

**Project Group Number/Name:** Group Number 8

**Group Members:**

- Christ Rodrigues
- Yash Zaveri
- Viral Suchak
- Kevin Rodrigues
- Omkar Vaity

**Project Topic/Name:** Google Product Inventory Management System

**Tech Stack:**

- **Frontend:** React and Bootstrap
- **Backend:** Java EE, Spring, Apache Tomcat
- **Database:** MySQL
- **Database Connectivity:** Hibernate

**Functionalities Implemented:**

1. **Inventory Management Core Features:**
  - Product management (CRUD operations for products)
  - Employee management with different access levels based on designation
  - Buyer/Customer management
  - Purchase order processing
  - Invoice generation and management
2. **Technical Architecture:**
  - Spring Boot backend with REST controllers
  - Hibernate for database interactions using SessionFactory
  - React frontend for user interface
  - Authentication system for employee login
3. **Document Generation:**
  - PDF generation for invoices using iText library
  - The system can create and format invoice documents for purchase orders
4. **Business Logic:**
  - Inventory tracking and stock management
  - Purchase order processing including payment tracking
  - Employee role-based access control (Manager vs regular employees)

## Design Patterns implemented:

### 1) Strategy Design Pattern

```
1 package edu.neu.csye7374.designpattern.strategy;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.server.ResponseStatusException;
5
6 import com.inventory.model.Buyer;
7 import com.inventory.repository.BuyerRepository;
8
9 public class BuyerStrategy implements StrategyAPI {
10
11     private int id;
12     private BuyerRepository buyerRepo;
13     private Buyer buyer;
14
15     public BuyerStrategy(BuyerRepository buyerRepo, Buyer buyer) {
16         this.buyerRepo = buyerRepo;
17         this.buyer = buyer;
18     }
19
20     public BuyerStrategy(BuyerRepository buyerRepo, int id) {
21         this.id = id;
22         this.buyerRepo = buyerRepo;
23     }
24
25     @Override
26     public void add() {
27         if(buyerRepo.companyExists(buyer.getCompanyName()))
28             throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Company already exists");
29     }
30 }
```

```
1 package edu.neu.csye7374.designpattern.strategy;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.server.ResponseStatusException;
5
6 import com.inventory.model.Employee;
7 import com.inventory.repository.EmployeeRepository;
8
9 public class EmployeeStrategy implements StrategyAPI{
10
11     private EmployeeRepository employeeRepo;
12     private int id;
13     private Employee employee;
14
15     public EmployeeStrategy(EmployeeRepository employeeRepo, Employee employee) {
16         this.employeeRepo = employeeRepo;
17         this.employee = employee;
18     }
19
20     public EmployeeStrategy(EmployeeRepository employeeRepo, int id) {
21         this.employeeRepo = employeeRepo;
22         this.id = id;
23     }
24
25     @Override
26     public void add() {
27         if(employeeRepo.usernameExists(employee.getUsername()))
28             throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Username already exists");
29     }
30 }
```

Files

main

Go to file

designpattern

command

state

strategy

BuyerStrategy.java

EmployeeStrategy.java

InventoryStrategy.java

InvoiceStrategy.java

OrderStrategy.java

ProductPOStrategy.java

ProductStrategy.java

StrategyAPI.java

model

repository

viral-suchak Buyer Side

Code Blame 22 lines (16 loc) · 386 Bytes

```
1 package edu.neu.csye7374.designpattern.strategy;
2
3 public class InventoryStrategy {
4
5     private StrategyAPI strategy;
6
7     public InventoryStrategy(StrategyAPI strategy) {
8         this.strategy = strategy;
9     }
10
11     public void executeAdd() {
12         strategy.add();
13     }
14
15     public void executeDelete() {
16         strategy.delete();
17     }
18
19     public void executeUpdate() {
20         strategy.update();
21     }
22 }
```

Files

main

Go to file

designpattern

command

state

strategy

BuyerStrategy.java

EmployeeStrategy.java

InventoryStrategy.java

InvoiceStrategy.java

OrderStrategy.java

ProductPOStrategy.java

ProductStrategy.java

StrategyAPI.java

model

repository

Driver.java

Code Blame 65 lines (52 loc) · 1.67 KB

```
1 package edu.neu.csye7374.designpattern.strategy;
2
3 import java.text.SimpleDateFormat;
4 import java.util.Date;
5
6 import com.inventory.designpattern.facade.PDFGen;
7 import com.inventory.model.Invoice;
8 import com.inventory.model.PurchaseOrder;
9 import com.inventory.repository.InvoiceRepository;
10 import com.inventory.repository.OrderRepository;
11
12 public class InvoiceStrategy implements StrategyAPI{
13
14     private InvoiceRepository invoiceRepo;
15     private int id;
16     private Invoice invoice;
17     private OrderRepository orderRepo;
18
19     public InvoiceStrategy(InvoiceRepository invoiceRepo, Invoice invoice) {
20         super();
21         this.invoiceRepo = invoiceRepo;
22         this.invoice = invoice;
23     }
24
25     public InvoiceStrategy(InvoiceRepository invoiceRepo, int id , OrderRepository orderRepo) {
26         this.invoiceRepo = invoiceRepo;
27         this.id = id;
28         this.orderRepo = orderRepo;
```

Files

main

Go to file

designpattern

command

state

strategy

model

repository

Driver.java

BuyerStrategy.java

EmployeeStrategy.java

InventoryStrategy.java

InvoiceStrategy.java

OrderStrategy.java

ProductPOStrategy.java

ProductStrategy.java

StrategyAPI.java

Code

Blame

94 lines (71 loc) · 2.69 KB

Raw

```
1 package edu.neu.csye7374.designpattern.strategy;
2
3 import java.util.List;
4
5 import com.inventory.InventoryCartAPI;
6 import com.inventory.designpattern.decorator.CustomDecorator;
7 import com.inventory.designpattern.decorator.Product;
8 import com.inventory.model.ProductPO;
9 import com.inventory.model.PurchaseOrder;
10 import com.inventory.repository.ProductPORepository;
11 import com.inventory.repository.ProductRepository;
12 import com.inventory.repository.OrderRepository;
13 import com.inventory.designpattern.state.StockAlert;
14 import com.inventory.designpattern.state.State;
15 import com.inventory.designpattern.state.StockUpdate;
16
17
18 public class OrderStrategy implements StrategyAPI{
19
20     private OrderRepository orderRepo;
21     private ProductPORepository productPORepo;
22     private ProductRepository productRepo;
23     private int id;
24     private PurchaseOrder purchaseOrder;
25     private PurchaseOrder insertedPO;
26
27     public OrderStrategy(OrderRepository orderRepo,
```

Files

main

Go to file

designpattern

command

state

strategy

model

repository

Driver.java

BuyerStrategy.java

EmployeeStrategy.java

InventoryStrategy.java

InvoiceStrategy.java

OrderStrategy.java

ProductPOStrategy.java

ProductStrategy.java

StrategyAPI.java

Code

Blame

41 lines (31 loc) · 921 Bytes

Raw

```
1 package edu.neu.csye7374.designpattern.strategy;
2
3 import com.inventory.model.ProductPO;
4 import com.inventory.repository.ProductPORepository;
5
6 public class ProductPOStrategy implements StrategyAPI{
7
8     private ProductPORepository productPORepo;
9     private int id;
10     private ProductPO productPO;
11
12     public ProductPOStrategy(ProductPORepository productPORepo, int id) {
13         super();
14         this.productPORepo = productPORepo;
15         this.id = id;
16     }
17
18     public ProductPOStrategy(ProductPORepository productPORepo, ProductPO productPO) {
19         super();
20         this.productPORepo = productPORepo;
21         this.productPO = productPO;
22     }
23
24     @Override
25     public void add() {
26         productPORepo.save(productPO);
27     }
28 }
```

Files

main

Go to file

designpattern

command

state

strategy

BuyerStrategy.java

EmployeeStrategy.java

InventoryStrategy.java

InvoiceStrategy.java

OrderStrategy.java

ProductPOStrategy.java

ProductStrategy.java

StrategyAPI.java

model

repository

Driver.java

CodeBlame

53 lines (42 loc) · 1.35 KB

1package edu.neu.csye7374.designpattern.strategy;

2

3import com.inventory.model.Product;

4import com.inventory.designpattern.observer.Notify;

5import com.inventory.designpattern.observer.UpdateBuyers;

6import com.inventory.designpattern.observer.UpdateDB;

7import com.inventory.repository.BuyerRepository;

8import com.inventory.repository.ProductRepository;

9

10public class ProductStrategy implements StrategyAPI{

11

12 private ProductRepository productRepo;

13 private int id;

14 private Product product;

15 private BuyerRepository buyerRepo;

16

17 public ProductStrategy(ProductRepository productRepo, int id) {

18 this.productRepo = productRepo;

19 this.id = id;

20 }

21

22 public ProductStrategy(ProductRepository productRepo, Product product) {

23 this.productRepo = productRepo;

24 this.product = product;

25 }

26

27 public ProductStrategy(ProductRepository productRepo, Product product, BuyerRepository buyerRepo) {

28 this.productRepo = productRepo;

Files

main

Go to file

designpattern

command

state

strategy

BuyerStrategy.java

EmployeeStrategy.java

InventoryStrategy.java

InvoiceStrategy.java

OrderStrategy.java

ProductPOStrategy.java

ProductStrategy.java

StrategyAPI.java

final-project-group\_8 / src / main / java / edu / neu / csye7374 / designpattern / strategy / StrategyAPI.java

viral-suchak

Strategy Design pattern

CodeBlame

11 lines (6 loc) · 166 Bytes

1package edu.neu.csye7374.designpattern.strategy;

2

3public interface StrategyAPI{

4

5 public void add();

6

7 public void update();

8

9 public void delete();

10

11}

## 2) Observer Design Pattern

+ Q

t

374

kevin-rodrigues-git observer pattern 95e9136 · 1 minute ago History

Code Blame 26 lines (18 loc) · 566 Bytes Raw Copy Download Edit View

```
1 package edu.neu.csye7374.designpattern.observer;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import com.inventory.model.Product;
7
8 public class Notify {
9
10     private List<ObserverAPI> subscribers = new ArrayList<ObserverAPI>();
11
12     public void setState(Product product) {
13         notifyAllSubscribers(product);
14     }
15
16     public void attach(ObserverAPI sub){
17         subscribers.add(sub);
18     }
19
20     public void notifyAllSubscribers(Product product){
21         for (ObserverAPI observer : subscribers) {
```

main + Q

Go to file t

> .idea

> .settings

> src/main/java/edu/neu/csye7374

> controller

> designpattern

> command

> facade

> observer

Notify.java

NotifyBuyers.java

ObserverAPI.java

UpdateBuyers.java

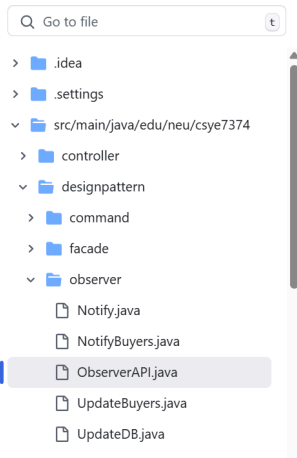
UpdateDB.java

> state

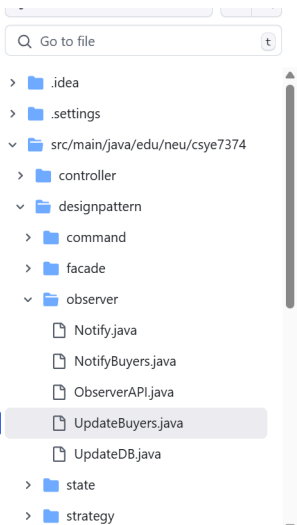
kevin-rodrigues-git observer pattern 95e9136 · 1 minute ago History

Code Blame 32 lines (25 loc) · 943 Bytes Raw Copy Download Edit View

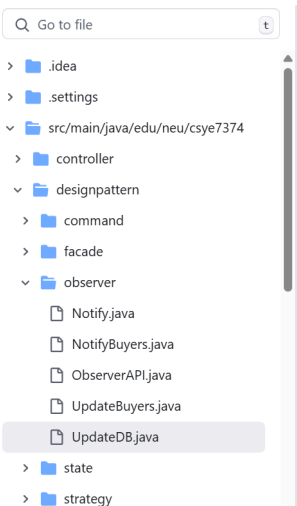
```
1 package edu.neu.csye7374.designpattern.observer;
2
3 import java.util.List;
4
5 import com.inventory.designpattern.facade.SendMessage;
6 import com.inventory.model.Buyer;
7 import com.inventory.model.Product;
8 import com.inventory.repository.BuyerRepository;
9
10 public class NotifyBuyers extends Buyer {
11
12     private List<Buyer> buyers;
13     private Product product;
14     private BuyerRepository buyerRepo;
15
16     public NotifyBuyers(Product product, BuyerRepository buyerRepo) {
17         this.product = product;
18         this.buyerRepo = buyerRepo;
19     }
20     public void notifyAllBuyers() {
21         StringBuilder sb = new StringBuilder();
```



```
Code Blame 11 lines (7 loc) · 254 Bytes
1 package edu.neu.csye7374.designpattern.observer;
2
3 import com.inventory.model.Product;
4 import com.inventory.designpattern.observer.Notify;
5
6 public abstract class ObserverAPI {
7
8     protected Notify notify;
9
10    public abstract void update(Product product);
11 }
```



```
Code Blame 24 lines (17 loc) · 564 Bytes
1 package edu.neu.csye7374.designpattern.observer;
2
3 import com.inventory.model.Product;
4 import com.inventory.repository.BuyerRepository;
5
6 public class UpdateBuyers extends ObserverAPI{
7
8     private BuyerRepository buyerRepo;
9
10    public UpdateBuyers(Notify notify, BuyerRepository buyerRepo) {
11        this.notify = notify;
12        this.notify.attach(this);
13        this.buyerRepo = buyerRepo;
14        // TODO Auto-generated constructor stub
15    }
16
17    @Override
18    public void update(Product product) {
19        NotifyBuyers notify = new NotifyBuyers(product, buyerRepo);
20        notify.notifyAllBuyers();
21    }
22 }
```



```
Code Blame 27 lines (19 loc) · 721 Bytes
1 package edu.neu.csye7374.designpattern.observer;
2
3 import org.springframework.http.HttpStatus;
4 import org.springframework.web.server.ResponseStatusException;
5
6 import com.inventory.model.Product;
7 import com.inventory.repository.ProductRepository;
8
9 public class UpdateDB extends ObserverAPI{
10
11     private ProductRepository productRepo;
12
13    public UpdateDB(Notify notify, ProductRepository productRepo) {
14        this.notify = notify;
15        this.productRepo = productRepo;
16        this.notify.attach(this);
17    }
18
19    @Override
20    public void update(Product product) {
21        if(productRepo.productExists(product.getProductName()))
22    }
```

### 3) State Design Pattern

The top screenshot shows the IDE interface with the file explorer on the left and the code editor on the right. The file explorer shows the project structure: `final-project-group_8 / src / main / java / edu / neu / csye7374 / designpattern / state / State.java`. The code editor displays the `State.java` file, which contains the following code:

```
1 package edu.neu.csye7374.designpattern.state;
2
3 public class State {
4
5     private StateAPI state;
6
7     public State() {
8         state = null;
9     }
10
11     public StateAPI getState() {
12         return state;
13     }
14
15     public void setState(StateAPI state) {
16         this.state = state;
17     }
18
19 }
20
```

The bottom screenshot shows the IDE interface with the file explorer on the left and the code editor on the right. The file explorer shows the project structure: `final-project-group_8 / src / main / java / edu / neu / csye7374 / designpattern / state / StateAPI.java`. The code editor displays the `StateAPI.java` file, which contains the following code:

```
1 package edu.neu.csye7374.designpattern.state;
2
3 import edu.neu.csye7374.designpattern.state.State;
4 public abstract class StateAPI {
5
6     /**
7      * Performs the action of sending an alert to the server and updating
8      * @param state
9      */
10     public abstract void action(State state, int stock);
11
12 }
```



Files

main

Go to file

designpattern

command

Client.java

CommandAPI.java

Communication.java

Server.java

ServerClient.java

state

State.java

StateAPI.java

StockAlert.java

StockUpdate.java

strategy

model

repository

final-project-group\_8 / src / main / java / edu / neu / csye7374 / designpattern / state / StockAlert.java

OmkarVaity State Design Pattern and Product 5649

Code Blame 20 lines (15 loc) · 535 Bytes

```
1 package edu.neu.csye7374.designpattern.state;
2
3 public class StockAlert extends StateAPI{
4
5     Product product;
6     ProductRepository productRepo;
7
8     public StockAlert(Product product, ProductRepository productRepo) {
9         this.product = product;
10        this.productRepo = productRepo;
11    }
12
13    @Override
14    public void action(State state,int stock) {
15        SendMessage.message("\n*****\nLOW STOCK for "+product.getProductName()+"\n*****\n");
16        product.setQuantity(stock);
17        productRepo.update(product);
18    }
19
20 }
```

Files

main

Go to file

designpattern

command

Client.java

CommandAPI.java

Communication.java

Server.java

ServerClient.java

state

State.java

StateAPI.java

StockAlert.java

StockUpdate.java

strategy

model

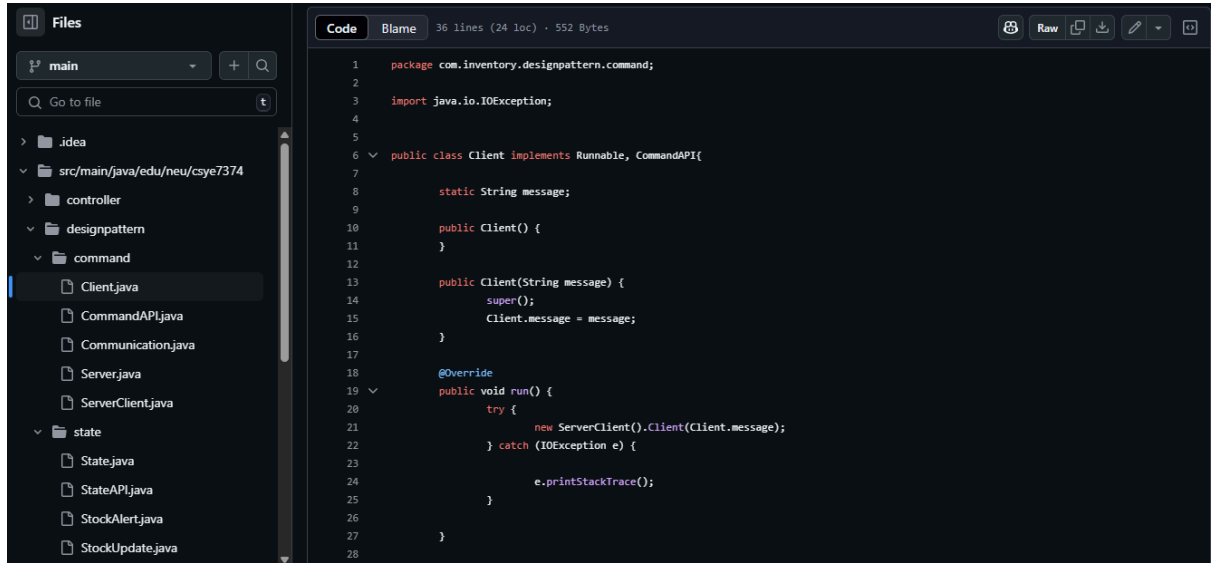
final-project-group\_8 / src / main / java / edu / neu / csye7374 / designpattern / state / StockUpdate.java

OmkarVaity State Design Pattern and Product 5649

Code Blame 20 lines (14 loc) · 445 Bytes

```
1 package edu.neu.csye7374.designpattern.state;
2
3 public class StockUpdate extends StateAPI{
4
5     Product product;
6     ProductRepository productRepo;
7
8     public StockUpdate(Product product, ProductRepository productRepo) {
9         this.product = product;
10        this.productRepo = productRepo;
11    }
12
13    @Override
14    public void action(State state, int stock) {
15
16        product.setQuantity(stock);
17        productRepo.update(product);
18    }
19
20 }
```

#### 4) Command Design Pattern



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows the project structure: `main` (selected), `idea`, `src/main/java/edu/neu/csye7374`, `controller`, `designpattern`, `command` (expanded), `Client.java` (selected), `CommandAPI.java`, `Communication.java`, `Server.java`, `ServerClient.java`, `state`, `State.java`, `StateAPI.java`, `StockAlert.java`, and `StockUpdate.java`. The code editor displays the `Client.java` file, which implements the `Runnable` and `CommandAPI` interfaces. The code is as follows:

```
1 package com.inventory.designpattern.command;
2
3 import java.io.IOException;
4
5
6 public class Client implements Runnable, CommandAPI{
7
8     static String message;
9
10    public Client() {
11    }
12
13    public Client(String message) {
14        super();
15        Client.message = message;
16    }
17
18    @Override
19    public void run() {
20        try {
21            new ServerClient().Client(Client.message);
22        } catch (IOException e) {
23
24            e.printStackTrace();
25        }
26    }
27
28 }
```



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows the project structure: `main` (selected), `idea`, `src/main/java/edu/neu/csye7374`, `controller`, `designpattern`, `command` (expanded), `Client.java`, `CommandAPI.java` (selected), `Communication.java`, `Server.java`, `ServerClient.java`, `state`, and `State.java`. The code editor displays the `CommandAPI.java` file, which defines the `CommandAPI` interface. The code is as follows:

```
1 package com.inventory.designpattern.command;
2
3 public interface CommandAPI {
4     void execute();
5 }
```

Files

main

Go to file

- .idea
- src/main/java/edu/neu/csye7374
  - controller
  - designpattern
    - command
      - Client.java
      - CommandAPI.java
      - Communication.java
      - Server.java
      - ServerClient.java
    - state
      - State.java
      - StateAPI.java
      - StockAlert.java
      - StockUpdate.java

Code Blame 35 lines (23 loc) · 686 Bytes

```
1 package com.inventory.designpattern.command;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Communication {
7
8
9     private List<CommandAPI> orderList = new ArrayList<CommandAPI>();
10
11     public void takeOperation(CommandAPI op) {
12         orderList.add(op);
13     }
14
15     public void executeOperation() {
16         for (CommandAPI op : orderList) {
17             op.execute();
18         }
19
20         orderList.clear();
21     }
22
23     public void triggerServerClient(String msg) {
24         Server s = new Server();
25         Client c = new Client(msg);
26
27         Communication com = new Communication();
28         com.takeOperation(s);
29         com.takeOperation(c);
30     }
31 }
```

Files

main

Go to file

- .idea
- src/main/java/edu/neu/csye7374
  - controller
  - designpattern
    - command
      - Client.java
      - CommandAPI.java
      - Communication.java
      - Server.java
      - ServerClient.java
    - state
      - State.java
      - StateAPI.java
      - StockAlert.java
      - StockUpdate.java

final-project-group\_8 / src / main / java / edu / neu / csye7374 / designpattern / command / Server.java

YashZ2002 Add files via upload

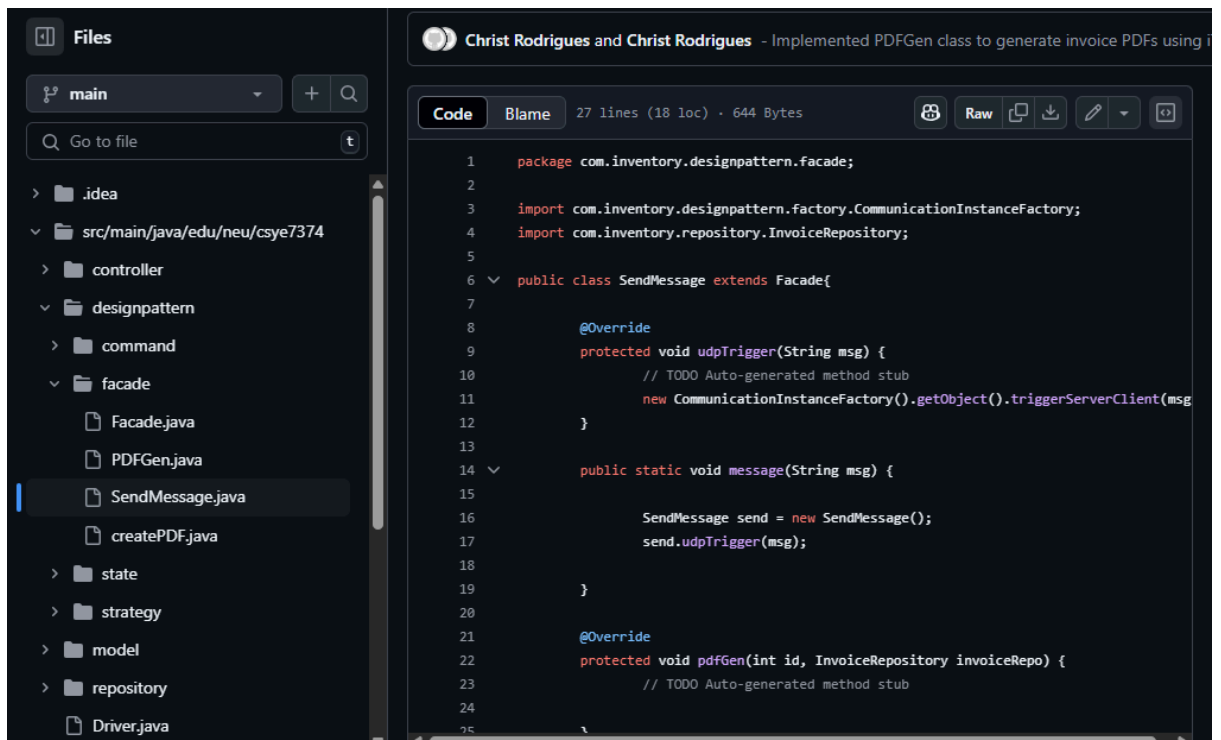
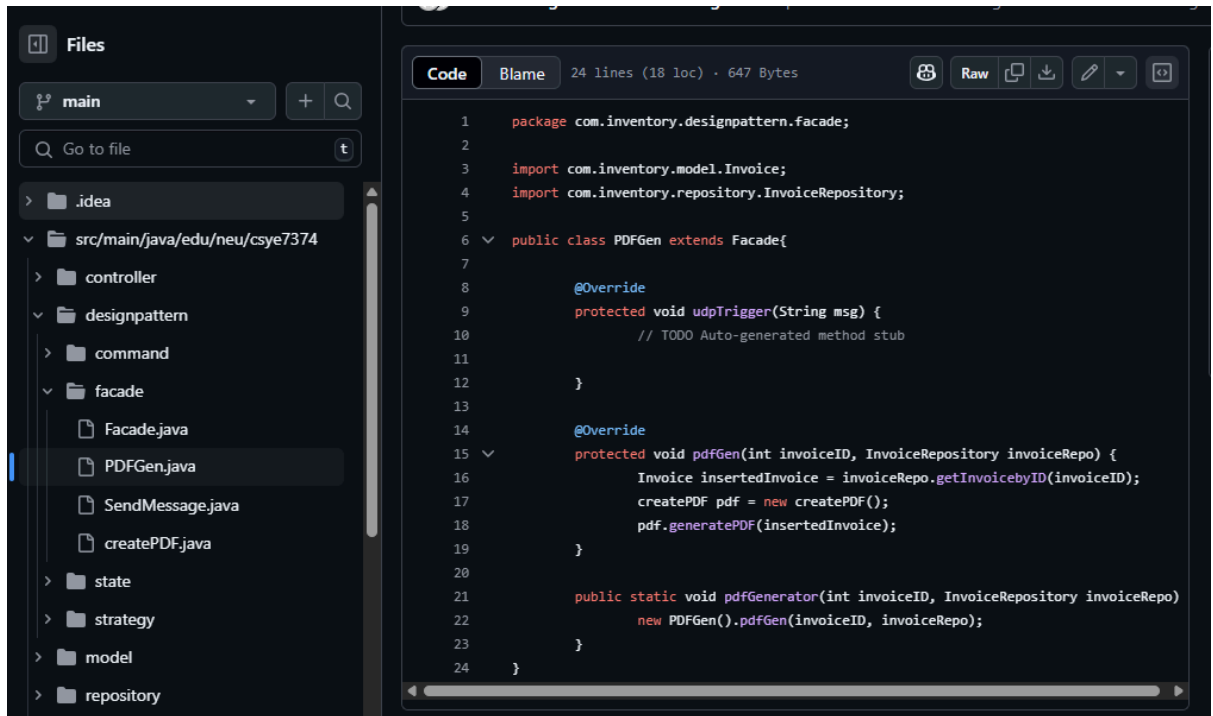
Code Blame 25 lines (18 loc) · 428 Bytes

```
1 package com.inventory.designpattern.command;
2
3 import java.io.IOException;
4
5 public class Server implements Runnable, CommandAPI{
6
7     @Override
8     public void run() {
9         try {
10             new ServerClient().Server();
11         } catch (IOException e) {
12             System.err.println("Error in Server");
13             e.printStackTrace();
14         }
15     }
16
17     @Override
18     public void execute() {
19         Thread t = new Thread(new Server());
20         t.start();
21     }
22
23 }
```

```
1 package com.inventory.designpattern.command;
2
3 import java.io.IOException;
4 import java.net.DatagramPacket;
5 import java.net.DatagramSocket;
6 import java.net.InetAddress;
7 import java.util.Scanner;
8
9 public class ServerClient {
10
11     int port = 1234;
12
13     public ServerClient() {
14
15     }
16
17     public void Server() throws IOException{
18         DatagramSocket ds = new DatagramSocket(port);
19         byte[] recieve = new byte[10000];
20
21         DatagramPacket dpPacket = null;
22         System.out.println("***** Server Listening on Port *****");
23         while(true) {
24             dpPacket = new DatagramPacket(recieve, recieve.length);
25
26             ds.receive(dpPacket);
27
28         }
29     }
30 }
```

## 5) Facade Design Pattern

```
1 package com.inventory.designpattern.facade;
2
3 import com.inventory.repository.InvoiceRepository;
4
5 public abstract class Facade {
6
7     protected abstract void udpTrigger(String msg);
8
9     protected abstract void pdfGen(int id, InvoiceRepository invoiceRepo);
10
11 }
```



```
final-project-group_8 / src / main / java / edu / neu / csye7374 / designpattern / facade
/ createPDF.java

Code Blame 81 lines (69 loc) · 2.56 KB

7 import java.util.stream.Stream;
8
9 import com.inventory.model.Invoice;
10 import com.inventory.model.ProductPO;
11 import com.itextpdf.text.*;
12 import com.itextpdf.text.pdf.*;
13
14 public class createPDF {
15
16     private Invoice invoice;
17
18     public void generatePDF(Invoice invoice) {
19         this.invoice = invoice;
20         Document document = new Document();
21         try {
22             String filename = "Invoice_" + invoice.getId() + "_" + invoice
23             PdfWriter.getInstance(document, new FileOutputStream(filename)
24         } catch (FileNotFoundException e) {
25             // TODO Auto-generated catch block
26             e.printStackTrace();
27         } catch (DocumentException e) {
28             // TODO Auto-generated catch block
29             e.printStackTrace();
30         }
31
32         document.open();
```

```
Files

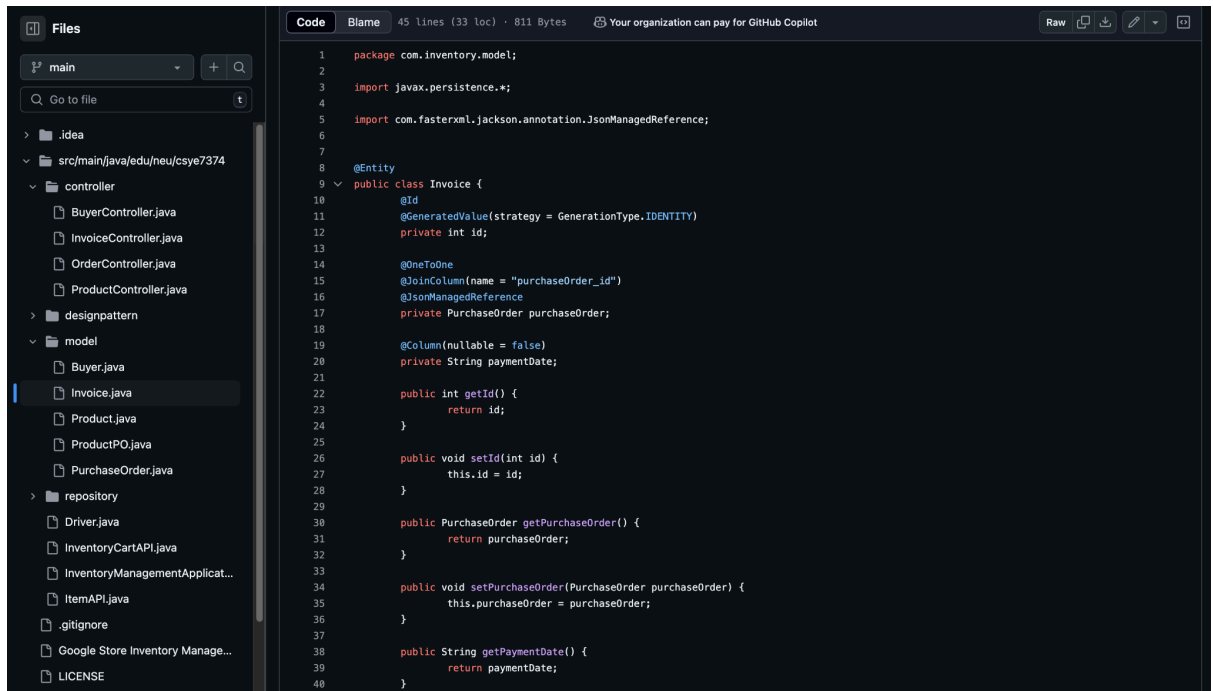
main
Go to file

src/main/java/edu/neu/csye7374
controller
designpattern
model
repository
Driver.java
InventoryCartAPI.java
InventoryManagementApplicat...
ItemAPI.java
.gitignore
Google Store Inventory Manage...
LICENSE

model
Buyer.java
Invoice.java
Product.java
ProductPO.java
PurchaseOrder.java

Code Blame 62 lines (46 loc) · 1.17 KB Your organization can pay for GitHub Copilot

1 package com.inventory.model;
2
3 import javax.persistence.*;
4
5 import com.fasterxml.jackson.annotation.JsonBackReference;
6
7 @Entity
8 public class ProductPO {
9     @Id
10     @GeneratedValue(strategy = GenerationType.IDENTITY)
11     private int id;
12
13     @ManyToOne(cascade = CascadeType.ALL)
14     @JoinColumn(name = "product_id")
15     @JsonBackReference(value="product")
16     private Product product;
17
18     @ManyToOne(cascade = CascadeType.ALL)
19     @JoinColumn(name = "purchaseOrder_id")
20     @JsonBackReference
21     private PurchaseOrder purchaseOrder;
22
23     @Column(nullable = false)
24     private int quantity;
25
26     public int getId() {
27         return id;
28     }
29
30     public void setId(int id) {
31         this.id = id;
32     }
33
34     public Product getProduct() {
35         return product;
36     }
37
38     public void setProduct(Product product) {
```



```
1 package com.inventory.model;
2
3 import javax.persistence.*;
4
5 import com.fasterxml.jackson.annotation.JsonManagedReference;
6
7
8 @Entity
9 public class Invoice {
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private int id;
13
14     @OneToOne
15     @JoinColumn(name = "purchaseOrder_id")
16     @JsonManagedReference
17     private PurchaseOrder purchaseOrder;
18
19     @Column(nullable = false)
20     private String paymentDate;
21
22     public int getId() {
23         return id;
24     }
25
26     public void setId(int id) {
27         this.id = id;
28     }
29
30     public PurchaseOrder getPurchaseOrder() {
31         return purchaseOrder;
32     }
33
34     public void setPurchaseOrder(PurchaseOrder purchaseOrder) {
35         this.purchaseOrder = purchaseOrder;
36     }
37
38     public String getPaymentDate() {
39         return paymentDate;
40     }
41 }
```

## 6) Factory Design Pattern:

Files

main

Go to file

Front-End

- src/main
  - java/edu/neu/csye7374
    - config
    - controller
    - designpattern
      - command
      - decorator
      - facade

final-project-group\_8 / src / main / java / edu / neu / csye7374 / designpattern / factory / AbstractFactoryAPI.java

OmkarVaity Updated logic for backend 47a1706 · 2 days ago History

Code Blame 18 lines (8 loc) · 243 Bytes

```
1 package edu.neu.csye7374.designpattern.factory;
2
3 import edu.neu.csye7374.designpattern.command.Communication;
4
5 public abstract class AbstractFactoryAPI {
6     /**
7      * Returns an object
8      */
9     public abstract Communication getObject();
10 }
```

final-project-group\_8 / src / main / java / edu / neu / csye7374 / designpattern / factory / CommunicationInstanceFactory.java

OmkarVaity Updated logic for backend 47a1706 · 2 days ago History

Code Blame 18 lines (13 loc) · 419 Bytes

```
1 package edu.neu.csye7374.designpattern.factory;
2
3 import edu.neu.csye7374.designpattern.command.Communication;
4
5 // Implementing Lazy Factory Pattern
6 public class CommunicationInstanceFactory extends AbstractFactoryAPI{
7
8     private static Communication comm;
9     @Override
10    public Communication getObject() {
11
12        if(comm == null) {
13            comm = new Communication();
14        }
15        return comm;
16    }
17
18 }
```



## 7) Decorator Design Pattern:

```
final-project-group_8 / src / main / java / edu / neu / csye7374 / designpattern / decorator / CartDecorator.java

OmkarVaity Updated logic for backend 47a1706 · 2 days ago History

Code Blame 18 lines (12 loc) · 351 Bytes

1 package edu.neu.csye7374.designpattern.decorator;
2
3 import edu.neu.csye7374.InventoryCartAPI;
4
5 public class CartDecorator implements InventoryCartAPI{
6
7     InventoryCartAPI cart;
8
9     public CartDecorator(InventoryCartAPI cart) {
10         this.cart = cart;
11     }
12
13     @Override
14     public double getCost() {
15         return this.cart.getCost();
16     }
17
18 }
```

```
final-project-group_8 / src / main / java / edu / neu / csye7374 / designpattern / decorator / CustomDecorator.java

OmkarVaity Updated logic for backend 47a1706 · 2 days ago History

Code Blame 21 lines (16 loc) · 563 Bytes

1 package edu.neu.csye7374.designpattern.decorator;
2
3 import edu.neu.csye7374.InventoryCartAPI;
4 import edu.neu.csye7374.model.Product;
5 import edu.neu.csye7374.model.ProductPO;
6
7 public class CustomDecorator extends CartDecorator {
8
9     int quantity;
10    double price;
11
12    public CustomDecorator(InventoryCartAPI cart, Product product, ProductPO proPo) {
13        super(cart);
14        this.quantity = proPo.getQuantity();
15        this.price = product.getPrice();
16    }
17
18    public double getCost() {
19        return super.getCost()+(this.quantity * this.price);
20    }
21 }
```

```
final-project-group_8 / src / main / java / edu / neu / csye7374 / designpattern / decorator / Product.java

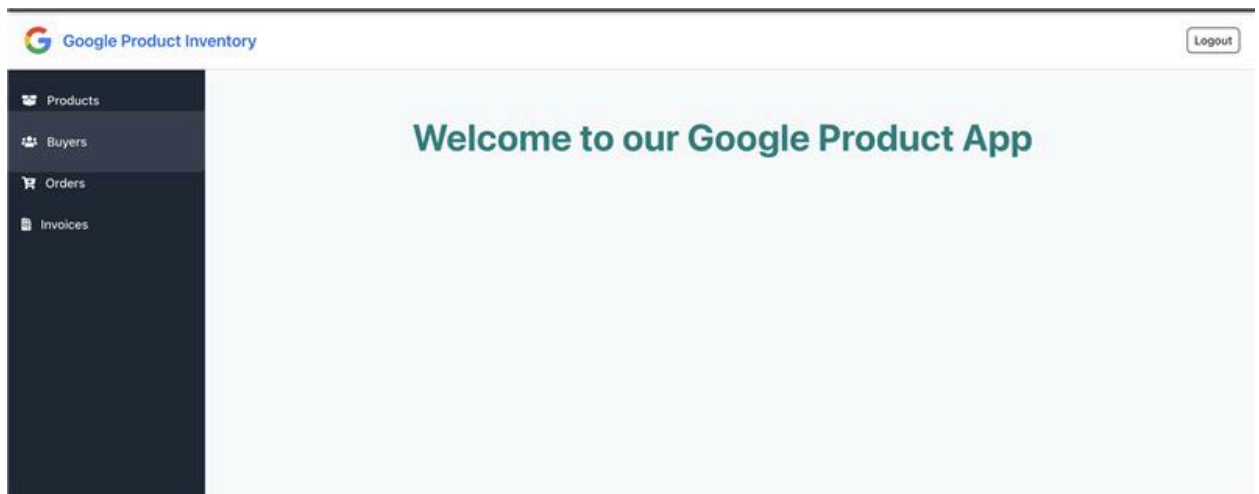
OmkarVaity Updated logic for backend 47a1706 · 2 days ago History

Code Blame 19 lines (13 loc) · 352 Bytes

1 package edu.neu.csye7374.designpattern.decorator;
2
3 import edu.neu.csye7374.InventoryCartAPI;
4
5 public class Product implements InventoryCartAPI{
6
7     double totalCost;
8
9     public Product() {
10         this.totalCost = 0;
11     }
12
13     @Override
14     public double getCost() {
15         // TODO Auto-generated method stub
16         return this.totalCost;
17     }
18
19 }
```

Screenshot of UI (feature-based) at every stage of the application:

### 1. Welcome Page:



## 2. Product Page:

The screenshot shows the 'Products' page of the Google Product Inventory application. On the left is a dark sidebar with navigation links: Products, Buyers, Orders, and Invoices. The main content area has a header with the Google logo and 'Google Product Inventory' text, and a 'Logout' button in the top right. Below the header, the title 'Products' is displayed next to a '+ Add Product' button. A table lists the products with columns: Sr. No., Product Name, Price, Quantity, and Action. The first row shows a product with Sr. No. 1, Product Name 'Google Pixel 9 Pro', Price 750, and Quantity 10. The Action column for this row contains 'Edit' and 'Delete' buttons.

| Sr. No. | Product Name       | Price | Quantity | Action  |
|---------|--------------------|-------|----------|---|
| 1       | Google Pixel 9 Pro | 750   | 10       | <button>Edit</button> <button>Delete</button> |

## 3. Add Product:

The screenshot shows the 'Add Product' form in the Google Product Inventory application. The layout is consistent with the previous page, featuring the same sidebar and header. The main content area is titled 'Add Product' and contains a form with three input fields: 'Product Name' (with placeholder text 'Enter product name'), 'Quantity' (with value 0), and 'Price' (with value 0). A blue 'Save Product' button is located at the bottom of the form.

**Add Product**

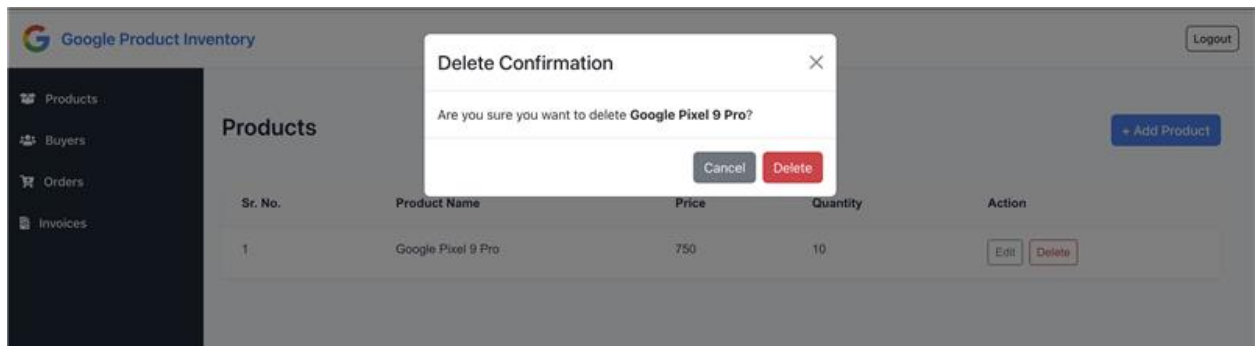
Product Name  
Enter product name

Quantity  
0

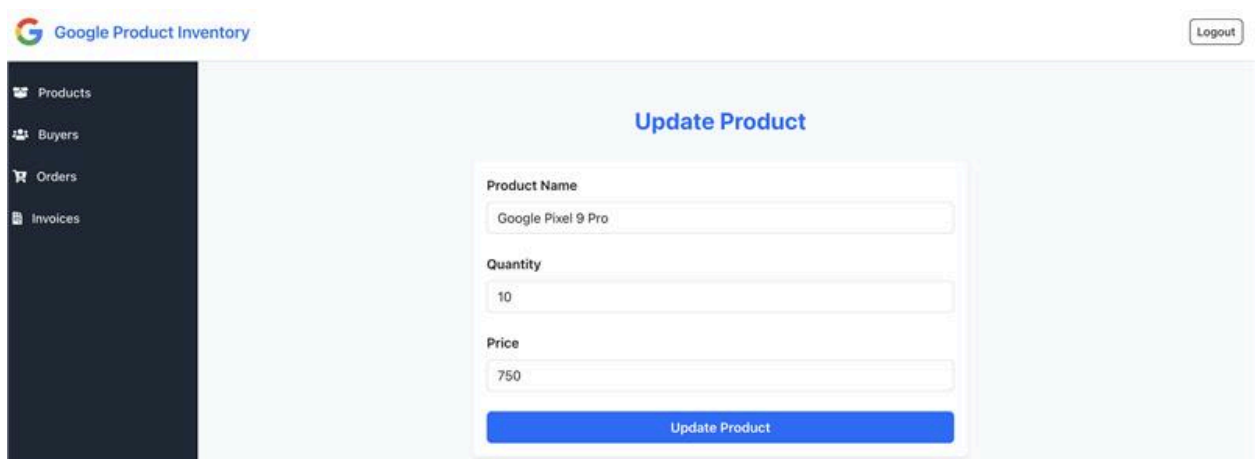
Price  
0

Save Product

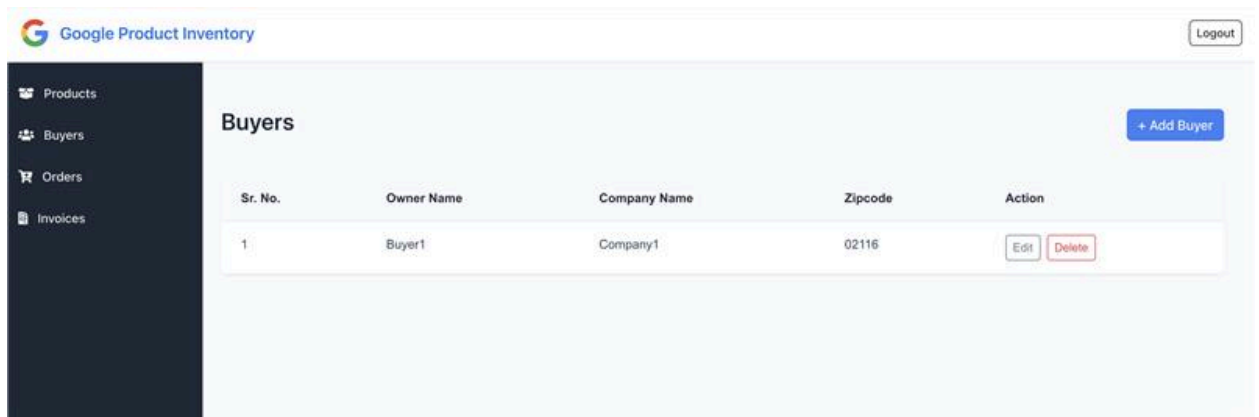
#### 4. Delete Product:



#### 5. Edit Product:



#### 6. Buyer Page:



## 7. Add Buyer:

The screenshot shows the 'Add Buyer' form in the Google Product Inventory application. The form is titled 'Add Buyer' and contains three input fields: 'Owner Name' (placeholder: 'Enter buyer name'), 'Company Name' (placeholder: 'Enter company name'), and 'Zipcode' (placeholder: 'Enter zipcode'). A blue 'Save Buyer' button is at the bottom of the form. The application header includes the Google logo, 'Google Product Inventory', and a 'Logout' button. The left sidebar contains links for 'Products', 'Buyers', 'Orders', and 'Invoices'.

## 8. Update Buyer:

The screenshot shows the 'Update Buyer' form in the Google Product Inventory application. The form is titled 'Update Buyer' and contains three input fields: 'Owner Name' (value: 'Buyer1'), 'Company Name' (value: 'Company1'), and 'Zipcode' (value: '02116'). A blue 'Update Buyer' button is at the bottom of the form. The application header includes the Google logo, 'Google Product Inventory', and a 'Logout' button. The left sidebar contains links for 'Products', 'Buyers', 'Orders', and 'Invoices'.

## 9. Delete Buyer:

The screenshot shows the 'Delete Buyer' confirmation dialog in the Google Product Inventory application. The dialog is titled 'Delete Confirmation' and contains the text 'Are you sure you want to delete Buyer1?'. There are 'Cancel' and 'Delete' buttons at the bottom of the dialog. The background shows the 'Buyers' table with one row: '1', 'Buyer1', 'Company1', '02116'. The 'Action' column has 'Edit' and 'Delete' buttons. The application header includes the Google logo, 'Google Product Inventory', and a 'Logout' button. The left sidebar contains links for 'Products', 'Buyers', 'Orders', and 'Invoices'.

| Sr. No. | Owner Name | Company Name | Zipcode | Action  |
|---------|------------|--------------|---------|---|
| 1       | Buyer1     | Company1     | 02116   | <button>Edit</button> <button>Delete</button> |

## 10. Order Page:

Google Product Inventory

Logout

Products

Buyers

Orders

Invoices

Purchase Orders

+ Add Order

| Sr. No. | Company Name | Total Products | Total Price | Payment Date | Status | Action                            |
|---------|--------------|----------------|-------------|--------------|--------|-----------------------------------|
| 1       | Company1     | 2              | 750         | 04-21-2025   | Unpaid | <div>Generate InvoiceDelete</div> |

## 11. Add Order:

Google Product Inventory

Logout

Products

Buyers

Orders

Invoices

Add Purchase Order

Select Buyer

Company1 - Buyer1

Payment Date

04/21/2025

Select Product

Google Pixel 9 Pro

Add

| #     | Product Name       | Price | Available Quantity | Order Quantity | Action            |
|-------|--------------------|-------|--------------------|----------------|-------------------|
| 1     | Google Pixel 9 Pro | 750   | 10                 | <div>1</div>   | <div>Delete</div> |
| Total |                    | 750   |                    | <div>1</div>   |                   |

Submit Order

## 12. Delete Order:

Google Product Inventory

Logout

Products

Buyers

Orders

Invoices

Add Purchase Order

Select Buyer  
Company1 - Buyer1

Payment Date  
04/21/2025

Select Product  
Google Pixel 9 Pro

Add

| #     | Product Name       | Price | Available Quantity | Order Quantity | Action |
|-------|--------------------|-------|--------------------|----------------|--------|
| 1     | Google Pixel 9 Pro | 750   | 10                 | 1              | Delete |
| Total |                    | 750   |                    | 1              |        |

Submit Order

## 13. Generate Invoice:

Google Product Inventory

Logout

Products

Buyers

Orders

Invoices

Purchase Orders

+ Add Order

Payment Confirmation

You are about to generate the invoice and mark the order as paid. Continue?

CancelYes, Generate

| Sr. No. | Company Name | Product | Price | Date       | Status | Action                 |
|---------|--------------|---------|-------|------------|--------|------------------------|
| 1       | Company1     | 2       | 750   | 04-21-2025 | Unpaid | Generate InvoiceDelete |

## 14. Invoice Page:

Google Product Inventory

Logout

Products

Buyers

Orders

Invoices

Buyers Invoices

| Sr. No. | Company Name | Total Products | Total Quantity | Total Price | Payment Date |
|---------|--------------|----------------|----------------|-------------|--------------|
| 1       | Company1     | 2              | 2              | 750         | 04-21-2025   |

**Contributions:**

- **Christ Rodrigues:** Facade Design Pattern, Frontend.
- **Yash Zaveri:** Command Design Pattern, Frontend.
- **Viral Suchak:** Strategy Design Pattern, Factory Design Pattern.
- **Kevin Rodrigues:** Observer Design Pattern, Frontend.
- **Omkar Vaity:** State Design Pattern, Singleton Design Pattern.