Inventory Management System  
  
Program:  
  
package ims\_package;

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.***in***);

Map<String, String> users = new HashMap<>();

users.put("admin", "1234");

System.***out***.print("Username: ");

String user = sc.nextLine();

System.***out***.print("Password: ");

String pass = sc.nextLine();

if (users.containsKey(user) && users.get(user).equals(pass)) {

System.***out***.println("Login successful.");

Map<Integer, String> productNames = new HashMap<>();

Map<Integer, Integer> productQuantities = new HashMap<>();

Map<Integer, Double> productPrices = new HashMap<>();

Map<Integer, Double> productCostPrices = new HashMap<>();

Map<Integer, Integer> productSales = new HashMap<>();

int id = *loadProducts*(productNames, productQuantities, productCostPrices, productPrices);

boolean work=true;

while(work) {

System.***out***.println("1.Product management\n2.Billing \n3.Investment management\n4.Exit ");

int workcase=sc.nextInt();

switch(workcase){

case 1:

System.***out***.println("Hello user !");

boolean on=true;

while(on) {

System.***out***.println("1.Add product\n2.Update product\n3.View product\n4.Delete product \n5.exit");

int operation=sc.nextInt();

switch(operation) {

case 1:

sc.nextLine();

System.***out***.println("Product Name: ");

String name = sc.nextLine();

System.***out***.println("Quantity: ");

int qty = sc.nextInt();

System.***out***.println("Cost Price: ");

double costPrice = sc.nextDouble();

productCostPrices.put(id, costPrice);

System.***out***.print("Selling Price: ");

double price = sc.nextDouble();

productNames.put(id, name);

productQuantities.put(id, qty);

productPrices.put(id, price);

System.***out***.println("Product added with ID: " + id);

*saveProducts*(productNames, productQuantities, productCostPrices, productPrices);

id++;

break;

case 2:

System.***out***.print("Enter Product ID to Update: ");

int pid = sc.nextInt();

if (productNames.containsKey(pid)) {

System.***out***.print("New Quantity: ");

productQuantities.put(pid, sc.nextInt());

System.***out***.print("New Price: ");

productPrices.put(pid, sc.nextDouble());

System.***out***.println("Product updated.");

*saveProducts*(productNames, productQuantities, productCostPrices, productPrices);

} else {

System.***out***.println("Product not found.");

}

break;

case 3:

System.***out***.println("ID | Name | Qty | Price");

for (int prid: productNames.keySet()) {

System.***out***.println(prid + " | " + productNames.get(prid) + " | " +

productQuantities.get(prid) + " | ₹" + productPrices.get(prid));

}

break;

case 4:

System.***out***.print("Enter Product ID to Delete: ");

int dpid = sc.nextInt();

if (productNames.containsKey(dpid)) {

productNames.remove(dpid);

productQuantities.remove(dpid);

productPrices.remove(dpid);

System.***out***.println("Product deleted.");

*saveProducts*(productNames, productQuantities, productCostPrices, productPrices);

} else {

System.***out***.println("Product not found.");

}

break;

case 5:

System.***out***.println("Exitting ...");

on=false;

break;

default:

System.***out***.println("Invalid option");

}}

break;

case 2:

double total = 0;

// Track sold quantities

while(true) {

System.***out***.print("Enter Product ID to buy (0 to exit): ");

int bpid = sc.nextInt();

if(bpid == 0) break;

if(productNames.containsKey(bpid)) {

System.***out***.print("Enter quantity: ");

int qty = sc.nextInt();

int stock = productQuantities.get(bpid);

if(qty <= stock) {

double price = productPrices.get(bpid);

double cost = price \* qty;

total += cost;

productQuantities.put(bpid, stock - qty);

*saveProducts*(productNames, productQuantities, productCostPrices, productPrices);

productSales.put(bpid, productSales.getOrDefault(bpid, 0) + qty);

System.***out***.println(productNames.get(bpid) + " x" + qty + " = ₹" + cost);

} else {

System.***out***.println("Only " + stock + " available.");

}

} else {

System.***out***.println("Product not found.");

}

}

System.***out***.println("Total Bill = ₹" + total);

break;

case 3:

System.***out***.println("Investment Management & Sales Report:");

double totalInvestment = 0;

double totalRevenue = 0;

double totalProfit = 0;

if(productSales == null || productSales.isEmpty()) {

System.***out***.println("No sales data available.");

} else {

for (Map.Entry<Integer, Integer> entry : productSales.entrySet()) {

int pid = entry.getKey();

int soldQty = entry.getValue();

double costPrice = productCostPrices.getOrDefault(pid, 0.0);

double sellingPrice = productPrices.getOrDefault(pid, 0.0);

totalInvestment += costPrice \* soldQty;

totalRevenue += sellingPrice \* soldQty;

}

totalProfit = totalRevenue - totalInvestment;

double profitPercent = (totalInvestment > 0) ? (totalProfit / totalInvestment) \* 100 : 0;

System.***out***.printf("Total Investment: ₹%.2f\n", totalInvestment);

System.***out***.printf("Total Revenue: ₹%.2f\n", totalRevenue);

System.***out***.printf("Total Profit: ₹%.2f (%.2f%%)\n", totalProfit, profitPercent);

System.***out***.println("\nTop 5 Best Selling Products:");

System.***out***.println("Rank | Product Name | Qty Sold | Price per unit");

List<Map.Entry<Integer, Integer>> salesList = new ArrayList<>(productSales.entrySet());

salesList.sort((a, b) -> b.getValue().compareTo(a.getValue()));

int rank = 1;

for(Map.Entry<Integer, Integer> sale : salesList) {

if(rank > 5) break;

int pid = sale.getKey();

int qtySold = sale.getValue();

String pname = productNames.get(pid);

double price = productPrices.get(pid);

System.***out***.printf("%d | %s | %d | ₹%.2f\n", rank, pname, qtySold, price);

rank++;

}

}

break;

case 4:

System.***out***.println("Good job >>>Have a Nice Job");

default:

System.***out***.println("Invalid selection.");

} } }else {

System.***out***.println("Login failed.");

}

}

public static void saveProducts(Map<Integer, String> productNames,

Map<Integer, Integer> productQuantities,

Map<Integer, Double> productCostPrices,

Map<Integer, Double> productPrices) {

try {

FileWriter fw = new FileWriter("products.txt");

for (Integer id : productNames.keySet()) {

fw.write(id + "," +

productNames.get(id) + "," +

productQuantities.get(id) + "," +

productCostPrices.get(id) + "," +

productPrices.get(id) + "\n");

}

fw.close();

} catch (IOException e) {

System.***out***.println("Error saving products: " + e.getMessage());

}

}

public static int loadProducts(Map<Integer, String> productNames,

Map<Integer, Integer> productQuantities,

Map<Integer, Double> productCostPrices,

Map<Integer, Double> productPrices) {

int maxId = 0;

try {

File file = new File("products.txt");

if (!file.exists()) return 1;

Scanner fileScanner = new Scanner(file);

while (fileScanner.hasNextLine()) {

String[] parts = fileScanner.nextLine().split(",");

int id = Integer.*parseInt*(parts[0]);

productNames.put(id, parts[1]);

productQuantities.put(id, Integer.*parseInt*(parts[2]));

productCostPrices.put(id, Double.*parseDouble*(parts[3]));

productPrices.put(id, Double.*parseDouble*(parts[4]));

if (id > maxId) maxId = id;

}

fileScanner.close();

} catch (Exception e) {

System.***out***.println("Error loading products: " + e.getMessage());

}

return maxId + 1;

}

}

