UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

"Thrift Store"

Submitted by:

Namita Patil	**PES1UG21CS357**
Nisha E	**PES1UG22CS822**
Mallidi Sirija Reddy	**PES1UG21CS923**
Navya Peram	**PES1UG21CS924**

6th Semester F Section

Prof. Course Instructor Name:
Ms. Bhargavi Mokashi
Assistant Professor

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

INTRODUCTION

1.1 Overview:

The thrift store project is a web application developed using Java with Eclipse as the Integrated Development Environment (IDE) and MySQL as the backend database. It aims to provide users with a convenient platform to browse, search for, and purchase second-hand items. The system includes essential functionalities such as user authentication, browsing items, searching for specific products, adding items to a shopping cart, processing orders, and tracking the status of placed orders.

1.2 Purpose:

The purpose of the project is to develop an online platform for a thrift store, providing customers with the ability to browse, purchase, and donate pre-owned items. The platform aims to promote sustainability by facilitating the reuse and recycling of goods while offering customers affordable shopping options.

1.3 Scope:

User Management:
- Registration process with required details.
- Login functionality using email and password.
- Access to order history for users to view past orders.

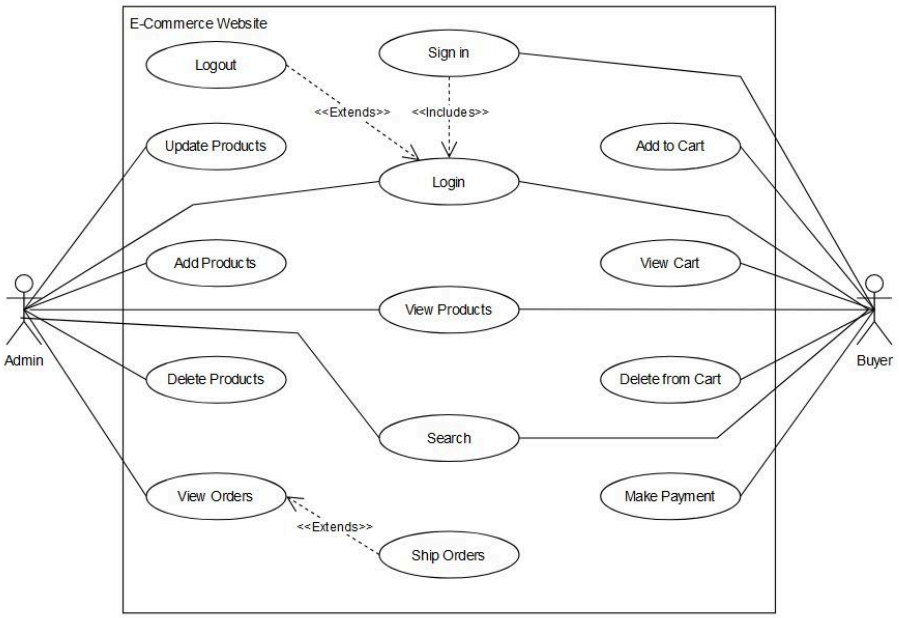
Product Management:
- Administrator capability to add, remove, and update product details: name, description, price, and quantity.
- User interface for browsing available products.
- Product details view and search functionality for users.

Order Management:
- Cart functionality for users to add products and place orders.
- Administrative access to view all orders.
- Tracking of transaction details: order IDs, transaction IDs, product IDs, quantities, and amounts.

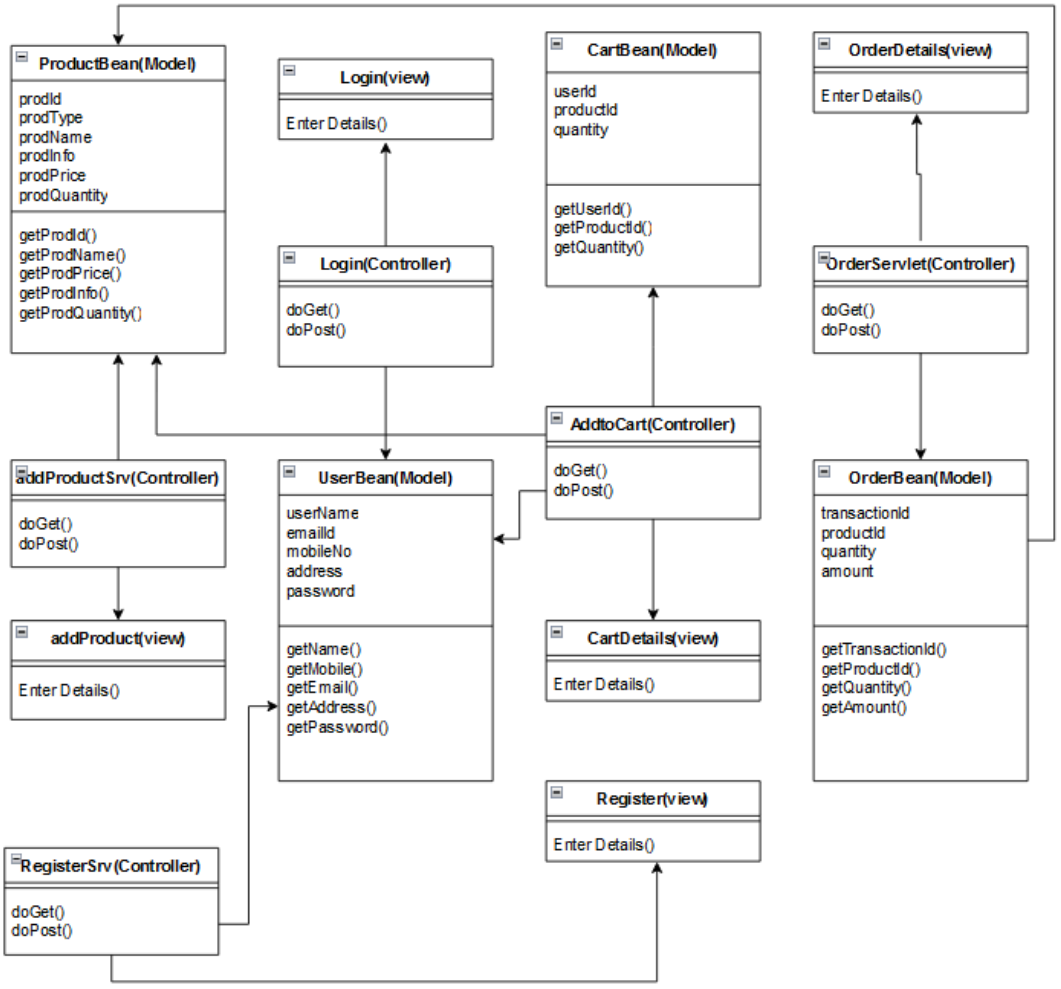
Email Notification:
- Automated email notifications sent to users upon successful shipment of orders.

REQUIREMENT ANALYSIS

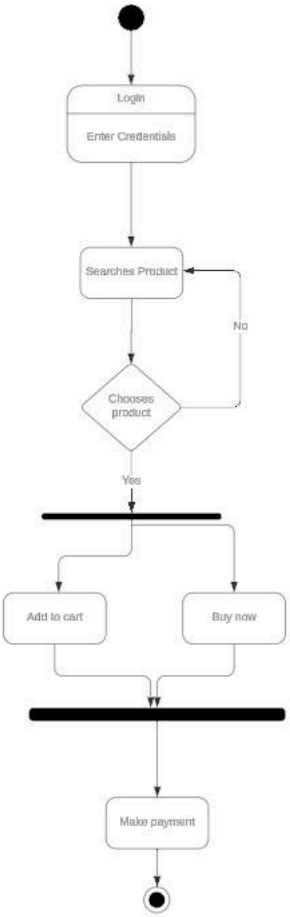
Use Case Diagram



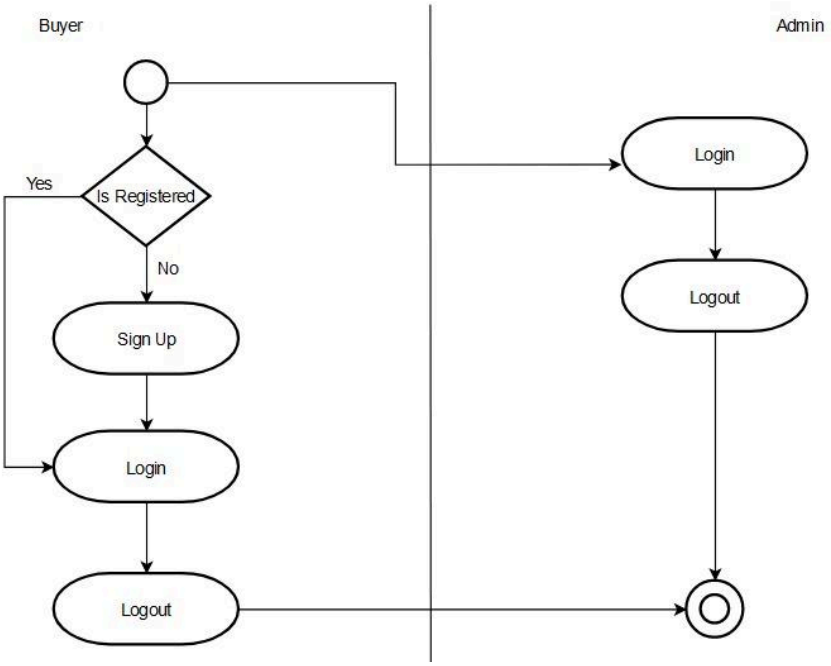
Class Diagram

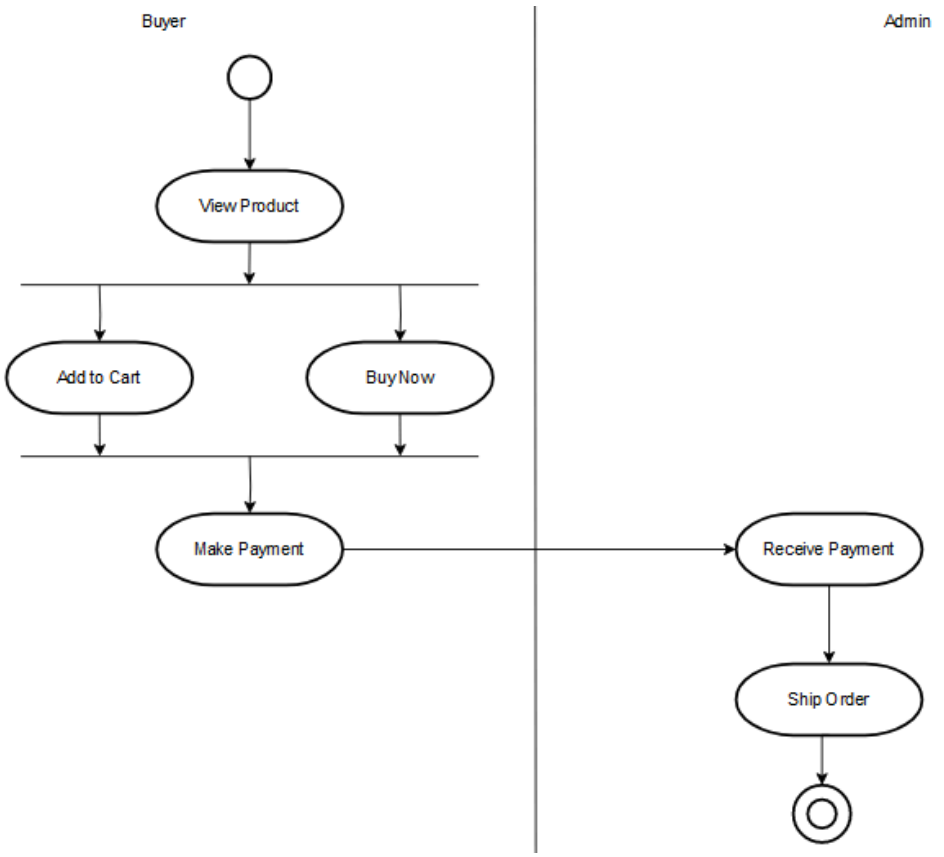


State Diagram

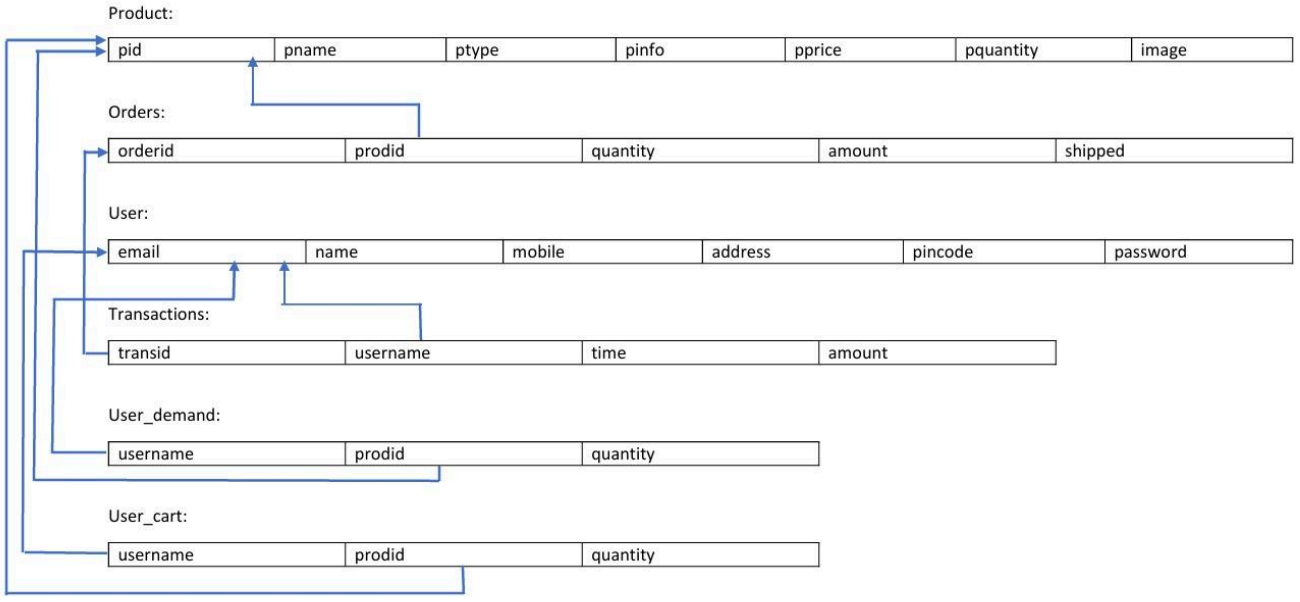


Activity Diagrams

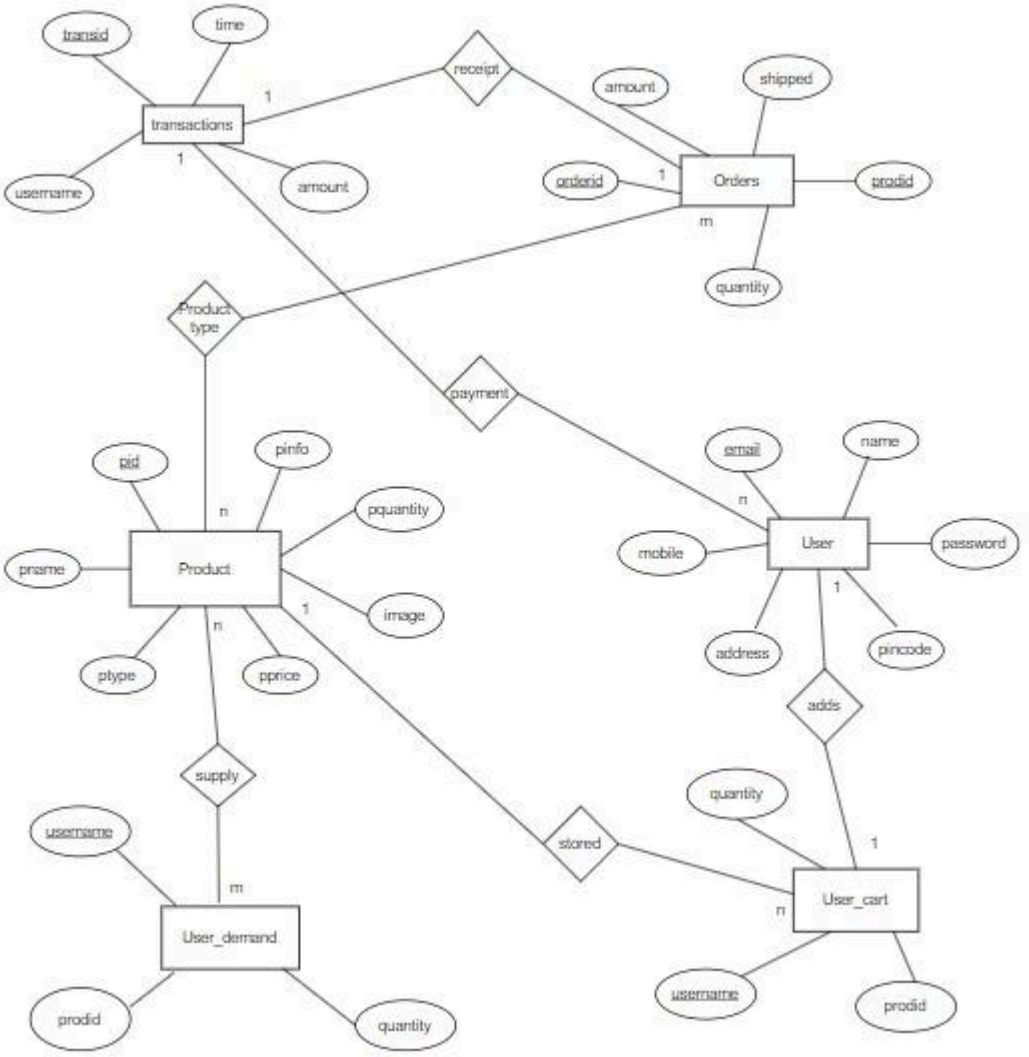




Schema Diagram



E R Diagram


transid
time
receipt
amount
shipped
transactions
1
1
username
amount
orderid
1
Orders
prodid
m
quantity
Product type
payment
pid
pinfo
email
name
n
n
pname
Product
pquantity
mobile
User
password
image
1
1
n
address
pincode
ptype
pprice
adds
supply
quantity
username
stored
1
m
n
User_cart
User_demand
username
prodid
quantity
prodid

USAGE OF DESIGN PATTERNS

Factory Pattern:
- The ProductService interface serves as a factory method abstraction, allowing the creation of instances of concrete classes like ProductServiceImpl.
- Each implementation of ProductService can act as a factory for creating objects related to product management.
- ProductServiceImpl provides concrete implementations for methods like addProduct, removeProduct, updateProduct, etc., handling product-related operations and object creation.
- In the TransactionBean class, constructors serve as factory methods for creating transaction objects, encapsulating the logic for generating unique transaction IDs and timestamps.

Builder Pattern:
- ProductBean utilises the Builder pattern, although it doesn't use a separate builder class.
- The constructor of ProductBean allows constructing instances with all necessary attributes, while not all attributes are required for every instance.
- Users can create ProductBean objects with partial information by passing only the required attributes to the constructor.
- UserBean class also follows a similar pattern, enabling construction of complex objects with optional parameters without a dedicated builder class.

Façade Pattern:
- The Facade pattern simplifies the usage of a subsystem by providing a unified interface.
- ShowImage servlet acts as a Facade, offering a simple interface for clients (web browsers) to access and display product images.
- Behind the scenes, ShowImage servlet encapsulates the complexity of retrieving and formatting images for display.

Command Pattern:
- The Command pattern encapsulates requests as objects, supporting parameterization, queuing, and undoable operations.
- ShipmentServlet servlet exemplifies the Command pattern by encapsulating the request to ship an order.

- ShipmentServlet acts as the Invoker, while OrderServiceImpl (not shown) likely serves as the Receiver, handling order shipment operations.
- This design allows requests to be queued and executed independently and supports undoing operations if necessary.

USAGE OF MVC ARCHITECTURE

Model:
- Classes such as UserBean, ProductBean, OrderBean, and TransactionBean serve as the core components of the model.
- These classes encapsulate data related to users, products, orders, and transactions, respectively.
- Service interfaces like UserService, ProductService, OrderService, and TransService define operations pertinent to users, products, orders, and transactions.
- Implementations of these interfaces (e.g., UserServiceImpl, ProductServiceImpl) handle the business logic and interaction with the underlying data storage.

View:
- JSP files, such as login.jsp, register.jsp, userHome.jsp, adminViewProduct.jsp, orderDetails.jsp, etc., form the presentation layer of the application.
- These files contain HTML markup combined with JSP tags to dynamically generate content based on the data provided by the controller.
- The view is responsible for presenting data to users in a human-readable format and capturing user interactions, such as form submissions and clicks.

Controller:
- Servlets, including LoginSrv, RegisterSrv, LogoutSrv, AddProductSrv, RemoveProductSrv, UpdateProductSrv, OrderServlet, ShipmentServlet, etc., serve as the intermediary between the model and the view.
- These servlets receive requests from clients, interpret user input, and invoke appropriate actions on the model to perform business operations.
- The controller determines the appropriate view to render based on the outcome of these operations, managing the flow of information between the model and the view.

- It handles user requests, updates the model accordingly, and selects the view to display, thus facilitating the interaction between users and the application's underlying logic.

Design Principles:

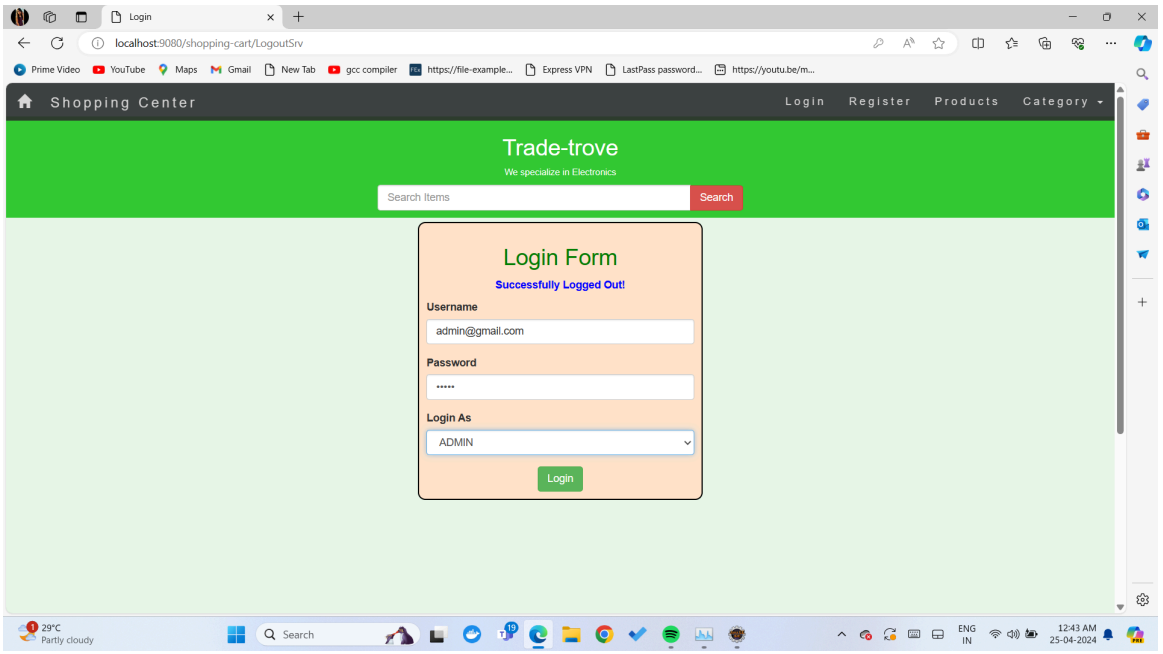
Single Responsibility Principle (SRP):
- Each class embodies a single responsibility, fostering clarity and maintainability.
- The AddtoCart servlet, for instance, exclusively manages adding products to the cart and related operations.
- Service classes (CartServiceImpl, ProductServiceImpl, DemandServiceImpl) encapsulate precise business logic, ensuring responsibility allocation.
- The ProductBean class solely represents product data, while servlets handle HTTP requests related to product management.
- The UserBean class effectively focuses on representing user data, encapsulating user-related properties and manipulation methods.

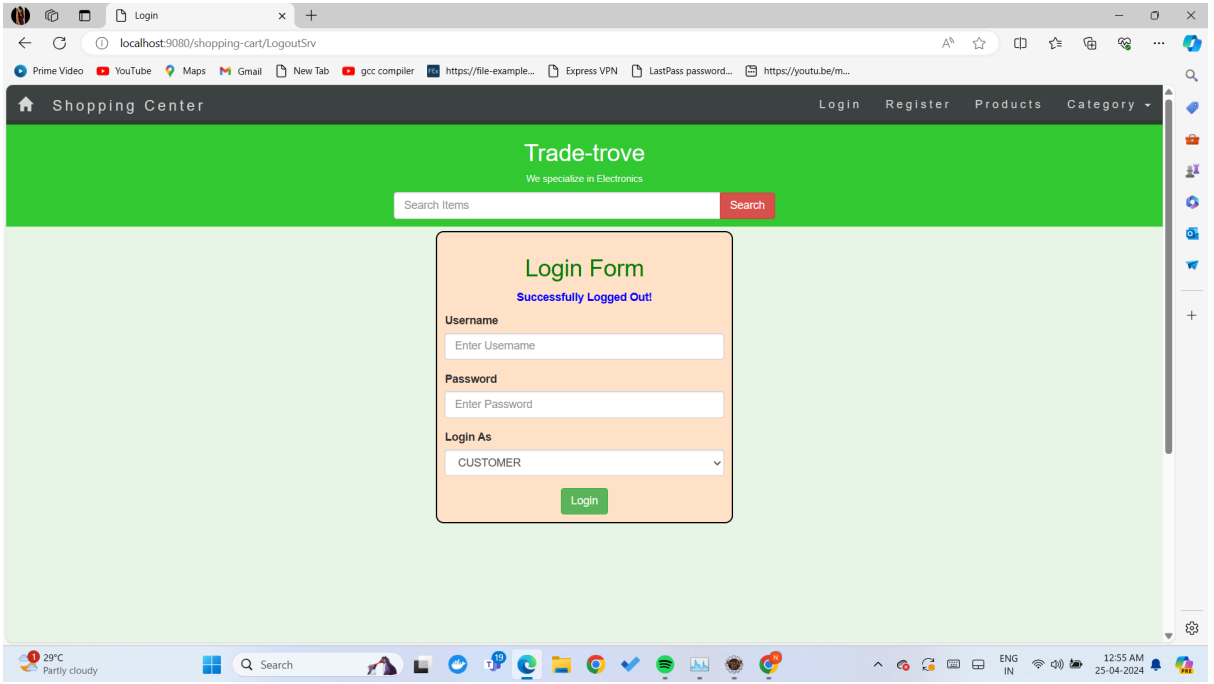
Open/Closed Principle (OCP):
- The codebase facilitates extension without modification.
- Interfaces (CartService) define contracts for various operations, enabling seamless addition of new functionalities without altering existing code.
- New functionalities can be seamlessly integrated by implementing these interfaces.
- Registration of users and credential validation can be added without modifying the existing codebase.

Dependency Inversion Principle (DIP):
- The codebase favors interfaces over concrete implementations, enhancing flexibility.
- Interfaces (e.g., ProductService) facilitate substitution of implementations, promoting adaptability.
- By depending on abstractions, the codebase adheres to DIP principles, enhancing maintainability and extensibility.

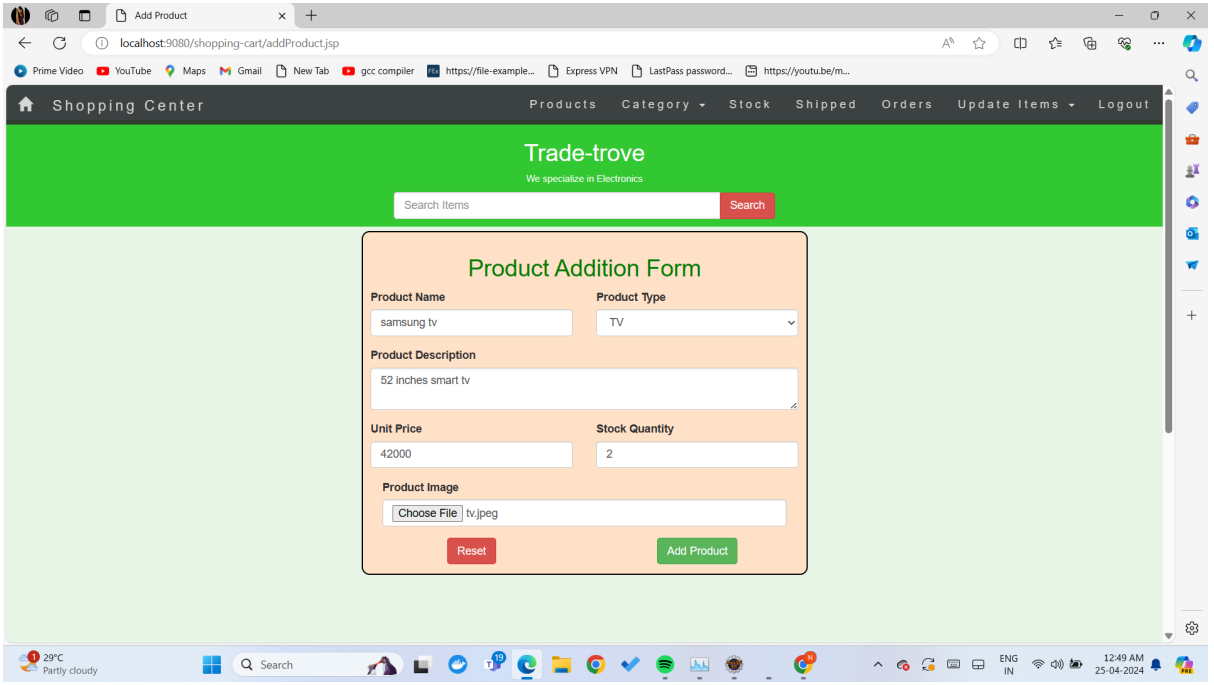
SNAPSHOTS

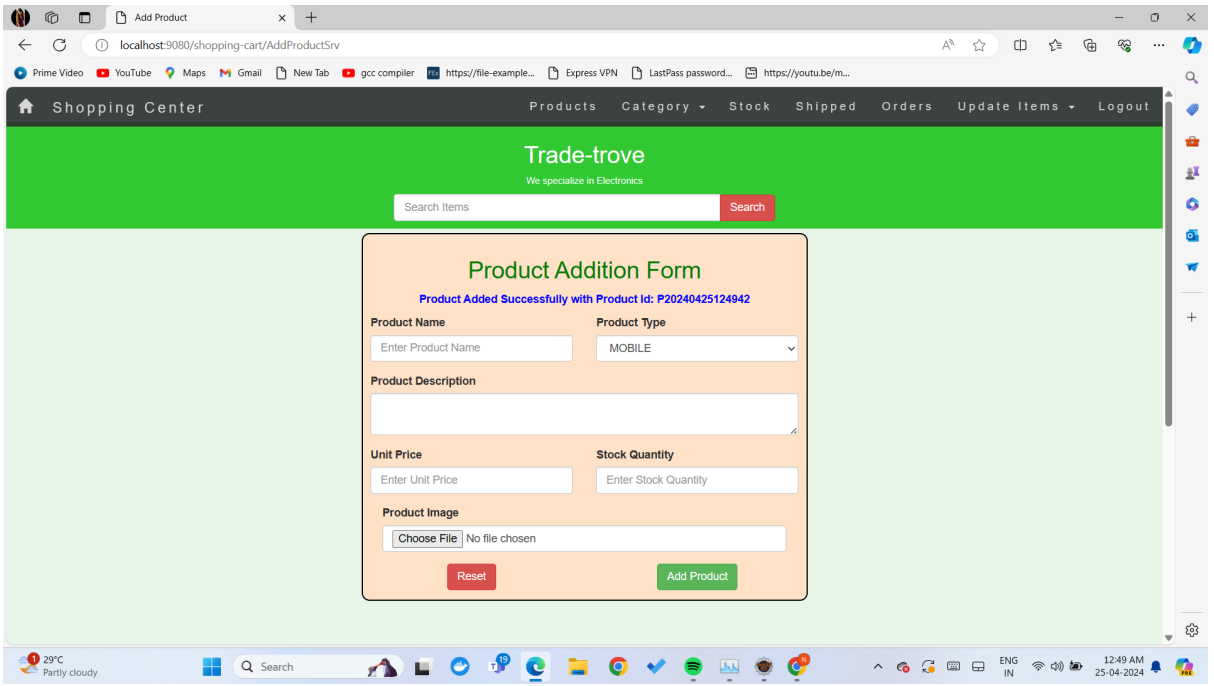
1. Login and logout as Admin:



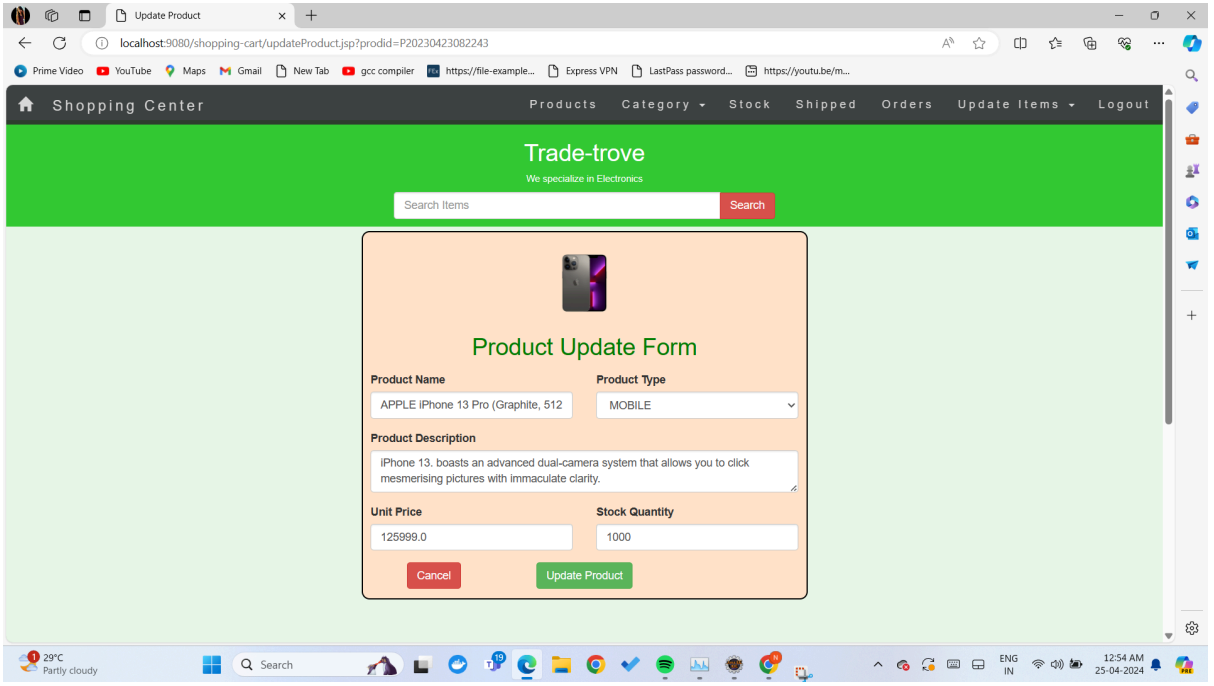


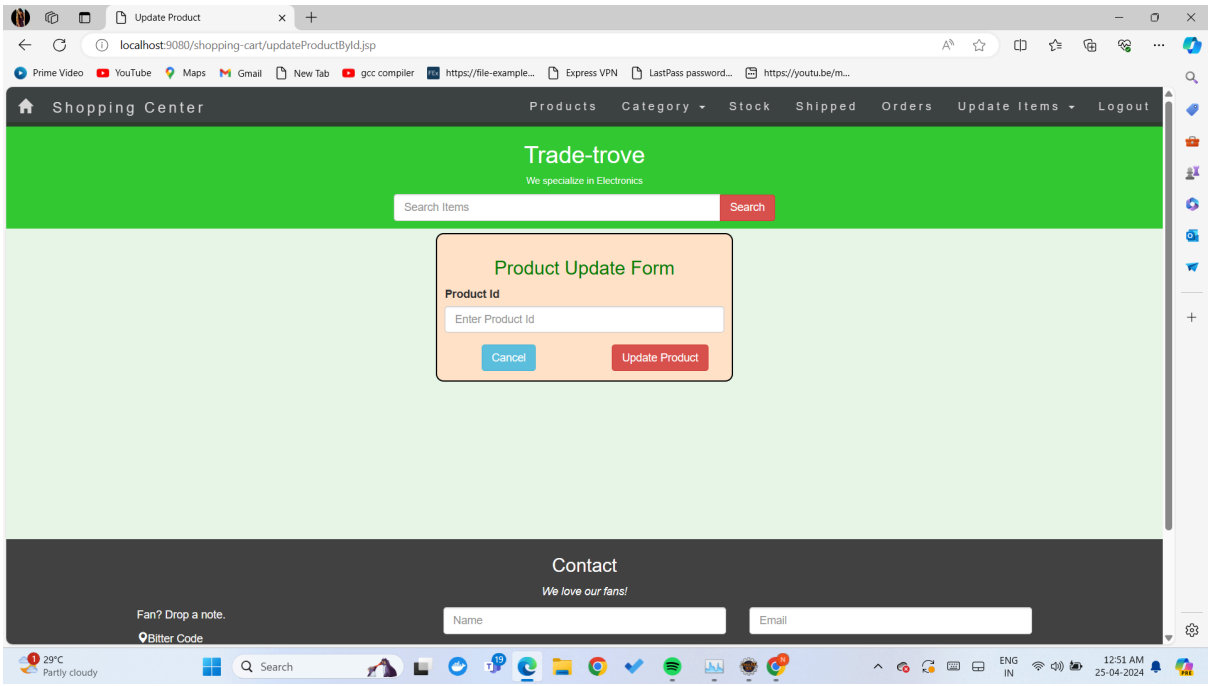
2. Add products as Admin:



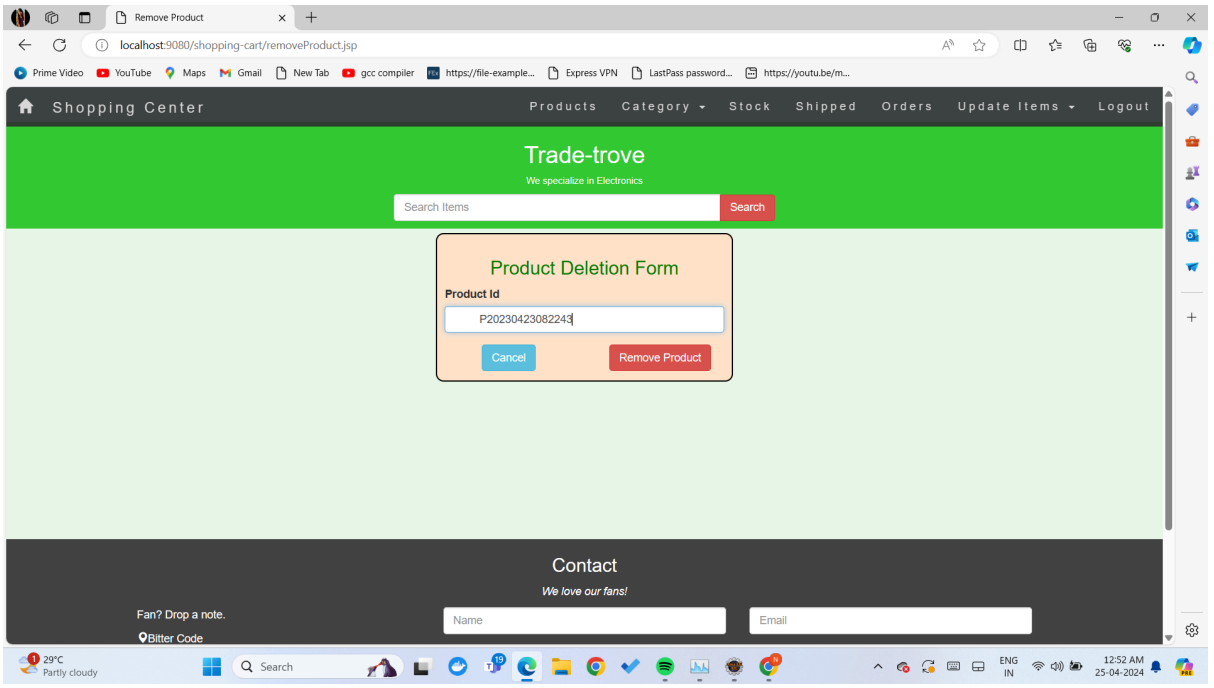


3. Updating products as Admin:

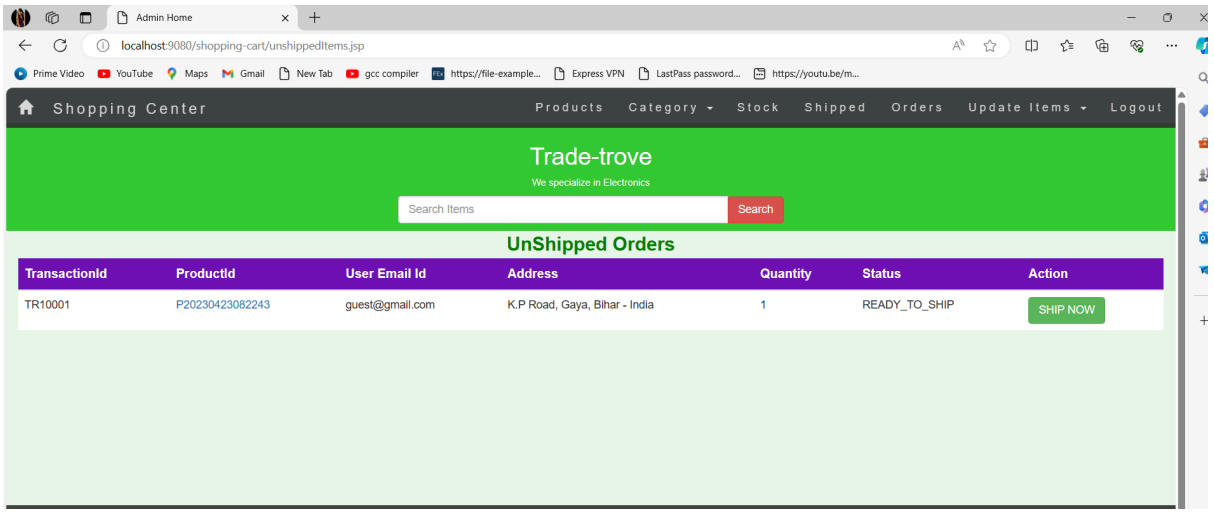

Update Product
localhost:9080/shopping-cart/updateProduct.jsp?prodid=P20230423082243
Shopping Center
Products
Category
Stock
Shipped
Orders
Update Items
Logout
Trade-trove
We specialize in Electronics
Search Items
Search
Product Update Form
Product Name
APPLE iPhone 13 Pro (Graphite, 512
Product Type
MOBILE
Product Description
iPhone 13. boasts an advanced dual-camera system that allows you to click mesmerising pictures with immaculate clarity.
Unit Price
125999.0
Stock Quantity
1000
Cancel
Update Product

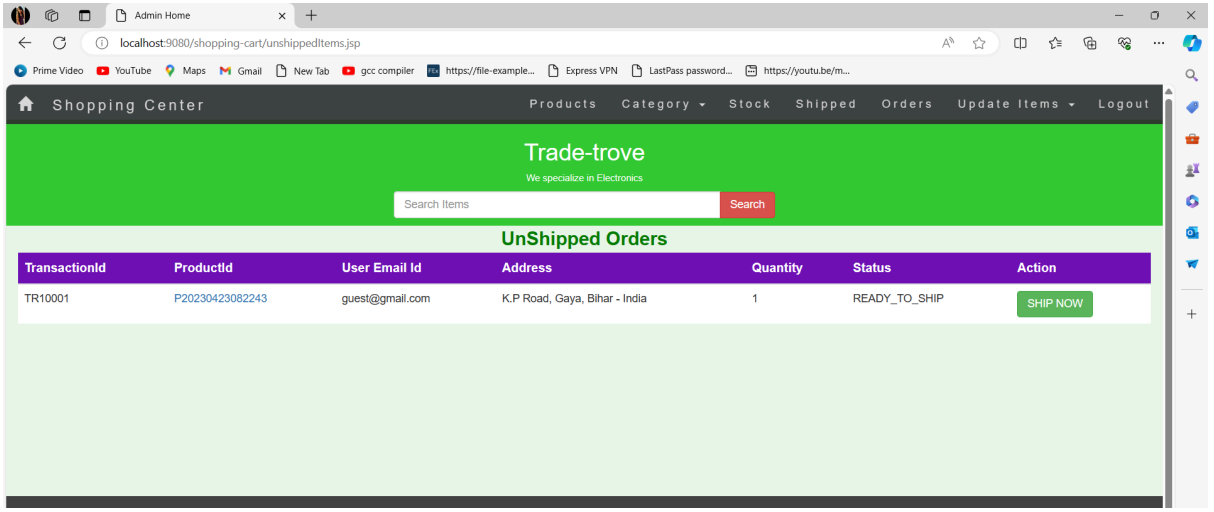

Update Product
localhost:9080/shopping-cart/updateProductById.jsp
Shopping Center
Products
Category
Stock
Shipped
Orders
Update Items
Logout
Trade-trove
We specialize in Electronics
Search Items
Search
Product Update Form
Product Id
Enter Product Id
Cancel
Update Product
Contact
We love our fans!
Fan? Drop a note.
Bitter Code
Name
Email

4. Removing Products as Admin:

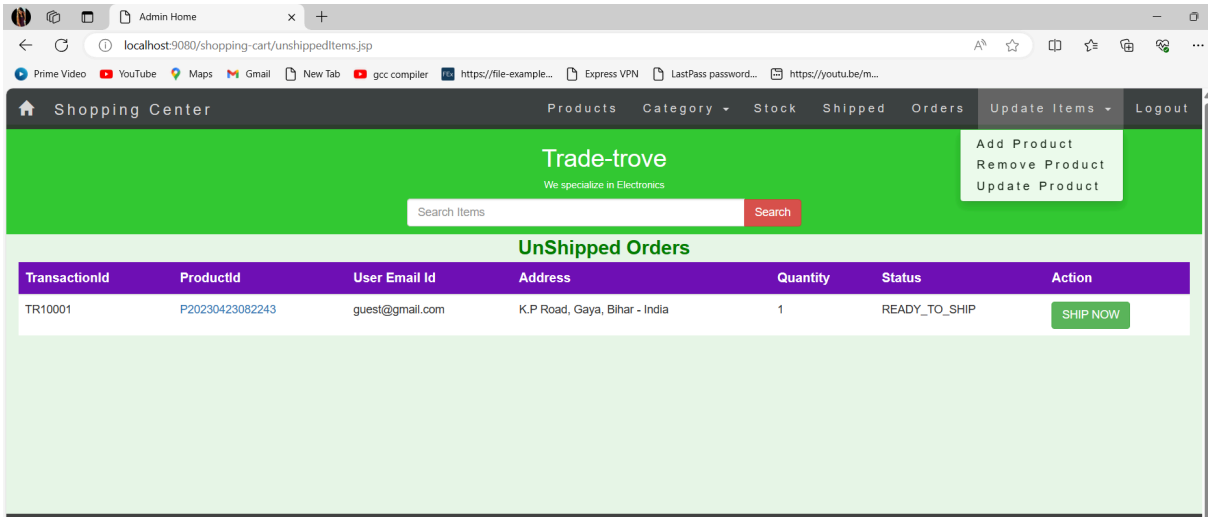


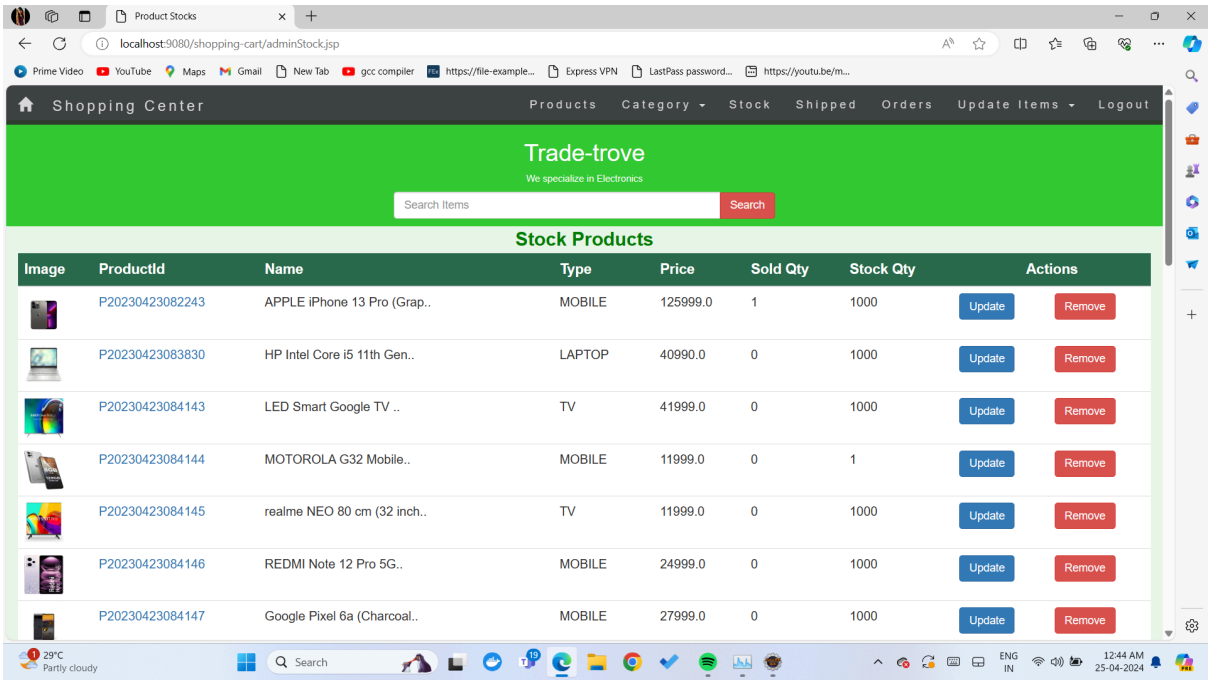
5. Tracking orders and shipments as Admin:



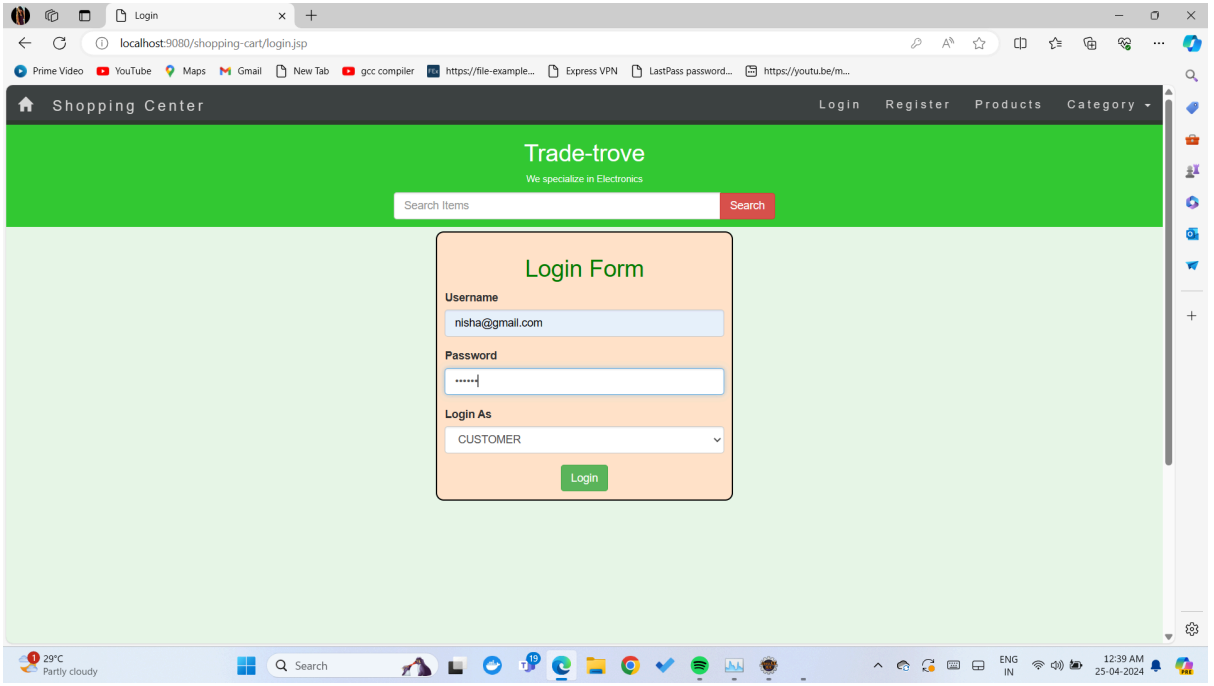


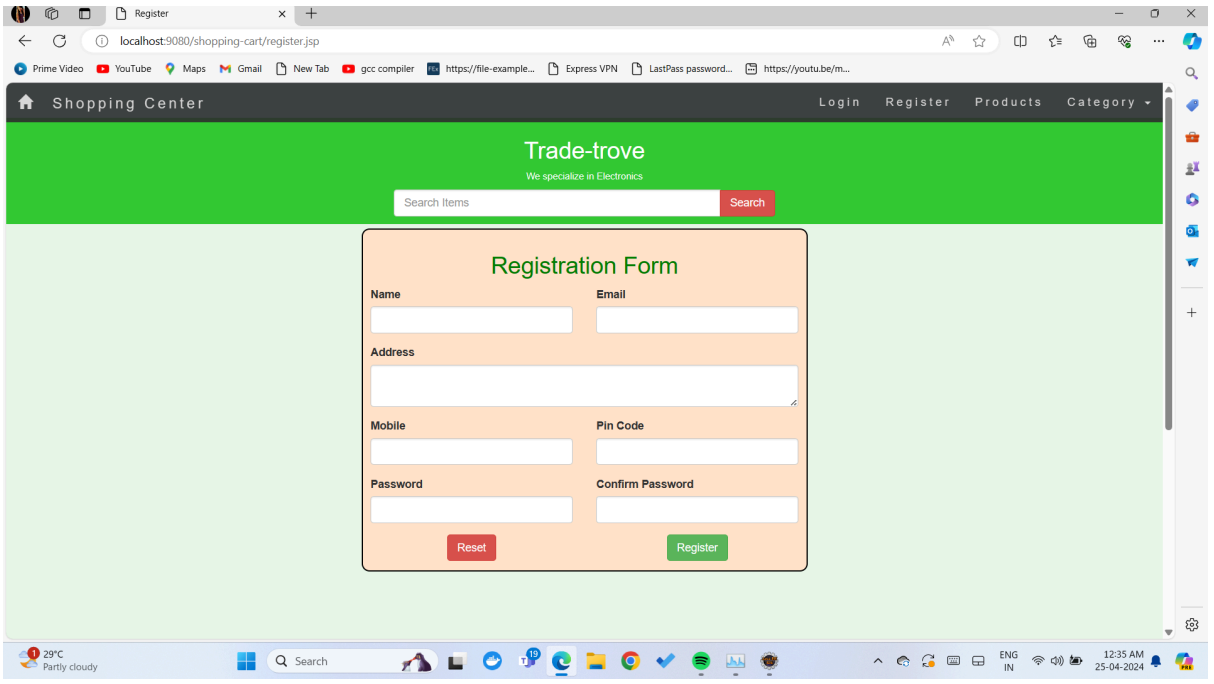
6. Updating shipment status and product stock as Admin:



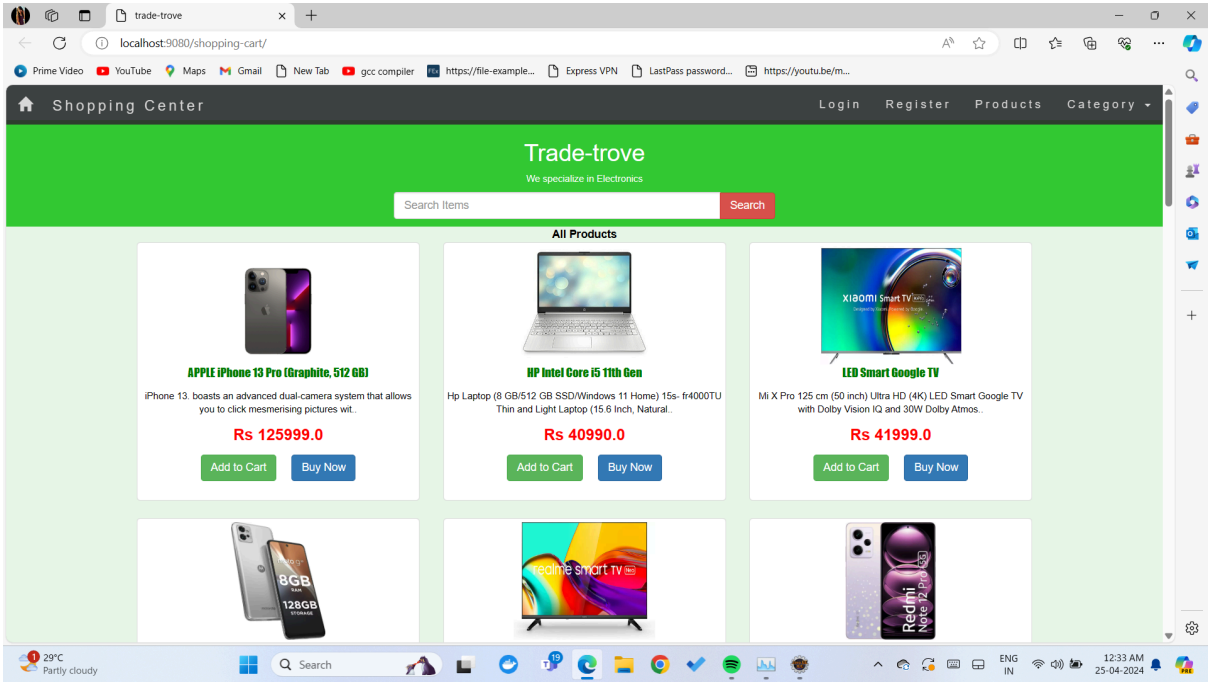


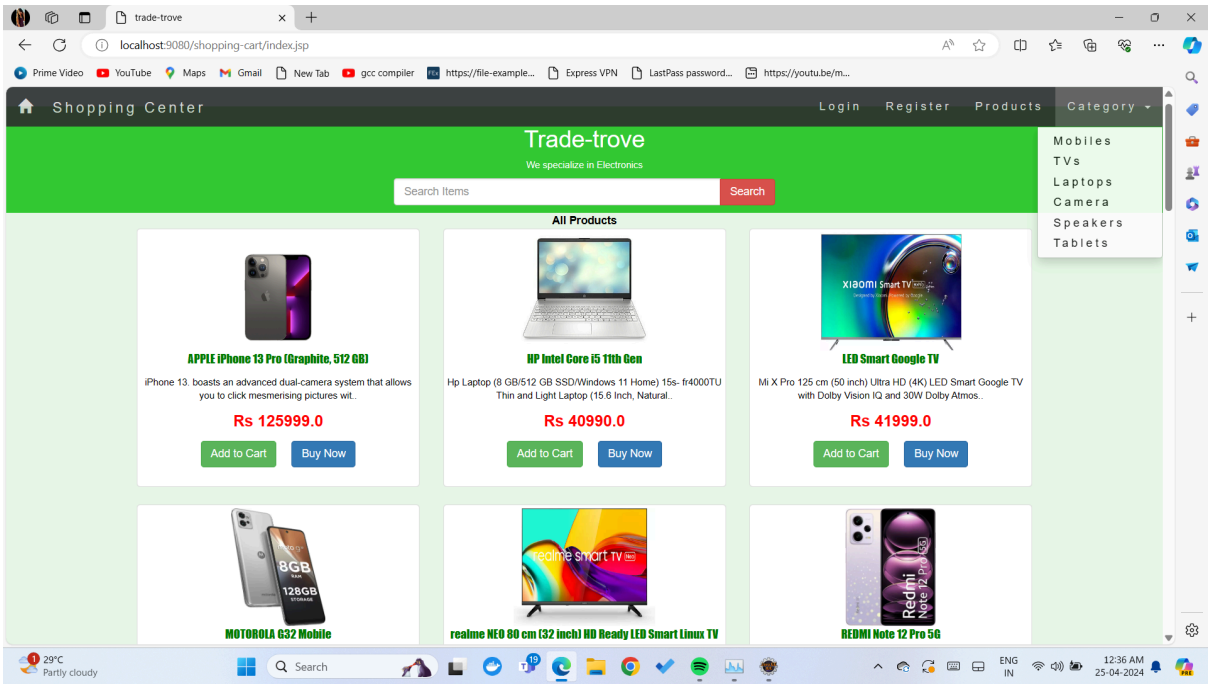
7. Login/ Register as User



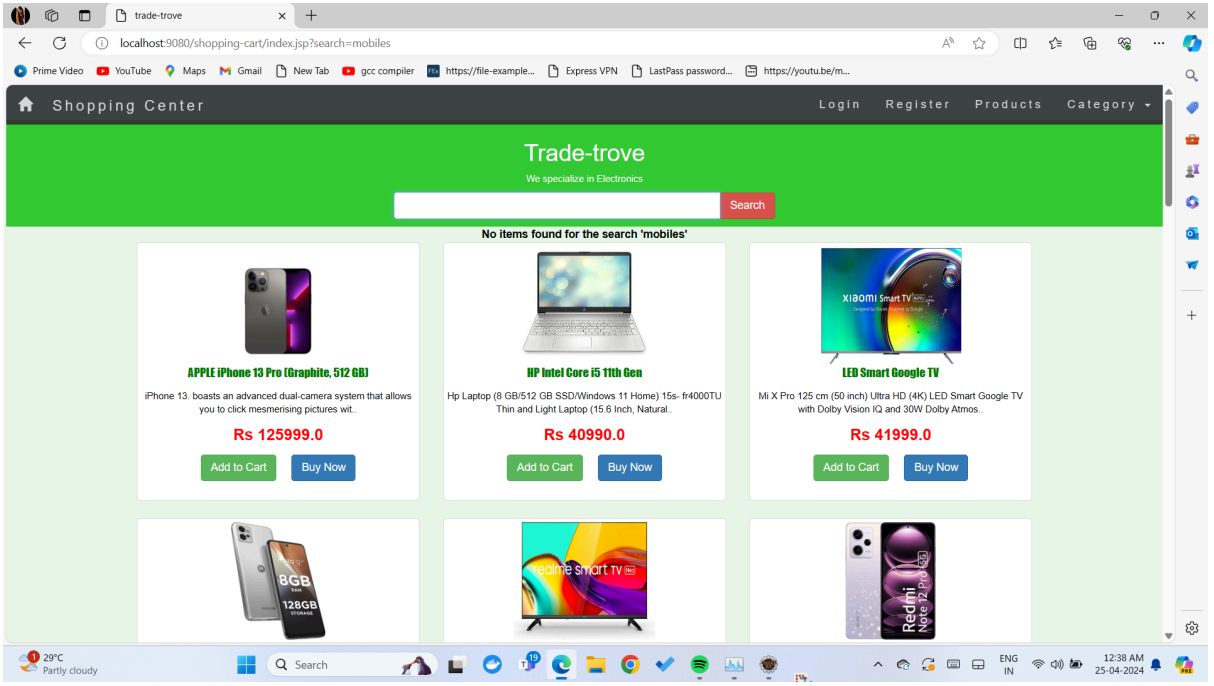


8. Navigate the homepage using categories:

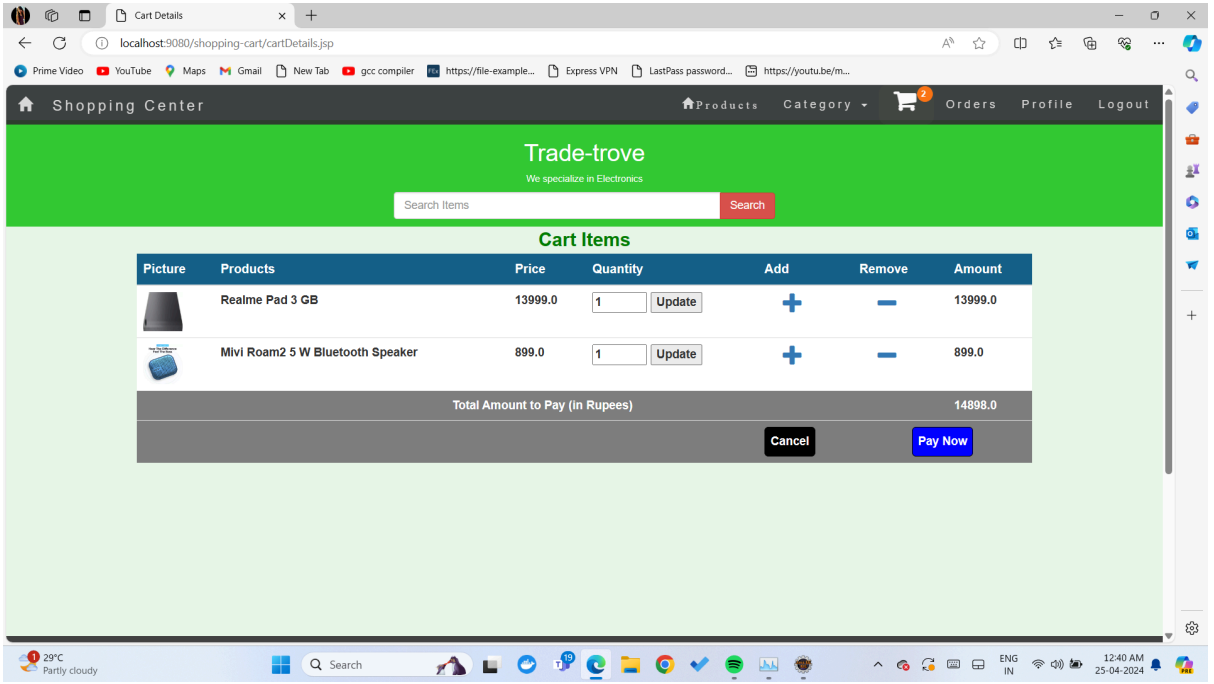


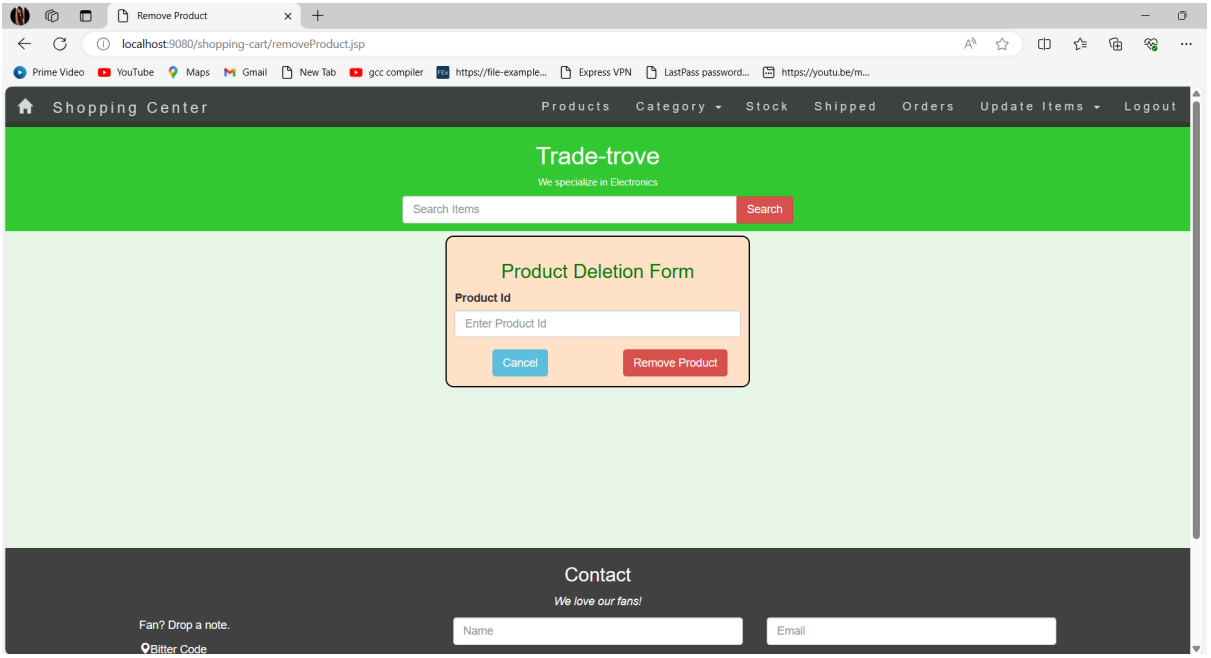


9. Search for products as User:

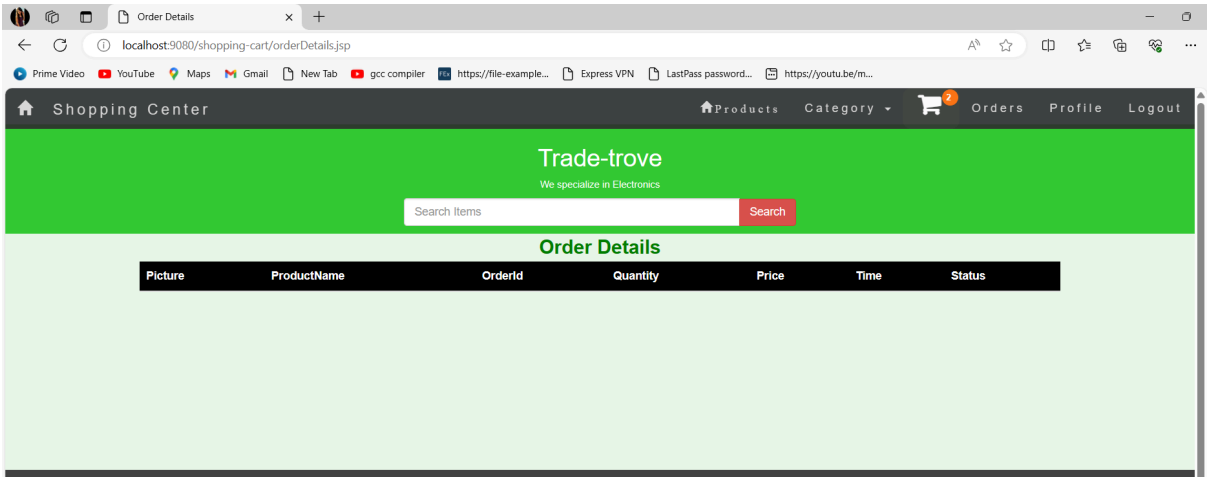


10. Add items to Cart:

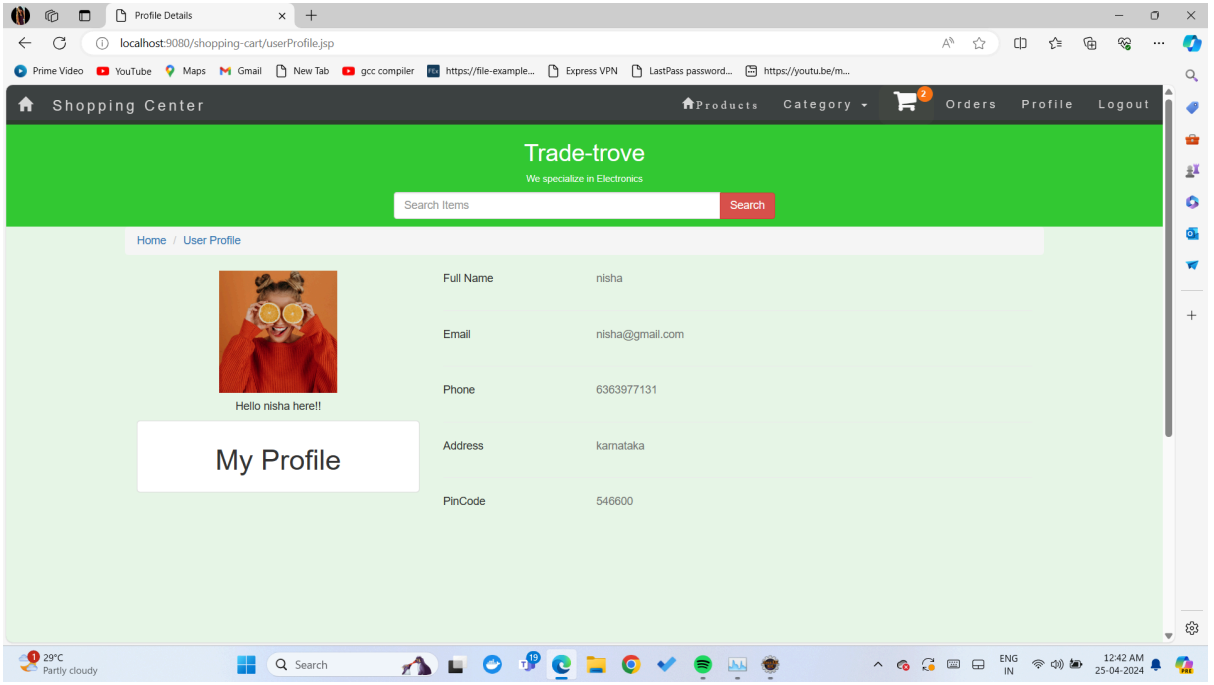




11. Track order status:



12. View Profile as User:



REPO LINK: https://github.com/NishaEBK/tradetrove

REFERENCES

- Internet & World Wide Web: How to Program Deitel, PJ Deitel.
- Bitter Code YouTube Channel.
- Database System Concepts, by Silberschatz, Sudarshan, and Korth.
- Fundamentals of Database Systems, RamezElmasri and Shamkant B. Navathe, 7th Edition. 2017, Pearson.