Group: Calvert Blair, Frank Gomez, Eric Pomerantz, John Mcmenamin

COP4331 002

December 7,2023

Professor: Dr. Ionut Cardei

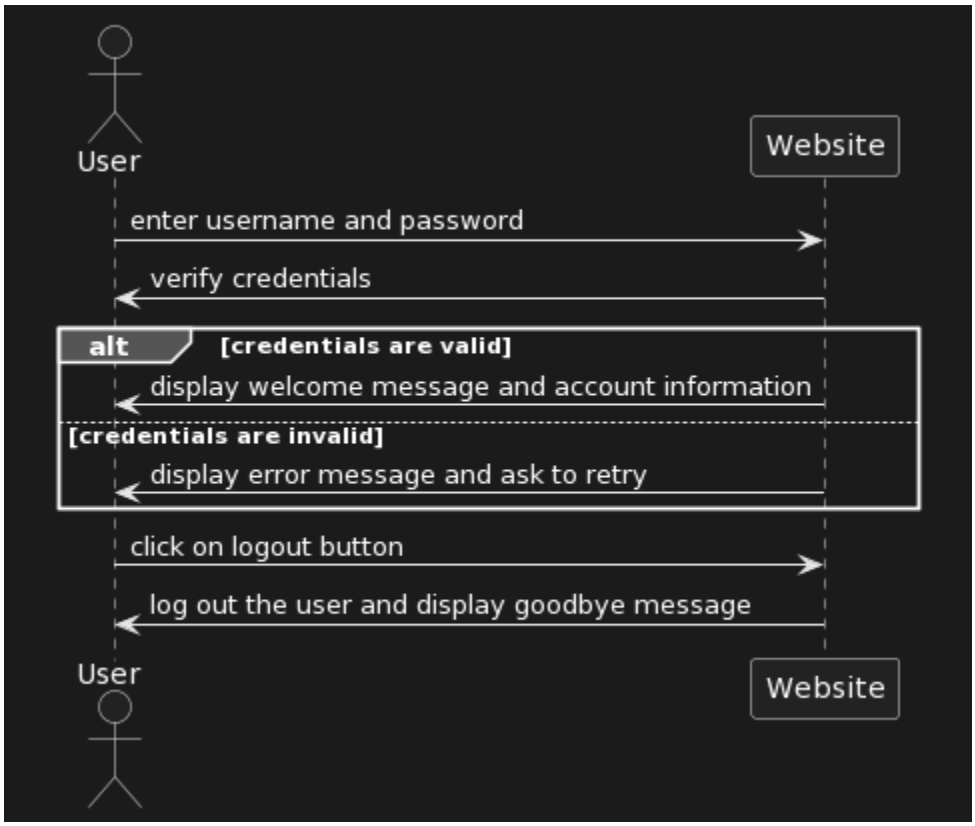
The Shopping Cart: Final Report

UML editor: [PlantUML Web Server]

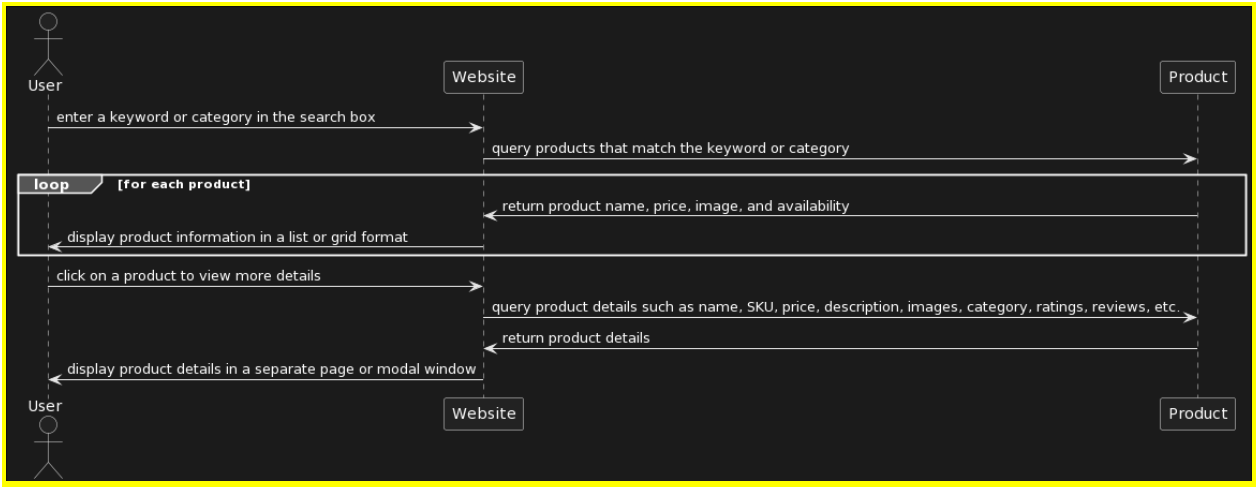
Class Diagrams:

Sequence Diagrams:

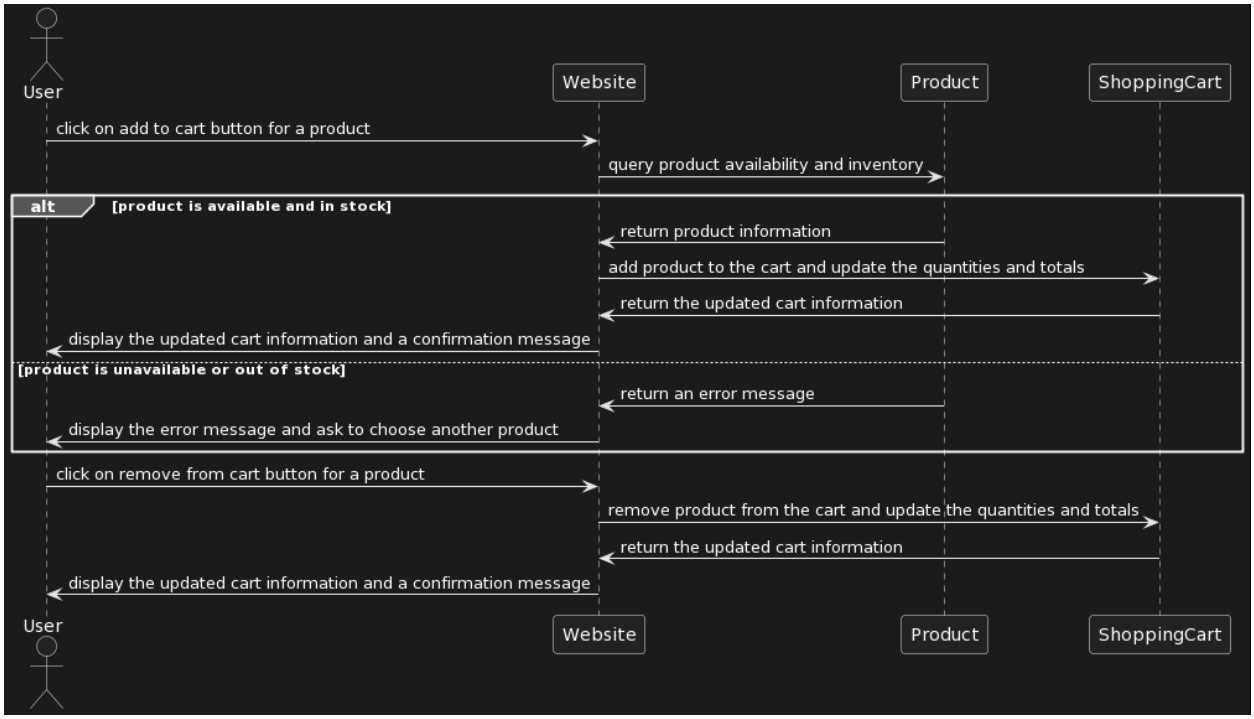
Use Case Scenario #1: User logs in and logs out of the website.



Use Case Scenario #2: User browses and searches for products.

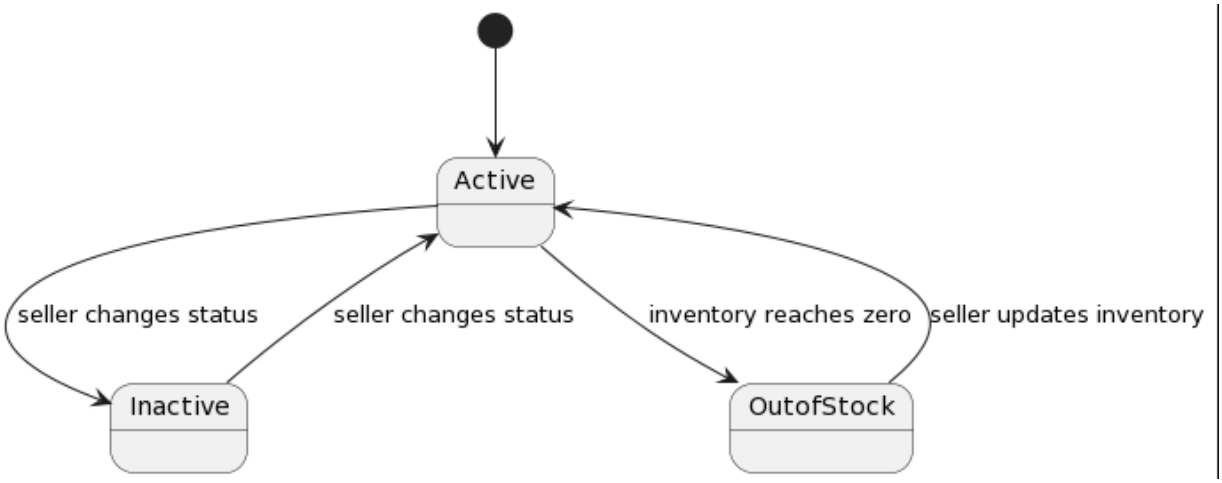


Use Case Scenario #3: User adds and removes items from their shopping cart.

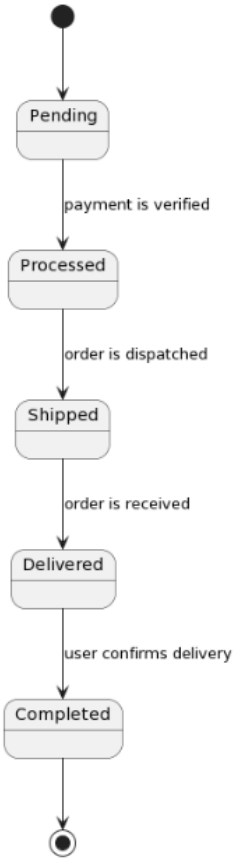


State Diagrams:

State Diagram for Product:



State Diagram for Order:



ShoppingApp.java

```java
package group21.com.fau.shopping_cart_app;


import javax.swing.*;
import java.util.Map;


public class ShoppingApp {


    /**
     * Main method to start the application.
     *
     * @param args Command-line arguments (not used).
     */
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            LoginFrame loginFrame = new LoginFrame();
            loginFrame.setVisible(true);
        });
    }


    /**
     * Calculate the total cost of items in the shopping cart.
     *
     * @return The total cost of items in the shopping cart.
     */
    public static double calculateTotal() {
        double total = 0.0;
```

```
        for (Map.Entry<Product, Integer> entry :
ShoppingCart.getCartItems().entrySet()) {
            total += entry.getKey().getPrice() * entry.getValue();
        }
        return total;
    }
}
```

Inventory.java

```
package group21.com.fau.shopping_cart_app;


import java.util.ArrayList;
import java.util.List;


/**
 * Represents the inventory of products available for purchase.
 */
class Inventory {
    private static final List<Product> products = new ArrayList<>();


    static {
        // Sample
        products.add(new Product(1, "A", 10.0, 20));
        products.add(new Product(2, "B", 15.0, 15));
        products.add(new Product(3, "C", 20.0, 10));
    }
```

```java
    /**
     * Get the list of products in the inventory.
     *
     * @return The list of products.
     */
    public static List<Product> getProducts() {
        return products;
    }

    /**
     * Get a product by its unique ID.
     *
     * @param productId The product ID to search for.
     * @return The product with the specified ID, or null if not found.
     */
    public static Product getProductById(int productId) {
        for (Product product : products) {
            if (product.getProductId() == productId) {
                return product;
            }
        }
        return null;
    }

    /**
     * Update the quantity of a product in the inventory.
     *
```

```
     * @param productId   The product ID to update.
     * @param newQuantity The new quantity to set.
     */
    public static void updateProductQuantity(int productId, int
newQuantity) {
        for (Product product : products) {
            if (product.getProductId() == productId) {
                product.setQuantity(newQuantity);
                return;
            }
        }
    }

    /**
     * Add a new product to the inventory.
     *
     * @param product The product to add.
     */
    public static void addProduct(Product product) {
        products.add(product);
    }
}
```

LoginFrame.java

```
package group21.com.fau.shopping_cart_app;


import javax.swing.*;
```

```java
import java.awt.*;
import java.util.ArrayList;
import java.util.List;


/**
 * Represents the login frame for the application.
 */
class LoginFrame extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;

    public LoginFrame() {
        setTitle("Login");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);
        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridLayout(4, 2));

        panel.add(new JLabel("Username:"));
        usernameField = new JTextField();
        panel.add(usernameField);

        panel.add(new JLabel("Password:"));
        passwordField = new JPasswordField();
        panel.add(passwordField);
```

```
        JButton loginButton = new JButton("Login");
        loginButton.addActionListener(e -> {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            User user = authenticate(username, password);

            if (user != null) {
                openProductBrowseWindow(user);
                dispose();
            } else {
                JOptionPane.showMessageDialog(this, "Invalid username or
password");
            }
        });
        panel.add(loginButton);

        JButton registerButton = new JButton("Register");
        registerButton.addActionListener(e ->
openUserRegistrationWindow());
        panel.add(registerButton);

        add(panel);
    }

    private User authenticate(String username, String password) {
        // Hardcoded user authentication (replace with actual
authentication logic)
        for (User existingUser : users) {
```

```
            if (existingUser.getUsername().equals(username) &&
existingUser.getPassword().equals(password)) {
                return existingUser;
            }
        }
        return null;
    }

    private void openProductBrowseWindow(User user) {
        ProductBrowseFrame productBrowseFrame = new
ProductBrowseFrame(user);
        productBrowseFrame.setVisible(true);
    }

    private void openUserRegistrationWindow() {
        UserRegistrationFrame registrationFrame = new
UserRegistrationFrame();
        registrationFrame.setVisible(true);
    }

    // Sample hardcoded users for simplicity (replace with database
storage)
    static final List<User> users = new ArrayList<>();
}
```

Product.java

```
package group21.com.fau.shopping_cart_app;
```

```java
import java.util.ArrayList;
import java.util.List;

/**
 * Represents a product that can be added to the inventory.
 */
class Product {
    private int productId;
    private String name;
    private double price;
    private int quantity;
    private List<Review> reviews = new ArrayList<>();

    /**
     * Constructor to create a Product object.
     *
     * @param productId The unique ID of the product.
     * @param name      The name of the product.
     * @param price     The price of the product.
     * @param quantity  The quantity of the product in stock.
     */
    public Product(int productId, String name, double price, int quantity)
{
        this.productId = productId;
        this.name = name;
        this.price = price;
```

```
        this.quantity = quantity;
    }

    /**
     * Get the product ID.
     *
     * @return The product ID.
     */
    public int getProductId() {
        return productId;
    }

    /**
     * Get the name of the product.
     *
     * @return The product name.
     */
    public String getName() {
        return name;
    }

    /**
     * Get the price of the product.
     *
     * @return The product price.
     */
    public double getPrice() {
```

```
        return price;
    }

    /**
     * Get the quantity of the product in stock.
     *
     * @return The product quantity.
     */
    public int getQuantity() {
        return quantity;
    }

    /**
     * Set the quantity of the product in stock.
     *
     * @param quantity The new quantity to set.
     */
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    /**
     * Get the list of reviews for the product.
     *
     * @return The list of reviews.
     */
    public List<Review> getReviews() {
```

```
        return reviews;
    }

    /**
     * Add a review for the product.
     *
     * @param reviewer The reviewer's name.
     * @param comment  The review comment.
     */
    public void addReview(String reviewer, String comment) {
        Review review = new Review(reviewer, comment);
        reviews.add(review);
    }
}
```

ProductBrowaseFrame.java

```
class ProductBrowseFrame extends JFrame {
    private User user;
    private JLabel totalLabel;
    private JTable inventoryTable;
    private JTextArea reviewTextArea;

    public ProductBrowseFrame(User user) {
        this.user = user;
        setTitle("Product Browse");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(600, 400);
```

```
        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new BorderLayout());

        inventoryTable = createInventoryTable();
        JScrollPane scrollPane = new JScrollPane(inventoryTable);
        panel.add(scrollPane, BorderLayout.CENTER);

        reviewTextArea = new JTextArea();
        JScrollPane reviewScrollPane = new JScrollPane(reviewTextArea);
        panel.add(reviewScrollPane, BorderLayout.EAST);

        JButton addToCartButton = new JButton("Add to Cart");
        addToCartButton.addActionListener(e -> addToCart());
        panel.add(addToCartButton, BorderLayout.SOUTH);

        totalLabel = new JLabel("Total: $0.0");
        panel.add(totalLabel, BorderLayout.NORTH);

        JButton viewDetailsButton = new JButton("View Details");
        viewDetailsButton.addActionListener(e -> viewProductDetails());
        panel.add(viewDetailsButton, BorderLayout.WEST);

        JButton checkoutButton = new JButton("Checkout");
        checkoutButton.addActionListener(e -> checkout());
        panel.add(checkoutButton, BorderLayout.EAST);
```

```
        // Panel for top buttons
        JPanel topButtonPanel = new JPanel(new
FlowLayout(FlowLayout.LEFT));

        JButton logoutButton = new JButton("Logout");
        logoutButton.addActionListener(e -> logout());
        topButtonPanel.add(logoutButton);

        if (user.getRole() == User.UserRole.SELLER) {
            JButton addProductButton = new JButton("Add New Product");
            addProductButton.addActionListener(e -> addNewProduct());
            topButtonPanel.add(addProductButton);
        }

        // Adding the top button panel to the NORTH region
        panel.add(topButtonPanel, BorderLayout.NORTH);

        add(panel);
    }

    private JTable createInventoryTable() {
        List<Product> products = Inventory.getProducts();
        Object[][] data = new Object[products.size()][4];

        int i = 0;
        for (Product product : products) {
            data[i][0] = product.getProductId();
```

```
            data[i][1] = product.getName();
            data[i][2] = product.getPrice();
            data[i][3] = product.getQuantity();
            i++;
        }

        String[] columnNames = {"ID", "Name", "Price", "Quantity"};

        DefaultTableModel tableModel = new DefaultTableModel(data,
columnNames) {
            @Override
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        };

        return new JTable(tableModel);
    }

    private void addToCart() {
        int selectedRow = inventoryTable.getSelectedRow();

        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a product
to add to the cart");
            return;
        }
```

```
        int productId = (int) inventoryTable.getValueAt(selectedRow, 0);
        String productName = (String) inventoryTable.getValueAt(selectedRow, 1);
        double productPrice = (double) inventoryTable.getValueAt(selectedRow, 2);
        int productQuantity = (int) inventoryTable.getValueAt(selectedRow, 3);

        Product selectedProduct = new Product(productId, productName, productPrice, productQuantity);
        int quantityToAdd = askQuantityToAdd(selectedProduct);

        if (quantityToAdd > 0 && quantityToAdd <= productQuantity) {
            ShoppingCart.addToCart(selectedProduct, quantityToAdd);
            JOptionPane.showMessageDialog(this, quantityToAdd + " " + productName + "(s) added to the cart");
        } else {
            JOptionPane.showMessageDialog(this, "Invalid quantity. Please enter a quantity between 1 and " + productQuantity);
        }
        updateCartTotal();
    }

    private void viewProductDetails() {
        int selectedRow = inventoryTable.getSelectedRow();

        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a product to view details");
```

```
            return;
        }

        int productId = (int) inventoryTable.getValueAt(selectedRow, 0);
        String productName = (String)
inventoryTable.getValueAt(selectedRow, 1);
        double productPrice = (double)
inventoryTable.getValueAt(selectedRow, 2);
        int productQuantity = (int) inventoryTable.getValueAt(selectedRow,
3);

        Product selectedProduct = Inventory
                .getProductById(productId);

        if (selectedProduct != null) {
            JOptionPane.showMessageDialog(this,
getProductDetails(selectedProduct));
            updateReviewsTextArea(selectedProduct);
        } else {
            JOptionPane.showMessageDialog(this, "Invalid product
selection");
        }
        int response = JOptionPane.showConfirmDialog(this,
getProductDetails(selectedProduct) +
                "\n\nWould you like to leave a review?", "Product
Details", JOptionPane.YES_NO_OPTION);

        if (response == JOptionPane.YES_OPTION) {
```

```
            String review = JOptionPane.showInputDialog("Enter your
review:");
            if (review != null && !review.trim().isEmpty()) {
                selectedProduct.addReview(user.getUsername(), review);
            }
        }
    }

    private void updateReviewsTextArea(Product product) {
        StringBuilder reviewsBuilder = new StringBuilder("Seller
Reviews:\n");
        for (Review review : product.getReviews()) {
            reviewsBuilder.append("Reviewer:
").append(review.getReviewer()).append("\n")
                    .append("Comment:
").append(review.getComment()).append("\n\n");
        }
        reviewTextArea.setText(reviewsBuilder.toString());
    }

    private String getProductDetails(Product product) {
        StringBuilder details = new StringBuilder();
        details.append("Product ID:
").append(product.getProductId()).append("\n")
                .append("Name: ").append(product.getName()).append("\n")
                .append("Price:
$").append(product.getPrice()).append("\n")
                .append("Quantity available:
").append(product.getQuantity()).append("\n\n")
```

```
                .append("Reviews:\n");


        for (Review review : product.getReviews()) {
            details.append("Reviewer:
").append(review.getReviewer()).append("\n")
                    .append("Comment:
").append(review.getComment()).append("\n\n");
        }


        return details.toString();
    }


    private int askQuantityToAdd(Product product) {
        String input = JOptionPane.showInputDialog(this, "Enter quantity
for " + product.getName() + ":");
        try {
            return Integer.parseInt(input);
        } catch (NumberFormatException e) {
            return 0;
        }
    }


    private void updateCartTotal() {
        double totalAmount = ShoppingCart.calculateTotal();


        if (totalLabel != null) {
            totalLabel.setText("Total: $" + totalAmount);
        }
```

```
    }

    private void addNewProduct() {
        if (user.getRole() != User.UserRole.SELLER) {
            JOptionPane.showMessageDialog(this, "Only sellers can add products.");
            return;
        }

        // Example of gathering product details (You should implement a more robust method)
        String name = JOptionPane.showInputDialog("Enter product name:");
        double price = Double.parseDouble(JOptionPane.showInputDialog("Enter product price:"));
        int quantity = Integer.parseInt(JOptionPane.showInputDialog("Enter product quantity:"));

        // Generate a new product ID
        int newProductId = Inventory.getProducts().size() + 1;

        Inventory.addProduct(new Product(newProductId, name, price, quantity));
        JOptionPane.showMessageDialog(this, "Product added successfully.");

        refreshInventoryTable();
    }

    private void refreshInventoryTable() {
```

```
        DefaultTableModel model = (DefaultTableModel)
inventoryTable.getModel();

        model.setRowCount(0); // Clear existing data


        for (Product product : Inventory.getProducts()) {

            model.addRow(new Object[]{product.getProductId(),
product.getName(), product.getPrice(), product.getQuantity()});

        }

    }


    private void checkout() {

        double totalAmount = ShoppingCart.calculateTotal();


        if (totalAmount > 0) {

            for (Map.Entry<Product, Integer> entry :
ShoppingCart.getCartItems().entrySet()) {

                Product product = entry.getKey();

                int purchasedQuantity = entry.getValue();

                int newQuantity = product.getQuantity() -
purchasedQuantity;

                Inventory.updateProductQuantity(product.getProductId(),
newQuantity);

            }


            ShoppingCart.getCartItems().clear();

            updateCartTotal();

            JOptionPane.showMessageDialog(this, "Checkout successful!
Total amount: $" + totalAmount);

        } else {
```

```
            JOptionPane.showMessageDialog(this, "Your cart is empty. Add
items before checking out.");

        }


        refreshInventoryTable();

    }


    private void logout() {

        this.dispose(); // Close the current window

        LoginFrame loginFrame = new LoginFrame();

        loginFrame.setVisible(true); // Open the login window

    }

}
```

Review.java

```
package group21.com.fau.shopping_cart_app;


/**
 * Represents a review for a product.
 */
class Review {

    private String reviewer;

    private String comment;


    /**
     * Constructor to create a Review object.
     *
```

```
     * @param reviewer The reviewer's name.
     * @param comment  The review comment.
     */
    public Review(String reviewer, String comment) {
        this.reviewer = reviewer;
        this.comment = comment;
    }

    /**
     * Get the reviewer's name.
     *
     * @return The reviewer's name.
     */
    public String getReviewer() {
        return reviewer;
    }

    /**
     * Get the review comment.
     *
     * @return The review comment.
     */
    public String getComment() {
        return comment;
    }
}
```

ShoppingCart.java

```java
package group21.com.fau.shopping_cart_app;


import java.util.HashMap;

import java.util.Map;


/**

 * Represents the shopping cart of a user.

 */

class ShoppingCart {

    private static final Map<Product, Integer> cartItems = new
HashMap<>();


    /**

     * Get the items in the shopping cart.

     *

     * @return A map of products to quantities in the cart.

     */

    public static Map<Product, Integer> getCartItems() {

        return cartItems;

    }


    /**

     * Add a product to the shopping cart.

     *

     * @param product  The product to add.

     * @param quantity The quantity to add.
```

```java
     */
    public static void addToCart(Product product, int quantity) {
        if (cartItems.containsKey(product)) {
            int currentQuantity = cartItems.get(product);
            cartItems.put(product, currentQuantity + quantity);
        } else {
            cartItems.put(product, quantity);
        }
    }

    /**
     * Remove a product from the shopping cart.
     *
     * @param product The product to remove.
     */
    public static void removeFromCart(Product product) {
        cartItems.remove(product);
    }

    /**
     * Calculate the total cost of items in the shopping cart.
     *
     * @return The total cost of items in the shopping cart.
     */
    public static double calculateTotal() {
        double total = 0.0;
        for (Map.Entry<Product, Integer> entry : cartItems.entrySet()) {
```

```
            total += entry.getKey().getPrice() * entry.getValue();
        }
        return total;
    }
}
```

User.java

```
package group21.com.fau.shopping_cart_app;


/**
 * Represents a user in the system.
 */
class User {
    private String username;
    private String password;

    private UserRole role; // New attribute for user role

    /**
     * Enum representing user roles (BUYER or SELLER).
     */
    public enum UserRole {
        BUYER, SELLER
    }

    /**
     * Constructor to create a User object.
```

```
     *
     * @param username The username of the user.
     * @param password The password of the user.
     * @param role     The role of the user (BUYER or SELLER).
     */
    public User(String username, String password, UserRole role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    /**
     * Get the username of the user.
     *
     * @return The username.
     */
    public String getUsername() {
        return username;
    }

    /**
     * Get the password of the user.
     *
     * @return The password.
     */
    public String getPassword() {
        return password;
```

```
    }

    /**
     * Get the role of the user.
     *
     * @return The user role (BUYER or SELLER).
     */
    public UserRole getRole() {
        return role;
    }
}
```

UserRegistrationFrame.java

```
package group21.com.fau.shopping_cart_app;

import javax.swing.*;
import java.awt.*;

public class UserRegistrationFrame extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;

    private JComboBox<User.UserRole> roleComboBox;

    public UserRegistrationFrame() {
        setTitle("User Registration");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
        setSize(300, 150);
        setLocationRelativeTo(null);

        JPanel panel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();

        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets(4, 4, 4, 4); // top, left, bottom, right
padding

        // Username label and text field
        gbc.gridx = 0; // column
        gbc.gridy = 0; // row
        panel.add(new JLabel("Username:"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 0;
        usernameField = new JTextField(10);
        panel.add(usernameField, gbc);

        // Password label and text field
        gbc.gridx = 0;
        gbc.gridy = 1;
        panel.add(new JLabel("Password:"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 1;
```

```
        passwordField = new JPasswordField(10);
        panel.add(passwordField, gbc);

        // Role label and combo box
        gbc.gridx = 0;
        gbc.gridy = 2;
        panel.add(new JLabel("Role:"), gbc);

        gbc.gridx = 1;
        gbc.gridy = 2;
        roleComboBox = new JComboBox<>(User.UserRole.values());
        panel.add(roleComboBox, gbc);

        // Register button
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.gridwidth = 2; // Span across two columns
        JButton registerButton = new JButton("Register");
        registerButton.addActionListener(e -> {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            if (registerUser(username, password)) {
                JOptionPane.showMessageDialog(this, "Account created
successfully");
                dispose();
            } else {
                JOptionPane.showMessageDialog(this, "Username already
exists. Please choose another.");
```

```
            }
        });
        panel.add(registerButton, gbc);

        add(panel);
    }

    private boolean registerUser(String username, String password) {
        for (User existingUser : LoginFrame.users) {
            if (existingUser.getUsername().equals(username)) {
                return false;
            }
        }

        User.UserRole selectedRole = (User.UserRole)
roleComboBox.getSelectedItem();
        LoginFrame.users.add(new User(username, password, selectedRole));
// Include role in user creation
        return true;
    }
}
```