STAR LION COLLEGE OF ENGINEERING AND TECHNOLOGY

Name	**V.Sriharan**
Department	**Computer Science And Engineering**
Project title	**E-commerce Application**

E-commerce Application on IBM Cloud Foundry

Project Title: E-commerce App

Phase 5: Project submisson

INTRODUCTION:

E-commerce, short for electronic commerce, has revolutionized the way businesses operate and consumers shop. In a fast-paced digital world, E-commerce applications have emerged as a pivotal tool for businesses to reach a broader audience and facilitate seamless transactions over the internet. These applications encompass online platforms where buyers and sellers engage in buying and selling products or services, making transactions efficient, convenient, and accessible 24/7.

E-commerce, short for electronic commerce, refers to the buying and selling of goods and services over the internet. It involves a wide range of online business activities, including online retail, digital payments, electronic auctions, and more. Ecommerce has transformed the way people conduct business and shop, offering several advantages, such as convenience, accessibility, a broader market reach, and cost savings. E-commerce has become a fundamental part of the global economy and has continuously evolved with advancements in technology and changing consumer behavior.

An E-commerce application serves as a virtual marketplace where customers can browse through a wide array of products or services, compare prices, read reviews, and make secure purchases without the need to visit physical stores. For businesses, these applications provide an avenue to showcase their offerings, manage inventory, process payments, and build lasting customer relationships.

In this rapidly evolving landscape, the success of an E-commerce application hinges on user experience, security, reliability, and adaptability to changing market trends. Providing a seamless, intuitive interface, secure payment gateways, personalized recommendations, and efficient logistics are paramount to ensuring customer satisfaction and retention.

E-Commerece:

E-commerce, short for electronic commerce, refers to the buying and selling of goods and services over the internet. It involves a wide range of online business activities, including online retail, digital payments, electronic auctions, and more. Ecommerce has transformed the way people conduct business and shop, offering several advantages, such as convenience, accessibility, a broader market reach, and cost savings. It encompasses various types of transactions, from business-toconsumer (B2C), where businesses sell products or services directly to individual consumers, to business-to-business (B2B), where companies trade goods and services with other businesses. E-commerce has become a fundamental part of the global economy and has continuously evolved with advancements in technology and changing consumer behavior.

Uses of E-Commerece:

E-commerce, or electronic commerce, has a wide range of uses and applications across various industries and for different purposes. Here are some of the key uses of e-commerce:

1. **Online Retail:** E-commerce enables businesses to sell products and services to customers over the internet. This includes a wide range of products, from clothing and electronics to books and groceries. Major online retailers like Amazon, eBay, and Alibaba operate on this model.

2. **Digital Products and Services:** E-commerce facilitates the sale of digital goods and services, such as e-books, software, online courses, streaming services (Netflix, Spotify), and downloadable music.

3. **B2B E-commerce:** Businesses use e-commerce to purchase goods and services from other businesses. This is common in industries like manufacturing and wholesale.

4. **Online Marketplaces:** E-commerce marketplaces like Etsy, eBay, and Airbnb provide a platform for individuals and businesses to buy and sell products and services.

5. **Auctions:** Online auction platforms like eBay enable users to bid on and purchase items in real-time auctions.

6. **Subscription Services:** E-commerce is used for subscription-based services such as streaming (Netflix, Hulu), meal kit delivery (Blue Apron), and subscription boxes (Birchbox, Loot Crate).

7. **Dropshipping:** Some e-commerce businesses operate on a dropshipping model, where they sell products without holding inventory. When a product is sold, the order is sent to a third-party supplier who ships the product directly to the customer.

8. **Digital Advertising:** E-commerce platforms often utilize digital advertising to promote products and services, including pay-per-click advertising, social media advertising, and search engine marketing.

9. **E-tickets and Reservations:** The travel and entertainment industry uses e-commerce to sell tickets for flights, concerts, movies, and events. Restaurants and hotels also use e-commerce for online reservations.

10. **Crowdfunding:** Crowdfunding platforms like Kickstarter and Indiegogo use e-commerce to facilitate fundraising for creative projects and innovative products.

11. **Banking and Financial Services:** E-commerce is used for online banking, investment services, insurance, and financial transactions, allowing customers to manage their finances online.

12. **Government Services:** E-commerce platforms enable citizens to pay taxes, obtain official documents, and access government services online.

13. **Healthcare and Telemedicine:** E-commerce is increasingly used for booking medical appointments, ordering prescription medications, and accessing telemedicine services.

14. **Education and E-Learning:** E-commerce is essential for online learning platforms and institutions that offer courses, certifications, and educational materials.

15. **Food Delivery:** E-commerce apps and websites like Uber Eats and DoorDash enable customers to order food for delivery from local restaurants.

16. **Charity and Fundraising:** Nonprofits and charitable organizations use e-commerce to accept donations and raise funds for various causes.

17. **Real Estate:** Online real estate platforms allow users to search for and purchase properties or rent apartments.

18. **Art and Collectibles:** E-commerce has made it easier to buy and sell art, collectibles, and rare items through online auctions and marketplaces.

19. **Event Ticketing:** Ticketing services like Eventbrite and Ticketmaster use e-commerce for event ticket sales and management.

20. **Global Trade:** International trade relies on e-commerce platforms to facilitate the buying and selling of goods and services across borders.

These are just a few examples of the many uses of e-commerce. The digital economy continues to evolve, with new e-commerce applications and models emerging regularly.

Define E-commerce:

E-commerce, short for electronic commerce, refers to the buying and selling of goods and services over the internet or other electronic channels. It encompasses a wide range of online activities, including online shopping, electronic payments, digital marketplaces, and internet banking, among others. E-commerce allows businesses to reach a global customer base, and it has become a vital part of the modern economy.

There are several types of e-commerce models, including:

1. Business-to-Consumer (B2C):

In B2C e-commerce, businesses sell products or services directly to individual consumers. Examples include online retail stores and streaming services.

2. Business-to-Business (B2B):

B2B e-commerce involves businesses selling products or services to other businesses. This can include wholesalers, manufacturers, and suppliers using online platforms to trade with one another.

3. Consumer-to-Consumer (C2C):

C2C e-commerce platforms enable individuals to buy and sell goods and services directly to one another. Online auction websites and classified ads platforms are examples of C2C e-commerce.

4. Consumer-to-Business (C2B):

C2B e-commerce reverses the traditional model, where consumers offer products or services, and businesses or organizations purchase them. This model is common in freelance marketplaces and influencer marketing.

5. Government-to-Citizen (G2C) and Government-to-Business (G2B):

In G2C and G2B e-commerce, government agencies provide online services and transactions to citizens or businesses, such as tax payments or permit applications.

E-commerce has revolutionized the way people and businesses conduct trade, providing convenience, accessibility, and a global reach that traditional brick-and-mortar businesses may not have. It has also led to the development of various payment methods and logistics solutions to support the digital marketplace.

HEADER:

Creating an effective header for an e-commerce website is crucial because it's one of the first things visitors see, and it helps them navigate your site. Here's a sample header for an e-commerce website.

1. **Logo:** Your logo, which links to the homepage. Make sure the logo is easily recognizable.

2. **Search Bar:** A search bar that allows users to search for products. This is a crucial feature for an e-commerce site.

3. **Navigation Menu:** A navigation menu with links to important sections of your website, such as products, categories, brands, the shopping cart, and the user's account.

4. **Styling:** Ensure that your header is visually appealing and consistent with your overall website design.

Remember to adapt the code and styling to match your specific e-commerce website's branding and design. Additionally, you may want to include additional elements, such as a user login option, a wishlist link, or promotional banners, depending on your site's requirements and goals.

SEARCH ENGINE:

If you're looking for a search engine specifically designed for ecommerce websites, there are a few popular options to consider. These search engines are tailored to the needs of online retailers and can help improve the search experience for your customers. Here are some notable e-commerce search engines:

1. **Algolia:** Algolia is a powerful search and discovery API for e-commerce sites. It provides fast and relevant search results, real-time indexing, and customizable search features.

2. **Elasticsearch:** Elasticsearch is an open-source search and analytics engine that can be used to create a highly customizable search experience for e-commerce websites. Many e-commerce platforms integrate Elasticsearch to power their search capabilities.

3. **Solr:** Apache Solr is another open-source search platform that is commonly used in e-commerce. It offers features like faceted search, filtering, and scalability.

4. **Swiftype:** Now part of Elastic, Swiftype offers a user-friendly search and discovery solution for e-commerce sites with features like autocomplete, faceted search, and analytics.

5. **SearchSpring:** SearchSpring is a search and merchandising platform designed for e-commerce. It provides features like intelligent autocomplete, faceted search, and personalization.

6. **Klevu:** Klevu is a popular search solution for e-commerce platforms. It offers AIpowered search, merchandising, and analytics to improve the shopping experience.

7. **Sajari:** Sajari is an AI-powered search and discovery platform designed for ecommerce, helping businesses improve search relevance and user experience.

When choosing a search engine for your e-commerce site, consider factors such as your specific requirements, budget, scalability, integration capabilities with your existing e-commerce platform. Additionally, many e-commerce platforms have their built-in search functionality, so you may want to explore those options as well.

PRODUCTS:

Creating an e-commerce app involves offering a wide range of products and providing detailed information about them to help customers make informed purchasing decisions. The specific products and details will depend on the type of ecommerce app and your target market, but here is a general list of product categories and details to consider:

Product Categories:

1. **Electronics:** Smartphones, laptops, tablets, cameras, accessories, and more.
2. **Clothing and Fashion:** Men's, women's, and children's clothing, footwear, accessories, and jewelry.
3. **Home and Furniture:** Furniture, home decor, kitchen appliances, and bedding.
4. **Beauty and Personal Care:** Skincare, makeup, haircare, grooming products, and fragrances.
5. **Health and Wellness:** Vitamins, supplements, fitness equipment, and wellness products.
6. **Books and Media:** Books, e-books, movies, music, and video games.
7. **Toys and Games:** Toys, board games, video games, and hobby items.
8. **Sports and Outdoors:** Sports equipment, outdoor gear, and activewear.
9. **Automotive:** Car accessories, maintenance products, and parts.
10. **Food and Groceries:** Fresh produce, packaged food, and beverages.

Product Details:

1. **Product Name:** Clearly state the product's name.
2. **Price:** Display the price, including any discounts or special offers.

3. **Product Description:** Provide a detailed description of the product, including features, benefits, and specifications.

4. **Images:** High-quality images from different angles and close-ups.

5. **Reviews and Ratings:** Allow customers to leave reviews and rate products.

6. **Product Availability:** Indicate whether the product is in stock or out of stock.

7. **Size and Color Options:** If applicable, allow customers to choose different sizes and colors.

8. **Shipping Information:** Include estimated delivery times and shipping costs.

9. **Returns and Refund Policy:** Explain the return and refund process.

10. **Related Products:** Suggest related or complementary products.

11. **Customer Q&A:** Enable customers to ask questions about the product.

12. **Product Variants:** If you sell variations of a product, such as different models or configurations, display these options.

13. **Sizing Charts:** For clothing and footwear, provide sizing charts.

14. **Warranty Information:** If applicable, include information about product warranties.

15. **Product Videos:** Include videos that demonstrate the product in use.

16. **Product Compare:** Allow customers to compare multiple products.

17. **Inventory Tracking:** Keep track of product availability in real-time.

18. **Social Sharing:** Enable customers to share products on social media.

19. **Product Recommendations:** Use algorithms to suggest relevant products to customers.
20. **Filters and Sorting Options:** Allow users to filter and sort products by criteria like price, popularity, and rating.

21. **Add to Cart and Checkout:** Implement a seamless shopping cart and checkout process.

22. **Payment Options:** Provide various payment methods, including credit cards, digital wallets, and more.

23. **Security:** Ensure the security of customer data and payment information.

24. **Customer Support:** Offer customer support options like live chat, email, or a helpline.

Remember that a well-organized and informative product page is essential for a successful e-commerce app. Make the shopping experience as smooth and convenient as possible for your customers to encourage sales and repeat business.

ADD TO CART:

Adding a "Add to Cart" functionality to an e-commerce website or application is a crucial step in the purchasing process. This feature allows customers to select products they want to buy and store them in a virtual shopping cart until they are ready to proceed to the checkout. Below, I'll provide a high-level overview of how you can implement an "Add to Cart" functionality for e-commerce:

1. Create the Product Database:

You need a database to store product information. This database should include details such as product name, price, quantity available, product ID, and any other relevant information.

2. Design the User Interface:

Design a user-friendly interface that displays product listings. Each product should have an "Add to Cart" button or icon next to it.

3. Shopping Cart Data Structure:

Implement a data structure (usually an array or list) to store the selected products, their quantities, and other relevant information like subtotal.

4. Add to Cart Functionality:

When a user clicks the "Add to Cart" button for a specific product, you need to capture the product's information and add it to the shopping cart data structure.

5. Update Cart Icon or Total:

After adding a product to the cart, update the cart icon in the user interface to reflect the number of items in the cart. You can also show the total price of the items in the cart.

6. View Cart:

Provide an option for users to view their cart. This should show the products they've added, their quantities, and the total price.

7. Modify Cart:

- Allow users to modify their cart by increasing/decreasing quantities or removing items.

8. Checkout:

Implement a "Checkout" button that takes users to the payment and shipping information page.

9. Store Cart Information:

You may need to store cart information in a user's session or database (if users have accounts). This ensures that their cart contents are saved even if they leave and come back later.

10. Payment Processing:

Implement a payment gateway to handle payment processing and order confirmation

11. Order Confirmation:

After a successful transaction, show an order confirmation page and send an email confirmation to the user.

12. Inventory Management:

Update the inventory in the database to reflect the products that have been purchased.

13. Security:

Ensure that the process is secure, including data encryption and handling sensitive user information.

14. Testing and Debugging:

Thoroughly test the entire process to ensure that it works correctly and is free from errors.

15. Optimization and Scalability:

As your e-commerce site grows, optimize your "Add to Cart" functionality for performance and consider scalability options.

This is a simplified overview of implementing "Add to Cart" functionality. In a real-world scenario, there are many more details and features to consider, including handling user accounts, order history, and more. The specific technologies and tools you use will depend on your platform and development stack.

User Authentication:

User authentication for an e-commerce platform is a critical component for securing customer accounts and transactions. Here's how you can implement user authentication for an e-commerce website:

1. User Registration:

Allow users to create accounts by providing their personal information, such as name, email address, and password.

2. Password Storage and Encryption:

Store user passwords securely by using strong encryption and hashing techniques. It's important to hash and salt passwords before storing them in the database to enhance security.

3. Email Verification:

Implement an email verification process where users receive an email with a confirmation link after registering. This ensures that the email address they provided is valid.

4. Social Login:

Offer the option for users to log in or register using their social media accounts (e.g., Facebook, Google) for a more seamless experience.

5. Two-Factor Authentication (2FA):

Allow users to enable 2FA for an additional layer of security. This often involves sending one-time codes to the user's mobile device or email for verification

6. User Profile Management:

Enable users to manage their profiles, including updating personal information, password changes, and security settings.

8. Password Recovery:

Implement a password recovery mechanism that allows users to reset their passwords if they forget them. This typically involves sending a reset link to their registered email address.

8. Account Lockout and Brute-Force Protection:

Implement account lockout mechanisms to temporarily suspend accounts after a certain number of failed login attempts. This helps protect against brute-force attacks.

9. Role-Based Access Control:

Implement role-based access control to differentiate between regular users, administrators, and other roles. Define what actions each role can perform within the platform.

10. Privacy and Data Protection Compliance:

Ensure that your authentication system complies with data protection regulations like GDPR, especially regarding user data handling and consent.

11. Session Management:

Use secure session management to keep users logged in after authentication. Consider using tokens or cookies with limited lifespans.

12. HTTPS:

Always use HTTPS to encrypt data transmission between the client and server, especially during login and account management.

13. Security Audits and Penetration Testing:

Regularly conduct security audits and penetration testing to identify vulnerabilities in your authentication system.

14. User Agreement and Terms of Service:

Require users to agree to your platform's terms of service and privacy policy during registration.

15. Logging and Monitoring:

Implement logging and monitoring to track authentication-related activities, including successful logins, failed login attempts, and changes to user accounts.

16. Password Policies:

Enforce strong password policies, such as minimum length, complexity requirements, and expiration intervals.

17. Password Storage Management:

Periodically update password storage mechanisms and algorithms to align with best practices and industry standards.

18. Third-Party Payment Providers:

If your e-commerce platform processes payments, consider using established third-party payment providers that offer secure and authenticated payment processes.

User Registration:

User registration is a fundamental component of an e-commerce app that allows users to create accounts and provides a personalized shopping experience. Here are the key steps and considerations for implementing user registration for an e-commerce app:

1. User Sign-Up Page:

Create a user-friendly registration page where users can provide their information to create an account. This typically includes fields for:

- Name
- Email address
- Password
- Confirm password

2. Registration Form Validation:

Implement client-side and server-side validation to ensure that users enter valid and complete information. Check for email format, password strength, and other relevant details.

3. Password Security:

Enforce strong password policies, including minimum length, complexity requirements, and possibly password expiration intervals. Hash and salt user passwords before storing them in the database.

4. Email Verification:

Send a verification email to the user's provided email address. This email should contain a unique verification link that the user must click to confirm their registration.

5. Social Sign-Up:

Offer users the option to register using their social media accounts (e.g., Google, Facebook) for a quicker and more convenient registration process.

6. Terms and Conditions:

Include a checkbox for users to agree to your app's terms and conditions or privacy policy during registration.

7. Captcha or Anti-Bot Measures:

Implement Captcha or other anti-bot mechanisms to prevent automated registration by bots.

8. Email Confirmation:

Once the user clicks the verification link in their email, confirm their registration and activate their account. Provide feedback to the user that their registration was successful.

9. Welcome Email:

Send a welcome email to the user, thanking them for registering and providing information about your app's features and benefits.

10. User Profile:

After registration, allow users to complete their profile, including additional information such as shipping addresses, contact details, and profile pictures.

11. User Authentication:

Implement secure authentication mechanisms to verify users during the login process. Use strong encryption and hashing for password storage.

12. User Session Management:

Create and manage user sessions securely to keep users logged in between visits. Use tokens or cookies with limited lifespans.

13. Security:

Ensure that user registration and authentication processes are secure, including data encryption, secure handling of user information, and compliance with industry security standards.

14. Mobile Responsiveness:

Ensure that the registration process is mobile-friendly to accommodate users on different devices.

15. Privacy and Data Protection Compliance:

Comply with data protection regulations, such as GDPR, and allow users to control their data and privacy settings.

16. Testing and Debugging:

Thoroughly test the registration process to ensure that it works correctly and is free from errors. Consider edge cases and potential issues in the user flow.

17. Account Deactivation and Removal:

Implement a process that allows users to deactivate or delete their accounts if they choose to do so.

18. Legal Compliance:

Ensure compliance with relevant e-commerce laws and regulations, especially regarding user data protection and consent.

Implementing a secure and user-friendly user registration process is crucial for the success of your e-commerce app. By following the steps and considerations above, you can create a robust and compliant registration system for your users.

SHOPING CART:

Creating a shopping cart for an e-commerce website or application is essential to enable users to select and manage the items they want to purchase. Below are the key steps and considerations for implementing a shopping cart:

1. Database for Products;

Maintain a database that stores product information, including details like product name, price, description, availability, and a unique product ID.

2. User Interface:

Design a user-friendly interface that displays product listings. Each product should have an "Add to Cart" button and provide a clear indication of product details, including images.

3. Shopping Cart Data Structure:

Implement a data structure (usually an array, list, or object) to store the selected products, their quantities, and other relevant information, such as subtotals.

4. Add to Cart Functionality:

When a user clicks the "Add to Cart" button for a specific product, capture the product's information, including its ID, quantity, and price, and add it to the shopping cart data structure.

5. Update Cart Icon or Total:

After adding a product to the cart, update the cart icon or the total price displayed on the user interface to reflect the items in the cart and their combined value.

6. View Cart:

Provide users with an option to view their shopping cart. This should display the products they've added, their quantities, and the total price of the cart.

7. Modify Cart:

Allow users to modify their shopping cart by adjusting product quantities or removing items. This typically involves providing "Increase," "Decrease," and "Remove" options for each item in the cart.

8. Checkout Process:

Implement a "Checkout" button that takes users to the payment and shipping information page. The cart contents are transferred to the checkout process.

9. Store Cart Information:

Store cart information for the user. You can use cookies, sessions, or database entries to keep track of the user's cart contents, allowing them to return to their cart even if they leave and come back later.

10. Payment Processing:

Implement a payment gateway to handle payment processing. This can include options for credit card payments, PayPal, and other popular payment methods.

11. Inventory Management:

Update the inventory in the database to reflect the products that have been purchased, ensuring that the quantities are reduced accordingly.

12. Security:

Ensure that the checkout and payment process is secure, including data encryption and secure handling of sensitive user information.

13. Testing and Debugging:

Thoroughly test the entire shopping cart process to ensure it works correctly and is free from errors. This includes testing for edge cases and potential issues in the user flow.

14. Optimization and Scalability:

As your e-commerce site grows, optimize the shopping cart functionality for performance and consider scalability options to handle a larger user base and increased traffic.

15. Mobile Responsiveness:

Ensure that your shopping cart and checkout process are mobile-friendly to accommodate users on different devices.

16. Promo Codes and Discounts:

If applicable, provide the option for users to apply promo codes or discounts to their order during the checkout process.

17. Order Confirmation:

After a successful transaction, display an order confirmation page and send an email confirmation to the user with order details.

Creating a smooth and user-friendly shopping cart experience is crucial for e-commerce success. The above steps will help you implement the essential features required for a functional shopping cart on your e-commerce platform.

CHECKOUT FUNCTIONALITY:

Implementing a checkout functionality for an e-commerce app is a crucial step in the buying process. It's the point at which customers provide shipping and payment information to complete their purchase. Below are the key steps and considerations for implementing a checkout functionality:

1. User Authentication:

Ensure that the user is authenticated or registered before proceeding to checkout. If not, prompt them to log in or register.

2. Cart Review:

Display a summary of the items in the user's shopping cart, including product details, quantities, prices, and subtotals.

3. Shipping Information:

Allow users to provide or select their shipping information, including the shipping address, contact details, and preferred shipping method (e.g., standard, expedited).

4. Payment Information:

Collect payment details from users. This may involve credit card information, PayPal, or other payment methods. Implement secure and PCI-compliant payment processing.

5. Billing Address:

If the billing address is different from the shipping address, allow users to provide a separate billing address.

6. Promo Codes and Discounts:

Provide a field for users to enter promo codes, coupons, or discounts if applicable. Apply these discounts to the order total.

7. Order Summary:

Display a summary of the entire order, including the items, shipping costs, taxes, discounts, and the final order total.

8. Order Review:

Allow users to review their order and make any necessary changes or updates before finalizing the purchase.

9. Payment Authorization:

Securely process the payment and authorize the transaction, ensuring that the user's payment method is valid and has sufficient funds.

10. Confirmation Page:

After successful payment authorization, display an order confirmation page with a unique order number. Include a summary of the order, shipping details, and estimated delivery date.

11. Email Confirmation:

containing all order details, including the order number, items, and shipping information.

12. Inventory Management:

Deduct the purchased items from the inventory database to ensure that the products are marked as sold and unavailable for further purchases.

13. Order History:

Store the order information in the user's account for future reference and tracking.

14. Shipping Tracking:

Provide a link or method for users to track the status of their shipment, including estimated delivery dates and tracking numbers.

15. Return and Refund Policies:

Include information about the return and refund policies in the confirmation email and on the confirmation page, providing details on how users can request returns or refunds.

16. Guest Checkout:

Allow users to complete a purchase without creating an account by providing an option for guest checkout. However, encourage them to register for easier tracking of orders and better user experience.

17. Security:

Ensure that the checkout process is secure, including data encryption, secure handling of payment information, and compliance with industry standards for online transactions.

18. Mobile Responsiveness:

Make sure the checkout process is mobile-friendly to accommodate users on various devices.

19. User Support:

Offer customer support options, such as chat or email, in case users encounter issues during the checkout process.

20. Testing and Debugging:

Thoroughly test the entire checkout process to ensure it works correctly and is free from errors. Test with various payment methods and shipping scenarios.

21. Legal Compliance:

Ensure compliance with relevant e-commerce laws, including tax collection and data protection regulations.

FOTTER:

Creating a footer for an e-commerce application is an essential part of the overall design. The footer provides valuable information and navigation options to users, making their experience more convenient. Here's what you should consider including in the footer of your e-commerce application:

1. Contact Information:

- Customer support email or phone number.
- Physical address (if applicable).

2. Links to Important Pages:

- About Us: Information about your company.
- FAQ: Frequently asked questions and answers.
- Shipping and Returns: Details about shipping policies and return procedures.
- Privacy Policy: Information about user data and privacy.
- Terms and Conditions: Legal terms for using the application.

3. Categories and Quick Links:

- Categories: Links to popular product categories.
- Search: A search bar for quick product searches.
- Featured Products: Links to bestsellers or featured items.
- New Arrivals: Links to recently added products.

4. Account and User Management:

- My Account: A link to user account settings.

- Order History: Access to order history and tracking.
- Wishlist: Link to saved items.
- Sign In/Sign Up: Options for user authentication.

5. Social Media Links:

- Icons or links to your social media profiles.
- Share and follow options for social engagement.

6. Newsletter Signup:

- A form for users to subscribe to newsletters for updates and promotions.

7. Payment and Security:

- Icons or badges for accepted payment methods.
- Security badges for trust and credibility.

8. Customer Support:

- Live Chat: Option for live customer support.
- Email Support: Link to contact customer service.
- Help Center: Additional help and support resources.

9. Language and Currency Selector:

- Options for users to select their preferred language and currency.

10. Copyright Information:

- The copyright notice and the year, e.g., "© 2023 Your Company Name."

11. Accessibility Options:

- Links to accessibility features and resources for all users.

12. Sitemap:

- A link to a sitemap for easy navigation.

13. Mobile App Links:

- Links to download your mobile application (if applicable).

14. Return to Top Button:

- A button that allows users to quickly scroll back to the top of the page.

15. Miscellaneous:

- Promotions, discounts, or announcements.

When designing your footer, keep it clean and well-organized. You can use columns, icons, and separators to make it visually appealing and easy to navigate. Ensure that the font size and color scheme are consistent with the overall design of your e-commerce application. Test the footer on different devices and screen sizes to ensure it's responsive and user-friendly.

PROGRAM:

HTML:

```
                    <!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <title>my project1</title>

</head>

<link rel="stylesheet" href="css/style.css">

<script type="module"
src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.
js"></script>

<body>

   <header>

      <div class="nav"> add to cart

         <a href="#" class="logo">

            <ion-icon  class="add-cart" name="cart-outline">

               online order

            </ion-icon>

         </a>

      </div>
```

```
    </header>


    <h1 class="sri">e-commerece</h1>


    <h1 class="he1">E-commerce Application</h1>

    <form action="search_results.php" method="GET">

        <input type="text" class="form" placeholder="Search for
products">

        <input type="submit"  class="btn"value="Search">

    </form>


    <div class="container">

    <div class="shop-content">


    <div class="box">

        <div class="pic">

            <img src="laptop.jpg"  class="img">

            <div class="name">laptop

                <div class="amount">RS.50000


                </div>

            </div>
```

```
<div class="box">
<div class="pic">
    <img src="watch.jpg"  class="img">
    <div class="name">watch
        <div class="amount">RS.500


        </div>
    </div>


        <div class="box">
    <div class="pic">
        <img src="mobile.webp"  class="img">
        <div class="name">mobile
            <div class="amount">RS.5000


            </div>
        </div>


        <div class="box">
            <div class="pic">
                <img src="speaker.webp"  class="img">
```

```
<div class="name">SPEAKER</div>

    <div class="amount">RS.3000



    </div>

</div>



<div class="box">

    <div class="pic">

        <img src="television.jpg"  class="img">

        <div class="name">TELEVISION

            <div class="amount">RS.10000



            </div>

        </div>



        <div class="box">

            <div class="pic">

                <img src="camera.jpg"  class="img">

                <div class="name">  CAMERA

                    <div class="amount">RS.8000



                    </div>
```

```
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <footer> @frontend e-commerece application 2023</footer>


    <script src="js/script.js"></script>
</body>
</html>


<!DOCTYPE html>
<html>
<head>
    <title>E-commerce Navigation Menu</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
```

```
</head>

<body>

    <nav>

        <ul>

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

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

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

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

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

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

        </ul>

    </nav>

</body>

</html>
```

CSS:

```
            @import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@1
,100&display=swap');



*{

    margin: 0;

    padding: 0;
    box-sizing: border-box;

    font-family: Verdana, Geneva, Tahoma, sans-serif;

    font-family: 'Poppins', sans-serif;


}


.sri

{

    font-size: 80px;

    font-weight: 1000;

    text-align: center;

    text-transform: uppercase;

    margin-top: 60px;

    transition: 0.2s;

}


body{

    width: 100%;

    min-height: 100vh;

    background-image: linear-gradient(to right, #616161, #9bc5c3);


}
```

```
img

{

   width: 100%;

}


.shop-content

{

      display: grid;

      grid-template-columns: 300px 300px 300px;

      justify-content: center;

      align-items: center;

      gap: 20px;

}


.container

{

   max-width: 1200px;

   padding: 4rem;

   margin: auto;


}
```

```
.box

{


    padding: 4px;

    border-radius: 6px;

    gap: 30px;

    width: 300px;


}


.name

{

    font-size: 20px;

    font-weight: 800;

    text-transform: uppercase;

    color: azure;

}


.pic

{

    overflow: hidden;
}

.pic:hover img
{
    transform: scale(1.5);
}

.img
{
    transition: 0.2s;
    aspect-ratio: 1/1;
    object-fit: cover;
}

.he1
{
    font-size: 30px;
    font-weight: 600;
    text-align: center;
    text-transform: uppercase;
}
```

```
.form
{
    width: 50%;
    max-width: 1000px;
    margin-left: 300px;
    border-radius: 6px;
    background-color: rgb(203, 214, 214);
    border-color: rgb(210, 231, 190);
}

.btn
{
    background-color: brown;
    border-radius: 6px;
    color: white;
}

.sri:hover
{
    font-size: 120px;
}
```

```
.logo

{

   font-size: 50px;

   font-weight: 600;

   text-transform: uppercase;

   color: white;

   text-align: center;

}


header

{

   width: 100%;

   top: 0;

   right: 0;

   background-color: #ff4757;

}


.nav

{

   font-size: 50px;

   font-weight: 800;

   text-transform: uppercase;
```

```
    text-align: center;
}

footer{
    width: 120%;
    height: 30px;
    background-color: black;
    color: white;
    text-transform: uppercase;
    border-radius: 10px;
    font-weight: 500;

}

nav
{
    width: 180%;
    height: 30px;
    top: 0;
    right: 0;
    background-color: #af932d;
}
```

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

li {
    display: inline;
    margin-right: 20px;
}

a {
    text-decoration: none;
    color: #fff;
    font-weight: bold;
    font-size: 18px;
}

a:hover
{
   color: #ff9900;
}
```

JAVA SCRIPT:

```
                    const products = [
   { id: 1, name: "Product 1", price: 10.00 },
   { id: 2, name: "Product 2", price: 15.00 }
];

const cart = [];
let cartTotal = 0;

const cartItemsElement = document.getElementById("cart-items");

const cartTotalElement = document.getElementById("cart-total");

function addToCart(product) {
   cart.push(product);
   cartTotal += product.price;
   updateCartDisplay();
}

function removeFromCart(product) {
   const index = cart.indexOf(product);
   if (index > -1) {
        cart.splice(index, 1);

        cartTotal -= product.price;

        updateCartDisplay();

    }

}


function updateCartDisplay()

{

    cartItemsElement.innerHTML = "";

    cart.forEach(item =>

 {

        const listItem = document.createElement("li");

        listItem.textContent = `${item.name} -
$${item.price.toFixed(2)}`;

        const removeButton = document.createElement("button");

        removeButton.textContent = "Remove";

        removeButton.addEventListener("click", () =>
removeFromCart(item));

        listItem.appendChild(removeButton);

        cartItemsElement.appendChild(listItem);

    });

    cartTotalElement.textContent = cartTotal.toFixed(2);

}
```

```
document.addEventListener("DOMContentLoaded", () =>

 {
    products.forEach(product =>

 {
        const productElement = document.querySelector(`[data-
id="${product.id}"]`);

        productElement.querySelector(".add-to-
cart").addEventListener("click", () => addToCart(product));
    });
});

const menuElement = document.getElementById("menu");

const menuItems = menuElement.querySelectorAll("li");

menuItems.forEach(item => {
    item.addEventListener("click", () => {
    });
});
```

JAVA:

```
package co.sandyedemo.ecomdemo.Activities;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.view.WindowManager;

import android.view.inputmethod.InputMethodManager;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.Toast;


import co.sandyedemo.ecomdemo.Common;

import co.sandyedemo.ecomdemo.Config;

import co.sandyedemo.ecomdemo.MVP.SignUpResponse;

import co.sandyedemo.ecomdemo.Retrofit.Api;

import co.sandyedemo.ecomdemo.R;


import java.util.List;
import butterknife.BindView;

import butterknife.BindViews;

import butterknife.ButterKnife;

import butterknife.OnClick;

import cn.pedant.SweetAlert.SweetAlertDialog;

import retrofit.Callback;

import retrofit.RetrofitError;

import retrofit.client.Response;


public class Login extends AppCompatActivity {


    @BindViews({R.id.email, R.id.password})

    List<EditText> editTexts;

    @BindView(R.id.loginLinearLayout)

    LinearLayout loginLinearLayout;

    @BindView(R.id.appIcon)

    ImageView appIcon;


    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_login);
        ButterKnife.bind(this);

        this.getWindow().setFlags(WindowManager.LayoutParams.FL
AG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        loginLinearLayout.setOnClickListener(new
View.OnClickListener() {

            @Override

            public void onClick(View view) {

                hideKeyboard(view);



            }

        });



    }

    protected void hideKeyboard(View view)

    {

        InputMethodManager in = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

        in.hideSoftInputFromWindow(view.getWindowToken(), 0);

    }

    @OnClick({R.id.txtSignUp, R.id.txtForgotPassword,
R.id.skipLoginLayout, R.id.signIn})

    public void onClick(View view) {

        switch (view.getId()) {
    }

            case R.id.txtSignUp:

                Config.moveTo(Login.this, SignUp.class);

                break;

            case R.id.txtForgotPassword:

                Config.moveTo(Login.this, ForgotPassword.class);

                break;

            case R.id.skipLoginLayout:

                Intent intent = new Intent(Login.this, MainActivity.class);

                intent.putExtra("from", "skip");

                startActivity(intent);

                finishAffinity();

                break;



            case R.id.signIn:

                if (Config.validateEmail(editTexts.get(0),Login.this) &&
validatePassword(editTexts.get(1))) {

                    login();

                }

                break;

        }

    }
    private boolean validatePassword(EditText editText) {

        if (editText.getText().toString().trim().length() > 5) {

            return true;

        } else if (editText.getText().toString().trim().length() > 0) {

            editText.setError("Password must be of 6 characters");

            editText.requestFocus();

            return false;

        }

        editText.setError("Please Fill This");

        editText.requestFocus();

        return false;

    }


    private void login() {

        final SweetAlertDialog pDialog = new
SweetAlertDialog(Login.this, SweetAlertDialog.PROGRESS_TYPE);

        pDialog.getProgressHelper().setBarColor(getResources().getC
olor(R.color.colorPrimary));

        pDialog.setTitleText("Loading");

        pDialog.setCancelable(false);

        pDialog.show();

        // sending gcm token to server

        Api.getClient().login(editTexts.get(0).getText().toString().trim(),
                editTexts.get(1).getText().toString().trim(),

                "email",

                new Callback<SignUpResponse>() {

                    @Override

                    public void success(SignUpResponse
signUpResponse, Response response) {

                        pDialog.dismiss();

                        Log.d("signUpResponse",
signUpResponse.getUserid() + "");

                        Toast.makeText(Login.this,
signUpResponse.getMessage(), Toast.LENGTH_SHORT).show();

                        if
(signUpResponse.getSuccess().equalsIgnoreCase("true")) {

                            Common.saveUserData(Login.this, "email",
editTexts.get(1).getText().toString());

                            Common.saveUserData(Login.this, "userId",
signUpResponse.getUserid() + "");

                            Config.moveTo(Login.this, MainActivity.class);

                            finishAffinity();

                        } else if
(signUpResponse.getSuccess().equalsIgnoreCase("notactive")) {

                            Config.moveTo(Login.this,
AccountVerification.class);

                        }

                    }
                    @Override
                    public void failure(RetrofitError error) {
                        pDialog.dismiss();


                        Log.e("error", error.toString());
                    }
                });
        }
}
package co.sandyedemo.ecomdemo.Activities;


import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


import co.sandyedemo.ecomdemo.Config;
import co.sandyedemo.ecomdemo.MVP.SignUpResponse;
import co.sandyedemo.ecomdemo.R;
import co.sandyedemo.ecomdemo.Retrofit.Api;


import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;
import cn.pedant.SweetAlert.SweetAlertDialog;
import retrofit.Callback;
import retrofit.RetrofitError;
import retrofit.client.Response;


public class AccountVerification extends AppCompatActivity {


    @BindView(R.id.resendEmail)
    Button resendEmail;
    @BindView(R.id.email)
    EditText email;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_account_verification);
        ButterKnife.bind(this);
```

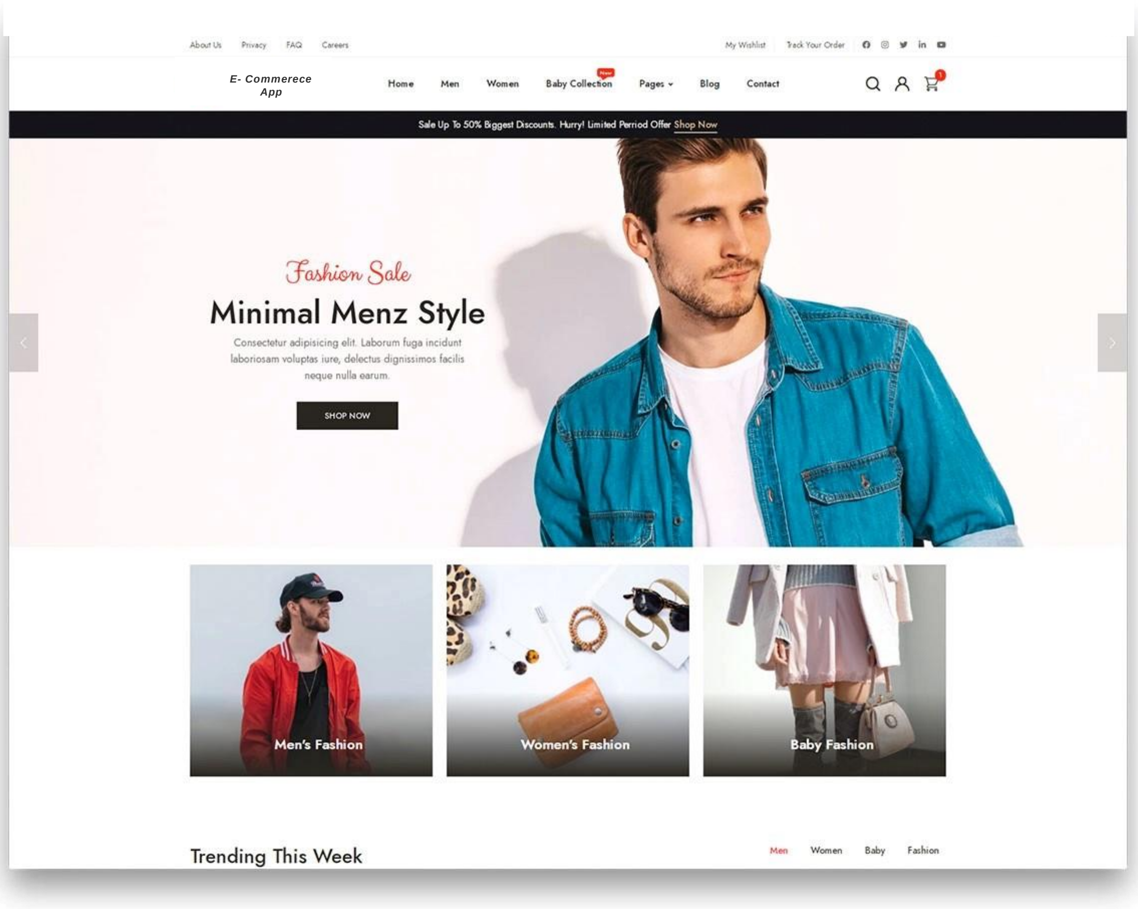
```
}


@OnClick({R.id.resendEmail, R.id.signUp,R.id.login, R.id.back})
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.resendEmail:
            if (Config.validateEmail(email,AccountVerification.this))
            resendEmail();
            break;
        case R.id.signUp:
            Config.moveTo(AccountVerification.this, SignUp.class);
            break;
        case R.id.login:
            Config.moveTo(AccountVerification.this, Login.class);
            break;
        case R.id.back:
            finish();
            break;
    }
}


private void resendEmail() {
        final SweetAlertDialog pDialog = new
SweetAlertDialog(AccountVerification.this,
SweetAlertDialog.PROGRESS_TYPE);

        pDialog.getProgressHelper().setBarColor(getResources().getC
olor(R.color.colorPrimary));

        pDialog.setTitleText("Loading");

        pDialog.setCancelable(false);

        pDialog.show();

        // sending gcm token to server

        Api.getClient().resentEmail(email.getText().toString().trim(),

                new Callback<SignUpResponse>() {

                    @Override

                    public void success(SignUpResponse
signUpResponse, Response response) {

                        pDialog.dismiss();

                        Log.d("resendEmailResponse",
signUpResponse.getSuccess() + "");

                        Toast.makeText(AccountVerification.this,
signUpResponse.getMessage(), Toast.LENGTH_SHORT).show();

                        if
(signUpResponse.getSuccess().equalsIgnoreCase("true")) {

                            finish();

                        }

                    }
                    @Override
                    public void failure(RetrofitError error) {
                        pDialog.dismiss();
                        Log.e("error", error.toString());
                    }
                });
    }
}
package co.sandyedemo.ecomdemo.Activities;


import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;


import co.sandyedemo.ecomdemo.Config;
import co.sandyedemo.ecomdemo.MVP.SignUpResponse;
import co.sandyedemo.ecomdemo.Retrofit.Api;
import co.sandyedemo.ecomdemo.R;
import java.util.List;


import butterknife.BindViews;

import butterknife.ButterKnife;

import butterknife.OnClick;

import cn.pedant.SweetAlert.SweetAlertDialog;

import retrofit.Callback;

import retrofit.RetrofitError;

import retrofit.client.Response;


public class ForgotPassword extends AppCompatActivity {

    @BindViews({R.id.emailId})

    List<EditText> editTexts;


    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_forgot_password);

        ButterKnife.bind(this);


    }


    @OnClick({R.id.back,R.id.submit})
    public void onClick(View view) {

        switch (view.getId()) {

            case R.id.back:

                finish();

                break;

            case R.id.submit:

                if
(Config.validateEmail(editTexts.get(0),ForgotPassword.this)) {

                    forgotPassword();

                }

                break;

        }

    }

    private void forgotPassword() {

        final SweetAlertDialog pDialog = new
SweetAlertDialog(ForgotPassword.this,
SweetAlertDialog.PROGRESS_TYPE);

        pDialog.getProgressHelper().setBarColor(getResources().getC
olor(R.color.colorPrimary));

        pDialog.setTitleText("Loading");

        pDialog.setCancelable(false);

        pDialog.show();

        Api.getClient().forgotPassword(editTexts.get(0).getText().toStri
ng().trim(),
                new Callback<SignUpResponse>() {

                    @Override

                    public void success(SignUpResponse
signUpResponse, Response response) {

                        pDialog.dismiss();

                        Log.d("signUpResponse",
signUpResponse.getMessage());

                        Toast.makeText(ForgotPassword.this,
signUpResponse.getMessage(), Toast.LENGTH_SHORT).show();

                        if
(signUpResponse.getSuccess().equalsIgnoreCase("true")) {

                            Config.moveTo(ForgotPassword.this, Login.class);

                            finishAffinity();

                        }

                   }

                    @Override

                    public void failure(RetrofitError error) {

                        pDialog.dismiss();

                        Log.e("error", error.toString());



                    }

                });

    }

}
```

PYTHON:

```
                    from flask import Flask, render_template, session,
redirect, url_for



app = Flask(__name)

app.secret_key = 'your_secret_key'  # Replace with a strong secret
key.



# Sample product data

products = [

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

    {'id': 2, 'name': 'Product 2', 'price': 15.00},

]



@app.route('/')

def index():

    return render_template('index.html')



@app.route('/products')

def view_products():

    return render_template('products.html', products=products)



@app.route('/product/<int:product_id>')
```

```
def view_product(product_id):
    product = next((p for p in products if p['id'] == product_id), None)
    if product:
        return render_template('product.html', product=product)
    return "Product not found"


@app.route('/add_to_cart/<int:product_id>')
def add_to_cart(product_id):
    product = next((p for p in products if p['id'] == product_id), None)
    if product:
        if 'cart' not in session:
            session['cart'] = []
        session['cart'].append(product)
    return redirect(url_for('index'))


@app.route('/cart')
def view_cart():
    cart = session.get('cart', [])
    total = sum(product['price'] for product in cart)
    return render_template('cart.html', cart=cart, total=total)
if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT:


About Us
Privacy
FAQ
Careers
My Wishlist
Track Your Order
E- Commerece App
Home
Men
Women
Baby Collection
New
Pages
Blog
Contact
Sale Up To 50% Biggest Discounts. Hurry! Limited Perriod Offer Shop Now
Fashion Sale
Minimal Menz Style
Consectetur adipisicing elit. Laborum fuga incidunt laboriosam voluptas iure, delectus dignissimos facilis neque nulla earum.
SHOP NOW
Men's Fashion
Women's Fashion
Baby Fashion
Trending This Week
Men
Women
Baby
Fashion