Ecommerce Application On IBM Cloud Foundry

**Sathishkumar.K( 420721104043)**

## **Implementing user registration and authentication**

**User logins:**

Secure Authentication:

Implement strong authentication methods, such as multi-factor authentication (MFA) for enhanced security.

Remember Me Functionality:

Offer a "Remember Me" option to keep users logged in across sessions, but make it clear that it's less secure.

Session Management:

Manage user sessions securely, ensuring that sessions expire after a period of inactivity

Implement secure session tokens and refresh them regularly.

Password Reset:

Provide a "Forgot Password" feature that allows users to reset their passwords via email verification.

Account Lockout:

Implement account lockout mechanisms after a certain number of unsuccessful login attempts to prevent brute force attacks.

User Profile:

Users to manage their profile information, including password changes and email updates.

Brute Force Protection:

Implement protection mechanisms against brute force and credential stuffing attacks, like rate limiting login attempts.

Session Timeout:

Automatically log users out after a period of inactivity to protect against unauthorized access.

Security Logs:

It can Keep logs of login attempts and security-related events for auditing and security monitoring.

Third-Party Authentication:

It Allow users to log in using third-party identity providers (e.g., Google, Facebook) for a seamless experience.

Role-Based Access Control:

It Implement role-based access control (RBAC) to manage user permissions within the application.

Regular Security Audits:

Conduct regular security audits and penetration testing to identify and address vulnerabilities.

By implementing these features e-commerce application's user registration and login processes are secure, user-friendly, and privacy-conscious .

Additionally, it's essential to stay updated on evolving security threats and technologies to adapt your security measures accordingly.

**Shopping cart :**

Display Cart Icon

Start by displaying a shopping cart icon in the website header or navigation bar, indicating the number of items in the cart.

Add to Cart Functionality:

We can Implement an "Add to Cart" button on product detail pages.It Allow users to specify product quantity and variations (e.g., size, color).

View Cart:

It Provide an option for users to view the contents of their cart.Show product names, quantities, prices, and a subtotal for each item. Allow users to remove items or change quantities in the cart.

Cart Summary:

Display the total price of the items in the cart.

Show any applied discounts or promotional codes.

Continue Shopping :

Offer a "Continue Shopping" button to return to the product catalog. Save Cart for Later: Provide an option for users to save items in their cart for later purchase.

**Checkout functionally**

Checkout Process:

Ensure that the checkout page is secure with HTTPS.

Display a padlock icon to reassure users.

Guest Checkout:

Allow users to complete a purchase without creating an account (guest checkout).

Encourage them to register after the purchase.

Shipping Information:

Collect shipping information, including name, address, and contact details.

Offer options for different shipping methods with associated costs.

Billing Information:

Collect billing information, if different from shipping details.

Allow users to save billing information for future purchases.

Payment Options:

Provide various payment options, such as credit/debit cards, digital wallets, and PayPal.

Display logos of accepted payment methods for user trust.

Order Review:

Summarize the order, including products, quantities, prices, and total cost. Allow users to review and confirm their order before finalizing the purchase.

Promotional Codes and Discounts:

Provide a field for users to enter promotional codes.

Automatically apply discounts and update the order total.

Terms and Conditions:

Include a checkbox for users to agree to terms and conditions before proceeding.

Confirmation and Receipt:

Display an order confirmation page with a summary of the purchase. Send a confirmation email to the user with order details and a receipt.

Order Tracking:

Offer order tracking and status updates for registered users.

Abandoned Cart Recovery:

Implement abandoned cart recovery mechanisms, such as email reminders to encourage users to complete their purchase.

Inventory Management:

Ensure that items are removed from inventory once an order is placed to prevent overselling.

Payment Gateway Integration:

Integrate with a secure payment gateway to process payments and handle refunds if necessary.

Notifications:

Send order confirmation and shipping notifications to users via email or SMS.

Security and Compliance:

Ensure compliance with payment card industry data security standards (PCI DSS) for secure handling of payment data.

Mobile Optimization:

Optimize the checkout process for mobile users for a responsive and seamless experience.

Regularly test and optimize the checkout process to reduce cart abandonment and improve conversion rates. User feedback and analytics data can provide insights into areas that need improvement.

**User authentication and registration using Node.js(with Express.js) and python(with flask) for backend server connection.**

Node.js with Express.js (JavaScript):

Setup and Dependencies:

Initializing a new Node.js project

Database Setup:

Choosing a database (e.g., MongoDB, PostgreSQL) and set up a connection.

User Model:

Creating a user model that defines the schema for user data, including fields like username, email, and hashed password.

User Registration:

Creating a route for user registration, which accepts user data (username, email, password).

Hash the password using a library like bcrypt and save the user data to the database.

User Authentication:

Implementing a route for user authentication, which accepts user credentials (email/username and password).

Verify the provided credentials, generate a JWT token using a library like jsonwebtoken, and return it to the client.

Middleware for Authentication:

Creating middleware to protect routes that require authentication. Verify the JWT token and provide access if it's valid.

Password Reset:

Implementing a route for users to reset their passwords. This typically includes generating a reset token, sending it to the user's email, and allowing them to set a new password.

User Profile Management:

Creating routes for users to update their profiles, change passwords, and log out.

Error Handling and Validation:

Implementing proper error handling and input validation to ensure data integrity and security.

Flask (Python):

Setup and Dependencies:

Initialize a new Python project and install Flask. Organize your project structure.

Database Setup:

Choosing a database (e.g., SQLite, PostgreSQL) and establishing a connection.

User Model:

Creating a user model that defines the schema for user data, including fields like username, email, and hashed password.

User Registration:

Creating a route for user registration, accepting user data (username, email, password).

Hash the password using a library like bcrypt and save the user data to the database.

User Authentication:

Implementing a route for user authentication, accepting user credentials (email/username and password).

Middleware for Authentication:

Creating middleware to protect routes that require authentication

Password Reset:

Implementing a route for users to reset their passwords. This typically includes generating a reset token, sending it to the user's email, and allowing them to set a new password.

User Profile Management:

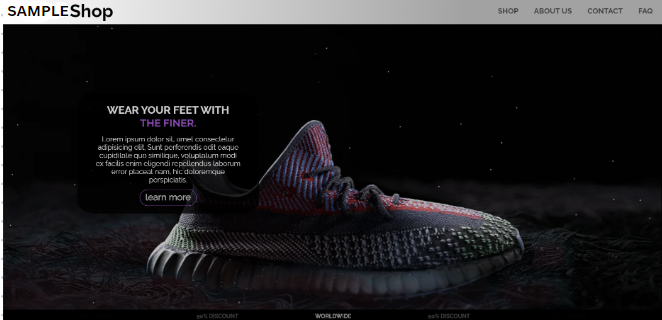
Create routes for users to update their profiles, change passwords, and log out.

Error Handling and Validation:

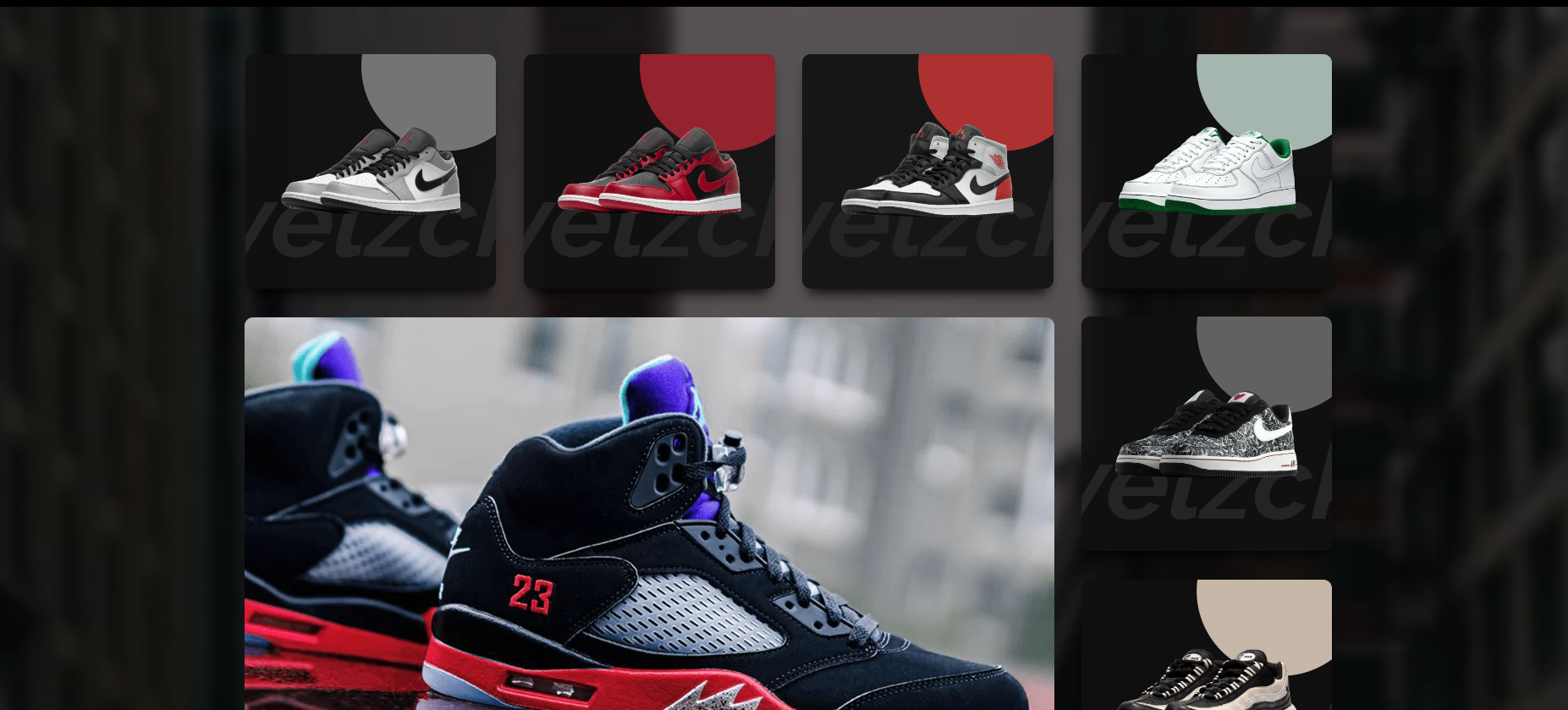
Implement proper error handling and input validation to ensure data integrity and security.

# OUTPUT :

HOME PAGE :



Shopping Cart:



ABOUT PAGE :

