## **Shanmuganathan engineering college , code:9124**

domain:cloud computing

**team members :**

* Thrisha k 912421104052
* Srividhya.S 912421104044
* Soundariya.r 912421104042
* Keerthika.s 912421104302

## **Project:**

## **E-commerce App**



**PHASE 4:**

**GIVEN STATEMENT:**

Problem Statement: Build an artisanal e-commerce platform using IBM Cloud Foundry. Connect skilled artisans with a global audience. Showcase handmade products, from exquisite jewelry to artistic home decor. Implement secure shopping carts, smooth payment gateways, and an intuitive checkout process. Nurture creativity and support small businesses through an artisan's dream marketplace!



***Creating a E- commerce applications on cloud foundry***

Creating an e-commerce application on Cloud Foundry involves several steps:

* **SET UP CLOUD FOUNDRY:**

Ensure you have access to a Cloud Foundry environment. You can use a platform like IBM Cloud, Pivotal Web Services, or SAP Cloud Platform, all of which support Cloud Foundry

* **SELECT A PROGRAMMING LANGUAGE:**

Choose a programming language and framework for your e-commerce application. Popular choices include Java (using Spring Boot), Node.js, or Ruby on Rails.

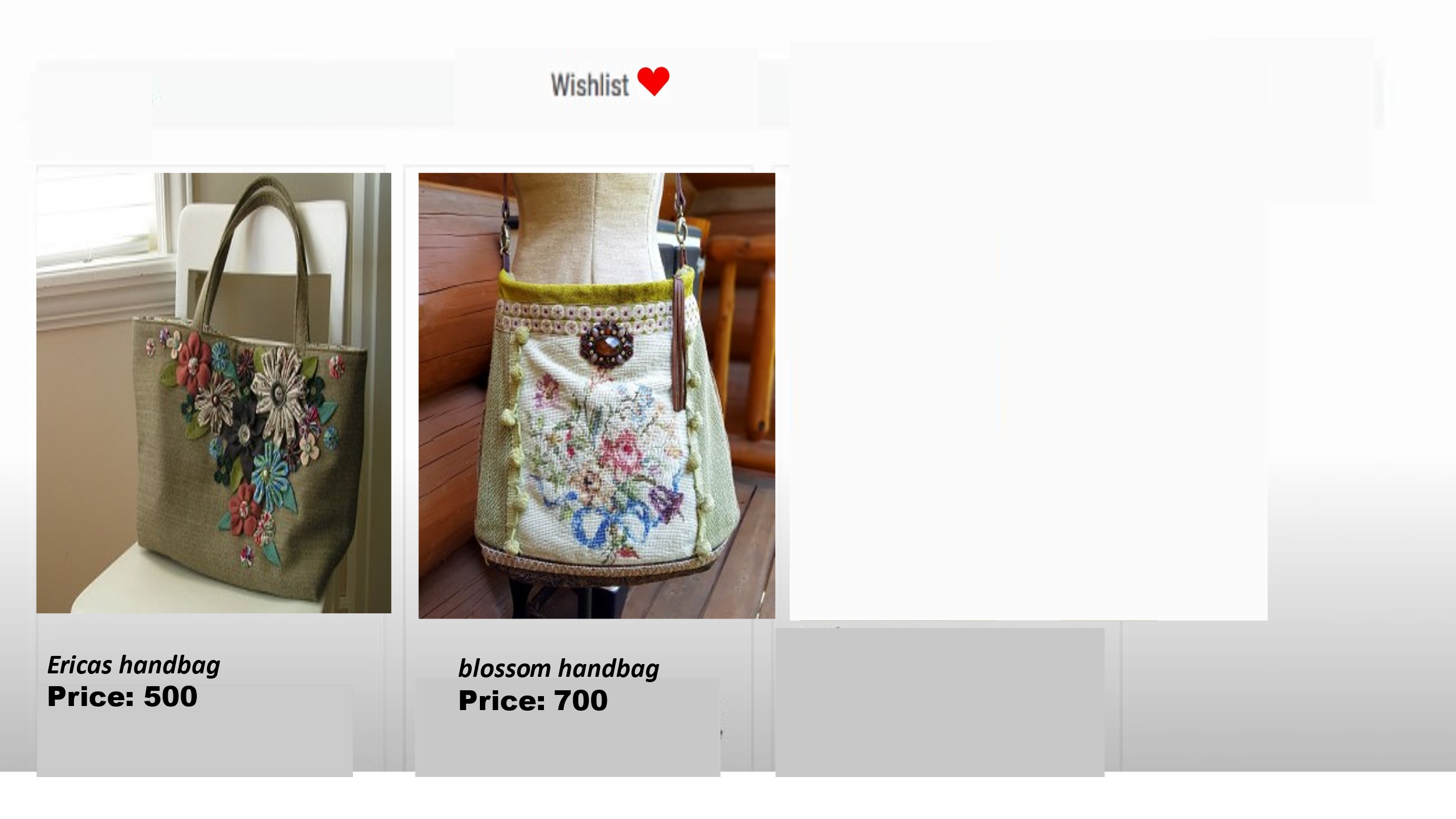
* **DATABASE**:   
                Decide on a database for your application. Cloud Foundry

supports various databases like PostgreSQL, MySQL, and MongoDB. You can choose the one that best fits your needs.

* **PUSH TO CLOUD FOUNDRY**:  
             Use the Cloud Foundry command-line interface (cf CLI) to push your application to the Cloud Foundry platform. This will make your application accessible on the cloud.
* **SERVICE INTEGRATION:**  
                 Integrate  any necessary services like payment gateways, caching, or CDN services. You can use Cloud Foundry's marketplace to add and bind these services to your app.
* **SCALING**:  
               Configure auto-scaling and load balancing to ensure your e-commerce application can handle varying levels of traffic.
* **MONITORING AND LOGGING** :  
               Set up monitoring and logging to keep an eye on the health and performance of your application. Cloud Foundry often provides tools and integrations for this purpose.

Wishlist products:

To implement a wishlist functionality in an e-commerce website using HTML, you'll typically need to use a combination of HTML, JavaScript, and potentially a back-end language for storing and retrieving wishlist dataTo implement a wishlist functionality in an e-commerce website using HTML, you'll typically need to use a combination of HTML, JavaScript, and potentially a back-end language for storing and retrieving wishlist data



8

Code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Product Page</title>

<style>

.wishlist-button {

cursor: pointer;

color: blue;

text-decoration: underline;

margin-left: 10px;

}

</style>

</head>

<body>

<h1>Product Name</h1>

<p>Description of the product.</p>

<button class="wishlist-button" onclick="addToWishlist()">Add to Wishlist</button>

<script>

function isLocalStorageSupported() {

try {

return 'localStorage' in window && window['localStorage'] !== null;

} catch (e) {

return false;

}

}

function addToWishlist() {

var product = {

id: 1,

name: "ericas handbag",

description: "the product is specialised."

};

if (isLocalStorageSupported()) {

// Retrieve the existing wishlist or create a new one

var wishlist = JSON.parse(localStorage.getItem('wishlist')) || [];

// Check if the product is not already in the wishlist

if (!wishlist.find(item => item.id === product.id)) {

// Add the product to the wishlist

wishlist.push(product);

// Save the updated wishlist to localStorage

localStorage.setItem('wishlist', JSON.stringify(wishlist));

// Inform the user

alert('Product added to wishlist!');

} else {

// Inform the user that the product is already in the wishlist

alert('Product is already in the wishlist!');

}

} else {

alert('Wishlist functionality is not available in your browser.');

}

}

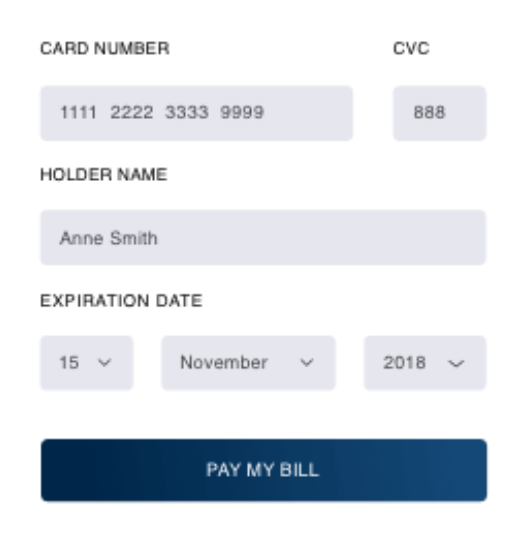
</script>

</body>

</html>

**Payment method:**

To working on an e-commerce website and need to handle payments, it's highly recommended to use a secure and established payment gateway to handle the payment process. Popular payment gateways such as Stripe, PayPal, or others provide secure mechanisms for processing payments without exposing sensitive information to your website.



**Code :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Your E-commerce Website</title>

<script src="https://js.stripe.com/v3/"></script>

</head>

<body>

<h1>Checkout</h1>

<!-- Your product details and form go here -->

<form id="payment-form">

<div id="card-element">

<!-- A Stripe Element will be inserted here. -->

</div>

<!-- Used to display form errors. -->

<div id="card-errors" role="alert"></div>

<button type="submit">Pay Now</button>

</form>

<script>

var stripe = Stripe('YOUR\_PUBLIC\_KEY');

var elements = stripe.elements();

// Create an instance of the card Element.

var card = elements.create('card');

// Add an instance of the card Element into the `card-element` div.

card.mount('#card-element');

// Handle real-time validation errors from the card Element.

card.addEventListener('change', function(event) {

var displayError = document.getElementById('card-errors');

if (event.error) {

displayError.textContent = event.error.message;

} else {

displayError.textContent = '';

}

});

// Handle form submission.

var form = document.getElementById('payment-form');

form.addEventListener('submit', function(event) {

event.preventDefault();

stripe.createToken(card).then(function(result) {

if (result.error) {

// Inform the user if there was an error.

var errorElement = document.getElementById('card-errors');

errorElement.textContent = result.error.message;

} else {

// Send the token to your server.

stripeTokenHandler(result.token);

}

});

});

function stripeTokenHandler(token) {

}

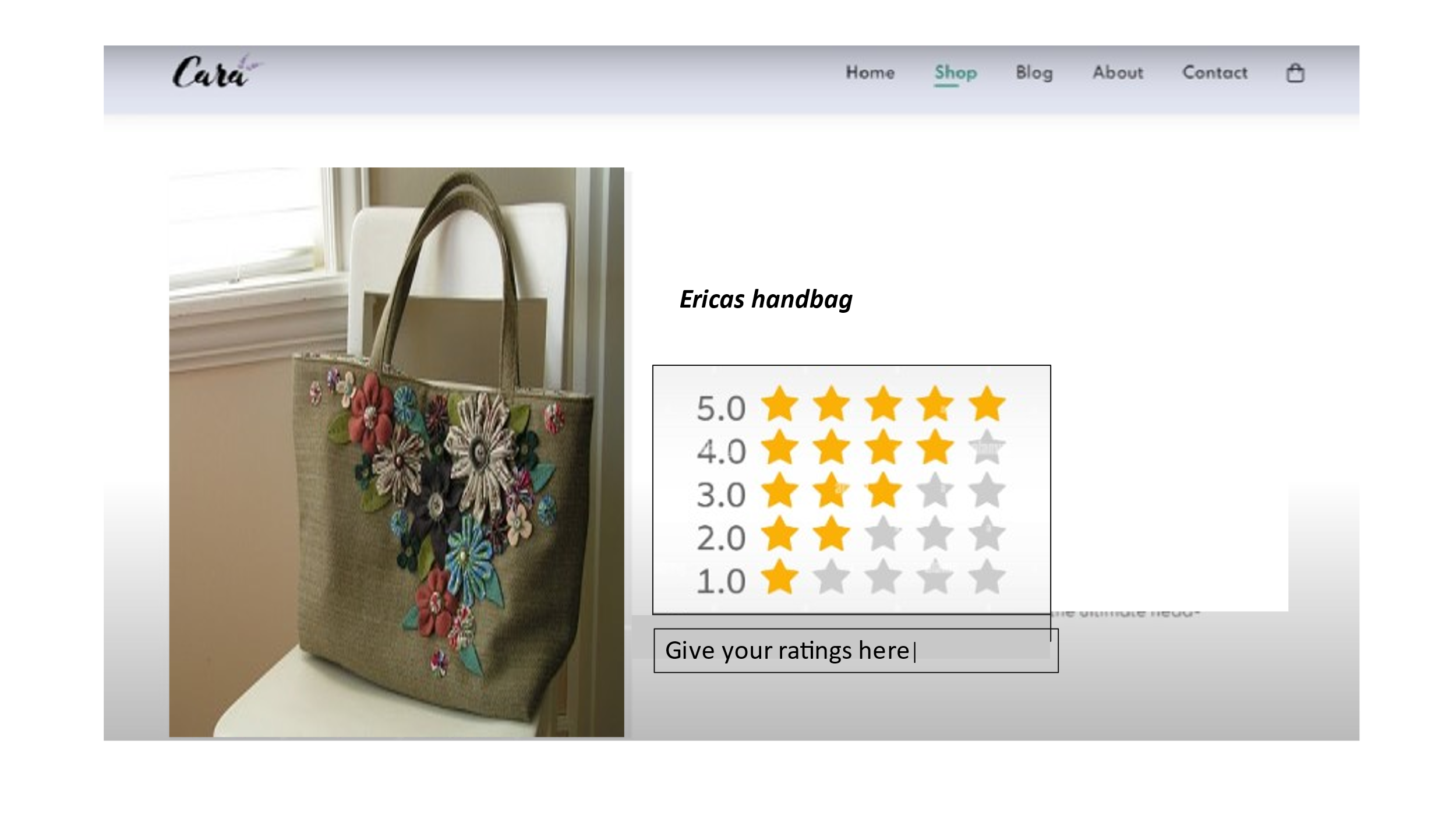
</script>

</body>

</html>

**Adding ratings to products:**

HTML and JavaScript code for adding product reviews on an e-commerce website. In this example, users can provide a review with a rating, and the reviews are displayed on the page.



**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Product Rating</title>

<style>

/\* Add some basic styling for clarity \*/

.rating-container {

margin-top: 20px;

font-size: 24px;

}

.star {

cursor: pointer;

color: gray;

}

.star:hover,

.star.checked {

color: gold;

}

</style>

</head>

<body>

<h1>Product Name</h1>

<p>Description of the product.</p>

<h2>Rate This Product</h2>

<div class="rating-container">

<span class="star" onclick="setRating(1)">★</span>

<span class="star" onclick="setRating(2)">★</span>

<span class="star" onclick="setRating(3)">★</span>

<span class="star" onclick="setRating(4)">★</span>

<span class="star" onclick="setRating(5)">★</span>

</div>

<p>Your Rating: <span id="selectedRating">0</span> stars</p>

<script>

var selectedRating = 0;

function setRating(rating) {

selectedRating = rating;

updateRatingDisplay();

}

function updateRatingDisplay() {

var ratingDisplay = document.getElementById('selectedRating');

ratingDisplay.textContent = selectedRating;

highlightStars();

}

function highlightStars() {

var stars = document.querySelectorAll('.star');

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

if (i < selectedRating) {

stars[i].classList.add('checked');

} else {

stars[i].classList.remove('checked');

}

}

}

</script>

</body>

</html>

**Conclusion:**

thus the part2 of development of ecommerce on cloud foundry is done