SHOPPING PLATFORM - SPRING

# TEAM MEMBERS:

|  |  |
| --- | --- |
| **1.** J. Abirami | 3122225001004 |
| **2.** Aditya Jyosyula | 3122225001006 |
| **3.** S. Aruna Devi | 3122225001015 |

**PROBLEM STATEMENT :**

The retail industry has grown a lot when we consider online shopping. As a result, there is a growing demand for user-friendly online shopping platforms. The aim of this project is to develop a Java-based online shopping application to provide customers with a comfortable and affordable shopping experience.

# MOTIVATION FOR THE PROBLEM :

Nowadays there are lots of online shopping platforms available. If we analyze the design of those platforms ,the inferences are that

* We can't store all information about the customers, retailers and the products in separate variables
* We can’t restrict the access of some information from being accessed from everywhere in the program which causes a lot of errors.

From these we can conclude that Object oriented programming is the best suited for this.

# SCOPE:

Creating an online shopping platform using Java involves several key aspects and functionalities. Scope for such a project is given below:

**Backend Development:**

* Framework: Using java features like polymorphism, inheritance etc., to develop the code for online shopping platform.
* Database: Use text files to store, retrieve data

**Frontend Development:**

* Framework/Library: Used swing and awt libraries in java to develop the front end of the project.

**User Management:**

* Authentication: Implement user authentication and authorization using Spring Security or equivalent libraries.
* User Profiles: Enable users to manage their profiles, addresses, and payment details.

**Product Management:**

* Product Catalog: Create, update, and display products with details such as name, description, price, images, and categories.

**Wish list:**

* Cart Management: Allow users to add/remove items, update quantities, and save items for later.
* Checkout Process: Implement a secure and intuitive checkout process with payment integration.

**Payment Integration:**

* Payment Gateways: Integrate with popular payment gateways like PayPal, Stripe, or others using their Java SDKs.
* Security: Ensure secure payment transactions with encryption and compliance with PCI-DSS standards.

**Testing and Quality Assurance:**

* Unit Testing: Write unit tests for backend services using JUnit or TestNG.
* Integration Testing: Ensure smooth interaction between various modules and components.
* Security Testing: Perform security audits to identify and fix vulnerabilities.

By focusing on these aspects, a Java-based online shopping platform can offer a secure, user-friendly, and efficient shopping experience.

# LIMITATIONS:

* Can handle only a single customer or retailer at a time.
* Stocks are not immediately reduced after the product is bought.
* The company name,product name and brand name need not be filled by the customer yet it is shown to the customer during registration.
* Order management of customers is not followed i.e. The records of the history of customers and their details of their purchase is not maintained properly

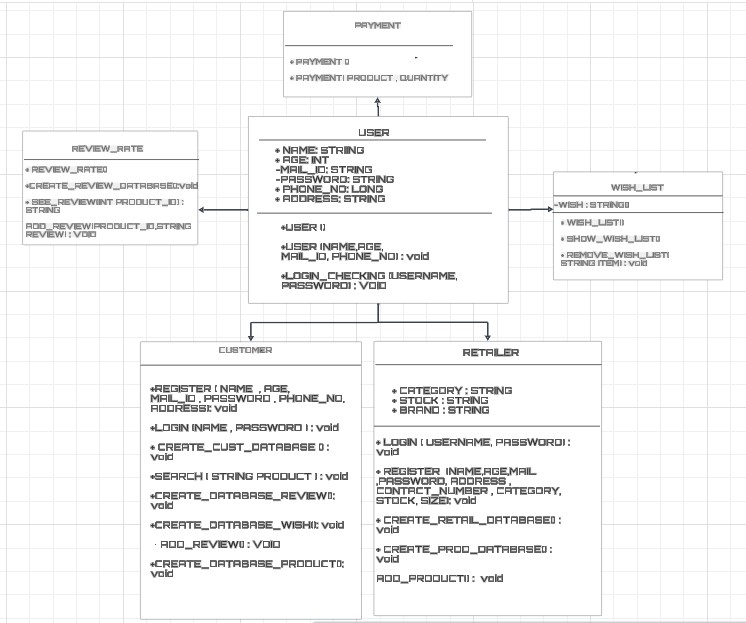
# OBJECT ORIENTED FEATURES USED:

* Inheritance :
  + The concept of inheritance has been used a lot of times.
  + Inheritance increases the efficiency of the code and helps in code reusability.
  + In this project we have used inheritance in the user type.
  + User is the parent or the base class which contains details about all the basic information which are common for customer and retailer.
  + Customer and retailer are two subclasses or derived classes of User.
  + Customers can view the products and add them to their wishlist.
  + From the wishlist they can proceed to pay.
  + For retailers, a method for adding products is included.
* Polymorphism:
* Here function polymorphism is used in several places.
* This is done to incorporate the different functionalities and the concepts involved in the project.
* For example, User is used to get the
* Exception handling:
* Certain in built exceptions like IOException,ArrayIndexOutOfBoundsException etc., are used.
* Used concepts of exceptions for handling exceptions like: the given user id does not exist

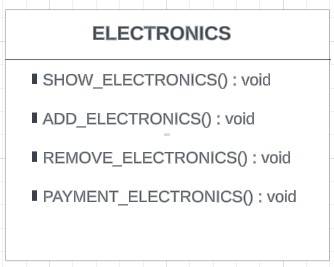
product is not present in the given etc.,

* Collections:
  + Collections are used to store large amounts of similar data.
  + In this project, collections are used to store the details of the customer and the retailer from the file.
  + Everytime when the program is run, the details of the registered customer and the retailers are stored as an object of the corresponding type.
  + Then all the customer objects are stored into TreeMap.
  + Similarly all the retailer objects are stored into TreeMap.

**Class Diagram**





# MODULE SPLIT-UP:

**Classes:**

1. **User:**

FIELDS:

Username,password,email,age,address,mobile METHODS:

* 1. **public User(String username, String password, String email, int age, String address, String mobile):**

It is a constructor for the class User.

* 1. **public String getUsername():**

This method returns username.

* 1. **public String getPassword():**

This method returns password.

1. **CUSTOMER:**

FIELDS:

username, password, email, age, address, mobile METHODS:

* 1. public Customer(String username, String password, String email, int age, String address, String mobile)

It is a constructor to get the required details for registration.

* 1. public String getPassword()

This method returns password.

# RETAILER:

FIELDS:

companyName,productCategory,numProducts,productCounter METHODS:

* 1. **public Retailer(String username, String password, String email, int age, String address, String mobile,String companyName, String productCategory, int numProducts**

It is a constructor for the class Retailer.

* 1. **public String generateProductID(String category):**

This method returns a product ID.

* 1. **private int generateRandomNumbers():**

This method is used to generate random numbers.

* 1. **private boolean isProductIDExists(String candidateID):**

This method returns true if the product ID already exists. Else it returns false.

* 1. **private String getProductDetailsFileName( String productCategory)**

This method returns a string specifying the category of the product which will be used as the file name to store information of the products of that category.

* 1. **public void addProductToCategory(Product product):**

This method adds the product details to its respective category.

* 1. **public int getNumProducts()**

This method returns the number of products of the retailer.

* 1. **public String getCompanyName():**

This method returns company name of retailer.

* 1. **public String getProductCategory():**

This method returns product category of retailer.

1. **Products:**

FIELDS:

prodId,productName,brand,price,stock,imagepath

* 1. public Product(String prodId, String productName, String brand, double price, int stock, String imagePath)

This method is used to obtain required information

* 1. public String getProdId()

This method returns prodId.

* 1. public String getProductName()

This method returns productName.

* 1. public String getBrand()

This method returns brand.

* 1. public double getPrice()

This method returns price.

* 1. public int getStock()

This method returns stock.

* 1. public String getImagePath()

This method returns image path.

* 1. public String getCategory()

This method returns category.

1. **ProductDetailsViewerc:**

FIELDS:

frame,productLabel,infoTextArea,products,currentProductIndex,reviewsFilePath,wishlistFile Path

METHODS:

* 1. **ProductDetailsViewerc(String):**

Constructor of class ProductDetailsViewer

* 1. **void toggleProduct():**

Toggles between different products.

* 1. **void updateProductInfo():**

Can update the product information.

* 1. **void loadProductDetailsFromFile:**

Loads all the product from the file.

* 1. **loadReviewsFromFile(String):**

Loads all reviews from the file.

* 1. **void addReview():**

Adds review to the product.

* 1. **void addToWishlist():**

Adds product to the wishlist of the customer.

* 1. **void showWishlist():**

Displays the wishlist of the customer.

* 1. **void performPayment(String, JPanel):**

Proceeds to the payment and completes the purchase of the product.

* 1. **void updateStockAndClearWishlist(String, JPanel):**

Once the customer has purchased, it updates the stock of the product and clears the product from the wishlist.

* 1. **void decreaseStock():**

Decreases the stock of the product from the file.

* 1. **double calculateTotalPrice(JPanel):**

Returns the total cost of the purchased products.

* 1. **void displayPurchasedProducts(String, JPanel):**

Displays all the purchased products.

* 1. **void clearWishlist(String, JPanel):**

Clears the wishlist.

* 1. **void removeFromWishlist(String, String, double, JFrame):**

Removes the selected product from the wishlist.

* 1. **void showMessage(String):**

Shows the messages.

* 1. **void show():**

Makes the frame visible.

* 1. **ImageIcon getScaledImageIcon(String, int, int):**

Scales the given image to the specified height and width and returns an Imageicon.

Similar to ProductDetailsViewerc , classes such as ProductDetailsViewerf, ProductDetailsViewere, ProductDetailsViewerh have the same methods but they operate on the products of their respective categories.

**IMPLEMENTATION SPECIFICS:**

This is the login application which executes initially when the code begins.

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

**import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*;**

**import java.util.TreeMap;**

**public class LoginApp extends JFrame {**

**private JTextField loginUsernameField, loginPasswordField;**

**private JRadioButton loginCustomerRadioButton, loginRetailerRadioButton; private JPanel loginPanel, registerPanel;**

**private JTextField registerUsernameField, registerEmailField, registerAgeField, registerAddressField,**

**registerMobileField,**

**registerCompanyNameField, registerProductCategoryField,**

**registerNumProductsField;**

**private JPasswordField registerPasswordField;**

**private JRadioButton registerCustomerRadioButton, registerRetailerRadioButton;**

**private static final String CUSTOMER\_FILE\_PATH = "customer\_file.txt"; private static final String RETAILER\_FILE\_PATH = "retailer\_file.txt";**

**private TreeMap<String, User> userDatabase = new TreeMap<>();**

**public LoginApp() { setTitle("Login/Register"); setSize(400, 300);**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); getContentPane().setBackground(new Color(240, 240, 240));**

**loadUsersFromFile(CUSTOMER\_FILE\_PATH, "customer"); loadUsersFromFile(RETAILER\_FILE\_PATH, "retailer");**

**JButton loginButton = new JButton("Login"); JButton registerButton = new JButton("Register");**

**loginButton.setBackground(new Color(46, 139, 87)); // Green color loginButton.setForeground(Color.WHITE); loginButton.setFont(new Font("Arial", Font.BOLD, 14));**

**registerButton.setBackground(new Color(0, 102, 204)); // Blue color registerButton.setForeground(Color.WHITE); registerButton.setFont(new Font("Arial", Font.BOLD, 14));**

**loginButton.addActionListener(new ActionListener() { @Override**

**public void actionPerformed(ActionEvent e) { showLoginPanel();**

**login();**

**}**

**});**

**registerButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) { showRegisterPanel();**

**register();**

**}**

**});**

**loginPanel = createLoginPanel(); registerPanel = createRegisterPanel();**

**JPanel buttonPanel = new JPanel(); buttonPanel.setBackground(new Color(240, 240, 240)); buttonPanel.add(loginButton); buttonPanel.add(registerButton);**

**add(buttonPanel, BorderLayout.NORTH); add(loginPanel, BorderLayout.CENTER);**

**pack(); setLocationRelativeTo(null); setVisible(true);**

**}**

**private JPanel createLoginPanel() { JPanel panel = new JPanel(); panel.setLayout(new GridLayout(4, 2));**

**panel.setBackground(new Color(240, 240, 240));**

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

**panel.add(new JLabel("Password:")); loginPasswordField = new JPasswordField(); panel.add(loginPasswordField);**

**panel.add(new JLabel("User Type:")); loginCustomerRadioButton = new JRadioButton("Customer");**

**loginRetailerRadioButton = new JRadioButton("Retailer");**

**ButtonGroup loginTypeGroup = new ButtonGroup(); loginTypeGroup.add(loginCustomerRadioButton); loginTypeGroup.add(loginRetailerRadioButton);**

**panel.add(loginCustomerRadioButton); panel.add(loginRetailerRadioButton);**

**return panel;**

**}**

**private JPanel createRegisterPanel() { JPanel panel = new JPanel(); panel.setLayout(new GridLayout(11, 2));**

**panel.setBackground(new Color(240, 240, 240));**

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

**panel.add(new JLabel("Password:")); registerPasswordField = new JPasswordField(); panel.add(registerPasswordField);**

**panel.add(new JLabel("Email:")); registerEmailField = new JTextField(); panel.add(registerEmailField);**

**panel.add(new JLabel("Age:")); registerAgeField = new JTextField(); panel.add(registerAgeField);**

**panel.add(new JLabel("Address:")); registerAddressField = new JTextField(); panel.add(registerAddressField);**

**panel.add(new JLabel("Mobile:"));**

**registerMobileField = new JTextField(); panel.add(registerMobileField);**

**panel.add(new JLabel("Company Name:")); registerCompanyNameField = new JTextField(); panel.add(registerCompanyNameField);**

**panel.add(new JLabel("Product Category:")); registerProductCategoryField = new JTextField(); panel.add(registerProductCategoryField);**

**panel.add(new JLabel("Number of Products:")); registerNumProductsField = new JTextField(); panel.add(registerNumProductsField);**

**panel.add(new JLabel("User Type:")); registerCustomerRadioButton = new JRadioButton("Customer"); registerRetailerRadioButton = new JRadioButton("Retailer");**

**ButtonGroup registerTypeGroup = new ButtonGroup(); registerTypeGroup.add(registerCustomerRadioButton); registerTypeGroup.add(registerRetailerRadioButton);**

**panel.add(registerCustomerRadioButton); panel.add(registerRetailerRadioButton);**

**return panel;**

**}**

**private void showLoginPanel() { remove(registerPanel);**

**add(loginPanel, BorderLayout.CENTER); revalidate();**

**repaint();**

**}**

**private void showRegisterPanel() { remove(loginPanel);**

**add(registerPanel, BorderLayout.CENTER); revalidate();**

**repaint();**

**}**

**private void login() {**

**String username = loginUsernameField.getText();**

**String password = new String(loginPasswordField.getText());**

**if (userDatabase.containsKey(username)) { User user = userDatabase.get(username);**

**if (user.getPassword().equals(password) && user instanceof Customer) { SwingUtilities.invokeLater(() -> {**

**showMessage("Login successful as Customer"); openHomePage();**

**});**

**} else if (user.getPassword().equals(password) && user instanceof Retailer) { SwingUtilities.invokeLater(() -> {**

**showMessage("Login successful as Retailer"); openRetailerFrame((Retailer) user);**

**});**

**} else {**

**SwingUtilities.invokeLater(() -> showMessage("Invalid password"));**

**}**

**} else {**

**SwingUtilities.invokeLater(() -> showMessage("User not found. Please register."));**

**}**

**}**

**private void openHomePage() { SwingUtilities.invokeLater(() -> new HomePage()); this.dispose();**

**}**

**private void register() {**

**String username = registerUsernameField.getText();**

**String password = new String(registerPasswordField.getPassword());**

**String email = registerEmailField.getText();**

**int age = Integer.parseInt(registerAgeField.getText()); String address = registerAddressField.getText(); String mobile = registerMobileField.getText();**

**if (registerRetailerRadioButton.isSelected()) {**

**String companyName = registerCompanyNameField.getText(); String productCategory = registerProductCategoryField.getText();**

**int numProducts = Integer.parseInt(registerNumProductsField.getText());**

**registerRetailer(username, password, email, age, address, mobile, companyName, productCategory, numProducts);**

**showMessage("Registration successful as Retailer"); showLoginPanel();**

**} else if (registerCustomerRadioButton.isSelected()) { registerCustomer(username, password, email, age, address, mobile); showMessage("Registration successful as Customer"); showLoginPanel();**

**} else {**

**showMessage("Please select user type for registration.");**

**}**

**}**

**private void showMessage(String message) { JOptionPane.showMessageDialog(this, message);**

**}**

**private void registerRetailer(String username, String password, String email, int age, String address, String mobile,**

**String companyName, String productCategory, int numProducts) { Retailer retailer = new Retailer(username, password, email, age, address, mobile,**

**companyName, productCategory, numProducts);**

**userDatabase.put(username, retailer);**

**saveRetailerToFile(RETAILER\_FILE\_PATH, username, age, email, password, mobile, address,**

**companyName, productCategory, String.valueOf(numProducts));**

**}**



**private void registerCustomer(String username, String password, String email, int age, String address,**

**String mobile) {**

**Customer customer = new Customer(username, password, email, age, address, mobile);**

**userDatabase.put(username, customer); saveCustomerToFile(CUSTOMER\_FILE\_PATH, username, age, email, password,**

**mobile, address);**

**}**

**private void openRetailerFrame(Retailer retailer) {**

**JFrame retailerFrame = new JFrame("Retailer Dashboard"); retailerFrame.setSize(400, 300);**

**JButton addProductButton = new JButton("Add Product"); addProductButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) { addProduct(retailer);**

**}**

**});**

**JPanel retailerPanel = new JPanel(); retailerPanel.setLayout(new GridLayout(1, 1)); retailerPanel.add(addProductButton);**

**retailerFrame.add(retailerPanel); retailerFrame.setLocationRelativeTo(null); retailerFrame.setVisible(true);**

**}**

**private void addProduct(Retailer retailer) {**

**String category = JOptionPane.showInputDialog("Enter category (Furniture, Clothing, Electronics, Footwear):");**

**String prodId = retailer.generateProductID(category);**

**String productName = JOptionPane.showInputDialog("Enter product name:");**

**String brand = JOptionPane.showInputDialog("Enter brand:");**

**double price = Double.parseDouble(JOptionPane.showInputDialog("Enter price:")); int stock = Integer.parseInt(JOptionPane.showInputDialog("Enter stock:"));**

**String imagePath = JOptionPane.showInputDialog("Enter image path:");**

**Product product = new Product(prodId, productName, brand, price, stock, imagePath); retailer.addProductToCategory(product);**

**showMessage("Product added successfully!");**

**}**

**private void loadUsersFromFile(String filePath, String userType) { try {**

**File file = new File(filePath); if (file.exists()) {**

**BufferedReader reader = new BufferedReader(new FileReader(file)); String line;**

**while ((line = reader.readLine()) != null) { String[] userData = line.split(" "); String username = userData[0];**

**int age = Integer.parseInt(userData[1]); String email = userData[2];**

**String password = userData[3]; String mobile = userData[4]; String address = userData[5];**

**if ("customer".equals(userType)) {**

**Customer customer = new Customer(username, password, email, age, address, mobile);**

**userDatabase.put(username, customer);**

**} else if ("retailer".equals(userType)) { String companyName = userData[6]; String productCategory = userData[7];**

**int numProducts = Integer.parseInt(userData[8]);**

**Retailer retailer = new Retailer(username, password, email, age, address, mobile, companyName,**

**productCategory, numProducts);**

**userDatabase.put(username, retailer);**

**}**

**}**

**reader.close();**

**}**

**} catch (IOException e) { e.printStackTrace();**

**}**

**}**

**private void saveCustomerToFile(String filePath, String username, int age, String email, String password, String mobile, String address) {**

**try {**

**FileWriter fileWriter = new FileWriter(filePath, true); StringBuilder userDataLine = new StringBuilder();**

**userDataLine.append(username).append(" "); userDataLine.append(age).append(" "); userDataLine.append(email).append(" "); userDataLine.append(password).append(" "); userDataLine.append(mobile).append(" "); userDataLine.append(address).append(" ");**

**fileWriter.write(userDataLine.toString().trim() + "\n"); fileWriter.close();**

**} catch (IOException e) { e.printStackTrace();**

**}**

**}**

**private void saveRetailerToFile(String filePath, String username, int age, String email, String password, String mobile, String address, String companyName, String productCategory, String numProducts) {**

**try {**

**FileWriter fileWriter = new FileWriter(filePath, true); StringBuilder userDataLine = new StringBuilder();**

**userDataLine.append(username).append(" ");**

**userDataLine.append(age).append(" "); userDataLine.append(email).append(" "); userDataLine.append(password).append(" "); userDataLine.append(mobile).append(" "); userDataLine.append(address).append(" "); userDataLine.append(companyName).append(" "); userDataLine.append(productCategory).append(" "); userDataLine.append(numProducts).append(" ");**

**fileWriter.write(userDataLine.toString().trim() + "\n"); fileWriter.close();**

**} catch (IOException e) { e.printStackTrace();**

**}**

**}**

**public static void main(String[] args) { SwingUtilities.invokeLater(new Runnable() {**

**public void run() { new LoginApp();**

**}**

**});**

**}**

**static class User {**

**private String username; private String password; private String email; private int age;**

**private String address; private String mobile;**

**public User(String username, String password, String email, int age, String address, String mobile) {**

**this.username = username; this.password = password; this.email = email;**

**this.age = age;**

**this.address = address; this.mobile = mobile;**

**}**

**public String getUsername() { return username;**

**}**

**public String getPassword() { return password;**

**}**

**}**

**static class Customer extends User {**

**public Customer(String username, String password, String email, int age, String address, String mobile) {**

**super(username, password, email, age, address, mobile);**

**}**

**public String getPassword() { return super.getPassword();**

**}**

**}**

**static class Retailer extends User { private String companyName; private String productCategory; private int numProducts;**

**private int productCounter = 1;**

**public Retailer(String username, String password, String email, int age, String address, String mobile,**

**String companyName, String productCategory, int numProducts) { super(username, password, email, age, address, mobile); this.companyName = companyName;**

**this.productCategory = productCategory; this.numProducts = numProducts;**

**}**



**public String generateProductID(String category) { String candidateID;**

**do {**

**candidateID = category.substring(0, 1) + generateRandomNumbers();**

**} while (isProductIDExists(candidateID));**

**return candidateID;**

**}**

**private int generateRandomNumbers() {**

**// Generate four random numbers**

**return 1000 + (int) (Math.random() \* 9000);**

**}**

**private boolean isProductIDExists(String candidateID) {**

**// Check if the product ID already exists in the file**

**String fileName = getProductDetailsFileName(candidateID);**

**try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) { String line;**

**while ((line = reader.readLine()) != null) { String[] parts = line.split(" ");**

**if (parts.length > 0 && parts[0].equals(candidateID)) { return true; // ID exists**

**}**

**}**

**} catch (IOException e) { e.printStackTrace();**

**}**

**return false; // ID doesn't exist**

**}**

**private String getProductDetailsFileName( String productCategory) { return productCategory.toLowerCase() + "\_details.txt";**

**}**

**public void addProductToCategory(Product product) {**

**String category = product.getCategory();**

**String fileName = getProductDetailsFileName(category);**

**try (PrintWriter out = new PrintWriter(new FileWriter(fileName, true))) { out.println(product.toString());**

**} catch (IOException e) { e.printStackTrace();**

**}**

**}**

**public int getNumProducts() { return numProducts;**

**}**

**public void setNumProducts(int numProducts) { this.numProducts = numProducts;**

**}**

**public String getCompanyName() { return companyName;**

**}**

**public String getProductCategory() { return productCategory;**

**}**

**}**

**static class Product { private String prodId;**

**private String productName; private String brand;**

**private double price; private int stock;**

**private String imagePath;**

**public Product(String prodId, String productName, String brand, double price, int stock, String imagePath) {**

**this.prodId = prodId; this.productName = productName; this.brand = brand;**

**this.price = price; this.stock = stock;**

**this.imagePath = imagePath;**

**}**

**public String getProdId() { return prodId;**

**}**

**public String getProductName() { return productName;**

**}**

**public String getBrand() { return brand;**

**}**

**public double getPrice() { return price;**

**}**

**public int getStock() { return stock;**

**}**

**public String getImagePath() { return imagePath;**

**}**

**public String getCategory() { return prodId.substring(0, 1);**

**}**

**@Override**

**public String toString() {**



**return prodId + " " + productName + " " + brand + " " + price + " " + stock + " " + imagePath;**

**}**

**}**

**}**

This page is executed after customer logs in

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

import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

public class HomePage extends JFrame { public HomePage() {

setTitle("Home Page"); setSize(600, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create buttons with images for each category

JButton clothingButton = createCategoryButton("Clothing", "c\_details.txt", "clothes.jpeg"); JButton electronicsButton = createCategoryButton("Electronics", "e\_details.txt",

"electronic.jpeg");

JButton furnitureButton = createCategoryButton("Furniture", "f\_details.txt", "furniture.jpeg"); JButton footwearButton = createCategoryButton("Footwear", "ft\_details.txt",

"FOOTWEAR.jpeg");

// Create panel for buttons

JPanel contentPanel = new JPanel(); contentPanel.setLayout(new GridLayout(2, 2)); contentPanel.add(clothingButton); contentPanel.add(electronicsButton); contentPanel.add(furnitureButton); contentPanel.add(footwearButton);

add(contentPanel); setLocationRelativeTo(null);

setVisible(true);

}

private JButton createCategoryButton(String categoryName, String filePath, String imagePath) { ImageIcon icon = new ImageIcon(imagePath);

Image scaledImage = icon.getImage().getScaledInstance(150, 150, Image.SCALE\_SMOOTH);

ImageIcon scaledIcon = new ImageIcon(scaledImage);

JButton button = new JButton(categoryName, scaledIcon); button.setVerticalTextPosition(SwingConstants.BOTTOM); button.setHorizontalTextPosition(SwingConstants.CENTER);

button.addActionListener(new ActionListener() { @Override

public void actionPerformed(ActionEvent e) { openProductDetailsViewer(categoryName, filePath);

}

});

return button;

}

private void openProductDetailsViewer(String category, String filePath) {

// Create instances of ProductDetailsViewer based on the selected category if (category.equalsIgnoreCase("Clothing")) {

SwingUtilities.invokeLater(() -> new ProductDetailsViewerc(filePath).show());

} else if (category.equalsIgnoreCase("Electronics")) { SwingUtilities.invokeLater(() -> new ProductDetailsViewere(filePath).show());

} else if (category.equalsIgnoreCase("Furniture")) { SwingUtilities.invokeLater(() -> new ProductDetailsViewerf(filePath).show());

} else if (category.equalsIgnoreCase("Home Appliances")) { SwingUtilities.invokeLater(() -> new ProductDetailsViewerh(filePath).show());

}

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> new HomePage());

}



}

This is the page for clothing

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

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*;

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

public class ProductDetailsViewerc { private JFrame frame;

private JLabel productLabel; private JTextArea infoTextArea; private List<Product> products; private int currentProductIndex;

private String reviewsFilePath; // Added field for reviews file path private String wishlistFilePath; // Added field for wishlist file path

public ProductDetailsViewerc(String filePath) { frame = new JFrame("Product Details Viewer"); productLabel = new JLabel();

productLabel.setHorizontalAlignment(JLabel.CENTER); productLabel.setVerticalAlignment(JLabel.CENTER);

infoTextArea = new JTextArea(); infoTextArea.setEditable(false); infoTextArea.setLineWrap(true); infoTextArea.setWrapStyleWord(true);

JButton toggleButton = new JButton("Next Product"); toggleButton.addActionListener(e -> toggleProduct());

// Add Review button

JButton addReviewButton = new JButton("Add Review");

addReviewButton.addActionListener(e -> addReview());

// Add Add to Wishlist button

JButton addToWishlistButton = new JButton("Add to Wishlist"); addToWishlistButton.addActionListener(e -> addToWishlist());

// Add Show Wishlist button

JButton showWishlistButton = new JButton("Show Wishlist"); showWishlistButton.addActionListener(e -> showWishlist());

JPanel buttonPanel = new JPanel(); buttonPanel.add(toggleButton); buttonPanel.add(addReviewButton); buttonPanel.add(addToWishlistButton); buttonPanel.add(showWishlistButton);

frame.setLayout(new BorderLayout()); frame.add(productLabel, BorderLayout.CENTER); frame.add(infoTextArea, BorderLayout.NORTH); frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setSize(400, 400);

frame.setLocationRelativeTo(null);

// Set reviews file path based on the product category reviewsFilePath = filePath.replace("\_details.txt", "\_reviews.txt");

// Set wishlist file path based on the product category wishlistFilePath = "wishlistfile.txt";

loadProductDetailsFromFile(filePath); updateProductInfo();

}

private void toggleProduct() {

currentProductIndex = (currentProductIndex + 1) % products.size(); updateProductInfo();

}

private void updateProductInfo() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex); infoTextArea.setText(currentProduct.toString());

// Show reviews

String reviews = loadReviewsFromFile(currentProduct.getProdId()); infoTextArea.append("\n\nReviews:\n" + reviews);

150);

// Assuming the last part of each line is the image path

ImageIcon productImage = getScaledImageIcon(currentProduct.getImagePath(), 200,

productLabel.setIcon(productImage);

} else {

infoTextArea.setText("No product details available."); productLabel.setIcon(null);

}

}

private ImageIcon getScaledImageIcon(String imagePath, int width, int height) { ImageIcon originalIcon = new ImageIcon(imagePath);

Image originalImage = originalIcon.getImage();

Image scaledImage = originalImage.getScaledInstance(width, height, Image.SCALE\_SMOOTH);

return new ImageIcon(scaledImage);

}

private void loadProductDetailsFromFile(String filePath) { products = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 6) { // Ensure at least six parts to construct a product String prodId = parts[0];

String productName = parts[1]; String brand = parts[2];

double price = Double.parseDouble(parts[3]); int stock = Integer.parseInt(parts[4]);

String imagePath = parts[5];

imagePath);

}

}

Product product = new Product(prodId, productName, brand, price, stock, products.add(product);

} catch (IOException e) { e.printStackTrace();

}

}

private String loadReviewsFromFile(String prodId) { StringBuilder reviews = new StringBuilder();

try (BufferedReader reader = new BufferedReader(new FileReader(reviewsFilePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+", 2);

if (parts.length >= 2 && parts[0].equals(prodId)) { reviews.append(parts[1]).append("\n");

}

}

} catch (IOException e) { e.printStackTrace();

}

return reviews.toString();

}

private void addReview() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex);

String review = JOptionPane.showInputDialog("Enter your review:");

try (PrintWriter writer = new PrintWriter(new FileWriter(reviewsFilePath, true))) {

writer.println(currentProduct.getProdId() + " " + review); showMessage("Review added successfully!");

} catch (IOException e) { e.printStackTrace(); showMessage("Error adding review.");

}

updateProductInfo();

}

}

private void addToWishlist() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex);

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) {

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath, true))) { writer.println(username + " " + currentProduct.getProductName() + " " +

currentProduct.getPrice());

showMessage("Added to Wishlist!");

} catch (IOException e) { e.printStackTrace();

showMessage("Error adding to Wishlist.");

}

} else {

showMessage("Username cannot be empty. Please try again.");

}

}

}

private void showWishlist() {

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:"); if (username != null && !username.trim().isEmpty()) {

displayWishlist(username);

} else {

showMessage("Username cannot be empty. Please try again.");

}

}

private void displayWishlist(String username) {

JFrame wishlistFrame = new JFrame("Wishlist for " + username); JPanel wishlistPanel = new JPanel();

wishlistPanel.setLayout(new BoxLayout(wishlistPanel, BoxLayout.Y\_AXIS));

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)) { String productName = parts[1];

double price = Double.parseDouble(parts[2]);

// Create a panel for each wishlist item JPanel itemPanel = new JPanel(); itemPanel.setLayout(new BorderLayout());

// Display wishlist item information

JLabel itemLabel = new JLabel("Product: " + productName + ", Price: " + price); itemPanel.add(itemLabel, BorderLayout.CENTER);

// Add Remove from Wishlist button

JButton removeFromWishlistButton = new JButton("Remove from Wishlist"); removeFromWishlistButton.addActionListener(e -> removeFromWishlist(username,

productName, price, wishlistFrame));

itemPanel.add(removeFromWishlistButton, BorderLayout.EAST);

wishlistPanel.add(itemPanel);

}

}

} catch (IOException e) { e.printStackTrace();

}

if (wishlistPanel.getComponentCount() > 0) {

// Add Pay button

JButton payButton = new JButton("Pay");

payButton.addActionListener(e -> performPayment(username, wishlistPanel)); wishlistPanel.add(payButton);

// Display wishlist in the new frame

JScrollPane scrollPane = new JScrollPane(wishlistPanel); scrollPane.setPreferredSize(new Dimension(300, 200)); wishlistFrame.add(scrollPane);

wishlistFrame.pack();

wishlistFrame.setLocationRelativeTo(frame); // Set location relative to the main frame wishlistFrame.setVisible(true);

} else {

showMessage("Wishlist is empty for " + username);

}

}

private void performPayment(String username, JPanel wishlistPanel) { double totalPrice = calculateTotalPrice(wishlistPanel);

// Prompt user to enter credit card details

String cardNumber = JOptionPane.showInputDialog("Enter your credit card number:");

String expiryDate = JOptionPane.showInputDialog("Enter your credit card expiry date (MM/YY):");

String cardholderName = JOptionPane.showInputDialog("Enter your credit cardholder name:");

String cvv = JOptionPane.showInputDialog("Enter your CVV number:");

if (cardNumber != null && !cardNumber.trim().isEmpty() && expiryDate != null && !expiryDate.trim().isEmpty()

&& cardholderName != null && !cardholderName.trim().isEmpty() && cvv != null && !cvv.trim().isEmpty()) {

// Display payment successful message showMessage("Payment successful!");

// Display purchased product information displayPurchasedProducts(username, wishlistPanel);

// Clear the wishlist after successful payment clearWishlist(username, wishlistPanel);

} else {

showMessage("Credit card details cannot be empty. Payment failed.");

}

}

private double calculateTotalPrice(JPanel wishlistPanel) { double totalPrice = 0.0;

// Iterate over the components in the wishlistPanel to calculate the total price for (Component component : wishlistPanel.getComponents()) {

if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents()) { if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText();

// Extract price from the text (assuming the format "Product: <productName>, Price:

<price>")

String[] parts = text.split("Price: "); if (parts.length >= 2) {

double price = Double.parseDouble(parts[1]); totalPrice += price;

}

}

}

}

}

return totalPrice;

}

private void displayPurchasedProducts(String username, JPanel wishlistPanel) {

// Display purchased product information in a new frame

JFrame purchaseFrame = new JFrame("Purchased Products for " + username); JPanel purchasePanel = new JPanel();

purchasePanel.setLayout(new BoxLayout(purchasePanel, BoxLayout.Y\_AXIS));

// Iterate over the components in the wishlistPanel to extract product information for (Component component : wishlistPanel.getComponents()) {

if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents()) { if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText();

JLabel purchasedLabel = new JLabel("Purchased: " + text); purchasePanel.add(purchasedLabel);

}

}

}

}

// Display purchased product information in a scrollable dialog JScrollPane scrollPane = new JScrollPane(purchasePanel); scrollPane.setPreferredSize(new Dimension(300, 200)); purchaseFrame.add(scrollPane);

purchaseFrame.pack();

purchaseFrame.setLocationRelativeTo(frame); // Set location relative to the main frame purchaseFrame.setVisible(true);

}

private void clearWishlist(String username, JPanel wishlistPanel) {

// Remove purchased items from the wishlistPanel List<Component> componentsToRemove = new ArrayList<>(); for (Component component : wishlistPanel.getComponents()) {

if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component; componentsToRemove.add(itemPanel);

}

}

for (Component component : componentsToRemove) {

wishlistPanel.remove(component);

}

// Refresh the wishlistPanel wishlistPanel.revalidate(); wishlistPanel.repaint();

// Remove purchased items from the wishlist file try {

List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)) { continue; // Skip the line for purchased item

}

lines.add(line);

}

}

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath))) { for (String line : lines) {

writer.println(line);

}

}

} catch (IOException e) { e.printStackTrace();

showMessage("Error clearing purchased items from Wishlist.");

}

}

private void removeFromWishlist(String username, String productName, double price, JFrame wishlistFrame) {

try {

// Read all lines from the wishlist file List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { lines.add(line);

}

}

// Rewrite the wishlist file excluding the item to be removed

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath))) { for (String line : lines) {

String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)

&& parts[1].equals(productName) && Double.parseDouble(parts[2]) == price) {

// Skip the line to remove from the wishlist continue;

}

writer.println(line);

}

}

// Close the previous wishlist frame wishlistFrame.dispose();

// Update the displayed wishlist displayWishlist(username);

} catch (IOException e) { e.printStackTrace();

showMessage("Error removing item from Wishlist.");

}

}

private void showMessage(String message) { JOptionPane.showMessageDialog(frame, message);

}

public void show() {

SwingUtilities.invokeLater(() -> frame.setVisible(true));

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new ProductDetailsViewerc("c\_details.txt").show());

}

static class Product { private String prodId;

private String productName; private String brand;

private double price; private int stock;

private String imagePath;

public Product(String prodId, String productName, String brand, double price, int stock, String imagePath) {

this.prodId = prodId; this.productName = productName; this.brand = brand;

this.price = price; this.stock = stock;

this.imagePath = imagePath;

}

public String getProdId() { return prodId;

}

public String getProductName() { return productName;

}

public String getBrand() { return brand;

}

public double getPrice() { return price;

}

public int getStock() {



return stock;

}

public String getImagePath() { return imagePath;

}

@Override

public String toString() {

return String.format("Product ID: %s\nName: %s\nBrand: %s\nPrice: %.2f\nStock: %d", prodId, productName, brand, price, stock);

}

}

}

This is the page for electronics

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

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*;

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

public class ProductDetailsViewere { private JFrame frame;

private JLabel productLabel; private JTextArea infoTextArea; private List<Product> products; private int currentProductIndex;

private String reviewsFilePath; // Added field for reviews file path private String wishlistFilePath; // Added field for wishlist file path

public ProductDetailsViewere(String filePath) { frame = new JFrame("Product Details Viewer");

productLabel = new JLabel(); productLabel.setHorizontalAlignment(JLabel.CENTER); productLabel.setVerticalAlignment(JLabel.CENTER);

infoTextArea = new JTextArea(); infoTextArea.setEditable(false); infoTextArea.setLineWrap(true); infoTextArea.setWrapStyleWord(true);

JButton toggleButton = new JButton("Next Product"); toggleButton.addActionListener(e -> toggleProduct());

// Add Review button

JButton addReviewButton = new JButton("Add Review"); addReviewButton.addActionListener(e -> addReview());

// Add Add to Wishlist button

JButton addToWishlistButton = new JButton("Add to Wishlist"); addToWishlistButton.addActionListener(e -> addToWishlist());

// Add Show Wishlist button

JButton showWishlistButton = new JButton("Show Wishlist"); showWishlistButton.addActionListener(e -> showWishlist());

JPanel buttonPanel = new JPanel(); buttonPanel.add(toggleButton); buttonPanel.add(addReviewButton); buttonPanel.add(addToWishlistButton); buttonPanel.add(showWishlistButton);

frame.setLayout(new BorderLayout()); frame.add(productLabel, BorderLayout.CENTER); frame.add(infoTextArea, BorderLayout.NORTH); frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setSize(400, 400);

frame.setLocationRelativeTo(null);

// Set reviews file path based on the product category reviewsFilePath = filePath.replace("\_details.txt", "\_reviews.txt");

// Set wishlist file path based on the product category wishlistFilePath = "wishlistfile.txt";

loadProductDetailsFromFile(filePath); updateProductInfo();

}

private void toggleProduct() {

currentProductIndex = (currentProductIndex + 1) % products.size(); updateProductInfo();

}

private void updateProductInfo() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex); infoTextArea.setText(currentProduct.toString());

// Show reviews

String reviews = loadReviewsFromFile(currentProduct.getProdId()); infoTextArea.append("\n\nReviews:\n" + reviews);

150);

// Assuming the last part of each line is the image path

ImageIcon productImage = getScaledImageIcon(currentProduct.getImagePath(), 200,

productLabel.setIcon(productImage);

} else {

infoTextArea.setText("No product details available."); productLabel.setIcon(null);

}

}

private ImageIcon getScaledImageIcon(String imagePath, int width, int height) { ImageIcon originalIcon = new ImageIcon(imagePath);

Image originalImage = originalIcon.getImage();

Image scaledImage = originalImage.getScaledInstance(width, height, Image.SCALE\_SMOOTH);

return new ImageIcon(scaledImage);

}

private void loadProductDetailsFromFile(String filePath) { products = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 6) { // Ensure at least six parts to construct a product String prodId = parts[0];

String productName = parts[1]; String brand = parts[2];

double price = Double.parseDouble(parts[3]); int stock = Integer.parseInt(parts[4]);

String imagePath = parts[5];

imagePath);

}

}

Product product = new Product(prodId, productName, brand, price, stock, products.add(product);

} catch (IOException e) { e.printStackTrace();

}

}

private String loadReviewsFromFile(String prodId) { StringBuilder reviews = new StringBuilder();

try (BufferedReader reader = new BufferedReader(new FileReader(reviewsFilePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+", 2);

if (parts.length >= 2 && parts[0].equals(prodId)) { reviews.append(parts[1]).append("\n");

}

}

} catch (IOException e) { e.printStackTrace();

}

return reviews.toString();

}

private void addReview() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex);

String review = JOptionPane.showInputDialog("Enter your review:");

try (PrintWriter writer = new PrintWriter(new FileWriter(reviewsFilePath, true))) { writer.println(currentProduct.getProdId() + " " + review); showMessage("Review added successfully!");

} catch (IOException e) { e.printStackTrace(); showMessage("Error adding review.");

}

updateProductInfo();

}

}

private void addToWishlist() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex);

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) {

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath, true))) { writer.println(username + " " + currentProduct.getProductName() + " " +

currentProduct.getPrice());

showMessage("Added to Wishlist!");

} catch (IOException e) {

e.printStackTrace();

showMessage("Error adding to Wishlist.");

}

} else {

showMessage("Username cannot be empty. Please try again.");

}

}

}

private void showWishlist() {

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) { displayWishlist(username);

} else {

showMessage("Username cannot be empty. Please try again.");

}

}

private void displayWishlist(String username) {

JFrame wishlistFrame = new JFrame("Wishlist for " + username); JPanel wishlistPanel = new JPanel();

wishlistPanel.setLayout(new BoxLayout(wishlistPanel, BoxLayout.Y\_AXIS));

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)) { String productName = parts[1];

double price = Double.parseDouble(parts[2]);

// Create a panel for each wishlist item JPanel itemPanel = new JPanel(); itemPanel.setLayout(new BorderLayout());

// Display wishlist item information

JLabel itemLabel = new JLabel("Product: " + productName + ", Price: " + price); itemPanel.add(itemLabel, BorderLayout.CENTER);

// Add Remove from Wishlist button

JButton removeFromWishlistButton = new JButton("Remove from Wishlist"); removeFromWishlistButton.addActionListener(e -> removeFromWishlist(username,

productName, price, wishlistFrame));

itemPanel.add(removeFromWishlistButton, BorderLayout.EAST);

wishlistPanel.add(itemPanel);

}

}

} catch (IOException e) { e.printStackTrace();

}

if (wishlistPanel.getComponentCount() > 0) {

// Add Pay button

JButton payButton = new JButton("Pay");

payButton.addActionListener(e -> performPayment(username, wishlistPanel)); wishlistPanel.add(payButton);

// Display wishlist in the new frame

JScrollPane scrollPane = new JScrollPane(wishlistPanel); scrollPane.setPreferredSize(new Dimension(300, 200)); wishlistFrame.add(scrollPane);

wishlistFrame.pack();

wishlistFrame.setLocationRelativeTo(frame); // Set location relative to the main frame wishlistFrame.setVisible(true);

} else {

showMessage("Wishlist is empty for " + username);

}

}

private void performPayment(String username, JPanel wishlistPanel) { double totalPrice = calculateTotalPrice(wishlistPanel);

// Prompt user to enter credit card details

String cardNumber = JOptionPane.showInputDialog("Enter your credit card number:");

String expiryDate = JOptionPane.showInputDialog("Enter your credit card expiry date (MM/YY):");

String cardholderName = JOptionPane.showInputDialog("Enter your credit cardholder name:");

String cvv = JOptionPane.showInputDialog("Enter your CVV number:");

if (cardNumber != null && !cardNumber.trim().isEmpty() && expiryDate != null && !expiryDate.trim().isEmpty()

&& cardholderName != null && !cardholderName.trim().isEmpty() && cvv != null && !cvv.trim().isEmpty()) {

// Display payment successful message showMessage("Payment successful!");

// Display purchased product information displayPurchasedProducts(username, wishlistPanel);

// Clear the wishlist after successful payment clearWishlist(username, wishlistPanel);

} else {

showMessage("Credit card details cannot be empty. Payment failed.");

}

}

private double calculateTotalPrice(JPanel wishlistPanel) { double totalPrice = 0.0;

// Iterate over the components in the wishlistPanel to calculate the total price for (Component component : wishlistPanel.getComponents()) {

if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents()) { if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText();

// Extract price from the text (assuming the format "Product: <productName>, Price:

<price>")

String[] parts = text.split("Price: "); if (parts.length >= 2) {

double price = Double.parseDouble(parts[1]); totalPrice += price;

}

}

}

}

}

return totalPrice;

}

private void displayPurchasedProducts(String username, JPanel wishlistPanel) {

// Display purchased product information in a new frame

JFrame purchaseFrame = new JFrame("Purchased Products for " + username); JPanel purchasePanel = new JPanel();

purchasePanel.setLayout(new BoxLayout(purchasePanel, BoxLayout.Y\_AXIS));

// Iterate over the components in the wishlistPanel to extract product information for (Component component : wishlistPanel.getComponents()) {

if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents()) { if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText();

JLabel purchasedLabel = new JLabel("Purchased: " + text); purchasePanel.add(purchasedLabel);

}

}

}

}

// Display purchased product information in a scrollable dialog JScrollPane scrollPane = new JScrollPane(purchasePanel); scrollPane.setPreferredSize(new Dimension(300, 200)); purchaseFrame.add(scrollPane);

purchaseFrame.pack();

purchaseFrame.setLocationRelativeTo(frame); // Set location relative to the main frame purchaseFrame.setVisible(true);

}

private void clearWishlist(String username, JPanel wishlistPanel) {

// Remove purchased items from the wishlistPanel List<Component> componentsToRemove = new ArrayList<>(); for (Component component : wishlistPanel.getComponents()) {

if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component; componentsToRemove.add(itemPanel);

}

}

for (Component component : componentsToRemove) { wishlistPanel.remove(component);

}

// Refresh the wishlistPanel wishlistPanel.revalidate(); wishlistPanel.repaint();

// Remove purchased items from the wishlist file try {

List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) { String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)) { continue; // Skip the line for purchased item

}

lines.add(line);

}

}

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath))) { for (String line : lines) {

writer.println(line);

}

}

} catch (IOException e) { e.printStackTrace();

showMessage("Error clearing purchased items from Wishlist.");

}

}

private void removeFromWishlist(String username, String productName, double price, JFrame wishlistFrame) {

try {

// Read all lines from the wishlist file List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) { String line;

while ((line = reader.readLine()) != null) { lines.add(line);

}

}

// Rewrite the wishlist file excluding the item to be removed

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath))) { for (String line : lines) {

String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)

&& parts[1].equals(productName) && Double.parseDouble(parts[2]) == price) {

// Skip the line to remove from the wishlist continue;

}

writer.println(line);

}

}

// Close the previous wishlist frame wishlistFrame.dispose();

// Update the displayed wishlist displayWishlist(username);

} catch (IOException e) { e.printStackTrace();

showMessage("Error removing item from Wishlist.");

}

}

private void showMessage(String message) { JOptionPane.showMessageDialog(frame, message);

}

public void show() {

SwingUtilities.invokeLater(() -> frame.setVisible(true));

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new ProductDetailsViewere("e\_details.txt").show());

}

static class Product { private String prodId;

private String productName; private String brand;

private double price; private int stock;

private String imagePath;

public Product(String prodId, String productName, String brand, double price, int stock, String imagePath) {

this.prodId = prodId; this.productName = productName; this.brand = brand;

this.price = price; this.stock = stock;

this.imagePath = imagePath;

}

public String getProdId() { return prodId;

}



public String getProductName() { return productName;

}

public String getBrand() { return brand;

}

public double getPrice() { return price;

}

public int getStock() { return stock;

}

public String getImagePath() { return imagePath;

}

@Override

public String toString() {

return String.format("Product ID: %s\nName: %s\nBrand: %s\nPrice: %.2f\nStock: %d", prodId, productName, brand, price, stock);

}

}

}

This is the page executed for footwear

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

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*;

import java.util.ArrayList;

import java.util.List;

public class ProductDetailsViewerf{ private JFrame frame;

private JLabel productLabel; private JTextArea infoTextArea; private List<Product> products; private int currentProductIndex;

private String reviewsFilePath; // Added field for reviews file path private String wishlistFilePath; // Added field for wishlist file

path

public ProductDetailsViewerf(String filePath) { frame = new JFrame("Product Details Viewer"); productLabel = new JLabel();

productLabel.setHorizontalAlignment(JLabel.CENTER); productLabel.setVerticalAlignment(JLabel.CENTER);

infoTextArea = new JTextArea(); infoTextArea.setEditable(false); infoTextArea.setLineWrap(true); infoTextArea.setWrapStyleWord(true);

JButton toggleButton = new JButton("Next Product"); toggleButton.addActionListener(e -> toggleProduct());

// Add Review button

JButton addReviewButton = new JButton("Add Review"); addReviewButton.addActionListener(e -> addReview());

// Add Add to Wishlist button

JButton addToWishlistButton = new JButton("Add to Wishlist"); addToWishlistButton.addActionListener(e -> addToWishlist());

// Add Show Wishlist button

JButton showWishlistButton = new JButton("Show Wishlist"); showWishlistButton.addActionListener(e -> showWishlist());

JPanel buttonPanel = new JPanel();

buttonPanel.add(toggleButton); buttonPanel.add(addReviewButton); buttonPanel.add(addToWishlistButton); buttonPanel.add(showWishlistButton);

frame.setLayout(new BorderLayout()); frame.add(productLabel, BorderLayout.CENTER); frame.add(infoTextArea, BorderLayout.NORTH); frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setSize(400, 400); frame.setLocationRelativeTo(null);

// Set reviews file path based on the product category reviewsFilePath = filePath.replace("\_details.txt",

"\_reviews.txt");

// Set wishlist file path based on the product category wishlistFilePath = "wishlistfile.txt";

loadProductDetailsFromFile(filePath); updateProductInfo();

}

private void toggleProduct() {

currentProductIndex = (currentProductIndex + 1) % products.size();

updateProductInfo();

}

private void updateProductInfo() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex); infoTextArea.setText(currentProduct.toString());

// Show reviews String reviews =

loadReviewsFromFile(currentProduct.getProdId());

infoTextArea.append("\n\nReviews:\n" + reviews);

// Assuming the last part of each line is the image path ImageIcon productImage =

getScaledImageIcon(currentProduct.getImagePath(), 200, 150); productLabel.setIcon(productImage);

} else {

infoTextArea.setText("No product details available."); productLabel.setIcon(null);

}

}

private ImageIcon getScaledImageIcon(String imagePath, int width, int height) {

ImageIcon originalIcon = new ImageIcon(imagePath); Image originalImage = originalIcon.getImage();

Image scaledImage = originalImage.getScaledInstance(width, height, Image.SCALE\_SMOOTH);

return new ImageIcon(scaledImage);

}

private void loadProductDetailsFromFile(String filePath) { products = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 6) { // Ensure at least six parts to construct a product

String prodId = parts[0]; String productName = parts[1]; String brand = parts[2];

double price = Double.parseDouble(parts[3]); int stock = Integer.parseInt(parts[4]); String imagePath = parts[5];

Product product = new Product(prodId, productName,

brand, price, stock, imagePath);

products.add(product);

}

}

} catch (IOException e) { e.printStackTrace();

}

}

private String loadReviewsFromFile(String prodId) { StringBuilder reviews = new StringBuilder();

try (BufferedReader reader = new BufferedReader(new FileReader(reviewsFilePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+", 2);

if (parts.length >= 2 && parts[0].equals(prodId)) { reviews.append(parts[1]).append("\n");

}

}

} catch (IOException e) { e.printStackTrace();

}

return reviews.toString();

}

private void addReview() {

if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex); String review = JOptionPane.showInputDialog("Enter your

review:");

try (PrintWriter writer = new PrintWriter(new FileWriter(reviewsFilePath, true))) {

writer.println(currentProduct.getProdId() + " " +

review);

showMessage("Review added successfully!");

} catch (IOException e) { e.printStackTrace();

showMessage("Error adding review.");

}

updateProductInfo();

}

}

private void addToWishlist() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex);

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) { try (PrintWriter writer = new PrintWriter(new

FileWriter(wishlistFilePath, true))) {

writer.println(username + " " + currentProduct.getProductName() + " " + currentProduct.getPrice());

showMessage("Added to Wishlist!");

} catch (IOException e) { e.printStackTrace();

showMessage("Error adding to Wishlist.");

again.");

}

} else {

showMessage("Username cannot be empty. Please try

}

}

}

private void showWishlist() {

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) { displayWishlist(username);

} else {

showMessage("Username cannot be empty. Please try again.");

}

}

private void displayWishlist(String username) {

JFrame wishlistFrame = new JFrame("Wishlist for " + username); JPanel wishlistPanel = new JPanel(); wishlistPanel.setLayout(new BoxLayout(wishlistPanel,

BoxLayout.Y\_AXIS));

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)) { String productName = parts[1];

double price = Double.parseDouble(parts[2]);

// Create a panel for each wishlist item JPanel itemPanel = new JPanel(); itemPanel.setLayout(new BorderLayout());

// Display wishlist item information

JLabel itemLabel = new JLabel("Product: " + productName + ", Price: " + price);

itemPanel.add(itemLabel, BorderLayout.CENTER);

// Add Remove from Wishlist button JButton removeFromWishlistButton = new

JButton("Remove from Wishlist");

removeFromWishlistButton.addActionListener(e -> removeFromWishlist(username, productName, price, wishlistFrame));

itemPanel.add(removeFromWishlistButton,

BorderLayout.EAST);

wishlistPanel.add(itemPanel);

}

}

} catch (IOException e) { e.printStackTrace();

}

if (wishlistPanel.getComponentCount() > 0) {

// Add Pay button

JButton payButton = new JButton("Pay"); payButton.addActionListener(e -> performPayment(username,

wishlistPanel));

wishlistPanel.add(payButton);

// Display wishlist in the new frame

JScrollPane scrollPane = new JScrollPane(wishlistPanel); scrollPane.setPreferredSize(new Dimension(300, 200)); wishlistFrame.add(scrollPane);

wishlistFrame.pack(); wishlistFrame.setLocationRelativeTo(frame); // Set location

relative to the main frame

wishlistFrame.setVisible(true);

} else {

showMessage("Wishlist is empty for " + username);

}

}

private void performPayment(String username, JPanel wishlistPanel) { double totalPrice = calculateTotalPrice(wishlistPanel);

// Prompt user to enter credit card details

String cardNumber = JOptionPane.showInputDialog("Enter your credit card number:");

String expiryDate = JOptionPane.showInputDialog("Enter your credit card expiry date (MM/YY):");

String cardholderName = JOptionPane.showInputDialog("Enter your credit cardholder name:");

String cvv = JOptionPane.showInputDialog("Enter your CVV number:");

if (cardNumber != null && !cardNumber.trim().isEmpty()

&& expiryDate != null && !expiryDate.trim().isEmpty() && cardholderName != null &&

!cardholderName.trim().isEmpty()

&& cvv != null && !cvv.trim().isEmpty()) {

// Display payment successful message showMessage("Payment successful!");

// Display purchased product information displayPurchasedProducts(username, wishlistPanel);

// Clear the wishlist after successful payment clearWishlist(username, wishlistPanel);

} else {

showMessage("Credit card details cannot be empty. Payment

failed.");

}

}

private double calculateTotalPrice(JPanel wishlistPanel) { double totalPrice = 0.0;

// Iterate over the components in the wishlistPanel to calculate the total price

for (Component component : wishlistPanel.getComponents()) { if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents())

{

if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText();

// Extract price from the text (assuming the

format "Product: <productName>, Price: <price>")

String[] parts = text.split("Price: "); if (parts.length >= 2) {

double price = Double.parseDouble(parts[1]); totalPrice += price;

}

}

}

}

}

return totalPrice;

}

private void displayPurchasedProducts(String username, JPanel wishlistPanel) {

// Display purchased product information in a new frame JFrame purchaseFrame = new JFrame("Purchased Products for " +

username);

JPanel purchasePanel = new JPanel(); purchasePanel.setLayout(new BoxLayout(purchasePanel,

BoxLayout.Y\_AXIS));

// Iterate over the components in the wishlistPanel to extract product information

for (Component component : wishlistPanel.getComponents()) { if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents())

{

+ text);

if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText(); JLabel purchasedLabel = new JLabel("Purchased: "

purchasePanel.add(purchasedLabel);

}

}

}

}

// Display purchased product information in a scrollable dialog JScrollPane scrollPane = new JScrollPane(purchasePanel); scrollPane.setPreferredSize(new Dimension(300, 200)); purchaseFrame.add(scrollPane);

purchaseFrame.pack();

purchaseFrame.setLocationRelativeTo(frame); // Set location relative to the main frame

purchaseFrame.setVisible(true);

}

private void clearWishlist(String username, JPanel wishlistPanel) {

// Remove purchased items from the wishlistPanel List<Component> componentsToRemove = new ArrayList<>();

for (Component component : wishlistPanel.getComponents()) { if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component; componentsToRemove.add(itemPanel);

}

}

for (Component component : componentsToRemove) { wishlistPanel.remove(component);

}

// Refresh the wishlistPanel wishlistPanel.revalidate(); wishlistPanel.repaint();

// Remove purchased items from the wishlist file try {

List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username))

{

continue; // Skip the line for purchased item

}

lines.add(line);

}

}

try (PrintWriter writer = new PrintWriter(new

FileWriter(wishlistFilePath))) {

for (String line : lines) { writer.println(line);

}

}

} catch (IOException e) { e.printStackTrace();

showMessage("Error clearing purchased items from Wishlist.");

}

}

private void removeFromWishlist(String username, String productName, double price, JFrame wishlistFrame) {

try {

// Read all lines from the wishlist file List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { lines.add(line);

}

}

removed

// Rewrite the wishlist file excluding the item to be

try (PrintWriter writer = new PrintWriter(new

FileWriter(wishlistFilePath))) {

for (String line : lines) {

String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username) && parts[1].equals(productName) &&

Double.parseDouble(parts[2]) == price) {

// Skip the line to remove from the wishlist continue;

}

writer.println(line);

}

}

// Close the previous wishlist frame wishlistFrame.dispose();

// Update the displayed wishlist displayWishlist(username);

} catch (IOException e) { e.printStackTrace();

showMessage("Error removing item from Wishlist.");

}

}

private void showMessage(String message) { JOptionPane.showMessageDialog(frame, message);

}

public void show() {

SwingUtilities.invokeLater(() -> frame.setVisible(true));

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> new

ProductDetailsViewerf("F\_details.txt").show());

}

static class Product { private String prodId;

private String productName; private String brand; private double price; private int stock;

private String imagePath;

public Product(String prodId, String productName, String brand, double price, int stock, String imagePath) {

this.prodId = prodId; this.productName = productName; this.brand = brand;

this.price = price;

this.stock = stock; this.imagePath = imagePath;

}

public String getProdId() { return prodId;

}

public String getProductName() { return productName;

}

public String getBrand() { return brand;

}

public double getPrice() { return price;

}

public int getStock() { return stock;

}

public String getImagePath() { return imagePath;

}

@Override

public String toString() {

return String.format("Product ID: %s\nName: %s\nBrand:

%s\nPrice: %.2f\nStock: %d",

prodId, productName, brand, price, stock);

}

}

}

This is the code for furniture.

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

import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*;

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

public class ProductDetailsViewerh{ private JFrame frame;

private JLabel productLabel; private JTextArea infoTextArea; private List<Product> products; private int currentProductIndex;

private String reviewsFilePath; // Added field for reviews file path private String wishlistFilePath; // Added field for wishlist file

path

public ProductDetailsViewerh(String filePath) { frame = new JFrame("Product Details Viewer"); productLabel = new JLabel();

productLabel.setHorizontalAlignment(JLabel.CENTER); productLabel.setVerticalAlignment(JLabel.CENTER);

infoTextArea = new JTextArea(); infoTextArea.setEditable(false); infoTextArea.setLineWrap(true); infoTextArea.setWrapStyleWord(true);

JButton toggleButton = new JButton("Next Product"); toggleButton.addActionListener(e -> toggleProduct());

// Add Review button

JButton addReviewButton = new JButton("Add Review"); addReviewButton.addActionListener(e -> addReview());

// Add Add to Wishlist button

JButton addToWishlistButton = new JButton("Add to Wishlist");

addToWishlistButton.addActionListener(e -> addToWishlist());

// Add Show Wishlist button

JButton showWishlistButton = new JButton("Show Wishlist"); showWishlistButton.addActionListener(e -> showWishlist());

JPanel buttonPanel = new JPanel(); buttonPanel.add(toggleButton); buttonPanel.add(addReviewButton); buttonPanel.add(addToWishlistButton); buttonPanel.add(showWishlistButton);

frame.setLayout(new BorderLayout()); frame.add(productLabel, BorderLayout.CENTER); frame.add(infoTextArea, BorderLayout.NORTH); frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setSize(400, 400); frame.setLocationRelativeTo(null);

// Set reviews file path based on the product category reviewsFilePath = filePath.replace("\_details.txt",

"\_reviews.txt");

// Set wishlist file path based on the product category wishlistFilePath = "wishlistfile.txt";

loadProductDetailsFromFile(filePath); updateProductInfo();

}

private void toggleProduct() {

currentProductIndex = (currentProductIndex + 1) % products.size();

updateProductInfo();

}

private void updateProductInfo() {

if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex); infoTextArea.setText(currentProduct.toString());

// Show reviews String reviews =

loadReviewsFromFile(currentProduct.getProdId()); infoTextArea.append("\n\nReviews:\n" + reviews);

// Assuming the last part of each line is the image path ImageIcon productImage =

getScaledImageIcon(currentProduct.getImagePath(), 200, 150); productLabel.setIcon(productImage);

} else {

infoTextArea.setText("No product details available."); productLabel.setIcon(null);

}

}

private ImageIcon getScaledImageIcon(String imagePath, int width, int height) {

ImageIcon originalIcon = new ImageIcon(imagePath); Image originalImage = originalIcon.getImage();

Image scaledImage = originalImage.getScaledInstance(width, height, Image.SCALE\_SMOOTH);

return new ImageIcon(scaledImage);

}

private void loadProductDetailsFromFile(String filePath) { products = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 6) { // Ensure at least six parts to construct a product

String prodId = parts[0];

String productName = parts[1]; String brand = parts[2];

double price = Double.parseDouble(parts[3]); int stock = Integer.parseInt(parts[4]); String imagePath = parts[5];

Product product = new Product(prodId, productName, brand, price, stock, imagePath);

products.add(product);

}

}

} catch (IOException e) { e.printStackTrace();

}

}

private String loadReviewsFromFile(String prodId) { StringBuilder reviews = new StringBuilder();

try (BufferedReader reader = new BufferedReader(new FileReader(reviewsFilePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+", 2);

if (parts.length >= 2 && parts[0].equals(prodId)) { reviews.append(parts[1]).append("\n");

}

}

} catch (IOException e) { e.printStackTrace();

}

return reviews.toString();

}

private void addReview() {

if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex); String review = JOptionPane.showInputDialog("Enter your

review:");

try (PrintWriter writer = new PrintWriter(new FileWriter(reviewsFilePath, true))) {

writer.println(currentProduct.getProdId() + " " +

review);

showMessage("Review added successfully!");

} catch (IOException e) { e.printStackTrace(); showMessage("Error adding review.");

}

updateProductInfo();

}

}

private void addToWishlist() { if (!products.isEmpty()) {

Product currentProduct = products.get(currentProductIndex);

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) { try (PrintWriter writer = new PrintWriter(new

FileWriter(wishlistFilePath, true))) {

writer.println(username + " " + currentProduct.getProductName() + " " + currentProduct.getPrice());

showMessage("Added to Wishlist!");

} catch (IOException e) { e.printStackTrace();

showMessage("Error adding to Wishlist.");

again.");

}

} else {

showMessage("Username cannot be empty. Please try

}

}

}

private void showWishlist() {

// Prompt user for username

String username = JOptionPane.showInputDialog("Enter your username:");

if (username != null && !username.trim().isEmpty()) { displayWishlist(username);

} else {

showMessage("Username cannot be empty. Please try again.");

}

}

private void displayWishlist(String username) {

JFrame wishlistFrame = new JFrame("Wishlist for " + username); JPanel wishlistPanel = new JPanel(); wishlistPanel.setLayout(new BoxLayout(wishlistPanel,

BoxLayout.Y\_AXIS));

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)) { String productName = parts[1];

double price = Double.parseDouble(parts[2]);

// Create a panel for each wishlist item JPanel itemPanel = new JPanel(); itemPanel.setLayout(new BorderLayout());

// Display wishlist item information

JLabel itemLabel = new JLabel("Product: " + productName + ", Price: " + price);

itemPanel.add(itemLabel, BorderLayout.CENTER);

// Add Remove from Wishlist button JButton removeFromWishlistButton = new

JButton("Remove from Wishlist");

removeFromWishlistButton.addActionListener(e -> removeFromWishlist(username, productName, price, wishlistFrame));

itemPanel.add(removeFromWishlistButton,

BorderLayout.EAST);

wishlistPanel.add(itemPanel);

}

}

} catch (IOException e) { e.printStackTrace();

}

if (wishlistPanel.getComponentCount() > 0) {

// Add Pay button

JButton payButton = new JButton("Pay"); payButton.addActionListener(e -> performPayment(username,

wishlistPanel));

wishlistPanel.add(payButton);

// Display wishlist in the new frame

JScrollPane scrollPane = new JScrollPane(wishlistPanel); scrollPane.setPreferredSize(new Dimension(300, 200)); wishlistFrame.add(scrollPane);

wishlistFrame.pack(); wishlistFrame.setLocationRelativeTo(frame); // Set location

relative to the main frame

wishlistFrame.setVisible(true);

} else {

showMessage("Wishlist is empty for " + username);

}

}

private void performPayment(String username, JPanel wishlistPanel) { double totalPrice = calculateTotalPrice(wishlistPanel);

// Prompt user to enter credit card details

String cardNumber = JOptionPane.showInputDialog("Enter your credit card number:");

String expiryDate = JOptionPane.showInputDialog("Enter your

credit card expiry date (MM/YY):");

String cardholderName = JOptionPane.showInputDialog("Enter your credit cardholder name:");

String cvv = JOptionPane.showInputDialog("Enter your CVV number:");

if (cardNumber != null && !cardNumber.trim().isEmpty()

&& expiryDate != null && !expiryDate.trim().isEmpty() && cardholderName != null &&

!cardholderName.trim().isEmpty()

&& cvv != null && !cvv.trim().isEmpty()) {

// Display payment successful message showMessage("Payment successful!");

// Display purchased product information displayPurchasedProducts(username, wishlistPanel);

// Clear the wishlist after successful payment clearWishlist(username, wishlistPanel);

} else {

showMessage("Credit card details cannot be empty. Payment

failed.");

}

}

private double calculateTotalPrice(JPanel wishlistPanel) { double totalPrice = 0.0;

// Iterate over the components in the wishlistPanel to calculate the total price

for (Component component : wishlistPanel.getComponents()) { if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents())

{

if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText();

// Extract price from the text (assuming the format "Product: <productName>, Price: <price>")

String[] parts = text.split("Price: "); if (parts.length >= 2) {

double price = Double.parseDouble(parts[1]); totalPrice += price;

}

}

}

}

}

return totalPrice;

}

private void displayPurchasedProducts(String username, JPanel wishlistPanel) {

// Display purchased product information in a new frame JFrame purchaseFrame = new JFrame("Purchased Products for " +

username);

JPanel purchasePanel = new JPanel(); purchasePanel.setLayout(new BoxLayout(purchasePanel,

BoxLayout.Y\_AXIS));

// Iterate over the components in the wishlistPanel to extract product information

for (Component component : wishlistPanel.getComponents()) { if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component;

for (Component subComponent : itemPanel.getComponents())

{

+ text);

if (subComponent instanceof JLabel) {

String text = ((JLabel) subComponent).getText(); JLabel purchasedLabel = new JLabel("Purchased: "

purchasePanel.add(purchasedLabel);

}

}

}

}

// Display purchased product information in a scrollable dialog JScrollPane scrollPane = new JScrollPane(purchasePanel); scrollPane.setPreferredSize(new Dimension(300, 200)); purchaseFrame.add(scrollPane);

purchaseFrame.pack(); purchaseFrame.setLocationRelativeTo(frame); // Set location

relative to the main frame

purchaseFrame.setVisible(true);

}

private void clearWishlist(String username, JPanel wishlistPanel) {

// Remove purchased items from the wishlistPanel List<Component> componentsToRemove = new ArrayList<>();

for (Component component : wishlistPanel.getComponents()) { if (component instanceof JPanel) {

JPanel itemPanel = (JPanel) component; componentsToRemove.add(itemPanel);

}

}

for (Component component : componentsToRemove) { wishlistPanel.remove(component);

}

// Refresh the wishlistPanel wishlistPanel.revalidate(); wishlistPanel.repaint();

// Remove purchased items from the wishlist file try {

List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username))

{

continue; // Skip the line for purchased item

}

lines.add(line);

}

}

try (PrintWriter writer = new PrintWriter(new FileWriter(wishlistFilePath))) {

for (String line : lines) { writer.println(line);

}

}

} catch (IOException e) { e.printStackTrace();

showMessage("Error clearing purchased items from Wishlist.");

}

}

private void removeFromWishlist(String username, String productName, double price, JFrame wishlistFrame) {

try {

// Read all lines from the wishlist file List<String> lines = new ArrayList<>();

try (BufferedReader reader = new BufferedReader(new FileReader(wishlistFilePath))) {

String line;

while ((line = reader.readLine()) != null) { lines.add(line);

}

}

removed

// Rewrite the wishlist file excluding the item to be

try (PrintWriter writer = new PrintWriter(new

FileWriter(wishlistFilePath))) {

for (String line : lines) {

String[] parts = line.split("\\s+");

if (parts.length >= 3 && parts[0].equals(username)

&& parts[1].equals(productName) && Double.parseDouble(parts[2]) == price) {

// Skip the line to remove from the wishlist continue;

}

writer.println(line);

}

}

// Close the previous wishlist frame wishlistFrame.dispose();

// Update the displayed wishlist displayWishlist(username);

} catch (IOException e) { e.printStackTrace();

showMessage("Error removing item from Wishlist.");

}

}

private void showMessage(String message) { JOptionPane.showMessageDialog(frame, message);

}

public void show() {

SwingUtilities.invokeLater(() -> frame.setVisible(true));

}

public static void main(String[] args) { SwingUtilities.invokeLater(() -> new

ProductDetailsViewerh("F\_details.txt").show());

}

static class Product { private String prodId;

private String productName; private String brand; private double price; private int stock;

private String imagePath;



public Product(String prodId, String productName, String brand, double price, int stock, String imagePath) {

this.prodId = prodId; this.productName = productName; this.brand = brand;

this.price = price; this.stock = stock; this.imagePath = imagePath;

}

public String getProdId() { return prodId;

}

public String getProductName() { return productName;

}

public String getBrand() { return brand;

}

public double getPrice() { return price;

}

public int getStock() { return stock;

}

public String getImagePath() { return imagePath;

}

@Override

public String toString() {

return String.format("Product ID: %s\nName: %s\nBrand:

%s\nPrice: %.2f\nStock: %d",



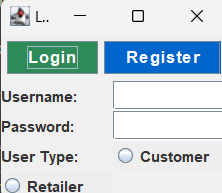
prodId, productName, brand, price, stock);

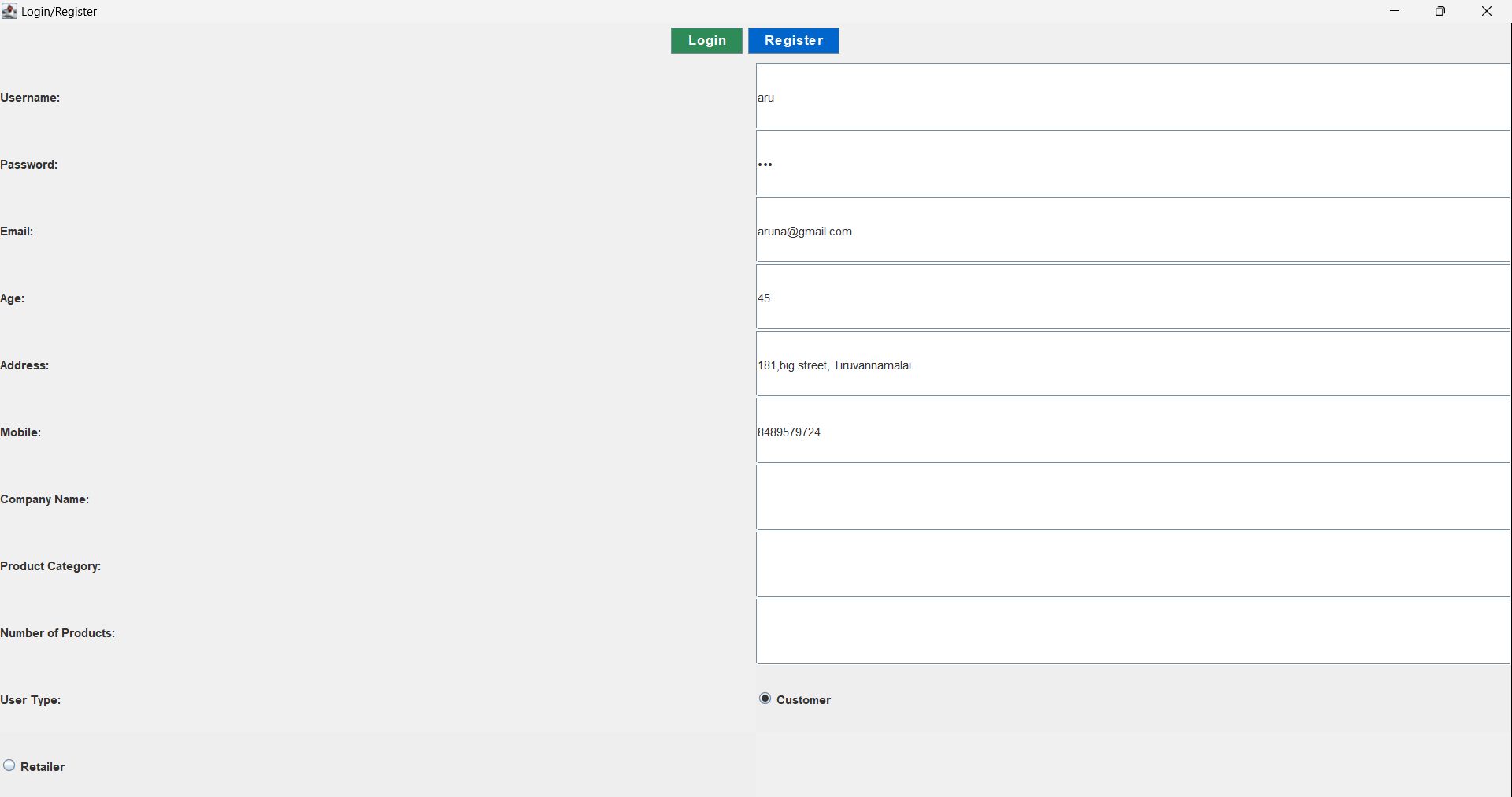
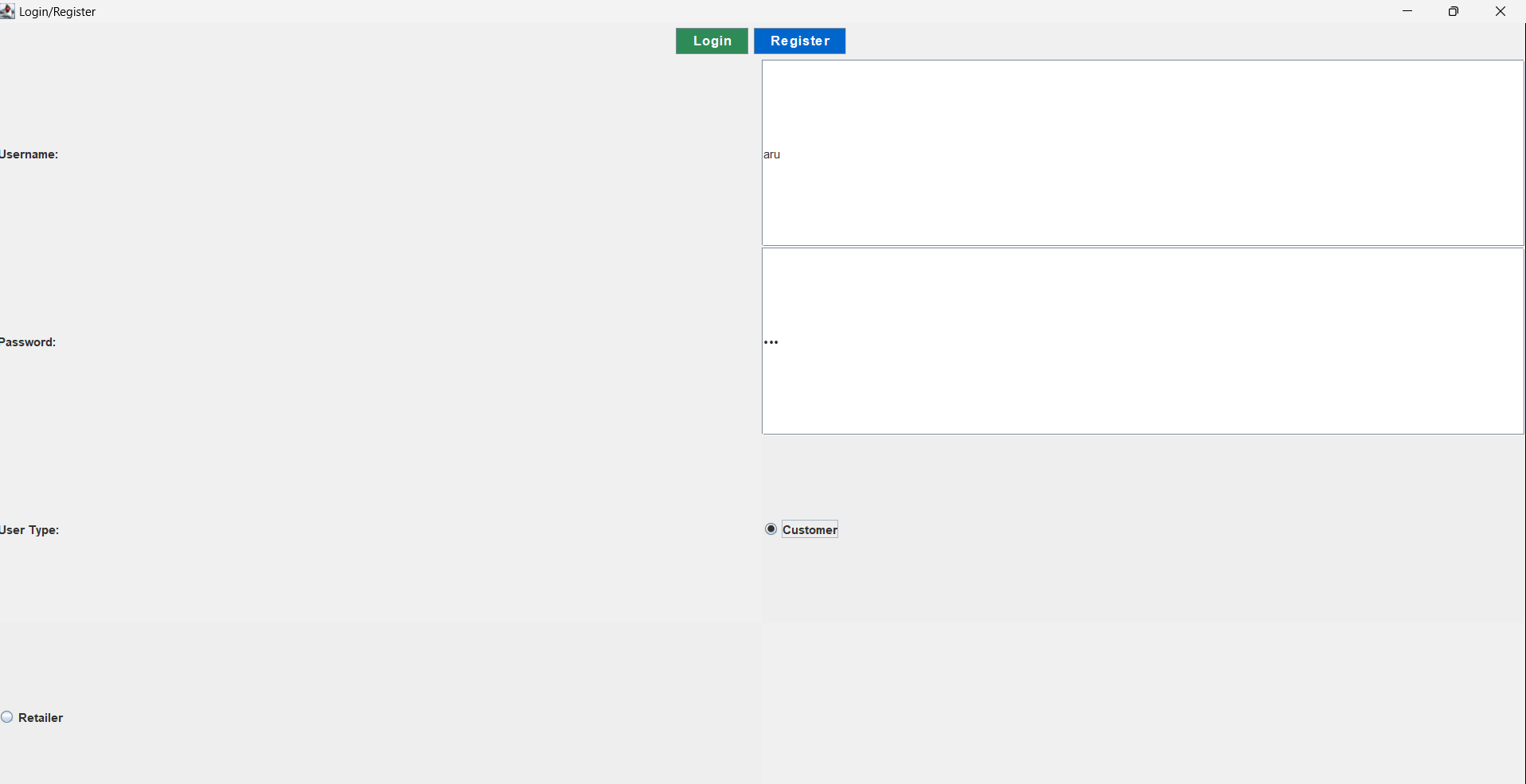
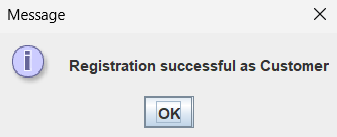
}

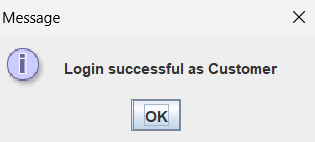
}

}

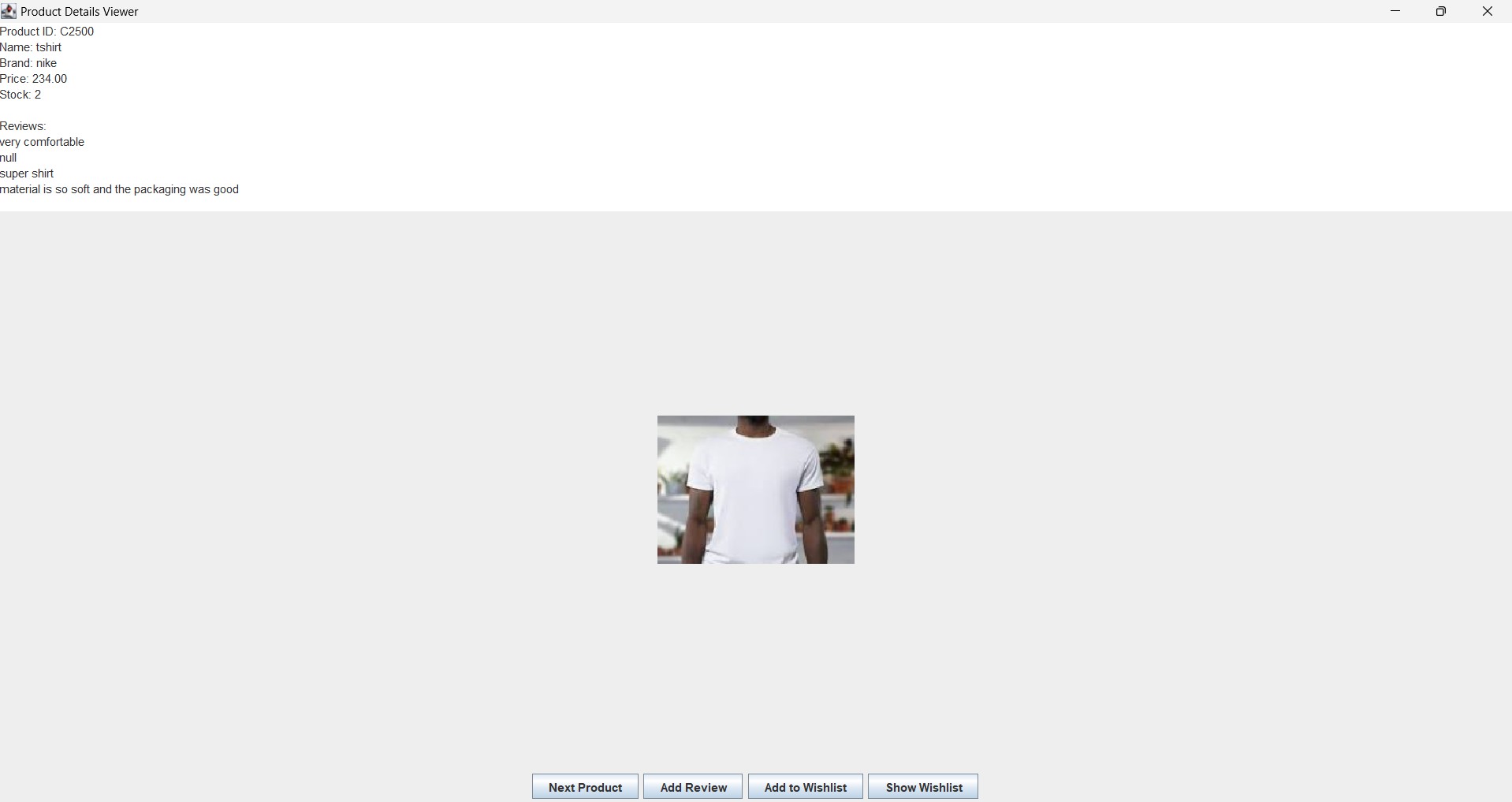
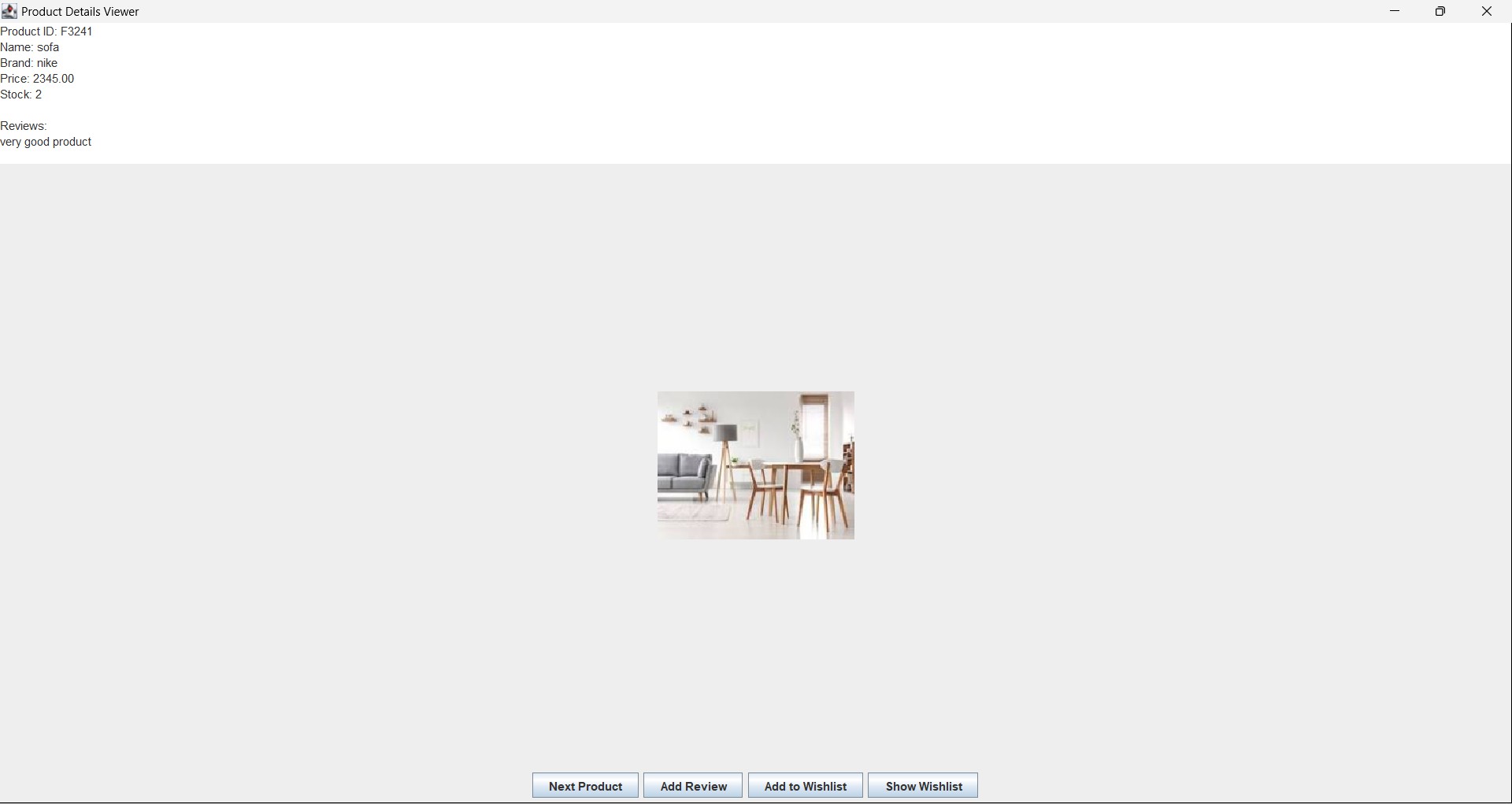
# OUTPUT:

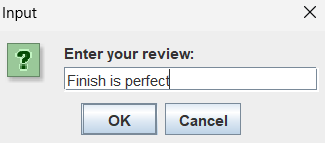


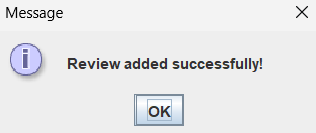


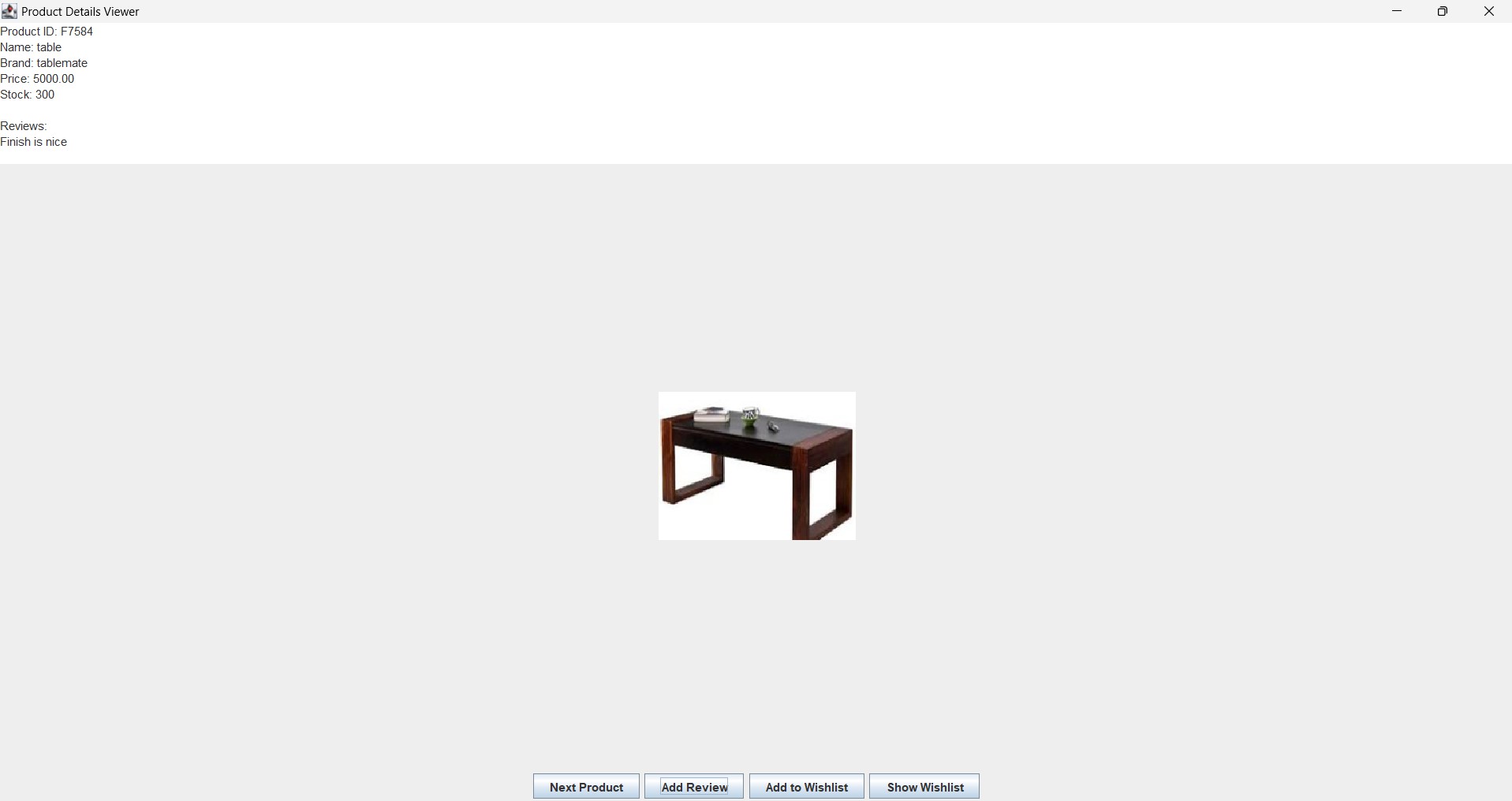


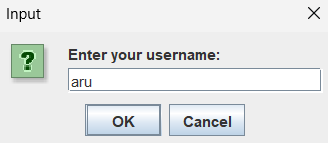


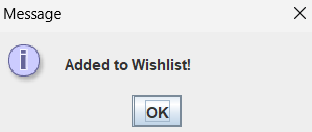


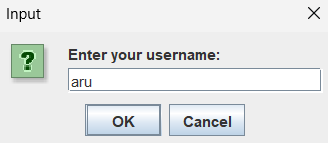


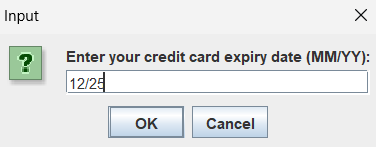
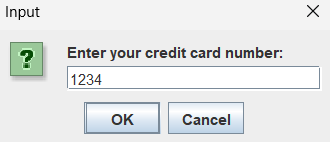
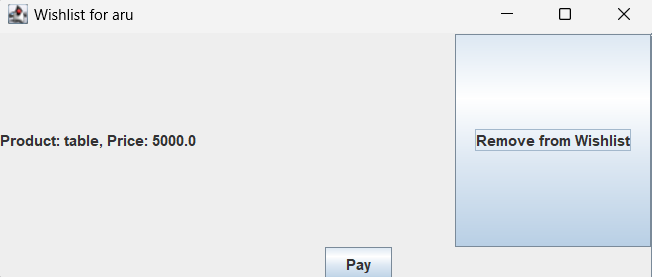


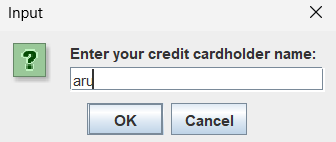


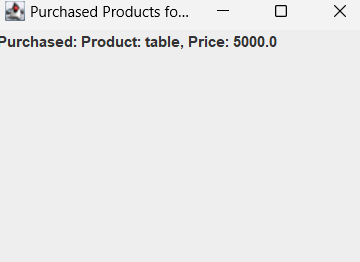
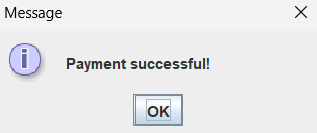
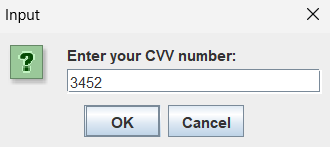


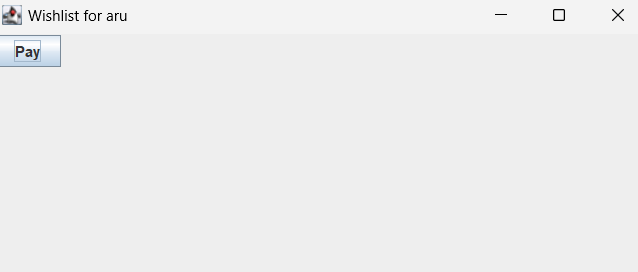


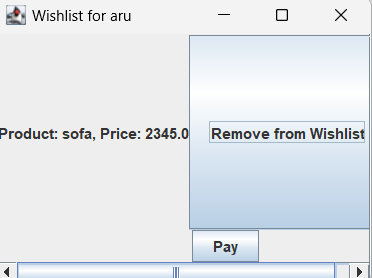
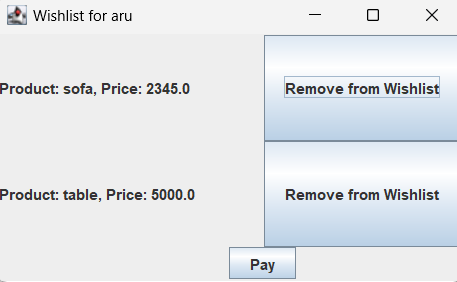


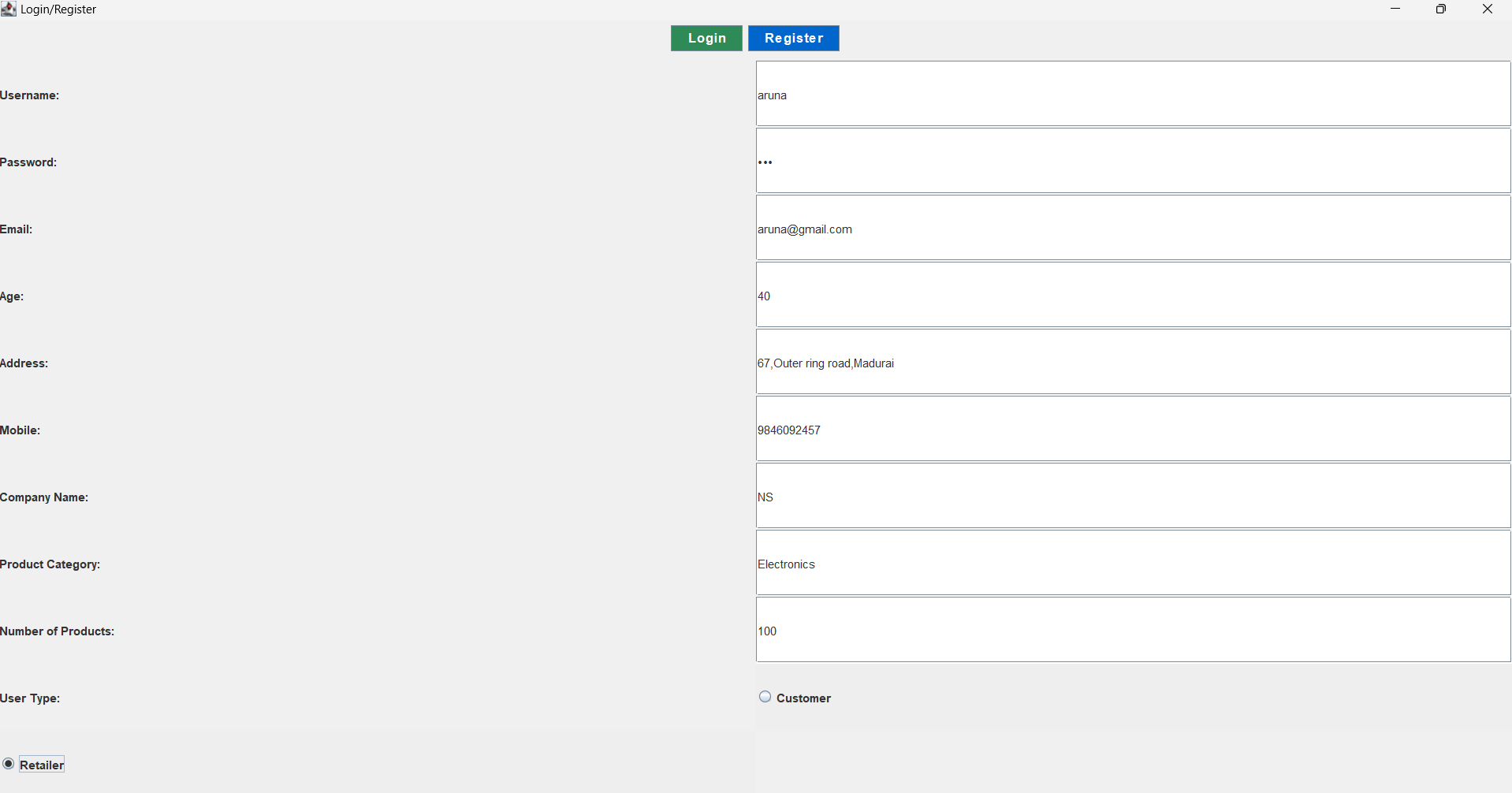
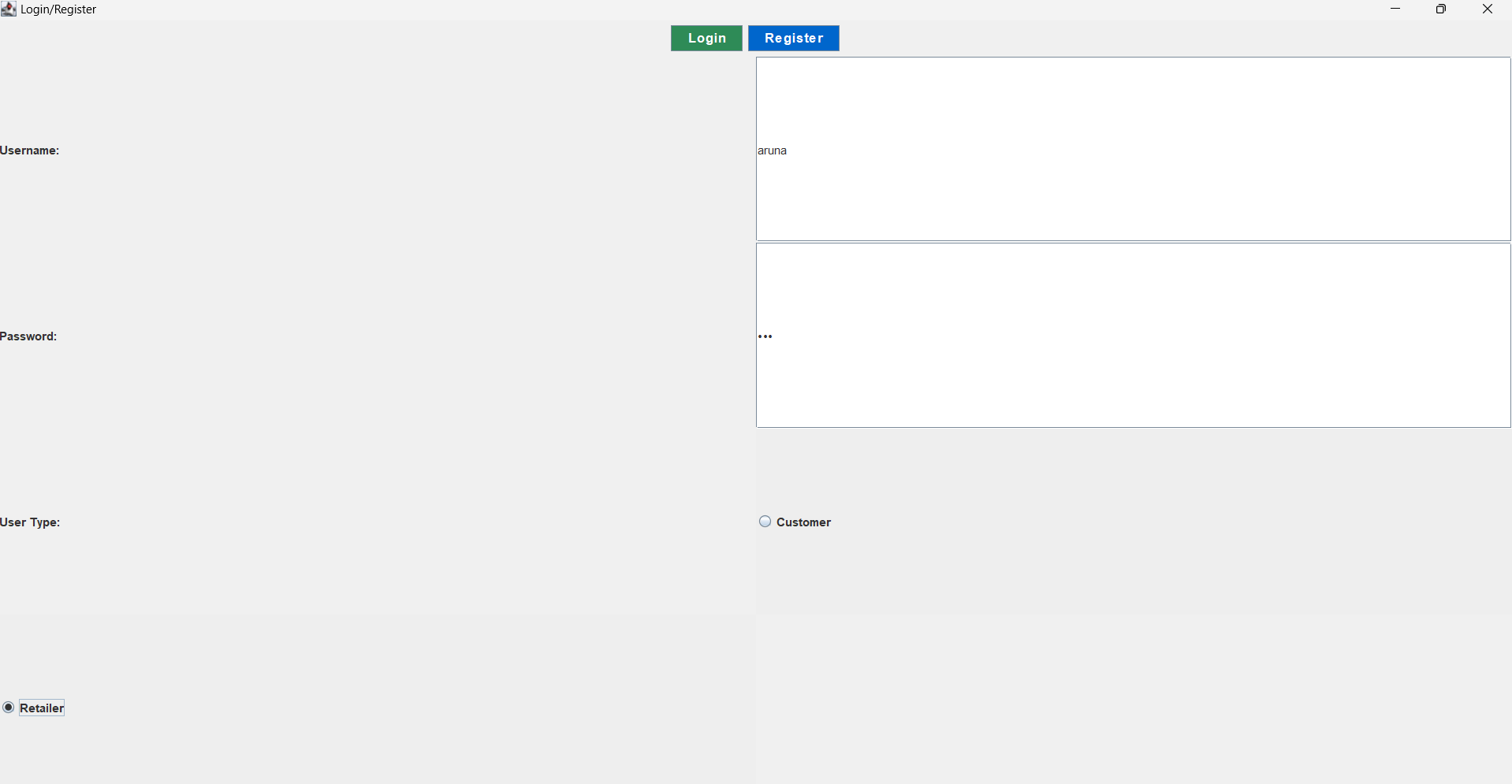
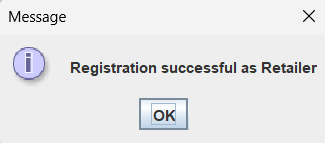


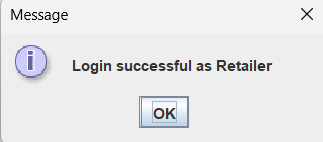


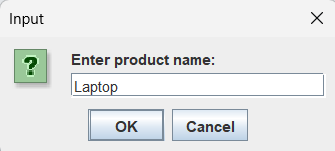
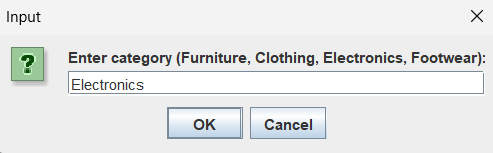
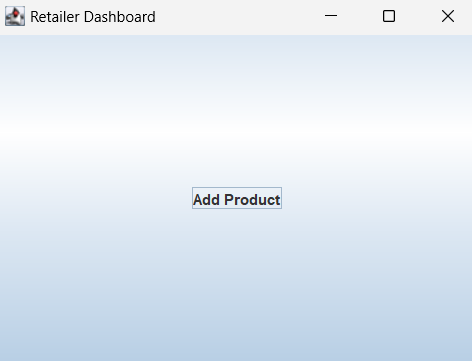


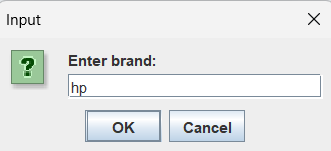


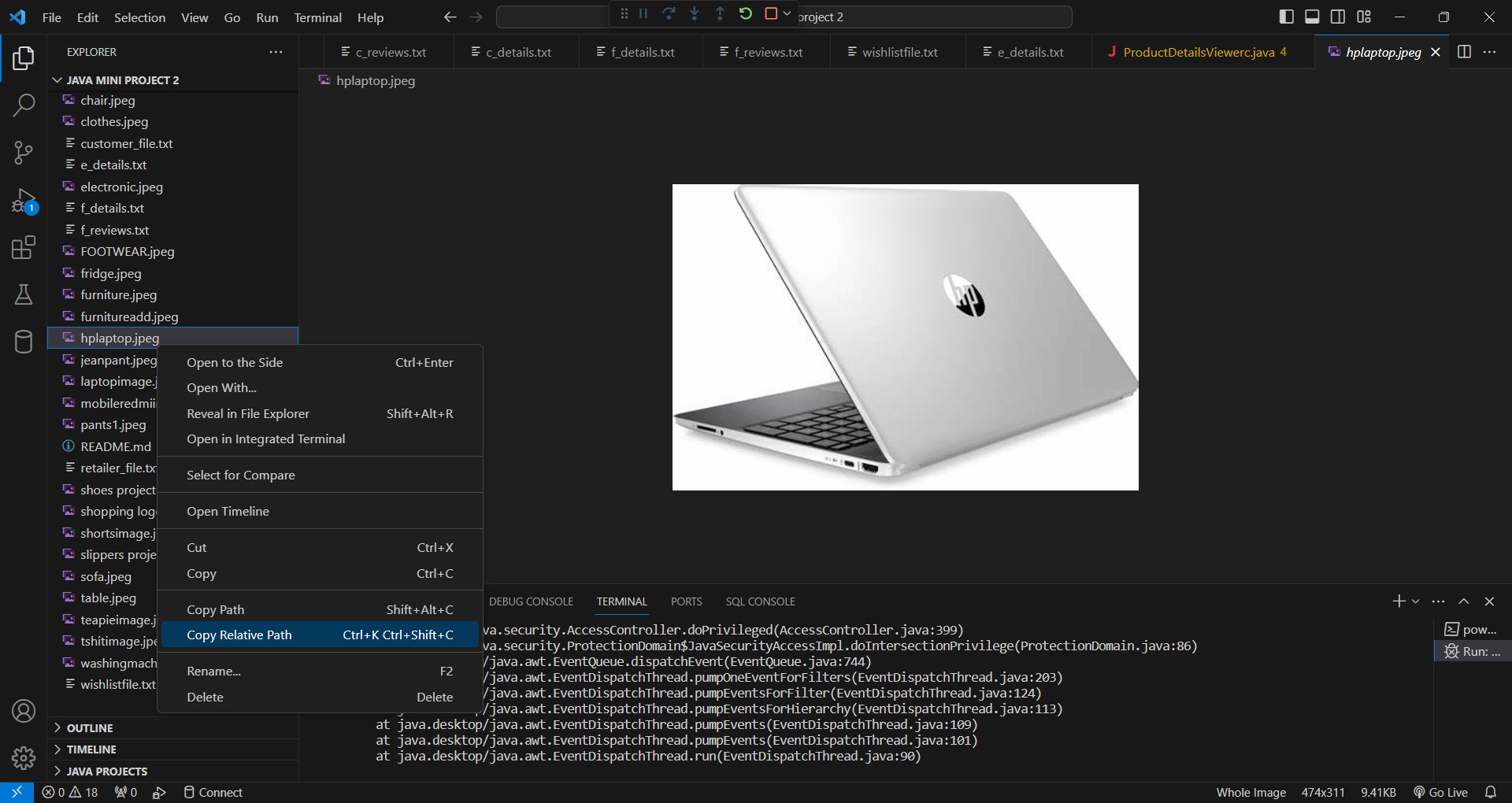
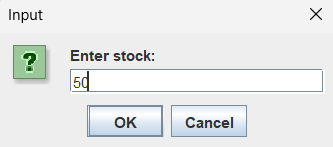
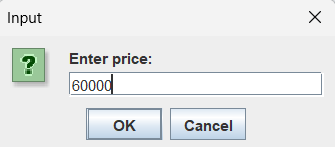


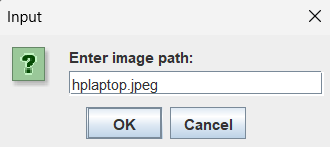


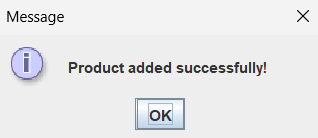












**FUTURE EXPANSION:**

We believe this project has huge scope of improvement in future. With a few touch ups and adding of more categories, this project has a lot of scope of growth. Also, by overcoming our limitations we will be able to develop a seamless interface of online shopping platform for the customers as well as retailers.

# INFERENCE:

Through this project we have learnt how to incorporate object oriented programming concepts and develop online shopping platforms. Also the front end development of the project helps to provide the users with a refreshing experience.