and allow customers to easily identify and choose environmentally friendly products. This can be combined with carbon footprint tracking for orders.

Eco-friendly Product Selection:

- Prioritize products that are sustainable, ethically sourced, and environmentally friendly.
- Highlight eco-friendly features and certifications for each product (e.g., organic, Fair Trade, energy-efficient).

Sustainable Packaging:

- Use minimal, biodegradable, or recyclable packaging materials.
- Encourage customers to choose minimal packaging options during the checkout process.
- Consider innovative packaging solutions such as reusable or upcycled materials.

Green Supply Chain:

- Partner with suppliers and manufacturers committed to sustainability and ethical production practices.
- Promote transparency in the supply chain by sharing information about your suppliers and their sustainability efforts.
- Energy-Efficient Cloud Infrastructure:
- Host your e-commerce platform on energy-efficient servers and utilize renewable energy sources, where possible.
- Optimize data centers and server operations to reduce energy consumption.

3.Voice Commerce:

Enable voice-activated shopping through platforms like Amazon Alexa or Google Assistant. Customers can make purchases and track orders using voice commands. Voice commerce, also knowas v-commerce or voice-activated shopping, is an emerging trend that allows users to make purchases through voice-activated devices such as smart speakers, virtual assistants, and voice-activated mobile apps. Integrating voice commerce into your e-commerce website can enhance the shopping experience and a broader audience.

Voice Assistant Integration:

- Choose the voice assistant(s) you want to integrate with your e-commerce platform. Popular options include Amazon Alexa, Google Assistant, and Apple Siri.

- Develop or customize voice commerce applications or skills that work with these voice assistants.

User Account Linking:

- Implement a secure user account linking process, allowing customers to link their e-commerce account with their voice assistant account.

Voice Search and Product Discovery:

- Enable voice search capabilities that allow users to search for products using natural language.
- Implement voice-based product recommendations and suggestions based on customer preferences and browsing history.

Voice Shopping Cart:

- Create a voice-activated shopping cart where users can add, remove, or modify items using voice commands.
- Ensure users can check their cart and review their items by asking the voice assistant.

4.Enhanced Security:

Enhanced security is crucial for an e-commerce website to protect sensitive customer data, ensure trust, and maintain a good reputation. Here are important security measures and best practices to consider for your e-commerce website

Use HTTPS: Implement SSL (Secure Sockets Layer) or TLS (Transport Layer Security) to encrypt data transmitted between your website and the user's browser. This ensures secure data`transfer, including personal and payment information

Payment Card Industry Data Security Standard (PCI DSS) Compliance:

Comply with PCI DSS requirements if you handle payment card information. Use a PCI-compliant payment processor to securely handle credit card transactions.

Strong Password Policies: Enforce strong password policies for both users and employees. Encourage users to create complex passwords and consider implementing multi-factor authentication (MFA) for added security.

Regular Software Updates: Keep your e-commerce platform, plugins, and extensions up to date to patch security vulnerabilities. Regularly monitor for security updates and apply them promptly.

Web Application Firewall (WAF): Implement a WAF to protect **against** common web application attacks, such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
Data Encryption: Encrypt sensitive data, including customer information and payment details, when stored in databases. Use strong encryption algorithms and key management.

LAYOUT

Designing the layout of an e-commerce website is a critical step in creating an attractive and user-friendly online store. The layout should focus on guiding users through the shopping process, showcasing products, and making it easy for them to make purchases. Here's a basic structure for an e-commerce website layout

Header:

Logo:
Place your company logo at the top-left corner for brand recognition.
Navigation Menu:
Include clear and concise navigation links such as Home, Shop, Categories, About Us, Contact, and a search bar.
User Account:
Provide options for users to log in, create an account, or access their profile.
Shopping Cart:
Show the number of items in the user's cart and a link to the cart page.
Contact Information:
Display a phone number or email for customer support.

Hero Banner:

- Use a large, eye-catching image or slideshow to feature promotional products or offers.
- Include a compelling call-to-action (CTA) button.

Product Categories:

- Display a grid of product categories or collections.
- Each category should have an image and a title.
- Consider using featured products in each category.

Featured Products:

- Showcase a selection of popular or new products.
- Include high-quality images, product names, prices, and brief descriptions.
- Add "Shop Now" or "View Details" buttons.

Product Listings:

- Display a grid or list of products within a selected category.
- Include product images, titles, prices, and user ratings.
- Implement sorting and filtering options to help users find what they're looking for.

Product Detail Page:

- When a user clicks on a product, they should be directed to a detailed product page.
- Display high-resolution images, detailed descriptions, pricing, and customer reviews.
- Add "Add to Cart" and "Buy Now" buttons.

Shopping Cart:

- Show a summary of items in the cart.
- Include the product name, quantity, price, and a "Proceed to Checkout" button.

Checkout Process:

- Split the checkout into multiple steps, such as shipping, payment, and review.
- Include form fields for shipping and billing information.
- Provide payment options and a summary of the order

Footer:

- Include links to important pages like Terms and Conditions, Privacy Policy, and Returns.
- Display trust badges and security certifications.

- Add contact information, social media links, and a newsletter signup.

Additional Elements:

- Consider including a live chat or customer support section.
- Implement a customer reviews and ratings section.
- Add a blog section for content related to your products or industry.
- Include a search bar at the top for quick product searches.

Remember to keep the layout clean, responsive, and mobile-friendly. Test the user experience to ensure that navigation is intuitive, and the site loads quickly. An attractive and user-friendly e-commerce website layout can enhance the shopping experience and improve your conversion rates.

Implementing user authentication, shopping cart, and checkout functionality in a web application is a common requirement for e-commerce websites. Here's a high-level overview of how you can go about implementing these features

User Authentication:

Choose an authentication method, such as email/password, social login, or single sign-on (SSO).
Use a framework or library that handles user authentication and security (e.g., Firebase, Auth0, or a custom solution with a technology like JWT).
Create user registration and login forms.
Implement user registration, login, and logout endpoints.
Secure user passwords by hashing and salting them.
Set up session management or token-based authentication to keep users authenticated.

Shopping Cart:

Design and create a database schema for the shopping cart, which typically includes tables for users, products, and cart items.
Implement an API to add, update, and remove products from the

shopping cart.
Use client-side technologies (HTML, JavaScript) to display the shopping cart and update it as users add or remove items.
Implement functionality to calculate the total price and quantity of items in the cart.
Provide a mechanism to store the cart data for the user, such as using cookies or local storage.

Checkout Functionality:

Design and create a database schema for orders, order items, and payment information.
Implement a checkout process that guides users through the steps of confirming their cart, providing shipping and billing information, and reviewing the order.
Validate user-provided data during the checkout process (e.g., credit card information, shipping address).
Calculate the total order cost and update the user with the final amount.
Integrate with payment gateways for processing payments securely
Create order confirmation pages and send confirmation emails to users.

User Registration

User registration is a process in which a user provides their personal information and creates an account or profile on a website, application, or system. It is a common feature in many online services and platforms. The primary purpose of user registration is to allow users to access and interact with the features and content of the platform while providing a way for the platform to identify and differentiate users.

User registration is a fundamental feature in many web applications, and you can implement it using Node.js. To create a basic user registration system, you'll need to use several libraries and technologies, including Node.js, Express.js for your web server, and a database.

HTML:
```
<!DOCTYPE html>
<html>
<head>
    <title>User Registration</title>
</head>
```

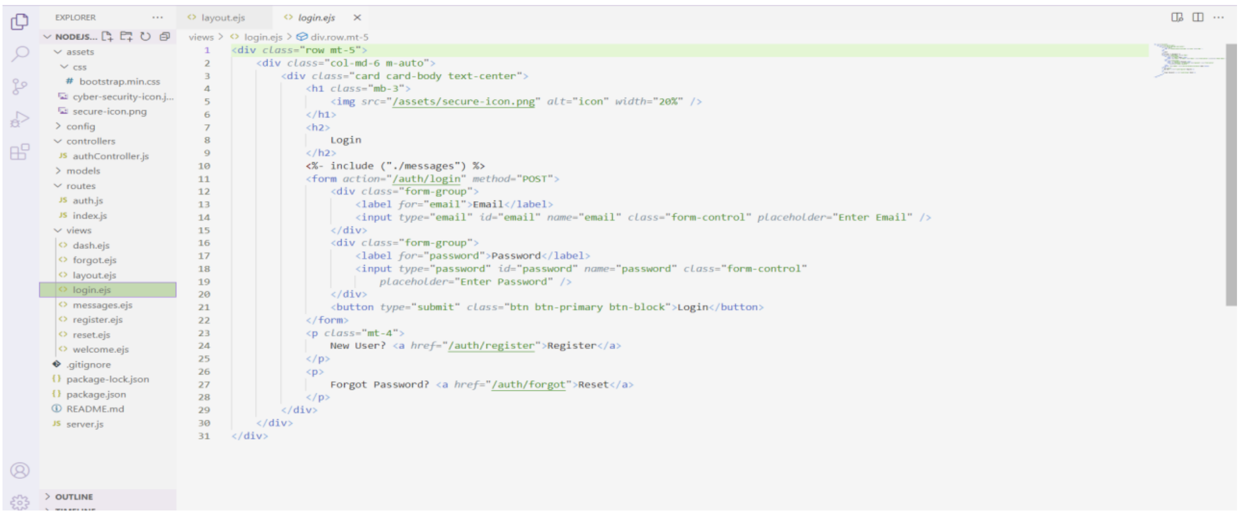
```
<body>
   <h2>User Registration</h2>
   <form action="/register" method="post">
      <label for="username">Username:</label>
      <input type="text" id="username" name="username"
required><br><br>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email"
required><br><br>

      <label for="password">Password:</label>
      <input type="password" id="password" name="password"
required><br><br>

      <input type="submit" value="Register">
   </form>
</body>
</html>
```

The <form> element is used to create the registration form. The action attribute specifies where the form data should be sent when the user submits the form. In this case, it's set to "/register," which is a common endpoint for handling user registration on the server.
The form contains three input fields for the username, email, and password, each with a corresponding label. The required attribute ensures that the fields must be filled out before the form can be submitted.
The type attribute of the "password" field is set to "password," which hides the entered text for security purposes.
The "Register" button is an <input> element with a type of "submit." When clicked, it submits the form.

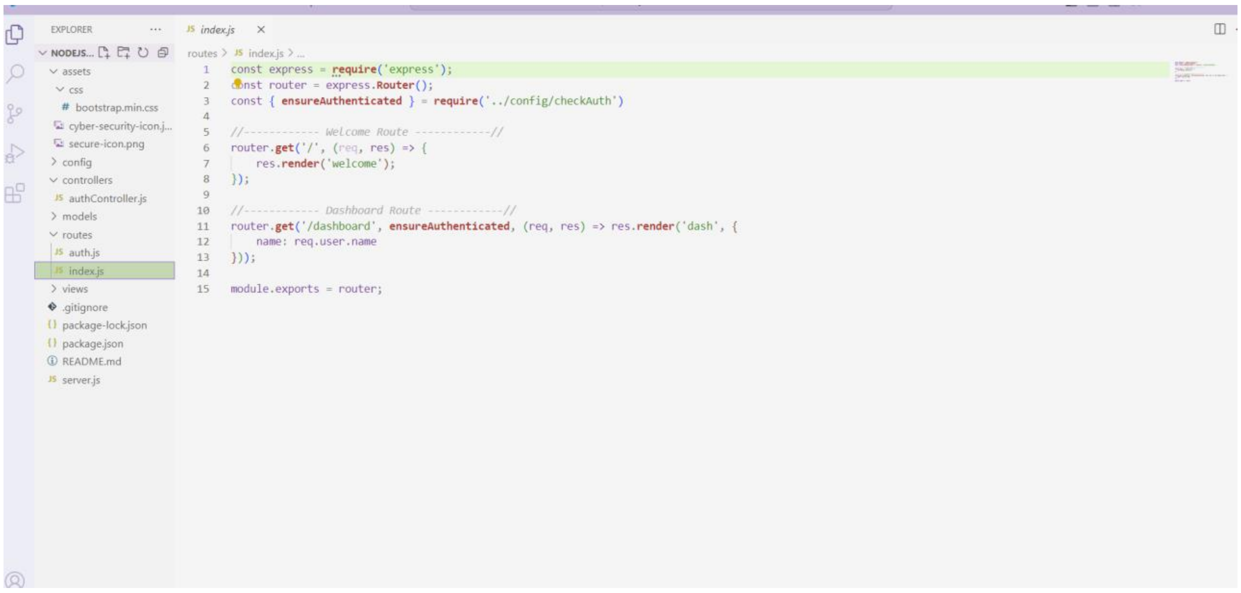
User registration is the process by which individuals sign up and create accounts on websites, applications, or online platforms. It is a common and essential feature in the digital world and serves several key purposes:
Identification: User registration helps a platform identify and distinguish users from one another. Each registered user is typically assigned a unique identifier, such as a username or user ID, which

allows the platform to recognize and track them.

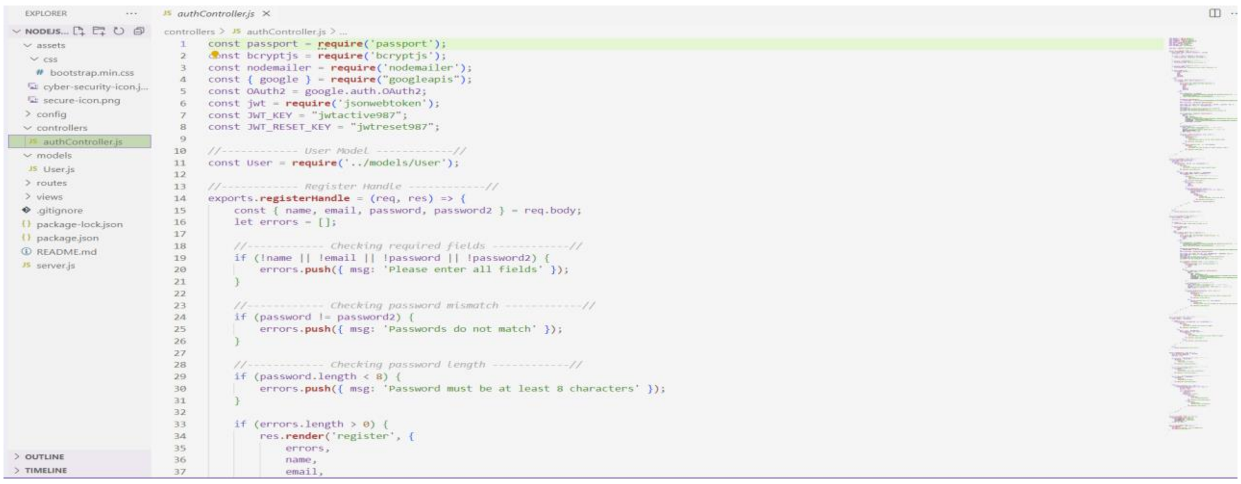
Authentication: Registered users must typically go through an authentication process to confirm their identity, often by providing a valid email address and creating a password. This authentication step is crucial for ensuring that the user is who they claim to be.
Access Control: After registration and authentication, users gain access to certain features, content, or functionality on the platform. This access control enables platforms to offer a personalized experience, as well as to restrict access to specific areas or actions for non-registered or unauthenticated users.
User Profiles: During the registration process, users often have the opportunity to create a user profile. This profile can include personal information, profile pictures, and other details that allow users to personalize their accounts.
Interaction: Registered users can engage with the platform in various ways, including posting content, commenting, messaging other users, and participating in discussions or transactions.
Communication: User registration provides a means for the platform to communicate with users, such as sending notifications, updates, or marketing messages. Users can also communicate with one another through the platform.
Security: User registration helps platforms maintain security by tracking user activities, enforcing rules and policies, and, when necessary, suspending or banning users who violate terms of service.
Data Collection: Platforms collect data from registered users, which can be used for analytics, improving user experience, personalizing content, and targeting marketing efforts.



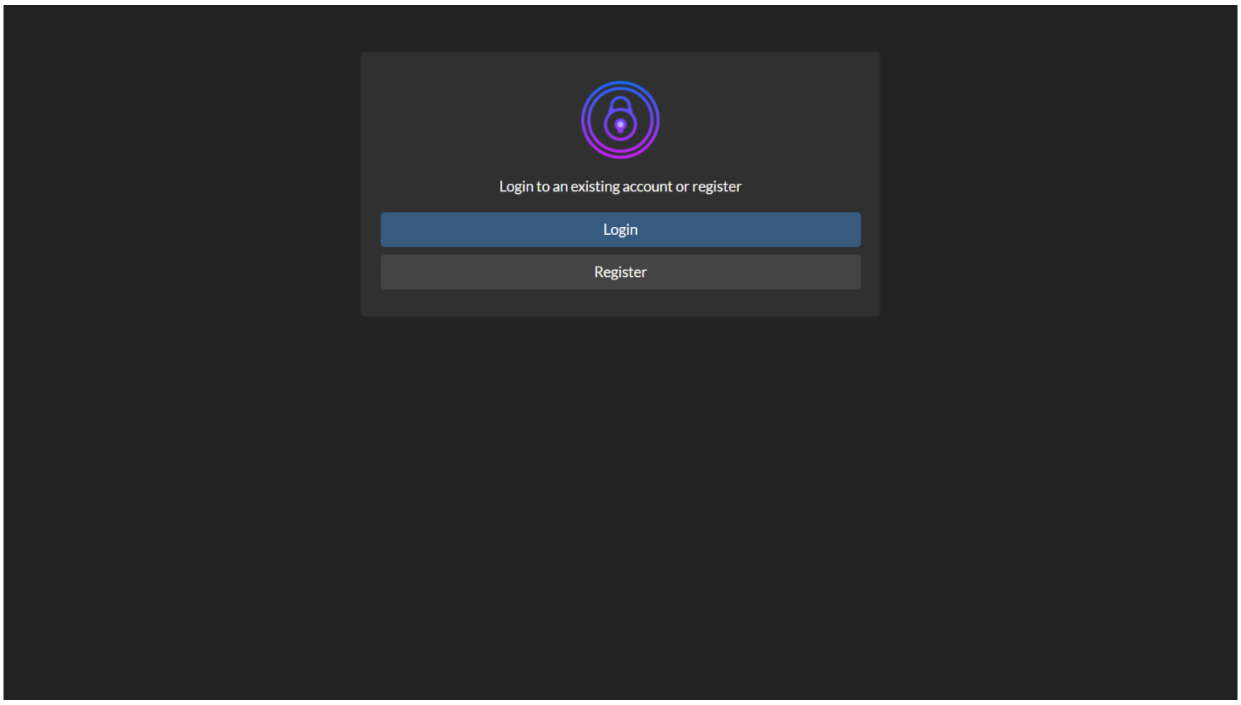
```
<div class="row mt-5">
    <div class="col-md-6 m-auto">
        <div class="card card-body text-center">
            <h1 class="mb-3">
                <img src="/assets/secure-icon.png" alt="icon" width="20%" />
            </h1>
            <h2>
                Login
            </h2>
            <%- include ("./messages") %>
            <form action="/auth/login" method="POST">
                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" id="email" name="email" class="form-control" placeholder="Enter Email" />
                </div>
                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" id="password" name="password" class="form-control"
                        placeholder="Enter Password" />
                </div>
                <button type="submit" class="btn btn-primary btn-block">Login</button>
            </form>
            <p class="mt-4">
                New User? <a href="/auth/register">Register</a>
            </p>
            <p>
                Forgot Password? <a href="/auth/forgot">Reset</a>
            </p>
        </div>
    </div>
</div>
```



```
const express = require('express');
const router = express.Router();
const { ensureAuthenticated } = require('../config/checkAuth')

//------------ Welcome Route ------------//
router.get('/', (req, res) => {
    res.render('welcome');
});

//------------ Dashboard Route ------------//
router.get('/dashboard', ensureAuthenticated, (req, res) => res.render('dash', {
    name: req.user.name
}));

module.exports = router;
```



```
const passport = require('passport');
const bcryptjs = require('bcryptjs');
const nodemailer = require('nodemailer');
const { google } = require("googleapis");
const OAuth2 = google.auth.OAuth2;
const jwt = require('jsonwebtoken');
const JWT_KEY = "jwtactive987";
const JWT_RESET_KEY = "jwtreset987";

//------------ User Model ------------//
const User = require('../models/User');

//------------ Register Handle ------------//
exports.registerHandle = (req, res) => {
    const { name, email, password, password2 } = req.body;
    let errors = [];

    //------------ Checking required fields ------------//
    if (!name || !email || !password || !password2) {
        errors.push({ msg: 'Please enter all fields' });
    }

    //------------ Checking password mismatch ------------//
    if (password != password2) {
        errors.push({ msg: 'Passwords do not match' });
    }

    //------------ Checking password length ------------//
    if (password.length < 8) {
        errors.push({ msg: 'Password must be at least 8 characters' });
    }

    if (errors.length > 0) {
        res.render('register', {
            errors,
            name,
            email,
```



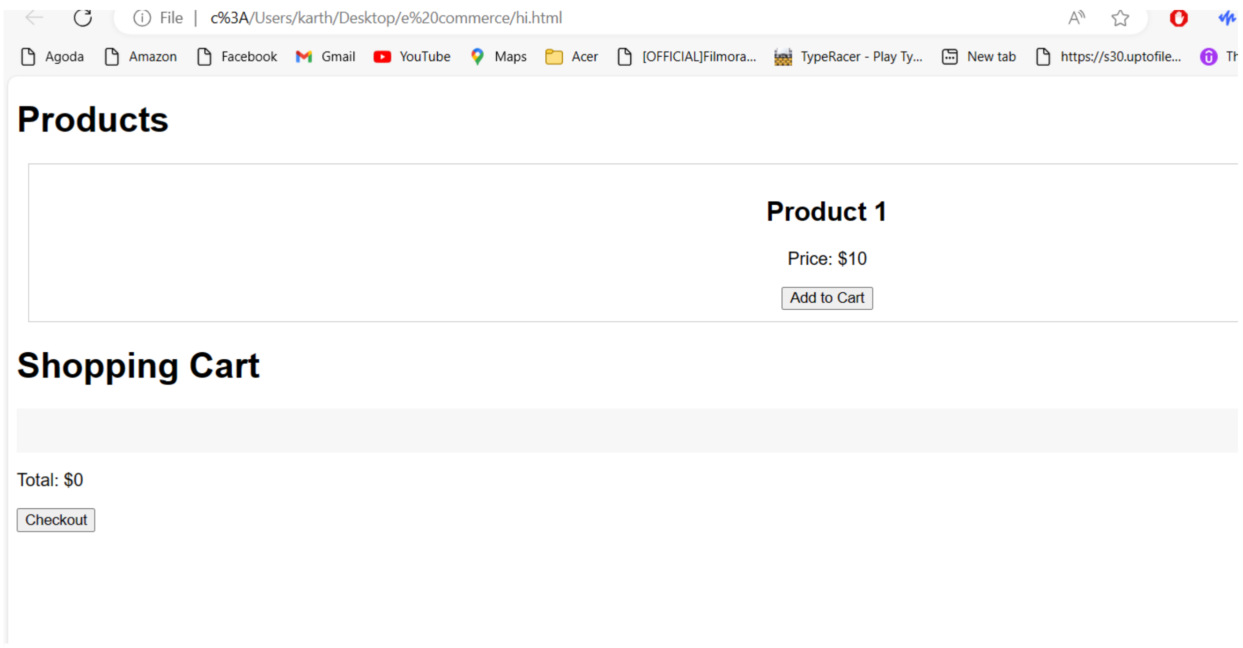
Implementing shopping cart functionality with total calculation and a smooth checkout process typically involves both front-end and back-end development. Below, I'll provide an example using JavaScript for the front-end and a simplified back-end. Please note that in a real-world scenario, you'd need a more robust back-end, database, and payment gateway integration for security and functionality.

HTML:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Shopping Cart</title>
    <link rel="stylesheet" href="./text.css">
</head>
<body>
    <h1>Products</h1>
    <div class="products">
        <div class="product">
            <h2>Product 1</h2>
            <p>Price: $10</p>
            <button onclick="addToCart('Product 1', 10)">Add to Cart</button>
        </div>
        <!-- Add more products here -->
    </div>
    <h1>Shopping Cart</h1>
    <ul id="cart">

        <!-- Cart items will be displayed here -->
    </ul>
    <p>Total: $<span id="total">0</span></p>
    <button onclick="checkout()">Checkout</button>

    <div id="checkout-form" style="display: none;">
        <h1>Checkout</h1>
        <!-- Add checkout form fields here -->
        <button onclick="completePurchase()">Complete Purchase</button>
    </div>

    <div id="confirmation" style="display: none;">
        <h1>Thank you for your purchase!</h1>
    </div>
    <script src="ji.js"></script>
</body>
</html>
```

JAVASCRIPT:

```javascript
let cart = [];
let total = 0;

function addToCart(productName, price) {
    cart.push({ name: productName, price: price });
    updateCart();
}

function updateCart() {
    const cartList = document.getElementById("cart");
    cartList.innerHTML = "";

    total = 0;
    cart.forEach(item => {
        const li = document.createElement("li");
        li.textContent = `${item.name} - $${item.price}`;
        cartList.appendChild(li);
        total += item.price;
    });

    document.getElementById("total").textContent = total;
}

function checkout() {
    document.getElementById("checkout-form").style.display = "block";
}

function completePurchase() {
    // Add logic to handle the purchase, e.g., send data to a server.
    // Show a confirmation message and clear the cart.
    cart = [];
    updateCart();
    document.getElementById("checkout-form").style.display = "none";
    document.getElementById("confirmation").style.display = "block";
}
```



Products

Product 1

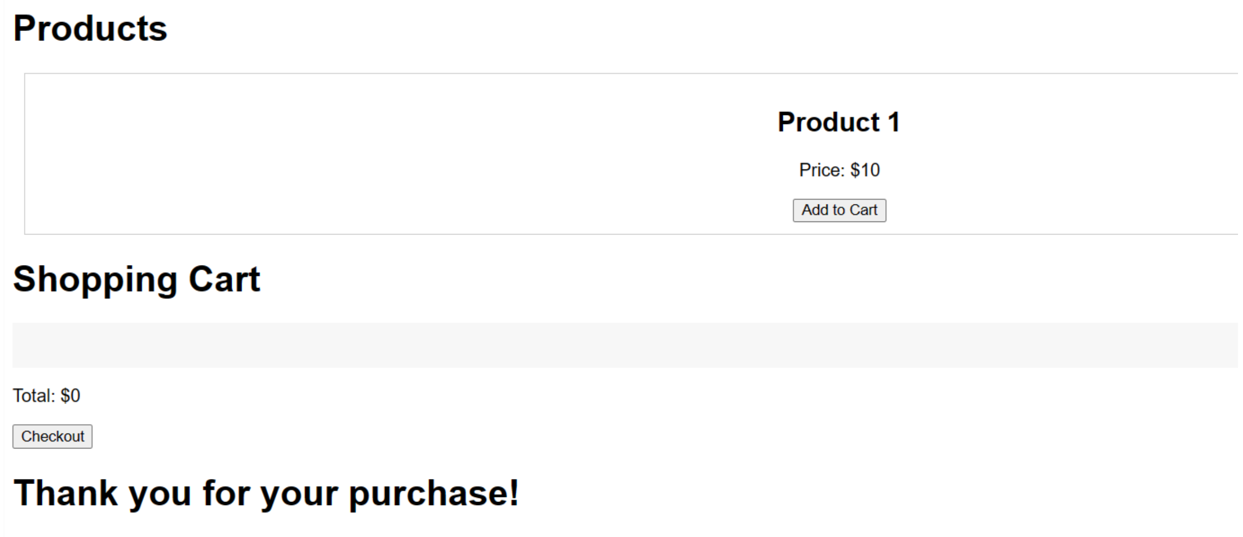
Price: $10

Add to Cart

Shopping Cart

Total: $0

Checkout



Products

Product 1

Price: $10

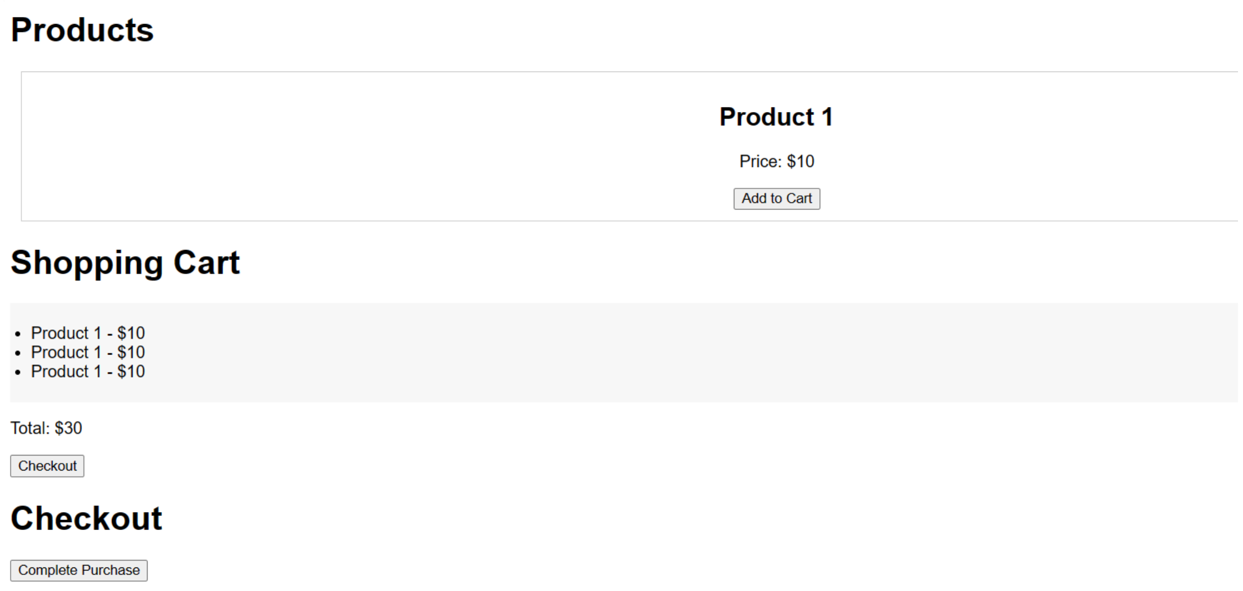
Add to Cart

Shopping Cart

Total: $0

Checkout

Thank you for your purchase!



Products

Product 1

Price: $10

Add to Cart

Shopping Cart

- Product 1 - $10
- Product 1 - $10
- Product 1 - $10

Total: $30

Checkout

Checkout

Complete Purchase