CAPSTONE PROJECT – REAL APPLICATION (COMPLETE CONCEPTS)

Project Planning and Requirements Gathering

Definethe project'sgoals,requiredinputs,expectedoutputs, libraries needed, and user interaction flow.

Code Organization and Structure

Organizecodeusingseparatemodulesforreadability and maintenance.

```
# main.py — main execution file
# utils.py — helper functions
# data_handler.py — loading and cleaning data
# analysis.py — calculations and charts
# report.py — report generation
```

Error Handling and User Experience

Usetry-exceptblockstoavoidcrashesandgive clear feedback to users.

```
try:
    df = pd.read_csv("data.csv")
except FileNotFoundError:
    print("Data file not found.")
except ValueError:
    print("Invalid data format.")
except Exception as e:
    print("Unexpected error:", e)
```

Documentation and Code Comments

Provideprojectoverview,filestructure,andexplain code functionality with comments.

```
# Calculate monthly revenue
monthly = df.groupby("Month")["Revenue"].sum()
```

Testing Basic Functionality

Testeachpartoftheprojectindividually before combining them.

```
assert len(df) > 0, "Dataset should not be empty"
assert "Revenue" in df.columns, "Revenue column missing"
```

Project Presentation and Demonstration

Presenttheproblem,features,workflow,outputs,charts, insights, and future improvements.

CAPSTONE PROJECT – HANDS-ON PRACTICE (CONTENT ONLY)

Plan a Complete Application from Start to Finish

Define the application's goals, required inputs, expected outputs, target users, workflow, and data sources. Break the application into modules and outline the main functionalities.

Implement the Application Using All Learned Concepts

Use functions, file handling, pandas, external libraries, data cleaning, visualization, and organized modules to build the full application. Ensure the structure is clean and scalable.

Add Proper Error Handling and User Feedback

Implement try-except blocks, validate inputs, and provide clear messages to the user. Prevent invalid operations and ensure the system does not crash unexpectedly.

Write Documentation for Your Code

Create project documentation that includes setup steps, how to run the application, explanation of modules, and sample input/output. Add comments inside the code to explain logic where necessary.

Test Your Application Thoroughly

Perform unit tests and manual scenario tests. Test data loading, user flows, edge cases, calculations, and reporting functionality. Ensure all components work together smoothly.

Prepare a Demonstration of Your Project

Show the complete workflow of your application, including features, outputs, reports, user flow, and project insights. Present improvements, limitations, and future enhancement ideas.

Project: Inventory Management System

Develop a complete Inventory Management System for a small business with modules for product tracking, stock updates, sales recording, and report generation. Include business insights such as top products, stock alerts, and monthly revenue summaries.

WEEK 8 - CAPSTONE PROJECT

```python
# inventory_system.py
"""
Inventory Management System (simple CLI)
Files used:
 - products.csv  -> columns: product_id,name,price,stock
 - sales.csv     -> columns: sale_id,product_id,quantity,unit_price,total,datetime
"""

import csv
import os
from datetime import datetime

PRODUCTS_FILE = "products.csv"
SALES_FILE = "sales.csv"

# ---------- Helper IO functions ----------
def ensure_files():
    # Create files with headers if they don't exist
    if not os.path.exists(PRODUCTS_FILE):
        with open(PRODUCTS_FILE, "w", newline="") as f:
            writer = csv.writer(f)
            writer.writerow(["product_id", "name", "price", "stock"])
    if not os.path.exists(SALES_FILE):
        with open(SALES_FILE, "w", newline="") as f:
            writer = csv.writer(f)
            writer.writerow(["sale_id", "product_id", "quantity", "unit_price", "total", "datetime"])

def read_products():
    products = {}
    try:
        with open(PRODUCTS_FILE, "r", newline="") as f:
            reader = csv.DictReader(f)
            for row in reader:
                pid = row["product_id"]
                products[pid] = {
                    "name": row["name"],
                    "price": float(row["price"]),
                    "stock": int(row["stock"])
                }
```

```
        except FileNotFoundError:
            ensure_files()
        return products

    def write_products(products):
        with open(PRODUCTS_FILE, "w", newline="") as f:
            writer = csv.writer(f)
            writer.writerow(["product_id", "name", "price", "stock"])
            for pid, p in products.items():
                writer.writerow([pid, p["name"], f"{p['price']:.2f}", p["stock"]])

    def append_sale(sale_record):
        with open(SALES_FILE, "a", newline="") as f:
            writer = csv.writer(f)
            writer.writerow([
                sale_record["sale_id"],
                sale_record["product_id"],
                sale_record["quantity"],
                f"{sale_record['unit_price']:.2f}",
                f"{sale_record['total']:.2f}",
                sale_record["datetime"]
            ])

    # ---------- Core operations ----------
    def add_product(products):
        pid = input("Product ID: ").strip()
        if pid in products:
            print("Product ID already exists.")
            return
        name = input("Name: ").strip()
        try:
            price = float(input("Price: ").strip())
            stock = int(input("Initial stock: ").strip())
        except ValueError:
            print("Invalid numeric input. Aborting add.")
            return
        products[pid] = {"name": name, "price": price, "stock": stock}
        write_products(products)
        print("Product added.")

    def update_stock(products):
        pid = input("Product ID: ").strip()
        if pid not in products:
            print("Product not found.")
            return
```

```
    append_sale(sale_record)
    print(f"Sale recorded. Total: {total:.2f}")

def view_products(products):
    print("Product List:")
    for pid, p in products.items():
        print(f"{pid} | {p['name']} | Price: {p['price']:.2f} | Stock: {p['stock']}")

def generate_reports():
    # Basic reports: top products by revenue, stock alert, monthly revenue
    try:
        with open(SALES_FILE, "r", newline="") as f:
            reader = csv.DictReader(f)
            sales = list(reader)
    except FileNotFoundError:
        print("No sales data found.")
        return

    # Top products by revenue
    revenue_by_product = {}
    for s in sales:
        pid = s["product_id"]
        total = float(s["total"])
        revenue_by_product[pid] = revenue_by_product.get(pid, 0.0) + total

    top_products = sorted(revenue_by_product.items(), key=lambda x: x[1], reverse=True)
    print("Top products by revenue:")
    for pid, rev in top_products[:10]:
        print(f"{pid}: {rev:.2f}")

    # Monthly revenue
    revenue_by_month = {}
    for s in sales:
        dt = s["datetime"]
        month = dt[:7]  # YYYY-MM
        total = float(s["total"])
        revenue_by_month[month] = revenue_by_month.get(month, 0.0) + total

    print("\nMonthly revenue:")
    for month, rev in sorted(revenue_by_month.items()):
        print(f"{month}: {rev:.2f}")
```

```
    try:
        add_qty = int(input("Quantity to add (use negative to reduce): ").strip())
    except ValueError:
        print("Invalid quantity.")
        return
    products[pid]["stock"] += add_qty
    if products[pid]["stock"] < 0:
        print("Warning: stock became negative. Setting to 0.")
        products[pid]["stock"] = 0
    write_products(products)
    print("Stock updated.")

def record_sale(products):
    pid = input("Product ID: ").strip()
    if pid not in products:
        print("Product not found.")
        return
    try:
        qty = int(input("Quantity sold: ").strip())
    except ValueError:
        print("Invalid quantity.")
        return
    if qty <= 0:
        print("Quantity must be positive.")
        return
    if products[pid]["stock"] < qty:
        print(f"Insufficient stock. Available: {products[pid]['stock']}")
        return
    unit_price = products[pid]["price"]
    total = unit_price * qty
    # Reduce stock
    products[pid]["stock"] -= qty
    write_products(products)
    # Create sale record
    sale_id = f"S{int(datetime.now().timestamp())}"
    sale_record = {
        "sale_id": sale_id,
        "product_id": pid,
        "quantity": qty,
        "unit_price": unit_price,
        "total": total,
        "datetime": datetime.now().isoformat()
    }
```

```
    # Stock alerts
    products = read_products()
    alerts = [(pid, p["stock"]) for pid, p in products.items() if p["stock"] <= 5]
    if alerts:
        print("\nStock alerts (<=5):")
        for pid, stock in alerts:
            print(f"{pid}: {stock}")

# ---------- Main CLI ----------
def main():
    ensure_files()
    products = read_products()
    while True:
        print("\nInventory Management - Menu")
        print("1) Add product")
        print("2) Update stock")
        print("3) Record sale")
        print("4) View products")
        print("5) Generate reports")
        print("6) Exit")
        choice = input("Choose (1-6): ").strip()
        if choice == "1":
            add_product(products)
        elif choice == "2":
            update_stock(products)
        elif choice == "3":
            record_sale(products)
        elif choice == "4":
            view_products(products)
        elif choice == "5":
            generate_reports()
        elif choice == "6":
            print("Goodbye.")
            break
        else:
            print("Invalid choice.")

if __name__ == "__main__":
    main()
```

```
…  Inventory Management — Menu
   1) Add product
   2) Update stock
   3) Record sale
   4) View products
   5) Generate reports
   6) Exit
   Choose (1-6): 1
   Product ID: 1034
   Name: Socks
   Price: 300
   Initial stock: 10
   Product added.

   Inventory Management — Menu
   1) Add product
   2) Update stock
   3) Record sale
   4) View products
   5) Generate reports
   6) Exit
   Choose (1-6): 4
   Product List:
   P001 | Laptop | Price: 75000.00 | Stock: 10
   P002 | Smartphone | Price: 45000.00 | Stock: 25
   P003 | Headphones | Price: 2500.00 | Stock: 50
   P004 | Keyboard | Price: 1200.00 | Stock: 40
   P005 | Monitor | Price: 18000.00 | Stock: 15
   1034 | Socks | Price: 300.00 | Stock: 10

   Inventory Management — Menu
   1) Add product
   2) Update stock
   3) Record sale
   4) View products
   5) Generate reports
   6) Exit
   Choose (1-6): 5
   Top products by revenue:
   P001: 150000.00
   P002: 90000.00
   P005: 18000.00
   P003: 7500.00

   Monthly revenue:
   2025-01: 265500.00
```