E-COMMERCE APPLICATION USING IBM CLOUD FOUNDRY

PHASE-4

Aadhavan.K

Ahamed Arsath .A

Balasubramanian.M

Lysander kagoo.B

E-commerce Application on IBM cloud foundry:

Introduction:

* E-commerce applications hosted on IBM Cloud Foundry provide businesses with a dynamic and scalable solution, enabling them to establish a strong online presence. This abstract explores the fundamental elements of e-commerce applications deployed on IBM Cloud Foundry, emphasizing their crucial role in the current digital commerce landscape.
* E-commerce applications on IBM Cloud Foundry leverage the power of cloud computing to provide businesses with a flexible and reliable platform for showcasing services. These applications are engineered to accommodate various business models from online stores to subscription services. Key components include a product catalog, shopping cart, secure payment processing, user account management, and robust security measures to protect sensitive data.
* The infrastructure of the IBM Cloud Foundry guarantees that e-commerce applications enjoy advantages such as high scalability, and effective resource utilization. Additionally, it provides an array of services for managing databases, conducting analytics, facilitating continuous integration and deployment (CI/CD), which simplifies both development and maintenance procedures.

**FRONT END:**

In a cloud application, the front-end serves as the user interface and is responsible for interacting with users, providing a means to access and interact with the application's features and data. When it comes to the front-end in a cloud application, several technologies and frameworks are commonly used to deliver a responsive and user-friendly experience. Some of these include:

1.Web-Based Interfaces**:** Many cloud applications utilize web-based front-ends that users can access through web browsers. Technologies like HTML, CSS, and JavaScript are the foundational for creating web interfaces. Frameworks like React, Angular, and Vue.js are often employed to build dynamic and responsive web applications.

2. Mobile Applications: In cases where a cloud application needs to be accessible on mobile devices, front-end development extends to mobile app development. Technologies like React Native, Flutter, or native app development for iOS and Android can be used to create mobile front-ends that connect to the cloud services.

3.Progressive Web Apps (PWAs): PWAs are web applications that can provide a native app-like experience on the web. They offer features like offline access, push notifications, and responsive design. These are typically built using web technologies and frameworks.

4**.**Desktop Applications: Such as: Some cloud applications require desktop front-end interfaces. Frameworks like Electron allow developers to create cross-platform desktop applications using web technologies.

5.UI/UX Design**:** Front-end development in cloud applications also involves creating a user-friendly and visually appealing user interface. This includes designing the layout, navigation, and overall look and feel of the application.

6.API Integration:Front-end developers often need to integrate with cloud-based APIs to access data and services from the cloud. This requires understanding API documentation and using technologies like REST, GraphQL, or WebSocket for real-time communication.

7. Authentication and Security**:** Implementing, authentication and security features is crucial in cloud applications. This involves technologies like OAuth, OpenID Connect, and secure communication protocols (HTTPS) to protect user data and application functionality.

8. Cloud Service Integration: Front-end developers need to work closely with back-end developers to ensure seamless integration with cloud services, databases, and storage solutions. Technologies like AWS Amplify, Firebase, or Azure Functions can help streamline this integration.

9. Responsive Design: Creating front-ends that work well on various devices and screen sizes is essential. Responsive web design techniques and CSS frameworks like Bootstrap or Foundation help achieve this.

10. Testing and Quality Assurance: Front-end testing frameworks like Jest, Cypress, and Selenium are employed to ensure the reliability and functionality of the front-end interface.

Creating the front-end of an e-commerce application in the cloud necessitates a synergy of design aesthetics and technical craftsmanship. It's a collaborative effort where front-end developers and designers, working hand-in-hand with their counterparts in the backend teams, strive to forge an immersive and fluid user journey. This harmonious interplay is pivotal in sculpting a user experience that seamlessly melds the art of design with the precision of development while harnessing the vast potential of cloud-based services.

### SOURCE CODE:

<!DOCTYPE html>

<html>

<head>

<title>Shopping Cart</title>

</head>

<body>

<h1>Shopping Cart</h1>

<ul>

{% for item in cart\_contents %}

<li>

{{ item['product']['name'] }} - ${{ item['product']['price'] }} x {{ item['quantity'] }}

</li>

{% endfor %}

</ul>

<a href="{{ url\_for('index') }}">Continue Shopping</a>

</body>

</html>

**BACKEND BUILDING:**

## USER AUTHENTICATION:

Creating a user login system for an e-commerce application using cloud computing involves leveraging cloud-based services for user authentication and security. Here are the fundamental steps and concepts involved:

* + **User Authentication:**

Implement login functionality by integrating the authentication service's features. Users enter their credentials (username/email and password) to authenticate themselves.

* + **Password Security:**

Ensure the secure storage of passwords. Store them using secure and modern hashing algorithms to prevent unauthorized access even if the data is somehow compromised.

* + **Secure Data Transmission:**

Utilize HTTPS (SSL/TLS) to the encrypt data transmission between the user's device and your application servers, ensuring the security of sensitive information during login processes.

* + **Access Control and Authorization:**

Define user roles and permissions within your e-commerce application. For instance, distinguish between regular users and administrators, providing appropriate access levels and privileges.

* + **Logging and Monitoring:**

Implement logging and monitoring systems to track login activities, identify potential security threats, and gather insights into user behavior.

**CODING:**

from flask import Flask, render\_template, request, redirect, url\_for

app = Flask(\_name\_)

users = {'user1': 'password1', 'user2': 'password2'} @app.route('/')

def login():

return render\_template('login.html') @app.route('/login', methods=['POST']) def login\_post():

username = request.form['username'] password = request.form['password']

if username in users and users[username] == password: return redirect(url\_for('profile'))

else:

# Failed login

return "Login failed. Please check your credentials." @app.route('/profile')

def profile():

return "Welcome to your profile!"

if \_name\_ == '\_main\_': app.run(debug=True)

## SHOPPING CART FUNCTIONALITY:

Shopping carts are an essential component of e-commerce applications, and they offer several advantages for both customers and online retailers. Here are some of the key advantages of using shopping carts in e-commerce applications:

**Convenience**: Shopping carts make it easy for customers to browse through a website, select products they want to purchase, and add them to their cart. This process mimics the physical shopping experience, making it familiar and convenient for users.

**Organization**: Shopping carts allow customers to keep track of the items they intend to purchase in an organized manner. They can easily see what they’ve added to the cart and adjust quantities or remove items before proceeding to checkout.

**Save for Later**: Customers can use the shopping cart to save items they are interested in but not ready to purchase immediately. This feature is particularly useful for comparison shopping and reducing the need to search for products again.

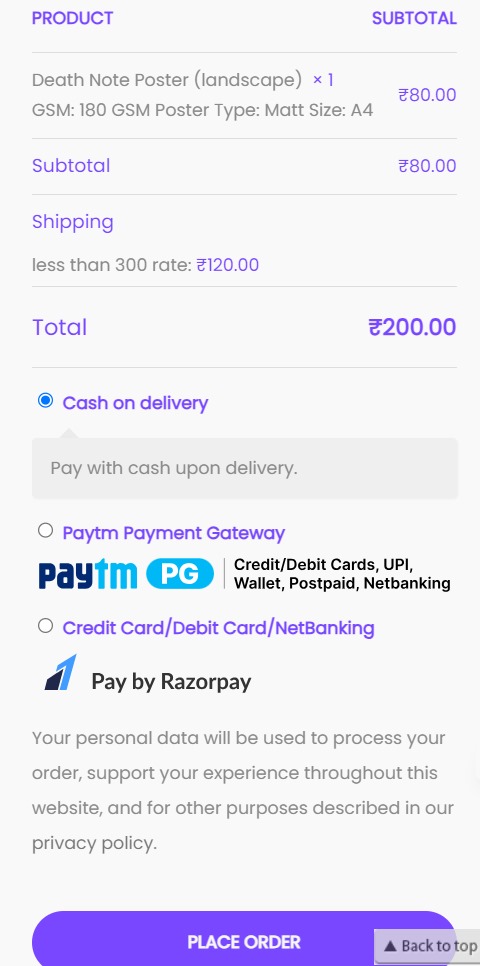
**Cross-Selling and Upselling:** E-commerce platforms can use shopping carts to suggest related or complementary products to customers, increasing the chances of upselling or cross-selling.

**Secure Transactions:** Shopping carts typically integrate with secure payment gateways, ensuring that financial transactions are handled safely. This builds trust with customers, which is essential in e-commerce.

**Inventory Management:** Shopping carts help manage inventory levels. When a product is added to the cart, it can be reserved for a limited time, preventing overselling and improving the customer experience.

**User Data Capture:** Shopping carts collect valuable data about customer preferences, which can be used for marketing and customer analysis. Retailers can gain insights into c tailor their offerings accordingly.

## CHECK-OUT FUNCTIONALITY:

Implementing a checkout functionality for an e-commerce application in a cloud computing environment involves several key steps.

* + **Database and Data Management:**

Utilize a reliable and scalable database service (such as AWS DynamoDB, Azure Cosmos DB, or Google Cloud Firestore) to manage orders, customer details, and transaction records.

* + **Integration with Payment Gateways:**

Integrate with payment gateway services (like Stripe, PayPal, or your preferred service) to securely process payments. Use cloud-based services or APIs provided by the payment gateways for this purpose.

* + **APIs for Checkout Operations:**

Develop APIs to manage the checkout process. Allow users to review their cart, enter shipping and billing information, choose shipping methods, and make payment transactions.

* + **Address Validation and Verification:**

Incorporate address validation services to ensure accurate delivery and reduce errors in shipping details.

* + **Shipping Integration:**

Integrate with shipping service APIs or third-party logistics providers to calculate shipping costs, provide delivery options, and generate shipping labels.

* + **Confirmation and Order Processing:**

Upon successful payment, confirm the order, generate an order ID, and trigger fulfillment processes. Update inventory and send order confirmation emails to customers.

Integrating a checkout functionality in an e-commerce application within a cloud environment involves careful consideration of security, reliability, and user experience.

### CODING:

from flask import Flask, render\_template, request, redirect, url\_for, session

app = Flask( name)

app.secret\_key = 'your\_secret\_key' # Change this to a secure random key

# Dummy product data products = {

'product1': {'name': 'Product 1', 'price': 10.00},

'product2': {'name': 'Product 2', 'price': 20.00},

'product3': {'name': 'Product 3', 'price': 30.00},

}

@app.route('/') def index():

return render\_template('index.html', products=products)

@app.route('/add\_to\_cart/<product\_id>') def add\_to\_cart(product\_id):

if 'cart' not in session: session['cart'] = {}

if product\_id in session['cart']: session['cart'][product\_id] += 1

else:

session['cart'][product\_id] = 1 return redirect(url\_for('index'))

@app.route('/cart') def view\_cart():

cart = session.get('cart', {}) cart\_contents = []

for product\_id, quantity in cart.items(): product = products.get(product\_id) if product:

cart\_contents.append({'product': product, 'quantity': quantity}) return render\_template('cart.html', cart\_contents=cart\_contents)

if \_name\_ == '\_main\_': app.run(debug=True)

## CONCLUSION:

To succeed in e-commerce project development, a comprehensive approach is necessary, encompassing technical, design, and business elements. Creating a thriving e-commerce platform involves a deep understanding of user requirements, market dynamics, and the seamless integration of technology.

This Details the implementation of a microservices architecture, efficient database management, and the integration of various cloud services for functions such as authentication, payment processing, and content delivery.

Continuous improvement, driven by user feedback and market trends, is essential to remain competitive and responsive to changing customer demands.

Adaptability and innovation are fundamental in this dynamic field. Embracing emerging technologies, outpacing competitors, and embracing change are all essential for sustainable growth in the ever-evolving e-commerce landscape.