COLLEGE CODE : 8207

COLLEGE NAME : AS-SALAM COLLEGE OF ENGINEERING & TECHNOLOGY

DEPARTMENT : AI&DS

STUDENT NM-ID : D15B7341355FE75B6044538B51952A5E
: 2A213E4AACF1FE0A9A747DA2415B7DE5
: 1B2EF2B073168EDB2DD4D5D8904806F0
: 8732B626D7C9A3091E843F6C0DBB8934

ROLL NO :820723243004 ,820723104023 ,820723243008
820723243003

DATE : 26/09/2025

Completed the project named as Phase 4

TECHNOLOGY PROJECT NAME : COMMERCE CART SYSTEM

SUBMITTED BY,

NAME :	MOBILE NO :
DHINA S	9677383024
VISHAL R	82704 22959
MAHALAKSHMI R	88708 99846
BAVIKARAN S	90803 76434

Phase 4 — Enhancements & Deployment

(Deadline – Week 9)

1. Additional Features

- **Wishlist/Save for Later:** Allow users to save products without immediate purchase.
- **Coupon & Discount System:** Promo codes and seasonal discounts.
- **Multi-payment Gateway Integration:** Support for UPI, credit/debit cards, wallets.
- **Order Tracking:** Real-time shipment and delivery status updates.
- **Product Recommendation Engine:** Personalized suggestions using past user behavior.

2. UI/UX Improvements

- **Responsive Design:** Mobile-first UI for better shopping experience.
- **Improved Checkout Flow:** One-click checkout with auto-filled details.
- **Accessibility Features:** High-contrast mode, voice search, keyboard navigation.
- **Product Filters & Sorting:** Easy search by category, price, rating, etc.
- **Micro-interactions:** Smooth animations, hover effects, and instant feedback.

3. API Enhancements

- **Authentication & Authorization:** Secure JWT tokens for login.
- **Scalable Endpoints:** Optimized APIs for products, cart, orders, and payments.
- **Third-Party Integration:** Shipping API and

- **Micro-interactions:** Smooth animations, hover effects, and instant feedback.

3. API Enhancements

- **Authentication & Authorization:** Secure JWT tokens for login.
- **Scalable Endpoints:** Optimized APIs for products, cart, orders, and payments.
- **Third-Party Integration:** Shipping API and payment gateway APIs.
- **Error Handling & Logging:** Detailed error responses and monitoring tools.
- **Versioning:** Maintain backward compatibility while adding new features.

4. Performance & Security Checks

- **Performance Testing:** Load testing with 1000+ concurrent users.
- **Database Optimization:** Indexing, caching,

- **Versioning:** Maintain backward compatibility while adding new features.

4. Performance & Security Checks

- **Performance Testing:** Load testing with 1000+ concurrent users.
- **Database Optimization:** Indexing, caching, and query tuning.
- **Security Measures:** SSL encryption, input validation, and data masking.
- **Penetration Testing:** Identify and fix vulnerabilities.
- **Backup & Recovery Plan:** Regular database backup and disaster recovery system.

5. Deployment Strategy

- **Environment Setup:** Separate dev, staging, and production e[illegible]ments.
- **CI/CD Pipeline:** Automated build, test, and

system.

5. Deployment Strategy

- **Environment Setup:** Separate dev, staging, and production environments.
- **CI/CD Pipeline:** Automated build, test, and deployment using GitHub Actions.
- **Cloud Hosting:** Deploy on AWS/Heroku with auto-scaling.
- **Monitoring & Analytics:** Track errors, user behavior, and system health.
- **Feedback Loop:** Collect user feedback post-deployment for future improvements.

✅ This makes a neat **4–5 page PDF**.

Do you want me to g ↓ te the **PDF file** directly for you now?

```
# Phase 4 - Enhanced E-Commerce Cart
System
# Features: Add to Cart, View Cart,
Remove Item, Checkout

cart = []

def add_to_cart(product_name, price,
quantity):
    cart.append({
        "product_name": product_name,
        "price": price,
        "quantity": quantity
    })
    print(f"✅ {product_name} added to
cart successfully!")

def view_cart():
    if not cart:
        print("🛒 Your cart is empty.")
    else:
        print("\n--- Cart Items ---")
        total = 0
        for item in cart:
            subtotal = item['price'] *
item['quantity']

print(f"{item['product_name']} -
{item['quantity']} x ₹{item['price']} =
₹{subtotal}")
            total += subtotal
        print(f"💰 Total: ₹{total}\n")

def remove_item(pr    t_name):
    for item in c
        if item["pr    uct_name"] ==
```

```
            subtotal = item['price'] *
item['quantity']

print(f"{item['product_name']} -
{item['quantity']} x ₹{item['price']} =
₹{subtotal}")
            total += subtotal
        print(f"💰 Total: ₹{total}\n")

def remove_item(product_name):
    for item in cart:
        if item["product_name"] ==
product_name:
            cart.remove(item)
            print(f"🗑️ {product_name}
removed from cart.")
            return
    print("❌ Item not found in cart.")

def checkout():
    if not cart:
        print("❌ Your cart is empty.")
    else:
        print("✅ Checkout successful!
Your order has been placed.")
        cart.clear()

# ---------- INPUT / OUTPUT EXAMPLE
----------
add_to_cart("Laptop", 55000, 1)
add_to_cart("Headphones", 2000, 2)
view_cart()
remove_item("Headphones")
view_cart()
checkout()
```

```
✅ Laptop added to cart successfully!
✅ Headphones added to cart
successfully!

--- Cart Items ---
Laptop - 1 x ₹55000 = ₹55000
Headphones - 2 x ₹2000 = ₹4000
💰 Total: ₹59000

🗑️ Headphones removed from cart.

--- Cart Items ---
Laptop - 1 x ₹55000 = ₹55000
💰 Total: ₹55000

✅ Checkout successful! Your order has
been placed.
```