

**Deliverable 2**

**Course: SCD**

**Submitted to: Maam Fatima Gillani**

**Submitted by: Adil Bilal (FL21499)**

**Sana Bushra (FL21526)**

**Section: BSSE-5B**

**Implementation and Testing for EZCart E-Commerce Platform**

Based on the project deliverables provided, I'll outline the implementation and testing plan for the EZCart e-commerce platform, following the design specifications from Deliverable 1 and the diagrams from Deliverable 1-1.

**Implementation Plan for EZCart E-Commerce Platform**

**Adil Bilal (FL21499)**

1. **User Authentication & Login System**
   * Implement login panel with username/password validation
   * Add password requirements (minimum length, no spaces)
   * Handle user session management
2. **Product Management System**
   * Design Product class with attributes (name, price, stock, category)
   * Implement product display logic in the GUI
   * Handle product filtering by category
3. **Shopping Cart Functionality**
   * Implement Cart class to manage added products
   * Develop cart display in the GUI
   * Calculate and display total price
4. **Order Processing**
   * Create Order class to handle order creation
   * Implement basic order processing logic

**Sana Bushra (FL21526)**

1. **Admin & Product Catalog Management**
   * Implement Admin class to manage product inventory
   * Add sample products to the catalog
   * Develop category-based product filtering
2. **Payment System Integration**
   * Design Payment class to simulate payment processing
   * Implement basic payment validation
   * Connect payment to order processing
3. **GUI Development & Layout**
   * Design main application window and panels
   * Implement product display grid with "Add to Cart" buttons
   * Create cart display area with scrollable view
4. **User Experience Improvements**
   * Add welcome message after login
   * Implement feedback messages for cart actions
   * Handle empty cart scenario during checkout

**Collaborative Tasks**

* **Code Review & Integration**
  + Merge individual components into a working system
  + Test end-to-end workflow (login → browse → cart → checkout)
* **Documentation**
  + Add code comments and JavaDoc
  + Prepare final project report
* **Bug Fixing & Final Testing**
  + Identify and resolve any integration issues
  + Perform final system testing before submission

**Code Smells and refactoring in EZCart App**

**1. Long Method (initialize()):**

The initialize() method is over 100 lines long and handles too many responsibilities:

-Creating all UI components

-Setting up layout

-Adding action listeners

-Managing login logic

Refactored Solution:

Break this into smaller methods with single responsibilities:

private void initialize() {

    frame = new JFrame("EZCart E-Commerce Platform");

    frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

    frame.setSize(800, 600);

    frame.setLayout(new BorderLayout());

    setupLoginPanel();

    setupMainPanel();

    setupEventListeners();

    frame.setVisible(true);

}

private void setupLoginPanel() {

    JPanel loginPanel = new JPanel(new GridLayout(3, 2, 5, 5));

    // ... login panel setup code ...

    frame.add(loginPanel, BorderLayout.NORTH);

}

private void setupMainPanel() {

    mainPanel = new JPanel();

    mainPanel.setLayout(new BorderLayout());

    setupWelcomePanel();

    setupProductSelectionPanel();

    setupCartPanel();

    setupCategoryPanel();

    setupCheckoutPanel();

    frame.add(mainPanel, BorderLayout.CENTER);

    mainPanel.setVisible(false);

}

private void setupEventListeners() {

    // Setup all action listeners here

}

**2. Primitive Obsession (Product and Cart handling):**

The code uses primitive types (String, double) to represent product prices and categories, and uses raw ArrayLists for product collections without proper encapsulation.

Refactored Solution:

// New class to handle money operations

public class Money {

    private final double amount;

    private final Currency currency;

    public Money(double amount, Currency currency) {

        this.amount = amount;

        this.currency = currency;

    }

    // Add proper methods for money operations

}

// Product class with better category handling

public class Product {

    private String name;

    private Money price;

    private int stock;

    private ProductCategory category;

    // ...

}

// Enum for categories

public enum ProductCategory {

    ELECTRONICS, CLOTHING, BOOKS;

    @Override

    public String toString() {

        // Custom display names

        return name().charAt(0) + name().substring(1).toLowerCase();

    }

}

**3- Duplicate Code**

 (Product display panels)The product display panel creation in showProductsByCategory() is repetitive. Extract to a method:

Refactored Solution:

private JPanel createProductPanel(Product product) {

    JPanel productPanel = new JPanel(new BorderLayout());

    productPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT\_GRAY));

    JLabel nameLabel = new JLabel(product.getName(), SwingConstants.CENTER);

    JLabel priceLabel = new JLabel("$" + product.getPrice(), SwingConstants.CENTER);

    JButton addButton = new JButton("Add to Cart");

    productPanel.add(nameLabel, BorderLayout.NORTH);

    productPanel.add(priceLabel, BorderLayout.CENTER);

    productPanel.add(addButton, BorderLayout.SOUTH);

    addButton.addActionListener(e -> {

        customer.getCart().addProduct(product);

        updateCart();

        JOptionPane.showMessageDialog(frame, product.getName() + " added to cart!");

    });

    return productPanel;

}

**4- Speculative Generality**

(Admin class)The Admin class is underutilized and could be merged with the product catalog functionality.

Refactored Solution:

class ProductCatalog {

private List<Product> products = new ArrayList<>();

public void addProduct(Product product) {

products.add(product);

}

public List<Product> getAllProducts() {

return Collections.unmodifiableList(products);

}

public List<Product> getProductsByCategory(ProductCategory category) {

return products.stream()

.filter(p -> p.getCategory() == category)

.collect(Collectors.toList());

}

// Additional product management methods could be added here

// like search, update, remove, etc.

}

These refactorings improve:

-Code readability

-Maintainability

-Encapsulation

-Reduced duplication

**Case Test:**

| **Test Id** | **Description** | **Test Data** | **Expected Results** | **Actual Result** | **Class** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| **TC001** | Valid password (8+ characters, no spaces) | Username: "User1", Password: "ValidPass123" | Login successful, main panel visible | **Login successful, main panel visible** | Login | **Pass** |
| **TC002** | Password with less than 8 characters | Username: "User2", Password: "short" | Error: "Password must be at least 8 characters long" | **Error message displayed** | Login | **Pass** |
| **TC003** | Password containing spaces | Username: "User3", Password: "Pass 1234" | Error: "Password cannot contain spaces" | **Error message displayed** | Login | **Pass** |
| **TC004** | Add products to cart and verify total price | Products: [$20, $30, $50] | Total = $100 | **Total = $100** | Cart | **Pass** |
| **TC005** | Filter products by category (Electronics) | 2 Electronics, 1 Clothing added | 2 products displayed | **4 Electronics displayed** *(code preloads 4 Electronics)* | Admin | **Fail** |
| **TC006** | Process order with non-empty cart | Cart: 3 items ($75 total) | Console shows "Order processed with total price: 75.0" | **Console output matches** | Order | **Pass** |
| **TC007** | Process valid payment | Amount: $150 | Payment success = true | **Returns true** | Payment | **Pass** |
| **TC008** | Clear cart after adding items | Add 3 items → clearCart() | Cart size = 0 | **Cart size = 0** | Cart | **Pass** |
| **TC009** | Create product with valid data | Name: "Laptop", Price: $800, Stock: 10 | All fields stored correctly | **Fields match input values** | Product | **Pass** |
| **TC010** | Admin product listing | Add 5 products via Admin | getProducts() returns 5 items | **Returns 5 items** | Admin | **Pass** |

**EZCart Bug Report**

**Bug EC-01**

* **Problem:** Stock quantity doesn't decrease when items are added to cart
* **Where:** Product class, reduceStock() never called
* **Example:** Add "Laptop" → stock remains at 10

**Bug EC-02**

* **Problem:** Same product can be added multiple times as separate items
* **Where:** Cart class, addProduct() method
* **Example:** "T-Shirt" appears twice instead of showing quantity

**Bug EC-03**

* **Problem:** Payment succeeds even for $0 or negative amounts
* **Where:** Payment class, processPayment() method
* **Example:** Empty cart checkout shows "Payment successful"

**Bug EC-04**

* **Problem:** No stock check before adding to cart
* **Where:** showProductsByCategory() method
* **Example:** Can add "Laptop" even if stock is 0

**Code Documentation in EZCartApp.java**

We've used three types of comments throughout our code to improve maintainability and clarity:

**1. Legal Comments**

Used at the file header to document ownership and copyright:

/\*\*

\* EZCart E-Commerce Platform

\*

\* @author Adil Bilal

\* @author Sana Bushra

\* @version 1.0

\* @copyright 2025 EZCart. All rights reserved.

\*/

**2. Amplification Comments**

Used to emphasize important implementation details:

// IMPORTANT: This method should be synchronized in a multi-threaded environment

// to prevent race conditions when multiple users purchase the same product

public void reduceStock(int quantity) {

this.stock -= quantity;

}

**3. TODO Comments**

Used to mark areas needing future work:

// TODO: Replace with actual payment gateway integration

// This is just a simulation - in production, connect to Stripe/PayPal/etc.

public boolean processPayment(double amount) {

return true;

}

These comments help:

* Identify code ownership (Legal)
* Highlight critical sections (Amplification)
* Track unfinished work (TODO)