Ecommerce Lab

***Note: To run all the program: node your\_file\_name.js [make sure the node JS is installed in your system]***

**1. Write a program to create products category list and add items to the created categories in an e-commerce store. Display items with their categories. You can write program in any of the language as per your comfort. The design is not required. Database is optional,you can also work using array or any other data structure as per your comfort. Though if you opt to use database,you can.**

-> Solution (In JS):

| *// This is the start of a class called EcommerceStore - think of it as a blueprint for our store*  *class EcommerceStore {*  *// This is the constructor - it runs when we create a new store*  *constructor() {*  *// Creates an empty object to hold our categories (like a filing cabinet)*  *// Each category will be a drawer, and items will be files in those drawers*  *this.categories = {};*  *}*  *// Method to create a new category (like adding a new drawer to our cabinet)*  *createCategory(categoryName) {*  *// Checks if the category doesn't exist yet*  *// !this.categories[categoryName] means "if there's no drawer with this name"*  *if (!this.categories[categoryName]) {*  *// Creates a new empty array (drawer) for this category name*  *this.categories[categoryName] = [];*  *// Prints a success message to the console (like a confirmation note)*  *console.log(`Category "${categoryName}" created successfully`);*  *} else {*  *// If the category already exists, just let us know*  *console.log(`Category "${categoryName}" already exists`);*  *}*  *}*  *// Method to add an item to a category (like putting a file in a drawer)*  *addItem(categoryName, item) {*  *// Checks if the category (drawer) exists*  *if (!this.categories[categoryName]) {*  *// If it doesn't exist, tells us it's creating it*  *console.log(`Category "${categoryName}" doesn't exist. Creating it now...`);*  *// Calls the createCategory method to make the new category*  *this.createCategory(categoryName);*  *}*  *// Creates a new object with details about the item*  *const itemDetails = {*  *// Gives the item a unique ID using the current time in milliseconds*  *id: Date.now(),*  *// Takes the item's name from what we passed in*  *name: item.name,*  *// Takes the item's price from what we passed in*  *price: item.price,*  *// Takes the stock number, uses 0 if none provided*  *stock: item.stock || 0*  *};*  *// Adds the item to the category's array (puts file in drawer)*  *this.categories[categoryName].push(itemDetails);*  *// Confirms the item was added with a message*  *console.log(`Item "${item.name}" added to "${categoryName}" category`);*  *}*  *// Method to show everything in our store*  *displayCatalog() {*  *// Prints a header for our catalog*  *console.log("\n=== E-commerce Store Catalog ===");*  *// Loops through each category and its items*  *// Object.entries turns our object into pairs of [category, items]*  *for (const [category, items] of Object.entries(this.categories)) {*  *// Shows category name and how many items it has*  *console.log(`\n${category} (${items.length} items):`);*  *// Loops through each item in the category*  *items.forEach(item => {*  *// Prints item details with nice formatting*  *// \n means "new line" to make it readable*  *console.log(*  *`- ${item.name} (ID: ${item.id})` +*  *`\n Price: Rs.${item.price}` +*  *`\n Stock: ${item.stock} units`*  *);*  *});*  *}*  *}*  *// Method to get all items from a specific category*  *getItemsByCategory(categoryName) {*  *// Checks if the category exists*  *if (this.categories[categoryName]) {*  *// Returns the array of items if it does*  *return this.categories[categoryName];*  *} else {*  *// If category doesn't exist, tells us and returns empty array*  *console.log(`Category "${categoryName}" not found`);*  *return [];*  *}*  *}*  *}*  *// This is a separate function to test our store*  *function main() {*  *// Creates a new store using our EcommerceStore class*  *const store = new EcommerceStore();*  *// Creates two categories to start with*  *store.createCategory("Electronics");*  *store.createCategory("Clothing");*  *// Adds a smartphone to Electronics category*  *store.addItem("Electronics", {*  *name: "Smartphone", // Item name*  *price: 599.99, // Price in dollars*  *stock: 50 // How many we have*  *});*  *// Adds a laptop to Electronics category*  *store.addItem("Electronics", {*  *name: "Laptop",*  *price: 1299.99,*  *stock: 30*  *});*  *// Adds a t-shirt to Clothing category*  *store.addItem("Clothing", {*  *name: "T-Shirt",*  *price: 19.99,*  *stock: 100*  *});*  *// Adds jeans to Clothing category*  *store.addItem("Clothing", {*  *name: "Jeans",*  *price: 49.99,*  *stock: 75*  *});*  *// Shows everything we've added*  *store.displayCatalog();*  *}*  *// Runs our test function*  *main();* |
| --- |

**Lab 2: Write a program to create product list and add to cart feature for the products in an e-commerce store and display cart items. You can write program in any of the language as per your comfort. The design is not required. Database is optional,you can also work using array or any other data structure as per your comfort. Though if you opt to use database,you can.**

-> Solution (in js):

| // This is like starting a new store system  class EcommerceStore {  // This runs when we create a new store  constructor() {  // Creates an empty list (array) to keep all our products  this.products = [];  // Creates an empty object to track what's in the shopping cart  // We'll use product IDs as labels and quantities as values  this.cart = {};  }  // This function adds a new product to our store  addProduct(product) {  // Creates a product with all its details  const productDetails = {  id: Date.now(), // Gives it a unique number based on current time  name: product.name, // Takes the name we provide  price: product.price,// Takes the price we provide  stock: product.stock || 0 // Takes stock amount, uses 0 if not given  };  // Adds this product to our product list  this.products.push(productDetails);  // Tells us the product was added  console.log(`Product "${product.name}" added to store`);  // Returns the product ID so we can use it later  return productDetails.id;  }  // This shows all products we have in the store  displayProducts() {  // Prints a title  console.log("\n=== Available Products ===");  // Checks if we have any products  if (this.products.length === 0) {  console.log("No products available");  return; // Stops here if there’s nothing to show  }  // Goes through each product one by one  this.products.forEach(product => {  // Shows product details with nice formatting  // \n means "start a new line"  console.log(  `ID: ${product.id}` +  `\n- Name: ${product.name}` +  `\n Price: Rs.${product.price}` +  `\n Stock: ${product.stock} units`  );  });  }  // This adds items to our shopping cart  addToCart(productId, quantity = 1) {  // Looks for the product in our list using its ID  const product = this.products.find(p => p.id === productId);    // If we can’t find the product  if (!product) {  console.log(`Product with ID ${productId} not found`);  return false; // Stops and says it failed  }  // Checks if we have enough in stock  if (product.stock < quantity) {  console.log(`Insufficient stock for ${product.name}. Available: ${product.stock}`);  return false; // Stops if we don’t have enough  }  // If item is already in cart, adds to the quantity  if (this.cart[productId]) {  this.cart[productId] += quantity;  } else {  // If it’s not in cart yet, adds it with the quantity  this.cart[productId] = quantity;  }  // Reduces the stock since we’re putting it in the cart  product.stock -= quantity;  // Confirms it worked  console.log(`${quantity} ${product.name}(s) added to cart`);  return true; // Says it succeeded  }  // This removes items from the cart  removeFromCart(productId, quantity = 1) {  // Checks if the item is in the cart  if (!this.cart[productId]) {  console.log("Item not in cart");  return false; // Stops if it’s not there  }  // Finds the product in our product list  const product = this.products.find(p => p.id === productId);  if (!product) {  console.log("Product not found");  return false; // Stops if product doesn’t exist  }  // If we’re removing all or more than we have  if (this.cart[productId] <= quantity) {  // Removes the item completely from cart  delete this.cart[productId];  console.log(`${product.name} removed from cart`);  } else {  // Just reduces the quantity in cart  this.cart[productId] -= quantity;  console.log(`${quantity} ${product.name}(s) removed from cart`);  }  // Puts the stock back since we’re removing from cart  product.stock += quantity;  return true; // Says it worked  }  // This shows what’s in our shopping cart  displayCart() {  // Prints a title  console.log("\n=== Shopping Cart ===");  // Checks if cart is empty  if (Object.keys(this.cart).length === 0) {  console.log("Cart is empty");  return; // Stops if there’s nothing to show  }  // Keeps track of total cost  let total = 0;  // Goes through each item in the cart  for (const [productId, quantity] of Object.entries(this.cart)) {  // Finds the product details  const product = this.products.find(p => p.id === Number(productId));  if (product) {  // Calculates cost for this item (price × quantity)  const subtotal = product.price \* quantity;  // Adds to total  total += subtotal;  // Shows item details  console.log(  `${product.name} (ID: ${productId})` +  `\n- Quantity: ${quantity}` +  `\n- Price: Rs.${product.price}` +  `\n- Subtotal: Rs.${subtotal.toFixed(2)}`  );  }  }  // Shows the total cost with 2 decimal places  console.log(`Total: $${total.toFixed(2)}`);  }  }  // This is like a test to show how our store works  function main() {  // Creates a new store  const store = new EcommerceStore();  // Adds a smartphone and saves its ID  const phoneId = store.addProduct({  name: "Smartphone",  price: 599.99,  stock: 10  });  // Adds a laptop and saves its ID  const laptopId = store.addProduct({  name: "Laptop",  price: 1299.99,  stock: 5  });  // Adds a t-shirt and saves its ID  const shirtId = store.addProduct({  name: "T-Shirt",  price: 19.99,  stock: 20  });  // Shows all products  store.displayProducts();  // Adds some items to the cart  store.addToCart(phoneId, 2); // 2 smartphones  store.addToCart(laptopId, 1); // 1 laptop  store.addToCart(shirtId, 3); // 3 t-shirts  // Shows what’s in the cart  store.displayCart();  // Removes 1 t-shirt from cart  store.removeFromCart(shirtId, 1);  // Shows updated cart  store.displayCart();  // Shows updated product list with new stock amounts  store.displayProducts();  }  // Starts our test  main(); |
| --- |

**Lab 3: Write a program to implement add to cart and checkout feature of e-commerce shopping store and display the ordered list of items. You can write program in any of the language as per your comfort. The design is not required. database is optional,you can also work using array or any other data structure as**

**per your comfort. Though if you opt to use database,you can.**

-> Solution (in js):

| *// This is like setting up our online store*  *class EcommerceStore {*  *// This runs when we start a new store*  *constructor() {*  *// List to hold all products we sell*  *this.products = [];*  *// Cart to hold items customers want to buy (ID: quantity)*  *this.cart = {};*  *// List to keep track of all completed orders*  *this.orders = [];*  *}*  *// Adds a new product to our store*  *addProduct(product) {*  *// Creates product details*  *const productDetails = {*  *id: Date.now(), // Unique ID using current time*  *name: product.name, // Product name*  *price: product.price,// Price of the product*  *stock: product.stock || 0 // How many we have, 0 if not specified*  *};*  *// Adds product to our product list*  *this.products.push(productDetails);*  *// Tells us it worked*  *console.log(`Product "${product.name}" added to store`);*  *// Gives back the ID so we can use it*  *return productDetails.id;*  *}*  *// Shows all products we have*  *displayProducts() {*  *console.log("\n=== Available Products ===");*  *// Checks if we have any products*  *if (this.products.length === 0) {*  *console.log("No products available");*  *return;*  *}*  *// Goes through each product*  *this.products.forEach(product => {*  *// Prints product info nicely*  *console.log(*  *`ID: ${product.id}` +*  *`\n- Name: ${product.name}` +*  *`\n Price: Rs.${product.price}` +*  *`\n Stock: ${product.stock} units`*  *);*  *});*  *}*  *// Puts items in the shopping cart*  *addToCart(productId, quantity = 1) {*  *// Finds the product by its ID*  *const product = this.products.find(p => p.id === productId);*    *// If product doesn’t exist*  *if (!product) {*  *console.log(`Product with ID ${productId} not found`);*  *return false;*  *}*  *// Checks if we have enough in stock*  *if (product.stock < quantity) {*  *console.log(`Not enough ${product.name}. Only ${product.stock} left`);*  *return false;*  *}*  *// Adds to cart or increases quantity if already there*  *if (this.cart[productId]) {*  *this.cart[productId] += quantity;*  *} else {*  *this.cart[productId] = quantity;*  *}*  *// Takes items out of stock*  *product.stock -= quantity;*  *console.log(`${quantity} ${product.name}(s) added to cart`);*  *return true;*  *}*  *// Shows what’s in the cart*  *displayCart() {*  *console.log("\n=== Shopping Cart ===");*  *// Checks if cart is empty*  *if (Object.keys(this.cart).length === 0) {*  *console.log("Cart is empty");*  *return;*  *}*  *let total = 0; // Keeps track of total cost*  *// Goes through each item in cart*  *for (const [productId, quantity] of Object.entries(this.cart)) {*  *const product = this.products.find(p => p.id === Number(productId));*  *if (product) {*  *const subtotal = product.price \* quantity; // Cost for this item*  *total += subtotal; // Adds to total*  *console.log(*  *`${product.name} (ID: ${productId})` +*  *`\n- Quantity: ${quantity}` +*  *`\n- Price: Rs.${product.price}` +*  *`\n- Subtotal: Rs.${subtotal.toFixed(2)}`*  *);*  *}*  *}*  *console.log(`Total: $${total.toFixed(2)}`); // Shows total cost*  *}*  *// Completes the purchase*  *checkout() {*  *// Checks if cart is empty*  *if (Object.keys(this.cart).length === 0) {*  *console.log("Cannot checkout - cart is empty");*  *return false;*  *}*  *// Creates an order from what’s in the cart*  *const order = {*  *orderId: Date.now(), // Unique order number*  *items: {}, // What’s being ordered*  *date: new Date().toLocaleString(), // When it happened*  *total: 0 // Total cost*  *};*  *// Fills order details*  *for (const [productId, quantity] of Object.entries(this.cart)) {*  *const product = this.products.find(p => p.id === Number(productId));*  *if (product) {*  *const subtotal = product.price \* quantity;*  *order.items[productId] = {*  *name: product.name,*  *quantity: quantity,*  *price: product.price,*  *subtotal: subtotal*  *};*  *order.total += subtotal;*  *}*  *}*  *// Adds order to our order history*  *this.orders.push(order);*  *// Clears the cart after checkout*  *this.cart = {};*  *console.log(`Checkout complete! Order ID: ${order.orderId}`);*  *return order.orderId;*  *}*  *// Shows all completed orders*  *displayOrders() {*  *console.log("\n=== Order History ===");*  *// Checks if we have any orders*  *if (this.orders.length === 0) {*  *console.log("No orders yet");*  *return;*  *}*  *// Goes through each order*  *this.orders.forEach(order => {*  *console.log(`\nOrder ID: ${order.orderId}`);*  *console.log(`Date: ${order.date}`);*  *// Shows each item in the order*  *for (const [productId, item] of Object.entries(order.items)) {*  *console.log(*  *`${item.name} (ID: ${productId})` +*  *`\n- Quantity: ${item.quantity}` +*  *`\n- Price: Rs.${item.price}` +*  *`\n- Subtotal: Rs.${item.subtotal.toFixed(2)}`*  *);*  *}*  *console.log(`Order Total: $${order.total.toFixed(2)}`);*  *});*  *}*  *}*  *// This tests our store*  *function main() {*  *// Starts a new store*  *const store = new EcommerceStore();*  *// Adds some products and saves their IDs*  *const phoneId = store.addProduct({*  *name: "Smartphone",*  *price: 599.99,*  *stock: 10*  *});*  *const laptopId = store.addProduct({*  *name: "Laptop",*  *price: 1299.99,*  *stock: 5*  *});*  *const shirtId = store.addProduct({*  *name: "T-Shirt",*  *price: 19.99,*  *stock: 20*  *});*  *// Shows what we have*  *store.displayProducts();*  *// Adds items to cart*  *store.addToCart(phoneId, 2); // 2 phones*  *store.addToCart(laptopId, 1); // 1 laptop*  *store.addToCart(shirtId, 3); // 3 shirts*  *// Shows the cart*  *store.displayCart();*  *// Completes the purchase*  *store.checkout();*  *// Shows the order history*  *store.displayOrders();*  *// Shows updated product list (stock reduced)*  *store.displayProducts();*  *// Shows empty cart after checkout*  *store.displayCart();*  *}*  *// Runs our test*  *main();* |
| --- |

**Lab 4: Write a program to create product list and add to wish list feature for the products in ecommerce store and display the wish list items. You can write program in any of the language as per your comfort. The design is not required. Database is optional, you can also work using array or any other data structure as per your comfort. Though if you opt to use database, you can.**

-> Solution (in js):

| *// This is like opening a new online store*  *class EcommerceStore {*  *// This runs when we start the store*  *constructor() {*  *// List to hold all our products*  *this.products = [];*  *// List to hold items customers want to save for later (wishlist)*  *this.wishlist = [];*  *}*  *// Adds a new product to the store*  *addProduct(product) {*  *// Creates a product with its details*  *const productDetails = {*  *id: Date.now(), // Unique ID based on current time*  *name: product.name, // Product name*  *price: product.price,// Price of the product*  *stock: product.stock || 0 // How many we have, 0 if not given*  *};*  *// Adds the product to our product list*  *this.products.push(productDetails);*  *// Tells us it worked*  *console.log(`Product "${product.name}" added to store`);*  *// Returns the ID so we can use it later*  *return productDetails.id;*  *}*  *// Shows all products in the store*  *displayProducts() {*  *console.log("\n=== Available Products ===");*  *// Checks if we have any products*  *if (this.products.length === 0) {*  *console.log("No products available");*  *return;*  *}*  *// Goes through each product*  *this.products.forEach(product => {*  *// Prints product details nicely*  *console.log(*  *`ID: ${product.id}` +*  *`\n- Name: ${product.name}` +*  *`\n Price: Rs.${product.price}` +*  *`\n Stock: ${product.stock} units`*  *);*  *});*  *}*  *// Adds a product to the wishlist*  *addToWishlist(productId) {*  *// Finds the product by its ID*  *const product = this.products.find(p => p.id === productId);*    *// If product doesn’t exist*  *if (!product) {*  *console.log(`Product with ID ${productId} not found`);*  *return false;*  *}*  *// Checks if it’s already in the wishlist*  *const alreadyInWishlist = this.wishlist.some(item => item.id === productId);*  *if (alreadyInWishlist) {*  *console.log(`${product.name} is already in your wishlist`);*  *return false;*  *}*  *// Adds the product to the wishlist*  *this.wishlist.push(product);*  *console.log(`${product.name} added to wishlist`);*  *return true;*  *}*  *// Removes a product from the wishlist*  *removeFromWishlist(productId) {*  *// Finds the index of the product in the wishlist*  *const index = this.wishlist.findIndex(item => item.id === productId);*    *// If it’s not in the wishlist*  *if (index === -1) {*  *console.log(`Product with ID ${productId} not found in wishlist`);*  *return false;*  *}*  *// Gets the product name before removing*  *const productName = this.wishlist[index].name;*  *// Removes the product from the wishlist*  *this.wishlist.splice(index, 1);*  *console.log(`${productName} removed from wishlist`);*  *return true;*  *}*  *// Shows everything in the wishlist*  *displayWishlist() {*  *console.log("\n=== Your Wishlist ===");*  *// Checks if wishlist is empty*  *if (this.wishlist.length === 0) {*  *console.log("Wishlist is empty");*  *return;*  *}*  *// Goes through each item in the wishlist*  *this.wishlist.forEach(item => {*  *// Shows item details*  *console.log(*  *`${item.name} (ID: ${item.id})` +*  *`\n- Price: Rs.${item.price}` +*  *`\n- Stock: ${item.stock} units`*  *);*  *});*  *}*  *}*  *// This tests our store*  *function main() {*  *// Creates a new store*  *const store = new EcommerceStore();*  *// Adds some products and saves their IDs*  *const phoneId = store.addProduct({*  *name: "Smartphone",*  *price: 599.99,*  *stock: 10*  *});*  *const laptopId = store.addProduct({*  *name: "Laptop",*  *price: 1299.99,*  *stock: 5*  *});*  *const shirtId = store.addProduct({*  *name: "T-Shirt",*  *price: 19.99,*  *stock: 20*  *});*  *// Shows all available products*  *store.displayProducts();*  *// Adds some items to the wishlist*  *store.addToWishlist(phoneId); // Adds smartphone*  *store.addToWishlist(laptopId); // Adds laptop*  *store.addToWishlist(shirtId); // Adds t-shirt*  *// Shows the wishlist*  *store.displayWishlist();*  *// Tries to add something already there*  *store.addToWishlist(phoneId); // Should say it’s already in wishlist*  *// Removes an item*  *store.removeFromWishlist(shirtId);*  *// Shows updated wishlist*  *store.displayWishlist();*  *// Shows products again (stock unchanged)*  *store.displayProducts();*  *}*  *// Runs our test*  *main();* |
| --- |

**Lab 5: Write a program to integrate payment gateway of your choice in an ecommerce page.**

-> Solution (Using ESEWA):

NOTE : Use Below Demo Credential for Esewa  
**eSewa ID:** 9806800001/2/3/4/5

**Password:** Nepal@123 **MPIN:** 1122 **Token:**123456

| *<!DOCTYPE html>*  *<html lang="en">*  *<head>*  *<meta charset="UTF-8">*  *<meta name="viewport" content="width=device-width, initial-scale=1.0">*  *<title>eSewa E-commerce Store</title>*  *<!-- Include CryptoJS libraries for HMAC SHA256 and Base64 encoding -->*  *<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/crypto-js.min.js"></script>*  *<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/hmac-sha256.min.js"></script>*  *<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/enc-base64.min.js"></script>*  *<!-- Basic styling for readability -->*  *<style>*  *body { font-family: Arial, sans-serif; margin: 20px; }*  *.product { margin: 10px 0; }*  *button { padding: 5px 10px; cursor: pointer; }*  *#cart { margin-top: 20px; }*  *</style>*  *</head>*  *<body>*  *<h1>Welcome to Our Store</h1>*  *<!-- Product List -->*  *<h2>Products</h2>*  *<div id="products"></div>*  *<!-- Cart Display -->*  *<h2>Your Cart</h2>*  *<div id="cart"></div>*  *<button onclick="payWithEsewa()">Pay with eSewa</button>*  *<script>*  *// Our store class to manage products and cart*  *class EcommerceStore {*  *// Runs when we create a new store*  *constructor() {*  *// Array to store all products*  *this.products = [];*  *// Object to store cart items (product ID: quantity)*  *this.cart = {};*  *}*  *// Add a new product to the store*  *addProduct(product) {*  *// Create product details*  *const productDetails = {*  *id: Date.now(), // Unique ID using current timestamp*  *name: product.name, // Product name*  *price: product.price,// Price in NPR*  *stock: product.stock || 0 // Stock quantity, default to 0 if not provided*  *};*  *// Add the product to our list*  *this.products.push(productDetails);*  *// Log to console for confirmation*  *console.log(`Product "${product.name}" added to store`);*  *// Return the ID for later use*  *return productDetails.id;*  *}*  *// Add an item to the cart*  *addToCart(productId, quantity = 1) {*  *// Find the product by its ID*  *const product = this.products.find(p => p.id === productId);*  *// If product isn’t found*  *if (!product) {*  *alert(`Product with ID ${productId} not found`);*  *return false;*  *}*  *// Check if enough stock is available*  *if (product.stock < quantity) {*  *alert(`Not enough ${product.name}. Only ${product.stock} left`);*  *return false;*  *}*  *// If product is already in cart, increase quantity*  *if (this.cart[productId]) {*  *this.cart[productId] += quantity;*  *} else {*  *// Otherwise, add it with the quantity*  *this.cart[productId] = quantity;*  *}*  *// Reduce stock since it’s in the cart*  *product.stock -= quantity;*  *console.log(`${quantity} ${product.name}(s) added to cart`);*  *return true;*  *}*  *// Calculate total amount in the cart*  *getCartTotal() {*  *let total = 0;*  *// Loop through cart items*  *for (const [productId, quantity] of Object.entries(this.cart)) {*  *// Find the product*  *const product = this.products.find(p => p.id === Number(productId));*  *if (product) {*  *// Add price \* quantity to total*  *total += product.price \* quantity;*  *}*  *}*  *return total;*  *}*  *// Show products on the page*  *displayProducts() {*  *const productsDiv = document.getElementById('products');*  *productsDiv.innerHTML = ''; // Clear existing content*  *// Loop through products*  *this.products.forEach(product => {*  *// Create a div for each product*  *const div = document.createElement('div');*  *div.className = 'product';*  *// Add product info and button*  *div.innerHTML = `*  *${product.name} - NPR ${product.price} (Stock: ${product.stock})*  *<button onclick="store.addToCart(${product.id}, 1); updateDisplays();">*  *Add to Cart*  *</button>*  *`;*  *productsDiv.appendChild(div);*  *});*  *}*  *// Show cart contents on the page*  *displayCart() {*  *const cartDiv = document.getElementById('cart');*  *cartDiv.innerHTML = ''; // Clear existing content*  *// Check if cart is empty*  *if (Object.keys(this.cart).length === 0) {*  *cartDiv.innerHTML = 'Cart is empty';*  *return;*  *}*  *let total = 0;*  *// Loop through cart items*  *for (const [productId, quantity] of Object.entries(this.cart)) {*  *const product = this.products.find(p => p.id === Number(productId));*  *if (product) {*  *const subtotal = product.price \* quantity;*  *total += subtotal;*  *// Add item details to cart display*  *cartDiv.innerHTML += `*  *${product.name} (ID: ${productId})*  *- Quantity: ${quantity}*  *- Subtotal: NPR ${subtotal.toFixed(2)}<br>*  *`;*  *}*  *}*  *// Show total*  *cartDiv.innerHTML += `<strong>Total: NPR ${total.toFixed(2)}</strong>`;*  *}*  *}*  *// Create a new store instance*  *const store = new EcommerceStore();*  *// eSewa test credentials from reference code and doc*  *const merchantCode = 'EPAYTEST'; // Test merchant code*  *const secretKey = '8gBm/:&EnhH.1/q'; // Test secret key*  *// Add sample products*  *const phoneId = store.addProduct({ name: "Smartphone", price: 25000, stock: 10 });*  *const laptopId = store.addProduct({ name: "Laptop", price: 80000, stock: 5 });*  *const shirtId = store.addProduct({ name: "T-Shirt", price: 1500, stock: 20 });*  *// Function to update product and cart displays*  *function updateDisplays() {*  *store.displayProducts();*  *store.displayCart();*  *}*  *// Initial display of products and cart*  *updateDisplays();*  *// Function to initiate eSewa payment (adapted from reference code)*  *function payWithEsewa() {*  *// Check if cart is empty*  *if (Object.keys(store.cart).length === 0) {*  *alert("Your cart is empty!");*  *return;*  *}*  *// Calculate total amount from cart*  *const totalAmount = store.getCartTotal();*  *// Generate transaction UUID (from reference code)*  *const currentTime = new Date();*  *const formattedTime = currentTime.toISOString().slice(2, 10).replace(/-/g, '') +*  *'-' + currentTime.getHours() + currentTime.getMinutes() + currentTime.getSeconds();*  *const transactionUuid = formattedTime;*  *// eSewa payment parameters*  *const amount = totalAmount; // Base amount*  *const taxAmount = 0; // Tax amount (0 for simplicity)*  *const productServiceCharge = 0; // Service charge (0 for simplicity)*  *const productDeliveryCharge = 0; // Delivery charge (0 for simplicity)*  *const totalAmountWithCharges = amount + taxAmount + productServiceCharge + productDeliveryCharge;*  *const productCode = 'EPAYTEST'; // From reference code*  *// Generate signature (from reference code)*  *const signatureString = `total\_amount=${totalAmountWithCharges},transaction\_uuid=${transactionUuid},product\_code=${productCode}`;*  *const hash = CryptoJS.HmacSHA256(signatureString, secretKey);*  *const signature = CryptoJS.enc.Base64.stringify(hash);*  *// Create form for eSewa submission*  *const form = document.createElement('form');*  *form.method = 'POST';*  *// Use eSewa test URL from reference code*  *form.action = 'https://rc-epay.esewa.com.np/api/epay/main/v2/form';*  *// eSewa form fields (based on reference code)*  *const paymentData = {*  *'amount': amount, // Base amount*  *'tax\_amount': taxAmount, // Tax amount*  *'total\_amount': totalAmountWithCharges, // Total including all charges*  *'transaction\_uuid': transactionUuid, // Unique transaction ID*  *'product\_code': productCode, // Merchant product code*  *'product\_service\_charge': productServiceCharge, // Service charge*  *'product\_delivery\_charge': productDeliveryCharge, // Delivery charge*  *'success\_url': 'https://esewa.com.np', // Success redirect URL*  *'failure\_url': 'https://google.com', // Failure redirect URL*  *'signed\_field\_names': 'total\_amount,transaction\_uuid,product\_code', // Fields to sign*  *'signature': signature // Generated signature*  *};*  *// Add all fields to the form*  *for (const [key, value] of Object.entries(paymentData)) {*  *const input = document.createElement('input');*  *input.type = 'hidden'; // Hidden from user*  *input.name = key; // Field name*  *input.value = value; // Field value*  *form.appendChild(input);*  *}*  *// Add form to page and submit to eSewa*  *document.body.appendChild(form);*  *form.submit();*  *}*  *</script>*  *</body>*  *</html>* |
| --- |